

コマンド ライン ツール ユーザー ガイド

(旧名『開発システム リファレンス ガイド』)

UG628 (v12.2) 2010 年 7 月 23 日



Xilinx is disclosing this user guide, manual, release note, and/or specification (the “Documentation”) to you solely for use in the development of designs to operate with Xilinx hardware devices. You may not reproduce, distribute, republish, download, display, post, or transmit the Documentation in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Xilinx expressly disclaims any liability arising out of your use of the Documentation. Xilinx reserves the right, at its sole discretion, to change the Documentation without notice at any time. Xilinx assumes no obligation to correct any errors contained in the Documentation, or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information.

THE DOCUMENTATION IS DISCLOSED TO YOU “AS-IS” WITH NO WARRANTY OF ANY KIND. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DOCUMENTATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS. IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOSS OF DATA OR LOST PROFITS, ARISING FROM YOUR USE OF THE DOCUMENTATION.

© Copyright 2002–2010 Xilinx Inc. All Rights Reserved. XILINX, the Xilinx logo, the Brand Window and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners. The PowerPC name and logo are registered trademarks of IBM Corp., and used under license. All other trademarks are the property of their respective owners.

本資料は英語版 (v.12.2) を翻訳したもので、内容に相違が生じる場合には原文を優先します。
資料によっては英語版の更新に対応していないものがあります。
日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

目次

1: はじめに	17
コマンドラインプログラムの概要	17
コマンドラインの構文	18
コマンドライン オプション	18
-i (コマンド ファイルの実行)	19
-h (ヘルプ)	20
-intstyle (統合スタイル)	20
-p (製品番号)	21
コマンドラインプログラムの起動	22
2: デザイン フロー	23
デザイン フローの概要	23
デザイン入力および合成	26
階層デザイン	26
パーティション	27
PXML ファイル	27
回路図入力の概要	28
ライブラリ エlement	28
CORE Generator ツール (FPGA のみ)	29
HDL 入力および合成	29
論理シミュレーション	30
制約	30
マップ制約 (FPGA のみ)	30
ブロックの配置	30
タイミング仕様	30
ネットリスト変換プログラム	30
デザイン インプリメンテーション	31
マップ (FPGA のみ)	32
配置配線 (FPGA のみ)	33
ビットストリームの生成 (FPGA のみ)	33
デザイン検証	33
シミュレーション	35
バックアノテーション	35
NetGen	36
論理シミュレーション	37
タイミング シミュレーション	37
HDL ベースのシミュレーション	37
スタティック タイミング解析 (FPGAのみ)	39
インサーキット検証	39
デザイン ルール チェック (FPGA のみ)	39
プローブ	39

ChipScope ILA および ChipScope Pro.....	40
FPGA デザインのヒント	40
デザインのサイズとパフォーマンス.....	40
3 : PARTGen.....	41
PARTGen の概要	41
デバイス サポート.....	41
PARTGen の入力ファイル	41
PARTGen の出力ファイル	41
パーツリスト ファイル	41
パッケージ ファイル	46
PARTGen の構文	48
PARTGen のオプション	48
-arch (アーキテクチャの情報の表示).....	48
-i (デバイス、パッケージ、スピードの一覧の表示)	49
-intstyle (統合スタイル)	49
-nopkgfile (パッケージ ファイルを生成しない)	50
-p (パーツリスト ファイルとパッケージ ファイルの生成)	50
-v (詳細なパーツリスト ファイルとパッケージ ファイルの生成)	51
4 : NetGen	53
NetGen の概要	53
NetGen のフロー	53
NetGen のデバイス サポート.....	55
NetGen シミュレーション フロー	55
NetGen の論理シミュレーション フロー	55
NetGen のタイミングシミュレーション フロー	56
NetGen シミュレーション フローのオプション	58
Verilog 特定の論理およびタイミング シミュレーション オプション	62
VHDL 特定の論理およびタイミング シミュレーション オプション	65
NetGen の等価チェック フロー	66
NGDBuild 後のフロー (FPGA).....	66
インプリメンテーション後のフロー (FPGA).....	66
NetGen 等価チェックの入力ファイル.....	67
NetGen 等価チェックの出力ファイル.....	67
NetGen 等価チェックの構文	67
NetGen 等価チェック フローのオプション	68
NetGen のスタティック タイミング解析フロー.....	70
スタティック タイミング解析フロー (FPGA).....	71
スタティック タイミング解析用の入力ファイル.....	71
スタティック タイミング解析用の出力ファイル.....	71
NetGen スタティック タイミング解析の構文.....	71
NetGen スタティック タイミング解析フローのオプション	72
階層ファイルの保持および出力	74
テストベンチ ファイル	75
階層情報ファイル.....	75
バックアノテーション シミュレーションでの専用グローバル信号の処理.....	76
Verilog ネットリストのグローバル信号	76
VHDL ネットリストのグローバル信号	76

5 :	論理的デザイン ルール チェック	79
	論理的 DRC の概要	79
	論理的 DRC のデバイス サポート	79
	論理的 DRC によるチェック	80
	ブロック チェック	80
	ネット チェック	80
	パッド チェック	81
	クロック バッファ チェック	82
	ネーム チェック	82
	プリミティブ ピン チェック	82
6 :	NGDBuild	83
	NGDBuild の概要	83
	NGDBuild デザイン フロー	83
	NGDBuild のデバイス サポート	84
	ネットリストから NGD ファイルへの変換	84
	NGDBuild の入力ファイル	84
	NGDBuild の中間ファイル	86
	NGDBuild の出力ファイル	86
	NGDBuild の構文	86
	NGDBuild のオプション	87
	-a (最上位ポート信号への PAD の追加)	87
	-aul (不一致の LOC を許可)	88
	-aut (不一致のタイムグループを許可)	88
	-bm (BMM ファイルを指定)	88
	-dd (保存先ディレクトリ)	89
	-f (コマンド ファイルの実行)	89
	-i (UCF ファイルを無視)	89
	-insert_keep_hierarchy (KEEP_HIERARCHY 制約の挿入)	89
	-intstyle (統合スタイル)	89
	-filter (フィルタ ファイルの指定)	90
	-l (検索するライブラリ)	90
	-nt (ネットリスト変換タイプ)	90
	-p (製品番号)	91
	-quiet (警告およびエラーのみを表示)	91
	-r (LOC 制約の無視)	91
	-sd (指定したディレクトリの検索)	91
	-u (未展開のブロックを許可)	92
	-uc (ユーザー制約ファイル)	92
	-ur (ユーザー ルール ファイルの読み込み)	92
	-verbose (すべてのメッセージの表示)	93
7 :	MAP	95
	MAP の概要	95
	MAP デザイン フロー	96
	MAP のデバイス サポート	96
	MAP の入力ファイル	96
	MAP の出力ファイル	97
	MAP のプロセス	97
	MAP の構文	98
	MAP のオプション	100
	-activity_file (アクティビティ ファイルの指定)	100

-bp (スライス ロジックのマップ).....	101
-c (スライスのパック).....	101
-cm (カバー モード).....	102
-detail (詳細な MAP レポートの生成).....	103
-equivalent_register_removal (重複したレジスタの削除).....	103
-f (コマンド ファイルの実行).....	103
-global_opt (グローバル最適化).....	103
-ignore_keep_hierarchy (KEEP_HIERARCHY を無視).....	104
-intstyle (統合スタイル).....	104
-ir (RPM の生成に RLOC を使用しない).....	104
-filter (フィルタ ファイルの指定).....	104
-lc (LUT の結合).....	105
-logic_opt (ロジックの最適化).....	105
-mt (マルチスレッド).....	105
-ntd (非タイミングドリブン).....	106
-o (出力ファイル名).....	106
-ol (総体的なエフォートレベル).....	106
-p (製品番号).....	107
-power (消費電力の最適化).....	107
-pr (レジスタを I/O にパック).....	108
-register_duplication (レジスタの複製).....	108
-retiming (グローバル最適化中のレジスタのリタイミング).....	108
-smartguide (SmartGuide).....	109
-t (配置コスト テーブル).....	109
-timing (タイミングドリブンのパックと配置).....	110
-u (未使用ロジックを保持).....	110
-w (既存ファイルの上書き).....	111
-x (パフォーマンス評価モード).....	111
-xe (追加エフォートレベル).....	111
-xt (追加配置コスト テーブル).....	112
再合成および物理合成最適化.....	112
ガイド マップ.....	112
マップ結果のシミュレーション.....	113
MAP レポート (MRP) ファイル.....	114
物理合成レポート (PSR) ファイル.....	120
MAP の停止.....	122
8: 物理的デザイン ルール チェック.....	123
DRC の概要.....	123
デバイス サポート.....	123
DRC の入力ファイル.....	123
DRC の出力ファイル.....	124
DRC の構文.....	124
DRC のオプション.....	124
-e (エラー レポート).....	124
-o (出力ファイル).....	124
-s (サマリ レポート).....	124
-v (詳細レポート).....	125
-z (不完全プログラムのレポート).....	125
DRC によるチェック.....	125
DRC のエラーと警告.....	126

9 :	PAR (Place and Route)	127
PAR の概要	127	
PAR フロー	128	
PAR のデバイス サポート	128	
PAR の入力ファイル	128	
PAR の出力ファイル	129	
PAR のプロセス	129	
配置	129	
配線	129	
タイミングドリブン PAR	129	
PAR の構文	130	
PAR のオプション	132	
-activity_file (アクティビティ ファイルの指定)	132	
-clock_regions (クロック領域レポートの生成)	132	
-f (コマンド ファイルの実行)	133	
-intstyle (統合スタイル)	133	
-filter (フィルタ ファイルの指定)	133	
-k (再配線)	133	
-mt (マルチスレッド)	134	
-nopad (パッド レポートを生成しない)	134	
-ntd (非タイミングドリブン)	134	
-ol (総体的なエフォート レベル)	134	
-p (配置なし)	135	
-pl (配置エフォートレベル)	135	
-power (低消費電力の PAR)	136	
-r (配線なし)	136	
-rl (配線エフォートレベル)	136	
-smartguide (SmartGuide)	137	
-t (配置コスト テーブル)	137	
-ub (ボンディングされた I/O を使用)	138	
-w (既存ファイルの上書き)	138	
-x (パフォーマンス評価モード)	138	
-xe (追加エフォートレベル)	138	
PAR のレポート	139	
配置配線 (PAR) レポート	139	
ガイド レポート ファイル (GRF)	145	
ReportGen	147	
ReportGen の構文	147	
ReportGen の入力ファイル	147	
ReportGen の出力ファイル	148	
ReportGen のオプション	148	
PAR の停止	149	
10 :	SmartXplorer	151
SmartXplorer の新機能	151	
12.2 の新機能	151	
12.1 の新機能	151	
SmartXplorer の概要	152	
SmartXplorer の利点	152	
SmartXplorer のデバイス サポート	153	
SmartXplorer の使用	154	
SmartXplorer の実行	154	

インプリメンテーション ストラテジのみの実行	154
SmartXplorer で提供されているストラテジの使用	155
SmartXplorer での XST の使用	158
SmartXplorer での Synplify の使用	158
最高のストラテジの選択	159
複数のストラテジの同時実行	160
ザイリンクス環境の設定	160
結果の保存場所	160
ホストリスト ファイル	161
カスタム ストラテジ	163
カスタム ストラテジ ファイルの新しいフォーマット	163
カスタム ストラテジ ファイルの以前のフォーマット	164
SmartXplorer の構文	165
コマンド ライン構文	165
SmartXplorer のファイルとディレクトリ	165
SmartXplorer のオプション	166
SmartXplorer のレポート	176
smartxplorer.html	176
smartxplorer.txt	178
DesignFile_sx.log	179
SSH で SmartXplorer を実行するための設定	179
11 : XPower (XPWR)	181
XPower の概要	181
XPower のデバイス サポート	181
XPower のファイル	182
XPower の構文	182
XPower のオプション	183
-l (行数の制限)	183
-ls (サポートされるデバイスをリスト)	183
-o (消費電力レポート ファイル名の変更)	183
-s (SAIF または VCD ファイルの指定)	183
-tcl (Tcl スクリプト)	184
-v (詳細レポート)	184
-wx (XML 設定ファイルの生成)	184
-x (XML 設定ファイルの指定)	184
XPower のコマンド ラインの例	185
XPower の使用	185
SAIF または VCD データ入力	185
その他のデータ入力方法	186
消費電力レポート	186
標準消費電力レポート	187
詳細レポート	187
12 : PIN2UCF	189
PIN2UCF の概要	189
PIN2UCF のデザイン フロー	189
PIN2UCF のデバイス サポート	190
PIN2UCF のファイル タイプ	190
PIN2UCF の入力ファイル	190

PIN2UCF の出力ファイル	190
PIN2UCF の構文.....	193
PIN2UCF のオプション.....	193
-o (出力ファイル名)	193
-r (レポートファイルへの出力)	193
13 : TRACE	195
TRACE の概要	195
TRACE の入力ファイルと出力ファイル	195
TRACE のデバイス サポート.....	196
TRACE の入力ファイル.....	196
TRACE の出力ファイル.....	196
TRACE の構文	196
TRACE のオプション	197
-a (アドバンス解析).....	197
-e (エラー レポートの生成).....	197
-f (コマンド ファイルの実行)	198
-fastpaths (最速パスをレポート)	198
-intstyle (統合スタイル)	198
-filter (フィルタ ファイルの指定).....	198
-l (タイミング レポートに表示される項目を制限)	199
-n (パスをエンドポイント別にレポート).....	199
-nodatasheet (データシートを出力しない).....	199
-o (出力タイミング レポートのファイル名を指定).....	199
-s (スピードの変更).....	199
-stamp (STAMP タイミング モデル ファイルの生成).....	200
-tsi (TSI レポートの生成)	200
-u (制約が適用されないパスをレポート).....	200
-v (詳細レポートの生成).....	201
-xml (XML 出力ファイル名).....	201
TRACE のコマンド ラインの例.....	201
TRACE のレポート.....	202
TRACE によるタイミング検証	203
TRACE のレポートの内容	205
データシートレポート	207
確認済みセットアップとホールドのレポート生成	209
サマリレポート.....	210
エラー レポート.....	214
詳細レポート.....	216
OFFSET 制約	219
OFFSET IN 制約.....	220
OFFSET OUT 制約	225
PERIOD 制約	228
PERIOD ヘッダ	229
PERIOD パス.....	229
PERIOD パスの詳細.....	231
PHASE を使用した PERIOD.....	231
位相がシフトされた PERIOD パス	232
最小周期統計	232
TRACE の停止	233

14 :	Speedprint.....	235
	Speedprint の概要	235
	Speedprint のフロー	235
	Speedprint のデバイス サポート	235
	Speedprint のファイル タイプ	235
	Speedprint の構文	239
	Speedprint のオプション	239
	-intstyle (統合スタイル)	239
	-min (最小スピード データの表示)	240
	-s (スピード グレード)	240
	-stepping (ステッピング)	240
	-t (温度の指定).....	240
	-v (電圧の指定)	240
15 :	BitGen.....	243
	BitGen の概要.....	243
	デザイン フロー	244
	BitGen のデバイス サポート.....	244
	BitGen の入力ファイル.....	244
	BitGen の出力ファイル.....	244
	BitGen の構文.....	246
	BitGen のオプション	247
	-b (ロービット ファイルの作成).....	247
	-bd (ブロック RAM のアップデート).....	247
	-d (DRC を実行しない).....	248
	-f (コマンド ファイルの実行)	248
	-g (コンフィギュレーションの設定)	248
	-intstyle (統合スタイル)	268
	-j (BIT ファイルを生成しない).....	268
	-l (ロジック アロケーション ファイルの作成).....	268
	-m (マスク ファイルの作成)	269
	-r (パーシャル BIT ファイルの作成)	269
	-w (既存の出力ファイルの上書き).....	269
16 :	BSDLAnno	271
	BSDLAnno の概要	271
	BitGen のデバイス サポート.....	272
	入力ファイル	272
	出力ファイル	272
	BSDLAnno の構文.....	272
	BSDLAnno のオプション	273
	-intstyle (統合スタイル)	273
	-s (BSDL ファイルの指定).....	273
	BSDL ファイルの構成と BSDLAnno による変換	274
	BSDL ファイルのエンティティ宣言	274
	BSDL ファイルのジェネリック パラメータ	274
	BSDL ファイルの論理ポート記述	274
	BSDL ファイルのパッケージ ピンのマップ	275
	BSDL ファイルの use 文	275
	BSDL ファイルのスキャン ポート識別	276
	BSDL ファイルの TAP 記述	276
	BSDL ファイルのバウンダリレジスタ記述	276

シングル エンド ピンの BSDL ファイルの変更	277
差動ピンの BSDL ファイルの変更	278
DESIGN_WARNING セクション	279
ヘッダ コメント	279
ザイリンクス デバイスのバウンダリ スキャン ビヘイビア	279
17 : PROMGen	281
PROMGen の概要	281
PROMGen のデバイス サポート	282
PROMGen の入力ファイル	282
PROMGen の出力ファイル	282
PROMGen の構文	282
PROMGen のオプション	283
-b (ビット スワップをオフ)	283
-bd (データ ファイルの指定)	284
-bm (BMM ファイルの指定)	284
-bpi.dc (シリアルまたはパラレル デイジー チェーン接続)	284
-c (チェックサム)	284
-config_mode (コンフィギュレーション モード)	284
-d (下方向に読み込み)	285
-data_file (データ ファイルを追加)	285
-data_width (PROM データ幅の指定)	285
-f (コマンド ファイルの実行)	286
-i (初期バージョンを選択)	286
-intstyle (統合スタイル)	286
-l (レンジス カウントのディスエーブル)	286
-n (BIT ファイルの追加)	287
-o (出力ファイル名)	287
-p (PROM フォーマット)	287
-r (PROM ファイルの読み込み)	288
-s (PROM のサイズ)	288
-spi (ビット スワップをオフ)	288
-t (テンプレート ファイル)	288
-u (上方向に読み込み)	289
-ver (バージョン)	289
-w (既存の出力ファイルの上書き)	289
-x (ザイリンクス PROM の指定)	289
-z (圧縮をイネーブル)	289
PROM ファイルのビット スワップ	290
PROMGen の例	290
18 : IBISWriter	291
IBISWriter の概要	291
IBISWriter フロー	292
IBISWriter のデバイス サポート	292
IBISWriter の入力ファイル	292
IBISWriter の出力ファイル	292
IBISWriter の構文	292
IBISWriter のオプション	293
-allmodels (アーキテクチャに使用できるバッファ モデルをすべて含む)	293
-g (参照電圧の設定)	293
-intstyle (統合スタイル)	294
-ml (複数言語のサポート)	294

-pin (詳細なピンごとのパッケージ寄生情報の生成)	294
-truncate (出力ファイルでの信号名の最大長を指定)	294
-vccaux (VCCAUX 電圧レベルを設定)	295
19 : CPLDFit	297
CPLDFit の概要	297
CPLDFit デザイン フロー	297
CPLDFit のデバイス サポート	297
CPLDFit の入力ファイル	297
CPLDFit の出力ファイル	298
CPLDFit の構文	298
CPLDFit のオプション	299
-blkfanin (ファンクション ブロックの最大ファンインを指定)	299
-exhaust (包括的なフィットを有効)	300
-ignoredatagate (DATA_GATE 属性を無視)	300
-ignoretspec (タイミング使用を無視)	300
-init (パワーアップ値を設定)	300
-inputs (最適化に使用する入力数)	301
-iostd (I/O 規格の指定)	301
-keepio (未使用の入力を最適化しない)	301
-loc (指定したロケーション制約を保持)	301
-localfbk (ローカル フィードバックを使用)	302
-log (ログ ファイルの指定)	302
-nofbnand (フォールドバック NAND の使用を無効)	302
-nogclkopt (グローバル クロックの最適化を無効)	302
-nogsropt (グローバル セット/リセットの最適化を無効)	302
-nogtsropt (グローバル出力イネーブルの最適化を無効)	302
-noisp (ISP ピンの予約を無効)	303
-nomlopt (複数レベルのロジック最適化を無効)	303
-nouim (FASTConnect/UIM 最適化を無効)	303
-ofmt (出力形式の指定)	303
-optimize (集積度/スピードに基づいて最適化)	303
-p (製品番号)	304
-pinfbk (ピンのフィードバックを使用)	304
-power (電力モードの設定)	304
-pterm (最適化に使用する積項数)	304
-slew (スルー レートの設定)	305
-terminate (終端モードに設定)	305
-unused (未使用 I/O の終端モードを設定)	305
-wysiwyg (最適化を実行しない)	305
20 : TSIM	307
TSIM の概要	307
TSIM のデバイス サポート	307
TSIM の入力ファイル	307
TSIM の出力ファイル	307
TSIM の構文	307
21 : TAEEngine	309
TAEEngine の概要	309
TAEEngine のフロー	309
TAEEngine のデバイス サポート	309
TAEEngine の入力ファイル	310
TAEEngine の出力ファイル	310

TAEngine の構文	310
TAEngine のオプション	310
-detail (詳細レポートの生成).....	310
-iopath (パスのトレース).....	310
-l (出力ファイル名の指定).....	311
22 : Hprep6	313
Hprep6 の概要	313
Hprep6 デザイン フロー	313
Hprep6 のデバイス サポート	313
Hprep6 の構文	313
Hprep6 の入力ファイル	314
Hprep6 の出力ファイル	314
Hprep6 のオプション	314
-autosig (シグネチャの自動生成).....	314
-intstyle (統合スタイル)	314
-n (リードバックのシグネチャ値を指定).....	315
-nopullup (プルアップを無効)	315
-s (ISC ファイルの生成).....	315
-tmv (テスト ベクタ ファイルの指定).....	315
23 : XFLOW.....	317
XFLOW の概要	317
XFLOW デザイン フロー	318
XFLOW のデバイス サポート.....	318
XFLOW の入力ファイル	318
XFLOW の出力ファイル	320
XFLOW の構文	323
XFLOW のフロー タイプ	323
-config (FPGA 用 BIT ファイルの生成).....	323
-ecn (等価チェック用ファイルの生成).....	324
-fit (CPLD のフィット).....	324
-fsim (論理シミュレーション用ファイルの生成).....	325
-implement (FPGA のインプリメンテーション).....	326
-sta (スタティック タイミング解析用ファイルの生成).....	326
-synth	327
-tsim (タイミング シミュレーション用ファイルの生成).....	328
フロー ファイル	329
ザイリンクス フロー ファイル.....	329
フロー ファイルのフォーマット	329
ユーザー コマンド ブロック.....	332
XFLOW のオプション ファイル	332
オプション ファイルのフォーマット	333
XFLOW のオプション	334
-ed (エクスポート ディレクトリにファイルをコピー).....	334
-f (コマンド ファイルの実行)	335
-g (グローバル変数の指定).....	335
-log (ログ ファイルの指定).....	335
-norun (スクリプト ファイルのみを生成).....	335
-o (出力ファイル名の変更)	336
-p (製品番号)	336
-rd (レポート ファイルのコピー).....	337

-wd (作業ディレクトリの指定).....	337
XFLOW の実行	338
XFLOW のフロー タイプを組み合わせ使用	338
Smart Flow	338
SCR、BAT、TCL ファイルの使用	338
XIL_XFLOW_PATH 環境変数の使用	339
24 : NGCBuild	341
NGCBuild の概要	341
NGCBuild のデバイス サポート.....	341
設計フローでの NGCBuild の使用	342
NGCBuild の入力ファイル (<infile[.ext]>).....	342
NGCBuild の出力ファイル (<outfile[.ngc]>).....	342
NGCBuild での NGC ファイルの検証	342
NGCBuild のメッセージとレポート.....	343
NGCBuild の構文	343
NGCBuild のオプション	343
-aul (不一致の LOC を許可).....	343
-dd (保存先ディレクトリ).....	344
-f (コマンド ファイルの実行)	344
-i (UCF ファイルを無視)	344
-insert_keep_hierarchy (KEEP_HIERARCHY 制約の挿入)	344
-intstyle (統合スタイル)	345
-filter (フィルタ ファイルの指定).....	345
-nt (ネットリスト変換タイプ).....	345
-p (製品番号)	345
-quiet (警告およびエラーのみを表示)	345
-r (LOC 制約の無視).....	346
-sd (指定したディレクトリの検索).....	346
-uc (ユーザー制約ファイル).....	346
-ur (ユーザー ルール ファイルの読み込み).....	347
-verbose (すべてのメッセージの表示).....	347
25 : Compplib	349
Compplib の概要	349
デザインフロー	350
Compplib のデバイス サポート.....	350
Compplib の構文	350
Compplib のオプション	351
-arch (デバイス ファミリ).....	351
-cfg (コンフィギュレーション ファイルの作成)	352
-dir (出力ディレクトリ).....	353
-e (既存のディレクトリ).....	353
-exclude_deprecated (廃止予定の EDK ライブラリを除外)	353
-exclude_sublib (EDK サブライブラリを除外).....	353
-f (コマンド ファイルの実行)	353
-info (コンパイル済みライブラリの情報の表示).....	354
-l (言語)	354
-lib (コンパイルするライブラリの名前の指定)	354
-log (ログ ファイル).....	355
-p (シミュレータのパス).....	355
-s (ターゲット シミュレータ).....	355
-source_lib (ソース ライブラリ).....	355
-verbose (詳細メッセージ).....	356

-w (コンパイル済みライブラリの上書き).....	356
Compplib のコマンド ラインの例	356
システム管理者としてのライブラリのコンパイル	356
ユーザーとしてのライブラリのコンパイル	356
Compplib のその他の使用例	357
ランタイム オプションの指定	358
EXECUTE.....	358
EXTRACT_LIB_FROM_ARCH.....	358
LOCK_PRECOMPILED	358
LOG_CMD_TEMPLATE.....	358
HIER_OUT_DIR	358
PRECOMPILED_INFO.....	359
BACKUP_SETUP_FILES	359
FAST_COMPILE	359
ABORT_ON_ERROR.....	359
ADD_COMPILATION_RESULTS_TO_LOG	359
USE_OUTPUT_DIR_ENV	359
OPTION	360
コンフィギュレーション ファイルの例 (Windows).....	360
26 : XWebTalk	365
WebTalk の概要.....	365
XWebTalk の構文.....	366
XWebTalk のオプション	366
-user (ユーザーごとの設定)	366
-install (インストールごとの設定)	367
-info (WebTalk 設定の確認).....	367
付録 A ISE Design Suite のファイル	369
付録 B EDIF2NGD と NGDBuild.....	375
EDIF2NGD の概要	375
EDIF2NGDデザイン フロー	376
EDIF2NGD のデバイス サポート	376
EDIF2NGD の構文	376
EDIF2NGD の入力ファイル	377
EDIF2NGD の出力ファイル	378
EDIF2NGD のオプション	378
-a (最上位ポート信号への PAD の追加).....	378
-aul (不一致の LOC を許可).....	378
-f (コマンド ファイルの実行)	379
-intstyle (統合スタイル)	379
-l (検索するライブラリ).....	379
-p (製品番号)	379
-r (LOC 制約の無視).....	380
NGDBuild.....	380
ネットリストから NGD ファイルへの変換.....	380
バスの一致.....	382
ネットリスト ラウンチャ	383
ネットリスト ラウンチャのルール ファイル	385
NGDBuild で使用するファイルの名前と保存先	390

付録 C Tcl リファレンス	391
Tcl の概要	391
Tcl のデバイス サポート	391
ザイリンクス Tcl シェル	391
ザイリンクス Tcl コマンドのヘルプの表示	392
Tcl の基礎	392
ザイリンクス名前空間	393
プロジェクト プロパティとプロセス プロパティ	393
プロジェクト プロパティ	394
プロセス プロパティ - [Synthesize - XST] プロセス	394
プロセス プロパティ - [Translate] プロセス	398
プロセス プロパティ - [Map] プロセス	399
プロセス プロパティ - [Place & Route] プロセス	401
プロセス プロパティ - [Generate Programming File] プロセス	402
プロセス プロパティ - [Generate Post-Place & Route Simulation Model] プロセ ス	406
一般用法のザイリンクス Tcl コマンド	408
lib_vhdl (VHDL ライブラリの制御)	408
process (プロジェクト プロセスの実行と制御)	411
project (プロジェクトの作成と制御)	414
xfile (ISE ソース ファイルの制御)	421
アドバンス スクリプト用のザイリンクス Tcl コマンド	426
globals (ザイリンクス グローバル データの操作)	426
collection (コレクションの作成と制御)	428
object (オブジェクト情報の表示)	436
search (デザイン オブジェクトの検索)	440
Tcl スクリプト例	442
標準 Tcl スクリプト例	443
一般用法の Tcl スクリプト例	445
その他のザイリンクス Tcl スクリプト例	445

はじめに

この章では、ISE® Design Suite のコマンドライン プログラムに関する基本的な情報を示します。このガイドの名前は、『開発システムリファレンスガイド』から『コマンドライン ツール ユーザー ガイド』に変更されました。この章には、次のセクションが含まれています。

- ・ コマンドライン プログラムの概要
- ・ コマンドラインの構文
- ・ コマンドライン オプション
- ・ コマンドライン プログラムの起動

コマンドライン プログラムの概要

ザイリンクス ソフトウェアのコマンドライン プログラムを使用すると、デザインのインプリメンテーションおよび検証を実行できます。次の表に、デザイン フローの各段階で使用するプログラムを示します。詳細は、「[デザイン フロー](#)」の章を参照してください。

デザイン フローで使用するコマンドライン プログラム

デザイン フローの段階	コマンドライン プログラム
デザイン インプリメンテーション	NGDBuild、MAP、PAR、SmartXplorer、BitGen
タイミングドリブンの配置配線、再合成、物理合成最適化	MAP メモ： 特定のオプションを使用してタイミングドリブンの配置配線 (-timing)、再合成および物理合成最適化をイネーブルにします。これらのオプションは、デザイン要件を満たすのに役立ちます。
タイミング シミュレーション、バック アノテーション (デザイン検証)	NetGen
スタティック タイミング解析 (デザイン検証)	TRACE

コマンドライン プログラムは、標準デザイン フローで実行するか、デザインを保持するよう特別なオプションを使用して実行できます。各プログラムには複数のオプションがあり、プログラムの実行方法を制御できます。たとえば、出力ファイル名の変更、デザインで使用するデバイスの設定、プログラムの実行時に読み込むファイルを指定するオプションがあります。また、ガイド ファイルを作成したり、ガイド モードを使用して以前にインプリメントしたデザインのパフォーマンスを保持できるオプションもあります。

一部のコマンドライン プログラムは、多くのザイリンクスのグラフィカル ユーザー インターフェイス (GUI) の基礎となっています。GUI は、コマンドライン プログラムと共に使用できます。GUI の詳細は、各ツールのヘルプを参照してください。

コマンドラインの構文

コマンドラインには、コマンドライン プログラム名、オプション、ファイル名の順に入力します。コマンドライン オプションを指定する際は、次の規則に従ってください。

- ・ オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。
- ・ 大文字と小文字は正しく入力します。
- ・ オプションにパラメータを指定する必要がある場合は、オプションとパラメータをスペースまたはタブで区切ります。たとえば、エフォートレベルを high に設定して PAR を実行する構文は次のとおりです。

- 正 : **par -ol high**

- 誤 : **par -olhigh**

- ・ 複数回指定できるオプションは、パラメータごとにオプションを指定する必要があります。たとえば、検索するライブラリのリストを指定する構文は次のとおりです。

- 正 : **-l xilinxun -l synopsys**

- 誤 : **-l xilinxun synopsys**

- ・ パラメータは、オプションの後ろに入力します。

- 正 : **-f command_file**

- 誤 : **command_file -f**

ファイル名を指定する際は、次の規則に従ってください。

- ・ ファイル名は、コマンドライン プログラムの章で指定されている順序で入力します。PAR プログラムの場合、入力ファイル、出力ファイル、物理制約ファイルの順に指定します。

- 正 : **par input.ncd output.ncd freq.pcf**

- 誤 : **par input.ncd freq.pcf output.ncd**

- ・ ファイルの拡張子には、小文字を使用します (.ncd など)。

コマンドライン オプション

ISE® Design Suite の多くのコマンドライン プログラムに共通のオプションを示します。

- ・ **-f** (コマンド ファイルの実行)
- ・ **-h** (ヘルプ)
- ・ **-intstyle** (統合スタイル)
- ・ **-p** (製品番号)

-f (コマンド ファイルの実行)

ザイリンクスのコマンド ライン プログラムを FPGA デザインに対して使用する場合、コマンド ライン プログラムのオプションとファイル名をコマンド ファイルに保存しておくことができます。このコマンド ファイルに含まれる引数を実行するには、プログラム名の後に **-f** オプションでコマンド ファイルを指定します。この方法は、コマンドを同じ引数を使用して繰り返し実行する場合や、コマンド ラインが長い場合に便利です。

構文

-f *command_file*

コマンド ファイルは、次のように使用します。

- ・ プログラムのすべてのコマンド オプションおよびファイル名が指定されている場合は、次のように入力します。

par -f *command_file*

command_file : コマンド オプションとファイル名を含むコマンド ファイルの名前です。

- ・ 特定のコマンド オプションおよびファイル名のみが指定されている場合は、次のように入力します。

par -f *placeoptions -f routeoptions design_i.ncd design_o.ncd*

・ *placeoptions* : 配置コマンド パラメータを含むファイルの名前です。

・ *routeoptions* : 配線コマンド パラメータを含むファイルの名前です。

コマンド ファイルは、ASCII 形式で作成します。コマンド ファイルを作成する際は、次の規則に従ってください。

- ・ プログラムのオプションとファイル名は、スペースで区切ります。
- ・ コメントは、シャープ記号 (#) の後に入力します。
- ・ Linux または DOS コマンド ライン上でスペースを入力する位置には、改行またはタブを使用できます。
- ・ 同じ行にすべての引数を入力するか、1 行に 1 つの引数を入力、またその両方を組み合わせることができます。
- ・ キャリッジリターンやその他の印刷できない文字は、スペースとして処理され、無視されます。
- ・ ファイル内で、行の長さに制限はありません。

例

次は、コマンド ファイルの例です。

```
#command line options for par for design mine.ncd
-w
01 5
/home/yourname/designs/xilinx/mine.ncd
#directory for output designs
/home/yourname/designs/xilinx/output.dir
#use timing constraints file
/home/yourname/designs/xilinx/mine.pcf
```

-h (ヘルプ)

プログラムを `-help` または `-h` オプションを使用して実行すると、そのプログラムのオプションおよびパラメータのリストと、オプションの説明、使用できるファイル タイプが表示されます。

構文

-h

-help

シンボル	説明
[]	オプションの項目を示します。
{ }	繰り返し可能な項目を示します。
イタリック フォント	実際の値に置き換える必要のある変数名または数値を示します。
,	整数変数の範囲を示します。
-	オプション名の開始を示します。
:	変数名の範囲を指定します。
	論理 OR と同義で、複数の項目から 1 つを選択することを示します。論理グループまたはリテラル キーワードを区切るだけの場合もあります。
()	サブフォーマットから選択する論理グループを示します。

例

次に、ファイル名の構文例を示します。

- ・ `infile[.ncd]` : 拡張子 `.ncd` の指定はオプションですが、指定する場合は `.ncd` を使用する必要があることを示します。
- ・ `infile.edn` : 拡張子 `.edn` の指定はオプションで、ファイル名にほかの拡張子が付いていない場合にのみ付加されることを示します。

BitGen のようにアーキテクチャ特有のオプションを使用できるプログラムの場合、次のように入力すると、アーキテクチャ特有のヘルプが表示されます。

```
program_name -h architecture_name
```

次のように入力すると、ヘルプ メッセージをファイルに保存し、表示したり印刷したりできます。

```
program_name -h > filename
```

Linux コマンドラインに次のように入力すると、ヘルプ メッセージをファイルに保存してから、コマンド プロンプトに戻ることができます。

```
program_name -h > & filename
```

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle ise|xflow|silent

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-p (製品番号)

デザインをインプリメントするパーツを指定します。

構文

-p part_number

このオプションでは、アーキテクチャのみ、完全なパーツ仕様 (デバイス、パッケージ、およびスピード)、または部分的な仕様 (デバイスとパッケージのみなど) を指定できます。製品番号 (デバイス名) は、システム上にインストールしたデバイス ライブラリに含まれているもののみを指定できます。

ザイリンクスの製品番号は、次の要素から構成されています。

- ・ アーキテクチャ (例 : spartan3e)
- ・ デバイス (例 : xc3s100e)
- ・ パッケージ (例 : vq100)
- ・ スピード (例 : -4)

メモ : Speedprint プログラムは、デバイスのスピード グレードでのブロック遅延をリストします。**-s** オプションを使用すると、スピード グレードを指定できます。スピード グレードを指定しない場合は、ターゲット デバイスのデフォルトのスピード グレードが表示されます。

製品番号の指定

製品番号は、次に示すデザイン フローのさまざまな段階で指定できます。**-p** を指定する必要がない場合もあります。

- ・ 入力ネットリスト内 (**-p** オプションは不要)
- ・ ネットリスト制約ファイル (NCF) 内 (**-p** オプションは不要)
- ・ ネットリストリーダー (EDIF2NGD) を実行するとき (**-p** オプションを使用)
- ・ ユーザー制約ファイル (UCF) 内 (**-p** オプションは不要)
- ・ NGDBuild 実行時 (**-p** オプションを使用)

NGDBuild を実行するまでに、デバイス アーキテクチャを指定しておく必要があります。

- ・ MAP 実行時 (**-p** オプションを使用)

MAP を実行する場合、MAP コマンド ラインか、デザイン フローのそれ以前の段階で、アーキテクチャ、デバイス、パッケージを指定する必要があります。スピードを指定しな

い場合は、デフォルトのスピードが使用されます。MAP は、NGCBuild の実行時に指定したアーキテクチャのデバイスに対してのみ実行できます。

- ・ SmartXplorer 実行時 (-p オプションを使用、FPGA デザインのみ)
- ・ CPLDFit 実行時 (-p オプションを使用、CPLD デザインのみ)

メモ： デザイン フローの後の段階で製品番号を指定すると、それ以前の段階で指定した製品番号は無効になります。たとえば、MAP の実行時に製品番号を指定すると、入力ネットリストで指定した 製品番号は無効になり、MAP の実行時に指定した製品番号が使用されます。

例

次に、コマンドラインでのパーツの指定方法を示します。

指定方法	例
アーキテクチャのみ	virtex4 virtex5 spartan3 spartan3a xc9500 xpla3 (CoolRunner™ XPLA3 デバイス)
デバイスのみ	xc4vfx12 xc3s100e
デバイス パッケージ	xc4fx12sf363 xc3s100evq100
デバイス - パッケージ	xc4vfx12-sf363 xc3s100e-vq100
デバイス スピード - パッケージ	xc4vfx1210-sf363 xc3s100e4-vq100
デバイス パッケージ - スピード	xc4fx12sf363-10 xc3s100evq100-4
デバイス - スピード - パッケージ	xc4vfx12-10-sf363 xc3s100e-4-vq100
デバイス - スピード パッケージ	xc4vfx12-10sf363 xc3s100e-4vq100

コマンドライン プログラムの起動

ザイリンクスのコマンドライン プログラムを起動するには、Linux または DOS のコマンドラインにコマンドを入力します。構文については、このマニュアルの各プログラムの章を参照してください。

XFLOW プログラムを使用すると、複数のプログラムの実行を自動化して 1 回で実行できるように設定できます。詳細は、「[XFLOW](#)」の章を参照してください。

デザイン フロー

この章では、FPGA デバイスおよび CPLD デバイスのデザインを作成、インプリメント、検証、ダウンロードするプロセスについて説明します。ザイリンクス FPGA および CPLD の詳細は、<http://japan.xilinx.com/support/documentation/index.htm> からデバイスのデータシートを参照してください。

この章に含まれるセクションは、次のとおりです。

- ・ [デザイン フローの概要](#)
- ・ [デザイン入力および合成](#)
- ・ [デザイン インプリメンテーション](#)
- ・ [デザイン検証](#)
- ・ [FPGA デザインのヒント](#)

デザイン フローの概要

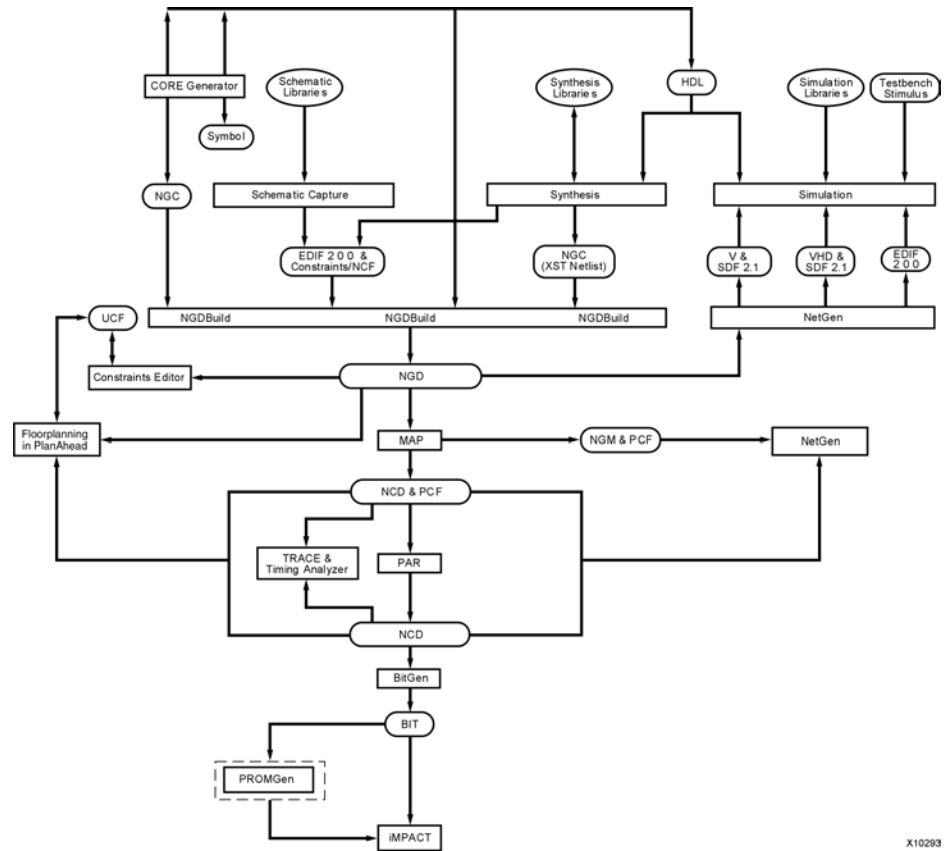
標準デザイン フローには、次の段階があります。

1. **デザイン入力および合成**：ザイリンクスがサポートしている回路図エディタ、テキストで入力するハードウェア記述言語 (HDL)、またはその両方を使用してデザインを作成します。テキスト エディタで入力する HDL ファイルを使用する場合は、HDL ファイルを EDIF ファイル (XST を使用する場合は NGC ファイル) に合成する必要があります。
2. **デザイン インプリメンテーション**：特定のザイリンクス アーキテクチャにインプリメントすることにより、デザイン入力段階および合成段階で作成した EDIF などの論理デザイン ファイルのフォーマットを物理ファイルに変換します。物理的な情報は、FPGA では NCD (Native Circuit Description) ファイルに、CPLD では VM6 ファイルに保存されます。これらのファイルからビットストリーム ファイルを作成し、その後、ザイリンクス デバイスのプログラムに使用する PROM または EPROM を必要に応じてプログラムします。
3. **デザイン検証**：ゲートレベルのシミュレータまたはケーブルを使用して、タイミング要件に適合しているか、デザインが正常に機能するかを検証します。ザイリンクスのダウンロード ケーブルとデモ ボードについては、iMPACT ヘルプを参照してください。

完全なデザイン フローは、デザインの入力から、インプリメンテーションおよび検証を経て、デザインが完成するまでのすべてのプロセスです。ISE® Design Suite に含まれるコマンド ライン ツールを使用すると、このデザイン フロー サイクルを迅速に繰り返すことができます。ザイリンクスのデバイスは何度でもプログラムし直すことができるため、回路にデザインを組み込んでデバッグする際に、デバイスを廃棄する必要はありません。

ザイリンクス ソフトウェアのデザイン フロー (FPGA)

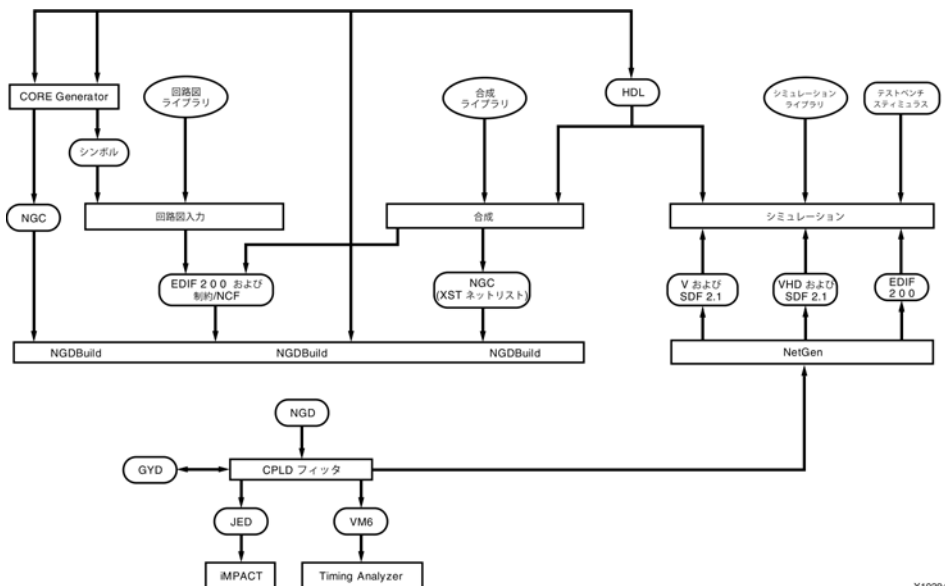
次の図に、FPGA デザインにおけるザイリンクス ソフトウェアのフローを示します。



X10293

ザイリンクス ソフトウェアのデザイン フロー (CPLD)

次の図に、CPLD デザインにおけるザイリンクス ソフトウェアのフローを示します。

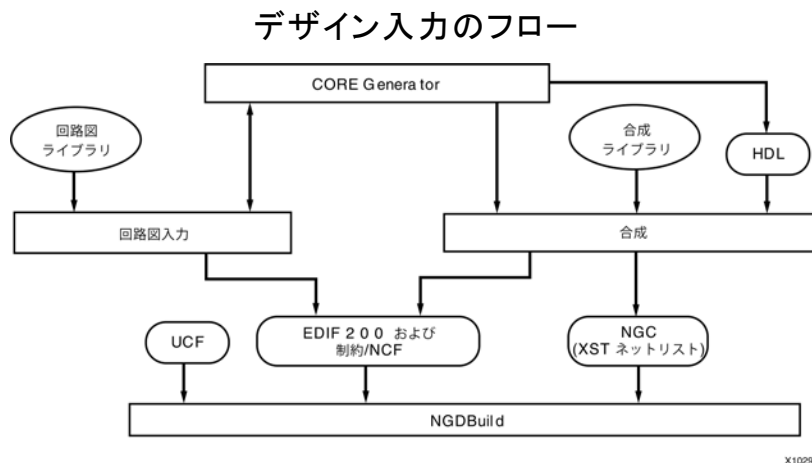


X10294

デザイン入力および合成

デザインは、回路図エディタまたはテキスト エディタで入力できます。デザインを入力するには、デザイン コンセプトを決定し、回路図または論理記述で表現します。元のデザインからネットリストを生成し、合成して、NGO (Native Generic Object) ファイルに変換します。このファイルを入力として読み込んで NGDBuild プログラムを実行し、論理的な NGD (Native Generic Database) ファイルを生成します。

次に、デザインの入力および合成のフローを示します。



階層デザイン

回路図入力および HDL 入力では、次のような理由からデザイン階層を使用することが重要です。

- ・ デザインの全体像を把握するのに役立ちます。
- ・ デザインを階層構造にできます。
- ・ デザインを容易にデバッグできます。
- ・ デザインの異なる部分に異なるデザイン入力方法 (回路図エディタ、HDL エディタ、またはステート エディタ) を使用できます。
- ・ デザインを部分ごとに段階的に設計、インプリメント、および検証していくことが可能です。
- ・ 最適化の時間を短縮できます。
- ・ 並行設計が容易になります。並行設計とは、複数の設計者がデザインを分割、分担して同時進行で開発していく方法です。

階層デザインでは、作成したライブラリ エlement、ブロック、インスタンスを階層名で識別します。次の例では、4 ビット カウンタに含まれるマルチプレクサの最初のインスタンスにある 2 入力 OR ゲートの階層名を表します。

```
/Acc/alu_1/mult_4/8count_3/4bit_0/mux_1/or2
```

デザイン内のコンポーネントとネットには、名前を付けてください。コンポーネント名とネット名は保存され、FPGA Editor で使用されます。バックアノテーションでもこれらの名前が使用され、デバッグ ツールと解析ツールで表示されます。コンポーネントとネットに名前を付けない場合、Schematic Editor により自動的に名前が生成されますが、たとえば次のような名前になってしまいます。

/ \$1a123 / \$1b942 / \$1c23 / \$1d235 / \$1e121 / \$1g123 / \$1h57

メモ：このように自動生成された名前では、回路図の解析が困難になります。

パーティション

デザイン保持およびパーシャル リコンフィギュレーションなどの階層デザイン フローでは、パーティションを使用して階層の境界を定義することにより、複雑なデザインを小さなブロックに分割できます。パーティションにより階層モジュールに境界が作成され、デザインのほかの部分と分離されます。インプリメントおよびエクスポートしたパーティションは、単純な切り取り / 貼り付け操作によりデザインに再挿入でき、そのモジュールの配置配線結果が保持されます。パーティションの定義および制御には、xpartition.pxml というファイルを使用します。異なる階層デザイン フローの使用およびパーティションのインプリメンテーションの詳細は、『[階層デザイン手法ガイド](#)』(UG748) を参照してください。

PXML ファイル

パーティションの定義は、xpartition.pxml ファイルに含まれます。PXML ファイルでは大文字と小文字が区別され、xpartition.pxml という名前を付ける必要があります。下位モジュールをパーティションとして定義するには、最上位モジュールをパーティションとして定義する必要があります。PXML ファイルは、手動で作成するか、スクリプトで作成するか、PlanAhead などのグラフィカル ユーザー インターフェイス (GUI) ソフトウェア ツールを使用して作成できます。PXML ファイルが現在の作業ディレクトリに存在していれば、ISE インプリメンテーション ツールで自動的に検出されます。xpartition.pxml ファイルの使用については、『[階層デザイン手法ガイド](#)』(UG748) を参照してください。PXML ファイルを手動で作成する場合は、%XILINX%/PlanAhead/testcases (%XILINX% はインストール ディレクトリ) にある xpartition.pxml ファイルの例を参照してください。

```
<?xml version="1.0" encoding="UTF-8" ?>

<Project FileVersion="1.2" Name="Example" ProjectVersion="2.0">
  <Partition Name="/top" State="implement" ImportLocation="NONE">
    <Partition Name="/top/module_A" State="import" ImportLocation="/home/user/Example/import" Preserve="routing">
    </Partition>
    <Partition Name="/top/module_B" State="import" ImportLocation="../import" Preserve="routing">
    </Partition>
    <Partition Name="/top/module_C" State="implement" ImportLocation="../import" Preserve="placement">
    </Partition>
  </Partition>
</Project>
```

プロジェクトを定義するための PXML 属性

属性名	値	説明
FileVersion	1.2	この値は変更しないでください。
Name	<i>Project_name</i>	プロジェクト名を指定します。
ProjectVersion	2.0	この値は変更しないでください。

パーティションを定義するための PXML 属性

属性名	値	説明
Name	<i>Partition_Name</i>	パーティションを適用するモジュールの階層名を指定します。
State	implement	パーティションは再インプリメントされます。
	import	パーティションはインポートされ、Preserve で設定されたレベルでインプリメンテーションが保持されます。
ImportLocation	<i>path</i>	インポート場所を指定します。State が import に設定されていない場合は無視されます。State が import に設定されている場合、パスを相対パスまたは絶対パスで指定できますが、指定した場所に export ディレクトリが含まれている必要があります。この属性が無視される場合、ディレクトリを設定したりこの属性を削除する代わりに、NONE キーワードを設定できます。
Preserve	routing	配置および配線が保持されます（最上位パーティションのデフォルト設定）。
	placement	配置が保持されます。配線は変更される場合があります。
	synthesis	配置および配線が変更される場合があります。

回路図入力の概要

回路図ツールを使用すると、グラフィカル インターフェイスを使用してデザインを入力できます。回路図エディタを使用して、デザイン内のロジック コンポーネントを表すシンボルを接続します。個別のゲートでデザインを構築したり、ゲートを組み合わせてファンクション ブロックを作成できます。このセクションでは、ライブラリ エLEMENT や CORE Generator™ ツールを使用したファンクション ブロックの入力方法について説明します。

ライブラリ エLEMENT

プリミティブとマクロは、コンポーネント ライブラリの基本ブロックです。ザイリンクスのライブラリには、プリミティブおよび一般的な上位マクロ ファンクションが含まれています。プリミティブは AND ゲートや OR ゲートなどの基本ELEMENTで、プリミティブには固有のライブラリ名、シンボル、記述があります。マクロには、プリミティブやほかのマクロなど、複数のライブラリ ELEMENTが含まれています。

ザイリンクスの FPGA では、次のタイプのマクロを使用できます。

- ・ ソフト マクロ：あらかじめ機能が定義されていますが、マップ、配置、配線を柔軟に実行できます。すべての FPGA で使用できます。
- ・ 相対配置マクロ (RPM)：固定マップや相対配置を実行できます。RPM は、XC9500 ファミリを除くすべてのデバイス ファミリで使用できます。

合成ツールには独自のモジュール ジェネレータが含まれており、RPM を必要としないので、マクロは合成では使用されません。合成ツールによるモジュール生成を無効にする場合は、CORE Generator で作成したモジュールをインスタンス化します。ほとんどの合成ツールでは、推論できないモジュールがある場合を除き、モジュール生成を無効にしても利点はありません。

CORE Generator ツール (FPGA のみ)

ザイリンクス CORE Generator ツールでは、ザイリンクス FPGA 用に最適化されたパラメータ指定コアを使用できます。ライブラリには、遅延エレメントなどの単純なコアから、DSP (デジタル信号処理) フィルタやマルチプレクサなどの複雑なコアまで多様なコアが含まれています。詳細は、ISE ヘルプに含まれる CORE Generator ヘルプ、または最新版の IP ソリューションを提供しているザイリンクス IP コアの Web ページ (<http://japan.xilinx.com/ipcenter>) を参照してください。このサイトから、デザイン再利用ツールやリファレンス デザイン、デジタル信号処理 (DSP) および PCI™ ソリューション、インプリメンテーション ツール、コア、特別なシステムレベルのサービス、サードパーティのコアなどを入手できます。

HDL 入力および合成

一般的なハードウェア記述言語 (HDL) では、論理記述にゲート文とネットリスト文を含めることができる混合記述がサポートされています。混合記述では、最初にシステム アーキテクチャを抽象度の高いレベルで記述し、徐々に詳細なゲート レベル インプリメンテーションを実現します。

HDL 記述には、次のような利点があります。

- ・ 設計の初期段階で、デザインの機能を検証できます。HDL で記述されたデザインは、すぐにシミュレーションできます。ハイレベル、インプリメンテーションの前のゲートレベルでデザインをシミュレーションすることにより、アーキテクチャおよびデザインの定義を評価できます。
- ・ HDL 記述は、ネットリスト記述や回路図記述よりも読みやすく、理解しやすい記述です。HDL ではテクノロジーに依存しない記述が可能なので、同じ HDL 記述を使用して異なるテクノロジーでデザインを生成できます。
- ・ HDL ツールでは、大規模なデザインを回路図ツールよりも簡単に扱うことができます。

HDL デザインを作成した後、デザインを合成する必要があります。合成では、HDL ファイル内のビヘイビアレベルの情報がネットリストに変換され、デザインがザイリンクス デバイス用に最適化されます。ザイリンクス では、サードパーティ ベンダー数社の HDL 合成ツールをサポートしています。また、ザイリンクスの合成ツール XST を提供しています。詳細は、『[XST ユーザー ガイド](#)』または『[XST ユーザー ガイド \(Virtex-6 および Spartan-6 デバイス用\)](#)』を参照してください。合成の詳細は、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

論理シミュレーション

デザイン作成を完了したら、シミュレーションを実行できます。論理シミュレーションでは、デザインのロジックをテストし、デザインが適切に動作するかどうかを検証します。デザインフローの初期段階で論理シミュレーションを実行しておく、その後の段階でデバッグにかかる時間を節約できます。詳細は、「[シミュレーション](#)」を参照してください。

制約

特定のタイミングや配置パラメータの範囲内にデザインを制約することが必要な場合があります。制約を使用すると、マップ、ブロック配置、タイミングの仕様を指定できます。

制約は手動で入力するか、Constraints Editor または FPGA Editor を使用して入力します。Timing Analyzer または TRACE コマンドライン プログラムを使用すると、デザインのスタティック タイミング解析を実行してデザインに設定された制約を解析できます。詳細は、「[TRACE](#)」の章および ISE ヘルプを参照してください。制約の詳細は、『[制約ガイド](#)』を参照してください。

マップ制約 (FPGA のみ)

すべての Spartan® および Virtex® FPGA アーキテクチャでロジックのブロックを CLB にどのようにマップするかは、FMAP を使用して指定できます。FMAP シンボルは回路図に使用できますが、これらの制約を多数設定すると、デザインの配線が困難になる場合があります。

ブロックの配置

ブロックの配置は、特定のロケーション、複数のロケーションのいずれか、またはロケーションの範囲に制約できます。ロケーションは、回路図、合成ツール、またはユーザー制約ファイル (UCF) を使用して指定します。ブロックの配置が適切でないと、デザインの配置と配線に悪影響を及ぼす場合があります。I/O ブロックだけは、外部ピン要件を満たすため、必ず配置する必要があります。

タイミング仕様

デザイン内のパスのタイミング要件を指定できます。PAR では、デザインを配置配線する際、これらのタイミング仕様を使用して最適なパフォーマンスを達成します。

ネットリスト変換プログラム

ネットリスト変換プログラムを使用すると、ネットリストをザイリンクス ソフトウェア ツールに読み込むことができます。EDIF2NGD では、EDIF (Electronic Data Interchange Format) 2 0 0 ファイルを読み込むことができます。NGDBuild は、必要に応じて変換プログラム EDIF2NGD を自動的に呼び出し、EDIF ファイルをザイリンクス ソフトウェア ツールで使用する NGD ファイルに変換します。ザイリンクスの合成ツール XST で生成された NGC ファイルは、NGDBuild で直接読み込むことができます。

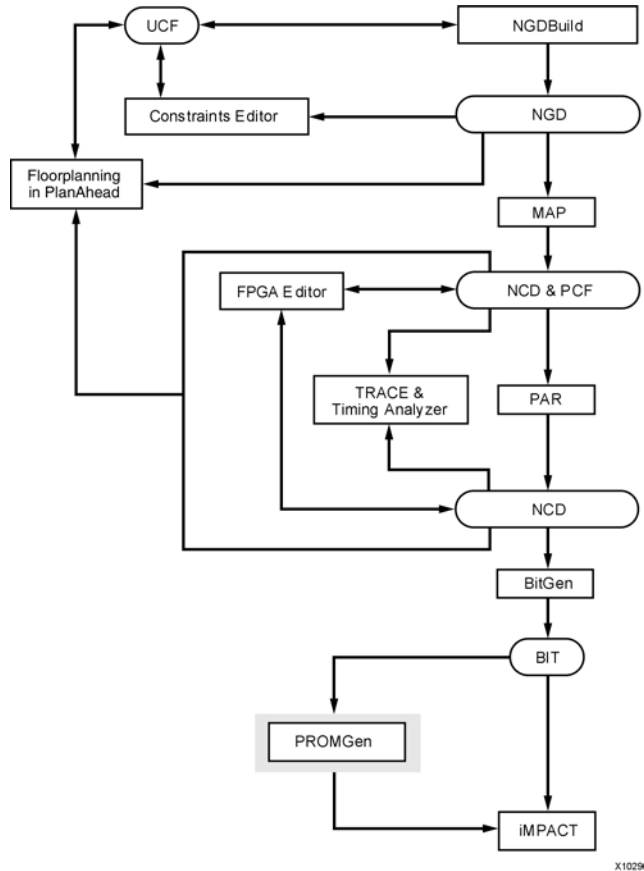
EDIF2NGD と NGDBuild の詳細は、「[NGDBuild](#)」の章および付録「[EDIF2NGD と NGDBuild](#)」を参照してください。

デザイン インプリメンテーション

デザイン インプリメンテーションは、まず論理的デザイン ファイルを特定のデバイスにマップまたはフィットし、物理的デザインを配線してビットストリームを生成すると完了します。デザイン入力段階と同様に、インプリメンテーション中にも制約を変更できます。詳細は、「[制約](#)」を参照してください。

次に、FPGA デザインのインプリメンテーション プロセスを示します。

FPGA デザイン インプリメンテーションのフロー



次に、CPLD デザインのインプリメンテーション プロセスを示します。

メモ： MAP には、標準のインプリメンテーションからさらにタイミングを向上させるアドバンス最適化を実行するオプションがあります。これらのアドバンス最適化は、配置前後のデザインを変換します。最適化は、ロジックのアーキテクチャ スライスへの初期マップ後、または配置後のデザインに対して適用できます。詳細は、「MAP」の章の「[再合成および物理合成最適化](#)」を参照してください。

配置配線 (FPGA のみ)

PAR コマンドライン プログラムは、マップ済みの NCD ファイルを読み込み、デザインを配置配線して、ビットストリーム ジェネレータ (BitGen) で使用する NCD ファイルを生成します。NCD ファイルは、配置配線を実行した後に、デザインに少しの変更を加えた後配置配線し直す場合に、ガイド ファイルとしても使用できます。詳細は、「[PAR](#)」の章を参照してください。

FPGA Editor を使用して、次の操作を実行することも可能です。

- ・ デザイン全体に対して自動配置配線ツールを実行する前に、重要なコンポーネントを配置配線します。
- ・ コンポーネントの配置配線を手動で変更します。FPGA Editor を使用すると、配置配線を自動または手動で実行できます。

メモ： 詳細は、FPGA Editor ヘルプを参照してください。

ビットストリームの生成 (FPGA のみ)

FPGA では、BitGen コマンドライン プログラムによりザイリンクス デバイスのコンフィギュレーション用にビットストリームが生成されます。BitGen は配線済みの NCD を入力ファイルとして読み込み、コンフィギュレーション ビットストリーム (拡張子が .bit のバイナリファイル) を生成します。BIT ファイルには、FPGA の内部ロジックやインターコネクトを定義する NCD ファイルからのコンフィギュレーション情報に加え、ターゲット デバイスに関連するその他のファイルからのデバイス特有の情報が含まれています。詳細は、「[BitGen](#)」の章を参照してください。

BIT ファイルを生成した後、iMPACT を使用してデバイスにダウンロードできます。または、PROMGen を使用して BIT ファイルを PROM ファイルに変換してから、iMPACT を使用してデバイスにダウンロードすることも可能です。詳細は、このマニュアルの「[PROMGen](#)」の章または iMPACT ヘルプを参照してください。

デザイン検証

デザイン検証は、デザインの機能と性能をテストします。次の方法で、ザイリンクス デザインを検証できます。

- ・ 論理シミュレーションおよびタイミング シミュレーション
- ・ スタティック タイミング解析
- ・ インサーキット検証

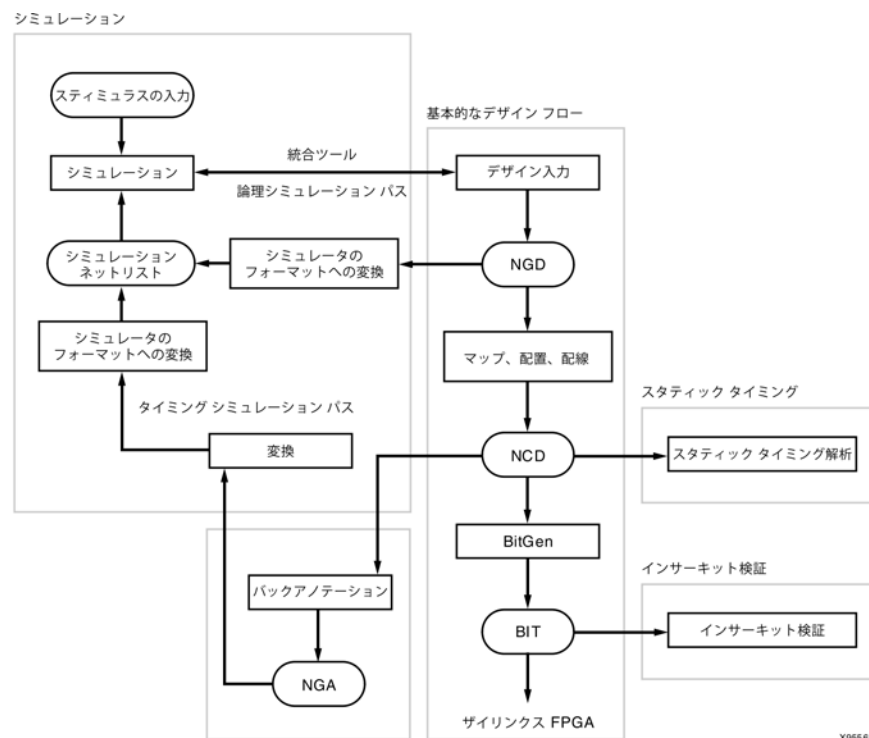
次に、各検証に使用するデザイン ツールを示します。

検証ツール

検証のタイプ	ツール
シミュレーション	サードパーティのシミュレータ (ザイリンクス ソフトウェアに統合されているもの、または統合されていないもの)
スタティック タイミング解析	TRACE (コマンドライン プログラム) Timing Analyzer (GUI) Mentor Graphics 社 TAU、Innoveda 社 BLAST ソフトウェア (STAMP ファイル フォーマット用、I/O タイミング検証のみ)
インサーキット検証	DRC (コマンドライン プログラム) ダウンロード ケーブル

次に示すように、デザイン検証はデザインプロセスのさまざまな段階で実行されます。

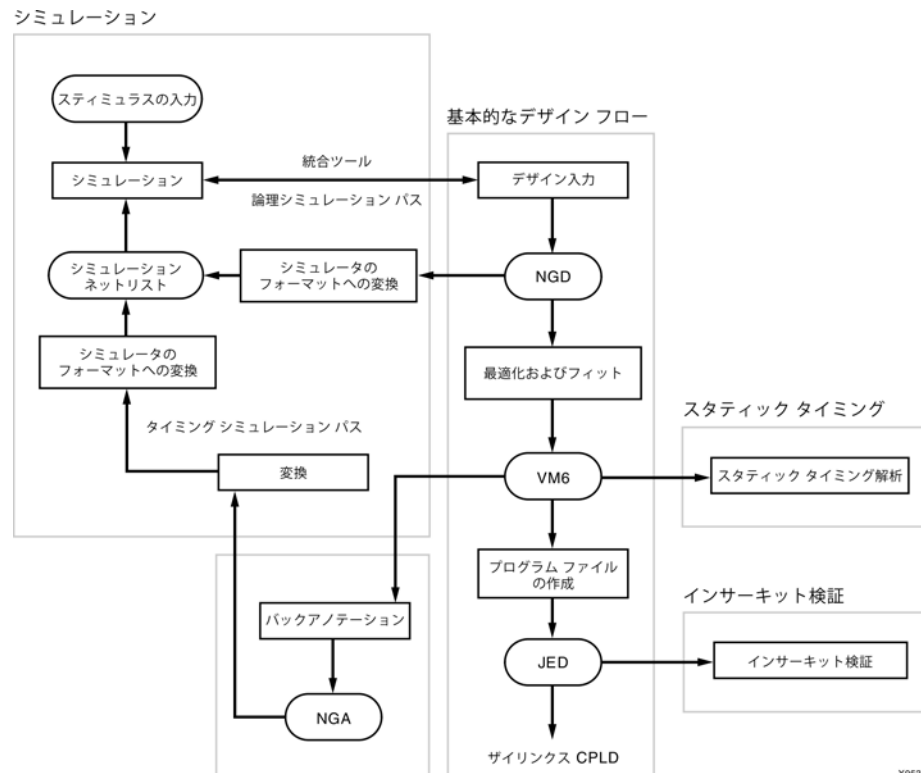
デザインフローの 3 つの検証方法 (FPGA)



次に、CPLD デザイン フローの検証方法を示します。

X9556

デザイン フローの 3 つの検証方法 (CPLD)



X953B

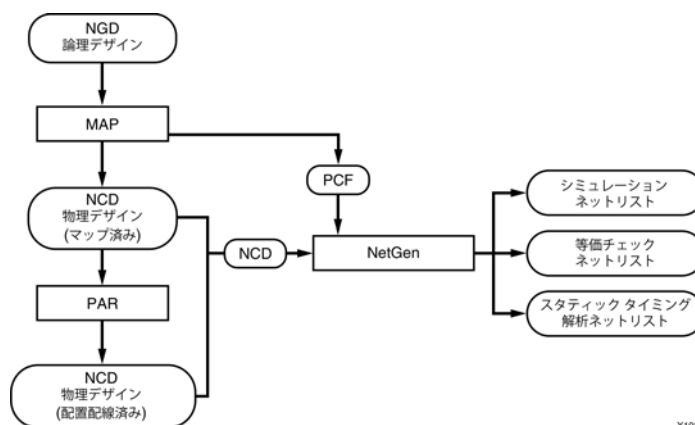
シミュレーション

論理シミュレーションまたはタイミング シミュレーションは、デザインを検証するために実行できます。このセクションでは、タイミング シミュレーションの前に必要なバックアノテーションについて説明します。また、回路図デザインおよび HDL ベースのデザインに対して、論理シミュレーションおよびタイミング シミュレーションを実行する方法についても説明します。

バックアノテーション

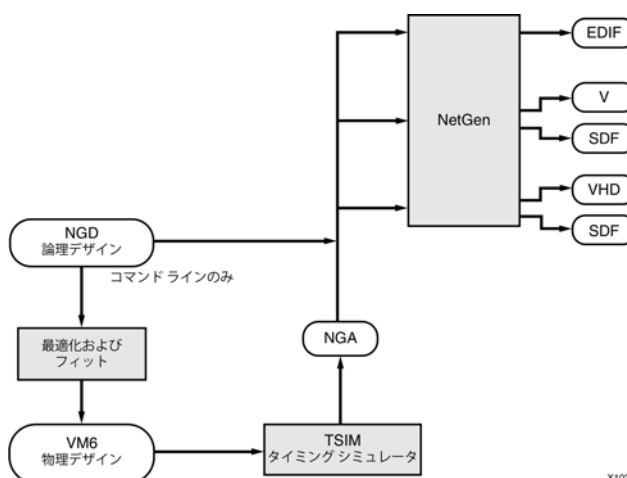
タイミング シミュレーションを実行する前に、物理デザイン情報を変換し、論理デザインに組み込む必要があります。バックアノテーションには、FPGA の場合は NetGen を、CPLD の場合は TSIM を使用します。これらのプログラムによりデータベースが作成され、バックアノテートされた情報がタイミング シミュレーションで使用されるネットリストフォーマットに変換されます。

バックアノテーション フロー (FPGA)



X10298

バックアノテーション フロー (CPLD)



X10297

NetGen

NetGen は、物理的な NCD ファイルの遅延、セットアップ/ホールド タイム、clock-to-out、パルス幅などの情報を論理的な NGD ファイルに渡し、タイミング シミュレーションや等価チェック、スタティック タイミング解析ツールで使用するための Verilog または VHDL ネットリストを生成します。

NetGen は、NCD ファイルを入力として読み込みます。マップのみが完了したデザイン、一部または完全に配置配線が終了したデザインの NCD ファイルを使用できます。MAP で生成される NGM ファイルも入力ファイルとして使用できます。NetGen は、オプションの NGM ファイルのマップ情報を NCD の配置、配線、タイミング情報と結合します。

メモ： CPLD デザインの場合、NGA ファイルが入力ファイルとして読み込まれ、タイミング シミュレーション ネットリストが生成されます。

詳細は、「[NetGen](#)」の章を参照してください。

論理シミュレーション

論理シミュレーションは、デバイスにデザインをインプリメントする前に、デザインの論理が正しいかどうかを確認するため実行します。論理シミュレーションは、デザイン フローの初期段階で実行できます。この段階では、インプリメントされたデザインのタイミング情報がいないため、ユニット遅延を使用してデザインの論理がテストされます。

メモ： 通常、デザイン フローの初期段階で論理シミュレーションを実行しておく、エラーを短時間で簡単に修正できます。

タイミング シミュレーション

タイミング シミュレーションは、ワーストケース条件下で指定されたスピードでデバイスが正しく動作するかどうかを検証します。タイミング シミュレーションは、FPGA の場合はデザインをマップ、配置、配線した後、CPLD の場合はフィットした後に実行します。この段階では、デザインの遅延はすべて既知になっています。

タイミング シミュレーションは、ワーストケース条件におけるデザインのタイミングを検証し、クリティカル パスを特定できるため重要です。また、デザインにセットアップ違反やホールド違反がないかどうか確認できます。

デザインをシミュレーションする前にバックアノテーションを実行する必要があります（前述の「バックアノテーション」を参照）。このプロセスで、個別のシミュレータに適合したフォーマットが作成されます。

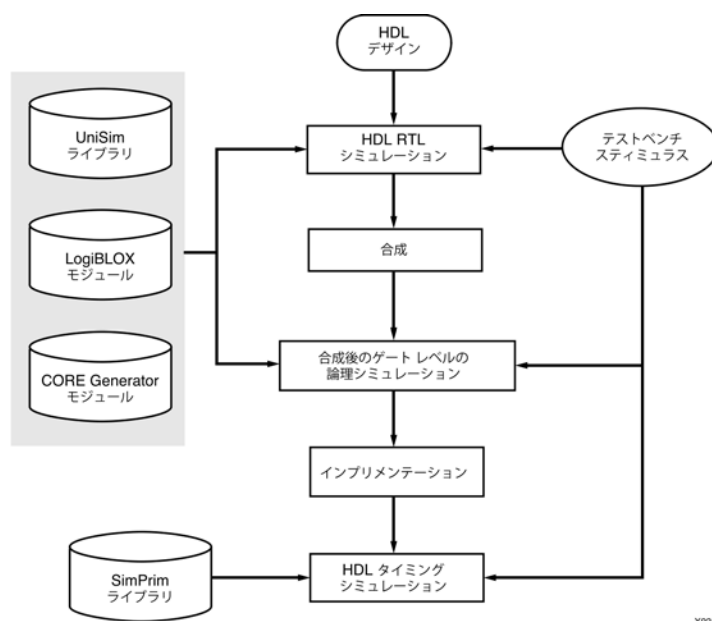
HDL ベースのシミュレーション

HDL デザインの場合、次の段階での論理シミュレーションおよびタイミング シミュレーションがサポートされます。

- ・ レジスタトランスファレベル(RTL) のシミュレーション。次のものが使用されます。
 - インスタンス化された UNISIM ライブラリ コンポーネント
 - CORE Generator™ モデル
 - ハード IP (SecureIP)
- ・ 合成後の論理シミュレーション。次のいずれかが使用されます。
 - ゲートレベルの UNISIM ライブラリ コンポーネント
 - CORE Generator モデル
 - ハード IP (SecureIP)
- ・ インプリメンテーション後にバックアノテーションして実行するタイミングシミュレーション。次のものが使用されます。
 - SIMPRIM ライブラリ コンポーネント
 - ハード IP (SecureIP)
 - 標準遅延フォーマット (SDF) ファイル

次に、論理シミュレーションおよびタイミング シミュレーションを実行する段階を示します。

HDL デザインのシミュレーション ポイント



3 つの主要なシミュレーション ポイントに加え、合成後に 2 つのシミュレーションを実行できます。これらのシミュレーションは、合成ツールで VHDL または Verilog を生成できない場合や、ネットリストが UNISIM コンポーネントレベルでない場合に使用します。次に、HDL デザイン フローで実行可能なすべてのシミュレーション ポイントを示します。

HDL デザイン フローにおける 5 つのシミュレーション ポイント

シミュレーション	UNISIM	SIMPRIM	SDF
RTL	X		
合成後	X		
NGDBuild 後の論理シミュレーション (オプション)		X	
MAP 後の論理シミュレーション (オプション)		X	X
配線後のタイミングシミュレーション		X	X

これらのシミュレーション ポイントについては、『[合成/シミュレーション デザイン ガイド](#)』の「HDL デザインフローのシミュレーションポイント」を参照してください。

シミュレーション フローに必要なライブラリについては、『[合成/シミュレーション デザイン ガイド](#)』の「VHDL および Verilog のライブラリとモデル」を参照してください。フローおよびライブラリでは、論理シミュレーションとタイミングシミュレーションの初期化動作が機能的にほぼ同じです。これは、グローバル セット/リセット (GSR) およびグローバル トライステート (GTS) の動作をシミュレーションするのに、別の手法とライブラリ セルを追加したためです。

ザイリンクスの VHDL シミュレーションは、VITAL 規格に対応しています。この規格では、VITAL に準拠したシミュレータでのシミュレーションが可能です。ビルトイン Verilog がサポートされているため、Cadence 社 Verilog-XL などの互換性のあるシミュレータを使用したシミュレーションも可能です。ザイリンクスの HDL シミュレーションでは、ザイリンクス FPGA および CPLD デバイスがすべてサポートされています。サポートされる VHDL 規格と Verilog 規格については、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

スタティック タイミング解析 (FPGAのみ)

スタティック タイミング解析では、デザインのパス遅延を確認できます。スタティック タイミング解析では、主に次の 2 つの処理を実行します。

- ・ タイミング検証

デザインがタイミング制約を満たしているかどうかを検証します。

- ・ レポート

制約違反をファイルに出力します。部分的または完全に配置配線されたデザインを解析できます。出力されるタイミング情報は、入力デザインの配置配線の状態によって異なります。

スタティック タイミング解析は、TRACE (Timing Reporter and Circuit Evaluator) コマンドラインプログラムを使用して実行できます。詳細は、『[TRACE](#)』の章を参照してください。Timing Analyzer でもスタティック タイミング解析を実行できます。詳細は、Timing Analyzer ヘルプを参照してください。これらのツールを使用して、配置配線後のデザインでタイミング制約が満たされているかどうかを評価します。

インサーキット検証

最終テストとして、デザインが実際のアプリケーションでどのように動作するかを検証します。インサーキット検証では、一般的な動作条件で回路をテストします。FPGA デバイスは繰り返しプログラムが可能のため、何度でもデザインを変更してデバイスに読み込み、インサーキットでテストすることが可能です。デザインをインサーキットで検証するには、ザイリンクス ケーブルを使用し、デザイン ビットストリームをデバイスにダウンロードします。

メモ： ザイリンクスのケーブルとハードウェアについては、iMPACT ヘルプを参照してください。

デザイン ルール チェック (FPGA のみ)

最終的なビットストリームを生成する前に、BitGen で DRC オプションを使用して NCD ファイルを評価し、デザインが正常に動作するかどうかを確認することが重要です。DRC は、**-d** オプションを使用しなければ自動的に実行されます。詳細は、『[物理的デザイン ルール チェック](#)』の章および『[BitGen](#)』の章を参照してください。

プローブ

FPGA Editor のプローブ機能では、いくつかの信号を同時に解析できるリアルタイム デバッグが可能です。この機能を使用すると、内部信号と使用可能な I/O ピンをすばやく特定して配線できるので、デザインを配置配線し直す必要はありません。信号のリアルタイムの動作は、ロジック アナライザ、ステート アナライザ、オシロスコープなどのテスト装置で確認できます。

ChipScope ILA および ChipScope Pro

ChipScope™ ツールセットは、PC ボード レベルでの作業を支援するために開発されました。ChipScope ILA は、デザインにロジック アナライザ コアを組み込みます。これらのロジック コアを使用することにより、FPGA の内部信号およびノードをすべて表示できます。トリガの変更は、ユーザーのロジックに影響を与えずに行うことができ、デザインを再コンパイルする必要もありません。

FPGA デザインのヒント

ザイリンクス FPGA アーキテクチャは、同期デザインに適しています。厳密な同期デザインにより、すべてのレジスタがクロック スキューなしに同じタイム ベースで駆動されます。このセクションでは、高性能な同期デザインを作成するためのヒントを示します。

デザインのサイズとパフォーマンス

デザインのサイズとパフォーマンスに関する情報は、デザインの最適化に有益です。デザインの配置配線を実行して生成されるレポート ファイルには、使用可能な CLB、IOB、その他のデバイス リソースの数がリストされます。MAP プログラムでデザインを処理すると、最初の概算値が得られます。

自動インプリメンテーション ソフトウェアを実行せずにデザインのサイズとパフォーマンスを判断する場合は、ザイリンクス FPGA アーキテクチャに基づいて概算すると簡単に予測できます。

PARTGen

この章では、PARTGen について説明します。次のセクションが含まれています。

- ・ [PARTGen の概要](#)
- ・ [PARTGen の構文](#)
- ・ [PARTGen のオプション](#)

PARTGen の概要

PARTGen は、サポートされているザイリンクス デバイスの詳細を表示するコマンドライン ツールです。

デバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

PARTGen の入力ファイル

入力ファイルは必要ありません。

PARTGen の出力ファイル

PARTGen の出力ファイルには、次の 2 種類があります。

- ・ [パーツリスト ファイル](#) (ASCII および XML)
- ・ [パッケージ ファイル](#) (ASCII)

パーツリスト ファイル

PARTGen のパーツリスト ファイルは、サポートされる合成ツールも含め、アーキテクチャとデバイスに関する詳細情報を表示します。パーツリスト ファイルは、ASCII (.xct) と XML (.xml) の 2 種類の形式で生成されます。

PARTGen を [-p \(パーツリスト ファイルとパッケージ ファイルの生成\)](#) または [-v \(詳細なパーツ リスト ファイルとパッケージ ファイルの生成\)](#) オプションを使用して実行すると、XML 形式のパーツリスト ファイルが生成されます。別のコマンドライン オプションは必要ありません。

パーツリスト ファイルは、パーツのリストです。インストールされているソフトウェアでサポートされている各パーツに対して、1 つのエントリが作成されます。次のセクションで、パーツファイルに含まれる情報について説明します。

- ・ パーツリスト ファイルのヘッダ
- ・ PARTGen の **-p** および **-v** オプションに共通するパーツリスト ファイルのデバイス属性
- ・ PARTGen の **-v** オプションでのみ表示されるパーツリスト ファイルのデバイス属性

パーツリスト ファイルのヘッダ

パーツリスト ファイルの最初の部分は、エントリのヘッダです。

```
part architecture family partname diename packagefilename
```

XC6VLX550TFF1759 デバイスのパーツファイル ヘッダ例

```
partVIRTEX XC6VLX550Tff1759 NA.die xc6vlx550tff1759.pkg
```

PARTGen の **-p** および **-v** オプションに共通するパーツリスト ファイルのデバイス属性

次のデバイス属性は、PARTGen の **-p** オプションと **-v** オプションの両方で表示されます。

- ・ CLB の行と列のサイズ
NCLBROWS=# NCLBCOLS=#
- ・ サブファミリの指定
STYLE=sub_family (STYLE = Virtex6 など)
- ・ 入力レジスタ数
IN_FF_PER_IOB=#
- ・ 出力レジスタ数
OUT_FF_PER_IOB=#
- ・ 行および列ごとのパッド数
NPADS_PER_ROW=# NPADS_PER_COL=#
- ・ ビットストリーム情報
 - フレーム数 : NFRAMES=#
 - フレームごとのビット数 : NBITSPERFRAME=#
- ・ サポートされるステッピング レベル : STEP=#
- ・ I/O 規格

各 I/O 規格のすべてのプロパティが解析可能なフォーマットで表示されるようになりました。これにより、サードパーティ ツールで I/O バンクのデザイン ルール チェック (DRC) を実行できます。

使用可能な各 I/O 規格に対して、partlist.xct および partlist.xml ファイルに次の情報が表示されます。

```
IOSTD_NAME: LVTTL \
  IOSTD_DRIVE: 12 2 4 6 8 16 24 \
  IOSTD_SLEW: SLOW FAST \
  IOSTD_DIRECTION: INPUT=1 OUTPUT=1 BIDIR=1 \
```

```

IOSTD_INPUTTERM: NONE \
IOSTD_OUTPUTTERM: NONE \
IOSTD_VCCO: 3.300000 \
IOSTD_VREF: 100.000000 \
IOSTD_VRREQUIRED: 0 \
IOSTD_DIFFTERMREQUIRED: 0 \

```

IOSTD_DRIVE および IOSTD_SLEW では、最初にリストされている値がデフォルト値です。true/false 値は、次のように示されます。

- **1** : true を示します。
- **0** : false を示します。

IOSTD_VREF の 100.000000 という値は、この規格ではこのキーワードが定義されていないことを示します。

SO および WASSO の算出

PARTGen で、I/O 規格とデバイス プロパティをマシンで読み出し可能なフォーマットにエクスポートできます。これにより、サードパーティツールで SSO および WASSO を算出できます。

SSO データは、次の 2 つの部分で構成されます。

- 電源/グラウンド ペアごとに許容される SSO の最大数
- バンクに含まれる電源/グラウンド ペアの数

各デバイス/パッケージの組み合わせに対して、partlist.xct および partlist.xml ファイルにこの情報が表示されます。電源/グラウンド ペアの場合は、バンクごとに表示されます。

```

PER_BANK_PWRGND_PAIRS\
  BANK_SSO NAME=0 TYPE=INT 1\
  BANK_SSO NAME=1 TYPE=INT 1\
  BANK_SSO NAME=2 TYPE=INT 1\
  BANK_SSO NAME=3 TYPE=INT 1\
  BANK_SSO NAME=4 TYPE=INT 1\
  BANK_SSO NAME=5 TYPE=INT 5\
  BANK_SSO NAME=6 TYPE=INT 5\
  BANK_SSO NAME=7 TYPE=INT 3\
  BANK_SSO NAME=8 TYPE=INT 3\

```

電源/グラウンド ペアごとに許容される SSO の最大数は、SSO_PER_IOSTD キーワードを使用して表示されます。2 列目に駆動電流、3 列目に IO 規格、4 列目にスルー レート、6 列目に許容される SSO の最大数が示されます。

たとえば、LVTTTL、駆動電流 12、スルー レートが SLOW の場合、電源/グラウンド ペアごとに許容される最大の SSO 数は 15 です。

```

MAX_SSO_PER_IOSTD_PER_BANK\
  IOSTD_SSO DRIVE=12 NAME=LVTTTL SLEW=SLOW TYPE=INT 15\
  IOSTD_SSO DRIVE=12 NAME=LVTTTL SLEW=FAST TYPE=INT 10\
  IOSTD_SSO DRIVE=2 NAME=LVTTTL SLEW=SLOW TYPE=INT 68\
  IOSTD_SSO DRIVE=2 NAME=LVTTTL SLEW=FAST TYPE=INT 40\
  IOSTD_SSO DRIVE=4 NAME=LVTTTL SLEW=SLOW TYPE=INT 41\
  IOSTD_SSO DRIVE=4 NAME=LVTTTL SLEW=FAST TYPE=INT 24\

```

```

IOSTD_SSO DRIVE=6 NAME=LVTTL SLEW=SLOW TYPE=INT 29\
IOSTD_SSO DRIVE=6 NAME=LVTTL SLEW=FAST TYPE=INT 17\
IOSTD_SSO DRIVE=8 NAME=LVTTL SLEW=SLOW TYPE=INT 22\
IOSTD_SSO DRIVE=8 NAME=LVTTL SLEW=FAST TYPE=INT 13\
IOSTD_SSO DRIVE=16 NAME=LVTTL SLEW=SLOW TYPE=INT 11\
IOSTD_SSO DRIVE=16 NAME=LVTTL SLEW=FAST TYPE=INT 8\
IOSTD_SSO DRIVE=24 NAME=LVTTL SLEW=SLOW TYPE=INT 7\
IOSTD_SSO DRIVE=24 NAME=LVTTL SLEW=FAST TYPE=INT 5\

```

- ・ デバイスのグローバル、ローカル、リージョナル クロック プロパティ

デバイス上の各クロック タイプに対し、次の情報が表示されるようになりました。

- クロック タイプで使用可能なピン番号
- クロック ピンで駆動可能な I/O

これにより、サードパーティツールでクロックの配置規則に従ってザイリンクス パッケージにピンを割り当てることができます。

デバイスの各領域に対して、partlist.xct および partlist.xml ファイルに次の情報が表示されます。

```

DEVICE_CLKRGN\
  NUM_CLKRGN TYPE=INT 8\
  NUM_CLKRGN_ROW TYPE=INT 4\
  NUM_CLKRGN_COL TYPE=INT 2\
    CLKRGN TYPE=STRING X0Y0\
  CLK_CAPABLE_SCOPE\
  UNASSOCIATED_PINS\
  NUM_UNBONDED_PINS TYPE=INT 2\
  UNBONDED_PIN_LIST TYPE=STRINGLIST T17R17\
  UNBONDED_IOB_LIST TYPE=STRINGLIST IOB_X0Y15IOB_X0Y17\
  ASSOCIATED_BUFIO\
  NUM_BUFIO TYPE=INT 4\
  BUFIO_SITES TYPE=STRINGLIST BUFIO_X0Y0BUFIO_X0Y1BUFIO_X1Y0BUFIO_X1Y1\
  ASSOCIATED_BUFR\
  NUM_BUFR TYPE=INT 2\
  BUFR_SITES TYPE=STRINGLIST BUFR_X0Y0BUFR_X0Y1\
  ASSOCIATED_PINS\
  NUM_BONDED_PINS TYPE=INT 39\
  BONDED_PIN_LIST TYPE=STRINGLIST V18V17W17Y17W19W18U17U16V20V19U15T15U19U18T18\
    T17R18R17T20T19R16R15R20R19W8W9Y9Y10W7Y7W10W11W6Y6Y11Y12W5Y5W12\
  BONDED_IOB_LIST TYPE=STRINGLIST IOB_X0Y0IOB_X0Y1IOB_X0Y2IOB_X0Y3IOB_X0Y4IOB_X0Y5IOB_\
    X0Y6IOB_X0Y7IOB_X0Y8IOB_X0Y9IOB_X0Y10IOB_X0Y11IOB_X0Y12IOB_X0Y13IOB_X0Y14IOB_\
    X0Y15IOB_X0Y16IOB_X0Y17IOB_X0Y18IOB_X0Y19IOB_X0Y22IOB_X0Y23IOB_X0Y24IOB_X0Y25IOB_\
    X1Y16IOB_X1Y17IOB_X1Y18IOB_X1Y19IOB_X1Y20IOB_X1Y21IOB_X1Y22IOB_X1Y23IOB_X1Y24IOB_\
    X1Y25IOB_X1Y26IOB_X1Y27IOB_X1Y28IOB_X1Y29IOB_X1Y30\

```

PARTGen の -v オプションでのみ表示されるパーツリスト ファイルのデバイス属性

次のデバイス属性は、**-v** オプションを指定した場合にのみ表示されます。

- ・ デバイスの IOB 数

NIOBS=#

- ・ ボンディングされた IOB 数

NBIOBS=#

- ・ CLB のスライス数 : SLICES_PER_CLB=#
スライス ベースのアーキテクチャの場合のみ (スライス ベースでないアーキテクチャの場合は、CLB ごとに 1 スライスと想定)
- ・ スライスのフリップフロップ数
FFS_PER_SLICE=#
- ・ スライスのラッチ
CAN_BE_LATCHES={TRUE|FALSE}
- ・ DCM、PLL、または MMCM の数
- ・ スライスの LUT : LUT_NAME=name LUT_SIZE=#
- ・ グローバル バッファ数 : NUM_GLOBAL_BUFFERS=#
(バッファがグローバル クロックの組み合わせを駆動できる場所の数)
- ・ ブロック RAM
NUM_BLK_RAMs=# BLK_RAM_COLS=# BLK_RAM_COL0=# BLK_RAM_COL1=#
BLK_RAM_COL2=# BLK_RAM_COL3=# BLK_RAM_SIZE=4096x1 BLK_RAM_SIZE=2048x2
BLK_RAM_SIZE=512x8 BLK_RAM_SIZE=256x16
ブロック RAM の位置は、CLB の列で決まります。次の例では、ブロック RAM 5 が CLB の列 32 に配置されています。
NUM_BLK_RAMs=10 BLK_RAM_COL_5=32 BLK_RAM_SIZE=4096X1
- ・ SelectRAM
NUM_SEL_RAMs=# SEL_RAM_SIZE=#X#
- ・ デュアル ポート SelectRAM
SEL_DP_RAM={TRUE|FALSE}
SelectRAM をデュアル ポート RAM として使用可能かどうかを示します。SelectRAM をデュアル ポート RAM として使用すると、アドレス可能なエレメントの数が半分に減少します。つまり、デュアル ポート モードでの SelectRAM のサイズは、SEL_RAM_SIZE で表すサイズの半分です。
- ・ スピード グレード情報 : SPEEDGRADE=#
遅延情報は、XCT および XML パーツリスト ファイルには含まれません。遅延情報は、Speedprint を使用すると表示されます。詳細は、「[Speedprint](#)」の章を参照してください。
- ・ スライスごとに構成できる LUT の最大幅
MAX_LUT_PER_SLICE=# (スライス内のすべての LUT)
- ・ CLB ごとに構成できる LUT の最大幅 : MAX_LUT_PER_CLB=#
スライスで使用可能な LUT を使用して CLB で構成できる LUT の幅を表します。
- ・ デバイスの内部トライステート バッファ数
NUM_TBUFS PER ROW=#
- ・ デバイスまたはパッケージで使用可能な場合は、次の情報が表示されます。
NUM_PPC=#
NUM_GT=#
NUM_MONITOR=#
NUM_DPM=#
NUM_PMCd=#

```

NUM_DSP=#
NUM_FIFO=#
NUM_EMAC=#
NUM_MULT=#

```

パッケージ ファイル

PARTGen のパッケージ ファイルは、IOB と出力ピン名を関連付ける ASCII 形式のファイルで、パッケージのピンに関する情報を XACT パッケージ フォーマットで示します。**-p** オプションを使用するとピン情報が 3 列で表示され、**-v** オプションを使用するとさらに 6 列が追加されます。次のセクションで、パーツファイルに含まれる情報について説明します。

- ・ PARTGen の **-p** オプションを使用した場合のパッケージ ファイル
- ・ PARTGen の **-v** オプションを使用した場合のパッケージ ファイル

PARTGen の **-p** オプションを使用した場合のパッケージ ファイル

-p オプションを使用すると、3 つの項目のピン情報を示すパッケージ ファイルが生成されます。次の表に、各項目の詳細を示します。

パッケージ ファイルの列の説明

列	内容	説明
1	pin (使用可能なピン) または pkgpin (専用ピン)	使用可能なピンであるか (pin)、専用ピンであるか (pkgpin) を示します。
2	ピン名	ピン名を示します。使用可能なピンの場合、デバイスの IOB に接続された、ボンディングされているパッドの名前、または多目的ピンの名前がピン名になります。専用ピンの場合は、ピンの機能を示す名前か、コネクタなし (No Connection) の意味で「N.C.」と表示されます。
3	パッケージ ピン	パッケージ ピンを指定します。

たとえば、「**partgen -p xc6vlx75t**」というコマンドを入力すると、次のパッケージ ファイルが生成されます。

- ・ xc6vlx75tff484.pkg
- ・ xc6vlx75tff784.pkg

-p オプションを使用した場合のパッケージ ファイル例

次に、**-p** オプションを使用した場合に生成される xc6vlx75tff484 パッケージのパッケージ ファイルの内容を示します。

```

package xc6vlx75tff484
pin IPAD_X1Y25 G3
pin IPAD_X0Y31 M11
pin IOB_X0Y39 M18
.
.
.

```

PARTGen の -v オプションを使用した場合のパッケージ ファイル

-v オプションを使用すると、9 つの項目のピン情報を示すパッケージ ファイルが生成されます。次の表に、各項目の詳細を示します。

パッケージ ファイルの列の説明

列	内容	説明
1	pin (使用可能なピン) または pkgpin (専用ピン)	使用可能なピンであるか (pin)、専用ピンであるか (pkgpin) を示します。
2	ピン名	ピン名を示します。使用可能なピンの場合、デバイスの IOB に接続された、ボンディングされているパッドの名前、または多目的ピンの名前がピン名になります。専用ピンの場合は、ピンの機能を示す名前か、コネクタなし (No Connection) の意味で「N.C.」と表示されます。
3	パッケージ ピン	パッケージ ピンを指定します。
4	VREF BANK	関連付けられたバンクを表す正の整数値か、関連付けられたバンクがない場合は「1」と表示されます。
5	VCCO BANK	関連付けられたバンクを表す正の整数値か、関連付けられたバンクがない場合は「1」と表示されます。
6	機能	ピンの使用法を示す文字列です。専用ピンの場合は、特定の機能を示す文字列が表示されます。ピンが汎用のユーザー ピンの場合は、「IO」という文字列が表示されます。多目的ピンの場合は、アンダースコア () で区切った文字列が表示されます。
7	CLB	ピンに最も近接した CLB の行と列が、R[0-9]C[0-9] または x[0-9]y[0-9] という形式で表示されます。
8	LVDS IOB	LVDS IOB が設定されているピンに対して表示される文字列です。この文字列は、インデックスと M または S の英文字から構成されます。インデックス値は、0 から LVDS ペア数の間の数です。LVDS ではないピンは、「N.A.」と表示されます。
9	フライト タイム データ	フライト タイム (遅延データ) をマイクロ単位で表します。データがない場合は、「N.A.」と表示されます。

-v オプションを使用した場合のパッケージ ファイル例

次に、**-v** オプションを使用した場合に生成されるパッケージ ファイルの例を示します。

```
package xc6v1x75tff484
# PartGen L.44
#      pad          pin      vref vcco  function      nearest diff.  tracelength
#      name         name    bank bank  name          CLB    pair    (um)
pin    IPAD_X1Y25    G3      -1  -1  MGTRXP0_115    N.A.    N.A.    8594
pin    IPAD_X0Y31    M11     0   0   VN_0          N.A.    N.A.    1915
pin    IOB_X0Y39     M18    14  14   IO_L0P_14     X0Y38    0M    4111
pin    IOB_X0Y38     N18    14  14   IO_L0N_14     X0Y38    0S    3390
```

PARTGen の構文

PARTGen のコマンド ライン構文は、次のとおりです。

partgen *options*

options : 「PARTGen のオプション」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

パーツリスト ファイルおよびパッケージ ファイルは、**-p** オプションおよび **-v** オプションを使用すると生成できます。

- ・ **-p** オプションを使用すると、3 項目のピン情報を示すファイルが生成されます。
- ・ **-v** オプションを使用すると、さらに 6 項目が追加されます。

PARTGen のオプション

このセクションでは、PARTGen のコマンド ライン オプションについて説明します。

- ・ **-arch** (アーキテクチャの情報の表示)
- ・ **-i** (デバイス、パッケージ、スピードの一覧の表示)
- ・ **-intstyle** (統合スタイル)
- ・ **-nopkgfile** (パッケージ ファイルを生成しない)
- ・ **-p** (パーツリスト ファイルとパッケージ ファイルの生成)
- ・ **-v** (詳細なパーツリスト ファイルとパッケージ ファイルの生成)

-arch (アーキテクチャの情報の表示)

指定のアーキテクチャについて、デバイス、パッケージ、スピードの一覧を表示します。

構文

-arch *architecture_name*

architecture_name の値

- ・ acr2 (オートモーティブ CoolRunner™-II)
- ・ aspartan3 (オートモーティブ Spartan®-3)
- ・ aspartan3a (オートモーティブ Spartan-3A)
- ・ aspartan3adsp (オートモーティブ Spartan-3A DSP)
- ・ aspartan3e (オートモーティブ Spartan-3E)
- ・ aspartan6 (オートモーティブ Spartan-6)
- ・ qrvirtex4 (QPro™ Virtex®-4 Rad Tolerant)
- ・ qvirtex4 (QPro Virtex-4 Hi-Rel)
- ・ qvirtex5 (QPro Virtex-5 Hi-Rel)
- ・ qspartan6 (QPro Spartan-6 Hi-Rel)
- ・ qvirtex6 (QPro Virtex-6 Hi-Rel)
- ・ spartan3 (Spartan-3)
- ・ spartan3a (Spartan-3A)
- ・ spartan3adsp (Spartan-3A DSP)
- ・ spartan3e (Spartan-3E)
- ・ spartan6 (Spartan-6)
- ・ virtex4 (Virtex-4)
- ・ virtex5 (Virtex-5)
- ・ virtex6 (Virtex-6)
- ・ virtex6l (Virtex-6 低消費電力)
- ・ xa9500xl (オートモーティブ XC9500XL)
- ・ xbr (CoolRunner-II)
- ・ xc9500 (XC9500)
- ・ xc9500xl (XC9500XL)
- ・ xpla3 (CoolRunner XPLA3)

-i (デバイス、パッケージ、スピードの一覧の表示)

インストールされている各デバイス、パッケージ、スピードのリストを表示します。

構文

-i

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle ise|xflow|silent

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-nopkgfile (パッケージ ファイルを生成しない)

-p オプションまたは **-v** オプションを指定している場合に、パッケージ ファイルを生成しないようにします。

構文

-nopkgfile

-p (パーツリスト ファイルとパッケージ ファイルの生成)

次のファイルを生成します。

- ・ ASCII (.xct) および XML (.xml) 形式のパーツリスト ファイル
- ・ ASCII (.pkg) 形式のパッケージ ファイル

構文

-p name

name には、次のものを指定できます。

- ・ アーキテクチャ
- ・ デバイス
- ・ パーツ

ファイルはすべて、作業ディレクトリに保存されます。

このオプションの後にアーキテクチャ、デバイス、またはパーツを指定しない場合は、インストールされているすべてのデバイスの詳細情報が `partlist.xct` ファイルに書き込まれます。詳細は、「[パーツリスト ファイル](#)」を参照してください。

-p オプションで出力される情報は、**-arch** オプションの出力よりも詳細ですが、**-v** オプションよりは簡略されたものになります。**-p** オプションと **-v** オプションを同時に使用することはできません。どちらか一方を使用するようにしてください。詳細は、次のセクションを参照してください。

- ・ [パッケージ ファイル](#)
- ・ [パーツリスト ファイル](#)

partgen -p オプションの例

name	コマンド ライン入力例
アーキテクチャ	-p virtex5
デバイス	-p xc5vlx110t
パーツ	-p xc5vlx110tff1136

-v (詳細なパーツリスト ファイルとパッケージ ファイルの生成)

次のファイルを生成します。

- ・ ASCII (.xct) および XML (.xml) 形式のパーツリスト ファイル
- ・ ASCII (.pkg) 形式のパッケージ ファイル

構文

-v *name*

name には、次のものを指定できます。

- ・ アーキテクチャ
- ・ デバイス
- ・ パーツ

アーキテクチャ、デバイス、パーツを指定しない場合、インストールされているすべてのデバイスの詳細情報がパーツリスト ファイルに出力されます。詳細は、「[パーツリスト ファイル](#)」を参照してください。

-v オプションを使用すると、**-p** オプションよりも詳細な情報が生成されます。**-p** オプションと **-v** オプションを同時に使用することはできません。どちらか一方を使用するようにしてください。詳細は、次のセクションを参照してください。

- ・ [パッケージ ファイル](#)
- ・ [パーツリスト ファイル](#)

partgen -v オプションの例

name	コマンド ライン入力例
アーキテクチャ	partgen -v virtex6
デバイス	partgen -v xc5vlx110t
パーツ	partgen -v xc5vlx110tff1136

NetGen

こので章は、NetGen について説明します。次のセクションが含まれています。

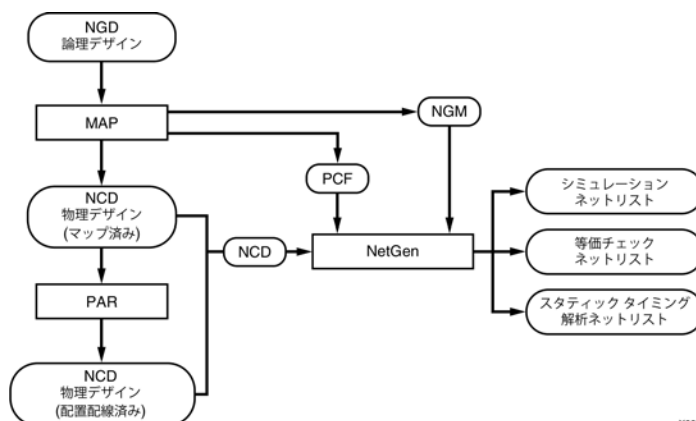
- ・ NetGen の概要
- ・ NetGen のシミュレーション フロー
- ・ NetGen の等価チェック フロー
- ・ NetGen のスタティック タイミング解析フロー
- ・ 階層ファイルの保持および出力
- ・ バックアノテーション シミュレーションでの専用グローバル信号の処理

NetGen の概要

NetGen は、デザイン ファイルのデータを読み込んで、シミュレーションや等価チェック、スタティック タイミング解析に使用する、サードパーティ シミュレーション用のネットリストを生成するコマンドライン プログラムです。

NetGen は、インプリメントされたデザイン ファイルを読み込み、デザイン全体のネットリストを 1 つ出力します。階層デザインの場合は、モジュールごとにネットリストを出力します。個別のモジュールは、そのみでシミュレーションするか、最上位でまとめてシミュレーションできます。KEEP_HIERARCHY 属性が指定されたモジュールは、[-mhf \(複数の階層ファイルを出力\)](#) オプションを使用すると、KEEP_HIERARCHY 制約で定義されたモジュールがユーザー指定の Verilog、VHDL、または SDF ネットリストに出力されます。詳細は、「[階層ファイルの保持および出力](#)」を参照してください。

NetGen のフロー



NetGen でサポートされるフローには、大きく分けてシミュレーション (論理シミュレーションとタイミングシミュレーション)、等価チェック、スタティック タイミング解析の 3 種類があります。この章には、各 NetGen フローの使用法、機能、およびサブフローを詳細に説明するセクションがあります。たとえば、シミュレーション フローには論理シミュレーションとタイミングシミュレーションがあります。

各セクションには、各 NetGen フローのコマンドライン構文、入力ファイル、出力ファイル、コマンドライン オプションが含まれます。

NetGen の構文は、実行するフロー タイプによって異なります。NetGen のフローと構文については、該当するセクションを参照してください。

使用できるネットリスト フローは次のとおりです。

- ・ **シミュレーション (-sim)** : 論理シミュレーションおよびタイミング シミュレーション用のネットリストを生成します。このタイプでは、**-ofmt** オプションを使用して、出力ファイルのタイプを Verilog または VHDL に指定する必要があります。

netgen -sim [options]

- ・ **等価チェック (-ecn)** : 等価チェック用の Verilog ネットリストを生成します。このタイプでは、**-ecn** オプションの後にツール名を指定する必要があります。指定可能なツール名は、**conformal** または **formality** です。

netgen -ecn tool_name [options]

- ・ **スタティック タイミング解析 (-sta)** : スタティック タイミング解析用のネットリストを生成します。

netgen -sta [options]

NetGen では、次のフローがサポートされます。

- ・ FPGA および CPLD デザインの論理シミュレーション
- ・ FPGA および CPLD デザインのタイミング シミュレーション
- ・ FPGA デザインの等価チェック
- ・ FPGA デザインのスタティック タイミング解析

NetGen では、入力デザイン ファイル (NGC、NGD、NCD) に基づいてフローが実行されます。次に、各入力ファイルに対応する出力ファイルのタイプを示します。

NetGen の出力ファイル

入力ファイル	出力ファイル
NGC	UNISIM ベースの論理シミュレーション ネットリスト
NGD	SIMPRIM ベースの論理シミュレーション ネットリスト
NGA (CPLD の場合)	SIMPRIM ベースのネットリスト (完全なタイミング SDF ファイルを使用)
NCD (MAP からの出力)	SIMPRIM ベースのネットリスト (部分的なタイミング SDF ファイルを使用)
NCD (PAR からの出力)	SIMPRIM ベースのネットリスト (完全なタイミング SDF ファイルを使用)

NetGen のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

NetGen シミュレーション フロー

NetGen のシミュレーション フローには、論理シミュレーションとタイミング シミュレーションという 2 種類のサブフローがあります。論理シミュレーションでは、NGC を入力ファイルとして読み込むと UNISIM ベースのネットリストが、NGD を読み込むと SIMPRIM ベースのネットリストが生成されます。同様に、タイミング シミュレーションも MAP 後のタイミング シミュレーションと PAR 後のタイミング シミュレーションの 2 種類に分類されますが、いずれのサブフローでも SIMPRIM ベースのネットリストが生成されます。

メモ： NGD ファイルを入力ファイルとして使用した場合、LOC パラメータがリストされません。この場合、LOC パラメータのデフォルト値として「UNPLACED」がレポートされます。

NetGen のシミュレーション フローまたはサブフローで使用するオプションを参照するには、コマンドラインに「**netgen -h sim**」と入力します。

NetGen の論理シミュレーション フロー

論理シミュレーション フローは、NGC または NGD ファイルを Verilog または VHDL ネットリストに変換するために使用します。

NetGen のコマンドラインで NGC を入力ファイルとして指定すると、論理シミュレーション フローが実行され、UNISIM ベースのネットリストが生成されます。NGD ファイルを指定すると、論理シミュレーション フローが実行され、SIMPRIM ベースのネットリストが生成されます。このとき、生成されるネットリストのタイプを Verilog または VHDL のいずれかに指定しておく必要があります。

論理シミュレーション フローに使用する入力ファイルは次のとおりです。

- ・ **NGC ファイル：** XST により生成され、UNISIM ベースのネットリストの生成に使用されます。UNISIM ベースのネットリストは、IP コアと共に使用し、合成後の論理シミュレーションを実行するのに適しています。
- ・ **NGD ファイル：** デザインの論理記述が含まれたこのファイルは、NGDBuild により生成され、SIMPRIM ベースのネットリストの生成に使用されます。

UNISIM ベースのネットリスト用の論理シミュレーション

XST を使用する場合、コマンドラインで NGC を入力ファイルとして指定できます。サードパーティの合成ツールを使用する場合は、まず ngdbuild コマンドを使用してすべてのネットリストを 1 つの NGC ファイルに変換する必要があります。NetGen では、NGC が入力ファイルとして使用されます。

最上位の EDIF ネットリストを NGC ファイルに変換するには、次のコマンドを使用します。

```
ngcbuild [options] top_level_netlist_file output_ngc_file
```

NetGen 論理シミュレーションの出力ファイル

- ・ **V ファイル** : IEEE 1364-2001 準拠の Verilog HDL ファイルで、入力デザイン ファイルからのネットリスト情報が含まれます。論理シミュレーションのモデルとなるファイルで、シミュレーションのみで使用できます。合成には使用できません。
- ・ **VHD ファイル** : IEEE 1076.4 VITAL-2000 準拠の VHDL ファイルで、入力デザイン ファイルからのネットリスト情報が含まれます。論理シミュレーションのモデルとなるファイルで、シミュレーションのみで使用できます。合成には使用できません。

NetGen 論理シミュレーションの構文

論理シミュレーションを実行するには、次のように指定します。

```
netgen -ofmt [verilog|vhdl] [options] input_file [.ngd|.ngc]
```

-ofmt : 出力ネットリストのフォーマット (verilog または vhdl) を指定します。

options : 「NetGen シミュレーション フローのオプション」にリストされているオプションを 1 つ以上指定できます。このセクションには、共通のオプションに加えて、Verilog および VHDL 特有のオプションが含まれます。

input_file : 入力ファイルを指定します。NGD ファイルを指定する場合は、拡張子 .ngd を付ける必要があります。

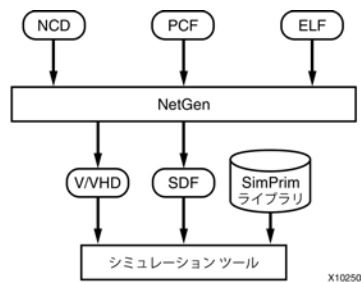
NetGen のタイミング シミュレーション フロー

NetGen のタイミング シミュレーション フローは、FPGA および CPLD デザインのタイミング検証に使用します。FPGA デザインの場合、タイミング シミュレーションは PAR 後に実行しますが、コンポーネント遅延情報のみが必要な場合には、MAP 後でも実行できます。タイミング シミュレーションを実行する場合、生成されるネットリストのタイプを Verilog または VHDL に指定する必要があります。ネットリストのほかに SDF ファイルも出力されます。Verilog および VHDL ネットリストにはデザインの機能が、SDF ファイルにはタイミング情報が記述されています。

入力ファイルのタイプは、使用する FPGA または CPLD デザインによって異なります。入力ファイルのタイプなどデザイン固有の情報は、この後の「FPGA タイミング シミュレーション」および「CPLD タイミング シミュレーション」を参照してください。

FPGA タイミング シミュレーション

NetGen タイミング シミュレーション フローは、FPGA デザインのタイミングを検証し、Verilog または VHDL ネットリストと SDF ファイルを生成するために使用します。次に、FPGA デザインを使用した NetGen のタイミング シミュレーション フローを示します。



FPGA タイミング シミュレーションの入力ファイルは、次のとおりです。

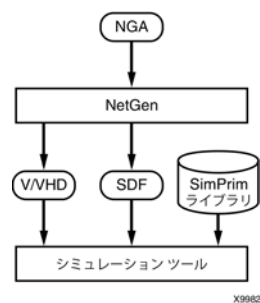
- ・ **NCD ファイル** : 物理的に記述されたデザイン ファイル。マップ済み、一部または完全な配置済み、一部または完全な配線済みのいずれかです。
- ・ **PCF ファイル (オプション)** : 物理制約ファイル。デザインに電圧または温度の設定が適用されている場合、このファイルから NetGen にデータを渡す必要があります。詳細は、「[-pcf \(PCF ファイル\)](#)」を参照してください。
- ・ **ELF (MEM) ファイル (オプション)** : BMM ファイルで指定されたブロック RAM の情報を含むファイル。詳細は、「[-bd \(ブロック RAM のデータ ファイル\)](#)」を参照してください。

FPGA タイミング シミュレーションの出力ファイルは、次のとおりです。

- ・ **SDF ファイル** : SDF 3.0 準拠の標準遅延フォーマット ファイルで、入力デザイン ファイルからの遅延が含まれます。
- ・ **V ファイル** : IEEE 1364-2001 準拠の Verilog HDL ファイルで、入力デザイン ファイルからのネットリスト情報が含まれます。論理シミュレーションのモデルとなるファイルで、シミュレーションのみで使用できます。合成には使用できません。
- ・ **VHD ファイル** : IEEE 1076.4 VITAL-2000 準拠の VHDL ファイルで、入力デザイン ファイルからのネットリスト情報が含まれます。論理シミュレーションのモデルとなるファイルで、シミュレーションのみで使用できます。合成には使用できません。

CPLD タイミング シミュレーション

NetGen タイミング シミュレーション フローは、CPLDFit を実行して CPLD デザインをインプリメントし、**-tsim** オプションを実行して遅延をアノテートした後で、デザインのタイミングを検証するために使用します。入力ファイルには、TSIM からアノテートされた NGA ファイルを使用します。



メモ : 詳細は、「[CPLDFit](#)」の章および「[TSIM](#)」の章を参照してください。

CPLD タイミング シミュレーションに使用する入力ファイルは次のとおりです。

NGA ファイル : TSIM により生成された、ザイリンクス プリミティブを含む論理的なデザイン ファイル。詳細は、「[TSIM](#)」の章を参照してください。

CPLD シミュレーション フローの出力ファイルは、次のとおりです。

- ・ **SDF ファイル** : 標準遅延フォーマット ファイルで、入力 NGA ファイルからの遅延が含まれます。
- ・ **V ファイル** : IEEE 1364-2001 準拠の Verilog HDL ファイルで、入力 NGA ファイルからのネットリスト情報が含まれます。論理シミュレーションのモデルとなるファイルで、シミュレーションのみで使用できます。合成には使用できません。
- ・ **VHD ファイル** : IEEE 1076.4 VITAL-2000 準拠の VHDL ファイルで、入力 NGA ファイルからのネットリスト情報が含まれます。論理シミュレーションのモデルとなるファイルで、シミュレーションのみで使用できます。合成には使用できません。

NetGen タイミング シミュレーション フローの構文

NetGen タイミング シミュレーション フローを実行するには、次のように入力します。

netgen -sim -ofmt [verilog|vhdl] [options] input_file [.ncd]

verilog または **vhdl** : 出力ネットリストのフォーマットを指定します。

options : 「NetGen シミュレーション フローのオプション」にリストされているオプションを 1 つ以上指定できます。このセクションには、共通のオプションに加えて、Verilog および VHDL 特有のオプションが含まれます。

input_file : 入力ファイルを指定します。

コマンドラインの使用法に関するヘルプを表示するには、「**netgen -h sim**」と入力します。

NetGen シミュレーション フローのオプション

このセクションでは、NetGen タイミング シミュレーション フローでサポートされているコマンドライン オプションについて説明します。

- ・ **-aka** (別名をコメントとして記述)
- ・ **-bd** (ブロック RAM のデータ ファイル)
- ・ **-bx** (ブロック RAM の INIT ファイルのディレクトリ)
- ・ **-dir** (ディレクトリ名)
- ・ **-fn** (階層のないネットリストの生成)
- ・ **-gp** (グローバル リセット ネットをポートとして指定)
- ・ **-insert_pp_buffers** (パス パルス バッファの挿入)
- ・ **-intstyle** (統合スタイル)
- ・ **-mhf** (複数の階層ファイルを出力)
- ・ **-ofmt** (出力形式)
- ・ **-pcf** (PCF ファイル)
- ・ **-s** (スピードの変更)
- ・ **-sim** (シミュレーション ネットリストの生成)
- ・ **-tb** (テストベンチ テンプレート ファイルの生成)
- ・ **-ti** (最上位インスタンス名)
- ・ **-tp** (グローバル トライステート ネットをポートに指定)
- ・ **-w** (既存ファイルの上書き)

-aka (別名をコメントとして記述)

元のユーザー定義の識別子をコメントとしてネットリストに記述します。NetGen の名前を有効にする処理によりユーザー定義の識別子に変更された場合、このオプションを使用すると便利です。

構文

-aka

-bd (ブロック RAM のデータ ファイル)

BMM ファイルで指定されたブロック RAM インスタンスにデータを渡すファイルのパスと名前を指定します。Data2MEM は、ELF (EDK により生成) または MEM ファイルに含まれるアドレスとデータ情報からデータを割り当てる **ADDRESS_BLOCK** を判断します。**-bd** オプションは、複数回使用できます。

tag tagname を指定すると、BMM ファイルで指定されたアドレス空間と同じ名前のアドレス空間のみが変換に使用され、*tagname* で指定されたアドレス空間以外のデータはすべて無視されます。

構文

-bd *filename* [.elf|.mem] [**tag** *tagname*]

-bx (ブロック RAM の INIT ファイルのディレクトリ)

ブロック RAM の INIT ファイルを保存するディレクトリを指定します。

構文

-bx *bram_output_dir*

-dir (ディレクトリ名)

出力ファイルを保存するディレクトリを指定します。

構文

-dir *directory_name*

-fn (階層のないネットリストの生成)

フラット化されたネットリストを生成します。フラットなネットリストにはデザインが階層がありません。

構文

-fn

-gp (グローバル リセット ネットをポートとして指定)

物理的なデザインでフリップフロップとラッチに接続されているグローバルリセット信号を、最上位モジュールのポートとして指定します。ポート名を指定することにより、フロントエンドで使ったポート名と一致させることができます。

グローバルリセット ネットが駆動されない場合にのみ使用します。たとえば、Virtex®-5 デザインに STARTUP_VIRTEX5 コンポーネントを使用すると、このコンポーネントがグローバルリセット ネットを駆動するため、**-gp** オプションは使用できません。

構文

-gp *port_name*

メモ : GR、GSR、PRLD、PRELOAD、RESET は、ザイリンクス ソフトウェアでの予約名なので、ポート名として使用しないでください。このオプションは、NGC ファイルを入力として使用する UNISIM ベースのフローでは無視されます。

-isert_pp_buffers (パス パルス バッファの挿入)

パルスが失われるのを回避するためにパス パルス バッファを挿入するかどうかを指定します。バックアノテートされたタイミング シミュレーションで、パルス幅よりコンポーネントの入力ポートの遅延の方が大きい場合、パルスが失われることがあります。たとえば、周期が 5ns (High が 2.5ns、Low が 2.5ns) のクロックがバッファを介して伝搬された場合、SDF でそのバッファの入力ポートの PORT または IOPATH 遅延が 2.5ns より大きいと、出力の波形は変化しません (シミュレーションの開始時に出力が X であれば X のまま)。

メモ : このオプションは、入力に NCD ファイルである場合に使用できます。

構文

-insert_pp_buffers true|false

デフォルトでは false に設定されています。

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle ise|xflow|silent

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-mhf (複数の階層ファイルを出力)

KEEP_HIERARCHY 属性が設定されたモジュールごとに階層ファイルを出力します。

メモ : 詳細は、「[階層ファイルの保持および出力](#)」を参照してください。

構文

-mhf

-ofmt (出力形式)

出力するネットリストの形式を Verilog または VHDL に指定します。

構文

```
-ofmt verilog|vhdl
```

-pcf (PCF ファイル)

NetGen の入力ファイルとして物理制約ファイル (PCF) を指定します。PCF ファイルを指定する必要があるのは、温度/電圧の制約が含まれている場合のみです。

温度/電圧の制約と遅延の比例配分については、『[制約ガイド](#)』を参照してください。

構文

```
-pcf pcf_file.pcf
```

-s (スピードの変更)

指定したデバイスのスピードがネットリストにアノテートされるよう指定します。

構文

```
-s speed grade|min
```

speed grade には、マイナス記号を付けても付けなくてもかまいません。たとえば、**-s 3** と **-s -3** は、どちらも有効です。

アーキテクチャによっては、**-s min** オプションが使用できます。このオプションを使用すると、ワーストケースの最大遅延ではなく、プロセス最小遅延がネットリストにアノテートされます。使用するアーキテクチャでプロセス最小遅延がサポートされているかどうかを確認するには、Speedprint または PARTGen を使用してください。詳細は、『[PARTGen](#)』の章を参照してください。

このオプションを使用すると、PCF 内の温度/電圧に関するタイミング パラメータがすべて無効になります。**-s min** を使用すると、生成された SDF ファイルのすべてのフィールド (MIN:TYP:MAX) がプロセス最小値に設定されます。

-sim (シミュレーション ネットリストの生成)

シミュレーションネットリストを出力します(デフォルト)。

構文

```
-sim
```

-tb (テストベンチ テンプレート ファイルの生成)

テストベンチ ファイルを生成します。拡張子は Verilog の場合は **.tv**、VHDL の場合は **.tvhd** です。このファイルは、入力 NCD ファイルを基にしたテンプレートファイルです。テンプレートファイルのタイプは、**-ofmt** オプションを使用して Verilog または VHDL に設定します。

構文

```
-tb
```

-ti (最上位インスタンス名)

-tb オプションを使用して生成されたテストベンチ ファイル内で、テストするデザインに対してユーザー インスタンス名を指定します。

構文

-ti *top_instance_name*

-tp (グローバルトライステート ネットをポートに指定)

グローバルトライステート信号 (すべての FPGA 出力をハイ インピーダンス状態にする) を、最上位モジュールまたは出力ファイルのポートとして指定します。ポート名を指定することにより、フロントエンドで使用したポート名と一致させることができます。

グローバルトライステート ネットが駆動されない場合にのみ使用します。

メモ: デザインで既に存在するワイヤ名やポート名を使用すると、エラーが発生します。このオプションは、NGC ファイルを入力として使用する UNISIM ベースのフローで無視されます。

構文

-tp *port_name*

-w (既存ファイルの上書き)

既存のネットリスト (VHD または V ファイル) を上書きします。デフォルトでは、ネットリスト ファイルが上書きされないように設定されています。

メモ: ほかのすべての出力ファイルは自動的に上書きされます。

構文

-w

Verilog 特定の論理およびタイミング シミュレーション オプション

このセクションでは、論理シミュレーションおよびタイミング シミュレーションに使用する Verilog オプションをリストします。

- ・ **-insert_glbl** (glbl.v モジュールの挿入)
- ・ **-ism** (Verilog ファイルに SIMPRIM モジュールを含める)
- ・ **-ne** (不正な名前をアンダースコアに置換)
- ・ **-pf** (PIN ファイルの生成)
- ・ **-sdf_anno** (\$sdf_annotate 文の記述)
- ・ **-sdf_path** (SDF ファイルの完全パスを指定)
- ・ **-shm** (\$shm 文の記述)
- ・ **-ul** ('uselib 文の記述)
- ・ **-vcd** (\$dumpfile/\$dumpvars 文の記述)

-insert_glbl (glbl.v モジュールの挿入)

出力 Verilog シミュレーション ネットリストに glbl.v モジュールを含めるかどうかを指定します。

構文

-insert_glbl [true|false]

デフォルトでは true に設定されています。

false に設定すると、出力 Verilog ネットリストに glbl.v モジュールは含まれません。glbl.v モジュールの詳細は、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

メモ : **-mhf** (複数の階層ファイルを出力) オプションを使用している場合、**-insert_glbl** オプションを true に設定することはできません。

-ism (Verilog ファイルに SIMPRIM モジュールを含める)

SIMPRIM ライブラリの SIMPRIM モジュールを出力 Verilog ファイル (拡張子 .v) に含めます。このオプションを使用すると、シミュレーション時にライブラリ パスを指定する必要がなくなりますが、ネットリスト ファイルのサイズが大きくなり、コンパイル時間が長くなります。

このオプションを使用すると、ライブラリ パスが正しく設定されているかどうかを確認されます。正しいライブラリ パスの例は次のとおりです。

```
$XILINX/verilog/src/simprim
```

コンパイルされたライブラリを使用している場合、このオプションを使用しても意味がありません。このオプションを使用する場合、**-ul** オプションは使用しないでください。

メモ : このオプションは、変換後 (NGD)、マップ後、および配置配線後のシミュレーションフローに使用できます。

構文

-ism

-ne (不正な名前をアンダースコアに置換)

不正な文字をアンダースコア () に置き換え、エスケープ処理が実行されないようにします。たとえば、「p1\$40/empty」というネット名は「p1\$40_empty」に変更されます。名前の前にバックスラッシュは付きません。エスケープ文字が追加された識別子が ベンダーの Verilog ソフトウェアにより正しく読み取られなかった場合、このオプションを使用して生成された Verilog ファイルを使用します。

構文

-ne

デフォルトでは (**-ne** オプションなし)、不正なブロック名またはネット名の前にエスケープ文字としてバックスラッシュ (\)、名前の後ろにスペースが追加されます。たとえば、「p1\$40/empty」というネット名は「\p1\$40/empty」に変更されます。不正な名前とは、input や output などの Verilog の予約名や、Verilog の命名規則に準拠しない文字のことです。

-pf (PIN ファイルの生成)

PIN ファイルを生成します。

このオプションは、FPGA/Cadence でのみ使用できます。

構文

-pf

-sdf_anno (\$sdf_annotate 文の記述)

Verilog ネットリストに \$sdf_annotate 文を記述するかどうかを指定します。デフォルトでは true に設定されています。このオプションをディスエーブルにするには、false に設定します。

メモ : このオプションは、タイミング シミュレーション フローに使用できます。

構文

```
-sdf_anno [true|false]
```

-sdf_path (SDF ファイルの完全パスを指定)

SDF ファイルの出力先の完全パスを指定します。このオプションを使用すると、完全パスと SDF ファイル名が \$sdf_annotate 文に記述されます。完全パスを指定しない場合、作業ディレクトリへの完全パスと SDF ファイル名が記述されます。

メモ : このオプションは、タイミング シミュレーション フローに使用できます。

構文

```
-sdf_path [path_name]
```

-shm (\$shm 文の記述)

NetGen により作成される Verilog ファイルに \$shm 文を記述します。\$shm 文により NC-Verilog ファイルでシミュレーション データが波形として表示されます。このオプションは、Cadence 社の NC-Verilog ファイルでのみ使用できます。

構文

```
-shm
```

-ul ('uselib 文の記述)

出力される Verilog (拡張子 .v) ファイルに SIMPRIM ライブラリへのパスを記述します。このパスは次のように記述されます。

```
uselib dir=$XILINX/verilog/src/simprims libext=.v
```

\$XILINX : ザイリンクス ソフトウェアのインストール ディレクトリです。

このオプションを指定しない場合、'uselib 文は Verilog ファイルに記述されません。

メモ : Verilog ファイルの最後に空の 'uselib 文が自動的に追加され、'uselib のデータがクリアされます。このオプションを使用する場合、**-ism** オプションは使用しないでください。

メモ : **-ul** オプションは、SIMPRIM ベースの論理シミュレーション フローとタイミング シミュレーション フローで使用できますが、すべてのシミュレータで 'uselib 文がサポートされているわけではありません。このオプションの使用には、注意が必要です。

構文

```
-ul
```

-vcd (\$dumpfile/\$dumpvars 文の記述)

\$dumpfile/\$dumpvars 文をテストベンチに記述します。このオプションは、Cadence 社の Verilog ファイルでのみ使用できます。

構文

-vcd

VHDL 特定の論理およびタイミング シミュレーション オプション

このセクションでは、論理シミュレーションおよびタイミング シミュレーションに使用する VHDL 特定のオプションをリストします。

- ・ **-a** (アーキテクチャのみ生成)
- ・ **-ar** (アーキテクチャ名の変更)
- ・ **-extid** (拡張識別子)
- ・ **-rpw** (ROC のパルス幅の指定)
- ・ **-tpw** (TOC のパルス幅の指定)

-a (アーキテクチャのみ生成)

出力ファイルにエンティティを生成しないように指定します。このオプションを使用すると、出力にアーキテクチャのみが生成されます。デフォルトでは、入力デザインのエンティティとアーキテクチャの両方が生成されます。

構文

-a

-ar (アーキテクチャ名の変更)

NetGen で生成されたアーキテクチャ名を変更します。デフォルトでは、ネットリストに記述されたエンティティのアーキテクチャ名は STRUCTURE です。

構文

-ar *architecture_name*

-extid (拡張識別子)

VHDL の拡張識別子を記述します。識別子には基本識別子と拡張識別子があり、デフォルトでは基本識別子のみが記述されます。

構文

-extid

-rpw (ROC のパルス幅の指定)

ROC コンポーネントのパルス幅をナノ秒単位で指定します。コンポーネントをシミュレーションするには、正の整数を指定する必要があります。このオプションは必須ではありません。デフォルトでは、ROC のパルス幅は 100ns に設定されています。

構文

-rpw *roc_pulse_width*

-tpw (TOC のパルス幅の指定)

TOC コンポーネントに対しパルス幅を ns (ナノ秒) で設定します。コンポーネントをシミュレーションするには、正の整数を指定する必要があります。グローバル セット/リセット ネットおよびグローバル トライステート ネットにソースがない場合など、TOC コンポーネントをインスタンス化する場合、このオプションを指定する必要があります。

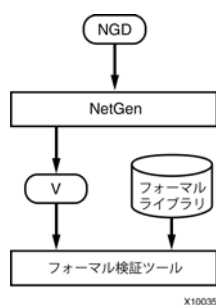
構文

```
-tpw toc_pulse_width
```

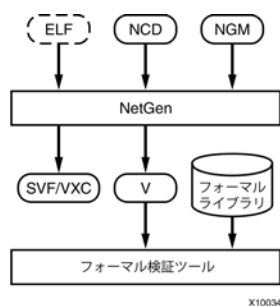
NetGen の等価チェック フロー

NetGen の等価チェック フローは、FPGA デザインのフォーマル検証に使用します。このフローでは、Verilog ネットリストと、等価チェック ツールで使用される Conformal-LEC または Formality アサーション ファイルが生成されます。

NGDBuild 後のフロー (FPGA)



インプリメンテーション後のフロー (FPGA)



NetGen 等価チェックの入力ファイル

NetGen 等価チェック フローの入力ファイルは、次のとおりです。

- ・ **NGD ファイル** : マップされていない FPGA デザインが論理的に記述されたファイル。
- ・ **NCD ファイル** : 物理的に記述されたデザイン ファイル。マップ済み、一部または完全な配置済み、一部または完全な配線済みのいずれかです。
- ・ **NGM ファイル** : MAP により生成されたマップ済みデザイン ファイルで、MAP 処理で削除、変更された内容が記述されています。詳細は、「[-ngm \(デザイン補正ファイル\)](#)」を参照してください。
- ・ **ELF (MEM) ファイル (オプション)** : BMM ファイルで指定されたブロック RAM 情報が記述されています。詳細は、「[-bd \(ブロック RAM のデータ ファイル\)](#)」を参照してください。

NetGen 等価チェックの出力ファイル

NetGen 等価チェック フローの出力ファイルは、次のとおりです。

- ・ **Verilog ファイル (.v)** : IEEE 1364-2001 準拠の Verilog HDL ファイルで、入力ファイルからのネットリスト情報が含まれます。このファイルは等価チェックのモデルとなり、合成など等価チェック以外の目的には使用できません。
- ・ **Formality ファイル (.svf)** : Formality 等価チェック ツールから出力されるアサーション ファイル。ザイリンクスのインプリメンテーション ツールにより、デザインに加えられた変更に関する情報が記述されています。
- ・ **Conformal-LEC ファイル (.vxc)** : Conformal-LEC 等価チェック ツールから出力されるアサーション ファイル。ザイリンクスのインプリメンテーション ツールにより、デザインに加えられた変更に関する情報が記述されています。

メモ : Conformal-LEC および Formality ツールについては、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

NetGen 等価チェックの構文

NetGen 等価チェック フローを実行するコマンドは次のとおりです。

```
netgen -ecn [tool_name] [options] input_file [.ncd|.ngd] ngm_file
```

options : 「[NetGen 等価チェック フローのオプション](#)」にリストされているオプションを 1 つ 以上指定できます。

tool_name : 等価チェック ツールに使用されるネットリストを生成するためのオプションです。設定可能な値は、**conformal** または **formality** です。等価チェックおよびフォーマル検証のツールについては、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

input_file : 入力ファイルです。NGD ファイルを指定する場合は、拡張子 **.ngd** を付ける必要があります。

ngm_file : MAP により生成されたデザイン ファイルで、MAP 処理で削除、変換された内容が記述されています。このファイルはオプションですが、指定することをお勧めします。

コマンドラインの使用法に関するヘルプを表示するには、「**netgen -h ecn**」と入力します。

NetGen 等価チェック フローのオプション

このセクションでは、NetGen 等価チェック フローでサポートされているオプションについて説明します。

- ・ `-aka` (別名をコメントとして記述)
- ・ `-bd` (ブロック RAM のデータ ファイル)
- ・ `-bx` (ブロック RAM の INIT ファイルのディレクトリ)
- ・ `-dir` (ディレクトリ名)
- ・ `-ecn` (等価チェック)
- ・ `-fn` (階層のないネットリストの生成)
- ・ `-intstyle` (統合スタイル)
- ・ `-mhf` (複数の階層ファイルを出力)
- ・ `-ne` (不正な名前をアンダースコアに置換)
- ・ `-ngm` (デザイン補正ファイル)
- ・ `-w` (既存ファイルの上書き)

`-aka` (別名をコメントとして記述)

元のユーザー定義の識別子をコメントとしてネットリストに記述します。NetGen の名前を有効にする処理によりユーザー定義の識別子を変更された場合、このオプションを使用すると便利です。

構文

`-aka`

`-bd` (ブロック RAM のデータ ファイル)

BMM ファイルで指定されたブロック RAM インスタンスにデータを渡すファイルのパスと名前を指定します。Data2MEM は、ELF (EDK により生成) または MEM ファイルに含まれるアドレスとデータ情報からデータを割り当てる **ADDRESS_BLOCK** を判断します。`-bd` オプションは、複数回使用できます。

tag tagname を指定すると、BMM ファイルで指定されたアドレス空間と同じ名前のアドレス空間のみが変換に使用され、*tagname* で指定されたアドレス空間以外のデータはすべて無視されます。

構文

`-bd filename [.elf|.mem] [tag tagname]`

`-dir` (ディレクトリ名)

出力ファイルを保存するディレクトリを指定します。

構文

`-dir directory_name`

`-ecn` (等価チェック)

FPGA デザインのフォーマル検証に使用する等価チェック ネットリストを生成します。

等価チェックおよびフォーマル検証のツールについては、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

構文

```
netgen -ecn tool_name
```

tool_name : ネットリストを出力するツールの名前を指定します。設定可能な値は、conformal または formality です。

-fn (階層のないネットリストの生成)

フラット化されたネットリストを生成します。フラットなネットリストにはデザインが階層がありません。

構文

```
-fn
```

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

```
-intstyle ise|xflow|silent
```

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-mhf (複数の階層ファイルを出力)

KEEP_HIERARCHY 属性が設定されたモジュールごとに階層ファイルを出力します。

メモ : 詳細は、『[階層ファイルの保持および出力](#)』を参照してください。

構文

```
-mhf
```

-ne (不正な名前をアンダースコアに置換)

不正な文字をアンダースコア () に置き換え、エスケープ処理が実行されないようにします。たとえば、「p1\$40/empty」というネット名は「p1\$40_empty」に変更されます。名前の前にバックスラッシュは付きません。エスケープ文字が追加された識別子が ベンダーの Verilog ソフトウェアにより正しく読み取られなかった場合、このオプションを使用して生成された Verilog ファイルを使用します。

構文

```
-ne
```

デフォルトでは (**-ne** オプションなし)、不正なブロック名またはネット名の前にエスケープ文字としてバックスラッシュ (\)、名前の後ろにスペースが追加されます。たとえば、「p1\$40/empty」というネット名は「\p1\$40/empty」に変更されます。不正な名前とは、input や output などの Verilog の予約名や、Verilog の命名規則に準拠しない文字のことです。

-ngm (デザイン補正ファイル)

デザイン補正ファイル (NGM) を指定します。このオプションは、等価チェックフローにのみ使用できます。

構文

-ngm [*ngm_file*]

-w (既存ファイルの上書き)

既存のネットリスト (VHD または V ファイル) を上書きします。デフォルトでは、ネットリスト ファイルが上書きされないように設定されています。

メモ : ほかのすべての出力ファイルは自動的に上書きされます。

構文

-w

NetGen のスタティック タイミング解析フロー

NetGen のスタティック タイミング解析フローは、FPGA デザインの最大遅延の最小値を含む FPGA デザインのタイミング解析に使用します。

最大遅延の最小値は、スキューやセットアップ/ホールド タイムを算出するため、スタティック タイミング解析ツールにより使用されます。この値は、スピード グレードや温度/電圧など指定した動作条件下でのデバイスの最小遅延値を表します。動作温度/電圧を指定しない場合、ワーストケースの温度/電圧が使用されます。最大遅延の最小値は、**-s min** オプションを使用して生成されるプロセス最小遅延とは異なります。

次に、相対最小値と最大遅延値が記述された DELAY プロパティを示します。

(DELAY)

(ABSOLUTE)

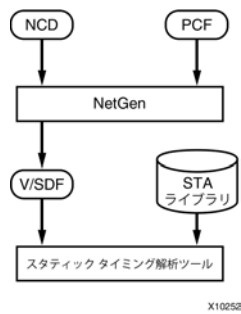
(PORT I (234:292:292) (234:292:292))

(IOPATH I O (392:489:489) (392:489:489))

メモ : TYP および MAX の両フィールドに最大遅延が記述されます。

NetGen は、スタティック タイミング解析フローを使用し、スタティック タイミング解析ツールで 사용되는 Verilog および SDF ネットリストを生成します。

スタティック タイミング解析フロー (FPGA)



スタティック タイミング解析用の入力ファイル

スタティック タイミング解析の入力ファイルは、次のとおりです。

- ・ **NCD ファイル**：物理的に記述されたデザイン ファイル。マップ済み、一部または完全な配置済み、一部または完全な配線済みのいずれかです。
- ・ **PCF ファイル (オプション)**：物理制約ファイル。デザインに電圧または温度の設定が適用されている場合、このファイルから NetGen にデータを渡す必要があります。詳細は、「[-pcf \(PCF ファイル\)](#)」を参照してください。

スタティック タイミング解析用の出力ファイル

スタティック タイミング解析の出力ファイルは、次のとおりです。

- ・ **SDF ファイル**：SDF 3.0 準拠の標準遅延フォーマット ファイルで、入力ファイルからの遅延が含まれます。
- ・ **Verilog ファイル (.v)**：IEEE 1364-2001 準拠の Verilog HDL ファイルで、入力ファイルからのネットリスト情報が含まれます。このファイルはスタティック タイミング解析にのみ使用されるタイミング シミュレーション モデルとなり、合成などスタティック タイミング解析以外の目的には一切使用できません。このネットリストには、シミュレーション プリミティブが使用されており、デバイスの実際のインプリメンテーションを表しているとは限りませんが、インプリメント済みデザインのファンクション モデルを表します。

NetGen スタティック タイミング解析の構文

NetGen スタティック タイミング解析フローを実行するコマンドは次のとおりです。

```
netgen -sta input_file [.ncd]
```

input_file：入力ファイル名を指定します。

コマンドラインの使用法に関するヘルプを表示するには、「**netgen -h sta**」と入力します。

NetGen スタティック タイミング解析フローのオプション

このセクションでは、NetGen スタティック タイミング解析フローでサポートされているコマンドライン オプションについて説明します。

- ・ `-aka` (別名をコメントとして記述)
- ・ `-bd` (ブロック RAM のデータ ファイル)
- ・ `-bx` (ブロック RAM の INIT ファイルのディレクトリ)
- ・ `-dir` (ディレクトリ名)
- ・ `-fn` (階層のないネットリストの生成)
- ・ `-intstyle` (統合スタイル)
- ・ `-mhf` (複数の階層ファイルを出力)
- ・ `-ne` (不正な名前をアンダースコアに置換)
- ・ `-pcf` (PCF ファイル)
- ・ `-s` (スピードの変更)
- ・ `-sta` (スタティック タイミング解析ネットリストの生成)
- ・ `-w` (既存ファイルの上書き)

`-aka` (別名をコメントとして記述)

元のユーザー定義の識別子をコメントとしてネットリストに記述します。NetGen の名前を有効にする処理によりユーザー定義の識別子に変更された場合、このオプションを使用すると便利です。

構文

`-aka`

`-bd` (ブロック RAM のデータ ファイル)

BMM ファイルで指定されたブロック RAM インスタンスにデータを渡すファイルのパスと名前を指定します。Data2MEM は、ELF (EDK により生成) または MEM ファイルに含まれるアドレスとデータ情報からデータを割り当てる **ADDRESS_BLOCK** を判断します。`-bd` オプションは、複数回使用できます。

tag tagname を指定すると、BMM ファイルで指定されたアドレス空間と同じ名前のアドレス空間のみが変換に使用され、*tagname* で指定されたアドレス空間以外のデータはすべて無視されます。

構文

`-bd filename [.elf|.mem] [tag tagname]`

`-dir` (ディレクトリ名)

出力ファイルを保存するディレクトリを指定します。

構文

`-dir directory_name`

-fn (階層のないネットリストの生成)

フラット化されたネットリストを生成します。フラットなネットリストにはデザインが階層がありません。

構文

-fn

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle ise|xflow|silent

次のいずれかのモードに設定します。

- ・ **-intstyle ise**: プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow**: プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent**: 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ: Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-mhf (複数の階層ファイルを出力)

KEEP_HIERARCHY 属性が設定されたモジュールごとに階層ファイルを出力します。

メモ: 詳細は、「[階層ファイルの保持および出力](#)」を参照してください。

構文

-mhf

-ne (不正な名前をアンダースコアに置換)

不正な文字をアンダースコア () に置き換え、エスケープ処理が実行されないようにします。たとえば、「p1\$40/empty」というネット名は「p1\$40_empty」に変更されます。名前の前にバックスラッシュは付きません。エスケープ文字が追加された識別子が ベンダーの Verilog ソフトウェアにより正しく読み取られなかった場合、このオプションを使用して生成された Verilog ファイルを使用します。

構文

-ne

デフォルトでは (-ne オプションなし)、不正なブロック名またはネット名の前にエスケープ文字としてバックスラッシュ (\)、名前の後ろにスペースが追加されます。たとえば、「p1\$40/empty」というネット名は「\p1\$40/empty」に変更されます。不正な名前とは、input や output などの Verilog の予約名や、Verilog の命名規則に準拠しない文字のことです。

-pcf (PCF ファイル)

NetGen の入力ファイルとして物理制約ファイル (PCF) を指定します。PCF ファイルを指定する必要があるのは、温度/電圧の制約が含まれている場合のみです。

温度/電圧の制約と遅延の比例配分については、『[制約ガイド](#)』を参照してください。

構文

```
-pcf pcf_file.pcf
```

-s (スピードの変更)

指定したデバイスのスピードがネットリストにアノテートされるよう指定します。

構文

```
-s speed grade | min
```

speed grade には、マイナス記号を付けても付けなくてもかまいません。たとえば、**-s 3** と **-s -3** は、どちらも有効です。

アーキテクチャによっては、**-s min** オプションが使用できます。このオプションを使用すると、ワーストケースの最大遅延ではなく、プロセス最小遅延がネットリストにアノテートされます。使用するアーキテクチャでプロセス最小遅延がサポートされているかどうかを確認するには、Speedprint または PARTGen を使用してください。詳細は、『[PARTGen](#)』の章を参照してください。

このオプションを使用すると、PCF 内の温度/電圧に関するタイミング パラメータがすべて無効になります。**-s min** を使用すると、生成された SDF ファイルのすべてのフィールド (MIN:TYP:MAX) がプロセス最小値に設定されます。

-sta (スタティック タイミング解析ネットリストの生成)

スタティック タイミング解析ネットリストを出力します。

構文

```
-sta
```

-w (既存ファイルの上書き)

既存のネットリスト (VHD または V ファイル) を上書きします。デフォルトでは、ネットリスト ファイルが上書きされないように設定されています。

メモ : ほかのすべての出力ファイルは自動的に上書きされます。

構文

```
-w
```

階層ファイルの保持および出力

KEEP_HIERARCHY 制約を使用して合成およびインプリメンテーションの実行中も階層を保持する場合は、NetGen **-mhf** オプションを使用して階層ごとにネットリストと SDF ファイルを出力できます。

STARTUP および glbl (Verilog のみ) モジュールの階層は、出力ネットリストに保持されます。**-mhf** オプションを使用し、KEEP_HIERARCHY 制約が設定された階層ブロックが少なくとも 1 つある場合は、STARTUP と glbl モジュールに対して別々のネットリストが出力されます。この制約が設定された階層ブロックがない場合は、**-mhf** オプションが無視されます。

このセクションでは、**-mhf** オプションを使用して生成するファイルのタイプについて説明します。ファイル タイプは、タイミング シミュレーション、等価チェック、スタティック タイミング解析のどのフローを実行しているかで異なります。たとえば、タイミング シミュレーションを実行した場合、Verilog または VHDL ファイルが出力されます。この場合、**-ofmt** オプションを使用して出力ファイルのタイプを指定する必要があります。

メモ： ファイル タイプに Verilog を指定すると、モジュールの Verilog ネットリストにそれぞれ `$sdf_annotate` 文が記述されます。

次に、階層ファイルの命名規則を示します。

階層ファイル

階層ファイル	シミュレーション	等価チェック	スタティック タイミング解析
最上位モジュールを含むファイル	[input_filename] (デフォルト) またはユーザー指定の出力ファイル名	[input_filename].ecn またはユーザー指定の出力ファイル名	[input_filename].sta またはユーザー指定の出力ファイル名
下位モジュールを含むファイル	[module_name].sim	[module_name].ecn	[module_name].sta

[module_name] は、フロントエンドからの階層モジュール名です。ただし、次のようにモジュール名が異なっていることがあります。

- ・ モジュールのインスタンスを複数使用した場合、モジュールのタイミングが異なるため、インスタンスもそれぞれ異なります。インスタンス名には、アンダースコアの後に「INST_」と番号が付けられます (numgen、numgen_INST_1、numgen_INST_2 など)。
- ・ 新規ファイルと既存ファイルに名前の競合があった場合、新規ファイルの名前は、[module_name]_[instance_name] のようになります。

テストベンチ ファイル

-tb オプションを使用すると、最上位デザインに対してテストベンチ ファイルが生成されます。このファイルのベース名はデザイン ファイルのベース名と同じです。拡張子は、Verilog の場合は .tv、VHDL の場合は .tvhd です。

階層情報ファイル

NetGen では、個別のネットリストだけではなく、階層情報が記述されたテキスト ファイルも出力されます。このテキスト ファイルには、次のような情報が記述されています。これらのファイルのうち 1 つでも関連情報が含まれていない場合は、階層情報が記述されません。

```
// Module : The name of the hierarchical design module.
// Instance : The instance name used in the parent module.
// Design File : The name of the file that contains the module.
// SDF File : The SDF file associated with the module.
// SubModule : The sub module(s) contained within a given module.
// Module, Instance : The sub module and instance names.
```

メモ： 最上位デザインの階層情報ファイルには、Instance フィールドは含まれません。

階層情報ファイルのベース名は `design_base_name_mhf_info.txt` です。

STARTUP ブロックは、最上位モジュールでのみサポートされます。「[バックアノテーション シミュレーションでの専用グローバル信号の処理](#)」で説明しているように、グローバル セット/リセット信号 (GSR) とグローバル トライステート信号 (GTS) の接続は保持されます。

バックアノテーション シミュレーションでの専用グローバル信号の処理

グローバル セット/リセット信号 (GSR)、グローバル トライステート信号 (GTS)、または CPLD の PRLD 信号は、コンポーネントをセット、リセット、またはトライステートの状態にできるグローバル 配線ネットです。これらの信号のシミュレーション ビヘイビアは、ザイリンクス SIMPRIM ライブラリのライブラリ セルと Verilog の gbl モジュール、または VHDL の X_ROC/X_TOC コンポーネントを使用したシミュレーション ネットリストでモデル化されています。

次のセクションでは、Verilog および VHDL ネットリストの接続について説明します。

Verilog ネットリストのグローバル信号

Verilog の gbl モジュールは、GSR 信号および GTS 信号のデフォルトのビヘイビアをモデル化するために使用されます。gbl.GSR および gbl.GTS は、グローバルの GSR/GTS 信号としてデザインの中のどの部分でも、またはどのライブラリ セルでも直接指定できます。

gbl モジュールの定義は、Verilog ネットリストに出力されます。階層になっていないデザインやファイルが 1 つのみの階層デザインの場合、gbl モジュールの定義はネットリストの最後の部分に出力されます。ファイルを 1 つ使用する階層デザインでは、gbl モジュールは最上位モジュール内で定義されます。複数のファイルを使用する階層デザイン (**-mhf** オプション) では、階層モジュールとして gbl.v が出力されます。

-gp オプションと **-tp** オプションを使用して GSR 信号と GTS 信号をポートとして最上位デザインに指定した場合、最上位モジュールには次のような接続が含まれます。

```
gbl.GSR = GSR_PORT  
gbl.GTS = GTS_PORT
```

GSR_PORT および GTS_PORT は、**-gp** オプションと **-tp** オプションによって生成された最上位モジュールのポートです。デザインに STARTUP ブロックを使用している場合、ブロックがバッファに変換され、ユーザー制御信号からグローバルの GSR および GTS (gbl.GSR および gbl.GTS) への接続が保持されます。

デザインに STARTUP ブロックを使用している場合、ブロックの階層レベルは常に出力ネットリストで保持されます。STARTUP の出力は、グローバルの GSR/GTS 信号 (gbl.GSR および gbl.GTS) に接続されます。

すべての階層デザインで、gbl モジュールはデザインと共にコンパイルされ、指定される必要があります。FPGA での GSR および GTS の設定については、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

VHDL ネットリストのグローバル信号

VHDL ネットリストのグローバル信号は、ライブラリ パッケージ Simprim_Vcomponents.vhd で宣言されている GSR および GTS です。GSR および GTS は、デザインのどの部分、またはどのライブラリ セルでも直接指定できます。

VHDL ライブラリの X_ROC および X_TOC コンポーネントは、GSR および GTS のデフォルトのビヘイビアをモデル化します。**-gp** オプションと **-tp** オプションを使用しない場合、X_ROC および X_TOC が出力ネットリストにインスタンス化されます。X_ROC と X_TOC のインスタンスは、各デザインに 1 つずつしかありません。階層デザインの場合、X_ROC と X_TOC は最上位モジュール ネットリストにインスタンス化されます。

X_ROC および X_TOC は、次のようにインスタンス化されます。

```
X_ROC (O => GSR);  
X_TOC (O => GTS);
```

-gp オプションと **-tp** オプションを使用して GSR と GTS を最上位デザインに指定した場合、ネットリストには X_ROC と X_TOC インスタンスは含まれません。代わりに、最上位モジュールに次のような接続が含まれます。

```
GSR<= GSR_PORT  
GTS<= GTS_PORT
```

GSR_PORT および GTS_PORT は、**-gp** オプションと **-tp** オプションによって生成された最上位モジュールのポートです。

デザインに STARTUP ブロックを使用している場合、ブロックの階層レベルが出力ネットリストで保持されます。STARTUP の出力はグローバルの GSR および GTS へ接続されます。

FPGA での GSR および GTS の設定については、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

論理的デザイン ルール チェック

この章では、論理的デザイン ルール チェック (DRC) について説明します。次のセクションが含まれています。

- ・ [論理的 DRC の概要](#)
- ・ [論理的 DRC によるチェック](#)

論理的 DRC の概要

論理的 DRC (デザイン ルール チェック) は、NGD (Native Generic Database) ファイルの論理的デザインを検証するための一連のテストです。論理的 DRC では、デバイスに依存しない検証が行われます。

論理的 DRC では、実行されたテストのステータスを示すメッセージが生成されます。ロジックが正しく動作しない場合はエラー メッセージが表示され、ロジックが不完全である場合は警告メッセージが表示されます。

論理的 DRC は、次の時点で自動的に実行されます。

- ・ NGDBuild の最終段階で、NGD ファイルが生成される前
DRC で警告が検出された場合は NGD ファイルが生成されますが、エラーが検出された場合にはファイルが生成されません。
- ・ NetGen の最終段階で、ネットリスト ファイルが生成される前
NetGen は、すべての DRC チェックを実行するわけではなく、ネット チェックとネーム チェックのみを実行します。DRC で警告やエラーが検出されても、ネットリスト ファイルは生成されます。

論理的 DRC のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

論理的 DRC によるチェック

論理的 DRC は、次のチェックを実行します。

- ・ [ブロック チェック](#)
- ・ [ネット チェック](#)
- ・ [パッド チェック](#)
- ・ [クロック バッファ チェック](#)
- ・ [ネーム チェック](#)
- ・ [プリミティブ ピン チェック](#)

ブロック チェック

ブロック チェックでは、NGD 階層内の各ターミナル シンボル (下位コンポーネントに分解されないシンボル) が NGD プリミティブであるかどうかを検証します。NGD プリミティブでない場合は、エラーが発生します。また、ブロック チェックの一部として、シンボルに設定されたユーザー定義プロパティとその値をチェックし、有効であることを確認します。

ネット チェック

ネット チェックでは、デザイン内の各信号に対して NGD プリミティブの出力ピン (ドライバ)、トライステートピン (ドライバ)、入力ピン (ロード) の数を確認します。信号にドライバ (またはトライステートドライバ) およびロードが 1 つもない場合、警告が生成されます。信号に複数の非トライステートドライバか、トライステートドライバと非トライステートドライバの組み合わせがある場合は、エラーが生成されます。ネット チェックの一部として、信号のユーザー定義プロパティとその値をチェックし、有効であることを確認します。

パッド チェック

パッド チェックでは、パッド プリミティブに接続されている各信号が次の規則に従っているかどうかを検証します。

- ・ PAD が入力パッドの場合、パッドに接続されている信号は、次のタイプのプリミティブにのみ接続できます。

- バッファ
- クロック バッファ
- PULLUP
- PULLDOWN
- KEEPER
- BSCAN

入力信号は複数のプリミティブに接続できますが、上記の各タイプ 1 つずつに限られます。たとえば、信号はバッファ プリミティブ 1 つ、クロック バッファ プリミティブ 1 つ、および PULLUP プリミティブ 1 つに接続できますが、1 つのバッファ プリミティブと 2 つのクロック プリミティブには接続できません。また、PULLDOWN プリミティブと PULLUP プリミティブの両方には接続できません。このルールに違反すると、エラーが発生します。また、信号が複数のプルアップまたは複数のプルダウンに接続されている場合や、上記のタイプのプリミティブに接続されていない場合にも警告が表示されます。

- ・ PAD が出力パッドの場合、パッドに接続する信号は次のプリミティブ出力のいずれかにのみ接続できます。

- 1 つのバッファ プリミティブ出力
- 1 つのトライステート プリミティブ出力
- 1 つの BSCAN プリミティブ

さらに、次のプリミティブのいずれかに接続できます。

- 1 つの PULLUP プリミティブ
- 1 つの PULLDOWN プリミティブ
- 1 つの KEEPER プリミティブ

上記以外のプリミティブ出力に接続されている場合は、エラーが発生します。

上記の条件を満たしている場合、出力 PAD 信号は、1 つのクロック バッファ プリミティブ入力または 1 つのバッファ プリミティブ入力、あるいはその両方に接続できます。

- ・ PAD が双方向パッドまたはボンディングされていないパッドの場合、そのパッドに接続された信号は、上記の入力パッドと出力パッドのルールに従っている必要があります。信号がその他のプリミティブに接続されている場合は、エラーが発生します。パッドに接続された信号は、入力信号および出力信号の両方として設定する必要があります。そのように設定しておかないと、警告が表示されます。
- ・ パッドに接続された信号がデザインの最上位シンボルに接続されている場合、その最上位シンボル ピンはパッド ピンと同じタイプにする必要があります。ただし、出力パッドは最上位のトリステートピンに接続できます。このルールに違反すると、警告が表示されます。
- ・ 1 つの信号が複数のパッドに接続されている場合、エラーが表示されます。1 つの信号が複数の最上位ピンに接続されている場合は、警告が表示されます。

クロック バッファ チェック

クロック バッファ チェックでは、各クロック バッファ プリミティブの出力がインバータ、フリップフロップ、ラッチ プリミティブのクロック入力、またはその他のクロック バッファ入力のみに接続されているかどうかを検証します。このルールに違反すると、警告が表示されます。

ネーム チェック

ネーム チェックでは、次の基準に従って、NGD オブジェクト上の名前が重複していないかどうかを検証します。

- ・ ピン名は、シンボル内で重複しないようにします。このルールに違反すると、エラーが表示されます。
- ・ インスタンス名は、その階層レベル内で重複しないようにします。つまり、1 つの階層に同じ名前のシンボルを 2 つ存在させることはできません。このルールに違反すると、警告が表示されます。
- ・ 信号名は、その階層レベル内で重複しないようにします。つまり、1 つのシンボル内に同じ名前の信号を 2 つ存在させることはできません。このルールに違反すると、警告が表示されます。
- ・ グローバル信号名は、デザイン内で重複しないようにします。このルールに違反すると、警告が表示されます。

プリミティブ ピン チェック

プリミティブ ピン チェックでは、特定の NGD プリミティブにあるピンがデザイン内の信号に接続されているかどうかを検証します。

NGDBuild

この章では、NGDBuild プログラムについて説明します。次のセクションが含まれています。

- ・ [NGDBuild の概要](#)
- ・ [NGDBuild の構文](#)
- ・ [NGDBuild のオプション](#)

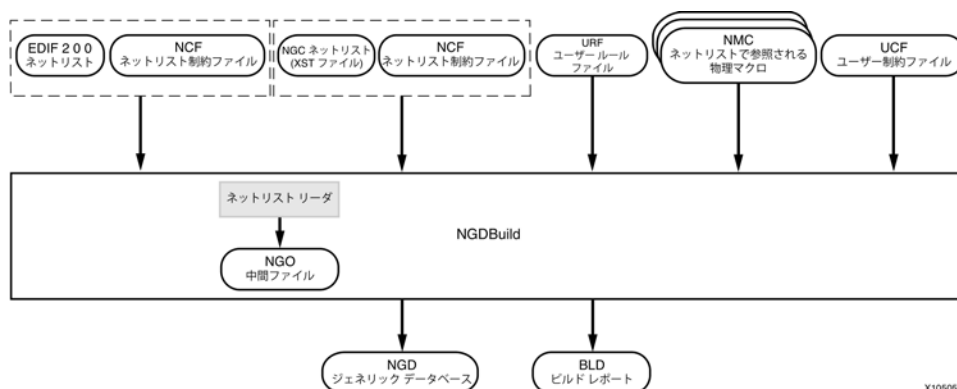
NGDBuild の概要

NGDBuild は、EDIF または NGC フォーマットのネットリストファイルを読み込み、論理デザインが記述された NGD (Native Generic Database) ファイルを生成します。論理デザインは、AND ゲート、OR ゲート、LUT、フリップフロップ、RAM などの論理エレメントで表現されています。

NGD ファイルには、ザイリンクス NGD プリミティブで表現されたデザインの論理記述と、入力ネットリストのオリジナルの階層で表現された記述が含まれています。出力された NGD ファイルは、使用するデバイス ファミリーにマップできます。

次に、NGDBuild のデザインフローを示します。NGDBuild では、この図には表示されていないプログラムも起動されます。

NGDBuild デザイン フロー



NGDBuild のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

ネットリストから NGD ファイルへの変換

NGDBuild は、次の手順でネットリストを NGD ファイルに変換します。

1. ソース ネットリストを読み込みます。

このステップでは、ネットリスト ラウンチャが起動します。ネットリスト ラウンチャにより入力 ネットリストのタイプが判断され、該当するネットリスト リーダ プログラムが起動します。ネットリスト リーダは、各ネットリストに対応する NCF ファイルを読み込みます。NCF ファイルには、各モジュールのタイミング制約とレイアウト制約が含まれています。ネットリスト ラウンチャについては、「[ネットリスト ラウンチャ](#)」を参照してください。

2. デザイン内のすべてのコンポーネントを NGD プリミティブに置き換えます。

ほかのファイルを参照するコンポーネントが結合されます。また、該当するシステムライブラリコンポーネント、物理マクロ (NMC ファイル)、ビヘイビア モデルも検索されます。

3. 変換されたデザインに対し、論理デザイン ルール チェック (DRC) を実行します。

論理的 DRC は、論理デザインに関する一連のテストです。詳細は、「[論理的デザイン ルール チェック](#)」の章を参照してください。

4. NGD ファイルを出力します。

メモ： 上記の変換手順、およびネットリスト ラウンチャ、ネットリスト リーダについては、[付録「EDIF2NGD と NGDBuild」](#)を参照してください。

NGDBuild の入力ファイル

NGDBuild に使用する入力ファイルは、次のとおりです。

入力デザインとして、EDIF 200 または NGC ネットリスト ファイルを使用します。入力ファイルとして別のフォーマットのネットリストが使用されても、そのネットリストがネットリスト ラウンチャで認識されるフォーマットであれば、必要なプログラムを起動してネットリストを EDIF フォーマットに変換してから、該当するネットリスト リーダ EDIF2NGD を起動します。

ネットリスト ラウンチャのデフォルトのオプションでは、次に示す拡張子が付いたファイルが識別され、処理されます。最上位デザイン ネットリストのディレクトリ内で、これらの拡張子の順にネットリスト ファイルが検索されます。デフォルトでは、最初に EDIF ファイルが検索されます。

ファイルの種類	拡張子
EDIF	.sedif、.edn、.edf、.edif
NGC	.ngc

古いネットリストは、ディレクトリからすべて削除してください。古いネットリストがあると、エラーが発生することがあります。

- ・ **UCF ファイル** : ASCII 形式のユーザー制約ファイル。このファイルは、Constraints Editor を使用するか、手動で作成できます。詳細は、Constraints Editor ヘルプを参照してください。UCF には、論理デザインをターゲット デバイスにインプリメントする方法に影響を与えるタイミング制約およびレイアウト制約が含まれます。ファイルに記述された制約は、NGD に出力されます。制約の詳細は、『[制約ガイド](#)』を参照してください。

入力デザイン ファイルとベース名が同じで拡張子が .ucf のファイルが存在する場合、このファイルがデフォルトで UCF として自動的に読み込まれます。**-uc** オプションを使用すると、別の制約ファイルを指定できます。詳細は、『[-uc \(ユーザー制約ファイル\)](#)』を参照してください。

- ・ **NCF ファイル** : CAE ベンダー ツールを使用して生成されるネットリスト制約ファイル。このファイルには、ツールで指定した制約が含まれます。NCF のファイル名が EDIF または NGC ネットリストと同じ場合、ネットリストリーダーにより NCF の制約が読み込まれます。制約は、中間 NGO と出力 NGD に書き込まれます。NCF は、EDIF2NGD によるファイル変換中に NGO に読み込まれ、アノートされます。UCF とは異なり、NCF に対応するネットリストは 1 つだけです。

メモ : NGO のデータが最新のものかどうかチェックされ、EDIF の日付が NGO よりも新しい場合に限り、EDIF2NGD が再実行されます。NCF のアップデートは、EDIF2NGD の再実行には影響しません。したがって、NGO が既にアップデートされており、NCF のみをアップデートした場合、**-nt on** オプションを使用して変更していない EDIF と新しい NCF から NGO を生成し直します。詳細は、『[-nt \(ネットリスト変換タイプ\)](#)』を参照してください。

- ・ **URF ファイル** : ASCII 形式のユーザー制約ファイル。ネットリスト ラウンチャによりこのファイルが読み込まれ、使用可能な入力ネットリスト ファイル、入力ファイルを読み込むネットリストリーダー、およびデフォルトのネットリストリーダー オプションが決定されます。また、デザインを処理するサードパーティのコマンドも指定できます。URF ファイルを使用すると、ルールを追加したり、システム ルール ファイルのルールを変更できます。

ユーザー ルール ファイルの保存先は、**-ur** オプションを使用して指定します。この URF ファイルの拡張子には、.urf を使用する必要があります。詳細は、『[-ur \(ユーザー ルール ファイルの読み込み\)](#)』または付録 B の『[ユーザー ルール ファイル \(URF\)](#)』を参照してください。

- ・ **NGC ファイル** : 最上位デザイン ファイルまたはモジュール ファイルとして使用するバイナリ ファイル。

最上位デザイン ファイル

XST から出力されたファイルです。デザイン ファイルに関する説明を参照してください。

メモ : これは本当のネットリストファイルではありませんが、NGC モジュール ファイルと区別するため、ここではネットリストと呼んでいます。NGC は XST および CORE Generator™ により生成され、NGO は EDIF2NGD により生成されますが、この 2 つのファイルは同等です。

- ・ **NMC ファイル** : デザイン内でインスタンス化されている物理的なマクロのインプリメンテーションを含むバイナリ ファイル。NGDBuild により NMC が読み込まれ、マクロの論理シミュレーションモデルが作成されます。

完全パスを指定しない場合は、次の場所からネットリスト、NCF、NGC、NMC、MEM の各ファイルが検索されます。

- ・ NGDBuild を起動した作業ディレクトリ。
- ・ NGDBuild コマンドラインで指定した最上位デザイン ネットリストのパス。
- ・ NGDBuild コマンドラインで **-sd (指定したディレクトリの検索)** を使用して指定したパス。

NGDBuild の中間ファイル

NGO ファイル : デザインのオリジナルのコンポーネントおよび階層で表現した論理記述を含むバイナリ ファイル。これらのファイルは、NGDBuild により入力 EDIF ネットリストが読み込まれる際に生成されます。ファイルが既に存在する場合は、既存のファイルが読み込まれます。ファイルが存在しない場合や、ファイルが古い場合は、NGDBuild により生成されます。

NGDBuild の出力ファイル

NGDBuild からの出力ファイルは、次のとおりです。

- ・ **NGD ファイル** : オリジナルのコンポーネントと階層、およびデザインを NGD プリミティブレベルで表現したデザインの論理記述を含むバイナリ ファイル
- ・ **BLD ファイル** : NGDBuild の実行と NGDBuild により実行されるサブプロセス (EDIF2NGD と URF で指定されたプログラム) に関する情報が記述されたビルド レポート ファイル。BLD ファイルのルート名は出力 NGD ファイルと同じで、拡張子は **.bld** です。このファイルは、出力 NGD ファイルと同じディレクトリに書き込まれます。

NGDBuild の構文

```
ngdbuild [options] design_name [ngd_file [.ngd]]
```

options : 「[NGDBuild のオプション](#)」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

design_name : 処理する最上位デザインのファイル名を指定します。デザインが適正に変換されるよう、[「NGDBuild の概要」](#)を参照して入力ファイルの拡張子を正しく指定します。間違った拡張子や存在しない拡張子を指定すると、NGD は生成されず、エラーが発生する原因となります。

メモ : 入力デザインとして NGC ファイルを使用する場合は、拡張子 **.ngc** を指定する必要があります。EDIF ネットリストまたは NGO がプロジェクト ディレクトリで検出された場合、NGC は検索されません。

ngd_file : NGD フォーマットの出力ファイルを指定します。出力ファイルの名前、拡張子、および保存場所は、次のように決定されます。

- ・ 出力ファイル名を指定しない場合、出力ファイルのベース名は入力ファイルと同じで、拡張子 **.ngd** が付けられます。
- ・ 拡張子を付けずに出力ファイル名を指定すると、ファイル名に拡張子 **.ngd** が付けられます。
- ・ 拡張子が **.ngd** 以外のファイル名を指定すると、エラー メッセージが表示され、NGDBuild は実行されません。
- ・ 出力ファイルが既に存在する場合、そのファイルは新規ファイルで上書きされます。

NGDBuild のオプション

このセクションでは、NGDBuild のコマンドライン オプションについて説明します。

- ・ `-a` (最上位ポート信号への PAD の追加)
- ・ `-aul` (不一致の LOC を許可)
- ・ `-aut` (不一致のタイムグループを許可)
- ・ `-bm` (BMM ファイルを指定)
- ・ `-dd` (保存先ディレクトリ)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-i` (UCF ファイルを無視)
- ・ `-insert_keep_hierarchy` (KEEP_HIERARCHY 制約の挿入)
- ・ `-intstyle` (統合スタイル)
- ・ `-filter` (フィルタ ファイルの指定)
- ・ `-l` (検索するライブラリ)
- ・ `-nt` (ネットリスト変換タイプ)
- ・ `-p` (製品番号)
- ・ `-quiet` (警告およびエラーのみを表示)
- ・ `-r` (LOC 制約の無視)
- ・ `-sd` (指定したディレクトリの検索)
- ・ `-u` (未展開のブロックを許可)
- ・ `-uc` (ユーザー制約ファイル)
- ・ `-ur` (ユーザー ルール ファイルの読み込み)
- ・ `-verbose` (すべてのメッセージの表示)

`-a` (最上位ポート信号への PAD の追加)

最上位の入力ネットリストが EDIF の場合、`-a` オプションを使用すると、ルートレベルのセルに配置されたポートに接続されているすべての信号に PAD シンボルが追加されます。このオプションは、下位のネットリストには影響しません。

構文

`-a`

`-a` を使用する必要があるかどうかは、ご使用のサードパーティ EDIF ライタによって決まります。EDIF ライタで、ほかのライブラリ コンポーネントと同様にパッドがインスタンスとして処理される場合は、`-a` オプションを使用しないでください。EDIF ライタでパッドが階層ポートとして処理される場合は、実際のパッド シンボルを推論するため、`-a` オプションを使用する必要があります。使用する必要があるのに `-a` を使用しなかった場合、マップ中にロジックが不正に削除されることがあります。Mentor Graphics 社および Cadence 社の回路図入力ツールで生成される EDIF ファイルの場合、`-a` オプションは自動的に設定されるので、入力する必要はありません。

メモ : NGDBuild の **-a** オプションは、EDIF2NGD の **-a** オプションに対応しています。NGDBuild で EDIF2NGD を自動的に実行するのではなく、最上位の EDIF ネットリストに対して EDIF2NGD を個別に実行する場合、**-a** オプションを一貫して使用する必要があります。NGDBuild を既に実行しており、NGO が存在する場合、**-a** オプションを初めて使用する際に **-nt on** オプションも使用する必要があります。これにより NGO が再生成され、EDIF2NGD で **-a** オプションを実行できるようになります。

-aul (不一致の LOC を許可)

デフォルトでは、UCF または NCF ファイルでピン名、ネット名、インスタンス名に指定した制約がデザインに含まれていない場合、エラーが発生し、NGD ファイルは生成されません。このオプションを使用すると、LOC 制約に対してエラーではなく警告が表示され、NGD ファイルが生成されます。

構文

-aul

HDL または回路図で定義していないピン名、ネット名、インスタンス名に設定したロケーション制約が制約ファイルに含まれている場合、**-aul** オプションを使用してプログラムを実行します。これにより、作成中のデザインと完成デザインの両方に、同じ制約ファイルを適用できます。

メモ : このオプションを使用する場合、デザインに含まれるネット名およびインスタンス名にスペルミスがないことを確認してください。スペルミスがあると、配置配線が正しく実行されない場合があります。

-aut (不一致のタイムグループを許可)

デフォルトでは、UCF または NCF ファイルで指定されているタイムグループがデザインに含まれていない場合、NGD ファイルは生成されません。このオプションを使用すると、LOC 制約に対してエラーではなく警告が表示され、NGD ファイルが生成されます。

構文

-aut

HDL または回路図で定義していないタイムグループ制約が制約ファイルに含まれている場合、**-aut** オプションを使用してプログラムを実行します。これにより、作成中のデザインと完成デザインの両方に、同じ制約ファイルを適用できます。

メモ : このオプションを使用する場合、デザインに含まれるタイムグループ名にスペルミスがないことを確認してください。スペルミスがあると、配置配線が正しく実行されない場合があります。

-bm (BMM ファイルを指定)

BMM ファイルを指定します。ファイルの拡張子を指定しない場合は、拡張子が **.bmm** のファイルが使用されます。

構文

-bm *file_name* [**.bmm**]

このオプションを指定しない場合、ルート名が ELF または MEM ファイルと同じで拡張子が .bmm のファイルが使用されます。このオプションのみを指定した場合、BMM ファイルの構文が正しいかどうか、BMM で指定されたインスタンスがデザイン内に存在するかどうかを確認されます。**-bm** オプションは 1 度しか使用できません。

-dd (保存先ディレクトリ)

中間ファイル (デザイン NGO ファイルとネットリスト ファイル) を保存するディレクトリを指定します。このオプションを指定しない場合、ファイルは作業ディレクトリに保存されます。

構文

-dd *NGOoutput_directory*

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f\(コマンド ファイルの実行\)](#)」を参照してください。

-i (UCF ファイルを無視)

UCF ファイルを無視します。デフォルトでは、最上位デザインのディレクトリに入力デザイン ファイルとベース名が同じで拡張子が .ucf のファイルが存在する場合、この UCF ファイルに記述された制約が自動的に読み込まれます。

構文

-i

メモ : このオプションを使用する場合は、**-uc** オプションを使用しないでください。

-insert_keep_hierarchy (KEEP_HIERARCHY 制約の挿入)

各入力ネットリストに KEEP_HIERARCHY 制約が自動的に追加されます。このオプションは、ネットリストが階層ごとに生成される、ボトムアップの合成フローを使用する場合にのみ使用できます。このオプションを使用する場合は、『[合成/シミュレーション デザイン ガイド](#)』で説明している設計手法を使用してください。

構文

-insert_keep_hierarchy

メモ : コアを含むデザインでこのオプションを使用する場合、コアは階層を保持するよう記述されていない可能性があるため、注意が必要です。

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle ise|xflow|silent

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-filter (フィルタ ファイルの指定)

フィルタ ファイルを指定します。フィルタ ファイルには、プログラムの実行中に生成されるメッセージを保存およびフィルタ処理するための設定が含まれています。

構文

-filter [*filter_file*]

デフォルトのフィルタ ファイル名は `filter.filter` です。

-l (検索するライブラリ)

デザイン構築に使用したライブラリ コンポーネントを識別する際に検索されるライブラリのリストを指定します。このオプションは、該当するネットリストリーダーに渡されます。この情報により、デザインのコンポーネントのソースが識別され、コンポーネントが NGD プリミティブに変換されるようになります。

構文

-l {*libname*}

ライブラリを複数指定するには、コマンドラインに **-l libname** を複数入力します。

libname に使用できる値は、次のとおりです。

- ・ **xilinxun** (ザイリンクス ユニファイド ライブラリ)
- ・ **synopsys**

メモ : **-l xilinxun** の指定はオプションです。NGDBuild はこれらのライブラリに自動的にアクセスします。また、ネットリストの拡張子が `.sedif` など、Synopsys のデザインが自動的に検出される場合は、**-l synopsys** の指定はオプションです。

-nt (ネットリスト変換タイプ)

ネットリスト ラウンチャによるタイムスタンプの処理方法を決定します。タイムスタンプは、ファイルが作成された日時を示す情報です。

構文

-nt timestamp|on|off

timestamp (デフォルト) : ネットリスト ラウンチャにより通常のタイムスタンプ チェックが実行され、タイムスタンプに応じて NGO がアップデートされます。

on : タイムスタンプにかかわらず、ネットリストが変換されます (すべての NGO を再構築)。

off : タイムスタンプにかかわらず、既存の NGO は再構築されません。

-p (製品番号)

デザインをインプリメントするパーツを指定します。

構文

-p *part_number*

メモ : 構文の詳細および例は、「はじめに」の章の「[-p \(製品番号\)](#)」を参照してください。

このオプションを使用定すると、そのアーキテクチャにマップするように最適化された NGD ファイルが生成されます。

回路図で PART プロパティを指定したり、UCF に CONFIG PART 文を含むなどして、NGO 内に既にベンダーやデバイス ファミリーに関する情報が記述されている場合は、**-p** オプションを指定する必要はありません。ただし、NGDBuild を実行する際に **-p** オプションを指定すると、NGO ファイルの情報を無効にできます。

-quiet (警告およびエラーのみを表示)

警告メッセージおよびエラーメッセージのみを表示します。

構文

-quiet

-r (LOC 制約の無視)

入力ネットリストまたは UCF に記述されたすべてのロケーション制約 (LOC=) が無視されます。あるアーキテクチャ内のロケーションが別のアーキテクチャ内のロケーションと一致しない場合があるので、異なるデバイスまたはアーキテクチャに移行する際は、このオプションを使用します。

構文

-r

-sd (指定したディレクトリの検索)

ファイル参照 (回路図で FILE=*filename* プロパティを使用して指定されたファイル) を解決する際、およびネットリスト、NGO、NGC、NMC、MEM の各ファイルを検索する際に、検索するディレクトリのリストに指定したパス (*search_path*) を追加します。最上位デザイン ネットリストのディレクトリは、NGDBuild により自動的に検索されるので、検索パスとして指定する必要はありません。

構文

-sd {*search_path*}

-sd オプションと *search_path* の間には、スペースまたはタブを挿入します。たとえば、「**-sd**designs」ではなく、「**-sd designs**」と指定します。各パスの前に **-sd** を付けて、1 つのコマンドラインに複数の検索パスを指定できます。1 つの **-sd** の後に複数の検索パスを指定することはできません。たとえば、2 つの検索パスを指定するには、次のように指定します。

```
-sd /home/macros/counter -sd /home/designs/pal2
```

次のように指定することはできません。

```
-sd /home/macros/counter /home/designs/pal2
```

-u (未展開のブロックを許可)

デフォルト (**-u** オプションなし) では、デザインのブロックが NGD プリミティブに展開できないとエラーが発生します。このエラーが発生すると、NGD ファイルは生成されません。このオプションを使用すると、ブロックを展開できない場合にエラーではなく警告が表示され、未展開のブロックを含む NGD ファイルが生成されます。

構文

-u

デザインが完成していない段階で、マップ、配置配線、タイミング解析、シミュレーションなどを実行するには、**-u** オプションを使用して NGDBuild を実行します。デザインをマップする際に未展開のブロックを保持するには、**-u** オプション (未使用のロジックを削除しない) を使用して MAP を実行します。詳細は、「[MAP](#)」の章を参照してください。

-uc (ユーザー制約ファイル)

ネットリストラウンチャで読み込む UCF を指定します。UCF には、論理デザインのターゲットデバイスへのインプリメント方法に影響するタイミング制約とレイアウト制約が含まれています。

-uc オプションは複数指定できます。複数の UCF ファイルは、コマンドラインに入力した順序で処理されます。

メモ : このオプションを使用する場合は、**-i** オプションを使用しないでください。

構文

```
-uc ucf_file [.ucf]
```

ucf_file : UCF ファイルの名前を指定します。UCF ファイルの拡張子は **.ucf** です。拡張子を指定しない場合、拡張子 **.ucf** が追加されます。拡張子が **.ucf** 以外のファイル名を指定すると、エラーメッセージが表示され、NGDBuild は実行されません。

-uc オプションを使用しない場合でも、入力デザインファイルと同じベース名で拡張子が **.ucf** の UCF が存在すれば、UCF に記述された制約が自動的に読み込まれます。

制約の詳細は、『[制約ガイド](#)』を参照してください。

-ur (ユーザー ルール ファイルの読み込み)

ネットリストラウンチャがアクセスするユーザー ルール ファイルを指定します。このファイルは、有効なネットリスト入力ファイル、これらのファイルを読み込むネットリストリーダー、およびネットリストリーダーのデフォルトのオプションを指定します。また、デザインを処理するサードパーティのコマンドも指定できます。

構文

```
-ur rules_file [.urf]
```

このファイルの拡張子は .urf にする必要があります。拡張子を付けずにユーザー ルール ファイル名を指定すると、ファイル名に .urf の拡張子が追加されます。.urf 以外の拡張子の付いたファイル名を指定すると、エラー メッセージが表示され、NGDBuild は実行されません。

詳細は、付録 B の「[ユーザー ルール ファイル \(URF\)](#)」を参照してください。

-verbose (すべてのメッセージの表示)

NGDBuild、ネットリスト ラウンチャ、ネットリストリーダの実行により出力されるすべてのメッセージを画面に表示します。このオプションは、ツールの実行に関する詳細情報が必要な場合に便利です。

構文

```
-verbose
```


MAP

この章では、インプリメンテーション プロセスで論理デザインをザイリンクス FPGA デバイスにマップする MAP プログラムについて説明します。次のセクションが含まれています。

- ・ [MAP の概要](#)
- ・ [MAP のプロセス](#)
- ・ [MAP の構文](#)
- ・ [MAP のオプション](#)
- ・ [再合成および物理合成最適化](#)
- ・ [ガイド マップ](#)
- ・ [マップ結果のシミュレーション](#)
- ・ [MAP レポート \(MPR\) ファイル](#)
- ・ [物理合成レポート \(PSR\) ファイル](#)
- ・ [MAP の停止](#)

MAP の概要

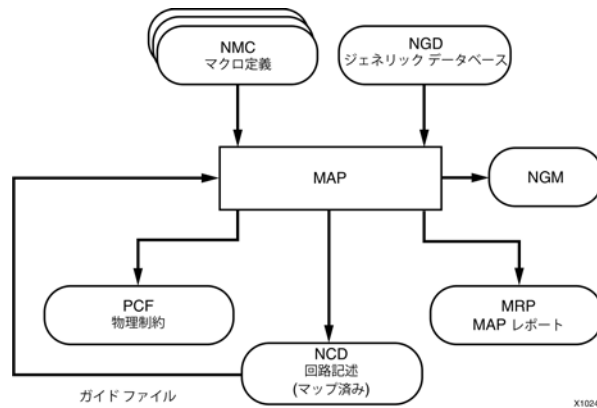
MAP は、論理的デザインをザイリンクス FPGA にマップするプログラムです。MAP には、NGDBuild プログラムで生成された NGD ファイルを入力します。NGD ファイルは、デザイン作成に使用した階層コンポーネントおよび下位のザイリンクス プリミティブで表現したデザインの論理記述、および物理的なマクロの定義を含む NMC (マクロ ライブラリ) から構成されます。使用するオプションによっては、MAP で配置も実行されます。

MAP は、まず NGD ファイルのデザインに対して論理的 DRC (デザイン ルール チェック) を実行します。その後、ターゲットのザイリンクス FPGA 内のコンポーネント (ロジック セル、I/O セル、その他のコンポーネント) にロジックをマップします。

デザインは、ザイリンクス FPGA 内のコンポーネントにマップされたデザインの物理記述である NCD (Native Circuit Description) ファイルに出力されます。NCD ファイルは、配置配線に使用されます。

次に、MAP のデザイン フローを示します。

MAP デザイン フロー



MAP のデバイス サポート

このプログラムは、次のデバイス ファミリーで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6

MAP の入力ファイル

MAP に使用する入力ファイルは次のとおりです。

- ・ **NGD ファイル** : Native Generic Database の略。デザインが最初に作成されたときに使用された階層と、階層が分割される下位のザイリンクス プリミティブが論理的に記述されています。また、デザイン入力でデザインに設定した制約、ユーザー制約ファイル (UCF) に入力された制約がすべて含まれます。このファイルは、NGDBuild により生成されます。
- ・ **NMC ファイル** : 物理的なマクロの定義が含まれたマクロ ライブラリ ファイル。NGD ファイルにマクロ インスタンスがある場合、NMC ファイルがマクロ インスタンスの定義に使用されます。デザイン ファイル内のマクロの各タイプに対して、NMC ファイルが 1 つずつ存在します。
- ・ **ガイド NCD ファイル** : 以前の MAP 実行で生成されたファイル (オプション)。NCD には、ターゲット デバイス内のコンポーネントレベルで表現したデザインの物理記述が含まれます。このファイルを入力として使用すると、これから実行する MAP のガイドとして使用されます。
- ・ **ガイド NGM ファイル** : オプションの入力デザイン ファイル。NGD ファイルに記述されたすべてのデータと、マップにより生成された物理デザインに関する情報を含むバイナリ形式のファイルです。詳細は、「[ガイド マップ](#)」を参照してください。
- ・ **アクティビティ ファイル** : オプションの入力ファイル。MAP では、.saif および .vcd の 2 種類のアクティビティ ファイルがサポートされています。

MAP の出力ファイル

MAP からの出力ファイルは、次のとおりです。

- ・ **NCD (Native Circuit Description) ファイル** : ターゲットのザイリンクス デバイスのコンポーネントで表現したデザインの物理記述です。出力ファイルの名前と場所については、「[-o \(出力ファイル名\)](#)」を参照してください。
- ・ **PCF (Physical Constraints File) ファイル** : デザイン入力で指定した、物理エレメントレベルの制約を含む ASCII 形式のテキスト ファイル。PCF の物理制約は、ザイリンクスの制約言語で記述されています。

MAP は、PCF が存在しない場合は PCF を生成し、PCF が存在する場合は回路図で生成されたセクション (SCHEMATIC START と SCHEMATIC END の間) を上書きします。既存の PCF に対しては、ユーザーが作成したセクションについても構文エラーをチェックし、エラーを検出すると処理を停止します。ユーザーが作成したセクションでエラーが検出されなかった場合は、変更されません。

- ・ **NGM ファイル** : 入力 NGD ファイルのすべてのデータと、マップにより生成された物理デザインに関する情報を含むバイナリ形式のデザイン ファイルです。NGM は、バックアノテートされたデザイン ネットリストをソース デザインの構造および名称と照合する際に使用します。このファイルは SmartGuide™ テクノロジーでも使用されます。
- ・ **MRP ファイル** : MAP の実行に関する情報が含まれたレポート ファイルです。MRP には、デザインで検出されたエラーと警告、設定されたデザイン制約がリストされ、削除または追加されたロジックや、論理的デザインの信号とシンボルが物理デザインの信号とコンポーネントにどのようにマップされたかなど、デザインのマップ方法に関する詳細が示されます。マップされたデザインにおけるコンポーネントの使用状況に関する統計情報も含まれます。詳細は、「[MAP レポートファイル \(MRP\)](#)」を参照してください。
- ・ **MAP ファイル** : MAP 実行中の出力が記述されたログ ファイルです。これに対し、レポート ファイル (MRP) は、MAP の実行が完了した後に作成され、フォーマットされています。
- ・ **PSR ファイル** : MAP の物理合成オプションで実行された最適化の詳細を含む物理合成レポート ファイルです。物理合成オプションは、**-global_opt**、**-register_duplication**、**-retiming**、**-equivalent_register_removal**、**-logic_opt**、および **-register_duplication** です。このレポートは、これらのオプションのいずれかを使用した場合にのみ生成されます。

MAP の実行で生成される MRP、MAP、PCF、および NGM ファイルは、すべて出力 NCD ファイルとルート名が同じで、適切な拡張子が付けられます。MRP、MAP、PCF、または NGM ファイルが既に存在する場合は、上書きされます。

MAP のプロセス

デザインをマップする際は、MAP により次の処理が実行されます。

1. ターゲットのザイリンクス デバイス、パッケージおよびスピードを選択します。MAP では、次のようにパーツが選択されます。
 - ・ MAP コマンドラインで指定したパーツを使用します。
 - ・ コマンドライン上でパーツが指定されていない場合は、入力 NGD ファイルで指定されたパーツを選択します。入力 NGD ファイルでアーキテクチャ、デバイス、パッケージの情報が不完全な場合は、エラー メッセージが表示され、MAP は停止します。必要に応じて、デフォルトのスピードが使用されます。
2. 入力デザイン ファイル内の情報を読み込みます。

3. 入力デザインに対し、論理的 DRC (デザイン ルール チェック) を実行します。DRC エラーが検出されると、MAP の実行は中止されます。DRC 警告が検出されると、警告が表示されますが、MAP の実行は続行されます。論理的 DRC については、「[論理的デザイン ルール チェック](#)」の章を参照してください。

メモ: NGDBuild の DRC でエラーが検出されなかった場合、ステップ 3 は省略されます。

4. 未使用ロジックを削除します。未使用のコンポーネントおよびネットは、次の場合を除きすべて削除されます。
 - ・ デザイン入力でネットに S (保存) 制約を設定した場合。未使用ネットに S 制約を指定すると、そのネットとネットに接続されているすべての使用ロジック (ドライバまたはロード) が保持されます。ネットに接続されている未使用ロジックはすべて削除されます。S 制約については、「[制約ガイド](#)」を参照してください。
 - ・ MAP コマンドラインで `-u` オプションを指定した場合。`-u` オプションを指定すると、未使用ロジックがすべて保持されます。
5. パッドと関連ロジックを IOB 内にマップします。
6. IOB やスライスなどのザイリンクス コンポーネントにロジックをマップします。マップ処理は、さまざまな制約に影響されます。制約については、「[制約ガイド](#)」を参照してください。
7. 入力 NGD ファイルからの情報をアップデートし、NGM ファイルに記述します。NGM ファイルには、デザインに関する論理情報とデザインのマップ方法に関する物理情報の両方が記述されています。NGM は、バックアノテーションでのみ使用されます。詳細は、「[ガイド マップ](#)」を参照してください。
8. 物理制約ファイル (PCF) を作成します。PCF ファイルは、デザイン入力中に指定したすべての制約を含むテキスト ファイルです。デザイン入力で制約を設定しなかった場合、空のファイルが生成されるので、テキスト エディタでファイルに制約を直接入力するか、FPGA Editor を使用して制約を入力できます。

MAP は、PCF ファイルが存在しない場合は PCF を生成し、PCF が存在する場合は回路図で生成されたセクション (SCHEMATIC START と SCHEMATIC END の間) を上書きします。既存の制約ファイルに対しては、ユーザーが作成したセクションもチェックし、エラーのある制約をコメントとして記述するか、処理を停止します。ユーザーが作成したセクションにエラーがない場合、そのセクションは変更されません。
9. Spartan®-3 および Virtex®-4 以外のアーキテクチャでは、デザインを自動的に配置します。Spartan-3 および Virtex-4 デバイスで MAP で配置を実行するには、**`-timing`** オプションを使用します。
10. マップ済みのデザインに対し、物理デザイン ルール チェック (DRC) を実行します。DRC エラーが発生した場合、NCD ファイルは生成されません。
11. 物理デザインを表す NCD ファイルを作成します。NCD には、CLB、IOB などザイリンクスのコンポーネントレベルでデザインが記述されます。
12. MAP レポート ファイル (MRP) を作成します。このレポートには、エラーと警告、デザインの詳細なマップ状況、コンポーネントの使用率などがリストされます。

MAP の構文

デザインをマップするには、次の構文を使用します。

```
map [options] infile[.ngd] [pcf_file.pcf]
```

options : 「MAP のオプション」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

infile : 入力 NGD ファイルを指定します。拡張子 `.ngd` を入力する必要はありません。

pcf file : 出力 PCF ファイルの名前を指定します。PCF のファイル名と保存先は、次のように決定されます。

- ・ コマンドラインで PCF ファイル名を指定しない場合、PCF ファイル名は出力ファイルと同じ名前で拡張子は `.pcf` になります。ファイルは、出力ファイルのディレクトリに保存されます。
- ・ 完全パスを指定せずに PCF ファイル名のみを指定すると (`/home/designs/cpu_1.pcf` ではなく `cpu_1.pcf` など)、PCF ファイルが作業ディレクトリに保存されます。
- ・ PCF ファイルの完全パス (`/home/designs/cpu_1.pcf` など) を指定すると、PCF ファイルは指定したディレクトリに保存されます。
- ・ PCF ファイルが既に存在している場合、MAP はそのファイルを読み込んで構文エラーをチェックし、ファイルの回路図で生成されたセクションを上書きします。また、ユーザーが作成したセクションについてもエラーをチェックし、ファイル内の物理制約をコメントアウトするか、処理を停止してエラーを修正します。ユーザーが作成したセクションでエラーが検出されなかった場合は、変更されません。

メモ : 出力ファイルの名前と場所については、「[-o \(出力ファイル名\)](#)」を参照してください。

MAP のオプション

このセクションでは、MAP のコマンドライン オプションについて説明します。

- ・ `-activity_file` (アクティビティ ファイルの指定)
- ・ `-bp` (スライス ロジックのマップ)
- ・ `-c` (スライスのパック)
- ・ `-cm` (カバー モード)
- ・ `-detail` (詳細な MAP レポートの生成)
- ・ `-equivalent_register_removal` (重複したレジスタの削除)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-global_opt` (グローバル最適化)
- ・ `-ignore_keep_hierarchy` (KEEP_HIERARCHY を無視)
- ・ `-intstyle` (統合スタイル)
- ・ `-ir` (RPM の生成に RLOC を使用しない)
- ・ `-filter` (フィルタ ファイルの指定)
- ・ `-lc` (LUT の結合)
- ・ `-logic_opt` (ロジックの最適化)
- ・ `-mt` (マルチスレッド)
- ・ `-ntd` (非タイミング ドリブン)
- ・ `-o` (出力ファイル名)
- ・ `-ol` (総体的なエフォートレベル)
- ・ `-p` (製品番号)
- ・ `-power` (消費電力の最適化)
- ・ `-pr` (レジスタを I/O にパック)
- ・ `-register_duplication` (レジスタの複製)
- ・ `-retiming` (グローバル最適化中のレジスタのリタイミング)
- ・ `-smartguide` (SmartGuide)
- ・ `-t` (配置コスト テーブル)
- ・ `-timing` (タイミング ドリブンのパックと配置)
- ・ `-u` (未使用ロジックを保持)
- ・ `-w` (既存ファイルの上書き)
- ・ `-x` (パフォーマンス評価モード)
- ・ `-xe` (追加エフォートレベル)
- ・ `-xt` (追加配置コスト テーブル)

`-activity_file` (アクティビティ ファイルの指定)

消費電力最適化用にスイッチ アクティビティ データ ファイルを指定します。

構文

```
-activity_file activity_file.{vhdl|saif}
```

MAP では、.saif および .vcd の 2 種類のアクティビティファイルがサポートされています。

メモ：このオプションは、すべての FPGA アーキテクチャでサポートされています。

メモ：このオプションは、**-power on** オプションを指定している場合にのみ有効です。詳細は、「[-power \(消費電力の最適化\)](#)」を参照してください。

-bp (スライス ロジックのマッピング)

ブロック RAM マッピングをイネーブルにします。

ブロック RAM マッピングをイネーブルにすると、LUT およびフリップフロップが、出力が 1 つのシングルポートのブロック RAM に配置されます。

ブロック RAM 出力に変換するレジスタ出力ネットのリストを含むファイルを生成できます。MAP でこのファイルを使用するには、環境変数 XIL_MAP_BRAM_FILE をファイル名に設定します。**-bp** オプションを指定するとこの環境変数が検索され、ファイルにリストされている出力ネットのみがブロック RAM 出力にマッピングされます。ブロック RAM 出力は同期しており、リセットしかできないので、ブロック RAM にパックするレジスタも同期リセットであることが必要です。

メモ：エリアグループ制約が設定された LUT は、ブロック RAM には配置されません。ブロック RAM にパックするロジックは、エリアグループには含めないでください。

構文

```
-bp
```

-c (スライスのパック)

デザインをマッピングする際のスライスの使用率を指定し、必要に応じて無関連のロジックが同じスライスにパックされるようにします。

メモ：スライスのパックおよび圧縮は、**-timing** (タイミングドリブンのパックと配置) を使用している場合は使用できません。

構文

```
-c [packfactor]
```

-c が使用されていない場合および **-c** に値が指定されていない場合の *packfactor* のデフォルト値は 100 です。

- ・ Spartan®-3, Spartan-3A, Spartan-3E, および Virtex®-4 デバイスで **-timing** が使用されていない場合、*packfactor* に有効な値は 0 ~ 100 です。
- ・ Spartan-3, Spartan-3A, Spartan-3E, および Virtex-4 デバイスで **-timing** が使用されている場合、*packfactor* に有効な値は 0、1、または 100 です。
- ・ Spartan-6, Virtex-5, および Virtex-6 デバイスでは **-timing** が常に使用され、*packfactor* に有効な値は 1 または 100 のみです。

メモ：これらのアーキテクチャでスライスのパック密度を増加させるには、**-lc** (LUT の結合) も使用してみてください。

packfactor を 0 以外の値にした場合、その値がスライスの使用率の目標になります。

- ・ *packfactor* を 0 にすると、関連ロジック (共通の信号を持つロジック) のみが同じスライスにパックされるので、デザインの密度が最小になります。
- ・ *packfactor* を 1 にすると、スライスの使用率が 1% にすることが目標となるので、パック密度が最大になります。
- ・ *packfactor* を 100 にすると、ターゲット デバイスのリソース使用率が 100% になるように無関連のロジックがパックされ、パック密度は最小になります。

packfactor を 1 ~ 100 にすると、デザインに必要なリソースがスライス使用率の目標を超えている場合に限り、無関連のロジックが同じスライスに配置されます。デバイスにデザインをフィットさせるために無関連のロジックを同じスライスにパックする必要がある場合は、**-c 100** を指定した場合に使用されるスライスの数と **-c 0** を指定した場合に使用されるスライスの数が同じになります。

packfactor を小さくするとデザインの密度は高くなりますが、配置配線が困難になる可能性があります。無関連のロジックをパックすると、スライスで配置の競合が発生したり、密度が高くなることによりローカル配線が密集します。

メモ : **-c 1** は、デザインをパックできる最大密度 (最小エリア) を確認する目的でのみ使用してください。実際のデザイン インプリメンテーションにはお勧めしません。通常、最大密度でパックされたデザインの場合、ランタイムが長く、PAR で配線が密集する原因となり、デザインのパフォーマンスが低下します。

-c 0 オプションでデザインを処理すると、デザインに必要なスライス数の初期見積もりができます。

-cm (カバー モード)

MAP のカバー フェーズで使用する基準を指定します。

メモ : このオプションは、Spartan®-6 および Virtex®-6 では使用できません。

構文

-cm [area|speed|balanced]

カバー フェーズでは、ロジックを CLB のファンクション ジェネレータ (LUT) に割り当てます。設定可能な値は area、speed、または balanced で、次のように使用します。

area : LUT 数 (つまりは CLB 数) が最小になるようマップされます (デフォルト)。

speed : ユーザー指定のタイミング制約が使用されているかどうかによって結果が異なります。ユーザー指定の制約が使用されているデザインでは、まずタイミング制約を満たすことを最優先とし、次に LUT のレベル数 (パスが通過する LUT 数) が最小になるようマップされます。ユーザー指定の制約が使用されていないデザインでは、まず最大システム周波数を達成することを最優先とし、次に LUT のレベル数が最小になるようマップされます。この設定を使用すると、デザインを配置配線した後、タイミング制約を簡単に達成できるようになります。ほとんどのデザインでは area と比較して LUT 数は少ししか増加しませんが、デザインによっては大幅に増加することがあります。

balanced : タイミング制約を満たす設定と LUT 数を最小にする設定のバランスを取ります。結果は speed に設定した場合に似ていますが、LUT 数の大幅な増加を回避できます。ユーザー指定の制約が設定されたデザインでは、タイミング制約を満たすと共に LUT 数が最小になるようマップされます。ユーザー指定の制約が設定されていないデザインでは、最大システム周波数を達成すると共に LUT 数が最小になるようマップされます。

-detail (詳細な MAP レポートの生成)

詳細な MAP レポートを生成するよう指定します。

構文

-detail

-detail オプションを使用すると、MAP レポート ファイルに DCM と PLL コンフィギュレーション データを示すセクション 12 (「Configuration String Details」) およびコントロール セットの情報を示すセクション 13 (「Control Set Information」、Virtex-5 のみ) が記述されます。

-equivalent_register_removal (重複したレジスタの削除)

重複するレジスタを削除します。

メモ : このオプションは、Spartan®-6、Virtex®-6、Virtex-5、および Virtex-4 デバイスでのみ使用できます。

構文

-equivalent_register_removal {on|off}

on に設定すると、重複した機能があるレジスタを削除した場合にクロック周波数が向上するかどうか調べられます。デフォルトでは **on** に設定されています。

メモ : **-global_opt** (グローバル最適化) を設定した場合にのみ使用できます。

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f\(コマンド ファイルの実行\)](#)」を参照してください。

-global_opt (グローバル最適化)

ネットリストに対してグローバル最適化を実行してからマップを実行するよう指定します。

メモ : このオプションは、Spartan®-6、Virtex®-6、Virtex-5、および Virtex-4 デバイスでのみ使用できます。

構文

-global_opt off|speed|area|power

off : グローバル最適化を実行しません (デフォルト)。

speed : スピードを最適化します。

area : エリアを最小にするよう最適化します (Virtex-4 デバイスでは使用不可)。

power : 消費電力を最小にするよう最適化します (Virtex-4 デバイスでは使用不可)。

グローバル最適化には、ロジックの再マップおよび削除、ロジックおよびレジスタの複製と最適化、トライステート バッファの再配置などが含まれます。これらの作業に余分に時間がかかるため、MAP のランタイムが長くなります。デフォルトでは off に設定されています。

メモ: **-global_opt power** オプションを使用する場合、**-activityfile** オプションでアクティビティ データを供給できます。

-u オプションは、**-global_opt** と共に使用できます。SmartGuide™ (**-smartguide**) をイネーブルにすると、ガイドされる割合は減少します。

メモ: **-global_opt** との使用については、「**-equivalent_register_removal** (重複したレジスタの削除)」および「**-retiming** (グローバル最適化中のレジスタのリタイミング)」を参照してください。「再合成および物理合成最適化」も参照してください。

-ignore_keep_hierarchy (KEEP_HIERARCHY を無視)

ブロックに設定された KEEP_HIERARCHY プロパティを無視します。

構文

```
-ignore_keep_hierarchy
```

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

```
-intstyle ise|xflow|silent
```

次のいずれかのモードに設定します。

- ・ **-intstyle ise**: プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow**: プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent**: 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ: Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-ir (RPM の生成に RLOC を使用しない)

RLOC の処理方法を指定します。

構文

```
-ir all|off|place
```

all: すべての RLOC 処理をディスエーブルにします。

off: すべての RLOC 処理をイネーブルにします。

place: RLOC 制約を使用してロジックがスライスにグループ化されますが、スライスの相対的な配置を制御する RPM (相対配置マクロ) は生成されません。

-filter (フィルタ ファイルの指定)

フィルタ ファイルを指定します。フィルタ ファイルには、プログラムの実行中に生成されるメッセージを保存およびフィルタ処理するための設定が含まれています。

構文

-filter [*filter_file*]

デフォルトのフィルタ ファイル名は `filter.filter` です。

-lc (LUT の結合)

2 つの LUT コンポーネントを 1 つの LUT6 サイトに配置します。この際、このサイトのデュアル出力ピンが使用されます。

メモ : このオプションは、Spartan®-6、Virtex®-6、および Virtex-5 デバイスでのみ使用できます。

構文

-lc `auto|area|off`

area : 可能な限り LUT を結合します。

auto : エリアとパフォーマンスのバランスを取ります。Spartan-6 デバイスでは、これがデフォルトです。

off : LUT の結合を実行しません。Virtex-6 および Virtex-5 デバイスでは、これがデフォルトです。

-logic_opt (ロジックの最適化)

タイミングおよびデザインのパフォーマンスを向上させるため、配置後にロジックの再構築を実行します。

構文

-logic_opt `on|off`

このオプションを設定すると、配置後のネットリストに対してロジックの再構築および再合成を実行することにより、タイミングがクリティカルな接続の最適化が試みられます。その後インクリメンタル配置およびインクリメンタル タイミング解析が実行され、最終的に完全に配置され、タイミングが最適化された NCD デザイン ファイルが生成されます。このオプションを使用するには、タイミングドリブン パックと配置 (**-timing** オプション) をイネーブルにしておく必要があります。SmartGuide™ (**-smartguide**) をイネーブルにすると、ガイドされる割合は減少します。

メモ : 「再合成および物理合成最適化」も参照してください。

-mt (マルチスレッド)

MAP で複数のプロセッサを使用できるようにし、配置ツールにマルチスレッド機能を含めます。

メモ : このオプションは、Spartan®-6、Virtex®-6、および Virtex-5 デバイスでのみ使用できます。

構文

-mt `off|2`

デフォルトは **off** です。**off** に設定すると、1 つのプロセッサのみが使用されます。**2** に設定すると、使用可能であれば 2 つのコアが使用されます。

-ntd (非タイミングドリブン)

タイミングドリブンでない配置を実行します。

構文

-ntd

このオプションをイネーブルにすると、すべてのタイミング制約が無視され、デザインの配置配線にタイミング情報は使用されません。

メモ：フロー全体をタイミング制約なしで実行する場合は、**-ntd** オプションを MAP および PAR の両方で指定してください。

-o (出力ファイル名)

デザインの出力 NCD ファイル名を指定します。

構文

-o outfile[.ncd]

拡張子 `.ncd` は省略できます。出力ファイルの名前と保存先は、次のように決定されます。

- ・ 出力ファイル名を指定しない場合、出力ファイルの名前は入力ファイルと同じで、拡張子が `.ncd` になります。ファイルは、入力ファイルのディレクトリに保存されます。
- ・ 完全パスを指定せずに出力ファイル名のみを指定すると (`/home/designs/cpu_dec.ncd` ではなく `cpu_dec.ncd` など)、NCD ファイルは作業ディレクトリに保存されます。
- ・ 出力ファイルの完全パス (`/home/designs/cpu_dec.ncd` など) を指定すると、出力ファイルが指定したディレクトリに保存されます。
- ・ 出力ファイルが既に存在する場合は、新規 NCD ファイルで上書きされます。ファイルの上書きの際、警告メッセージは表示されません。

メモ：入力ファイルでパッドに接続された信号およびフリップフロップ、ラッチ、RAM の各出力に接続された信号は、バックアノテーションのため保持されます。

-ol (総体的なエフォート レベル)

MAP の総体的なエフォート レベルを指定します。エフォート レベルは、配置に CPU の使用量が多いまたは少ないアルゴリズムを選択することにより、パックおよび配置にかかる時間を制御します。

構文

-ol std|high

- ・ **std**：エフォート レベルを低に設定します (ランタイム最短、QoR 低)。
- ・ **high**：エフォート レベルを高に設定します (QoR 最良、ランタイム最長)。

デフォルトのエフォート レベルは、すべてのアーキテクチャで **high** です。

このオプションは、**-timing** オプションを使用してタイミングドリブンのパックと配置を実行する場合にのみ使用できます。

メモ：MAP のエフォート レベルは、PAR のエフォート レベル以上にすることをお勧めします。

例

```
map -timing -ol std design.ncd output.ncd design.pcf
```

この例では、MAP の総体的なエフォートレベルを **std** (ランタイム最短、QoR 低) に設定します。

-p (製品番号)

デザインをインプリメントするパーツを指定します。

構文

```
-p part_number
```

メモ：構文の詳細および例は、「はじめに」の章の「[-p \(製品番号\)](#)」を参照してください。

製品番号を指定しない場合、入力 NGD ファイルで指定したパーツが選択されます。入力 NGD ファイルでデバイスおよびパッケージの完全な情報を指定していない場合、このオプションを使用してデバイスとパッケージの仕様を入力する必要があります。必要に応じて、デフォルトのスピード値が使用されます。

入力 NGD ファイルで指定したアーキテクチャと同じアーキテクチャを指定する必要があります。NGDBuild の実行時や、ISE® Design Suite でアーキテクチャを指定した場合、合成ツールのオプションとしてアーキテクチャを指定した場合など、デザイン入力プロセスの初期段階でアーキテクチャを選択している場合があります。アーキテクチャが一致していない場合、NGDBuild を再実行して、アーキテクチャを指定し直す必要があります。

-power (消費電力の最適化)

消費電力を削減するよう配置を最適化します。Virtex®-6 デバイスでは、**high** および **xe** に設定することにより、新しいアルゴリズムを使用してさらに消費電力を削減するよう指定できます。

構文

```
-power on|off|high|xe
```

off：ランタイム、メモリ、パフォーマンスに悪影響を与える消費電力の最適化は実行されません。これがデフォルトです。

on (標準)：配置中に消費電力最適化アルゴリズムを使用して、データおよびクロック ネットの容量付加を削減することにより、全体的なダイナミック消費電力を削減します。この設定を使用すると、ランタイムが増加し、配置が変更されることによりパフォーマンスが多少低下する可能性があります。この設定は、すべてのアーキテクチャで使用できます。

high：クロック ゲート アルゴリズムを使用して、全体的な信号のスイッチを削減することによりデザインのダイナミック消費電力を削減します。このオプションを使用すると、ランタイム、エリア (少量)、システム メモリ要件が増加し、データ パスまたは制御パスにロジックが追加されることによりパフォーマンスが低下する可能性があります。消費電力の削減量は通常 **on** (標準) に設定した場合よりもかなり大きくなります。この設定は、Virtex-6 デバイスでのみ使用できます。

xe (追加エフォート)：ダイナミック消費電力を最大限に削減するため、標準アルゴリズムと high アルゴリズムの両方を使用します。この設定を使用すると、通常ランタイム、エリア、メモリの使用量、パフォーマンスへの影響が最も大きくなります。この設定は、デザインに十分なタイミング スラックがあり、ランタイムおよびメモリ使用量の増加が許容される場合にのみ使用することをお勧めします。この設定は、Virtex-6 デバイスでのみ使用できます。

-power on を使用すると、消費電力をさらに最適化するためスイッチ アクティビティ ファイルも指定できます。詳細は、「[-activity_file \(アクティビティ ファイルの指定\)](#)」を参照してください。

-power オプションを **-global_opt power** と共に使用することにより、さらに消費電力を最適化できます。詳細は、「[-global_opt \(グローバル最適化\)](#)」を参照してください。

-pr (レジスタを I/O にパック)

レジスタを I/O 内に配置します。

構文

-pr off|i|o|b

デフォルト (**-pr** オプションなし) では、合成ツールまたはユーザー制約ファイル (UCF) でレジスタに IOB = TRUE 属性が適用されている場合のみ、I/O コンポーネント内にフリップフロップまたはラッチが配置されます。**-pr** オプションを使用すると、フリップフロップまたはラッチを I/O コンポーネント内に配置するよう指定していない場合でも、入力レジスタ (**i** を指定)、出力レジスタ (**o** を指定)、またはその両方 (**b** を指定) にパックできます。このオプションを指定しない場合、デフォルトで off に設定されます。レジスタに IOB プロパティ (TRUE または FALSE) が設定されている場合、**-pr** オプションよりも優先されます。

-register_duplication (レジスタの複製)

レジスタを複製します。

構文

-register_duplication on|off

このオプションは、**-timing** オプションを使用してタイミングドリブンのパックと配置を実行する場合にのみ使用できます。**-register_duplication** オプションを使用すると、レジスタの複製を生成してタイミングを向上できます。「[-timing \(タイミングドリブンのパックと配置\)](#)」を参照してください。

-retiming (グローバル最適化中のレジスタのリタイミング)

グローバル最適化中にレジスタをリタイミングします。

メモ : このオプションは、Spartan®-6、Virtex®-6、Virtex-5、および Virtex-4 デバイスでのみ使用できます。

構文

-retiming on|off

on に設定すると、レジスタがロジックの順方向または逆方向に移動されてタイミングパスの遅延のバランスが取られ、全体的なクロック周波数が向上します。デフォルトでは **off** に設定されています。

この処理によりレジスタの数が変わることがあります。

メモ : [-global_opt \(グローバル最適化\)](#) を設定した場合にのみ使用できます。

-smartguide (SmartGuide)

前回のインプリメンテーション結果 (配置配線済みの NCD ファイル) を、インプリメンテーションを実行する際のガイドとして使用します。SmartGuide™ テクノロジを使用すると、タイミングドリブン パックと配置 (**-timing** オプション) が自動的に設定されます。タイミングドリブン パックと配置は、リソースの使用率が高いデザインでパフォーマンスとタイミングを向上します。

SmartGuide テクノロジをイネーブルにする前に、**map -timing** オプションを使用して配置配線済みの NCD ガイドファイルを作成しておく、結果が向上する場合があります。SmartGuide テクノロジは、コマンドラインまたは Project Navigator の [Design] パネルの [Hierarchy] ペインからイネーブルにできます。

構文

-smartguide *design_name.ncd*

メモ: NGM ファイルがあると、ガイド率が高くなります。NGM ファイルには、MAP プロセスで実行された変換に関する情報が含まれています。MAP で NGM ファイルを検出する方法については、「[MAP のプロセス](#)」を参照してください。

SmartGuide テクノロジでのガイドは、MAP の BEL レベルで行われます。ガイドには、パック、配置、配線が含まれます。デザインのパックおよび配置が最適に変更され、PAR の段階でネットが新たに配線されます。SmartGuide テクノロジでは、デザインの変更されていない部分のインプリメンテーションを保持し、変更された部分でタイミング要件を満たすことが第 1 の目標であり、ランタイムの短縮が第 2 の目標です。変更されていない部分のインプリメンテーションは変更されず、同じタイミング スコアが保持されます。タイミングを満たしておらず、変更されていないパスは、100% ガイドされます。タイミングを満たしておらず、変更されたパスは、再インプリメントされます。

MAP の実行結果は、マップ レポート ファイル (MRP) に保存されます。ガイドされたネット、新しいコンポーネント、ガイドされたコンポーネント、再インプリメントされたコンポーネントの数など、ガイドの統計がリストされます。これは予測される統計であり、最終的な統計は配置配線 レポート (PAR) に記述されます。PAR では、ガイド レポート ファイル (GRF) も生成されます。PAR コマンドラインで **-smartguide** オプションを使用すると、詳細なガイド レポート ファイルが生成され、**-smartguide** を使用しない場合はサマリ ガイド レポート ファイルが生成されます。ガイド レポート ファイルには、再インプリメントされたコンポーネントおよびネット、新規のコンポーネントおよびネットがリストされます。

-timing オプションを使用すると、タイミングドリブンのパックと配置に関連するすべてのオプションがイネーブルになります。これには、デザインのパックおよび配置に使用されるエフォートレベルを設定する **-ol** オプションも含まれます。詳細は、「[-ol \(総体的なエフォートレベル\)](#)」を参照してください。**-timing** オプションを使用すると、**-logic_opt**、**-ntd**、**-ol**、**-register_duplication**、**-x**、および **-xe** がイネーブルになります。各オプションの詳細は、該当するオプションのセクションを参照してください。「[-timing \(タイミングドリブンのパックと配置\)](#)」も参照してください。

-t (配置コスト テーブル)

配置で使用するコスト テーブルを指定します。

構文

-t [*placer_cost_table*]

placer_cost_table 配置で使用するコストテーブルを指定します。配置コストテーブルについては、「[PAR](#)」の章を参照してください。有効な値は 1 ～ 100 で、デフォルト値は 1 です。

複数のコストテーブルを使用してインプリメンテーションを自動的に複数回実行する方法については、「[SmartXplorer](#)」の章を参照してください。

メモ：-t オプションは、-timing オプションを使用してタイミングドリブンのパックと配置を実行する場合にのみ使用できます。

-timing (タイミングドリブンのパックと配置)

デザイン パフォーマンスを向上します。このオプションを使用すると、MAP でデザインのパックおよび配置が実行されます。パックと配置は、ユーザーが UCF/NCF ファイルで設定したタイミング制約を基にして実行されます。

メモ：-timing は、すべての Spartan®-3 ファミリーおよび Virtex®-4 デバイスではオプションですが (デフォルトはオフ)、Spartan-6、Virtex-6、および Virtex-5 デバイスでは常にオンです。

構文

-timing

-timing を使用すると、配置は PAR ではなく MAP で実行されるので、PAR のランタイムは短縮されますが、MAP のランタイムが長くなることがあります。

デバイスリソースの使用率が高いデザインのパフォーマンス、タイミング、パックを向上するには、タイミングドリブンのパックと配置をお勧めします。パックされた関連しないロジックの数 (MAP レポートファイルの「Design Summary」セクションに表示) が 0 ではない場合、-timing オプションを使用してデバイスにより多くのロジックを簡単にパックできます。デザインにローカル クロックが含まれている場合にも、タイミングドリブンのパックと配置をお勧めします。ユーザー タイミング制約を設定せずにタイミングドリブンのパックと配置を選択すると、内部クロックに対してタイミング制約が自動的に生成され、ダイナミックに調整されます。このように MAP および PAR を実行することを「パフォーマンス評価モード」と言います。詳細は、「[-x \(パフォーマンス評価モード\)](#)」を参照してください。このモードを使用すると、デザイン内のすべてのクロックのパフォーマンスが 1 つのパスで評価されます。このモードにより達成されるパフォーマンスは、必ずしも各クロックの最適パフォーマンスとはなりませんが、クロック全体でバランスの取れたパフォーマンスになります。

-timing オプションを使用すると、タイミングドリブンのパックと配置に関連するすべてのオプションがイネーブルになります。これには、デザインのパックおよび配置に使用されるエフォートレベルを設定する -ol オプションも含まれます。詳細は、「[-ol \(総体的なエフォートレベル\)](#)」を参照してください。-timing オプションを使用すると、-logic_opt、-ntd、-ol、-register_duplication、-x、および -xe がイネーブルになります。各オプションの詳細は、該当するオプションのセクションを参照してください。この章の「[再合成および物理合成最適化](#)」も参照してください。

-u (未使用ロジックを保持)

未使用のコンポーネントおよびネットを削除しないよう指定します。

構文

-u

デフォルト (**-u** オプションなし) では、マップの前に未使用のコンポーネントおよびネットがデザインから削除されます。未使用のロジックとは、駆動されていないロジック、ほかのロジックを駆動しないロジック、または「サイクル」として動作し、デバイスの出力に影響しないロジックのことです。**-u** オプションを使用すると、未使用の信号に S (NOCLIP) プロパティが設定され、デザインでロジックが削除されないようになります。未使用のコンポーネントは、未使用の信号があり NOCLIP プロパティが設定されない場合、削除される可能性があります。

-w (既存ファイルの上書き)

デザイン ファイル (NCD) も含め、既存のファイルを上書きします。

構文

-w

-x (パフォーマンス評価モード)

ユーザー制約ファイルで設定されているタイミング制約の代わりに、ツールで生成されたタイミング制約を使用して MAP と PAR を実行し、デザインの各クロックのパフォーマンスを評価します。

構文

-x

この機能を「パフォーマンス評価モード」と呼びます。このモードは、**-x** オプションを使用するか、デザインにタイミング制約が設定されていない場合に実行されます。ツールにより生成されたタイミング制約は、各クロックに個別に設定され、PAR の実行中に調整されます。MAP のエフォート レベルにより、ランタイムを最短にすることを目標とするか (std)、パフォーマンスを最高にすることを目標とするか (high) を制御します。

メモ: **-x** オプションを使用すると、UCF/NCF ファイルでユーザーが指定したタイミング制約は無視されますが、LOC や AREA_GROUP などの物理制約は使用されます。

メモ: **-x** オプションと **-ntd** オプションを同時に使用することはできません。ユーザー タイミング制約を使用しない場合、どちらか 1 つの自動タイミング モードしか設定できません。

-xe (追加エフォート レベル)

追加エフォート レベルを指定します。このオプションは、**-timing** オプションを使用してタイミングドリブンのパックと配置を実行しする場合にのみ使用できます。

構文

-xe effort_level

effort_level には、**n** (標準) または **c** (継続) を設定できます。**c** に設定すると、それ以上ほとんど改善しないという時点まで、パックが向上するよう実行を継続します。

map -ol high -xe n design.ncd output.ncd design.pcf

-xt (追加配置コスト テーブル)

リソースの使用率が高いデザインに適したコスト テーブルを指定します。これらのコスト テーブルは、通常のコスト テーブル (**-t** オプション) と共に使用できます。

このオプションは、Spartan®-6 および Virtex®-6 デバイスでのみ使用できます。

構文

-xt *cost_table*

cost_table : デザインをさらに最適化するアルゴリズムを選択します。0 ~ 5 の整数を指定できます。デフォルトは 0 です。

再合成および物理合成最適化

MAP には、標準のインプリメンテーションからさらにタイミングを向上させるアドバンス最適化を実行するオプションがあります。これらのアドバンス最適化は、配置前後のデザインを変換します。

最適化は、ザイリンクス デザイン フローの 2 つの段階で適用できます。1 つ目の段階は、ロジックがアーキテクチャ スライスにマップされた後です。MAP の **-global_opt** オプションを使用すると、配置の前にマップ済みのデザインに対してグローバル最適化ルーチンが実行されます。詳細は、「**-global_opt (グローバル最適化)**」および「**-retiming (グローバル最適化中のレジスタのリタイミング)**」を参照してください。

2 つ目の段階は配置後で、タイミングを満たしていないパスが評価され、再合成されます。MAP は初期ネットリストを読み込み、配置して、デザインのタイミングを解析します。タイミングが満たされていない場合は、物理合成最適化を実行し、タイミングが満たされるようにネットリストを変更します。物理合成最適化をイネーブルにするには、タイミングドリブン パックと配置 (**-timing**) を設定する必要があります。

物理合成最適化は、**-logic_opt (ロジックの最適化)** および **-register_duplication (レジスタの複製)** オプションを使用するとイネーブルになります。各オプションの詳細および使用法は、「**MAP のオプション**」を参照してください。

ガイド マップ

ガイド マップでは、既存の NCD をガイドとして現在の MAP 実行に使用します。ガイド ファイルは、インプリメンテーションのどの段階 (未配置または配置済み、未配線または配線済み) のものでもかまいません。ザイリンクスでは、最新バージョンのソフトウェアを使用して新規 NCD ファイルを生成し直すことをお勧めします。以前のリリースで生成された NCD ファイルを使用してもガイド マップは実行できますが、サポートされているとは限りません。

メモ : **-timing** オプションを使用してガイド マップを実行する場合、配置済みの NCD ファイルをガイド ファイルとして使用することをお勧めします。配置済み NCD は、MAP を **-timing** オプションを使用して実行するか、または PAR を実行すると生成されます。

SmartGuide™ テクノロジは、前回のインプリメンテーション結果を次回インプリメンテーションを実行する際のガイドとして使用できるようにする機能です。SmartGuide テクノロジを使用すると、MAP および PAR で **-smartguide** オプションで指定した NCD ファイルが、コンポーネントおよびネットのインプリメンテーションのガイドとして使用されます。タイミングを満たすため、ガイドされたコンポーネントおよびネットが移動される場合があります。SmartGuide テクノロジでは、タイミング要件を満たすことによりデザイン パフォーマンスを保持することが第 1 の目標であり、ランタイムの短縮が第 2 の目標です。

SmartGuide テクノロジは、デザイン サイクルの最終段階で、タイミングが満たされており、デザインに小さな変更を加える場合に最適です。タイミングを満たすことが困難なパスに変更を加える場合は、SmartGuide テクノロジを使用しない方が最適なパフォーマンスが得られます。SmartGuide テクノロジは、次のようなデザインの変更でも有益です。

- ・ ピン ロケーションの変更
- ・ インスタンス化されたコンポーネントの属性の変更
- ・ タイミング制約の緩和
- ・ ChipScope™ コアの追加

このリリースでは、**-gm** および **-gf** オプションに置き換わる方法として SmartGuide が使用されます。

メモ： 詳細は、「[-smartguide \(SmartGuide\)](#)」を参照してください。

NCD ファイルと NGM ファイルがガイドとして使用されます。NGM ファイルには、MAP プロセスで実行された変換に関する情報が含まれています。NGM ファイルをコマンドラインで指定する必要はありません。指定されたガイド NCD ファイルから、対応する NGM ファイルが自動的に判断されます。対応する NGM ファイルが見つからない場合は、埋め込まれた名前 (完全パスおよびファイル名) に基づいて適切な NGM ファイルが検索されます。NGM ファイルが NCD ファイルと同じディレクトリ、作業中のディレクトリ、または埋め込まれた名前に基づく場所にはない場合は、警告メッセージが表示されます。この場合、NCD ファイルのみがガイドファイルとして使用されますが、ガイドの効率が低下する可能性があります。

メモ： NGM ファイルがあると、ガイド率が高くなります。

MAP の実行結果は、マップ レポート ファイル (MRP) に保存されます。ガイドされたネット、新しいコンポーネント、ガイドされたコンポーネント、再インプリメントされたコンポーネントの数など、ガイドの統計がリストされます。これは予測される統計であり、最終的な統計は配置配線レポートに記述されます。PAR を **-smartguide** オプションを使用して実行すると、ガイド レポート ファイル (GRF) も生成されます。GRF ファイルには、再インプリメントされたコンポーネントおよびネット、新規のコンポーネントおよびネットがリストされます。

メモ： 詳細は、「[-smartguide \(SmartGuide\)](#)」を参照してください。

マップ結果のシミュレーション

NGC ファイルを使用してシミュレーションを実行した場合、マップした結果がシミュレーションされるのではなく、論理的な回路記述がシミュレーションされます。NCD ファイルをシミュレーションすると、物理的な回路記述がシミュレーションされます。

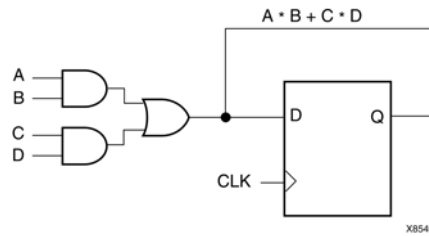
MAP により、バックアノテートしたシミュレーション ネットリストでは検出されないエラーが発生する場合があります。たとえば、MAP を実行した後で次のコマンドを実行し、バックアノテートしたシミュレーション ネットリストを生成するとします。

```
netgen mapped.ncd mapped.ngm -o mapped.nga
```

このコマンドを実行すると、mapped.ngm という名前の論理と物理の相互参照ファイルを使用して、バックアノテートしたシミュレーション ネットリストが作成されます。この相互参照ファイルには、論理的なデザイン ネットリストに関する情報が含まれ、バックアノテートしたシミュレーション ネットリスト (mapped.nga) は、実際には論理的なデザインをバックアノテートしたものです。この場合、アクティブ High のファンクションにアクティブ Low のファンクションをインプリメントするなど、MAP で物理的なエラーが発生しても、mapped.nga ファイルでは検出されないため、このエラーはシミュレーション ネットリストには表示されません。

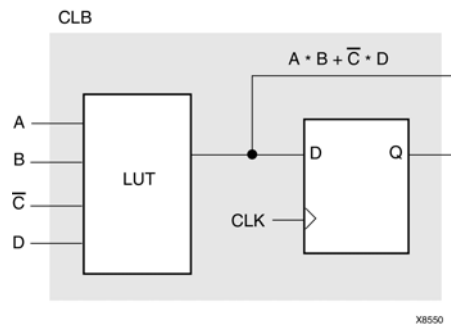
たとえば、デザイン ファイルから NGDBuild で次のような論理回路を生成したとします。

論理回路表現



組み合わせロジックからのブール出力に注目してください。上記の回路で MAP を実行して、次のような結果が得られたとします。

CLB コンフィギュレーション



MAP により、アクティブ High (C) ではなくアクティブ Low (C) が生成されています。このため、組み合わせロジックのブール出力が正しくなくなります。遅延は物理デザインではなく正しい論理的デザインにバックアノテートされるので、mapped.ngm ファイルを使用して NetGen を実行しても、この論理的なエラーは検出されません。

エラーを検出するには、mapped.ngm 相互参照ファイルを使用せずに NetGen コマンドを実行するのが 1 つの方法です。

netgen mapped.ncd -o mapped.nga

上記のコマンドを実行すると、mapped.nga ファイルを使用した物理的なシミュレーションにより、物理的なエラーが検出されます。ただし、エラーのタイプは簡単には特定できません。エラーを特定するには、FPGA Editor を使用するか、ザイリンクスのカスタマ サポートにご連絡ください。エラーがレポートされた場合でも、実際にはエラーはなく、CLB コンフィギュレーションが正しい場合もあります。この場合は、FPGA Editor を使用すると、CLB が正しくモデリングされているかを調べることができます。

論理的なシミュレーションでも物理的なシミュレーションでも発生するエラーが検出されない場合は、シミュレーションでさらにテスト ベクタを使用する必要があります。

MAP レポート (MRP) ファイル

MAP レポート (MRP) ファイルは、MAP 実行に関する情報を含む ASCII 形式のテキストファイルです。レポートに含まれる情報は、指定したデバイスと **-detail** オプションを使用したかどうかによって異なります。詳細は、「[-detail \(詳細な MAP レポートの生成\)](#)」を参照してください。

ここで示す例は、短縮されたファイルであり、通常の MRP レポートファイルは、これよりも長くなります。ファイルは多数のセクションに分かれており、セクションに関する情報がない場合でも表示されます。MRP ファイルに含まれるセクションは、次のとおりです。

- ・ **Design Information (デザイン情報)** : MAP コマンドライン、デザインがマップされたデバイス、マップを実行した日時などを示します。
- ・ **Design Summary (デザイン サマリ)** : エラーと警告の数、マップされたデザインにより使用されるターゲット デバイスのリソース数など、マップ実行の概要を示します。
- ・ **Table of Contents (目次)** : MAP レポートの残りのセクションをリストします。
- ・ **Errors (エラー)** : 次のエラーを示します。
 - MAP の開始時に実行された論理的 DRC テストに関するエラー。このエラーは、マップするデバイスには依存しません。
 - パッドがロジックに接続されていない、デザインに双方向パッドが配置されているがパッドを通過する信号が 1 方向のみなど、MAP により発見されたエラー。このエラーは、マップするデバイスによって異なる場合があります。
 - マップ済みのデザインに対して実行された物理的 DRC に関するエラー
- ・ **Warning (警告)** : 次の警告を示します。
 - MAP の開始時に実行された論理的 DRC テストに関する警告。この警告は、マップするデバイスには依存しません。
 - MAP により発見された警告。この警告は、マップするデバイスによって異なる場合があります。
 - マップ済みのデザインに対して実行された物理的 DRC に関する警告
- ・ **Informational (情報用)** : 特に対処する必要はないが、重要な事項を示します。このメッセージには、フローの後の方で問題が発生した場合に有益な情報が含まれます。
- ・ **Removed Logic Summary (削除されたロジックのサマリ)** : デザインから削除されたブロックおよび信号の数を示します。次のようなロジックについてレポートします。
- ・ **Removed Logic (削除されたロジック)** : デザインをマップする際に入力 NGD ファイルから削除されたすべてのロジック (デザイン コンポーネントとネット) の詳細を示します。通常、ロジックは次の理由から削除されます。
 - デザインでライブラリ マクロのロジックの一部のみを使用している。
 - 完成していないデザインをマップした。
 - MAP でデザイン ロジックが最適化された。
 - 回路図入力時に未使用ロジックが誤って作成された。

このセクションには、結合されたネットについても示されます。ネットは、2 つのネットを分離するコンポーネントが削除されると結合されます。

また、信号またはシンボルが削除された結果、さらに別の信号やシンボルが削除される場合、それを示す行はインデントされます。このインデントは、関連したロジックのチェーンが削除される場合は繰り返されます。ロジック チェーンが削除された原因を見つけるには、目的のロジックの上にあるインデントされていない一番上の行を探します。
- ・ **IOB Properties (IOB プロパティ)** : ユーザーが制約を指定した IOB とその IOB に指定されているプロパティをリストします。
- ・ **RPMs (相対配置マクロ)** : デザインに使用される各 RPM と RPM のインプリメンテーションに使用されるデバイスのコンポーネント数を示します。

- ・ **SmartGuide Report (SmartGuide レポート)** : SmartGuide™ テクノロジを使用してマップを実行した場合、ガイドされたコンポーネントおよびネットの予測される割合をリストします。正確な結果は配置配線レポートに記述されます。詳細は、「PAR」の章の「ReportGen」を参照してください。
- ・ **Area Group Summary & Partition Summary (エリア グループ サマリとパーティション サマリ)** : 各エリア グループまたはパーティションの結果の概要を示します。エリア グループは、異なる物理的エリアにパックされた論理ブロックのグループを指定するために使用されます。デザインにエリア グループまたはパーティションがない場合は、そのことが示されます。
- ・ **Timing Report (タイミング レポート)** : **-timing** オプションにより生成されるこのセクションは、MAP 実行中に考慮されたタイミング制約に関する情報を示します。このレポートはデフォルトでは生成されず、**-detail** オプションを指定した場合にのみ生成されます。
- ・ **Configuration String Information (コンフィギュレーション文字列)** : **-detail** オプションにより生成されるこのセクションは、DCM、BRAM、GT などの特殊コンポーネントのコンフィギュレーション文字列およびプログラム プロパティを示します。DCM および PLL のレポートが含まれます。スライスと IOB のコンフィギュレーション文字列には、「SECURE」と表示されます。このレポートはデフォルトでは生成されず、**-detail** オプションを指定した場合にのみ生成されます。
- ・ **Control Set Information (コントロール セット情報)** : コントロール セットの情報を示します (Virtex®-5 デバイスのみ)。このレポートはデフォルトでは生成されず、**-detail** オプションを指定した場合にのみ生成されます。
- ・ **Utilization by Hierarchy (階層ごとの使用率)** : 階層ごとのリソースの使用率を示します (Virtex-4、Virtex-5、Spartan®-3 アーキテクチャのみ)。このレポートはデフォルトでは生成されず、**-detail** オプションを指定した場合にのみ生成されます。

メモ : MRP ファイルは、等幅フォントでの表示用にフォーマットされます。レポートの表示に使用するテキスト エディタで等幅ではないフォントを使用した場合、レポートの列は正しく並びません。

メモ : MAP レポートには、値が DIFFSI、DIFFMI、および _NDT のピンを含むピン配置表が生成されます。

MAP レポートの例 1

```
Xilinx Mapping Report File for Design 'wave_gen'

Design Information
-----
Command Line   : map -intstyle ise -p xc6vlx75t-ff484-1 -w -ol high -t 1 -xt 0 -register_duplication off -global_opt off
-mt off -ir off -pr o -lc off -power off -o wave_gen_map.ncd wave_gen.ngd wave_gen.pcf
Target Device  : xc6vlx75t
Target Package : ff484
Target Speed   : -1
Mapper Version : virtex6 -- $Revision: 1.52 $
Mapped Date    : Thu Feb 25 16:16:02 2010

Design Summary
-----
Number of errors:      0
Number of warnings:    0
Slice Logic Utilization:
  Number of Slice Registers:      569 out of 93,120    1%
    Number used as Flip Flops:    568
    Number used as Latches:       1
    Number used as Latch-thrus:   0
    Number used as AND/OR logics: 0
  Number of Slice LUTs:          958 out of 46,560    2%
    Number used as logic:         854 out of 46,560    1%
    Number using O6 output only: 658
```

Number using O5 output only:	15		
Number using O5 and O6:	181		
Number used as ROM:	0		
Number used as Memory:	0 out of	16,720	0%
Number used exclusively as route-thrus:	104		
Number with same-slice register load:	102		
Number with same-slice carry load:	2		
Number with other load:	0		

Slice Logic Distribution:

Number of occupied Slices:	328 out of	11,640	2%
Number of LUT Flip Flop pairs used:	988		
Number with an unused Flip Flop:	569 out of	988	57%
Number with an unused LUT:	30 out of	988	3%
Number of fully used LUT-FF pairs:	389 out of	988	39%
Number of unique control sets:	52		
Number of slice register sites lost to control set restrictions:	223 out of	93,120	1%

A LUT Flip Flop pair for this architecture represents one LUT paired with one Flip Flop within a slice. A control set is a unique combination of clock, reset, set, and enable signals for a registered element. The Slice Logic Distribution report is not meaningful if the design is over-mapped for a non-slice resource or if Placement fails. OVERMAPPING of BRAM resources should be ignored if the design is over-mapped for a non-BRAM resource or if placement fails.

IO Utilization:

Number of bonded IOBs:	17 out of	240	7%
IOB Flip Flops:	11		

Specific Feature Utilization:

Number of RAMB36E1/FIFO36E1s:	0 out of	156	0%
Number of RAMB18E1/FIFO18E1s:	2 out of	312	1%
Number using RAMB18E1 only:	2		
Number using FIFO18E1 only:	0		
Number of BUFG/BUFGCTRLs:	3 out of	32	9%
Number used as BUFGs:	3		
Number used as BUFGCTRLs:	0		
Number of ILOGICE1/ISERDESE1s:	0 out of	360	0%
Number of OLOGICE1/OSERDESE1s:	11 out of	360	3%
Number used as OLOGICE1s:	11		
Number used as OSERDESE1s:	0		
Number of BSCANS:	0 out of	4	0%
Number of BUFHCEs:	1 out of	72	1%
Number of BUFOS:	0 out of	18	0%
Number of BUFIODQSS:	0 out of	36	0%
Number of BUFRRs:	0 out of	18	0%
Number of CAPTUREs:	0 out of	1	0%
Number of DSP48E1s:	0 out of	288	0%
Number of EFUSE_USRs:	0 out of	1	0%
Number of GTXE1s:	0 out of	8	0%
Number of IBUFDS_GTXE1s:	0 out of	6	0%
Number of ICAPs:	0 out of	2	0%
Number of IDELAYCTRLs:	0 out of	9	0%
Number of IODELAYE1s:	0 out of	360	0%
Number of MMCM_ADVs:	1 out of	6	16%
Number of PCIE_2_0s:	0 out of	1	0%
Number of STARTUPs:	0 out of	1	0%
Number of SYSMONs:	0 out of	1	0%
Number of TEMAC_SINGLES:	0 out of	4	0%

Average Fanout of Non-Clock Nets: 3.95

Peak Memory Usage: 727 MB
 Total REAL time to MAP completion: 1 mins 10 secs
 Total CPU time to MAP completion: 1 mins 5 secs

Table of Contents

 Section 1 - Errors
 Section 2 - Warnings
 Section 3 - Informational

Section 4 - Removed Logic Summary
 Section 5 - Removed Logic
 Section 6 - IOB Properties
 Section 7 - RPMs
 Section 8 - Guide Report
 Section 9 - Area Group and Partition Summary
 Section 10 - Timing Report
 Section 11 - Configuration String Information
 Section 12 - Control Set Information
 Section 13 - Utilization by Hierarchy

Section 1 - Errors

Section 2 - Warnings

Section 3 - Informational

INFO:LIT:243 - Logical network uart_rx_i0/frm_err has no load.
 INFO:MapLib:564 - The following environment variables are currently set:
 INFO:MapLib:591 - XIL_MAP_NODRC Value: 1
 INFO:LIT:244 - All of the single ended outputs in this design are using slew
 rate limited output drivers. The delay on speed critical single ended outputs
 can be dramatically reduced by designating them as fast outputs.
 INFO:Pack:1716 - Initializing temperature to 85.000 Celsius. (default - Range:
 0.000 to 85.000 Celsius)
 INFO:Pack:1720 - Initializing voltage to 0.950 Volts. (default - Range: 0.950 to
 1.050 Volts)
 INFO:Timing:3386 - Intersecting Constraints found and resolved.
 For more information, see the TSI report. Please consult the Xilinx
 Command Line Tools User Guide for information on generating a TSI report.
 INFO:Map:215 - The Interim Design Summary has been generated in the MAP Report
 (.mrp).
 INFO:Pack:1650 - Map created a placed design.

Section 4 - Removed Logic Summary

4 block(s) removed
 21 block(s) optimized away
 2 signal(s) removed

Section 5 - Removed Logic

The trimmed logic report below shows the logic removed from your design due to
 sourceless or loadless signals, and VCC or ground connections. If the removal
 of a signal or symbol results in the subsequent removal of an additional signal
 or symbol, the message explaining that second removal will be indented. This
 indentation will be repeated as a chain of related logic is removed.

To quickly locate the original cause for the removal of a chain of logic, look
 above the place where that logic is listed in the trimming report, then locate
 the lines that are least indented (begin at the leftmost edge).

The signal "uart_rx_i0/frm_err" is sourceless and has been removed.
 The signal "DAC_SPI_controller_i0/out_ddr_flop_spi_clk_i0/N1" is sourceless and
 has been removed.

Unused block "char_fifo_i0/GND" (ZERO) removed.
 Unused block "char_fifo_i0/VCC" (ONE) removed.
 Unused block "samp_ram_i0/GND" (ZERO) removed.
 Unused block "samp_ram_i0/VCC" (ONE) removed.

Optimized Block(s):

TYPE	BLOCK
GND	XST_GND
GND	char_fifo_i0/BU2/XST_GND
VCC	char_fifo_i0/BU2/XST_VCC
GND	samp_ram_i0/BU2/XST_GND
VCC	samp_ram_i0/BU2/XST_VCC
VCC	DAC_SPI_controller_i0/XST_VCC
GND	DAC_SPI_controller_i0/out_ddr_flop_spi_clk_i0/XST_GND
VCC	DAC_SPI_controller_i0/out_ddr_flop_spi_clk_i0/XST_VCC

```

GND  clk_gen_i0/clk_core_i0/XST_GND
VCC  clk_gen_i0/clk_core_i0/XST_VCC
GND  clk_gen_i0/clk_div_i0/XST_GND
VCC  clk_gen_i0/clk_div_i0/XST_VCC
VCC  cmd_parse_i0/XST_VCC
VCC  resp_gen_i0/XST_VCC
GND  resp_gen_i0/to_bcd_i0/XST_GND
VCC  resp_gen_i0/to_bcd_i0/XST_VCC
GND  rst_gen_i0/reset_bridge_clk_clk_samp_i0/XST_GND
GND  rst_gen_i0/reset_bridge_clk_rx_i0/XST_GND
GND  rst_gen_i0/reset_bridge_clk_tx_i0/XST_GND
GND  samp_gen_i0/XST_GND
VCC  samp_gen_i0/XST_VCC

```

To enable printing of redundant blocks removed and signals merged, set the detailed map report option and rerun map.

Section 6 - IOB Properties

IOB Name	Type	Direction	IO Standard	Diff Term	Drive Strength	Slew Rate	Reg (s)	Resistor	IOB Delay
DAC_clr_n_pin	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
DAC_cs_n_pin	IOB	OUTPUT	LVC MOS25		12	SLOW			
SPI_MOSI_pin	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
clk_pin	IOB	INPUT	LVC MOS25						
lb_sel_pin	IOB	INPUT	LVC MOS25						
led_pins<0>	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
led_pins<1>	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
led_pins<2>	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
led_pins<3>	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
led_pins<4>	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
led_pins<5>	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
led_pins<6>	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
led_pins<7>	IOB	OUTPUT	LVC MOS25		12	SLOW	OFF		
rst_pin	IOB	INPUT	LVC MOS25						
rx_d_pin	IOB	INPUT	LVC MOS25						
spi_clk_pin	IOB	OUTPUT	LVC MOS25		12	SLOW	ODDR		
tx_d_pin	IOB	OUTPUT	LVC MOS25		12	SLOW			

Section 7 - RPMs

Section 8 - Guide Report

Guide not run on this design.

Section 9 - Area Group and Partition Summary

Partition Implementation Status

No Partitions were found in this design.

Area Group Information

No area groups were found in this design.

Section 10 - Timing Report

A logic-level (pre-route) timing report can be generated by using Xilinx static timing analysis tools, Timing Analyzer (GUI) or TRCE (command line), with the mapped NCD and PCF files. Please note that this timing report will be generated using estimated delay information. For accurate numbers, please generate a

timing report with the post Place and Route NCD file.

For more information about the Timing Analyzer, consult the Xilinx Timing Analyzer Reference Manual; for more information about TRCE, consult the Xilinx Command Line Tools User Guide "TRACE" chapter.

Section 11 - Configuration String Details

Use the "-detail" map option to print out Configuration Strings

Section 12 - Control Set Information

Use the "-detail" map option to print out Control Set Information.

Section 13 - Utilization by Hierarchy

Use the "-detail" map option to print out the Utilization by Hierarchy section.

物理合成レポート (PSR) ファイル

物理合成 レポート (PSR) ファイルは、次の MAP オプションに関する情報を含む ASCII 形式のテキスト ファイルです。

- ・ **-glob_opt** (グローバル最適化)
- ・ **-retiming** (グローバル最適化中のレジスタのリタイミング)
- ・ **-equivalent_register_removal** (重複したレジスタの削除)
- ・ **-logic_opt** (ロジックの最適化)
- ・ **-register_duplication** (レジスタの複製)
- ・ **-power high** または **-power xe** (クロック ゲート アルゴリズムを使用した消費電力の最適化)

レポートの最初の部分は、-power 以外のオプションに関する情報を含み、次の 3 つのセクションで構成されます。

- ・ **Physical Synthesis Options Summary (物理合成オプションのサマリ)**: インプリメンテーションで使用された物理合成オプションとターゲット デバイスを示します。
- ・ **Optimizations Statistics (最適化の統計)**: 物理合成最適化によって追加または削除されたレジスタおよび SRL の数を示します。
- ・ **Optimization Details (最適化の詳細)**: 新規または修正されたインスタンス、そのインスタンスに影響した最適化、その最適化の目標を示します。

次に、インスタンスに影響する可能性のある最適化および各最適化の目標を示します。

最適化	目標	最適化を実行させたオプション
SRL の推論	エリア	<code>-global_opt</code>
同期最適化	パフォーマンス	<code>-global_opt speed</code>
最大ファンアウトの削減	ファンアウトの最適化	<code>-register_duplication +</code> MAX_FANOUT 制約
複製	ファンアウトの最適化	<code>-register_duplication +</code> MAX_FANOUT 制約
重複したレジスタの削除	簡略化	<code>-equivalent_register_removal</code>
逆方向のリタイミング	パフォーマンス	<code>-retiming</code>
順方向のリタイミング	パフォーマンス	<code>-retiming</code>
ロジックの自動削除	簡略化	<code>-global_opt</code>
SmartOpt 自動削除	エリア	<code>-global_opt area</code>

`-power high` または `-power xe` オプションを使用すると、レポートに次の 3 つのセクションが追加されます。

- ・ **Power Opt Slice clock gating summary (消費電力最適化スライス クロック ゲート サマリー)** : 使用されたオプションを示します。
- ・ **Optimization Statistics (最適化の統計)** : ゲートされたスライス レジスタと処理されたクロック イネーブル ネットを示します。
- ・ **Optimization Details (最適化の詳細)** : 変更されたコンポーネント インスタンスの名前、タイプ、そのクロック イネーブル ネットの名前、および最適化の目標 (power) を示します。

PSR レポートの例

TABLE OF CONTENTS

- 1) Physical Synthesis Options Summary
- 2) Optimizations statistics and details

```

=====
*           Physical Synthesis Options Summary           *
=====

---- Options

Global Optimization                : ON

    Retiming                      : OFF
    Equivalent Register Removal    : ON

Timing-Driven Packing and Placement : ON

    Logic Optimization            : ON
    Register Duplication          : ON
---- Target Parameters

Target Device                      : 4vsx25ff668-12

=====
*           Optimizations                               *
=====

---- Statistics

    Number of registers added by Replication           | 1
    Number of registers removed by Equivalence Removal | 1
    Overall change in number of registers              | 0

---- Details

New or modified components | Optimization | Objective
-----|-----|-----
sd_data_t_24_1             | Replication  | Fanout Optimization
Removed components         | Optimization |
-----|-----|-----
data_addr_n_reg_1         | Equivalence Removal

```

MAP の停止

MAP を停止するには、Ctrl + C キー (ワークステーション) または Ctrl + Break キー (PC) を押します。ワークステーションの場合、Ctrl + C キーを押すときに、MAP を実行しているウィンドウがアクティブになっていることを確認してください。実行中の処理が停止されます。MAP レポート ファイルや PCF ファイルなど、MAP が停止されても残るファイルはありますが、これらは処理が完了していない状態を表しているため、破棄してもかまいません。

物理的デザイン ルール チェック

この章では、物理的デザイン ルール チェック (DRC) プログラムについて説明します。次のセクションから構成されています。

- ・ [DRC の概要](#)
- ・ [DRC の構文](#)
- ・ [DRC のオプション](#)
- ・ [DRC によるチェック](#)
- ・ [DRC のエラーと警告](#)

DRC の概要

物理的デザイン ルール チェック (DRC) は、デザインの物理的なエラーと一部の論理的なエラーを検出するための一連のテストです。物理的 DRC は、次のように実行されます。

- ・ デザインをマップした後、MAP により自動的に実行されます。
- ・ デザインを配線する際に、PAR によりネットに対して自動的に実行されます。
- ・ デバイスをプログラムするための BIT ファイルを作成する BitGen で自動的に実行されます。
- ・ 物理的 DRC は、FPGA Editor から実行できます。また、FPGA Editor でロジック セルを変更したり、ネットを手動で配線すると自動的に実行されます。FPGA Editor での DRC の使用法については、FPGA Editor のオンライン ヘルプを参照してください。
- ・ Linux または DOS コマンド ラインから実行できます。

デバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6

DRC の入力ファイル

DRC には NCD ファイルを入力します。NCD ファイルには、マップ済みまたは配置配線済みデザインが物理的に記述されています。

DRC の出力ファイル

DRC から出力されるファイルは、TDR ファイルです。このファイルは、ASCII 形式の DRC レポートです。ファイルの内容は、DRC コマンドで指定するオプションによって異なります。

DRC の構文

DRC を実行するには、次のコマンドを使用します。

```
drc [options] file_name.ncd
```

- ・ *options*: 「DRC のオプション」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。
- ・ *file_name*: DRC を実行する NCD ファイルを指定します。

DRC のオプション

このセクションでは、DRC のコマンドライン オプションについて説明します。

- ・ **-e** (エラー レポート)
- ・ **-o** (出力ファイル)
- ・ **-s** (サマリ レポート)
- ・ **-v** (詳細レポート)
- ・ **-z** (不完全プログラムのレポート)

-e (エラー レポート)

エラーに関する詳細のみを含むレポートを作成します。警告に関する詳細は、表示されません。

構文

```
-e
```

-o (出力ファイル)

レポートファイルを、デフォルトの *file_name.tdr* ではなく、*outfile_name.tdr* という名前で出力します。

構文

```
-o outfile_name.tdr
```

-s (サマリ レポート)

サマリレポートのみを作成します。レポートには、検出されたエラーと警告の数が表示されますが、その詳細は表示されません。

構文

```
-s
```

-v (詳細レポート)

すべての警告およびエラーをレポートします。DRC のデフォルト オプションです。

構文

-v

-z (不完全プログラムのレポート)

デザインが完全にプログラムされなかった場合にエラーとしてレポートします。一部の DRC 違反は、DRC が BitGen コマンドの一部として実行された場合はエラーと見なされますが、それ以外では警告と見なされます。通常、これらの DRC 違反は、デザインが完全にプログラムされていないことを示します。ロジック セルが部分的にしかプログラムされていない場合や、信号にドライバがない場合などがその例です。違反がある場合、デバイスをプログラムしようとするエラーが発生するため、BitGen でデバイスをプログラムする BIT ファイルが作成される際に、エラーとしてレポートされます。-z オプションを使用せずにコマンドラインから DRC を実行すると、これらの違反は警告としてレポートされますが、-z オプションを使用すると、エラーとしてレポートされます。

構文

-z

DRC によるチェック

物理的 DRC は、次のチェックを実行します。

- ・ ネット チェック
配線済みまたは未配線の信号を調べ、ピン数、トライステート バッファの不一致、フロートしているセグメント、アンテナ、部分的な配線などの問題点をすべてレポートします。
- ・ ブロック チェック
配置済みまたは未配置のコンポーネントを調べ、ロジック、物理的なピン接続、プログラムに関する問題点をすべてレポートします。
- ・ チップ チェック
デバイスの 1 辺に対する配置規則など、信号およびコンポーネントをチップ レベルでチェックします。
- ・ すべてをチェック
ネット チェック、ブロック チェック、チップ チェックをすべて実行します。

コマンドラインから DRC を実行すると、ネット チェック、ブロック チェック、チップ チェックがすべて自動的に実行されます。

FPGA Editor を使用すると、選択したオブジェクトまたはデザイン内のすべての信号に対して ネット チェックを実行できます。さらに、選択したコンポーネント、またはデザインのすべてのコンポーネントに対してブロック チェックを実行できます。デザイン内のすべてのコンポーネントに対してブロック チェックを実行する場合、各コンポーネントのチェックに加えて、ロングラインを共有するトライステート バッファやコンフィギュレーションされたオシレータ回路などデザイン全体に対するテストも実行されます。FPGA Editor では、ネット チェックとブロック チェックを個別に、または同時に実行できます。

DRC のエラーと警告

DRC のエラー メッセージは、ネットにドライバがない場合や、ロジック ブロックが正しくプログラムされていない場合など、配線またはコンポーネント ロジックが正しく動作していない状態を示します。DRC の警告メッセージは、ネットが完全に配線されていない場合や、ロジック ブロックが信号を処理するようにプログラムされているのに対応するロジック ブロック ピンに信号がない場合など、配線またはロジックが不完全な状態を示します。

アプリケーションや信号接続によって、同じメッセージが警告として表示されたりエラーとして表示されたりすることがあります。たとえば、ネット チェックで、デコーダに接続されている信号にプルアップ抵抗が使用されていない場合にはエラー メッセージが表示されますが、トライステート バッファに接続されている信号にプルアップ抵抗が使用されていない場合は警告メッセージが表示されます。

ドライバのない信号や部分的にプログラムされたロジック セルなど、プログラムが不完全な場合、DRC が BitGen コマンドの一部として実行されるとエラー メッセージが表示されますが、それ以外のプログラムで実行されるときには、警告メッセージが表示されます。`-z` オプションを使用して DRC コマンドを実行すると、不完全なプログラムは警告ではなくエラーとしてレポートされます。`-z` オプションの詳細は、「[-z \(不完全プログラムのレポート\)](#)」を参照してください。

PAR (Place and Route)

この章には、次のセクションが含まれています。

- ・ [PAR の概要](#)
- ・ [PAR のプロセス](#)
- ・ [PAR の構文](#)
- ・ [PAR のオプション](#)
- ・ [PAR のレポート](#)
- ・ [ReportGen](#)
- ・ [PAR の停止](#)

PAR の概要

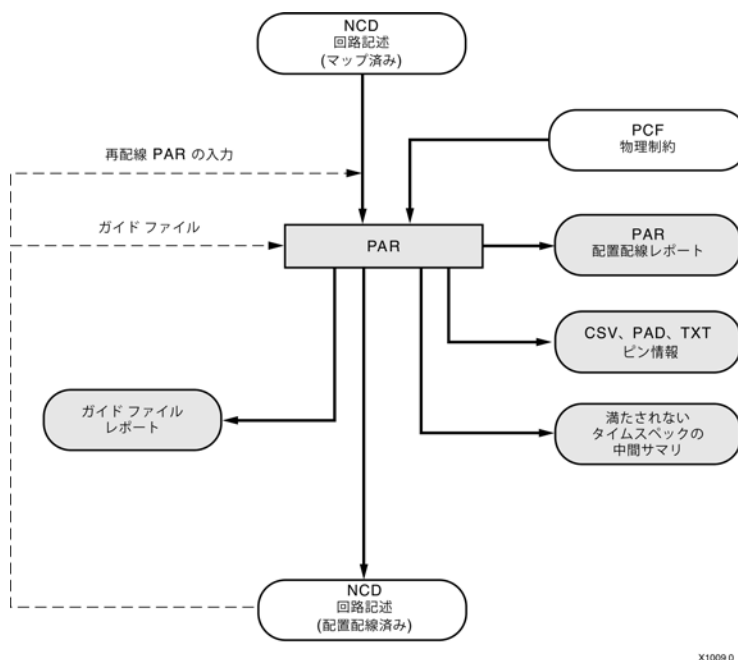
MAP を使用して NCD (Native Circuit Description) ファイルを生成した後は、PAR を使用して配置配線を実行します。PAR は、NCD を入力ファイルとして読み込み、デザインを配置配線して、NCD ファイルを出力します。出力された NCD は、ビットストリーム ジェネレータ (BitGen) で使用します。詳細は、[「BitGen」の章](#)を参照してください。

PAR により生成された NCD は、デザインに多少の変更を加えた後、MAP および PAR で SmartGuide™ を実行する際のガイド ファイルとしても使用できます。詳細は、[「-smartguide \(SmartGuide\)」](#)を参照してください。PAR は、次の基準に基づいてデザインを配置配線します。

- ・ **タイミングドリブン方式**：ザイリンクスのタイミング解析ソフトウェアを使用し、タイミング制約に基づいてデザインを配置配線する方法です。
- ・ **非タイミングドリブン (コスト基準) 方式**：制約、接続の長さ、使用可能な配線リソースなどの関連する要素に重みを付けた値を割り当てるコスト テーブルを使用して、配置配線する方法です。この方式は、タイミング制約を設定していない場合に使用します。

次に、PAR のデザイン フローを示します。ここでは、NCD ファイルを 1 つ生成しています。

PAR フロー



X1009.0

PAR のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6

PAR の入力ファイル

PAR の入力ファイルは、次のとおりです。

- ・ **NCD ファイル**：マップ済みのデザイン ファイルです。
- ・ **PCF ファイル**：UCF または NCF で指定したタイミングおよび物理的な配置に関する制約や、その他の属性を記述した ASCII 形式のファイル。PAR では、『[制約ガイド](#)』にリストされているすべてのタイミング制約がサポートされています。
- ・ **ガイド NCD ファイル**：デザインを配置配線する際にガイドとして使用される、配置配線済みの NCD ファイル (オプション)。

PAR の出力ファイル

PAR の出力ファイルは、次のとおりです。

- ・ NCD ファイル：配置配線済みのデザイン ファイル。配置配線の完成度が異なる場合があります。
- ・ PAR ファイル：配置配線のすべての実行に関するサマリ情報を含む PAR レポート
- ・ PAD ファイル：I/O ピン割り当て情報を記述した、解析可能なデータベース形式のファイル
- ・ CSV ファイル：I/O ピン割り当て情報を記述した、表計算プログラムでサポートされるファイル
- ・ TXT ファイル：I/O ピン割り当て情報を記述した ASCII 形式のファイル。テキスト エディタで開きます。
- ・ XRPT ファイル：PAR の実行で生成されるさまざまなレポートにあるレポート データを含む XML ファイル
- ・ UNROUTES ファイル：配線されていない信号のリストを含むファイル

PAR のプロセス

このセクションでは、PAR による配置および配線、タイミングドリブン PAR、自動タイムスペック設定機能について説明します。

配置

複数の配置フェーズが実行されます。すべての配置フェーズが完了すると、NCD が生成されます。

配置では、PCF ファイルで設定されている制約、接続の長さ、使用可能な配線リソースなどのデータに基づいて、コンポーネントがサイトに配置されます。

MAP を **-timing** (タイミングドリブン パックと配置) を使用して実行した場合、配置は MAP により完了しているので、PAR では配線のみが実行されます。

配線

デザインの配置が完了すると、複数の配線フェーズが実行されます。配線では、デザインの配線を完了し、タイミング制約を満たすため、処理が繰り返し実行されます。デザインが完全に配線されると、NCD が出力されます。

配線ステップ全体を通して、配線が改善されるたびに新しい NCD が生成されます。

メモ： PCF 内でタイミング制約が検出されると、タイミングドリブン方式の配置配線が自動的に実行されます。

タイミングドリブン PAR

タイミングドリブン PAR は、統合スタティック タイミング解析ツールであるザイリンクスのタイミング解析ソフトウェアに基づいており、回路への入力信号には依存していません。配置配線は、デザイン プロセスの初期段階で指定したタイミング制約に従って実行されます。タイミング解析ソフトウェアは PAR と連動し、デザインに設定されたタイミング制約を満たしているかどうかを確認します。

タイミングドリブン PAR を実行するには、次のいずれかの方法でタイミング制約を設定します。

- ・ 回路図入力ツールまたは HDL デザイン入力ツールで、タイミング制約をプロパティとして入力します。ほとんどの場合、NCF ファイルが合成ツールで自動的に生成されます。
- ・ タイミング制約をユーザー制約ファイル (UCF) に記述します。このファイルは、論理的なデザイン データベースを生成する際に NGDDBuild により処理されます。

Constraints Editor を使用すると、タイミング制約を UCF に手作業で入力する必要がなく、簡単に制約を作成できます。Constraints Editor の使用法は、Constraints Editor ヘルプを参照してください。

- ・ MAP で生成される物理制約ファイル (PCF) にタイミング制約を入力します。PCF には、前述した 2 つの方法で指定したタイミング制約と、ファイルに直接入力した制約が含まれます。PCF ファイルを変更することは、通常お勧めできません。

デザインにタイミング制約を設定しなかった場合、または Project Navigator の [Ignore User Timing Constraints] プロパティをオンにした場合、すべての内部クロックに対してタイミング制約が自動的に生成され、PAR の実行中により良いパフォーマンスを得るため調整されます。パフォーマンスは、PAR のエフォートレベルによって決まります。エフォートレベルを std に設定すると、ランタイムは大幅に短縮されますが、パフォーマンスが低くなります。エフォートレベルを high に設定すると、最良のパフォーマンスを得られますが、ランタイムが長くなります。

PCF 内でタイミング制約が検出されると、タイミングドリブン方式の配置配線が自動的に実行されます。PCF は、タイミング解析ソフトウェアへの入力として読み込まれます。制約の詳細は、『[制約ガイド](#)』を参照してください。

メモ： タイミング制約のタイプと値によっては PAR のランタイムが長くなることがあります。

PAR が終了すると、PAR レポートのタイミング サマリを表示したり、TRACE (Timing Reporter And Circuit Evaluator) または Timing Analyzer を使用してデザインのタイミング特性が PCF の制約を満たしているかどうかを確認できます。TRACE は、タイミング警告、タイミング エラー、およびデザインに関するその他の情報をレポートします。TRACE の詳細は、『[TRACE](#)』の章を参照してください。

PAR の構文

デザインを配置配線するには、次の構文を使用します。

```
par [options] infile [.ncd] outfile [pcf_file [.pcf]]
```

- ・ *options* : 「[PAR のオプション](#)」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。
- ・ *infile* : 配置配線を実行するデザイン ファイルです。拡張子 .ncd が付いたファイルを指定する必要がありますが、コマンドラインで拡張子 .ncd を指定する必要はありません。
- ・ *outfile* : PAR の終了後に作成されるターゲット デザイン ファイルです。出力デザイン ファイルが 1 つ作成されるコマンド オプションを指定した場合、出力ファイルの拡張子は .ncd です。拡張子が .ncd の場合は NCD フォーマットの出力ファイルが作成されます。複数の出力デザイン ファイルを作成するコマンド オプションを指定した場合は、出力ファイルに拡張子付ける必要があります。複数の出力ファイルは、デフォルトの拡張子 .ncd のディレクトリ内に保存されます。

メモ： 指定したファイルまたはディレクトリが既に存在している場合は、エラー メッセージが表示され、処理は実行されません。ただし、**-w** オプションを使用すると、既存のファイルが自動的に上書きされます。

pcf file : 物理制約ファイル (PCF) です。このファイルには、デザイン入力で設定した制約、ユーザー制約ファイル (UCF) を使用して追加した制約、PCF に直接追加した制約が含まれます。コマンドラインで PCF を指定しなくても、入力ファイル (*infile*) とルート名が同じで拡張子が *.pcf* のファイルが作業ディレクトリに既に存在する場合、既存の PCF が自動的に使用されます。

例 1

```
par input.ncd output.ncd
```

この例は、*input.ncd* ファイルのデザインを配置配線し、結果を *output.ncd* に出力します。

メモ : 入力 NCD ファイルと同じルート名の PCF があれば、自動的に検出され、読み込まれます。

例 2

```
par -k previous.ncd reentrant.ncd pref.pcf
```

このコマンドは、配置ステップをスキップしてすべての配線情報をロックすることなく保持し (再配線)、*pref.pcf* ファイルのタイミング制約に従って実行します。デザインが完全に配線されていて、タイミング制約が満たされていない場合、そのタイミング要件が達成されるまで、またはタイミング要件が達成できないと判断されるまで、再配線が続行されます。

PAR のオプション

このセクションでは、PAR のコマンドライン オプションについて説明します。

- ・ `-activity_file` (アクティビティ ファイルの指定)
- ・ `-clock_regions` (クロック領域レポートの生成)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-intstyle` (統合スタイル)
- ・ `-filter` (フィルタ ファイルの指定)
- ・ `-k` (再配線)
- ・ `-mt` (マルチスレッド)
- ・ `-nopad` (パッド レポートを生成しない)
- ・ `-ntd` (非タイミングドリブン)
- ・ `-ol` (総体的なエフォートレベル)
- ・ `-p` (配置なし)
- ・ `-pl` (配置エフォートレベル)
- ・ `-power` (低消費電力の PAR)
- ・ `-r` (配線なし)
- ・ `-rl` (配線エフォートレベル)
- ・ `-smartguide` (SmartGuide)
- ・ `-t` (配置コスト テーブルの開始点)
- ・ `-ub` (ボンディングされた I/O を使用)
- ・ `-w` (既存ファイルの上書き)
- ・ `-x` (パフォーマンス評価モード)
- ・ `-xe` (追加エフォートレベル)

`-activity_file` (アクティビティ ファイルの指定)

消費電力最適化用にスイッチ アクティビティ データ ファイルを指定します。

構文

```
-activity_file activity_file .{vhdl|saif}
```

PAR では、`.saif` および `.vcd` の 2 種類のアクティビティ ファイルがサポートされています。

このオプションを使用するには、`-power` オプションを使用する必要があります。

`-clock_regions` (クロック領域レポートの生成)

PAR を実行したときにクロック領域レポートを生成します。

構文

```
-clock_regions generate_clock_region_report
```

このレポートには、各クロック領域でのリソースの使用率、クロック領域内のグローバル クロックの競合に関する情報が含まれます。

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f\(コマンド ファイルの実行\)](#)」を参照してください。

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle *ise|xflow|silent*

次のいずれかのモードに設定します。

- ・ **-intstyle ise**: プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow**: プログラムが統合パッチ フローの一部として実行されます。
- ・ **-intstyle silent**: 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ: Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-filter (フィルタ ファイルの指定)

フィルタ ファイルを指定します。フィルタ ファイルには、プログラムの実行中に生成されるメッセージを保存およびフィルタ処理するための設定が含まれています。

構文

-filter [*filter_file*]

デフォルトのフィルタ ファイル名は *filter.filter* です。

-k (再配線)

既存の配置配線を保持しながら再配線を実行します。デフォルトでは *off* に設定されています。

構文

-k *previous_NCD.ncd reentrant.ncd*

この方法では、既存の配置と配線が出発点として使用されますが、配線は変更される可能性があります。

デザインの一部分を手動で配線してから自動配線を続行する場合、Ctrl + C キーを押して停止した配線を再開する場合、または配線を追加で実行する場合に使用すると有効です。

メモ : Virtex®-5 デバイスでは、[Place And Route Mode] プロパティは [Route Only] および [Reentrant Route] のみ選択可能です。デフォルトでは、Virtex-5 デバイスの場合は [Route Only]、その他のデバイスの場合は [Normal Place and Route] に設定されています。

-mt (マルチスレッド)

PAR で複数のプロセッサを使用できるようにし、配置ツールにマルチスレッド機能を含めます。

メモ : このオプションは、Spartan®-6、Virtex®-6、および Virtex-5 デバイスでのみ使用できます。マルチスレッド機能は、**-smartguide**、**-power on**、**-x**、パーティション、または PCF ファイルのないプロジェクトを使用している場合は使用できません。

構文

-mt off|2|3|4

デフォルトは **off** です。**off** に設定すると、1 つのプロセッサのみが使用されます。**2**、**3**、または **4** に設定すると、その数までのコアが使用されます。

-nopad (パッド レポートを生成しない)

3 つの形式のパッド レポートが生成されないように指定します。

構文

-nopad

デフォルトでは、3 つの形式のパッド レポートがすべて生成されます。

-ntd (非タイミング ドリブン)

タイミング ドリブンでない配置を実行します。

構文

-ntd

このオプションをイネーブルにすると、すべてのタイミング制約が無視され、デザインの配置配線にタイミング情報は使用されません。

メモ : このオプションは、MAP と PAR の両方にあります。フロー全体をタイミング制約なしで実行する場合は、**-ntd** オプションを MAP および PAR の両方で指定してください。

-ol (総体的なエフォート レベル)

PAR の総体的なエフォート レベルを指定します。

構文

-ol std|high

- ・ **std** : エフォート レベルを低に設定します (ランタイム最短、QoR 低)。
- ・ **high** : エフォート レベルを高に設定します (QoR 最良、ランタイム最長)。

デフォルトのエフォート レベルは、すべてのアーキテクチャで **high** です。

このオプションは、**-timing** オプションを使用してタイミング ドリブンのパックと配置を実行する場合にのみ使用できます。

メモ: MAP のエフォートレベルは、PAR のエフォートレベル以上にすることをお勧めします。

例

```
par -ol std design.ncd output.ncd design.pcf
```

この例では、PAR の総体的なエフォートレベルを **std** (ランタイム最短、QoR 低) に設定します。

-p (配置なし)

配置を実行せずに配線プロセスを実行します。このオプションを使用する時点でデザインが完全に配置されていないと、エラー メッセージが表示され、PAR が終了します。

構文

-p

このオプションを使用すると、既存の配線が解除されてから配線が開始します。既存の配線を保持するには、**-p** オプションではなく、**-k** (再配線) オプションを使用します。

メモ: 以前の NCD の配置を保持し、配線のみを再実行する場合にこのオプションを使用してください。

例

```
par -p design.ncd output.ncd design.pcf
```

このコマンドを使用すると、配置がスキップされ、配線のみが実行されます。配置が完了していないデザインを使用すると、エラー メッセージが表示され、PAR は実行されません。

-pl (配置エフォート レベル)

PAR の配置ツールのエフォートレベルを設定します。総体的なエフォートレベルの設定より優先されます。

メモ: このオプションは、Spartan®-3、Spartan-3A、Spartan-3E、および Virtex®-4 デバイスでのみ使用できます。その他のデバイスでは、**-ol** (総体的なエフォートレベル) を使用してください。

構文

```
-pl std|high
```

- ・ **std**: 配置エフォートレベルを低に設定します (ランタイム最短)。この設定は、比較的単純なデザインに適しています。
- ・ **high**: 配置エフォートレベルを高に設定します (最良の結果、ランタイム最長)。この設定は、複雑なデザインに適しています。

デフォルトの配置エフォートレベルは、**std** です。

例

```
par -pl high design.ncd output.ncd design.pcf
```

総体的なエフォートレベルを無効にし、配置ツールのエフォートレベルを **high** に設定します。

-power (低消費電力の PAR)

タイミングがクリティカルではない信号の容量を最適化します。

構文

-power [on|off]

デフォルトでは、**off** に設定されています。**on** に設定すると、消費電力最適化用にスイッチ アクティビティ ファイルも指定できます。「[-activity_file \(アクティビティ ファイルの指定\)](#)」を参照してください。

-r (配線なし)

配置を実行した後、配線を実行せずに終了します。

構文

-r

メモ：配置済みのデザインで配置をスキップするには、**-p** (配置なし) オプションを使用します。

例

```
par -r design.ncd route.ncd design.pcf
```

このコマンドを使用すると、配線プロセスの前で PAR が終了します。

-rl (配線エフォート レベル)

PAR の配線ツールのエフォートレベルを設定します。総体的なエフォートレベルの設定より優先されます。

メモ：このオプションは、Spartan®-3、Spartan-3A、Spartan-3E、および Virtex®-4 デバイスでのみ使用できます。

構文

-rl std|high

- ・ **std**：配線エフォートレベルを低に設定します (ランタイム最短)。この設定は、比較的単純なデザインに適しています。
- ・ **high**：配線エフォートレベルを高に設定します (最良の結果、ランタイム最長)。この設定は、複雑なデザインに適しています。

デフォルトの配線エフォートレベルは、**std** です。

例

```
par -rl high design.ncd output.ncd design.pcf
```

総体的なエフォートレベルを無効にし、配線ツールのエフォートレベルを **high** に設定します。

-smartguide (SmartGuide)

前回のインプリメンテーション結果 (配置配線済みの NCD ファイル) を、インプリメンテーションを実行する際のガイドとして使用します。SmartGuide™ テクノロジを使用すると、タイミングドリブン パックと配置 (**-timing** オプション) が自動的に設定されます。タイミングドリブン パックと配置は、リソースの使用率が高いデザインでパフォーマンスとタイミングを向上します。

SmartGuide テクノロジをイネーブルにする前に、**map -timing** オプションを使用して配置配線済みの NCD ガイドファイルを作成しておく、結果が向上する場合があります。SmartGuide テクノロジは、コマンドラインまたは Project Navigator の [Design] パネルの [Hierarchy] ペインからイネーブルにできます。

構文

-smartguide *design_name.ncd*

SmartGuide テクノロジでのガイドは、MAP の BEL レベルで行われます。ガイドには、パック、配置、配線が含まれます。デザインのパックおよび配置が最適に変更され、PAR の段階でネットが新たに配線されます。SmartGuide テクノロジでは、デザインの変更されていない部分のインプリメンテーションを保持し、変更された部分でタイミング要件を満たすことが第 1 の目標であり、ランタイムの短縮が第 2 の目標です。変更されていない部分のインプリメンテーションは変更されず、同じタイミング スコアが保持されます。タイミングを満たしておらず、変更されていないパスは、100% ガイドされます。タイミングを満たしておらず、変更されたパスは、再インプリメントされます。

MAP の実行結果は、マップ レポート ファイル (MRP) に保存されます。ガイドされたネット、新しいコンポーネント、ガイドされたコンポーネント、再インプリメントされたコンポーネントの数など、ガイドの統計がリストされます。これは予測される統計であり、最終的な統計は配置配線 レポート (PAR) に記述されます。PAR では、ガイド レポート ファイル (GRF) も生成されます。PAR コマンドラインで **-smartguide** オプションを使用すると、詳細なガイド レポート ファイルが生成され、**-smartguide** を使用しない場合はサマリ ガイド レポート ファイルが生成されます。ガイド レポート ファイルには、再インプリメントされたコンポーネントおよびネット、新規のコンポーネントおよびネットがリストされます。詳細および例は、「[ガイド レポート ファイル \(GRF\)](#)」を参照してください。

-t (配置コスト テーブル)

配置で使用するコスト テーブルを指定します。

構文

-t [*placer_cost_table*]

placer_cost_table: 配置で使用するコスト テーブルを指定します。有効な値は 1 ~ 100 で、デフォルト値は 1 です。

複数のコスト テーブルを使用してインプリメンテーションを複数回実行する方法については、「[SmartXplorer](#)」の章を参照してください。

メモ: このオプションは、Spartan®-3、Spartan-3A、Spartan-3E、および Virtex®-4 デバイスでのみ使用できます。その他のデバイスで使用するコスト テーブルを変更する場合は、MAP の **-t** オプションを使用してください。

例

```
par -t 10 -pl high -rl std design.ncd output_directory design.pcf
```

このコマンドを使用すると、コストテーブル 10 が使用されます。配置のエフォートレベルは high で、配線のエフォートレベルは std です。

-ub (ボンディングされた I/O を使用)

ボンディングされた I/O を通過する配線を許可します。

構文

-ub

デフォルト (-ub オプションなし) では、MAP で内部ロジックとして識別された I/O ロジックは、ボンディングされていない I/O サイトにしか配置できません。-ub オプションを指定すると、I/O パッドが使用されていないボンディングされた I/O に内部 I/O ロジックを配置できます。このオプションを使用する場合は、外部信号、電源、またはグランドに接続されているボンディングされた I/O にロジックが配置されていないことを確認してください。これを防ぐには、該当するボンディングされた I/O に PROHIBIT 制約を指定します。制約の詳細は、『[制約ガイド](#)』を参照してください。

-w (既存ファイルの上書き)

既存の NCD ファイルを上書きします。

構文

-w

デフォルト (-w オプションなし) では、既存の NCD は上書きされません。この場合、NCD が既に存在していると、PAR でエラーが発生し、配置配線が実行されません。

-x (パフォーマンス評価モード)

既存のタイミング制約を無視し、すべての内部クロックに対してタイミング制約を新たに生成します。

構文

-x

このオプションは、PCF にタイミング制約が設定されているが、ツールにより生成されたタイミング制約を使用して PAR を実行し、内部クロックのパフォーマンスを評価する場合に使用します。この機能を「パフォーマンス評価モード」と呼びます。このモードは、-x オプションを使用するか、デザインにタイミング制約が設定されていない場合に実行されます。ツールにより生成されたタイミング制約は、各クロックに個別に設定され、PAR の実行中に調整されます。PAR のエフォートレベルにより、ランタイムを最短にすることを目標とするか (std)、パフォーマンスを最高にすることを目標とするか (high) を制御します。

design.pcf のタイミング制約はすべて無視されますが、LOC および AREA_RANGE などの物理制約はすべて使用されます。

-xe (追加エフォート レベル)

困難なタイミングを満たすための配置配線の追加エフォートレベルを指定します。

構文

-xe n|c

n (標準) : タイミング制約を満たすことができないと判断されると、そのことを示すメッセージが表示され、PAR が終了します。

c (継続) : タイミング制約を満たすことができないと判断されても、タイミングが改善されなくなるまで配線が実行されます。

メモ : 追加エフォートレベルを **c** に設定すると、ランタイムが長くなります。

-xe オプションを使用するには、**-ol** (総体的 なエフォートレベル) オプションを **high** に設定するか、**-pl** (配置エフォートレベル) オプションおよび **-rl** (配線エフォートレベル) オプションを **high** に設定する必要があります。

例

```
par -ol high -xe n design.ncd output.ncd design.pcf
```

このコマンドを使用すると、追加エフォートレベルが使用され、タイミング制約が満たされないと判断されると PAR が終了します。

PAR のレポート

PAR の出力は、配置配線済みの NCD ファイル (出力デザイン ファイル) です。PAR では、出力デザイン ファイルに加え、拡張子が **.par** の配置配線レポートファイルが生成されます。**-smartguide** オプションを使用すると、ガイド レポート ファイル (GRF) も生成されます。

配置配線 (PAR) レポートには、配置配線の実行に関する情報と制約に関するメッセージが含まれます。PAR のレポートの詳細は、「[配置配線 \(PAR\) レポート](#)」および「[ガイド レポート ファイル \(GRF\)](#)」を参照してください。

PAR の実行時に NCD ファイルを 1 つ生成するオプションを指定した場合、NCD ファイル、PAR ファイル、パッド ファイルが出力されます。**-smartguide** オプションを使用すると、GRF ファイルも生成されます。PAR および GRF ファイル、パッド ファイルのルート名は、NCD ファイルのルート名と同じです。

メモ : ReportGen は、パッド レポート ファイル (拡張子が **.pad**、**pad.txt**、**pad.csv**) を生成するユーティリティです。ピン配置が記述された PAD ファイルは、ユーザー スクリプトで解析するためのファイルです。PAD.TXT ファイルは、テキスト エディタで開くことができます。PAD.CSV ファイルは、表計算プログラムで使用します。レポートの生成およびカスタマイズについては、「[ReportGen](#)」を参照してください。

レポートは、等幅フォントでの表示用にフォーマットされます。レポートの表示に使用するテキスト エディタで等幅ではないフォントを使用した場合、レポートの列は正しく並びません。PAD.CSV レポートは、表計算プログラムにインポートしたり、ユーザー スクリプトで解析できるようなフォーマットされています。PAR で個別のファイルまたは PAR ファイル内に生成されるレポートは、**<design name>_par.xrpt** という XML データ ファイルでも参照できます。

配置配線 (PAR) レポート

PAR レポート ファイルは、PAR の実行に関する情報を含む ASCII 形式のテキスト ファイルです。レポートの情報は、指定したデバイスおよびオプションによって異なります。このレポートには、配置配線が完了するまでの各ステップが記述されます。

PAR レポートの構成

PAR レポートには、次のセクションがあります。

- ・ **Design Information (デザイン情報)**: PAR コマンドライン、デザインが配置配線されたデバイス、入力ファイル (NCD および PCF) に関する情報、配置配線を実行した日時などを示します。警告メッセージおよび情報メッセージが表示される場合もあります。
- ・ **Design Summary (デザイン サマリ)**: 使用しているリソースの内訳とデバイス使用率のサマリを示します。
- ・ **Placer Results (配置結果)**: 配置の各フェーズと、どのフェーズが実行されたかを示します。チェックサム値はデバッグ目的のみに使用され、配置の質を表すものではありません。

メモ: MAP の **-timing** オプションと SmartGuide™ を使用した場合、配置は MAP で実行されるので、配置結果は配置配線レポートには含まれません。

- ・ **Router Results (配置結果)**: 配線の各フェーズと未配線ネットの数を示します。また、タイミング スコアがかつこ内に表示されます。
- ・ **SmartGuide Report (SmartGuide レポート)**: ガイドの結果を示します。ガイドされたコンポーネント、再インプリメントされたコンポーネント、新規コンポーネントの数を含め、入力デザインとガイド デザインとの差が正確に記述されます。
- ・ **Partition Implementation Status (パーティションインプリメンテーションステータス)**: 保持されたパーティション、再インプリメントされたパーティションとその理由を示します。デザインにパーティションがない場合は、そのことが示されます。
- ・ **Clock Report (クロックレポート)**: デザインで使用されるすべてのクロックをリストし、配線リソース、ファンアウト数、最大ネット スキュー、最大遅延を表形式で示します。[Locked] 列は、クロックドライバ (BUFGMUX) が特定のサイトに割り当てられているか未接続かを示します。

メモ: この表に示されるクロック スキューおよび遅延は、TRACE および Timing Analyzer でレポートされる値とは異なります。PAR ではクロック ピンを駆動するネットのみが考慮されるのに対し、TRACE および Timing Analyzer ではクロック パス全体が対象となるからです。

- ・ **Timing Score (タイミング スコア)**: 入力 PCF ファイルに含まれるタイミング制約に関する情報と、満たされたタイミング制約の数を示します。このセクションの 1 行目には、タイミング スコアが示されます。タイミング制約が満たされていない場合、タイミング スコアは 0 より大きい値になります。スコアが低いほど、良い結果であることを示します。

メモ: 入力 PCF ファイルに制約が含まれていない場合や、**-x** オプションを使用した場合、このセクションに制約の表は示されません。

- ・ **Summary (サマリ)**: デザインが完全に配置配線されたかを示します。また、PAR の実行にかかった総時間を実時間と CPU 時間で示し、エラー、警告、情報メッセージの数をリストします。

PAR レポートの例

次に、PAR レポートの例を示します。ほとんどの PAR レポート ファイルは、ここで示す例よりもかなり大きくなります。この例は、**-smartguide** オプションを使用して実行した結果です。SmartGuide を使用すると配置が MAP で実行されるので、「Placer Results」セクションは含まれていません。ここに示す例では、一部の行は削除されています。

```
par -w -intstyle ise -ol high -mt off wave_gen_map.ncd wave_gen.ncd
```

wave_gen.pcf

Constraints file: wave_gen.pcf.

Loading device for application Rf_Device from file '6v1x75t.nph' in environment /build/xfndry/M.49/rtf.

"wave_gen" is an NCD, version 3.2, device xc6v1x75t, package ff484, speed -1

Initializing temperature to 85.000 Celsius. (default - Range: 0.000 to 85.000 Celsius)

Initializing voltage to 0.950 Volts. (default - Range: 0.950 to 1.050 Volts)

Device speed data version: "ADVANCED 1.05c 2010-02-23".

Device Utilization Summary:

Slice Logic Utilization:

Number of Slice Registers:	569 out of	93,120	1%
Number used as Flip Flops:	568		
Number used as Latches:	1		
Number used as Latch-thrus:	0		
Number used as AND/OR logics:	0		
Number of Slice LUTs:	958 out of	46,560	2%
Number used as logic:	854 out of	46,560	1%
Number using O6 output only:	658		
Number using O5 output only:	15		
Number using O5 and O6:	181		
Number used as ROM:	0		
Number used as Memory:	0 out of	16,720	0%
Number used exclusively as route-thrus:	104		
Number with same-slice register load:	102		
Number with same-slice carry load:	2		
Number with other load:	0		

Slice Logic Distribution:

Number of occupied Slices:	328 out of	11,640	2%
Number of LUT Flip Flop pairs used:	988		
Number with an unused Flip Flop:	569 out of	988	57%
Number with an unused LUT:	30 out of	988	3%
Number of fully used LUT-FF pairs:	389 out of	988	39%
Number of slice register sites lost to control set restrictions:	0 out of	93,120	0%

A LUT Flip Flop pair for this architecture represents one LUT paired with one Flip Flop within a slice. A control set is a unique combination of clock, reset, set, and enable signals for a registered element.

The Slice Logic Distribution report is not meaningful if the design is over-mapped for a non-slice resource or if Placement fails.

OVERMAPPING of BRAM resources should be ignored if the design is over-mapped for a non-BRAM resource or if placement fails.

IO Utilization:

Number of bonded IOBs:	17 out of	240	7%
IOB Flip Flops:	11		

Specific Feature Utilization:

Number of RAMB36E1/FIFO36E1s:	0 out of	156	0%
Number of RAMB18E1/FIFO18E1s:	2 out of	312	1%
Number using RAMB18E1 only:	2		
Number using FIFO18E1 only:	0		
Number of BUFG/BUFGCTRLs:	3 out of	32	9%
Number used as BUFGs:	3		
Number used as BUFGCTRLs:	0		
Number of ILOGICE1/ISERDESE1s:	0 out of	360	0%
Number of OLOGICE1/OSERDESE1s:	11 out of	360	3%
Number used as OLOGICE1s:	11		
Number used as OSERDESE1s:	0		
Number of BSCANS:	0 out of	4	0%
Number of BUFHCEs:	1 out of	72	1%
Number of BUFIODQs:	0 out of	36	0%
Number of BUFRs:	0 out of	18	0%
Number of CAPTUREs:	0 out of	1	0%

```

Number of DSP48E1s:          0 out of    288    0%
Number of EFUSE_USRs:        0 out of     1    0%
Number of GTXE1s:            0 out of     8    0%
Number of IBUFDS_GTXE1s:     0 out of     6    0%
Number of ICAPs:             0 out of     2    0%
Number of IDELAYCTRLs:       0 out of     9    0%
Number of IDELAYE1s:         0 out of   360    0%
Number of MMCM_ADVs:         1 out of     6   16%
Number of PCIE_2_0s:         0 out of     1    0%
Number of STARTUPs:          0 out of     1    0%
Number of SYSMONs:           0 out of     1    0%
Number of TEMAC_SINGLES:     0 out of     4    0%

```

```

Overall effort level (-ol):  High
Router effort level (-rl):   High

```

INFO:Timing:3386 - Intersecting Constraints found and resolved.

For more information, see the TSI report. Please consult the Xilinx

Command Line Tools User Guide for information on generating a TSI report.

Starting initial Timing Analysis. REAL time: 22 secs

Finished initial Timing Analysis. REAL time: 22 secs

Starting Router

Phase 1 : 4986 unrouted; REAL time: 24 secs

Average Wirelength on CLB-Grid (driver-load model):

Fanout	NumSigs	Sigs(%)	AvgWireLen	AvgWireLen Per Pin
1	679	53.0	1.0	1.0
2	182	14.0	3.0	1.5
3	121	9.0	7.0	2.3
4	53	4.0	8.0	2.0
10	153	12.0	15.0	1.5
50	81	6.0	122.0	2.4
100	4	0.0	1428.0	14.3
500	0	0.0	0.0	0.0
5000	0	0.0	0.0	0.0
20000	0	0.0	0.0	0.0
50000	0	0.0	0.0	0.0

Wirelength distribution for Fanout-1:

Wirelength	Signals	percent
0	222	32.00
1	171	25.00
2	113	16.00
3	94	13.00
4	35	5.00
5	10	1.00
6	34	5.00

Total wirelength for fanout 1,2,3 and 4 nets: 3213

Phase 2 : 4451 unrouted; REAL time: 25 secs

Phase 3 : 1632 unrouted; REAL time: 27 secs

Phase 4 : 1632 unrouted; (Setup:0, Hold:2924, Component Switching Limit:0) REAL time: 32 secs

Updating file: wave_gen.ncd with current fully routed design.

Phase 5 : 0 unrouted; (Setup:0, Hold:2752, Component Switching Limit:0) REAL time: 34 secs

Phase 6 : 0 unrouted; (Setup:0, Hold:2752, Component Switching Limit:0) REAL time: 34 secs

Phase 7 : 0 unrouted; (Setup:0, Hold:2752, Component Switching Limit:0) REAL time: 34 secs

Phase 8 : 0 unrouted; (Setup:0, Hold:2752, Component Switching Limit:0) REAL time: 34 secs

Phase 9 : 0 unrouted; (Setup:0, Hold:0, Component Switching Limit:0) REAL time: 35 secs

Phase 10 : 0 unrouted; (Setup:0, Hold:0, Component Switching Limit:0) REAL time: 36 secs

Total REAL time to Router completion: 36 secs

Total CPU time to Router completion: 35 secs

Partition Implementation Status

No Partitions were found in this design.

Generating "PAR" statistics.

Generating Clock Report

Clock Net	Resource	Locked	Fanout	Net Skew(ns)	Max Delay(ns)
clk_rx	BUFGCTRL_X0Y1	No	97	0.152	1.662
clk_tx	BUFGCTRL_X0Y2	No	65	1.201	1.661
clk_samp	Local		19	0.066	1.039
clk_gen_i0/clk_core_ i0/mmcadv_inst_ML_ NEW_I1	Local		3	0.000	1.184
clk_gen_i0/clk_core_ i0/MMCM_PHASE_CALIBR ATTION_ML_LUT2_7_ML_N EW_CLK	Local		3	0.260	0.624

* Net Skew is the difference between the minimum and maximum routing only delays for the net. Note this is different from Clock Skew which is reported in TRCE timing report. Clock Skew is the difference between the minimum and maximum path delays which includes logic delays.

Timing Score: 0 (Setup: 0, Hold: 0, Component Switching Limit: 0)

Number of Timing Constraints that were not applied: 1

Asterisk (*) preceding a constraint indicates it was not met.

This may be due to a setup or hold violation.

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
NET "lb_ctl_i0/debouncer_i0/meta_harden_s ignal_in_i0/signal_meta" MAXDELAY = 2 ns	MAXDELAY	1.460ns	0.540ns	0	0
NET "samp_gen_i0/meta_harden_samp_gen_go_ i0/signal_meta" MAXDELAY = 2 ns	MAXDELAY	1.590ns	0.410ns	0	0
NET "clkx_pre_i0/meta_harden_bus_new_i0/s ignal_meta" MAXDELAY = 2 ns	MAXDELAY	1.615ns	0.385ns	0	0
OFFSET = IN 5 ns VALID 8 ns BEFORE COMP "	SETUP	1.692ns	3.308ns	0	0
clk_pin" "RISING"	HOLD	3.441ns		0	0
OFFSET = OUT 8 ns AFTER COMP "clk_pin"	MAXDELAY	1.778ns	6.222ns	0	0
NET "clkx_spd_i0/meta_harden_bus_new_i0/s	MAXDELAY	1.810ns	0.190ns	0	0

signal_meta" MAXDELAY = 2 ns					
NET "clkx_nsamp_i0/meta_harden_bus_new_i0 /signal_meta" MAXDELAY = 2 ns	MAXDELAY	1.810ns	0.190ns	0	0
NET "uart_rx_i0/meta_harden_rxd_i0/signal_meta" MAXDELAY = 2 ns	MAXDELAY	1.810ns	0.190ns	0	0
TS_clk_tx_to_clk_rx = MAXDELAY FROM TIMEGRP "TNM_clk_tx" TO TIMEGRP "TNM_clk_rx" 5 ns DATAPATHONLY	SETUP HOLD	3.509ns 0.115ns	1.491ns	0 0	0 0
TS_clk_rx_to_clk_tx = MAXDELAY FROM TIMEGRP "TNM_clk_rx" TO TIMEGRP "TNM_clk_tx" 5 ns DATAPATHONLY	SETUP HOLD	3.533ns 0.127ns	1.467ns	0 0	0 0
TS_clk_gen_i0_clk_core_i0_clkout1 = PERIOD TIMEGRP "clk_gen_i0_clk_core_i0_clkout1" TS_clk_pin / 0.909090909 HIGH 50%	SETUP HOLD	3.574ns 0.014ns	5.746ns	0 0	0 0
TS_clk_gen_i0_clk_core_i0_clkout0 = PERIOD TIMEGRP "clk_gen_i0_clk_core_i0_clkout0" TS_clk_pin HIGH 50%	SETUP HOLD	3.811ns 0.030ns	6.189ns	0 0	0 0
COMP "spi_clk_pin" OFFSET = OUT 8 ns AFTER COMP "clk_pin" "RISING"	MAXDELAY	4.402ns	3.598ns	0	0
COMP "spi_clk_pin" OFFSET = OUT 8 ns AFTER COMP "clk_pin" "FALLING"	MAXDELAY	4.402ns	3.598ns	0	0
TS_clk_pin = PERIOD TIMEGRP "clk_pin" 10 ns HIGH 50%	MINLOWPULSE	5.840ns	4.160ns	0	0
TS_to_bcd = MAXDELAY FROM TIMEGRP "TNM_send_resp_data" TO TIMEGRP "TNM_to_bcd_flops" TS_clk_gen_i0_clk_core_i0_clkout0 * 2	SETUP HOLD	7.736ns 0.415ns	12.264ns	0 0	0 0
TS_clk_samp = MAXDELAY FROM TIMEGRP "TNM_clk_samp" TO TIMEGRP "TNM_clk_samp" TS_clk_gen_i0_clk_core_i0_clkout1 * 32	SETUP HOLD	347.413ns 0.108ns	4.587ns	0 0	0 0
TS_uart_rx_ctl = MAXDELAY FROM TIMEGRP "TNM_uart_rx_ctl" TO TIMEGRP "TNM_uart_rx_ctl" TS_clk_gen_i0_clk_core_i0_clkout0 * 54	SETUP HOLD	537.423ns 0.091ns	2.577ns	0 0	0 0
TS_uart_tx_ctl = MAXDELAY FROM TIMEGRP "TNM_uart_tx_ctl" TO TIMEGRP "TNM_uart_tx_ctl" TS_clk_gen_i0_clk_core_i0_clkout1 * 54	SETUP HOLD	590.988ns 0.091ns	3.012ns	0 0	0 0

Derived Constraint Report

Review Timing Report for more details on the following derived constraints.

To create a Timing Report, run "trce -v l2 -fastpaths -o design_timing_report design.ncd design.pcf" or "Run Timing Analysis" from Timing Analyzer (timingan).

Derived Constraints for TS_clk_pin

Constraint	Period Requirement	Actual Period		Timing Errors		Paths Analyzed	
		Direct	Derivative	Direct	Derivative	Direct	Derivative
TS_clk_pin	10.000ns	4.160ns	6.189ns	0	0	0	139441872
TS_clk_gen_i0_clk_core_i0_clkout0	10.000ns	6.189ns	6.132ns	0	0	30713	139383971
TS_to_bcd	20.000ns	12.264ns	N/A	0	0	139383821	0
TS_uart_rx_ctl	540.000ns	2.577ns	N/A	0	0	150	0
TS_clk_gen_i0_clk_core_i0_clkout1	11.000ns	5.746ns	0.143ns	0	0	23229	3959
TS_uart_tx_ctl	594.000ns	3.012ns	N/A	0	0	92	0

	TS_clk_samp		352.000ns	4.587ns	N/A	0	0	3867	0
+-----+-----+-----+-----+-----+-----+-----+-----+									

All constraints were met.

Generating Pad Report.

All signals are completely routed.

Total REAL time to PAR completion: 38 secs
Total CPU time to PAR completion: 37 secs

Peak Memory Usage: 569 MB

Placer: Placement generated during map.
Routing: Completed - No errors found.
Timing: Completed - No errors found.

Number of error messages: 0
Number of warning messages: 0
Number of info messages: 1

Writing design to file wave_gen.ncd

PAR done!

ガイド レポート ファイル (GRF)

ガイド レポート ファイル (GRF) は、ガイドの結果を示す ASCII 形式のテキスト ファイルで、PAR レポートと同じサマリ情報と、ガイドされなかったコンポーネントおよびネットが示されます。GRF ファイルに含まれていないコンポーネントまたはネットは、ガイドされています。ガイドされたコンポーネントおよびネットは、ファイルのサイズを削減するためリストされていません。

ガイド レポートの構成

GRF ファイルには、SmartGuide™ の結果を示す次のセクションが含まれます。

SmartGuide Results (SmartGuide 結果) : 配線が実行された後のガイド結果のサマリを示します。入力デザインとガイド デザインの差を、次の項目別に示します。

- ・ Number of Guided Components (ガイドされたコンポーネントの数) : ガイドされたコンポーネントは、入力デザインとガイド デザインで同じ名前と同じ場所に配置されたものです。LUT の論理式、ピンなどが異なる場合があります。
- ・ Number of Re-implemented Components (再インプリメントされたコンポーネントの数) : 再インプリメントされたコンポーネントは、入力デザインとガイド デザインで同じ名前、ガイド デザインでは配置されていないか、デザインの総体的なタイミングを満たすために移動されたものです。
- ・ Number of New/Changed Components (新規/変更されたコンポーネントの数) : 新規または変更されたコンポーネントは、ガイド デザインにはないが入力デザインには存在するコンポーネントです。デザイン ソースが変更されたか、合成により名前が変更されています。
- ・ Number of Guided Nets (ガイドされたネットの数) : ガイドされたネットは、入力デザインとガイド デザインでソース ピンとロード ピンが同じで、デバイス上の物理的な配線がまったく同じものです。
- ・ Number of partially guided Nets (部分的にガイドされたネット) : 部分的にガイドされたネットは、入力デザインとガイド デザインの両方に存在し、一部またはすべての配線セグメントが異なるものです。
- ・ Number of Re-routed Nets (再配線されたネット) : 再配線されたネットは、入力デザインとガイド デザインの両方に存在し、すべての配線セグメントが異なるものです。デザインの総体的なタイミングを満たすために再配線されています。

メモ : SmartGuide では、ガイドにネット名は使用されないで、ネット名を変更しても結果は変わりません。ネットのソース ピンとロード ピンを比較し、ガイド可能かどうか判断されます。

- ・ Number of New/Changed Nets (新規/変更されたネットの数) : 新規または変更されたネットは、入力デザインにのみ存在するネットです。デザイン ソースが変更されたか、合成によりネットの接続が変更されています。

GRF ファイルには、次の詳細も示されます。

- ・ 再インプリメントされたコンポーネント
- ・ 新規または変更されたコンポーネント
- ・ 再インプリメントされたネットワーク
- ・ 新規または変更されたネットワーク

GRF レポート ファイルの例

次に、GRF レポートの例を示します。ほとんどの GRF レポート ファイルは、ここで示す例よりも大きくなります。

```
Release 11.1 - par HEAD
Copyright (c) 1995-2009 Xilinx, Inc. All rights reserved.

Tue Oct 17 20:57:38 2009

SmartGuide Results
-----
This section describes the guide results after invoking the Router.
This report accurately reflects the differences between the input design and the guide design.

Number of Components in the input design | 99
  Number of guided Components           | 99 out of 99 100.0%
  Number of re-implemented Components   | 0 out of 99 0.0%
  Number of new/changed Components      | 0 out of 99 0.0%
Number of Nets in the input design       | 67
  Number of guided Nets                  | 65 out of 67 97.0%
  Number of re-routed Nets               | 2 out of 67 3.0%
  Number of new/changed Nets             | 0 out of 67 0.0%

The following Components were re-implemented.
-----

The following Components are new/changed.
-----

The following Nets were re-routed.
-----
GLOBAL_LOGIC0.
GLOBAL_LOGIC1.

The following Nets are new/changed.
-----
```

ReportGen

ReportGen ユーティリティは、ReportGen オプションを使用してコマンドラインで指定されたレポートを生成します。NCD を入力ファイルとして読み込み、さまざまなパッドレポートや、著作権情報およびレポートの使用法を含むログ ファイルを出力します。

メモ： レポートによっては、入力として配置配線済みの NCD ファイルが必要です。

ReportGen の構文

ReportGen を実行するには、次の構文を使用します。

reportgen [*options*] *infile* [.ncd]

- ・ *options* : 「ReportGen のオプション」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。
- ・ *infile* : 配置配線を実行するデザイン ファイルです。拡張子 .ncd が付いたファイルを指定する必要がありますが、コマンドラインで拡張子を指定する必要はありません。

ReportGen の入力ファイル

ReportGen の入力ファイルは、次のとおりです。

NCD ファイル： マップ済みのデザイン

ReportGen の出力ファイル

ReportGen の出力レポート ファイルは、次のとおりです。

- ・ **DLY ファイル** : デザインの各ネットに関する遅延情報を含むファイル
- ・ **PAD ファイル** : I/O ピン割り当て情報を記述した、解析可能なデータベース形式のファイル
- ・ **CSV ファイル** : I/O ピン割り当て情報を記述した、表計算プログラムでサポートされるファイル
- ・ **TXT ファイル** : I/O ピン割り当て情報を記述した ASCII 形式のファイル。テキスト エディタで開きます。
- ・ **CLK_RGN ファイル** : デザインのグローバル クロック領域の使用に関する情報を記述したファイル。Virtex®-4 および Virtex-5 アーキテクチャでのみ生成されます。

ReportGen により出力されたファイルは、作業ディレクトリか、コマンドラインの **-o** オプションで指定したディレクトリに保存されます。パッド ファイルのルート名は、基本的に出力デザイン ファイルと同じルート名ですが、TXT および CSV ファイルの場合は、`output_pad.txt` のようにルート名の後に `_pad` が付きます。

ReportGen のオプション

コマンドラインから ReportGen を実行するときにオプションを指定すると、レポート出力をカスタマイズできます。生成するレポートの種類を指定する必要があります。

PAD レポートの列には、SPLIT や NONE など、使用される DCI 終端のタイプが示されます。

次に、ReportGen のオプションと構文、その機能を示します。

オプション	機能	使用法
-clock_regions	クロック領域レポートを生成します。	<code>reportgen -clock_regions</code>
-delay	遅延レポートを生成します。	<code>reportgen -delay</code>
-f	コマンド ファイルで指定したコマンドライン引数とオプションを実行します。	<code>reportgen -f cmdfile.cmd</code>
-h	ReportGen の使用法とヘルプを表示します。	<code>reportgen -h</code>
-intstyle	実行している統合スタイルに基づいて、画面への出力をエラーと警告メッセージに制限します。	<code>reportgen -intstyle {ise xflow silent}</code>
-o	レポートのファイル名と保存先を指定します。	<code>reportgen -o</code>
-pad	パッド レポート ファイルを生成します。このオプションを指定した場合、 -padfmt および -padsortcol を使用してパッド レポートを制御できます。	<code>reportgen design.ncd -pad</code>
-padfmt all csv pad text	パッド レポートの形式を指定します。このオプションを使用するには、 -pad を指定する必要があります。	<code>reportgen design.ncd -pad -padfmt all csv pad text</code>
-padsortcol	パッド レポートに表示する列と並び替え順を指定します。このオプションを使用するには、 -pad を指定する必要があります。	<code>reportgen design.ncd -pad -padfmt csv -padsortcol 1, 3:5, 8</code>

オプション	機能	使用法
	デフォルト: 並べ替えなし、すべての列を表示	値および範囲は、カンマで区切ります。たとえば、「1, 3:5, 8」と指定すると、1 列目を基準に並べ替えられ、1、3、4、5、および 8 列目が表示されます。
-unrouted_nets	未配線のネットワーク レポートを生成します。	reportgen -unrouted_nets

PAR の停止

Ctrl + C キーを割り込み文字として設定しないと、**Ctrl + C** キーを押しても PAR を停止できません。割り込み文字を設定するには、`.login` または `.cshrc` ファイルに「**stty intr ^C**」と入力します。

Ctrl + C キーを押して PAR を停止すると、数秒以内に次のメッセージが表示されます。

```
Ctrl-C interrupt detected.
STATUS:
+-----+
| Most recent SmartPreview on disk: | xxx.ncd |
| Fully placed: | YES |
| Fully routed: | YES |
| SmartPreview status: | ready for bitgen |
| Timing score: | 988 |
| Timing errors: | 25 |
| Number of failing constraints: | 1 |
+-----+
Option 3 in the menu below will save the SmartPreview design file and a timing summary in ./SmartPreview.

MENU: Please choose one of the following options:
1. Ignore interrupt and continue processing.
2. Exit program immediately.
3. Preserve most recent SmartPreview and continue (see STATUS above).
4. Cancel current 'par' job at next check point.
```

メモ: ワークステーションのバックグラウンドで PAR を実行している場合、**-fg** コマンドを使用してそのプロセスを前面に移動してから、PAR を停止する必要があります。

PAR を実行した後、FPGA Editor で NCD ファイルを開き、結果を検証したり、編集できます。また、TRACE や Timing Analyzer を使用してスタティック タイミング解析も実行できます。デザインの配線が希望どおりであれば、生成されたファイルを BitGen の入力として使用します。BitGen は、デザイン コンフィギュレーションをターゲット FPGA にダウンロードする際に使用されるファイルを生成します。BitGen の詳細は、「[BitGen](#)」の章を参照してください。

SmartXplorer

この章には、次のセクションが含まれています。

- ・ [SmartXplorer の新機能](#)
- ・ [SmartXplorer の概要](#)
- ・ [SmartXplorer の使用](#)
- ・ [最高のストラテジの選択](#)
- ・ [複数のストラテジの同時実行](#)
- ・ [カスタム ストラテジ](#)
- ・ [SmartXplorer の構文](#)
- ・ [SmartXplorer のレポート](#)
- ・ [SSH で SmartXplorer を実行するための設定](#)

SmartXplorer の新機能

12.2 の新機能

- ・ SmartXplorer で自動的に BitGen が実行され、最高の結果を使用して FPGA プログラムデータ ファイルが生成されるようになりました。新しい SmartXplorer **-bo** オプションを使用すると、BitGen オプションを変更できます。**-bo off** を使用すると、BitGen は実行されません。
- ・ **-best** オプションを使用すると、SmartXplorer 実行のすべての結果ではなく、最高のものから N 個の結果のみを保存できます。

12.1 の新機能

SmartXplorer をコマンド ラインから実行する場合、合成がサポートされるようになりました。合成をサポートするため導入された新しいカスタム ファイル フォーマットを使用すると、合成ストラテジとインプリメンテーション ストラテジを同時に指定できます。

また、新しく追加された SmartXplorer オプションを使用して、次の操作を実行できます。

- ・ **-area_report** オプションを使用して SmartXplorer レポートの表にエリア情報を表示
- ・ **-pwo** オプションを使用して消費電力解析を実行し、SmartXplorer レポートの表に消費電力情報を表示 (コマンド ライン モードのみ)
- ・ **-pwo** オプションを使用して、最高のストラテジの選択基準として電力情報を使用
- ・ **-to** オプションを使用して TRACE を制御し、詳細 TRACE レポートを生成
- ・ Virtex®-6 および Spartan®-6 ファミリーで、**-cr** オプションを使用することにより、配線の密集を緩和 (配線性を向上) する新しいストラテジを適用

SmartXplorer の概要

タイミング クロージャは、FPGA デザインにおいて最大の課題です。ザイリンクスでは、タイミング クロージャの達成をサポートするため、次のことに力を注いでいます。

- ・ 合成およびインプリメンテーション アルゴリズムの向上
- ・ PlanAhead™ および FPGA Editor などのグラフィカル解析ツールの提供

FPGA ツールは使いやすくなってきており、高度な機能を提供するようになってきていますが、すべてのデザイン状況を予測するのは困難です。デザイン サイクルの最終段階の製品を出荷する直前になってから問題が見つかる場合もあります。

SmartXplorer は、最短時間でタイミング クロージャを達成することを目標としています。

SmartXplorer の利点

SmartXplorer には、主に次の 2 つの利点があります。

- ・ あらかじめ定義されたインプリメンテーション ストラテジまたはカスタム インプリメンテーション ストラテジを複数使用して、タイミングを満たすようにデザインを実行します。

メモ: デザイン ストラテジは、エリア、スピード、消費電力など、特定のデザインの目標を達成するためのプロセス プロパティとその値の組み合わせです。

- ・ 複数のストラテジを複数のマシンで並行して実行し、ジョブを短時間で完了します。

デザイン ストラテジ

SmartXplorer では、あらかじめ定義されたストラテジが複数提供されています。これらのストラテジは、各 FPGA ファミリー用に個別に選択され、調整されています。この選択は、特定のソフトウェア バージョンで最適に機能することを確実にするため、各メジャー リリースで修正されます。

独自の経験に基づいて、カスタム ストラテジまたはスクリプトを作成することもできます。SmartXplorer ではこれらのカスタム ストラテジをシステムに統合し、それらのみを使用したり、SmartXplorer で提供されているストラテジと組み合わせで使用できます。

SmartXplorer は、プロジェクトの最終段階での緊急事態を解決するのに有益ですが、定期的に行うことでタイミングが許容範囲内にあることを確認し、最終段階で緊急事態が発生するのを回避するようにすることをお勧めします。

並列実行

複数のデザイン ストラテジ (ジョブ) を同時に実行できると、プロジェクトを短時間で完了できます。この機能は、ご使用のオペレーティング システムによって異なります。

Linux ネットワーク : ネットワーク上の複数のマシンで複数のジョブを並列実行できます。これには、次の 2 つの方法があります。

- ・ 通常の Linux ネットワークを使用している場合は、ネットワーク上でのジョブの分配は SmartXplorer で管理されます。この場合、使用するマシンのリストが必要です。
- ・ LSF (Load Sharing Facility) または SGE (Sun Grid Engine) コンピュータ ファームを使用している場合、ジョブの分配は LSF または SGE で管理されます。この場合、SmartXplorer に同時に割り当て可能なマシンの数を指定します。

1 つの Linux マシン : 1 つのマシンにマルチコア プロセッサまたは複数のプロセッサが搭載されている場合、複数のストラテジを同時に実行できます。

Microsoft Windows : 1 つのマシンにマルチコア プロセッサまたは複数のプロセッサが搭載されている場合、複数のストラテジを同時に実行できます。

1 つの Linux または Windows マシンの使用

ネットワーク上の Linux サーバーへのアクセスがなく、ローカル コンピュータしか使用できない場合は、マシンにマルチコア プロセッサがあるか複数のプロセッサがあることを確認してください。

まず、ご使用のマシンで同時に実行可能なジョブの数を予測する必要があります。

論理的には、同時に実行可能なジョブの数は次のように計算されます。

$$\text{ジョブの数} = P * C$$

P : プロセッサの数

C : プロセッサごとのコアの数

たとえば、デュアル コア プロセッサが 4 つある場合、8 個のジョブを同時に実行できます。

ただし、使用可能なメモリ、スピード、ハードドライブのスピードによっては、コンピュータで上記の式で求めた最大数のジョブを処理できない可能性があります。その場合、同時に実行するジョブの数を減らすことをお勧めします。

ヒント : 次に、並列実行に関するヒントを示します。

- ・ デザイン サイズのためマシンで一度に 1 つのストラテジしか実行できない場合は、すべてのストラテジを順次に実行する必要があります。これは、簡単に夜間に実行できます。
- ・ タイミングの問題を解決している場合、デザインに含まれるブロックを個別に実行することもできます。これらのブロックに対してであれば複数のストラテジを同時に実行できる可能性があり、時間を節約できます。

SmartXplorer のデバイス サポート

このプログラムは、次のデバイス ファミリーで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6

SmartXplorer の使用

SmartXplorer を使用すると、複数のストラテジを実行し、またそれらを並行して同時に実行することにより、タイミング クロージャを短時間で達成できます。このセクションでは、複数のストラテジの設定方法を説明します。複数のストラテジを並行して同時に実行する方法は、「[複数のストラテジの同時実行](#)」を参照してください。

12.1 リリースから、SmartXplorer で Xilinx Synthesis Technology (XST) および Synplify 合成ツールがサポートされるようになりました。複数のインプリメンテーション ストラテジを実行する前に、複数の合成ストラテジを実行してインプリメンテーションの実行に使用する最高の合成済みネットリストを選択できます。SmartXplorer で合成を実行する必要はありません。これまでと同様、インプリメンテーションのみに SmartXplorer を使用することも可能です。

メモ： SmartXplorer での合成の実行は、コマンドラインのみでサポートされ、ISE 環境では使用できません。

SmartXplorer で合成を実行する場合、SmartXplorer の実行は合成とインプリメンテーションの 2 段階プロセスになります。

- ・ **段階 1 (合成)：** 複数の合成ストラテジを実行し、パフォーマンスが最高のネットリストを判断します。合成ツールではタイミング スコアは生成されません。合成された各ネットリストに対して、ランタイムが最短になるよう最適化されたストラテジを 1 つ使用して MAP および PAR を 1 回実行し (クイック インプリメンテーション)、タイミング スコアを算出します。
- ・ **段階 2 (インプリメンテーション)：** 最高のネットリストを使用し、複数のインプリメンテーション ストラテジを実行してタイミング要件が満たされるようにします。

メモ： 合成段階またはインプリメンテーション段階でタイミング スコアが 0 になると、SmartXplorer の実行は終了します。smartxplorer コマンドで **-ra** オプションを使用すると、タイミングスコアが 0 になった場合でも、すべてのストラテジが実行されます。

SmartXplorer の実行

SmartXplorer で合成段階およびインプリメンテーション段階を実行するには、次のいずれかの入力ファイルを指定します。

- ・ XST スクリプトファイル (*design.xst*)：XST での合成に使用します。たとえば、次のように入力します。

```
smartxplorer -p xc6slx16-2-csg324 -uc stopwatch.ucf -sd ".;ipcore_dir" stopwatch.xst
```

- ・ Synplify プロジェクトファイル (*design.prj*)：Synplify での合成に使用します。たとえば、次のように入力します。

```
smartxplorer -p xc6slx16-2-csg324 -uc stopwatch.ucf -sd ".;ipcore_dir" stopwatch.prj
```

詳細は、XST および Synplify のサブセクションを参照してください。

インプリメンテーション ストラテジのみの実行

合成をスキップしてインプリメンテーション ストラテジのみを実行する場合は、SmartXplorer コマンドラインで合成済みのネットリスト (NGC または EDIF) を指定します。たとえば、次のように入力します。

```
smartxplorer -p xc6slx16-2-csg324 -uc stopwatch.ucf -sd ".;ipcore_dir" stopwatch.ngc
```

SmartXplorer で提供されているストラテジの使用

SmartXplorer では、定義済みの合成ストラテジが数個提供されています。たとえば、Spartan®-6 用には 7 つの XST ストラテジと 5 つの Synplify ストラテジがあります。デフォルト モードでは、これらのストラテジが実行され、パフォーマンスが最高のネットリストが判断されます。最高のネットリストが選択されると、SmartXplorer で提供されているインプリメンテーション ストラテジが実行され、タイミング スコアが最低のものが判断されます。-1a オプションを使用すると、SmartXplorer で提供されている定義済みのストラテジのリストを表示できます。

メモ： カスタム合成ストラテジおよびカスタム インプリメンテーション ストラテジも作成できます。詳細は、「[カスタム ストラテジ](#)」を参照してください。

例

Spartan-6 用には、7 つの XST ストラテジと 7 つのインプリメンテーション ストラテジがあります。定義済みのインプリメンテーション ストラテジの 1 つは合成実行時のクイック インプリメンテーション ストラテジとして使用されるので、デフォルト モードでは 13 個のストラテジが実行されます。定義済みのストラテジを実行するには、次のコマンドを実行します。

```
smartxplorer -p xc6slx16-2-csg324 -uc stopwatch.ucf -wd res_dir
-sd ".;ipcore_dir" stopwatch.xst
```

SmartXplorer の実行を開始すると、ステータスと最終的な結果を示す表が表示されます。この表の各行が、1 つの定義済みストラテジを表します。この表は、次の場所に表示されます。

- ・ ターミナル ウィンドウ
- ・ smartxplorer.html ファイル。このファイルは、-wd オプションを使用していない場合、SmartXplorer を起動したディレクトリにあります。Web ブラウザで開いてください。詳細は、「[SmartXplorer のレポート](#)」を参照してください。

この表は、SmartXplorer の実行中随時アップデートされます。次に、smartxplorer.html の中間状態の例を示します。

Strategy	Host	Output	Status	Timing Score	Total Run Time
XSTOptReshRedcon_MapRunTime	host_1	run1	None	None	None
XSTOptOnehot_MapRunTime	None	None	None	None	None
XSTOnehot_MapRunTime	None	None	None	None	None
XSTOptOnehotRedcon_MapRunTime	None	None	None	None	None
XSTRegbalOptOnehot_MapRunTime	None	None	None	None	None
XSTOnehotRedcon_MapRunTime	None	None	None	None	None
XSTRegbalOptOnehotReshRedcon_MapRunTime	None	None	None	None	None

合成段階では、すべての合成ストラテジとタイミング スコアを算出するために使用されるクイック インプリメンテーション ストラテジが表示されます。

メモ： 1 番左の列に示されるストラテジ名では、合成ストラテジとクイック インプリメンテーション ストラテジがアンダースコア () でつながられています。たとえば、XSTOptReshRedcon_MapRunTime は XSTOptReshRedcon という合成ストラテジとクイック インプリメンテーション ストラテジである MapRunTime を表します。

すべての合成ストラテジの実行が完了すると、タイミング スコアに基づいて最高のネットリストが選択され、そのネットリストが異なるインプリメンテーション ストラテジを使用して実行されます。HTML は次のようにアップデートされます。

Strategy	Host	Output	Status	Timing Score	Total Run Time
XSTOptReshRedcon_MapRunTime	host_1	run1	Done	11340	0h 3m 4s
XSTOptOnehot_MapRunTime	host_1	run2	Done	12893	0h 7m 54s
XSTOnehot_MapRunTime	host_1	run3	Done	12893	0h 3m 24s
XSTOptOnehotRedcon_MapRunTime	host_1	run4	Done	12893	0h 1m 29s
XSTRegbalOptOnehot_MapRunTime	host_1	run5	Done	8907	0h 1m 30s
XSTOnehotRedcon_MapRunTime	host_1	run6	Done	12893	0h 1m 19s
XSTRegbalOptOnehotReshRedcon_MapRunTime	host_1	run7	Done	8907	0h 1m 25s
XSTRegbalOptOnehotReshRedcon_MapGlobOptLogOptRegDup	host_1	run8	Mapping	None	0h 0m 28s
XSTRegbalOptOnehotReshRedcon_MapGlobOptIOReg	None	None	None	None	None
XSTRegbalOptOnehotReshRedcon_MapRegDup	None	None	None	None	None
XSTRegbalOptOnehotReshRedcon_MapExtraEffortIOReg	None	None	None	None	None
XSTRegbalOptOnehotReshRedcon_MapLogOptRegDup	None	None	None	None	None
XSTRegbalOptOnehotReshRedcon_MapExtraEffort2	None	None	None	None	None

この例では、タイミング スコアとランタイムに基づいて、XSTRegbalOptOnehotReshRedcon 合成ストラテジ (run7) が最高のネットリストとして選択されます。XSTRegbalOptOnehot もタイミング スコアは同じですが、ランタイムが少し長くなっています。このネットリストを使用して、6 つのインプリメンテーション ストラテジ (run8 ~ run13) が実行されます。すべてのストラテジの実行が完了すると、最高のストラテジが緑色にハイライトされます (run8)。

Strategy	Host	Output	Status	Timing Score	Total Run Time
XSTOptReshRedcon_MapRunTime	host_1	run1	Done	11340	0h 3m 4s
XSTOptOnehot_MapRunTime	host_1	run2	Done	12893	0h 7m 54s
XSTOnehot_MapRunTime	host_1	run3	Done	12893	0h 3m 24s
XSTOptOnehotRedcon_MapRunTime	host_1	run4	Done	12893	0h 1m 29s
XSTRegbalOptOnehot_MapRunTime	host_1	run5	Done	8907	0h 1m 30s
XSTOnehotRedcon_MapRunTime	host_1	run6	Done	12893	0h 1m 19s
XSTRegbalOptOnehotReshRedcon_MapRunTime	host_1	run7	Done	8907	0h 1m 25s
XSTRegbalOptOnehotReshRedcon_MapGlobOptLogOptRegDup	host_1	run8	Done	7756	0h 1m 9s
XSTRegbalOptOnehotReshRedcon_MapGlobOptIOReg	host_1	run9	Done	7756	0h 1m 29s
XSTRegbalOptOnehotReshRedcon_MapRegDup	host_1	run10	Done	8907	0h 1m 9s
XSTRegbalOptOnehotReshRedcon_MapExtraEffortIOReg	host_1	run11	Done	10967	0h 1m 5s
XSTRegbalOptOnehotReshRedcon_MapLogOptRegDup	host_1	run12	Done	10076	0h 1m 14s
XSTRegbalOptOnehotReshRedcon_MapExtraEffort2	host_1	run13	Done	9714	0h 1m 15s

メモ: SmartXplorer で最高のストラテジを選択するアルゴリズムについては、「[最高のストラテジの選択](#)」を参照してください。

「Run Summary」の表には、いくつかのリンクが含まれています。

- ・ [Timing Score] 列のリンクをクリックすると、そのストラテジのタイミング レポート サマリが表示されます。

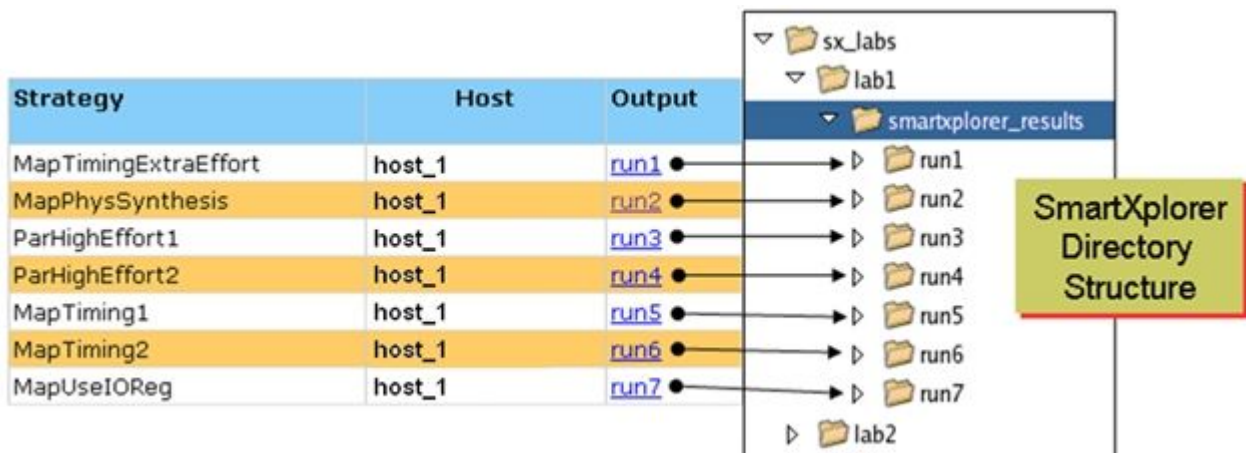
Asterisk (*) preceding a constraint indicates it was not met.
This may be due to a setup or hold violation.

Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
TS_inst_dcm1_CLK0_BUF = PERIOD TIMEGRP "Inst_dcm1_CLK0_BUF" TS_CLK HIGH 50%	SETUP	-0.155	4.355	3	191
	HOLD	1.310		0	0
TS_CLK = PERIOD TIMEGRP "CLK" 4.2 ns HIGH 50%	MINPERIOD	0.034	4.166	0	0

- ・ [Output] 列のリンクをクリックすると、そのストラテジの統合ログ ファイル (合成、MAP、PAR など) stopwatch_sx.log が表示されます。

結果ディレクトリの構造

各 SmartXplorer ストラテジの結果は、run1、run2、run3 など個別のディレクトリに保存されます。**-wd** オプションで結果ディレクトリの場所が定義されている場合以外は、これらのディレクトリはすべて SmartXplorer を起動したディレクトリに含まれます。次の例では、SmartXplorer は smartxplorer_results というディレクトリから起動されています。



実行されるストラテジの数の制御

-m integer オプションを使用すると、実行するストラテジの合計数を変更できます。

- ・ *integer* を 13 にすると、SmartXplorer がデフォルト モードで実行されます。
- ・ *integer* を 8 にすると、7 つの合成ストラテジがクイック インプリメンテーション ストラテジと共に実行され、最高のネットリストを使用して 1 つのインプリメンテーション ストラテジが実行されます。
- ・ *integer* を 3 にすると、3 つの合成ストラテジがクイック インプリメンテーション ストラテジと共に実行されます。
- ・ *integer* を 14 以上にすると、SmartXplorer で提供されている定義済みの合成ストラテジとインプリメンテーション ストラテジがすべて実行され、タイミング スコアが最低 (パフォーマンスが最高) のインプリメンテーション ストラテジと異なるコスト テーブルを使用して、追加のインプリメンテーションが実行されます。たとえば、**-m 15** を指定すると、7 つの合成ストラテジがクイック インプリメンテーション ストラテジと共に実行され、6 つのインプリメンテーション ストラテジが実行された後、最高の結果を生成したストラテジとコスト テーブル 2 および 3 を使用してインプリメンテーションが 2 回実行されます。

SmartXplorer での XST の使用

合成ツールとして XST を使用する場合は、*design.xst* スクリプトファイルを使用して SmartXplorer で合成をイネーブルにする必要があります。ISE® Project Navigator がデフォルトのデザイン環境である場合は、プロジェクト ディレクトリに *design.xst* が含まれています。このファイルは、XST を実行すると Project Navigator で自動的に生成されます。これを SmartXplorer を起動する際に入力ファイルとして使用する必要があります。ザイリンクス ツールをコマンド ライン モードで使用している場合は、コマンドライン モードで XST を実行するのに使用するのと同じ *design.xst* ファイルを使用する必要があります。

SmartXplorer で XST ストラテジを実行するには、SmartXplorer を *design.xst* を含むディレクトリから起動します。*design.xst* ファイルを変更する必要はありません。

例

```
smartxplorer -p xc6slx16-2-csg324 -uc stopwatch.ucf -wd res_dir  
-sd ".;ipcore_dir" stopwatch.xst
```

このコマンドは、SmartXplorer で提供されている合成ストラテジとインプリメンテーション ストラテジを実行します。合成には XST を使用します。

SmartXplorer での Synplify の使用

合成ツールとして Synplify を使用する場合は、Tcl スクリプトファイル *design.prj* を使用して SmartXplorer で合成をイネーブルにする必要があります。このファイルは Synplify により自動的に作成されます。Synplify の GUI を使用する場合、コマンドライン モードで Synplify を実行するのに使用するのと同じ *design.prj* ファイルを使用する必要があります。

SmartXplorer から Synplify が正しく実行されるようにするため、次を実行する必要があります。

- ・ *design.prj* 内では、ファイルおよびディレクトリへの参照をすべて絶対パスで指定します。そのため、SmartXplorer を実行する前に *design.prj* を手動で変更する必要がある場合があります。
- ・ *design.prj* に **project -run** コマンドを含めます。SmartXplorer により、このコマンドの前に合成ストラテジに対応する合成オプションが挿入されます。
- ・ SmartXplorer を *design.prj* を含むディレクトリから起動します。*design.prj* を変更する必要はありません。

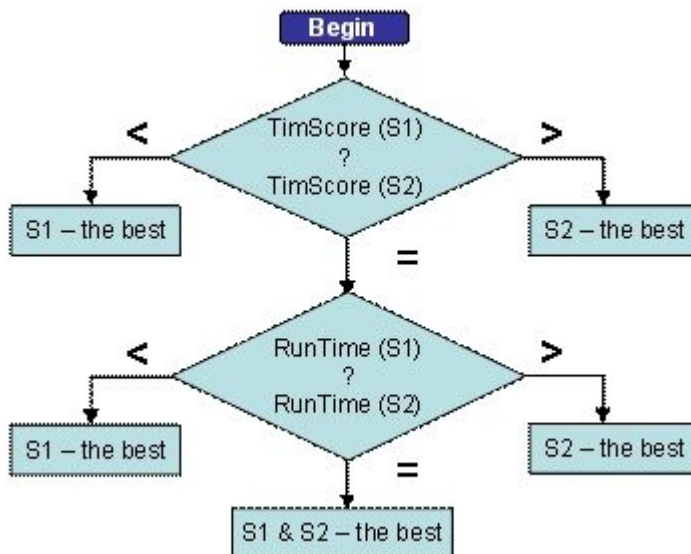
例

```
smartxplorer -p xc6slx16-2-csg324 -uc stopwatch.ucf -wd res_dir  
-sd ".;ipcore_dir" stopwatch.prj
```

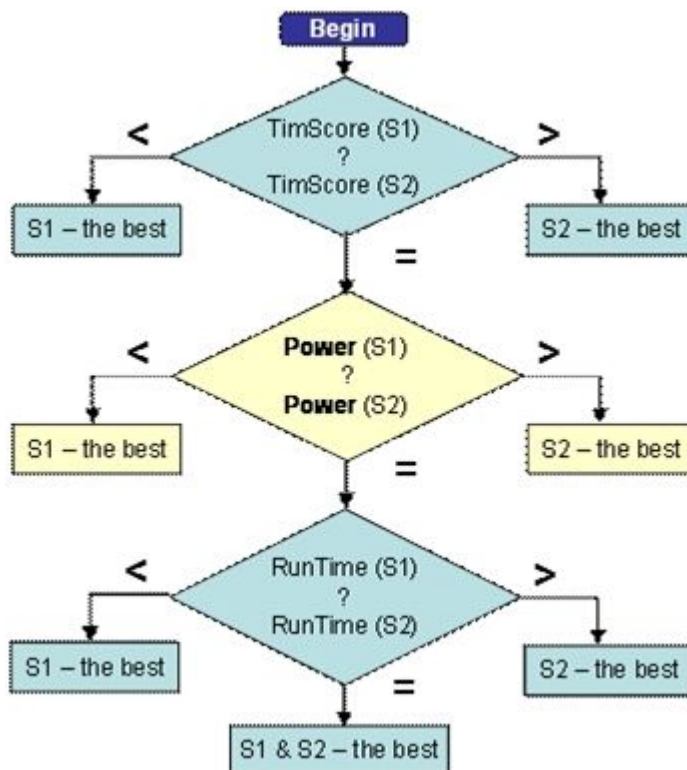
このコマンドは、SmartXplorer で提供されている合成ストラテジとインプリメンテーション ストラテジを実行します。合成には Synplify を使用します。

最高の戦略の選択

最短時間でタイミング クロージャを達成するのが SmartXplorer の最終的な目的であるので、タイミング スコアおよび合計ランタイムが最高の戦略を判断する上での主要な要素となります。次の図に、2 つの戦略 (S1 および S2) のタイミング スコアと合計ランタイムに基づいて、最高の戦略を判断するプロセスを示します。



12.1 ソフトウェアから、**-pwo** オプションを使用することにより、XPower Analyzer を実行してデザインの合計消費電力を算出できます。消費電力解析をイネーブルにした場合、最高の戦略を選択するのに電力データも使用されます。次の図に、消費電力解析をイネーブルにした場合の最高の戦略を選択するアルゴリズムを示します。



最高のストラテジを選択する重要な要素として、インプリメントされたデザインが使用するエリアもあります。現在のところ、SmartXplorer では最高のストラテジを判断するのにエリア情報は使用されませんが、「Run Summary」の表にエリア情報を表示すると便利です。

Run Summary

Strategy	Host	Output	Status	Timing Score	Luts	Slice Registers	Total RunTime
MapRunTime	host_1	run1	Done	800	51 (1%)	18 (1%)	0h 1m 29s
MapGlobOptLogOptRegDup	host_2	run2	Routing	None	54 (1%)	18 (1%)	0h 1m 35s

エリア情報のレポートの詳細は、「[-area_report \(エリア情報の表示の制御\)](#)」を参照してください。

複数のストラテジの同時実行

SmartXplorer を複数のマシンで実行する際、次の 3 つの事項を考慮する必要があります。

- ・ 各マシンでザイリンクス ソフトウェア環境がどのように設定されるか
- ・ SmartXplorer の結果の保存場所
- ・ 使用するマシンのリスト (ホストリスト ファイル) を SmartXplorer にどのように渡すか

ザイリンクス環境の設定

まず、デザイン ストラテジの実行に 3 つの Linux マシン (L1、L2、および L3) を使用する通常の Linux ネットワークの場合を考えてみます。SmartXplorer の起動には L1 を使用します。

L2 (L3) でジョブを起動する前に、SmartXplorer により自動的に L2 (L3) マシンで \$XILINX 環境変数が L1 の \$XILINX と同じ値に設定されます。これは、次のことを意味します。

- ・ ザイリンクス ソフトウェアがネットワーク上にインストールされている場合、L2 (L3) でそのソフトウェアにアクセスする必要があり、また L1 に定義されたネットワーク パスがすべてのマシンで有効となるよう同じネットワーク割り当てポイントを使用する必要があります。
- ・ ザイリンクス ソフトウェアが L1 のローカル ディスクにインストールされている場合、L2 (L3) のローカル ディスクの L1 と同じパスに同じバージョンのザイリンクス ソフトウェアがインストールされていることが必要です。

環境変数がこのように設定されるのは、各デザイン ストラテジが同じ条件下で実行されるようにするためです。\$XILINX に加え、SmartXplorer は L1 で設定されているすべてのザイリンクス環境変数 (接頭辞「XIL_」で始まるもの) を検出し、L2 および L3 に適用します。

LSF および SGE コンピュータ ファームを使用する場合でも、ザイリンクス ソフトウェアを実行する各マシンで、SmartXplorer を実行したマシンと同じザイリンクス環境変数が設定されている必要があります。

結果の保存場所

各 SmartXplorer ストラテジの結果は、run1、run2、run3 など個別のディレクトリに保存されます。これらのディレクトリはすべて同じディスク エリアに配置され、**-wd** オプションで結果ディレクトリの場所が定義されている場合以外は、SmartXplorer を起動したディレクトリに含まれます。そのため、すべてのマシンにこのディスク エリアに対する読み取り/書き込み権限が必要です。

ホスト リスト ファイル

複数のマシンで複数のストラテジを同時に実行するには、SmartXplorer で使用するマシンのリストを含むホスト リスト ファイルが必要です。

ホスト リスト ファイル名を指定するには、SmartXplorer コマンド ラインで **-l** オプションを指定します。たとえば、次のように入力します。

```
smartxplorer -p xc3s100e-4-vq100 -uc stopwatch.ucf -l
my_hostlist.txt stopwatch.ngc
```

ホスト リスト ファイルを使用する場合、次の 3 つのケースがあります。

- ・ 通常の Linux ネットワーク
- ・ LSF または SGE コンピュータ ファーム
- ・ Microsoft Windows

通常の Linux ネットワークの使用

通常の Linux ネットワークを使用する場合、次の例に示すようにホストのリストを記述します。各マシン名を個別の行に記述する必要があります。

```
host_1
host_2
host_3
host_3
```

この例では、3 つのマシン host_1、host_2、host_3 を使用します。host_3 は 2 回指定されているので、このホストでは 2 つのストラテジが実行されます。

SmartXplorer を実行すると、smartxplorer.html の「Run Summary」に各ストラテジを実行しているホスト名が表示されます。

Run Summary

Strategy	Host	Output	Status	Timing Score	Luts	Slice Registers	Total RunTime
MapRunTime	host_1	run1	Done	800	51 (1%)	18 (1%)	0h 1m 29s
MapGlobOptLogOptRegDup	host_2	run2	Routing	None	54 (1%)	18 (1%)	0h 1m 35s

コンピュータ ファームの使用

LSF または SGE コンピュータ ファームを使用する場合にもホストリスト ファイルは必要ですが、必要な情報は異なります。次に、LSF と SGE でサポートされるフォーマットを示します。

LSF	:LSF { "queue_name": "MYQUEUE", "max_concurrent_runs":N, "bsub_options": "additional_options " }
SGE	:SGE { "queue_name": "MYQUEUE", "max_concurrent_runs":N, "qsub_options": "additional_options " }

queue_name : キューの名前を指定します。「MYQUEUE」を LSF または SGE キューの名前に置き換えてください。

max_concurrent_runs : 並行して実行可能なジョブの最大数を指定します。N に正の整数を指定してください。

bsub_options : 追加の LSF オプションを指定します。「additional_options」を LSF オプションに置き換えてください。オプションを使用しない場合は、"" (ダブル クォーテーション 2 つ) と記述します。

qsub_options : 追加の SGE オプションを指定します。「additional_options」を SGE オプションに置き換えてください。オプションを使用しない場合は、"" (ダブル クォーテーション 2 つ) と記述します。

例

キューの名前が lin64_q、並行して実行可能なジョブの最大数が 6、指定する LSF または SGE オプションがない場合は、ホストリスト ファイルに次のように記述します。

LSF	:LSF { "queue_name": "lin64_q", "max_concurrent_runs":6, "bsub_options": "" }
SGE	:SGE { "queue_name": "lin64_q", "max_concurrent_runs":6, "qsub_options": "" }

Microsoft Windows

Microsoft Windows では、1 つのマシンにマルチコア プロセッサまたは複数のプロセッサが搭載されている場合、複数のストラテジを同時に実行できます。同じマシンで複数のストラテジを実行するには、ホストリスト ファイルにそのマシンの名前を複数回記述する必要があります。

```
host_1
host_1
host_1
```

この例では、host_1 で 3 つのストラテジが同時に実行されます。

カスタム ストラテジ

SmartXplorer では、カスタム ストラテジ ファイルを作成し、合成、MAP、および PAR のオプションを任意に組み合わせたストラテジを作成できます。ストラテジ ファイルは、**-sf** オプションを使用して指定します。

メモ： カスタム ストラテジ ファイルのフォーマットは、合成のサポートの追加に伴い変更されています。以前のストラテジ ファイルのフォーマットもサポートされており、SmartXplorer をインプリメンテーション フローのみで実行する場合に使用できますが、できるだけ早く新しいストラテジ ファイルのフォーマットに移行してください。このセクションでは、両方のフォーマットを説明します。

カスタム ストラテジ ファイルの新しいフォーマット

カスタム ストラテジ ファイルの新しいフォーマットでは、合成およびインプリメンテーション用のカスタム ストラテジを定義できます。

メモ： 間違いを避けるため、ストラテジ ファイルの最後の波かっこ (「」) を除く閉じかっこ (「」) および波かっこの後に、カンマ (「,」) を使用してください。

カスタム ストラテジの例 (XST)

次に、XST で使用するカスタム ストラテジ ファイルの例を示します。

```
# This is a custom Strategy file for XST
{
  "spartan6":
  {
    "XST options":
    (
      { "name": "my_xst1",
        "xst": "-opt_level 1 -fsm_extract yes" },
      { "name": "my_xst2",
        "xst": "-opt_level 2 -fsm_extract no" },
    ),
    "Map-Par options":
    (
      { "name": "my_impl1",
        "map": " -timing -ol high -xe n -global_opt on -retiming on ",
        "par": " -ol high" },
      { "name": "my_impl2",
        "map": " -timing -ol high -xe n ",
        "par": " -ol high" },
    ),
  },
}
```

このストラテジ ファイル例には、2 つの合成ストラテジ (my_xst1 および my_xst2) と 2 つのインプリメンテーション ストラテジ (my_impl1 および my_impl2) が含まれます。どちらのストラテジも、Spartan®-6 ファミリのデバイスをターゲットとするデザインでのみ実行されます。この例をストラテジ ファイルのテンプレートとして使用できます。

カスタム ストラテジの例 (Synplify)

次に、Synplify で使用するカスタム ストラテジ ファイルの例を示します。

```
# This is a custom Strategy file for Synplify
{
  "spartan6":
  {
    "Synplify options":
    (
      {"name": "my_smpl1",
       "synplify": " set_option -symbolic_fsm_compiler true set_option -default_enum_encoding onehot"},
      {"name": "my_smpl2",
       "synplify": " set_option -symbolic_fsm_compiler false "},
    ),
    "Map-Par options":
    (
      {"name": "my_impl1",
       "map": " -timing -ol high -xe n -global_opt on -retiming on ",
       "par": " -ol high"},
      {"name": "my_impl2",
       "map": " -timing -ol high -xe n ",
       "par": " -ol high"},
    ),
  },
}
```

このストラテジ ファイル例には、2 つの合成ストラテジ (my_smpl1 および my_smpl2) と 2 つのインプリメンテーション ストラテジ (my_impl1 および my_impl2) が含まれます。どちらのストラテジも、Spartan-6 ファミリのデバイスをターゲットとするデザインでのみ実行されます。この例をストラテジ ファイルのテンプレートとして使用できます。

カスタム ストラテジ ファイルの以前のフォーマット

カスタム ストラテジ ファイルの以前のフォーマットでは、インプリメンテーション用のカスタム ストラテジを定義できます。合成用のストラテジはサポートしていません。MAP および PAR のオプションを任意に組み合わせたストラテジをいくつでも入力できます。次に、以前のフォーマットを使用した単純なストラテジ ファイルの例を示します。

```
{
  "virtex4":
  (
    {"name": "strategy1",
     "map": " -timing -ol high -xe n -global_opt on -retiming on ",
     "par": " -ol high "},
    {"name": "strategy2",
     "map": " -timing -ol high -xe n ",
     "par": " -ol high "},
  ),
}
```

上記のストラテジ ファイルの例には、strategy1 (4 行目) と strategy2 (7 行目) という 2 つのストラテジが含まれています。これらのストラテジは、Virtex®-4 ファミリのデバイスをターゲットとするデザインでのみ実行されます。この例をストラテジ ファイルのテンプレートとして使用できます。

SmartXplorer の構文

このセクションには、次の内容が含まれています。

- ・ [コマンド ライン構文](#)
- ・ [SmartXplorer のファイルとディレクトリ](#)
- ・ [SmartXplorer のオプション](#)

コマンド ライン構文

SmartXplorer のコマンド ライン構文を次に示します。

```
smartxplorer -p PartNumber [-l HostListFile] [options]  
DesignName [.edf|.ngd|.ngc|.xst|.prj]
```

- ・ **-p**: デザインをインプリメントするデバイスを指定します (必須)。PartName には、完全なザイリンクス デバイスの製品番号を指定する必要があります。詳細は、「[-p \(製品番号\)](#)」を参照してください。
- ・ **options**: 「[SmartXplorer のオプション](#)」にリストされている SmartXplorer オプションを複数指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。
- ・ **DesignName [.edf|.ngd|.ngc|.xst|.prj]** は、インプリメントするデザインを含むデザイン ファイルです。入力ファイルのタイプにより、SmartXplorer で使用するフローが判断されます。
 - EDF、NGD、および NGC ファイルの場合、インプリメンテーション ストラテジのみを実行します。
 - XST ファイル (XST スクリプト) の場合、XST 合成ストラテジを実行し、最高のネットリストを選択してインプリメンテーション ストラテジを実行します。
 - PRJ ファイル (Synplify の Tcl ベースのプロジェクト ファイル) の場合、Synplify 合成ストラテジを実行し、最高のネットリストを選択してインプリメンテーション ストラテジを実行します。

SmartXplorer のファイルとディレクトリ

次に、SmartXplorer で使用されるファイルとディレクトリを示します。

SmartXplorer の入力ファイル

ファイル名	説明
<i>DesignName</i> [.edf .ngd .ngc .xst .prj]	デザイン ファイル。SmartXplorer の実行で使用するフローを指定します。詳細は、「 コマンド ライン 構文 」を参照してください。
<i>filename</i> .ucf	タイミング制約、物理配置制約、およびその他の制約を含むユーザー制約ファイル (UCF)。詳細は、「 -uc (UCF ファイル) 」を参照してください。制約の詳細は、『 制約ガイド 』を参照してください。
ホストリスト ファイル (デフォルト : smartxplorer.hostlist)	マスタ マシンからジョブを割り当てることのできるホストのリストを含むファイル。詳細は、「 -l (ホスト リスト ファイル) 」を参照してください。
カスタム ストラテジ ファイル	ビルトインのストラテジの代わりに使用するユーザー定義のストラテジを含むファイル。詳細は、「 -sf (ストラテジ ファイル) 」を参照してください。
smartxplorer.config	-n オプションを使用している場合に、メール サーバーを設定するのに必要なファイル。詳細は、「 -n (通知) 」を参照してください。

SmartXplorer の出力ディレクトリ

出力ディレクトリ名	説明
/run[i]	SmartXplorer では、ストラテジの数だけ run[i] ディレクトリが作成されます (i は n-1、n は実行されたストラテジの数)。これらの各ディレクトリには、ストラテジで設定された合成、MAP、PAR、TRACE で生成されたすべてのレポートおよびファイルが含まれます。

SmartXplorer の出力ファイル

出力ファイル名	説明
smartxplorer.html	HTML 形式の SmartXplorer レポート。SmartXplorer の実行中、動的にアップデートされます。詳細は、「 SmartXplorer のレポート 」を参照してください。
smartxplorer.txt	各ストラテジの実行に関する詳細を示し、最後に最高の結果が得られたストラテジを示します。詳細は、「 SmartXplorer のレポート 」を参照してください。
<i>DesignName</i> _sx.log	フローの異なる段階 (合成、MAP、PAR など) からの標準出力を含むログ ファイル。実行ストラテジごとに作成され、run[i] ディレクトリに保存されます。詳細は、「 SmartXplorer のレポート 」を参照してください。

SmartXplorer のオプション

このセクションでは、SmartXplorer のコマンド ライン オプションについて説明します。

- ・ **-area_report** (エリア情報の表示の制御)
- ・ **-b** (バッチ モード)
- ・ **-best** (最高の N 個の結果を保存)
- ・ **-bo** (BitGen オプション)
- ・ **-cr** (配線密集の緩和)
- ・ **-l** (ホスト リスト ファイル)
- ・ **-la** (すべてのストラテジをリスト)

- ・ -m (実行回数)
- ・ -mo (MAP オプション)
- ・ -n (通知)
- ・ -p (製品番号)
- ・ -po (PAR オプション)
- ・ -pwo (消費電力オプション)
- ・ -ra (すべてのストラテジを実行)
- ・ -rcmd (リモート コマンド)
- ・ -sd (ソース ディレクトリの指定)
- ・ -sf (ストラテジ ファイル)
- ・ -to (TRACE オプション)
- ・ -uc (UCF ファイル)
- ・ -vp (異なるコスト テーブルを使用して実行)
- ・ -wd (出力結果の保存ディレクトリ)

-area_report (エリア情報の表示の制御)

SmartXplorer レポートでのエリア データの表示を制御します。

デフォルトでは、SmartXplorer レポートに LUT およびスライス レジスタの数が表示されます。

Strategy	Host	Output	Status	Timing Score	Luts	Slice Registers	Total RunTime
MapRunTime	host_1	run1	Done	800	51 (1%)	18 (1%)	0h 1m 4s

各 FPGA リソースに対して、2 つの値が表示されます。

- 最初の値は、使用される FPGA リソースの数を示します。上図の例では、51 個の LUT が使用されます。
- 2 番目の値は、ターゲット FPGA の使用率 (パーセント) を示します。上図の例では、使用可能な LUT のうち 1% のみが使用されます。

構文

-area_report [on|off|column_spec]

on : エリア情報の表示をオンにし、レポートに LUT およびスライス レジスタの数を表示します (デフォルト)。

off : エリア情報を表示しません。

column_spec : 表に表示する列を変更します。複数の値を同時に指定するには、値全体をダブル クォーテーションで囲み、各値をセミコロン (;) で区切ります。表には、値を入力した順序で対応する列が表示されます。

オプション	列名
lut	LUT
slice_reg	スライス レジスタ
スライス	スライス
bram	BRAM
dsp48	DSP48
mult18x18	MULT18X18

メモ : BRAM に対しては、BRAM の数のみが表示されます。

例

-area_report "slice;lut;dsp48"

Strategy	Host	Output	Status	Timing Score	Slices	Luts	DSP48s	Total RunTime
MapRunTime	host_1	run1	Done	800	21 (1%)	51 (1%)	0 (0%)	0h 1m 5s

-b (バッチ モード)

SmartXplorer をバッチ モードで実行します。

デフォルトでは、標準出力がリアルタイムでアップデートされ、出力をファイルに記述したり、SmartXplorer をバックグラウンドで実行することはできません。**-batch_mode** オプションを使用すると、画面の出力をファイルに記述したり、SmartXplorer をバックグラウンドで実行できるようになります。

構文

-b

-batch_mode

-best (最高の N 個の結果を保存)

SmartXplorer 実行のうち最高のものから N 個の結果を保存します。その他の結果は、レポートとログ ファイル以外はすべて削除されます。

構文

-best

-best_n_runs *results*

results 保存する結果の数を指定します。

-bo (BitGen オプション)

BitGen オプションを変更します。デフォルトでは、SmartXplorer の最高の結果を使用して、BitGen でプログラム データ ファイルが生成されます。ファイルを指定すると、BitGen でそのファイルに含まれるオプションが使用されます。BitGen を実行しないようにするには、**-bo off** を使用します。

構文

-bo

-bitgen_options *options_file* | **off**

options_file BitGen で使用するオプションを含むファイルを指定します。

off BitGen を実行しないように指定します。

-cr (配線密集の緩和)

Virtex®-6 および Spartan®-6 ファミリで、配線の密集を緩和 (配線性を向上) するインプリメンテーション (MAP および PAR) ストラテジを実行します。これらの専用ストラテジが Virtex-6 および Spartan-6 用に提供されている標準ストラテジの代わりに使用されます。

構文

-cr

-congestion_reduction

メモ : 配線の密集を緩和するストラテジをリストするには、**-cr** オプションと **-la** オプションを一緒に使用します。

-l (ホスト リスト ファイル)

ホストリスト ファイルを指定します。ホスト リスト ファイルには、SmartXplorer ストラテジを実行する際に使用するマシンの名前をリストします。

構文

-l *host_list_file*

-host_list *host_list_file*

デフォルトでは、SmartXplorer は起動ディレクトリから `smartxplorer.hostlist` という名前のファイルを検索します。このファイルを使用する場合、**-l** オプションを使用する必要はありません。ホストリストの詳細は、「[ホストリスト ファイル](#)」を参照してください。

-la (すべてのストラテジをリスト)

指定のデバイス ファミリ用にあらかじめ定義されているストラテジをリストします。このオプションを使用した場合はストラテジがリストされるだけで、ストラテジは実行されません。

構文

-la

-list_all_strategies

メモ : このオプションを使用する場合は、**-part** オプションでデバイス名を指定する必要があります。

例 (XST ストラテジをリスト)

XST 合成ストラテジおよびインプリメンテーション ストラテジをリストするには、XST スクリプト ファイル (*DesignName .xst*) を入力ファイルとして指定します。

```
smartxplorer -p xc6slx16-3-csg324 -la stopwatch.xst
```

例 (Synplify ストラテジをリスト)

Synplify 合成ストラテジおよびインプリメンテーション ストラテジをリストするには、Synplify プロジェクト ファイル (*DesignName .prj*) を入力ファイルとして指定します。

```
smartxplorer -p xc6slx16-3-csg324 -la stopwatch.prj
```

例 (インプリメンテーション ストラテジをリスト)

インプリメンテーション ストラテジのみをリストするには、次のコマンドを使用します。

```
smartxplorer -p xc6slx16-3-csg324 -la stopwatch.ngc
```

または

```
smartxplorer -p xc6slx16-3-csg324 -la
```

次に、このオプションを使用した場合に表示されるインプリメンテーション ストラテジのリストを示します。一部の行は削除されています。

```
smartxplorer -p xc6slx16-3-csg324 -la

===== List Of Strategies for Part xc6slx16-3-csg324 =====

Strategy MapRunTime:
-----
map options:  -ol high -w
par options:  -ol high

Strategy MapGlobOptIOReg:
-----
map options:  -ol high -global_opt speed -pr b -w
par options:  -ol high -xe n
```

-m (実行回数)

SmartXplorer で実行するストラテジの数を指定します。詳細は、「[SmartXplorer で提供されているストラテジの使用](#)」を参照してください。

構文

```
-m number_of_runs
```

```
-max_runs number_of_runs
```

-m を使用しない場合、SmartXplorer で提供されているストラテジまたはカスタム ストラテジすべてが実行されます。ただし、タイミング スコアが 0 になった場合は、SmartXplorer の実行はそこで停止します。

メモ : **-vp** オプションと **-m** オプションを同時に使用することはできません。

-mo (MAP オプション)

MAP オプションを変更します。

構文

-mo *options*

-map_options *options*

options : 「MAP」の章にリストされている MAP オプションを指定します。これらのオプションは、ダブル クォーテーションで囲む必要があります。

-mo を使用して指定したオプションはすべてのストラテジに適用され、SmartXplorer で提供されているストラテジまたはカスタム ストラテジで指定されている MAP オプションよりも優先されます。

レポートでは、このオプションにより MAP オプションが変更されているストラテジの横にアスタリスク (*) が表示されます。

Run Summary

Strategy	Host	Output	Status	Timing Score	Luts	Slice Registers	Total RunTime
MapRunTime*	host_1	run1	Routing	None	51 (1%)	18 (1%)	0h 0m 44s

メモ : **-mo** オプションと **-po** オプションを両方使用すると、すべてのストラテジファイルが無視され、**-mo** オプションおよび **-po** オプションで指定されたオプションのみが使用されます。

例

-mo "-ol high -w"

この例では、SmartXplorer の実行中にすべての MAP オプションがダブル クォーテーションで囲まれたオプションに変更されます。

-n (通知)

すべてのジョブが完了した後に電子メールまたは携帯電話にテキスト メッセージを送付するよう指定します。このメッセージには、次の情報が含まれます。

- ・ 達成された最低のタイミング スコア
- ・ smartxplorer.txt レポート ファイル (詳細は「[SmartXplorer のレポート](#)」を参照)

メモ : 携帯電話へのテキスト メッセージの送信は、携帯電話にテキスト メッセージの機能がある場合にのみ使用可能で、北米でのみサポートされます。

構文

-n "useraddr[;useraddr[;...]]"

-notify "useraddr[;useraddr[;...]]"

電子メールのアドレスまたは携帯電話番号はクォーテーションで囲んで指定し、複数のディレクトリはセミコロン (;) で区切ります。SmartXplorer の実行が終了すると、ここで指定した電子メール アドレスおよび携帯電話に通知が送付されます。

例

```
-notify='user1@myCompany.com,user2@myCompany.com,8005551234'
```

メール サーバーの設定

デフォルトでは、ローカル ホストをメール サーバー (SMTP) として使用して電子メールが送信されます。この設定は、次に示すように smartxplorer.config ファイルで XIL_SX_MAIL_SERVER 環境変数を設定することにより変更できます。

```
# Sample "smartxplorer.config" File for Mail Server Configuration
# -----
{
  "XIL_SX_MAIL_SERVER":    "E-mail_Server.my_comp.com",
}
# End    of file
```

メモ : XIL_SX_USE_CONFIG_FILE 環境変数が 1 に設定されてない場合、SmartXplorer で smartxplorer.config ファイルは読み込まれません。

-p (製品番号)

デザインをインプリメントするパーツを指定します。

構文

```
-p part number
```

part number デバイス、パッケージ、スピードを含む完全な製品番号を指定する必要があります (例 : xc4vlx60-10- ff256)。

メモ : 構文の詳細および例は、「はじめに」の章の「**-p (製品番号)**」を参照してください。

-po (PAR オプション)

PAR オプションを変更します。

構文

```
-po options
```

```
-par_options options
```

options : 「**PAR**」の章にリストされている PAR オプションを指定します。これらのオプションは、ダブル クォーテーションで囲む必要があります。

-po を使用して指定したオプションはすべてのストラテジに適用され、SmartXplorer で提供されているストラテジまたはカスタム ストラテジで指定されている PAR オプションよりも優先されます。

レポートでは、このオプションにより MAP オプションが変更されているストラテジの横にアスタリスク (*) が表示されます。

Run Summary

Strategy	Host	Output	Status	Timing Score	Luts	Slice Registers	Total RunTime
MapRunTime*	host_1	run1	Routing	None	51 (1%)	18 (1%)	0h 0m 44s

メモ: **-mo** オプションと **-po** オプションを両方使用すると、すべてのストラテジファイルが無視され、**-mo** オプションおよび **-po** オプションで指定されたオプションのみが使用されます。

例

-po "-ol high -xe n"

この例では、SmartXplorer の実行中にすべての PAR オプションがダブル クォーテーションで囲まれたオプションに変更されます。

-pwo (消費電力オプション)

配置配線済みのデザインで XPower Analyzer を実行し、`smartxplorer.html` および `smartxplorer.txt` レポートファイルにデザインの合計消費電力を表示します。

-pwo オプションを使用すると、次の図に示すように [Power (mW)] 列に電力情報が表示されます。

Run Summary

Strategy	Host	Output	Status	Timing Score	Luts	Slice Registers	Power (mW)	Total RunTime
MapRunTime	host_1	run1	Done	800	51 (1%)	18 (1%)	49.06	0h 1m 15s

構文

-pwo [off|on|options]

-power_options [off|on|options]

off : XPower Analyzer を実行しません (デフォルト)。

on : XPower Analyzer をデフォルト オプションで実行します。

options : XPower Analyzer を指定したオプションで実行します。「XPower」の章の「[XPower のオプション](#)」にリストされているオプションを指定できます。オプションは、ダブル クォーテーションで囲む必要があります。

例

-pwo "-v"

この例は、配置配線済みのデザインに対して XPower Analyzer を **-v** オプションを使用して実行し、デザインの合計消費電力を表示します。

-ra (すべてのストラテジを実行)

あらかじめ定義されたストラテジおよびユーザー定義ストラテジすべてを実行します。

デフォルトでは、タイミングを満たすストラテジが見つかり (タイミング スコア 0)、最高の結果が保存され、SmartXplorer が終了します。タイミングを満たすストラテジが見つかってもすべてのストラテジを完了するまで実行を続ける場合は、このオプションを使用します。

構文

```
-ra
-run_all_strategies
```

-rcmd (リモート コマンド)

リモート ホストにログインし、ホストでコマンドを実行するのに使用するプログラムを指定します。

構文

```
-rcmd [rsh|ssh]
-remote_command [rsh|ssh]
```

有効な値は **rsh** または **ssh** で、デフォルト値は **rsh** です。

-sd (ソース ディレクトリの指定)

デザイン ファイルを検索する追加のパスを指定します。このオプションは、デザイン ディレクトリ以外に CORE Generator™ など生成された中間ネットリスト ファイルがある場合に使用すると便利です。

構文

```
-sd "source_dir_path:[source_dir_path];..."
-source_dir "source_dir_path:[source_dir_path];..."
```

ディレクトリはダブル クォーテーションで囲んで指定し、複数のディレクトリはセミコロン (;) で区切ります。デフォルト値は、SmartXplorer を起動したディレクトリです。

例

```
-source_dir
"path_to_directory1;path_to_directory2;path_to_directory3"
```

この例では、デザイン ディレクトリを検索する前に path_to_directory1、path_to_directory2、および path_to_directory3 を検索するよう指定します。

-sf (ストラテジ ファイル)

SmartXplorer であらかじめ定義されたストラテジの代わりに使用するカスタム ストラテジ ファイルを指定します。詳細は、「[カスタム ストラテジ](#)」を参照してください。

構文

```
-sf strategy_file
-strategy_file strategy_file
```

-to (TRACE オプション)

デフォルトでは、SmartXplorer から実行される TRACE では短縮されたタイミング レポートが生成されます。このオプションを使用すると、TRACE の動作を変更し、詳細レポートを生成するなど、ニーズに合った TRACE レポートを生成できます。

構文

-to options

options: 「TRACE」の章の「[TRACE のオプション](#)」にリストされているオプションを指定できます。オプションは、ダブル クォーテーションで囲む必要があります。

例

-to "-v 10"

この例では、TRACE を **-v** オプションを 10 に設定して実行します。

-uc (UCF ファイル)

デザインの UCF ファイルを指定します。タイミング制約、物理配置制約、およびその他の制約が含まれます。制約の詳細は、[『制約ガイド』](#)を参照してください。

構文

-uc ucf_file

-ucf ucf_file

-uc を使用しない場合、*design_name.ucf* が検索されます。指定された UCF ファイルが見つからない場合は、**-sd** オプションで指定されたディレクトリが検索されます。同じ名前のファイルが複数ある場合は、最初に見つかった UCF ファイルが使用されます。

メモ: NGD ファイルを入力デザイン ファイルとして指定した場合は、NGD ファイルにタイミング、配置、およびその他の属性が組み込まれているので、UCF ファイルを使用する必要はありません。

SmartXplorer では、複数の UCF ファイルがサポートされます。複数のファイルを指定するには、ファイルのリストをダブル クォーテーションで囲み、各ファイルをセミコロン (;) で区切ります。

例

-uc "file1.ucf;file2.ucf"

この例では、**-sd** オプションで指定したディレクトリ内で *file1.ucf* および *file2.ucf* が検索されます。

-vp (異なるコスト テーブルを使用して実行)

SmartXplorer で特定のストラテジ (最高のストラテジ) を異なるコスト テーブルを使用して実行し、タイミングをさらに向上させます。ベース ストラテジ (MAP および PAR オプション) は、次のように指定できます。

- ・ SmartXplorer コマンドラインで **-mo** および **-po** を使用して直接指定
- ・ カスタム ストラテジ ファイルで指定

このオプションは、このモードで使用するコスト テーブルの数を指定します。詳細は、[「SmartXplorer の使用」](#)を参照してください。

構文

-vp *number_of_cost_tables*

-variability_passes *number_of_cost_tables*

メモ : **-vp** オプションと **-m** オプションを同時に使用することはできません。

-wd (出力結果の保存ディレクトリ)

出力結果を保存する場所を指定します。このオプションを指定しない場合 (デフォルト)、結果は SmartXplorer を起動したディレクトリに保存されます。

構文

-wd *write_dir_path*

-write_dir *write_dir_path*

SmartXplorer のレポート

SmartXplorer では、次の 3 つのレポートが生成されます。

- ・ *smartxplorer.html* (HTML)
- ・ *smartxplorer.txt* (テキスト)
- ・ *DesignFile_sx.log* (テキスト)

smartxplorer.html

smartxplorer.html は HTML 形式の SmartXplorer レポートで、次の場所に保存されます。

- ・ SmartXplorer を実行したディレクトリ (**-wd** オプションが指定されていない場合)
- ・ **-wd** オプションで指定されたディレクトリ

このレポートは複数の部分から構成されており、SmartXplorer の実行中随時アップデートされます。

レポートの最初にある著作権情報の下に、その SmartXplorer 実行で使用されたコマンドが示されます。

```
smartxplorer -p xc6slx16-3-csg324 -uc stopwatch.ucf -wd res_dir -l my_hostlist.txt -sd ".;ipcore_dir" stopwatch.ngc
```

コマンドの下に実行サマリを示す「Run Summary」の表があります。

Run Summary

Strategy	Host	Output	Status	Timing Score	Luts	Slice Registers	Total RunTime
MapRunTime	host_1	run1	Done	800	51 (1%)	18 (1%)	0h 1m 29s
MapGlobOptLogOptRegDup	host_2	run2	Routing	None	54 (1%)	18 (1%)	0h 1m 35s
MapGlobOptIOReg	None	None	None	None	None	None	None
MapRegDup	None	None	None	None	None	None	None
MapExtraEffortIOReg	None	None	None	None	None	None	None
MapLogOptRegDup	None	None	None	None	None	None	None
MapExtraEffort2	None	None	None	None	None	None	None

この表には、ストラテジの実行の進行状況と最終的な結果のサマリが含まれ、実行中随時アップデートされます。各行は 1 つのストラテジを示し、デフォルトで次の情報が含まれます。

- ・ **Strategy (ストラテジ)** : ストラテジ名。ストラテジ名にカーソルを置くと、使用された合成オプション、MAP オプション、および PAR オプションが表示されます。
- ・ **Host (ホスト)** : ストラテジが実行されたホスト マシン。ホスト名にカーソルを置くと、OS のタイプ、プロセッサの数、メモリ サイズが表示されます。
- ・ **Output (出力)** : ログ ファイル (*DesignFile_sx.log*) へのリンクを示します。ログ ファイルには、フローの異なる段階 (合成、MAP、PAR) からの標準出力が含まれます。
- ・ **Status (ステータス)** : 現在実行されているフローの段階を示します。
- ・ **Timing Score (タイミング スコア)** : ストラテジのタイミング スコア。タイミング スコアが 0 の場合、すべてのタイミング制約が満たされています。タイミング スコア 0 に付いている下線は、タイミング レポート ファイル (*DesignFile.twx.html*) へのリンクを示します。
- ・ **Luts (LUT)、Slice Registers (スライス レジスタ)** : ストラテジのエリア情報を示します。使用されるリソースの数 (LUT の数など) とターゲット FPGA の使用率 (パーセント) を示します。

メモ : エリア情報は MAP レポートから抽出されたものです。**-area_report** オプションを使用すると、追加のエリア情報を表示できます。

- ・ **Power (mW) (消費電力)** : デザインの合計消費電力を示します。デフォルトでは表示されません。消費電力の情報を表示するには、**-pwo** オプションを使用して Power Analyzer を実行する必要があります。
- ・ **Total Run Time (合計ランタイム)** : フロー全体の合計ランタイム。

上図の例では、最初のストラテジ (MapRunTime) が完了しています。このストラテジの実行には 1 分 29 秒かかり、タイミング スコアは 800 で、タイミング制約が満たされていないことを示しています。この行は緑色で表示されており、現時点ではこのストラテジで最高のタイミングが得られていることを示します。MapGlobOptLogOptRegDup ストラテジは現在実行中で、配線段階です。その他のストラテジのステータスは「None」であり、実行が開始されていないことを示します。

「Run Summary」の表の下には「Best Strategy」の表があり、最高の結果を生成したコマンドが表示されます。

Best Strategy: MapRunTime (run1), Timing Score: 800, Runtime: 0h 1m 29s

Command Lines

```
map stopwatch.ngd -ol high -w -p xc6slx16-3-csg324 -o stopwatch_map.ncd /home/test/temp/546235/res_dir/run1/stopwatch.pcf

par stopwatch_map.ncd -ol high -w stopwatch.ncd /home/test/temp/546235/res_dir/run1/stopwatch.pcf

trce /home/test/temp/546235/res_dir/run1/stopwatch.ncd /home/test/temp/546235/res_dir/run1/stopwatch.pcf -xml /home/test/temp/546235/res_dir/run1/stopwatch.twx -o /home/test/temp/546235/res_dir/run1/stopwatch.twr
```

「Best Strategy」の表の下には「Environment Variables」の表があり、プラットフォーム特定の環境変数およびザイリンクス特定の環境変数を示します。

Environment Variables

Name	Value
PATH	/xilinx/12.1/bin/lin64:/usr/bin:/bin:/usr/X11R6/bin:/usr/local/bin:./home/test/bin:/usr/sbin:/products/valgrind-1.9.6/bin
LD_LIBRARY_PATH	/xilinx/12.1/bin/lin64:/xilinx/12.1/lib/lin64:/usr/lib
XILINX	/xilinx/12.1
LM_LICENSE_FILE	1111@server

smartxplorer.txt

smartxplorer.txt はテキスト形式の SmartXplorer レポートで、次の場所に保存されます。

- ・ SmartXplorer を実行したディレクトリ (**-wd** オプションが指定されていない場合)
- ・ **-wd** オプションで指定されたディレクトリ

smartxplorer.txt ファイルには、各ストラテジの実行に関する詳細が含まれ、最後に最高の結果が得られたストラテジが示されます。次に、一般的な smartxplorer.rpt ファイルの例を示します。ただし、一部の行は削除されています。

```
-----
Strategy : MapRunTime
-----
```

```
Run index                : run1
Map options               : -ol high -w
Par options               : -ol high
Number of Luts            : 51 (1%)
Number of Slice Registers : 18 (1%)
Status                   : Done
Achieved Timing Score     : 900
Current Best (Lowest) Timing Score : 900
Current Best Strategy    : MapGlobOptLogOptRegDup
-----
```

```
#####
BestStrategy : MapLogOptRegDup
#####
```

```
Run index                : run6
Map options               : -ol high -xe n -logic_opt on -t 2 -w
Par options               : -ol high -xe n
Number of Luts            : 51 (1%)
Number of Slice Registers : 18 (1%)
Achieved Timing Score     : 800
#####
```

```
Total Real Time:482.5(secs)
SmartXplorer Done
```

DesignFile_sx.log

DesignFile_sx.log は、フローの異なる段階 (合成、MAP、PAR など) からの標準出力を含むログ ファイルです。実行ストラテジごとに作成され、run[i] ディレクトリに保存されます。

SSH で SmartXplorer を実行するための設定

SmartXplorer では、異なるホスト マシンにジョブを割り当てるのに、2 つのプロトコル RSH および SSH を使用できます。デフォルトのプロトコルは RSH です。**-rcmd** オプションを使用して SSH を指定できます。SSH を使用する場合、パスワードが不要となるように、SSH を設定する必要があります。SSH でパスワードが必要な場合、SmartXplorer は実行できません。SSH をパスワードが不要となるように設定するには、次の Linux コマンドを使用します。

SmartXplorer を SSH で実行できるよう設定するには

1. 既に SSH を設定している場合は、バックアップを作成します。

```
$ cp -r $HOME/.ssh $HOME/.ssh.bak
```

2. 次のコマンドを実行して、パブリック キーおよびプライベート キーを生成します。

```
$ mkdir -p $HOME/.ssh
$ chmod 0700 $HOME/.ssh
$ ssh-keygen -t dsa -f $HOME/.ssh/id_dsa -P ""

$HOME/.ssh/id_dsa および $HOME/.ssh/id_dsa.pub の 2 つのファイルが生成されます。
```

3. 次のコマンドを実行して設定します。

```
$ cd $HOME/.ssh
$ touch authorized_keys2
$ cat id_dsa.pub >> authorized_keys2
$ chmod 0600 authorized_keys2
```

4. OpenSSH のバージョンによって、次のコマンドは不要場合があります。

```
$ ln -s authorized_keys2 authorized_keys
```

5. これで、SSH をパスワードなしで実行できます。テストするには、次のコマンドを実行します。

```
$ ssh <hostname>uname -a
```

上記の手順を実行しても SSH でパスワードが必要な場合は、システム管理者に連絡してください。

XPowerr (XPWR)

この章では、XPWR (XPowerr) コマンド ライン ツールについて説明します。次のセクションが含まれています。

- ・ [XPowerr の概要](#)
- ・ [XPowerr の構文](#)
- ・ [XPowerr のオプション](#)
- ・ [XPowerr のコマンド ラインの例](#)
- ・ [XPowerr の使用](#)
- ・ [消費電力レポート](#)

XPowerr の概要

XPowerr は、PAR (FPGA デザイン) または CPLDFit (CPLD デザイン) を実行した後、消費電力や温度の解析に使用します。XPowerr には、次のような機能があります。

- ・ デザインで使用される電力量を概算する。
- ・ ネットまたはロジック エLEMENTごとに使用される電力量を計算する。
- ・ ジャンクション温度の限界値を超えていないことを確認する。

XPowerr のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II

XPower のファイル

XPower で使用されるファイルは次のとおりです。

- ・ **CXT ファイル** : CPLDFit により生成されるファイル。消費電力量の計算や表示に使用します。
- ・ **NCD ファイル** : MAP および PAR により生成される、FPGA の情報が含まれた物理デザイン ファイル。電力を正確に概算するには、PAR により生成された配置配線済みの NCD を使用してください。MAP のみで生成された NCD を使用すると精度が低下することがあります。
- ・ **PCF ファイル** : オプションで MAP により生成される ASCII 形式の物理制約ファイル。ユーザーによる変更が可能です。PCF にはタイミング制約が含まれており、PERIOD 制約を使用してクロック ネットのスイッチ レートを特定するために使用します。これらの制約をユーザー制約ファイル (UCF) で設定した場合、温度および電圧情報も確認できます。
- ・ **VCD ファイル** : シミュレータから出力されるファイル。デザインの内部信号の周波数およびアクティビティ レートを設定する際に使用します。サポートされるシミュレータは、「SAIF または VCD データ入力」を参照してください。
- ・ **SAIF ファイル** : シミュレータから出力されるファイルで、より凝縮されたスイッチ データを提供します。SAIF は VCD よりもかなり小さく、高速に処理されますが、VCD と同様の結果が得られるはずです。
- ・ **XPO ファイル** : XPower からの設定ファイル。XPower で使用した周波数、トグル レート、容量性負荷などの設定を保存するファイル。次回同じデザインを読み込む際に、その設定が共に読み込まれるため便利です。

XPower の構文

FPGA デバイスで XPower を実行するには、次のコマンドライン構文を使用します。

```
xpwr infile[.ncd] [constraints_file[.pcf]] [options] -o design_name.pwr
```

CPLD デバイスで XPower を実行するには、次のコマンドライン構文を使用します。

```
xpwr infile[.cxt] [options] -o design_name.pwr
```

infile : 物理デザイン ファイルの名前を指定します。拡張子を付けずにファイル名を指定すると、NCD ファイルが使用されます。NCD が存在しない場合は、CXT ファイルが使用されます。

constraints_file : PCF ファイルの名前を指定します。このファイルは、デザインのタイミング制約を定義するためにオプションで使用します。PCF を指定しない場合、入力 NCD ファイルと同じルート名の PCF が使用されます。CXT を使用した場合、PCF は使用されません。

options : 「XPower のオプション」にリストされているオプションを 1 つ以上指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

design_name : 出力される消費電力レポートファイルの名前を指定します。拡張子は .pwr です。-o オプションでファイル名を指定しない場合、デフォルトで **infile** と同じルート名で拡張子が .pwr のファイルが生成されます。

XPower のオプション

このセクションでは、XPower のコマンドライン オプションについて説明します。

- ・ `-l` (行数の制限)
- ・ `-ls` (サポートされるデバイスをリスト)
- ・ `-s` (SAIF または VCD ファイルの指定)
- ・ `-o` (消費電力レポート ファイル名の変更)
- ・ `-tcl` (Tcl スクリプト)
- ・ `-v` (詳細レポート)
- ・ `-wx` (XML 設定ファイルの生成)
- ・ `-x` (XML 設定ファイルの指定)

使用できるオプションのリストと使用方法を表示するには、コマンドラインで「**xpwr -h**」と入力してください。

-l (行数の制限)

XPower で作成する詳細レポートの行数を制限します。

構文

`-l limit`

limit : 詳細レポートの最大行数を指定します。

-ls (サポートされるデバイスをリスト)

最新のソフトウェアでサポートされるザイリンクス デバイスをリストします。特定のアーキテクチャのデバイスのみを表示することも可能です。

構文

`-ls [architecture]`

architecture : デバイスをリストするアーキテクチャ (virtex5 など) を指定します。

-o (消費電力レポート ファイル名の変更)

出力する電力レポート ファイルを指定します。

構文

`-o reportname.pwr`

reportname.pwr : 消費電力レポートの名前を指定します。

このオプションを使用しない場合、消費電力レポートの名前は入力デザイン ファイル名に拡張子 `.pwr` を付けたものになります。

-s (SAIF または VCD ファイルの指定)

SAIF または VCD ファイルのデータを使用して、信号のアクティビティレートおよび周波数を設定します。

構文

-s [*simdata*.[*saif*|*vcd*]]

simdata : 使用する SAIF または VCD ファイルの名前を指定します。

ファイルを指定しない場合、入力デザイン ファイルと同じ名前で拡張子が *.vcd* のファイルが使用されます。

-tcl (Tcl スクリプト)

設定を適用するために使用する Tcl スクリプトを指定します。

構文

-tcl *tcl_script*

tcl_script : 設定を適用するために使用する Tcl スクリプトです。

-v (詳細レポート)

詳細レポートが生成されるよう指定します。

構文

-v

詳細は、「[消費電力レポート](#)」を参照してください。

-wx (XML 設定ファイルの生成)

現在の XPower 実行の設定情報を含む XML 設定ファイルを作成します。

構文

-wx [*userdata.xpa*]

userdata.xpa : 設定情報を保存する XML ファイルの名前を指定します。

ファイル名を指定しない場合、出力ファイル名は入力デザイン ファイル名に拡張子 *.xpa* を付けたものになります。

-x (XML 設定ファイルの指定)

既存の XML 設定ファイルを使用して、信号の周波数やその他の値を設定します。

構文

-x [*userdata.xpa*]

userdata.xpa : 設定情報を取得する XML ファイルの名前を指定します。

ファイル名を指定しない場合、入力デザイン ファイルと同じ名前で拡張子が *.xpa* のファイルが検索されます。

XPower のコマンド ラインの例

次のように入力すると、標準レポート (`mydesign.pwr`) が生成されます。信号のアクティビティレートと周波数は、SAIF ファイルを使用して指定できます。出力の負荷は変更されず、デフォルトの 10pF が使用されます。このデザインは、FPGA 用です。

```
xpwr mydesign.ncd mydesign.pcf -s timesim.saif
```

次のように入力すると、上記の操作に加えて、`mysettings.xpa` という設定ファイルが出力されます。設定ファイルには、SAIF ファイルのすべての情報が含まれます。

```
xpwr mydesign.ncd mydesign.pcf -s timesim.saif -wx mysettings.xpa
```

次のように入力すると、上記のすべてのコマンドが実行され、標準レポートの代わりに詳細レポートが出力されます。この場合、詳細レポートの行数は 100 行に制限されています。

```
xpwr mydesign.ncd mydesign.pcf -v -l 100 -s timesim.vcd -wx mysettings.xpa
```

XPower の使用

このセクションでは、消費電力および温度を正確に概算するための設定、および XPower で使用できるその他の概算方法について説明します。このセクションは、FPGA デザインに適用されます。CPLD デザインの場合は、ザイリンクスのサポート ページ <http://japan.xilinx.com/support> からアプリケーション ノート XAPP360『Obtaining Accurate Power Estimation for CoolRunner XPLA3 CPLDs Using XPower』を参照してください。

SAIF または VCD データ入力

XPower のフローには、PAR 後のシミュレーションから生成された SAIF または VCD ファイルを使用することをお勧めします。SAIF または VCD ファイルを生成するには、ザイリンクスでサポートされているシミュレータが必要です。詳細は、『[合成/シミュレーション デザイン ガイド](#)』を参照してください。

メモ： VCD ファイルは SAIF ファイルよりもサイズが大きく、処理にも時間がかかるので、通常は SAIF ファイルが推奨されます。

XPower では、次のシミュレータがサポートされています。

- ISim
- Mentor Graphics ModelSim
- NC-SIM (Cadence 社)
- Synopsys VCS and VCS MX

XPower は、SAIF または VCD ファイルを使用してデザイン内の信号のトグル レートや周波数を設定します。次の項目については、手動で入力してください。

- 電圧 (データブックの推奨値と異なる場合)
- 周囲温度 (デフォルトは 25°C)
- 出力負荷 (抵抗エレメントによる容量および電流)

最初に XPower を実行する場合、電圧と温度の制約が設定されていれば、PCF から電圧と周囲温度が使用されます。

デザインを次回 XPower に読み込むときに時間を節約するため、設定ファイル (XPA) を作成できます。電圧、温度、周波数、出力ロードなどの設定は、このファイルに保存されます。詳細は、『[-wx \(XML 設定ファイルの生成\)](#)』を参照してください。

その他のデータ入力方法

非同期信号は、すべて絶対周波数 (MHz) を使用するように設定されています。同期信号は、すべてアクティビティレートを使用するように設定されています。

アクティビティレートは 0 ~ 100% で表され、レジスタを介したエレメントの出力がアクティブ クロック エッジで変化する頻度を示します。たとえば、アクティビティレート 100% で、フリップフロップへ 100MHz のクロックが入力された場合、出力周波数は 50MHz です。

その他の方法で入力する場合は、次の項目を設定する必要があります。

- ・ 電圧 (データブックの推奨値と異なる場合)
- ・ 周囲温度 (デフォルトは 25°C)
- ・ 出力負荷 (抵抗エレメントによる容量および電流)
- ・ 全入力信号の周波数
- ・ 全同期信号のアクティビティレート

アクティビティレートを設定しない場合は、同期ネットすべてに 0% が適用され、入力信号の周波数には 0MHz が適用されます。デフォルトの周囲温度は、25°C です。デフォルトの電圧は、デバイスの推奨動作電圧です。

メモ: 上記の信号を正しく設定しておかないと、消費電力と温度の正確な値が算出されません。少なくとも、クロック ネット、クロック イネーブル、出力ネット、高速信号や負荷の大きい信号など、大量に電力を消費するネットは設定しておく必要があります。

消費電力レポート

このセクションでは、消費電力レポート (拡張子 `.pwr`) について説明します。

消費電力レポートには、次の 3 種類があります。

- ・ 標準消費電力レポート (デフォルト)
- ・ 詳細消費電力レポート (`-v` (詳細レポートの生成) オプションで生成されるレポート)

標準消費電力レポート

標準消費電力レポートには、次の項目が含まれています。

- ・ レポートのヘッダ
 - XPower のバージョン
 - 著作権情報
 - デザインおよび関連ファイルに関する情報 (読み込まれているデザイン ファイル名と PCF、シミュレーション ファイル名を含む)
 - データのバージョン情報
- ・ 消費電力サマリ (Power Summary) : 消費電力および電流の合計、その他のサマリ
- ・ 温度サマリ (Thermal Summary) : 次の項目が含まれています。
 - エアフロー
 - 概算されたジャンクション温度
 - 周囲温度
 - ケース温度
 - 熱抵抗 (Theta J-A)
- ・ デカップリング ネットワーク サマリ (Decoupling Network Summary) : キャパシタンス値、推奨値、個別のキャパシタンス範囲に分割された各電圧ソースの合計
- ・ フッタ : 解析を実行した日時

詳細レポート

詳細消費電力レポートには、標準レポートに表示されるすべての情報に加えて、ロジック、信号、クロック、入力、出力の詳細な電力情報が表示されます。

PIN2UCF

この章では、PIN2UCF について説明します。次のセクションから構成されています。

- ・ [PIN2UCF の概要](#)
- ・ [PIN2UCF の構文](#)
- ・ [PIN2UCF のオプション](#)

PIN2UCF の概要

PIN2UCF は、ピン固定制約をユーザー制約ファイル (UCF) に書き込むコマンドライン プログラムです。

FPGA の場合：

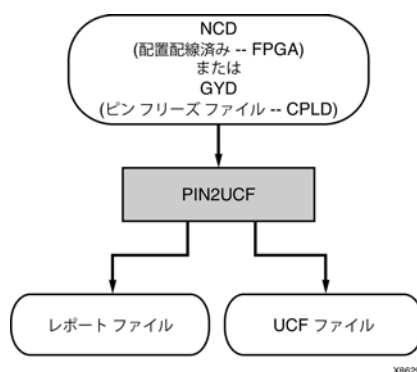
- ・ 配置配線済みデザインが必要です。
- ・ NCD (Native Circuit Description) ファイルを読み込みます。

CPLD の場合：

- ・ フィット済みデザインが必要です。
- ・ ガイド ファイル (GYD) を読み込みます。

ピン固定制約は、既存の UCF に書き込まれます。既存の UCF が存在しない場合は、新規の UCF が生成されます。

PIN2UCF のデザイン フロー



PIN2UCF のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

PIN2UCF のファイル タイプ

ファイル	タイプ	短縮形	デバイス	拡張子
Native Circuit Description	入力	NCD	FPGA	.ncd
ガイド ファイル	入力	GYD	CPLD	.gyd
レポート	出力	RPT	FPGA および CPLD	.rpt
ユーザー制約ファイル	出力	UCF	FPGA および CPLD	.ucf

PIN2UCF の入力ファイル

FPGA デザイン : NCD ファイルを入力します。少なくとも配置済みの NCD が必要ですが、タイミング仕様を満たしているか、ほぼ満たしている配置配線済みの NCD を使用するのが最適です。

CPLD デザイン : ガイド ファイル (GYD) を入力します。PIN2UCF は、CPLD デザインでピンを固定するために使用されていた以前の GYD ファイル構造に置き換わるものです。GYD をピン固定を制御するために使用することは可能ですが、GYD をガイド ファイルとして指定せずに、PIN2UCF を実行することをお勧めします。

PIN2UCF の出力ファイル

このセクションでは、PIN2UCF から出力される次のファイルについて説明します。

- ・ [PIN2UCF のユーザー制約ファイル \(UCF\)](#)
- ・ [PIN2UCF のピン レポート ファイル](#)

PIN2UCF のユーザー制約ファイル (UCF)

このセクションでは、PIN2UCF のユーザー制約ファイルについて説明します。次の項目が含まれます。

- ・ PIN2UCF のユーザー制約ファイル (UCF) について
- ・ PIN2UCF のユーザー制約ファイル (UCF) の PINLOCK セクション
- ・ PIN2UCF によるユーザー制約ファイル (UCF) への書き込み
- ・ PIN2UCF のユーザー制約ファイル (UCF) のコメント

PIN2UCF のユーザー制約ファイル (UCF) について

PIN2UCF は、入力ファイルからの情報をユーザー制約ファイル (UCF) に書き込みます。既存の UCF が存在しない場合は、新規の UCF が生成されます。PIN2UCF を実行するときに `output.ucf` ファイルを指定せず、デザイン ファイルのディレクトリ内にデザインと同じルート名の UCF が存在する場合、制約の競合がなければ、このファイルに制約が書き込まれます。詳細は、「PIN2UCF によるユーザー制約ファイル (UCF) への書き込み」を参照してください。

PIN2UCF のユーザー制約ファイル (UCF) の PINLOCK セクション

ピン固定制約は、UCF ファイルの PINLOCK セクションに書き込まれます。PINLOCK セクションは、次の文で識別できます。

- ・ #PINLOCK BEGIN で開始します。
- ・ #PINLOCK END で終了します。

デフォルトでは、競合する制約は UCF に書き込まれません。

ユーザーが指定したピン固定制約は、UCF で上書きされません。ただし、ユーザー指定の制約と PIN2UCF により生成された制約が完全に一致する場合、ユーザー指定のロケーション制約文の前にシャープ記号 (#) が追加されます。シャープ記号は、文がコメントであることを示します。

元の UCF (PINLOCK セクションを含まないファイル) を復元するには、次を削除します。

- ・ PINLOCK セクション
- ・ ユーザー指定のロケーション制約文のシャープ記号

PIN2UCF では、UCF の既存の制約が有効なピン固定制約であるかどうかは確認されません。

PIN2UCF によるユーザー制約ファイル (UCF) への書き込み

PIN2UCF は、次の表に示す条件に基づいて UCF に制約を書き込みます。

PIN2UCF による処理

状況	PIN2UCF による処理	作成/更新されるファイル
UCF が存在しない	UCF を生成し、ピン固定制約をその UCF に記述。	pinlock.rpt <i>design_name.ucf</i>
UCF が存在する PINLOCK セクションの内容が PIN2UCF で生成されたものとすべて一致しており、PINLOCK セクションと UCF のその他の部分との間に競合がない PINLOCK セクションの内容がすべてコメントになっており、PINLOCK セクション以外の部分に競合がない PINLOCK セクションがなく、UCF に競合がない	既存の UCF ファイルに記述	pinlock.rpt <i>design_name.ucf</i>

状況	PIN2UCF による処理	作成/更新されるファイル
UCF が存在する UCF にピン固定制約がないか、PINLOCK セクション外にユーザー指定のピン固定制約が含まれている ユーザー指定の制約と PIN2UCF により生成された制約との間に競合がない	既存の UCF ファイルに記述ファイルの最後の PINLOCK セクションにピン固定制約を追加。	pinlock.rpt <i>design_name.ucf</i>
UCF が存在する UCF の PINLOCK セクションの内部または外部にユーザー指定のピン固定制約が含まれている ユーザー指定の制約の一部に PIN2UCF により生成された制約と競合するものがある	既存の UCF ファイルに記述 PINLOCK セクションは記述しない。エラー メッセージを表示して終了し、競合する制約のリストを出力。	pinlock.rpt
UCF が存在する UCF にピン固定制約が含まれていない 前回の PIN2UCF の実行で生成されたか、ユーザーが手動で追加した PINLOCK セクションが UCF に含まれている PINLOCK セクションの制約と PIN2UCF によって生成された制約との間に競合がない	既存の UCF ファイルに記述 既存の PINLOCK セクションを削除して、新規 PINLOCK セクションを UCF に記述。既存 PINLOCK セクションの内容を新規 PINLOCK セクションに移動。	pinlock.rpt <i>design_name.ucf</i>

PIN2UCF のユーザー制約ファイル (UCF) のコメント

新たに PIN2UCF を実行した場合、既存の PINLOCK セクション内のコメントは保持されません。CSTTRANS コメントが検出された場合、「INST *name*」と「NET *name*」が同等と見なされ、コメントがチェックされます。

PIN2UCF のピン レポート ファイル

PINLOCK セクションが作成される前に UCF ファイルに競合する制約が検出されると、pinlock.rpt というレポート ファイルにその情報が出力されます。このレポート ファイルは、デフォルトでは作業ディレクトリに保存されます。別のディレクトリに保存する場合は、**-r** オプションを使用してディレクトリを指定します。詳細は、「[r \(レポートファイルへの出力\)](#)」を参照してください。

レポート ファイルには、次のセクションが含まれます。

- ・ 制約の競合に関する情報
- ・ エラーと警告のリスト

制約の競合に関する情報

このセクションには、次のサブセクションがあります。

- ・ ピン上でのネット名の競合
- ・ ネット上でのピン名の競合

競合する制約がない場合は、両方のサブセクションに競合がないことが示されます。

制約の競合に関する情報のセクションは、入力がない場合や入力が無効な場合など、致命的な入力エラーがある場合は出力されません。

エラーと警告のリスト

このセクションは、エラーまたは警告がある場合のみ表示されます。

PIN2UCF の構文

PIN2UCF を実行するには、次のコマンドを使用します。

```
pin2ucf ncd_file.ncd|pin_freeze_file.gyd [-rreport_file_name -o  
output.ucf]
```

- ・ *ncd_file* : 配置配線済みの NCD ファイル名 (FPGA)
- ・ *pin_freeze_file* : フィット済みの GYD ファイル名 (CPLD)

PIN2UCF のオプション

このセクションでは、PIN2UCF のコマンドライン オプションについて説明します。

- ・ **-o** (出力ファイル名)
- ・ **-r** (レポート ファイルへの出力)

-o (出力ファイル名)

デフォルト (**-o** オプションなし) では、*ncd_file.ucf* という UCF ファイルに書き込まれます。このオプションは、次のような場合に使用します。

- ・ デザインとは異なるルート名の UCF を書き込む場合
- ・ デザインとは異なるルート名の UCF にピン固定制約を書き込む場合
- ・ UCF を異なるディレクトリに保存する場合

構文

```
-o outfile.ucf
```

-r (レポート ファイルへの出力)

デフォルト (**-r** オプションなし) では、*pinlock.rpt* というレポートファイルが生成されます。このオプションを使用すると、レポートファイルの名前を指定できます。

構文

```
-r report_file_name.rpt
```


TRACE

この章では、TRACE (Timing Reporter And Circuit Evaluator) について説明します。次のセクションが含まれています。

- ・ [TRACE の概要](#)
- ・ [TRACE の構文](#)
- ・ [TRACE のオプション](#)
- ・ [TRACE のコマンドラインの例](#)
- ・ [TRACE のレポート](#)
- ・ [OFFSET 制約](#)
- ・ [PERIOD 制約](#)
- ・ [TRACE の停止](#)

TRACE の概要

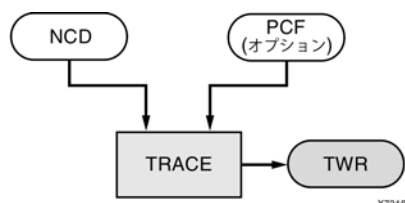
TRACE (Timing Reporter And Circuit Evaluator) プログラムを使用すると、入力したタイミング制約に基づいて、FPGA デザインのスタティック タイミング解析を実行できます。

TRACE には、主に次の 2 つの機能があります。

- ・ **タイミング検証**：デザインがタイミング制約を満たしているかどうかを検証する機能。
- ・ **レポート**：デザインが入力した制約をどれだけ満たしているかをレポートする機能。未配置のデザイン、配置のみが終了したデザイン、配置配線が一部終了したデザイン、完全に配置配線されたデザインに TRACE を実行できます。

次に、TRACE に使用する入力ファイルと出力ファイルを示します。NCD ファイルは、MAP または PAR から出力されたデザイン ファイル (拡張子 .ncd) です。オプションの物理制約ファイル (PCF) の拡張子は .pcf です。TWR は、タイミング レポート ファイル (拡張子 .twr) です。

TRACE の入力ファイルと出力ファイル



TRACE のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

TRACE の入力ファイル

TRACE に入力するファイルは、マップ済み、配置済み、または完全に配置配線された NCD ファイルと、オプションの PCF ファイルです。PCF は、指定したタイミング制約に基づいて MAP により生成されます。制約として、入力信号のクロック スピード、複数の信号間の外部タイミング関係、デザイン パスの絶対最大遅延、ピン タイプごとの一般的なタイミング要件などを指定できます。

- ・ **NCD ファイル**：マップ済み、配置済み、または配置配線済みのデザイン ファイル。TRACE から出力されるタイミング情報の内容は、入力するデザインが配置されていないか (MAP 後)、配置のみが終了しているか、配置配線がすべて終了しているかによって異なります。
- ・ **PCF ファイル**：MAP により作成される ASCII 形式の物理制約ファイル。TRACE のタイミング解析で使用するタイミング制約が含まれており、オプションで変更が可能です。

TRACE の出力ファイル

TRACE から出力されるタイミング レポートは、次のとおりです。

- ・ **TWR ファイル**：デフォルトのタイミング レポート。コマンドラインで入力するオプションによって、異なるレポートが生成されます。エラー レポートを出力するには **-e** オプションを、詳細レポートを出力するには **-v** オプションを使用します。
- ・ **TWX ファイル**：**-xml** オプションを使用すると出力される XML タイミング レポート。レポートは、Timing Analyzer で開くことができます。**-e** オプションおよび **-v** オプションは、TWX ファイルおよび TWR ファイルの両方に適用されます。詳細は、「[-xml \(XML 出力ファイル名\)](#)」を参照してください。

TRACE の **-stamp** オプションを使用すると、STAMP タイミング モデルを生成できます。詳細は、「[-stamp \(STAMP タイミング モデル ファイルの生成\)](#)」を参照してください。

メモ：TRACE で生成されるタイミング レポートについては、「[TRACE のレポート](#)」を参照してください。

TRACE の構文

TRACE を実行するには、次の構文を使用します。

```
trce [options] design[.ncd] [constraint [.pcf]]
```

options：「[TRACE のオプション](#)」にリストされているオプションをいくつでも指定できます。**-stamp (STAMP タイミング モデル ファイルの生成)** オプションを使用しない場合は、オプションを特定の順序で入力する必要はありません。オプションを複数指定する場合は、スペースで区切ります。

design：入力デザイン ファイルを指定します。拡張子を付けずにファイル名を指定すると、その名前の NCD ファイルが使用されます。

constraint : 物理制約ファイル (PCF) の名前を指定します。このファイルは、デザインのタイミング制約を定義します。PCF ファイルを指定しない場合、入力 NCD ファイルと同じルート名の PCF が使用されます。

TRACE のオプション

このセクションでは、TRACE のコマンドライン オプションについて説明します。

- ・ `-a` (アドバンス解析)
- ・ `-e` (エラー レポートの生成)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-fastpaths` (最速パスをレポート)
- ・ `-intstyle` (統合スタイル)
- ・ `-filter` (フィルタ ファイルの指定)
- ・ `-l` (タイミング レポートに表示される項目を制限)
- ・ `-n` (パスをエンドポイント別にレポート)
- ・ `-nodatasheet` (データシートを出力しない)
- ・ `-o` (出力タイミング レポートのファイル名を指定)
- ・ `-s` (スピードの変更)
- ・ `-stamp` (STAMP タイミング モデル ファイルの生成)
- ・ `-tsi` (TSI レポートの生成)
- ・ `-u` (制約が適用されないパスをレポート)
- ・ `-v` (詳細レポートの生成)
- ・ `-xml` (XML 出力ファイル名)

`-a` (アドバンス解析)

PCF からのタイミング制約を使用しない場合にのみ使用します。このオプションは、次の情報を含むタイミング レポートを生成します。

- ・ すべてのクロックと、各クロックに必要な OFFSET の解析
- ・ 組み合わせのロジックのみを含むパスの解析 (遅延順)

この情報は、レポートの種類 (サマリ、エラー、詳細) によって表示されるデフォルトの情報の代わりに表示されます。

構文

`-a`

メモ : 特定のクロック信号に関連するパスの解析では、始点および終点と同じクロック エッジで読み込まれるパスに対してのみ、ホールド違反 (レース コンディション) がチェックされます。

`-e` (エラー レポートの生成)

エラー レポートを生成します。エラー レポートの例は、「[エラー レポート](#)」を参照してください。

構文

-e [*limit*]

レポート名は、入力デザインと同じルート名に拡張子 `.twr` が付いたものになります。

limit : 各タイミング制約に対してレポートされるパス数を制限します。有効な値は 0 ~ 32,000 の整数で、デフォルト値は 3 です。

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f \(コマンド ファイルの実行\)](#)」を参照してください。

-fastpaths (最速パスをレポート)

デザインの最速パスをレポートします。

構文

-fastpaths

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle *ise* | *xflow* | *silent*

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-filter (フィルタ ファイルの指定)

フィルタ ファイルを指定します。フィルタ ファイルには、プログラムの実行中に生成されるメッセージを保存およびフィルタ処理するための設定が含まれています。

構文

-filter [*filter_file*]

デフォルトのフィルタ ファイル名は `filter.filter` です。

-l (タイミング レポートに表示される項目を制限)

レポート ファイルで、タイミング制約ごとにレポートされる項目の数を制限します。limit に設定可能な値は、0 ~ 2,000,000,000 の整数で、デフォルト値は 3 です。

構文

-l limit

メモ： 指定する値が大きいほど、タイミング レポートの生成に時間がかかります。

-n (パスをエンドポイント別にレポート)

パスをエンドポイント別にレポートします。デフォルトでは、パスは制約別にレポートされます。エンドポイント数を制限して、レポートの生成時間を短縮できます。

構文

-n limit

limit： レポートするエンドポイントの数を指定します。有効な値は、0 ~ 2,000,000,000 の整数値です。

メモ： 指定する値が大きいほど、タイミング レポートの生成に時間がかかります。

-nodatasheet (データシートを出力しない)

レポートにデータシートを含めません。

構文

-nodatasheet

-o (出力タイミング レポートのファイル名を指定)

出力されるタイミング レポートの名前を指定します。拡張子 *.twr* は省略できます。このオプションを使用しない場合、タイミング レポートのルート名は NCD ファイルと同じになります。

構文

-o report [.twr]

-s (スピードの変更)

入力 NCD ファイルに含まれるデバイスのスピードではなく、このオプションで指定したデバイスのスピードを解析に使用します。**-s** オプションは、TRACE の実行で生成されるすべてのレポートタイプに適用されます。このオプションを使用することで、タイミング要件に合うスピード グレードを調べることができます。

構文

-s [speed]

デバイスのスピード (*speed*) には、マイナス記号を付けることができます。たとえば、**-s 3** と **-s -3** は、どちらも有効な値です。

アーキテクチャによっては、最小タイミング解析がサポートされています。最小タイミング解析を実行するためのコマンドライン構文は、「`trce -s min`」です。min の前にはマイナス記号を付けないでください。

メモ : このオプションは、タイミング解析時のスピード グレードを変更するだけです。このスピード グレードは、NCD ファイルには保存されません。

-stamp (STAMP タイミング モデル ファイルの生成)

デザインのタイミング特性を記述した STAMP タイミング モデル ファイル (stampfile.mod および stampfile.data) を生成します。

構文

```
-stamp stampfile design.ncd
```

メモ : STAMP ファイル (stampfile) は、NCD ファイルの前に入力する必要があります。

スタティック タイミング解析では、どのプリント基板にも STAMP コンパイラを使用できます。

-stamp オプションを使用して TRACE を実行し、完全な STAMP モデルのレポートを生成するには、次の 4 つの方法があります。

- ・ **-a** オプションを使用して高度な解析を実行する。
- ・ デフォルトの解析を実行する (制約ファイルを使用せず、高度な解析を実行しない)。
- ・ デザインのすべてのパスに制約が適用されるよう設定する。
- ・ デザインの一部のみに適用される制約に対して、制約が適用されないパスのレポート (**-u** オプション) を使用して解析を実行する。

後の 2 つの方法で実行する場合は、モデルからパスが除外されるのを防ぐため、PCF に TIG を含めないでください。

-tsi (TSI レポートの生成)

タイミング制約間の相互関係を示す TSI (Timing Specification Interaction) レポートを生成します。このファイルは、次の用途に使用できます。NCD や PCF と関連のない名前でもかまいません。制約を解析するため NCD ファイルおよび PCF ファイルも指定できます。

構文

```
-tsi designfile.tsi designfile.ncd designfile.pcf
```

-u (制約が適用されないパスをレポート)

タイミング制約が適用されていないパスの遅延をレポートします。limit を指定すると、レポートされるパスの数を制限できます。既存の制約に「制約のないパスの解析」という制約が追加されます。制約が適用されていないパスに対して、デフォルトのパスがリストされます。デフォルトのパスには、順次コンポーネント上のデータおよびクロック ピンへの回路パスや、プライマリ出力のデータ ピンへの回路パスなどが含まれます。

構文

```
-u limit
```


limit : タイミング制約ごとにレポートされる、制約のないパスの数を制限します (オプション)。有効な値は 1 ~ 2,000,000,000 の整数で、デフォルト値は 3 です。

TRACE レポートには、制約が適用されていないパスに対して次の情報が含まれます。

- ・ 制約が適用されていないパスから順次コンポーネントへの最小周期
- ・ 組み合わせロジックのみを含む制約が適用されていないパスの最大遅延
- ・ 順次パスの周期と組み合わせパスの遅延リスト (詳細レポートのみ)。遅延は大きい順にリストされます。表示される項目の数は、**-v** オプションを使用する時に指定できます。

メモ : 制約が適用されていないパスに含まれるレジスタ間のパスの場合、始点および終点と同じクロック エッジで読み込まれるパスに対してのみ、ホールド違反 (レース コンディション) がチェックされます。

-v (詳細レポートの生成)

詳細レポートを生成します。レポート名は、入力デザインと同じルート名に拡張子 **.twr** が付いたものになります。ルート名に異なる名前を指定できますが、拡張子は必ず **.twr** にしてください。

構文

-v limit

limit : タイミング制約ごとにレポートされる項目の数を制限します。有効な値は 1 ~ 32,000 の整数で、デフォルト値は 3 です。

-xml (XML 出力ファイル名)

出力される XML タイミング レポート (TWX) ファイルの名前を指定します。拡張子 **.twx** は省略できます。

メモ : XML レポートは、Timing Analyzer でのみ表示できます。詳細は、Timing Analyzer ヘルプを参照してください。

構文

-xml outfile[.twx]

TRACE のコマンド ラインの例

例 1

trce design1.ncd group1.pcf

このコマンドは、デザイン ファイル **design1.ncd** のタイミング特性を検証し、サマリレポートを生成します。ファイル **group1.pcf** に含まれるタイミング制約は、デザインのタイミング制約です。出力されるサマリレポートのファイル名は、**design1.twr** です。

例 2

trce -v 10 design1.ncd group1.pcf -o output.twr

このコマンドは、**group1.pcf** に含まれるタイミング制約を使用して、デザイン **design1.ncd** のタイミング特性を検証し、詳細レポートを生成します。詳細レポートのファイル名は、**output.twr** です。

例 3

```
trce -v 10 design1.ncd group1.pcf -xml output.twx
```

このコマンドは、group1.pcf に含まれるタイミング制約を使用して、デザイン design1.ncd のタイミング特性を検証し、詳細レポートを TWR と XML の両方の形式で生成します。TWR レポートのファイル名は design1.twr、XML レポートのファイル名は output.twx になります。

例 4

```
trce -e 3 design1.ncd timing.pcf
```

このコマンドは、timing.pcf に含まれるタイミング制約を使用して、デザイン design1.ncd のタイミング特性を検証し、エラー レポートを生成します。エラー レポートは、timing.pcf に記述された制約ごとにワースト エラーを 3 つずつ報告します。エラー レポートのファイル名は、design1.twr です。

TRACE のレポート

TRACE では、デザインのタイミング制約が満たされているかどうかを確認できる ASCII 形式のタイミング レポート ファイルが出力されます。ファイルには拡張子 .twr が付いた名前が付けられ、作業ディレクトリに保存されます。デフォルトのファイル名は、NCD のルート名と同じです。ルート名に異なる名前を指定できますが、拡張子は必ず .twr にしてください。拡張子を指定しない場合、.twr が自動的に付けられます。

タイミング レポートには、デザインに関する統計、検出されたタイミング エラー、警告に関する情報が表示されます。

「Timing errors」には、次の内容を含む、絶対的または相対的なタイミング制約違反が表示されます。

- ・ **パス遅延エラー**：パス遅延がパスに設定された MAXDELAY 制約を超えている。
- ・ **ネット遅延エラー**：ネット接続遅延がネットに設定された MAXDELAY 制約を超えている。
- ・ **オフセット エラー**：外部クロックと関連するデータ入力ピンの間の遅延オフセットが内部ロジックのタイミング要件を満たすには不十分である。または、外部クロックと関連するデータ出力ピンの間の遅延オフセットが外部ロジックのタイミング要件を超えている。
- ・ **ネット スキュー エラー**：ネット接続間のスキューがネットの最大スキュー制約を超えている。

タイミング エラーを修正するには、デザインまたは制約を変更するか、PAR を再実行します。

「Warnings」は、回路のループや、どのパスにも適用されない制約などの問題点を指摘します。

出力されるレポートには、サマリ、エラー、詳細の 3 種類があります。レポートタイプを指定するには、該当する TRACE オプションを入力するか、Timing Analyzer からレポートタイプを選択してください（「[TRACE のオプション](#)」を参照）。各レポートタイプの詳細は、「[TRACE のレポートの内容](#)」を参照してください。

ASCII 形式のタイミング レポート (TWR) に加え、**-xml** オプションを使用すると、XML タイミング レポート (TWX) を生成できます。XML レポートは、Timing Analyzer でのみ表示できます。

TRACE によるタイミング検証

TRACE は、NCD ファイルの遅延をタイミング制約と比較します。遅延が許容範囲を超えている場合、タイミング エラーが報告されます。

メモ： タイミング制約の値は、2ms 以下に制限することをお勧めします。これ以上の値にすると、タイミング レポートで良くない結果が報告されます。

ネット遅延制約

MAXDELAY 制約を使用した場合、制約を設定したネットの遅延がチェックされ、routedelay (配線遅延) が netdelayconstraint (NETDELAY 制約) 以下 ($\text{routedelay} \leq \text{netdelayconstraint}$) であるかどうかを確認されます。

routedelay： ネット上のドライバ ピンとロード ピンの間の信号遅延です。デザインが配置されただけで配線されていない場合は、概算値になります。

ネットの遅延がこの条件を満たさない場合、タイミング レポートにタイミング エラーが表示されます。

ネット スキュー制約

USELOWSKEWLINES 制約または MAXSKEW 制約を使用した場合、ロード ピンが複数あるネットで発生する信号スキューは、最小遅延と最大遅延の差 ($\text{signalskew} = (\text{maxdelay} - \text{mindelay})$) になります。

- ・ **mindelay：** ドライバ ピンとロード ピンの間の最大遅延です。
- ・ **maxdelay：** ドライバ ピンとロード ピンの間の最小遅延です。

メモ： MAXDELAY 制約に含まれるレジスタ間のパスの場合、始点および終点が同じクロックエッジで読み込まれるパスに対してのみ、ホールド違反 (レース コンディション) がチェックされます。

PCF で制約が設定されているネットの場合、スキューを検証し、signalskew が MAXSKEW 制約以下 ($\text{signalskew} \leq \text{maxskewconstraint}$) であるかどうかを確認します。

信号スキューが MAXSKEW 制約を超えている場合、タイミング レポートにスキュー エラーが表示されます。

パス遅延制約

PERIOD 制約を使用すると、pathdelay (パス遅延) は、ロジック (コンポーネント) 遅延、配線 (ワイヤ) 遅延、およびセットアップ タイム (該当する場合) の合計から、クロック スキュー (該当する場合) を引いた値 ($\text{pathdelay} = \text{logicdelay} + \text{routedelay} + \text{setuptime} - \text{clockskew}$) になります。

- ・ **logicdelay：** コンポーネントのピン間の遅延です。
- ・ **routedelay：** パスのコンポーネントピン間の信号遅延です。デザインが配置されただけで配線されていない場合は、概算値になります。
- ・ **setuptime：** クロック信号のトリガ エッジが到達するまでに、データが入力ピンに準備されていなければならない時間です (クロックが供給されるパスのみ)。
- ・ **clockskew：** クロック信号がデスティネーション レジスタに到達するまでの時間と、ソース レジスタに到達するまでの時間の差です (レジスタ間にクロックが供給されるパスのみ)。クロック スキューについては、次のセクションで説明します。

制約が設定されたパスの遅延を検証し、pathdelay が maxpathdelayconstraint 以下 ($\text{pathdelay} \leq \text{maxpathdelayconstraint}$) であるかどうかを確認します。

パス遅延がこの条件を満たさない場合、タイミング レポートにタイミング エラーが表示されます。

クロック スキューとセットアップの検証

レジスタ間のセットアップの検証では、クロック スキューを考慮する必要があります。レジスタ間のパスでは、1 クロック周期以内に、データがデスティネーション レジスタに到達する必要があります。タイミング解析ソフトウェアでは、ソース レジスタとデスティネーション レジスタの間のクロック スキューがこの検証で考慮されます。

メモ : デフォルトでは、専用以外のすべてのクロック、ローカル クロック、専用クロックのスキューが解析されます。

レジスタ間のパスで実行されるセットアップの検証では、 $\text{Slack} = \text{constraint} + \text{Tsk} - (\text{Tpath} + \text{Tsu})$ であることが確認されます。

- ・ **constraint** : パスに必要な時間です。FROM-TO 制約を使用して明示的に指定するか、PERIOD 制約から算出します。
- ・ **Tpath** : パス上のコンポーネントと接続遅延の合計です。
- ・ **Tsu (セットアップ)** : デスティネーション レジスタのセットアップ要件です。
- ・ **Tsk (スキュー)** : 信号がデスティネーション レジスタに到着する時間と、ソース レジスタに到着する時間の差です。
- ・ **TSlack** : 値が負の場合、ソース レジスタのデータが次のクロック エッジまでにターゲット レジスタでセットアップされないため、セットアップ エラーが発生する可能性があることを示します。

クロック スキュー

クロック スキュー Tsk は、クロック入力 (CLKIOB) からレジスタ D (TclkD) までの遅延から、クロック入力 (CLKIOB) からレジスタ S (TclkS) までの遅延を引いた値です。デスティネーション レジスタのスキューが負の場合、データ パスに許容される時間が短くなり、正の場合はデータ パスに許容される時間が長くなります。

複数のバッファを経由するクロック

クロック パス遅延の合計を使用して、ソース レジスタにクロックが到着する時間 (TclkS) とデスティネーション レジスタにクロックが到着する時間 (TclkD) が決定されるので、ソース クロックとデスティネーション クロックが同じチップ入力から供給されていれば、異なるクロック バッファや配線リソースを経由する場合でも、この検証が適用されます。

ソース クロックとデスティネーション クロックが異なるチップ入力から供給される場合は、TRACE でクロックの到着時間や位相情報が識別されないため、2 つのクロック入力の関係は明確になりません。

異なるデバイス入力から供給されるクロック

FROM-TO 制約を指定すると、チップ入力間の外部タイミング関係が考慮されていると TRACE で判断されるため、両方のクロック入力と同時に到着し、デスティネーション クロックの到着時間 (TclkD) とソース クロックの到着時間 (TclkS) の差に 2 つのチップ クロック入力への到着時間の差が含まれていないと想定されます。

ソース レジスタおよびデスティネーション レジスタへのクロック パスに異なるクロックが供給されている場合、PERIOD 制約が適用されているセットアップの検証で、クロック スキュー Tsk は考慮されません。

TRACE のレポートの内容

TRACE により生成されるタイミング レポートは、特定デザインの ASCII 形式のファイル (TWR) で、デザインに関する統計情報、タイミングの警告とエラーのサマリ、詳細なネットおよびパス遅延 (オプション) をレポートします。ASCII 形式のレポートは、すべて等幅フォントでの表示用にフォーマットされます。レポートの表示に使用するテキスト エディタで等幅ではないフォントを使用した場合、レポートの列は正しく並びません。

-xml オプションを使用すると、TWR ファイルに加え、XML タイミング レポート (TWX) ファイルを生成できます。XML タイミング レポートの内容は、ASCII タイミング レポートと同じです。XML レポートは、Timing Analyzer でのみ表示できます。

このセクションでは、TRACE で生成されるタイミング レポートについて説明します。

- ・ **サマリ レポート** : サマリ情報、デザイン統計情報、PCF ファイルの制約に関する統計情報を表示します。
- ・ **エラー レポート** : タイミング エラーと、関連するネットまたはパスの遅延に関する情報を表示します。
- ・ **詳細レポート** : すべてのネットおよびパスの遅延に関する情報を表示します。

レポートのヘッダには、レポートの生成に使用したコマンドライン、レポートの種類、入力デザイン名、オプションで入力した PCF 名、スピード ファイルのバージョン、入力 NCD のデバイス、スピード データが記載されます。レポートの最後には、次の情報を含むサマリがあります。

- ・ デザインで検出されたタイミング エラー数。この情報は、すべてのレポートに表示されます。
- ・ デザインのすべてのタイミング制約に対するエラーの合計を示したタイミング スコア (単位 : ピコ秒)
- ・ 制約が適用されるパスおよびネットの数
- ・ 配線遅延数とタイミング制約が適用される接続の割合

メモ : タイミング制約が適用される接続の割合は、「% coverage」と表示されます。この統計情報は、制約が適用されるパスの割合ではなく、制約が適用される接続の割合を表します。デザイン内のすべてのパスに制約が適用されていても、この割合が 100% 未満になる場合があります。これは、STARTUP コンポーネントへの接続など、スタティック タイミング解析に含まれない接続があるためです。

次のセクションで、各レポートの説明と例を示します。

タイミング レポートには、次の情報も示されます。

- ・ どのレポートに対しても、指定した物理制約ファイルに無効なデータが含まれている場合、物理制約ファイルのエラーのリストがレポートの始めに表示されます。これには、制約構文のエラーも含まれます。
- ・ タイミング レポートで、遅延の前にチルダ (~) が付いている場合、遅延値が概算であることを表します。このような値は、ネット上の過剰な遅延、抵抗、容量のため、正確には計算できません。パスが複雑すぎるため、正確には計算できないということです。

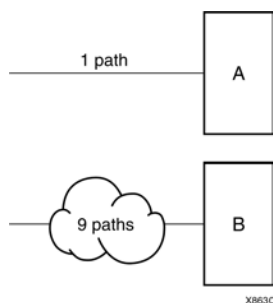
チルダ (~) は、パスが数値を最大で 20% 超えている可能性があることも表します。PENALIZE TILDE 制約を使用すると、このような遅延に対して、指定の割合でペナルティを課すことができます。PENALIZE TILDE 制約の詳細は、『[制約ガイド](#)』を参照してください。

- ・ 遅延の前に e が付いている場合、パスが配線されていないため、遅延が概算であることを表します。
- ・ TRACE は、パスのループを検出し、デザインで検出されたループの合計数をレポートします。ループが検出されると、そのループは無効となり、解析されません。ループが多く、配線経路から構成される場合、問題のループに接続するすべてのパスの配線が無効になり、合計数がレポートに含まれます。

パスは、デザイン内のほかのパスに影響されることなくループすると見なされます。したがって、有効なパスが別のパスのループにつながっていても、駆動出力ではなく入力で合流している場合は、そのパスは無効にはならず、ほかのパスで無効になる可能性のあるループの一部を含むことになります。

- ・ 次に示すように、エラー数は、タイミング制約を満たさないパスの数ではなく、そのパスの終点 (レジスタ セットアップ入力、出力パッド) の数です。

タイミング制約のエラー



A と B のエンドポイントでエラーが発生した場合、タイミング レポートには各エンドポイントに 1 つずつエラーが表示されます。

データシート レポート

データシートレポートは、デザインの外部タイミング パラメータを要約します。エラー レポートと詳細レポートの場合、データシートレポートには、制約が適用された入力、出力、クロックのみが表示されます。レポートに表示される内容は、タイミング パスの種類や、適用されたタイミング制約によって異なります。制約のないパスの解析を実行すると、制約ファイルで明示的に指定されていないパスを含めたレポートが生成されます。PCF ファイルを使用しない場合、すべての I/O タイミングが解析され、レポートされます。ただし、デフォルトのバストレーン制御の影響はレポートされません。データシートレポートには、ソースとデスティネーションのパッド名に加え、ソースとデスティネーション間の伝搬遅延、またはデスティネーションに対応するソースのセットアップ要件とホールド要件が含まれます。一部のパッケージでは、パッケージのフライト タイムも含まれます。

TRACE を実行して完全なデータシートレポートを生成するには、次の 4 つの方法があります。

- ・ **-a** オプションを使用して高度な解析を実行する。
- ・ デフォルトの解析を実行する (制約ファイルを使用せず、高度な解析を実行しない)。
- ・ デザインのすべてのパスに制約が適用されるよう設定する。
- ・ デザインの一部のみに適用される制約に対して、制約のないパス レポートを使用して解析を実行する。

データシートレポートに表示される項目は次のとおりです。

- ・ Input Setup and Hold Times (入力セットアップ/ホールド タイム)

この表は、ソース パッドの入力クロックに対して、入力信号のセットアップ タイム/ホールド タイムを表示します。DCM/DLL からの位相は考慮されません。入力信号が 2 つの異なるデスティネーションに送られている場合、その信号のセットアップ タイム/ホールド タイムには、ワースト ケースのものが表示されます。この場合、セットアップ タイムとホールド タイムのデスティネーションが異なっている可能性があります。

- ・ Output Clock to Out Times (出力 clock-to-out タイム)

この表は、ソース パッドの入力クロックに対して clock-to-out 信号を表示します。DCM/DLL からの位相は考慮されません。出力信号が、同じクロックが供給される異なるソースの組み合わせ結果である場合、clock-to-out は、ワースト ケース パスの値が表示されます。

- ・ Clock Table (クロック表)

異なるクロック同士の関係を表します。Source Clock (ソース クロック) 列にすべての入力クロック、2 列目にソース クロックとデスティネーション クロックの立ち上がりエッジ間の遅延が、3 列目にソース クロックの立ち下がりエッジとデスティネーション クロックの立ち上がりエッジのデータ遅延が表示されます。

ソース フリップフロップごとにデスティネーション フリップフロップを 1 つずつ指定しておくと、デザインは正常に動作します。1 つのソースが関連のないクロックを持つ 2 つのフリップフロップに送られた場合、データ遅延が異なるため、一方のフリップフロップがデータを受信しても、もう一方のフリップフロップは受信しない可能性があります。

階層レポート ブラウザで [Data Sheet report] をクリックすると、データシートレポートにすばやく移動できます。

- ・ External Setup and Hold Requirements (外部セットアップ/ホールド要件)

プライマリ クロック入力のすべての派生クロックに対してクロック位相関係や DCM 位相シフトが考慮され、プライマリ入力ごとにセットアップ要件とホールド要件が個別のデータシートレポートに表示されます。

デバイスのデータ入力の最大セットアップ タイムと最大ホールド タイムは、各クロック入力を基準に表示されます。デバイスのクロック入力に対してデータ入力からのパスが複数存在する場合は、ワーストケースのセットアップ タイムとホールド タイムが表示されます。デザインのデータ入力とクロック入力の組み合わせに対して、それぞれワーストケースのセットアップ タイムとホールド タイムが 1 つずつ表示されます。

外部セットアップ要件とホールド要件は、次のように表示されます。

Setup/Hold to clock ck1_i			
-----+-----+-----+-----			
	Setup to	Hold to	
Source Pad	clk (edge)	clk (edge)	
-----+-----+-----+-----			
start_i	2.816 (R)	0.000 (R)	
-----+-----+-----+-----			

- ・ User-Defined Phase Relationships (ユーザー定義の位相関係)
ユーザー定義の内部クロックのセットアップ要件とホールド要件が個別に表示されます。ユーザー定義の外部クロックでは、個別に表示されません。
- ・ Clock-to-Clock Setup and Hold Requirements (clock-to-clock セットアップ/ホールド要件)
内部クロックのセットアップ要件とホールド要件は、個別に表示されません。
- ・ Guaranteed Setup and Hold (確認済みセットアップ/ホールド要件)
スピード ファイルに含まれる確認済みセットアップ/ホールド要件は、詳細なタイミング解析で算出されたセットアップ/ホールド要件よりも優先されます。確認済みセットアップ/ホールド要件に、位相シフト、DCM のデューティサイクルの歪み、ジッタは含まれません。
- ・ Synchronous Propagation Delays (同期伝搬遅延)
プライマリ クロック入力に対して、プライマリ出力のクロック位相関係や DCM の位相シフトが考慮され、タイミング制約の対象となるプライマリ出力の clock-to-clock や最大伝搬遅延の範囲が個別にレポートされます。
クロック入力ごとに、クロック入力からデバイスのデータ出力への最大伝搬遅延が表示されます。クロック入力からデータ出力へのパスが複数存在する場合は、ワーストケースの伝搬遅延が表示されます。データ出力とクロック入力の組み合わせごとに、伝搬遅延が 1 つずつ表示されます。

clock-to-output 伝搬遅延は、次のように表示されます。

Clock ck1_i to Pad				
-----+-----+-----+-----+-----				
	clk (edge)			
Destination Pad	to PAD			
-----+-----+-----+-----+-----				
out1_o	16.691 (R)			
-----+-----+-----+-----+-----				
Clock to Setup on destination clock ck2_i				
-----+-----+-----+-----+-----				
	Src/Dest	Src/Dest	Src/Dest	Src/Dest
Source Clock	Rise/Rise	Fall/Rise	Rise/Fall	Fall/Fall
-----+-----+-----+-----+-----				
ck2_i	12.647			


```

ckl_i      |10.241 |      |      |
-----+-----+-----+-----+

```

デバイスの入力と出力の間に組み合わせパスが存在する場合、デバイス入力からデバイス出力への最大伝搬遅延が表示されます。デバイス入力とデバイス出力の間にパスが複数存在する場合は、ワーストケースの伝搬遅延が表示されます。デザインの入力と出力の組み合わせごとに、ワーストケースの伝搬遅延が 1 つずつ表示されます。

input-to-output 伝搬遅延は、次のように表示されます。

Pad to Pad

Source Pad	Destination Pad	Delay
BSLOT0	D0S	37.534
BSLOT1	D09	37.876
BSLOT2	D10	34.627
BSLOT3	D11	37.214
CRESETN	VCASN0	51.846
CRESETN	VCASN1	51.846
CRESETN	VCASN2	49.776
CRESETN	VCASN3	52.408
CRESETN	VCASN4	52.314
CRESETN	VCASN5	52.314
CRESETN	VCASN6	51.357
CRESETN	VCASN7	52.527

- ・ User-Defined Phase Relationships (ユーザー定義の位相関係)

ユーザー定義の内部クロックの clock-to-output と最大伝搬遅延の範囲が個別にレポートされます。ユーザー定義の外部クロックでは、個別に表示されません。外部クロック別に表示されます。

レポートの凡例

データシートレポートに表示される **X**、**R**、**F** は、次を意味します。

メモ : FPGA デザインのみに適用されます。

X	不定
R	立ち上がりエッジ
F	立ち下がりエッジ

確認済みセットアップとホールドのレポート生成

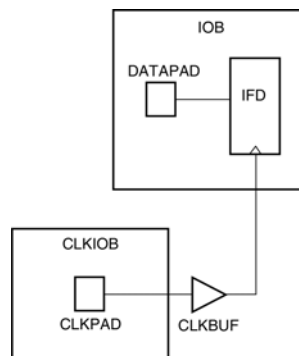
IOB 入力レジスタに指定したクロック配線リソースからクロックが供給され、指定のデバイスとスピードに対して確認済みセットアップ タイムとホールド タイムがスピード ファイルに存在する場合は、それらの値が IOB 入力レジスタのデータシートレポートで使用されます。

特定のクロック配線リソースとは、クロック IOB からクロック バッファを介してクロック配線リソースに達し、IOB レジスタに直接配線されるクロック ネットワークのことです。

確認済みセットアップ/ホールド タイムは、入力 OFFSET 制約のレポートにも使用されます。

次に、外部セットアップ タイムとホールド タイムの関係を示します。

確認済みセットアップ/ホールド要件



上図では、クロック入力コンポーネント CLKIOB のパッド CLKPAD がグローバル クロック バッファ CLKBUF を駆動し、このバッファがフリップフロップ IFD を駆動しています。フリップフロップ IFD は、コンポーネント IOB 内の DATAPAD で駆動されるデータ入力にクロックを供給します。

セッティングアップ タイム

外部セットアップ タイムは、CLKIOB 内の CLKPAD に対する IOB 内の DATAPAD のセットアップ タイムとして定義されます。特定の DATAPAD と CLKPAD のペアおよびコンフィギュレーションに対して確認済みの外部セットアップ タイムがスピード ファイルに存在する場合は、この値がタイミング レポートで使用されます。特定の DATAPAD と CLKPAD のペアに対して確認済みの外部セットアップ タイムがスピード ファイルに存在しない場合は、DATAPAD から IFD までのパスの最大遅延に IFD の最大セットアップ タイムを加え、CLKPAD から IFD までのパスの最大遅延の最小値を引いた値が、外部セットアップ タイムとしてレポートされます。

ホールドタイム

外部ホールド タイムは、CLKIOB 内の CLKPAD に対する IOB 内の DATAPAD のホールド タイムとして定義されます。特定の DATAPAD と CLKPAD のペアおよびコンフィギュレーションに対して確認済みの外部ホールド タイムがスピード ファイル内に存在する場合は、この値がタイミング レポートで使用されます。

特定の DATAPAD と CLKPAD のペアに対して確認済みの外部ホールド タイムがスピード ファイルに存在しない場合は、CLKPAD から IFD までのパスの最大遅延に IFD の最大ホールド タイムを加え、DATAPAD から IFD までのパスの最大遅延の最小値を引いた値が、外部ホールド タイムとしてレポートされます。

サマリ レポ^oート

サマリレポートには、解析されたデザインファイルの名前、デバイスのスピードとレポートレベル、サマリ情報やデザインの統計情報などの簡単な統計情報が含まれます。また、制約に対するタイミング エラーの数など、PCF ファイルの制約ごとの統計情報も表示されます。

サマリレポートは、TRACE コマンドラインに **-e** (エラー レポート) オプションまたは **-v** (詳細レポート) オプションを入力しない場合に生成されます。

ここでは、2 つのサマリレポートの例を示します。最初のサンプルは、PCF ファイルを使用しない場合の結果です。2 番目のサンプルは、PCF ファイルを使用した場合の結果です。

PCF ファイルを使用しない場合や、PCF ファイルにタイミング制約が記述されていない場合、デフォルトのパスとネットがリストされ、タイミング解析の統計情報が表示されます。デフォルトパスのリストには、順次コンポーネントのデータピンとクロックピンへの回路パス、プライマリ出力の全データピンへの回路パスが含まれます。デフォルトのネットリストには、すべてのネットが含まれます。

サマリ レポート (物理制約ファイルを指定しない場合)

ここに示すサマリ レポートの例は、次のコマンドを使用して生成したものです。

```
trce -o summary.twr rambl6_s1.ncd
```

レポートの名前は、summary.twr です。PCF は指定されておらず、rambl6_s1.ncd のあるディレクトリにも rambl6_s1.pcf という名前の PCF はありません。

```
-----
Xilinx TRACE
```

```
Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
```

```
Design file: rambl6_s1.ncd
```

```
Device,speed: xc2v250,-6
```

```
Report level: summary report
-----
```

```
WARNING:Timing - No timing constraints found, doing default enumeration.
Asterisk (*) preceding a constraint indicates it was not met.
```

```
-----
```

Constraint	Requested	Actual	Logic
			Levels
Default period analysis		2.840ns	2
Default net enumeration		0.001ns	

```
-----
```

```
All constraints were met.
```

```
Data Sheet report:
```

```
-----
All values displayed in nanoseconds (ns)
```

```
Setup/Hold to clock clk
```

```
-----+-----+-----+
```

Source Pad	Setup to clk (edge)	Hold to clk (edge)
ad0	0.263(R)	0.555(R)
ad1	0.263(R)	0.555(R)
ad10	0.263(R)	0.555(R)
ad11	0.263(R)	0.555(R)
ad12	0.263(R)	0.555(R)
ad13	0.263(R)	0.555(R)

```
-----+-----+-----+
```

```
.
```

```

.
.
-----+-----+-----+
Clock clk to Pad
-----+-----+-----+
                | clk (edge) |
Destination Pad | to PAD    |
-----+-----+-----+
d0              | 7.496(R)   |
-----+-----+-----+

Timing summary:
-----
Timing errors: 0 Score: 0

Constraints cover 20 paths, 21 nets, and 21 connections (100.0% coverage)

Design statistics:
Minimum period: 2.840ns (Maximum frequency: 352.113MHz)
Maximum combinational path delay: 6.063ns
Maximum net delay: 0.001ns
Analysis completed Wed Mar 8 14:52:30 2000
-----

```

サマリ レポート (物理制約ファイルを指定した場合)

ここに示すサマリ レポートの例は、次のコマンドを使用して生成したものです。

```
trce -o summary1.twr ramb16_s1.ncd clkperiod.pcf
```

レポート ファイル名は、summary1.twr です。このファイルに表示されているタイミング解析は、clkperiod.pcf ファイルを参照して実行されています。

```

-----
Xilinx TRACE
Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.

Design file: ramb16_s1.ncd
Physical constraint file: clkperiod.pcf
Device,speed: xc2v250,-6
Report level: summary report
-----

Asterisk (*) preceding a constraint indicates it was not met.

-----
                Constraint | Requested | Actual | Logic
                        |           |        | Levels
-----
TS01 = PERIOD TIMEGRP "clk" 10.0ns |          |        |
-----
OFFSET = IN 3.0 ns AFTER COMP

```

```

"clk" TIMEG                      | 3.000ns   | 8.593ns | 2
RP "rams"
-----
* TS02 = MAXDELAY FROM TIMEGRP
"rams" TO TI                      | 6.000ns   | 6.063ns | 2
MEGRP "pads" 6.0 ns              |           |         |
-----

1 constraint not met.

Data Sheet report:
-----
All values displayed in nanoseconds (ns)

Setup/Hold to clock clk
-----+-----+-----+
Source Pad      | Setup to    | Hold to     |
                  | clk (edge)  | clk (edge)  |
-----+-----+-----+
ad0              | 0.263(R)    | 0.555(R)    |
ad1              | 0.263(R)    | 0.555(R)    |
ad10             | 0.263(R)    | 0.555(R)    |
ad11             | 0.263(R)    | 0.555(R)    |
ad12             | 0.263(R)    | 0.555(R)    |
ad13             | 0.263(R)    | 0.555(R)    |
.
.
.
-----+-----+-----+
Clock clk to Pad
-----+-----+
                  | clk (edge)  |
Destination Pad | to PAD      |
-----+-----+
d0              | 7.496(R)    |
-----+-----+

Timing summary:
-----

Timing errors: 1 Score: 63

Constraints cover 19 paths, 0 nets, and 21 connections (100.0% coverage)

Design statistics:
Maximum path delay from/to any node: 6.063ns
Maximum input arrival time after clock: 8.593ns

Analysis completed Wed Mar 8 14:54:31 2006
-----

```

PCF ファイルにタイミング制約が含まれている場合、サマリレポートにはタイミング制約が適用されるデザイン内の接続の割合が表示されます。タイミング制約がない場合は、100% と表示されます。タイミング制約を満たしていない制約の前にはアスタリスク (*) が付きます。

エラー レポート

エラー レポートには、タイミング エラーと関連するネットとパスの遅延情報がリストされます。エラーは PCF 内の制約順に並べられ、制約内ではスラック順に並べられます。スラックとは、制約と解析された値との差で、値が負の場合はエラーを表します。制約ごとのエラー数は、コマンドライン オプションを使用して制限できます。また、エラー レポートには、PCF で定義されたタイム グループのリストや、グループごとに定義されたメンバーも表示されます。

エラー レポートの本文には、入力 PCF に含まれるタイミング制約がすべてリストされます。制約が満たされている場合、スコアに含まれたアイテムの数、タイミング エラーが検出されなかったこと、特定の制約に対する最大遅延などの重要な情報が表示されます。制約が満たされていない場合は、スコアに含まれたアイテムの数、タイミング エラー数、エラーの詳細が表示されます。

パス遅延が個々のネット遅延とコンポーネント遅延に分割される際のエラーについては、物理リソースと、物理リソースを生成した論理リソースが表示されます。

ほかのレポートと同様、最初に基本的な情報が表示されます。タイミング サマリは、常にレポートの最後に表示されます。

ここに示すエラー レポート (error.twr) の例は、次の TRACE コマンドを使用して生成したものです。

```
trce -e 3 ramb16_s1.ncd clkperiod.pcf -o error_report.twr
```

```
-----
Xilinx TRACE
```

```
Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.
```

```
trce -e 3 ramb16_s1.ncd clkperiod.pcf -o error_report.twr
```

```
Design file: ramb16_s1.ncd
```

```
Physical constraint file: clkperiod.pcf
```

```
Device,speed: xc2v250,-5 (ADVANCED 1.84 2001-05-09)
```

```
Report level: error report
-----
```

```
=====
Timing constraint: TS01 = PERIOD TIMEGRP "clk" 10.333ns ;
```

```
0 items analyzed, 0 timing errors detected.
-----
```

```
=====
Timing constraint: OFFSET = IN 3.0 ns AFTER COMP "clk" TIMEGRP "rams" ;
```

```
18 items analyzed, 0 timing errors detected.
```

```
Maximum allowable offset is 9.224ns.
```

```
-----
=====
Timing constraint: TS02 = MAXDELAY FROM TIMEGRP "rams" TO TIMEGRP "pads" 8.0 ns ;
```

```
1 item analyzed, 1 timing error detected.
Maximum delay is 8.587ns.
-----
```

```
Slack: -0.587ns (requirement - data path)
Source: RAMB16.A
Destination: d0
Requirement: 8.000ns
Data Path Delay: 8.587ns (Levels of Logic = 2)
Source Clock: CLK rising at 0.000ns
```

```
Data Path: RAMB16.A to d0
```

```
Location      Delay type  Delay(ns)  Physical Resource
                                     Logical Resource(s)
```

```
-----
RAMB16.DOA0  Tbcko      3.006      RAMB16
                                     RAMB16.A
IOB.O1       net        e 0.100    N$41
              (fanout=1)
IOB.PAD      Tioop      5.481      d0
                                     I$22
                                     d0
-----
```

```
-----
Total                8.587ns   (8.487ns logic, 0.100ns
.....route)
.....(98.8%   logic, 1.2%
.....route)
-----
```

```
1 constraint not met.
```

```
Data Sheet report:
```

```
-----
All values displayed in nanoseconds (ns)
```

```
Setup/Hold to clock clk
```

```
-----+-----+-----+
| Setup to | Hold to |
| Source Pad | clk (edge) | clk (edge) |
|-----+-----+-----+
ad0      | -0.013(R) | 0.325(R) |
ad1      | -0.013(R) | 0.325(R) |
ad10     | -0.013(R) | 0.325(R) |
ad11     | -0.013(R) | 0.325(R) |
ad12     | -0.013(R) | 0.325(R) |
```

```

ad13          | -0.013(R) | 0.325(R) |
.
.
.
-----+-----+-----+

Clock clk to Pad
-----+-----+
          | clk (edge) |
Destination Pad| to PAD |
-----+-----+
d0          | 9.563(R) |
-----+-----+

Timing summary:
-----

Timing errors: 1 Score: 587

Constraints cover 19 paths, 0 nets, and 21 connections (100.0% coverage)

Design statistics:
Maximum path delay from/to any node: 8.587ns
Maximum input arrival time after clock: 9.224ns

Analysis completed Mon Jun 03 17:47:21 2007
-----

```

詳細レポート

詳細レポートは、エラー レポートと似ていますが、デザイン内の制約が設定されたパスとネットの遅延に関する詳細情報が示されます。エレメントは PCF 内の制約順に並べられ (UCF または NCF の順序と異なる場合あり)、制約内ではスラック順に並べられます。スラックの値が負の場合、エラーがあることを表します。制約ごとのアイテム数は、コマンドライン オプションを使用して制限できます。

メモ: データシートレポートと STAMP モデルには、専用ではないクロックリソースのスキューが表示されます。この値は、通常の詳細レポートのデフォルトの周期解析では表示されません。データシートレポートと STAMP モデルにスキューが表示されるのは、スキューが外部タイミング モデルに影響するためです。

詳細レポートには、PCF で定義されたタイム グループのリストと、グループごとに定義されたメンバーも表示されます。

確認レポートの本体には、入力 PCF ファイル内の制約がリストされ、制約ごとにスコアに含まれたアイテム数と検出されたエラーの数が示されます。各アイテムはスラックの大きい順に並べられます。各アイテムの Report 行には、ネットの遅延、ネットのファンアウト、ロジックの位置 (配置されている場合)、どの程度制約を満たしているかなど、重要な情報が表示されます。

パス制約でエラーが発生した場合、どの程度制約からはずれているかが表示されます。パス遅延が個々のネット遅延とコンポーネント遅延に分割される際のエラーについては、物理リソースと、物理リソースを生成した論理リソースが示されます。

詳細レポートの例

ここに示す詳細レポート(verbose.twr)の例は、次のコマンドを使用して生成したものです。

```
trce -v 1 ramb16_s1.ncd clkperiod.pcf -o verbose_report.twr
```

Xilinx TRACE

Copyright (c) 1995-2007 Xilinx, Inc. All rights reserved.

```
trce -v 1 ramb16_s1.ncd clkperiod.pcf -o verbose_report.twr
```

```
Design file:                ramb16_s1.ncd
Physical constraint file:    clkperiod.pcf
Device,speed:               xc2v250,-5 (ADVANCED 1.84 2001-05-09)
Report level:               verbose report, limited to 1 item per constraint
```

```
=====  
Timing constraint: TS01 = PERIOD TIMEGRP "clk" 10.333ns ;  
0 items analyzed, 0 timing errors detected.  
=====
```

```
=====  
Timing constraint: OFFSET = IN 3.0 ns AFTER COMP "clk" TIMEGRP "rams" ;  
18 items analyzed, 0 timing errors detected.  
Maximum allowable offset is 9.224ns.  
=====
```

```
Slack:                6.224ns (requirement - (data path - clock path  
- clock arrival))
```

```
Source:                ssr
Destination:           RAMB16.A
Destination Clock:     CLK rising at 0.000ns
Requirement:           7.333ns
Data Path Delay:       2.085ns (Levels of Logic = 2)
Clock Path Delay:      0.976ns (Levels of Logic = 2)
```

Data Path: ssr to RAMB16.A

Location Delay type Delay(ns)

Physical Resource

Logical Resource(s)

```
-----  
IOB.I      Tiopi  
          0.551      ssr  
  
                      ssr  
                      I$36  
RAM16.SSRA  net      e 0.100      N$9  
            (fanout=1)  
RAM16.CLKA  Tbrck    1.434      RAMB16  
RAMB16.A
```

```

Total                                2.085ns (1.985ns logic, 0.100ns
                                     route)
                                     (95.2% logic, 4.8%
                                     route)

Clock Path:  clk to RAMB16.A
Location      Delay type  Delay(ns)  Physical Resource
                                     Logical Resource(s)
-----
IOB.I          Tiopi      0.551      clk
                                     clk
                                     clk/new_buffer
BUFGMUX.IO     net          e 0.100    clk/new_buffer
               (fanout=1)
BUFGMUX.O      Tgi0o        0.225      I$9
                                     I$9
RAM16.CLKA     net          e 0.100      CLK
               (fanout=1)
-----
Total                                0.976ns (0.776ns logic, 0.200ns
                                     route)
                                     (79.5% logic, 20.5%
                                     route)
-----

=====
Timing constraint: TS02 = MAXDELAY FROM TIMEGRP "rams" TO TIMEGRP "pads"
8.0 nS ;

1 item analyzed, 1 timing error detected.
Maximum delay is 8.587ns.
-----

Slack: -0.587ns (requirement - data path)
Source: RAMB16.A
Destination: d0
Requirement: 8.000ns
Data Path Delay: 8.587ns (Levels of Logic = 2)
Source Clock: CLK rising at 0.000ns
Data Path: RAMB16.A to d0
Location      Delay type      Delay(ns)  Physical Resource
                                     Logical Resource(s)
-----
RAMB16.DOA0    Tbcko              3.006      RAMB16
                                     RAMB16.A
IOB.O1         net (fanout=1)     e 0.100    N$41
IOB.PAD        Tioop              5.481      d0
                                     I$22
                                     d0
-----
Total                                8.587ns (8.487ns logic,
                                     0.100ns route)

```

(98.8% logic, 1.2% route)

1 constraint not met.

Data Sheet report:

All values displayed in nanoseconds (ns)

Setup/Hold to clock clk

Source Pad	Setup to clk (edge)	Hold to clk (edge)
ad0	-0.013(R)	0.325(R)
ad1	-0.013(R)	0.325(R)
ad10	-0.013(R)	0.325(R)
ad11	-0.013(R)	0.325(R)

.
.

.

Clock clk to Pad

Destination Pad	clk (edge) to PAD
d0	9.563(R)

Timing summary:

Timing errors: 1 Score: 587

Constraints cover 19 paths, 0 nets, and 21 connections (100.0% coverage)

Design statistics:

Maximum path delay from/to any node: 8.587ns

Maximum input arrival time after clock: 9.224ns

Analysis completed Mon Jun 03 17:57:24 2007

OFFSET 制約

OFFSET 制約は、初期タイム 0ns に対して、入力および出力のタイミング制約を定義します。

関連の PERIOD 制約は、初期クロック エッジを定義します。PERIOD 制約に HIGH を設定した場合、初期クロック エッジは立ち上がりエッジになり、LOW を設定した場合は立ち下がりエッジになります。この設定は、OFFSET 制約で HIGH/LOW キーワードを使用すると変更できます。OFFSET 制約は、セットアップ タイムおよびホールド タイムをチェックします。制約の詳細は、『[制約ガイド](#)』を参照してください。

OFFSET IN 制約

このセクションでは、タイミング解析レポートの「Timing Constraints」に表示される OFFSET IN 制約について説明します。OFFSET IN 制約の情報は、次の項目に分類されます。

- ・ OFFSET IN ヘッダ
- ・ OFFSET IN パスの詳細
- ・ OFFSET IN 詳細パス データ
- ・ OFFSET IN 詳細パス クロック パス
- ・ 位相シフトのあるクロック付き OFFSET IN

OFFSET IN ヘッダ

ヘッダには、タイミング制約、解析するアイテム数、検出されたタイミング エラー数が表示されます。解析するアイテムとタイミング エラーについては、「PERIOD ヘッダ」を参照してください。

```
=====
```

```
Timing constraint: OFFSET = IN 4 nS BEFORE COMP "wclk_in" ;
```

```
113 items analyzed, 30 timing errors detected.
```

```
Minimum allowable offset is 4.468ns
```

```
-----
```

最小許容オフセット値は 4.468ns です。ここで示す制約は OFFSET IN BEFORE なので、クロックの初期エッジの 4.468ns 前までにデータが有効になる必要があります。PERIOD 制約に HIGH を設定しているため、クロックの初期エッジは立ち上がりエッジです。

OFFSET IN パスの詳細

このパスは、0.468ns の差でタイミング要件を満たしていません。Slack 行のかっこ内の式は、スラックの算出方法を示します。この式からわかるように、データ遅延が増加するとセットアップ時間が長くなり、クロック遅延が増加するとセットアップ時間が短くなります。クロック到着時間も考慮されます。例では、クロック到着時間は 0.0ns であるため、スラックには影響しません。

=====

Slack: -0.468ns (requirement - (data path - clock path - clock arrival + uncertainty))

Source: wr_enl (PAD)

Destination: wr_addr[2] (FF)

Destination Clock: wclk rising at 0.000ns

Requirement: 4.000ns

Data Path Delay: 3.983ns (Levels of Logic = 2)

Clock Path Delay: -0.485ns (Levels of Logic = 3)

Clock Uncertainty: 0.000ns

Data Path: wr_enl to wr_addr[2]

OFFSET IN 詳細パス データ

最初のセクションはデータ パスです。この例では、パスの始点は IOB で、LUT を通過して、デスティネーション フリップフロップであるクロック イネーブル ピンに到達します。

Data Path: wr_enl to wr_addr[2]

Location	Delay type	Delay(ns)	Logical Resource(s)

C4.I	Tiopi	0.825	wr_enl
			wr_enl_ibuf
SLICE_X2Y9.G3	net (fanout=39)	1.887	wr_enl_c
SLICE_X2Y9.Y	Tilo	0.439	G_82
SLICE_X3Y11.CE	net (fanout=1)	0.592	G_82
SLICE_X3Y11.CLK	Tceck	0.240	wr_addr[2]

Total		3.983ns	(1.504ns logic, 2.479ns route)
			37.8% logic, 62.2% route)

OFFSET IN 詳細パス クロック パス

2 番目のセクションはクロック パスです。この例では、クロックが IOB から供給され、DCM を通過して CLK0 から出力され、グローバル バッファ (BUFGHUX) に到達します。フリップフロップのクロック ピンが終点となります。

Tdcmino は、算出された遅延です。

Clock Path: wclk_in to wr_addr[2]

Location	Delay type	Delay(ns)	Logical Resource(s)

D7.I	Tiopi	0.825	wclk_in
			write_dcm/IBUFG
DCM_X0Y1.CLKIN	net (fanout=1)	0.798	write_dcm/IBUFG
DCM_X0Y1.CLK0	Tdcmino	-4.297	write_dcm/CLKDLL
BUFGMUX3P.I0	net (fanout=1)	0.852	write_dcm/CLK0
BUFGMUX3P.O	Tgi0o	0.589	write_dcm/BUFG
SLICE_X3Y11.CLK	net (fanout=41)	0.748	wclk

Total -0.485ns (-2.883ns logic, 2.398ns route)

位相シフトのあるクロック付き OFFSET IN

この例では、クロックは DCM の CLK90 出力で、クロック到着時間は 2.5ns です。rclk90 は 2.5ns で立ち上がります。この数字は rclk_in の PERIOD (10ns) から算出されます。この 2.5ns はスラックに影響します。クロックの遅延が 2.5ns なので、データがデスティネーションに到達するまでに、2.5ns 長くなります。

パスでクロックの立ち下がりエッジが使用された場合、デスティネーション クロックは 7.5ns (位相で 2.5ns、クロック エッジで 5.0ns) で立ち下がります。最小許容オフセット値は、クロックの初期エッジが基準となっているため、負の値になることもあります。最小許容オフセット値が負の場合、データがクロックの初期エッジの後に到着することを表します。これは、初期エッジが立ち上がりと定義されていて、デスティネーション クロックが立ち下がりである場合や、クロックが位相シフトされている場合に発生します。

=====

Timing constraint: OFFSET = IN 4 nS BEFORE COMP "rclk_in" ;

2 items analyzed, 0 timing errors detected.

Minimum allowable offset is 1.316ns.

Slack: 2.684ns (requirement - (data path - clock path - clock arrival + uncertainty))

Source: wclk_in (PAD)

Destination: ffl_reg (FF)

Destination Clock: rclk_90 rising at 2.500ns

Requirement: 4.000ns

Data Path Delay: 3.183ns (Levels of Logic = 5)

Clock Path Delay: -0.633ns (Levels of Logic = 3)

Clock Uncertainty: 0.000ns

Data Path: wclk_in to ffl_reg

Location	Delay type	Delay(ns)	Logical Resource(s)

D7.I	Tiopi	0.825	wclk_in
			write_dcm/IBUFG
DCM_X0Y1.CLKIN	net (fanout=1)	0.798	write_dcm/IBUFG
DCM_X0Y1.CLK0	Tdcmino	-4.297	write_dcm/CLKDLL
BUFGMUX3P.I0	net (fanout=1)	0.852	write_dcm/CLK0
BUFGMUX3P.O	Tgi0o	0.589	write_dcm/BUFG
SLICE_X2Y11.G3	net (fanout=41)	1.884	wclk
SLICE_X2Y11.Y	Tilo	0.439	un1_full_st
SLICE_X2Y11.F3	net (fanout=1)	0.035	un1_full_st
SLICE_X2Y11.X	Tilo	0.439	full_st_i_0.G_4.G_4.G_4
K4.O1	net (fanout=3)	1.230	G_4
K4.OTCLK1	Tioock	0.389	ffl_reg

Total 3.183ns (-1.616ns logic, 4.799ns route)			

Clock Path: rclk_in to ffl_reg

Location	Delay type	Delay(ns)	Logical Resource(s)

A8.I	Tiopi	0.825	rclk_in
			read_ibufg
CM_X1Y1.CLKIN	net (fanout=1)	0.798	rclk_ibufg
CM_X1Y1.CLK90	Tdcmino	-4.290	read_dcm
UFGMUX5P.IO	net (fanout=1)	0.852	rclk_90_dcm
BUFGMUX5P.O	Tgi0o	0.589	read90_bufg
4.OTCLK1	net (fanout=2)	0.593	rclk_90

Total		-0.633ns	(-2.876ns logic, 2.243ns route)

OFFSET OUT 制約

このセクションでは、タイミング解析レポートの「Timing Constraints」に表示される OFFSET OUT 制約について説明します。OFFSET OUT 制約の情報は、次の項目に分類されます。

- ・ OFFSET OUT ヘッダ
- ・ OFFSET OUT パスの詳細
- ・ OFFSET OUT 詳細クロック パス
- ・ OFFSET OUT 詳細パス データ

OFFSET OUT ヘッダ

ヘッダには、タイミング制約、解析するアイテム数、検出されたタイミング エラー数が表示されます。解析するアイテムとタイミング エラーについては、「PERIOD ヘッダ」を参照してください。

```
=====
Timing constraint: OFFSET = OUT 10 nS AFTER COMP "rclk_in"    ;
50 items analyzed, 0 timing errors detected.
Minimum allowable offset is 9.835ns.
-----
```

OFFSET OUT パスの詳細

このパスは、0.533ns の差でタイミング制約を満たしています。Slack 行のかっこ内の式は、スラックの算出方法を示します。データ遅延およびクロック遅延が増加すると、clock-to-out タイムが長くなります。クロック到着時間も考慮されます。この例では、クロック到着時間は 0.0ns であるため、スラックには影響しません。

クロック エッジが異なる時間に発生する場合、その値も clock-to-out タイムに加算されます。この例で、クロックが 5.0ns で立ち下がる場合、対応する PERIOD 制約の初期エッジが High であるため、スラックに 5.0ns が加算されます。

メモ： クロックが 5.0ns で立ち下がるように指定するには、PERIOD 制約を使用して PERIOD 10 HIGH 5 のように定義します。

```
=====
Slack: 0.533ns (requirement - (clock arrival + clock
path + data path + uncertainty))
```

```
Source:                wr_addr[2] (FF)

Destination:           efl (PAD)

Source Clock:           wclk rising at 0.000ns

Requirement:           10.000ns

Data Path Delay:        9.952ns (Levels of Logic = 4)

Clock Path Delay:       -0.485ns (Levels of Logic = 3)

Clock Uncertainty:      0.000ns
```

OFFSET OUT 詳細クロック パス

この例では、OFFSET OUT の設定されたパスがクロックで始まるため、クロック パスが最初に示されています。クロックは IOB から供給され、DCM を通過して CLK0 から出力され、グローバル バッファに到達します。フリップフロップのクロック ピンが終点となります。

Tdcmينو は、算出された遅延です。

Clock Path: rclk_in to rd_addr[2]

Location	Delay type	Delay(ns)	Logical Resource(s)

A8.I	Tiopi	0.825	rclk_in
			read_ibufg
DCM_X1Y1.CLKIN	net (fanout=1)	0.798	rclk_ibufg
DCM_X1Y1.CLK0	Tdcmينو	-4.290	read_dcm
BUFGMUX7P.I0	net (fanout=1)	0.852	rclk_dcm
BUFGMUX7P.O	Tgi0o	0.589	read_bufg
SLICE_X4Y10.CLK	net (fanout=4)	0.738	rclk

Total		-0.488ns (-2.876ns logic, 2.388ns route)	

OFFSET OUT 詳細パス データ

2 番目のセクションはデータ パスです。パスの始点はフリップフロップで、3 つの LUT を通過して、IOB が終点となります。

Data Path: rd_addr[2] to efl

Location	Delay type	Delay(ns)	Logical Resource(s)

SLICE_X4Y10.YQ	Tcko	0.568	rd_addr[2]
SLICE_X2Y10.F4	net (fanout=40)	0.681	rd_addr[2]
SLICE_X2Y10.X	Tilo	0.439	G_59
SLICE_X2Y10.G1	net (fanout=1)	0.286	G_59
SLICE_X2Y10.Y	Tilo	0.439	N_44_i
SLICE_X0Y0.F2	net (fanout=3)	1.348	N_44_i
SLICE_X0Y0.X	Tilo	0.439	empty_st_i_0
M4.O1	net (fanout=2)	0.474	empty_st_i_0
M4.PAD	Tioop	5.649	efl_obuf
			efl

Total		10.323ns (7.534ns logic, 2.789ns route)	
		(73.0% logic, 27.0% route)	

PERIOD 制約

PERIOD 制約は、指定したクロック信号名で制御される順次エレメント間のパスに適用されません。制約の詳細は、『[制約ガイド](#)』を参照してください。

このセクションでは、タイミング解析レポートの「Timing Constraints」に表示される PERIOD 制約について説明します。PERIOD 制約の情報は、次の項目に分類されます。

- ・ PERIOD ヘッダ
- ・ PERIOD パス
- ・ PERIOD パスの詳細
- ・ PHASE を使用した PERIOD

PERIOD ヘッダ

次の例は、変換 (NGDBuild) の段階で生成された制約を示します。TS_write_dcm_CLK0 という新しいタイムスペック (制約) の名前が作成されています。write_dcm は DCM のインスタンス名で、CLK0 は出力クロックです。PERIOD 制約は、タイムグループ write_dcm_CLK0 に設定され、TS_wclk に関連付けられています。この例では、元の制約に 1 が掛け合わされており、位相オフセットもないため、PERIOD 制約は元の制約と同じです。つまり、TS_wclk の PERIOD 制約が 12ns と定義されているので、TS_write_dcm_CLK0 の PERIOD 制約も 12ns になります。

ここでは、296 のアイテム (パスまたはネット) が解析されました。この制約はパスに関連しているため、この場合のアイテムは 1 つのパスを指します。同じエンドポイントに到達するパスが 2 つある場合は、2 つのパスとして数えられます。MAXDELAY 制約またはネットに関連付けられた制約の場合は、アイテムはネットを指します。タイミング エラー数は、タイミング要件を満たしていないエンドポイント数とホールド タイムに違反したエンドポイント数です。ホールド タイム違反の数が示されていない場合、その制約に対してホールド タイム違反はありません。同じエンドポイントに到達する複数のパスがある場合、タイミング エラーは 1 つと見なされます。この場合、パスごとに詳細がレポートされます。

次の行には、この制約の最小周期が表示されます。これはクロックの速度を表します。

```
=====
Timing constraint: TS_write_dcm_CLK0 = PERIOD TIMEGRP "write_dcm_CLK0" TS_wclk *
1.000000 HIGH
50.000 % ;
296 items analyzed, 0 timing errors detected.
Minimum period is 3.825ns.
-----
```

PERIOD パス

解析されたタイミング制約の各パスに関する詳細情報は、詳細パス セクションに表示されます。最も重要な情報は、パスがタイミング制約を満たしているかどうかです。この情報は、最初の行に「Slack」として記述されます。スラックの値が正の場合、スラック値の差でパスがタイミング制約を満たしており、負の場合はスラック値の差でパスがタイミング制約を満たしていないことになります。カッコ内の式には、スラックの計算方法が示されています。「Requirement」(要件) は、タイミング制約の値です。この例の場合、元のタイムスペック TS_wclk の値 12ns となります。データ パス遅延は 3.811ns で、クロック スキューは -0.014ns です ($12 - (3.811 - 0.014) = 8.203$)。詳細パスはスラックを基準に並び替えられます。スラック値が最小のパスが「Timing Constraints」セクションの最初に記述されます。

「Source」は、パスの始点です。ソース名のすぐ後には、コンポーネントのタイプが示されます (例では FF)。FF グループには SRL16 も含まれます。ほかのコンポーネントとして、RAM (分散 RAM、ブロック RAM)、PAD、LATCH、HSIO (ギガビットトランシーバのような高速 I/O)、MULT (乗算器)、CPU (PowerPC® プロセッサ) などがあります。Timing Analyzer では、FPGA デザインのソースはクロスプローブできるようハイパーリンクになっています。

「Destination」は、パスの終点です。デスティネーション コンポーネントのタイプおよびクロスプローブについては、「Source」の説明を参照してください。

「Requirement」は、タイミング制約とクロック エッジの時間を基に算出された値です。このパスのソース クロックとデスティネーション クロックは同じで、要件全体が使用されます。ソース クロックまたはデスティネーション クロックが関連クロックである場合、新しい要件はクロック エッジの時間差になります。ソースとデスティネーションのクロックが同じで、異なるエッジを使用する場合、新しい要件は元の PERIOD 制約の 1/2 になります。

「Data Path Delay」は、ソースからデスティネーションまでのデータ パスの遅延です。「Level of Logic」(ロジックのレベル数) は、ソースとデスティネーションの間にある LUT の数で、デスティネーションの clock-to-out およびセットアップは含まれません。デスティネーションと同じスライス内に LUT がある場合は、それがロジックのレベルとして数えられます。ソースとデスティネーションの間にロジックがない場合は、ロジックのレベル数は 0 になります。

「Clock Skew」は、クロック信号がソース フリップフロップに到達する時間と、デスティネーション フリップフロップに到達する時間の差を示します。クロック スキューがチェックされない場合は、レポートされません。

「Source Clock」および「Destination Clock」には、ソースおよびデスティネーションのクロック名が表示されます。クロック エッジが立ち上がりか立ち下がりか、およびエッジが発生する時間に関する情報が含まれます。DCM/DLL でクロック位相がシフトされている場合は、クロックの到着時間が表示されます。これには、固定位相シフトまたは可変位相シフトを使用した、コース グレイン位相 (CLK90、CLK180、または CLK270) およびファイン グレイン位相が含まれます。

OFFSET 制約の「Clock Uncertainty」は、同じクロックの PERIOD 制約のものと異なる場合があります。OFFSET 制約では 1 つのクロック エッジのみを使用しますが、PERIOD 制約ではソース レジスタとデスティネーション レジスタの両方のクロック誤差が考慮されます。

```
-----
Slack: 8.175ns (requirement - (data path - clock skew + uncertainty))
Source: wr_addr[0] (FF)
Destination: fifo_ram/BU5/SP (RAM)
Requirement: 12.000ns
Data Path Delay: 3.811ns (Levels of Logic = 1)
clock skew: -0.014ns
Source Clock: wclk rising at 0.000ns
Destination Clock: wclk rising at 12.000ns
Clock Uncertainty: 0.000ns
-----
```

PERIOD パスの詳細

最初の行は、制約向上ウィザード (Constraint Improvement Wizard) へのリンクです。パスが制約を満たしていない場合、このウィザードに問題を解決するヒントが表示されます。「Data Path」セクションには、パス内のコンポーネントとネットの遅延がすべて表示されます。最初の列「Location」は、FPGA 内のコンポーネントの位置を示します。デフォルトでは表示されません。次の列「Delay Type」には、遅延のタイプが表示されます。ネットの場合は、ファンアウトも表示されます。遅延の名前はデータシートと対応しています。遅延名をクリックすると、説明ページに遅延名についての詳細が表示されます。

次の列には、「Physical Resource」(物理的なリソース) および「Logical Resource」(論理的なリソース) が表示されます。物理名は、マップで生成されたコンポーネント名です。デフォルトでは表示されません。論理名は、合成プログラムまたは回路図入力プログラムにより作成される名前、デザイン ファイル内での名前となります。

このセクションの最後には、遅延の合計時間と、ロジックおよび配線での割合が示されます。この情報は、タイミングが満たされていない場合のデバッグで有効です。タイミング問題の修正方法については、タイミング向上ウィザードの指示に従ってください。

```
-----
Constraints Improvement Wizard
Data Path: wr_addr[0] to fifo_ram/BU5/SP
Location Delay type Delay(ns) Logical Resource(s)
-----
SLICE_X2Y4.YQ Tcko 0.568 wr_addr[0]
SLICE_X6Y8.WF1 net (fanout=112) 2.721 wr_addr[0]
SLICE_X6Y8.CLK Tas 0.522 fifo_ram/BU5/SP
-----
Total 3.811ns (1.090ns logic, 2.721ns route)
(28.6% logic, 71.4% route)
-----
```

PHASE を使用した PERIOD

位相がシフトされたクロックに設定した PERIOD 制約の例を表示します。この制約は、変換 (NGDBuild) の段階で TS_rclk 制約に 2.5ns の位相を追加することにより生成されています。クロックは DCM の CLK90 出力です。PERIOD 制約は 10ns であるため、CLK90 出力からのクロック位相は 2.5ns (元の制約の 1/4) です。これは PHASE キーワードを使用して定義できます。

```
Timing constraint: TS_rclk_90_dcm = PERIOD TIMEGRP "rclk_90_dcm"
    TS_rclk * 1.000000 PHASE + 2.500
nS HIGH 50.000 % ;
6 items analyzed, 1 timing error detected.
Minimum period is 21.484ns.
-----
```

位相がシフトされた PERIOD パス

ここに示す例は、PHASE を使用しない PERIOD 制約に類似しています。このパスの違いは、ソース クロックとデスティネーション クロックです。デスティネーション クロックにより、パスに使用される PERIOD 制約が定義されます。デスティネーション クロックは rclk_90 であるため、パスは TS_rclk PERIOD ではなく、TS_rclk90_dcm PERIOD に含まれます。

「Requirement」が 10ns ではなく、2.5ns になっていることに注意してください。これはソース クロック (0.0ns で立ち上がり) とデスティネーション クロック (2.5ns で立ち上がり) の時間差です。

スラックの値が負であるため、このパスは制約を満たしていません。制約を満たしていないパスは、階層レポート ブラウザ内で赤色で表示されます。

```
-----
Slack:    -2.871ns (requirement - (data path - clock skew + uncertainty))
Source: rd_addr[1] (FF)
Destination: ffl_reg (FF)
Requirement: 2.500ns
Data Path Delay: 5.224ns (Levels of Logic = 2)
Clock    Skew: -0.147ns
Source Clock: rclk rising at 0.000ns
Destination Clock: rclk_90 rising at 2.500ns
Clock Uncertainty: 0.000ns
Data Path: rd_addr[1] to ffl_reg
Location   Delay type Delay(ns) Logical Resource(s)
-----
SLICE_X4Y19.XQ Tcko 0.568 rd_addr[1]
SLICE_X2Y9.F3 net (fanout=40) 1.700 rd_addr[1]
SLICE_X2Y9.X Tilo 0.439 full_st_i_0.G_4.G_4.G_3_10
SLICE_X2Y11.F2 net (fanout=1) 0.459 G_3_10
SLICE_X2Y11.X Tilo 0.439 full_st_i_0.G_4.G_4.G_4
K4.O1 net (fanout=3) 1.230 G_4
K4.OTCLK1 Tioock 0.389 ffl_reg
-----
Total 5.224ns (1.835ns logic, 3.389ns route)
(35.1% logic, 64.9% route)
-----
```

最小周期統計

FROM TO 制約で指定されたパスはタイミングで考慮されますが、最小周期の値には複数サイクル制約で許容される時間は含まれません。

次に、最小周期統計の例を示します。この値は、すべてのパスが 1 サイクルであることを前提に算出されます。

```
-----  
Design statistics:  
Minimum period: 30.008ns (Maximum frequency: 33.324MHz)  
Maximum combinational path delay: 42.187ns  
Maximum path delay from/to any node: 31.026ns  
Minimum input arrival time before clock: 12.680ns  
Maximum output required time before clock: 43.970ns  
-----
```

TRACE の停止

TRACE を停止するには、**Ctrl + C** キー (Linux) または **Ctrl + Break** キー (Windows) を押します。Linux の場合、**Ctrl + C** キーを押すときに TRACE を実行しているウィンドウがアクティブになっていることを確認してください。停止を確認するメッセージが表示されます。TRACE レポート ファイルや物理制約ファイルなど、TRACE が停止しても残るファイルはありますが、処理が終了していない状態を表しているため、破棄してもかまいません。

Speedprint

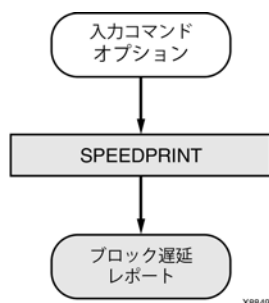
この章では、Speedprint について説明します。次のセクションから構成されています。

- ・ [Speedprint の概要](#)
- ・ [Speedprint の構文](#)
- ・ [Speedprint のオプション](#)

Speedprint の概要

Speedprint は、ブロック遅延値に関する一般的な情報を提供するコマンドライン ツールです。特定のパスのブロック遅延は、TRACE レポートでそのパスの情報を確認してください。

Speedprint のフロー



Speedprint のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6

Speedprint のファイル タイプ

Speedprint にファイル タイプはありません。レポート出力は、標準出力 (**std out**) に表示されます。Speedprint の出力を保存するには、コマンドラインに次の例のように入力します。

```
speedprint 5vlx30 > report1.txt
```

Speedprint のレポート例 1 : speedprint 5vlx30

次のレポートは、通常のデバイスに対してオプションを使用せずに生成したもので、デフォルトのスピード グレードでワーストケースの温度と電圧の場合の値を示します。

```
Family virtex5, Device xc5vlx30

Block delay report for device: 5vlx30, speed grade: -3, Stepping Level: 0

Version identification for speed file is: PRODUCTION 1.58_a 2007-10-05

Speed grades available for this device: -MIN -3 -2 -1

This report prepared for speed grade of -3 using a junction
temperature of 85.000000 degrees C and a supply voltage of 0.950000 volts.

Operating condition ranges for this device:
Voltage 0.950000 to 1.050000 volts
Temperature 0.000000 to 85.000000 degrees Celsius

This speed grade does not support reporting delays for specific
voltage and temperature conditions.

Default System Jitter for this device is 50.00 picoseconds.

Setup/Hold Calculation Support

Delay Adjustment Factors:

    Note: This speed file does not contain any delay adjustment factors.
    The following list of packages have individual package flight times for each pin on the device:

ff324
ff676

No external setup and hold delays

This report is intended to present the effect of different speed
grades and voltage/temperature adjustments on block delays.
For specific situations use the Timing Analyzer report instead.

Delays are reported in picoseconds.

When a block is placed in a site normally used for another type of block,
for example, an IOB placed in a Clock IOB site, small variations in delay
may occur which are not included in this report.

NOTE: The delay name is followed by a pair of values representing a relative minimum
delay value and its corresponding maximum value. If a range of values exists for
a delay name, then the smallest pair and the largest pair are reported.

BUFG
Tbgcko_O 173.00 / 188.00

BUFGCTRL
Tbccck_CE 265.00 / 265.00
Tbccck_S 265.00 / 265.00
Tbccck_CE 0.00 / 0.00
Tbccck_S 0.00 / 0.00
Tbccck_O 173.00 / 188.00

BUFIO
Tbufiock_O 594.00 / 1080.00

.
.
```

Speedprint のレポート例 2 : speedprint -help

次のレポートは、-help オプションを使用してオプションに関する情報を表示したものです。

```
Usage: speedprint [-s <sgrade>] [-t <temp>] [-v <volt>] [-stepping <level>] [-intstyle <style>] <device>
You must specify a device whose delays you want to see.
For example, speedprint 2v250e.
Options and arguments are:
-s <sgrade> Desired speed grade. Default is used if not specified.
Use -s min for the absolute minimum delay values.
-t <temp> Junction temperature of device. Default is worst case.
-v <volts> Supply voltage. Default is worst case.
-stepping <level> Stepping Level. Default is production shipping.
-intstyle <style> Integration flow. ise|flow|silent.
```

Speedprint のレポート例 3 : speedprint xa3s200a -s 4Q -v 1.2 -t 75

Family aspartan3a, Device xa3s200a

Block delay report for device: xa3s200a, speed grade: -4Q

Version identification for speed file is: PRODUCTION 1.39_a 2007-10-05

Speed grades available for this device: -MIN -4 -4Q

This report prepared for speed grade of -4Q using a junction temperature of 75.000000 degrees C and a supply voltage of 1.200000 volts.

Operating condition ranges for this device:
Voltage 1.140000 to 1.260000 volts
Temperature -40.000000 to 125.000000 degrees Celsius

This speed grade supports reporting delays for specific voltage and temperature conditions over the above operating condition ranges.

Setup/Hold Calculation Support

Delay Adjustment Factors:

Note: This speed file does not contain any delay adjustment factors.

No external setup and hold delays

This report is intended to present the effect of different speed grades and voltage/temperature adjustments on block delays. For specific situations use the Timing Analyzer report instead.

Delays are reported in picoseconds.

When a block is placed in a site normally used for another type of block, for example, an IOB placed in a Clock IOB site, small variations in delay may occur which are not included in this report.

NOTE: The delay name is followed by a pair of values representing a relative minimum delay value and its corresponding maximum value. If a range of values exists for a delay name, then the smallest pair and the largest pair are reported.

BUFGMUX

Tgi0o 195.59 / 217.32
Tgi0s 0.00 / 0.00
Tgi1o 195.59 / 217.32
Tgi1s 0.00 / 0.00
Tgsi0 613.60 / 613.60
Tgsi1 613.60 / 613.60

DCM

Tdmck_PSEN 16.72 / 16.72
Tdmck_PSINCDEC 16.72 / 16.72
Tdmckc_PSEN 0.00 / 0.00
Tdmckc_PSINCDEC 0.00 / 0.00
Tdmcko_CLK 14.16 / 16.72
Tdmcko_CLK2X 14.16 / 16.72
Tdmcko_CLKDV 14.16 / 16.72
Tdmcko_CLKFX 14.16 / 16.72
Tdmcko_CONCUR 14.16 / 16.72
Tdmcko_LOCKED 14.16 / 16.72
Tdmcko_PSDONE 14.16 / 16.72
Tdmcko_STATUS 14.16 / 16.72
.
.
.

Speedprint の構文

Speedprint の構文は、次のとおりです。

speedprint [*options*] [*device_name*]

options : 「Speedprint のオプション」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

device_name : 情報を表示するデバイスを指定します。

Speedprint コマンドの例

コマンド	説明
speedprint	コマンドの使用法を表示します。
speedprint 2v80	デフォルトのスピード グレードでの遅延を表示します。
speedprint -s 5 2v80	スピード グレード -5 のブロック遅延を表示します。
speedprint -2v50e -v 1.9 -t 40	電圧 1.9V、温度 40°C、デフォルトのスピード グレードでの遅延を表示します。
speedprint v50e -min	最小スピード グレードでの遅延を表示します。

Speedprint のオプション

このセクションでは、Speedprint のコマンドライン オプションについて説明します。

- ・ **-intstyle** (統合スタイル)
- ・ **-min** (最小スピード データの表示)
- ・ **-s** (スピード グレード)
- ・ **-stepping** (ステッピング)
- ・ **-t** (温度の指定)
- ・ **-v** (電圧の指定)

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle ise|xflow|silent

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-min (最小スピード データの表示)

デバイスの最小スピード データを表示します。**-s** オプションと同時に指定した場合、**-min** オプションが優先されます。

構文

-min

-s (スピード グレード)

指定したスピード グレードのデータを表示します。**-s** オプションを使用しない場合は、デフォルト (最速のスピード グレード) の遅延データが表示されます。

構文

-s [*speed_grade*]

注意 : スピード グレードの値の前にマイナス記号を付けないでください。たとえば、「**speedprint 5vlx30 -s 3**」は正しい入力ですが、「**speedprint 5vlx30 -s -3**」という入力は不正です。

-stepping (ステッピング)

指定したステッピングのデバイスの遅延を表示します。デフォルトのステッピングは、デバイスによって異なります。このオプションを指定しない場合、デフォルトのステッピングの遅延が表示されます。ステッピングが遅延に影響する場合があります。

構文

-stepping *stepping_value*

例

```
speedprint -stepping 0
speedprint -stepping ES
```

-t (温度の指定)

チップの動作温度を摂氏で指定します。このオプションを指定しない場合、ワーストケースの温度が使用されます。

構文

-t *temperature*

例

```
speedprint -t 85
speedprint -t -20
```

-v (電圧の指定)

デバイスの動作電圧をボルト単位で指定します。このオプションを指定しない場合、ワーストケースの電圧が使用されます。

構文

-v *voltage*

例

```
speedprint -v 1.2  
speedprint -v 5
```


BitGen

この章では、BitGen について説明します。次のセクションが含まれています。

- ・ [BitGen の概要](#)
- ・ [BitGen の構文](#)
- ・ [BitGen のオプション](#)

BitGen の概要

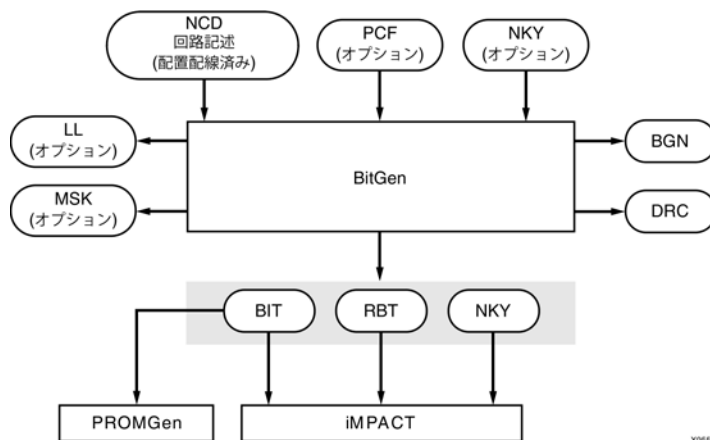
BitGen は、ザイリンクス デバイスのコンフィギュレーションに使用するビットストリームを生成するコマンドライン ツールです。デザインの配線が完了した後、BitGen で生成されたファイルを使用してデバイスをコンフィギュレーションします。BitGen は、完全に配線が終了した NCD (Native Circuit Description) ファイルを入力として読み込み、コンフィギュレーションビットストリームファイル (BIT) を出力します。BIT ファイルは、拡張子が `.bit` のバイナリファイルです。

BIT ファイルには、NCD ファイルからのコンフィギュレーション情報が含まれます。NCD ファイルは、FPGA デバイスの内部ロジックやインターコネクトの定義、ターゲット デバイスに関連するその他のファイルからのデバイス特有の情報が含まれています。BIT ファイル内のバイナリ データは、FPGA デバイスのメモリ セルにダウンロードするか、PROM ファイルを生成する ([「PROMGen」の章](#)を参照) のに使用します。

メモ： NGDBuild の入力として BMM ファイルを指定した場合、BitGen でこの BMM ファイルが BRAM ロケーション (PAR で指定) にアップデートされ、アップデートされたバック アノテート `*_bd.bmm` ファイルが生成されます。

`*_bd.bmm` ファイルは、指定の NCD に BMM 情報が含まれており、**-bd** オプションで ELF/MEM ファイルが指定されている場合に生成されます。

デザイン フロー



BitGen のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6

BitGen の入力ファイル

次に、BitGen の入力ファイルを示します。

ファイル	データ型	短縮形	デバイス	拡張子	説明
Native Circuit Description	入力	NCD ファイル	FPGA	.ncd	ターゲット デバイスにマップ、配置配線されたデザインの物理的な記述。完全に配線が完了した NCD を使用する必要があります。
物理制約ファイル	入力	PCF	FPGA	.pcf	ユーザーが変更可能な ASCII 形式の物理制約ファイル (オプション)。
暗号キー	暗号化	NKY	FPGA	.nky	暗号キー ファイル (オプション)。暗号化については、 http://japan.xilinx.com/products/ipcenter/DES.htm を参照してください。

BitGen の出力ファイル

次に、BitGen の出力ファイルを示します。

拡張子	形式	内容	メモ	生成条件
.bgn	ASCII	コマンドライン オプション、エラー、警告などの BitGen 実行のログ情報	なし	常時生成
.bin	バイナリ	コンフィギュレーション データのみ	BIT ファイルのようなヘッダは含まれない	bitgen -g Binary:Yes を指定した場合

拡張子	形式	内容	メモ	生成条件
.bit	バイナリ	著作権ヘッダ情報、コンフィギュレーション データ	PROMGen や iMPACT などのザイリンクス ツールへの入力ファイルとして使用	bitgen -j を指定した場合を除き、常時生成
.drc	ASCII	エラー、警告を含むデザインルール チェック (DRC) のログ情報	なし	bitgen -d を指定した場合を除き、常時生成
.isc	ASCII	IEEE1532 フォーマットのコンフィギュレーション データ	一部のアーキテクチャでは IEEE1532 フォーマットはサポートされない	bitgen -g IEEE1532:Yes を指定した場合
.ll	ASCII	リードバックでキャプチャされるデザイン内の各ノードの情報 (リードバック ストリームでの絶対ビット位置、フレーム アドレス、フレーム オフセット、使用されるロジック リソース、デザイン内のコンポーネント名)	なし	bitgen -l を指定した場合
.msd	ASCII	パッド ワードおよびフレームなどの検証用マスク情報のみ	コマンドは含まれない	bitgen -g Readback を指定した場合
.msk	バイナリ	BIT ファイルと同じコンフィギュレーション コマンド、コンフィギュレーション データの位置を示すマスク データ	マスク ビットが 0 の場合はビットをビットストリームのデータと比較、マスク ビットが 1 の場合はビットは検証されない	bitgen -m を指定した場合
.nky	ASCII	暗号化を使用する場合の暗号キー情報	iMPACT でキーをプログラムする際の入力として使用 (デバイスのコンフィギュレーションには使用不可)	bitgen -g Encrypt:Yes を指定した場合
<outname>_key.isc	ASCII	IEEE1532 フォーマットで暗号キーをプログラムするためのデータ	一部のアーキテクチャでは IEEE1532 フォーマットはサポートされない	bitgen -g IEEE1532:Yes および bitgen -g Encrypt:Yes を指定した場合
.rba	ASCII	リードバック コマンド (コンフィギュレーション コマンドではない)、コンフィギュレーション データが通常保存されている場所に含まれるリードバック データ	なし	bitgen -g Readback を指定する際に -b オプションを使用
.rbb	バイナリ	リードバック コマンド (コンフィギュレーション コマンドではない)、コンフィギュレーション データが通常保存されている場所に含まれるリードバック データ	RBA ファイルと同じ内容のバイナリ ファイル	bitgen -g Readback を指定した場合

拡張子	形式	内容	メモ	生成条件
.rbd	ASCII	パッドワードおよびフレームなどの予期されるリードバックデータのみ (コマンドは含まれない)	なし	bitgen -g Readback を指定した場合
.rbt	ASCII	BIT ファイルと同じ情報	BIT ファイルと同じ内容の ASCII 形式のファイル	bitgen -b を指定した場合

暗号化については、<http://japan.xilinx.com/products/ipcenter/DES.htm> を参照してください。

BitGen の構文

BitGen のコマンドライン構文は、次のとおりです。

bitgen [*options*] *infile* [.ncd] [*outfile*] [*pcf_file*.pcf]

- ・ *options*: 「BitGen のオプション」にリストされているオプションを 1 つ以上指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。
- ・ *infile*: ビットストリームを生成する NCD (Native Circuit Description) ファイルの名前です。
- ・ *outfile*: 出力ファイル名です。
 - 出力ファイル名を指定しない場合、入力ファイルのディレクトリ内にビットストリーム (BIT) ファイルが作成されます。
 - 表「BitGen のオプションと出力ファイル」に示すオプションを指定すると、BIT ファイルと共に対応するファイルが作成されます。
 - 拡張子を指定しない場合は、適切な拡張子が自動的に追加されます。
 - BitGen のすべての出力を含むレポート ファイルは、出力ファイルと同じディレクトリに自動的に作成されます。
 - レポート ファイル名は、出力ファイルと同じルート名で、拡張子は .bgn です。
- ・ *pcf_file*: 物理制約ファイル (PCF) の名前です。BitGen は、PCF ファイルを使用して CONFIG 制約を解釈します。CONFIG 制約には、次の機能があります。
 - ビットストリーム オプションを制御
 - デフォルトのビヘイビアよりも優先される
 - コンフィギュレーション オプションよりも優先度が低い
 デフォルトでは、PCF が自動的に読み込まれます。
 - PCF ファイルをコマンドラインで 2 番目のファイルとして指定する場合は、拡張子 .pcf を必ず指定してください。
 - PCF ファイルをコマンドラインで 3 番目のファイルとして指定する場合は、拡張子を指定する必要はなく、指定しない場合は .pcf となります。

PCF ファイル名を指定する場合は、そのファイルが存在することを確認してください。存在しない場合、入力デザインと同じルート名で拡張子が .pcf のファイルが読み込まれます。

BitGen のオプションと出力ファイル

BitGen のオプション	出力ファイル
-l	<i>outfile_name</i> .ll
-m	<i>outfile_name</i> .msk
-b	<i>outfile_name</i> .rbt

BitGen のオプション

このセクションでは、BitGen のコマンドライン オプションについて説明します。

- ・ [-b \(ロービット ファイルの作成\)](#)
- ・ [-bd \(ブロック RAM のアップデート\)](#)
- ・ [-d \(DRC を実行しない\)](#)
- ・ [-f \(コマンド ファイルの実行\)](#)
- ・ [-g \(コンフィギュレーションの設定\)](#)
- ・ [-intstyle \(統合スタイル\)](#)
- ・ [-j \(BIT ファイルを生成しない\)](#)
- ・ [-l \(ロジック アロケーション ファイルの作成\)](#)
- ・ [-m \(マスク ファイルの作成\)](#)
- ・ [-r \(パーシャル BIT ファイルの作成\)](#)
- ・ [-w \(既存の出力ファイルの上書き\)](#)

-b (ロービット ファイルの作成)

ロービット ファイル (*file_name*.rbt) を作成します。

構文

-b

-b オプションと共に **-g Readback** オプションを指定した場合、ASCII 形式のリードバック コマンド ファイル (*file_name*.rba) も生成されます。

ロービットファイルでは、ビットストリーム ファイルのデータが 1 と 0 で表されています。マイクロプロセッサを使用して 1 つの FPGA をコンフィギュレーションする場合、ソース コードにロービット ファイルをテキスト ファイルとして含め、コンフィギュレーション データとすることができます。ロービット ファイル内の文字の順序と FPGA デバイスに書き込まれるビットの順序は同じです。

-bd (ブロック RAM のアップデート)

指定した ELF または MEM に記述されたブロック RAM のデータを使用してビットストリームをアップデートします。

構文

-bd *file_name* { .elf | .mem }

-d (DRC を実行しない)

このオプションを使用すると、デザイン ルール チェック (DRC) は実行されません。

構文

-d

-d オプションを使用しない場合は DRC が実行され、次の 2 つの出力ファイルに結果が保存されます。

- ・ BitGen レポート ファイル (*file_name.bgn*)
- ・ DRC ファイル (*file_name.drc*)

-d オプションを使用すると、次のようになります。

- ・ レポート ファイルに DRC 情報は含まれません。
- ・ DRC ファイルは作成されません。

ビットストリームを作成する前に DRC を実行することにより、FPGA デバイスの誤動作の原因となるエラーを検出できます。DRC でエラーが検出されない場合は、ビットストリーム ファイルが作成されます (「[-j \(BIT ファイルを生成しない\)](#)」に説明されている **-j** オプションを使用しない場合)。

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f \(コマンド ファイルの実行\)](#)」を参照してください。

-g (コンフィギュレーションの設定)

ザイリンクス FPGA デバイスのスタートアップ タイミング オプションやその他のビットストリーム オプションを指定します。コンフィギュレーション設定には、以下に示すサブオプションを使用します。

構文

-g *sub-option:setting design.ncd design.bit design.pcf*

たとえば、リードバックを有効にするには、次の構文を使用します。

bitgen -g readback

アーキテクチャ別の設定を表示するには、**bitgen -h [architecture]** コマンドを実行します。デフォルトの値は、アーキテクチャによって異なります。

サブオプションとその設定

この後のセクションで、**-g** オプションのサブオプションとその設定をリストします。

<ul style="list-style-type: none"> · ActivateGCLK · ActiveReconfig · Binary · BPL_1st_read_cycle · BPL_page_size · BusyPin · CclkPin · Compress · ConfigFallBack · ConfigRate · CRC · CsPin · DCIUpdateMode · DCMShutdown · DebugBitstream · DinPin · DONE_cycle · DonePin · DonePipe · drive_away · DriveDone · Encrypt · EncryptKeySelect · en_porb · en_sw_gsr · ExtMasterCclk_divide · ExtMasterCclk_en 	<ul style="list-style-type: none"> · failsafe_user · Glutmask · golden_config_addr · GTS_cycle · GWE_cycle · Hswapen · IEEE1532 · InitPin · HKey · JTAG_SysMon · Key0 · KeyFile · LCK_cycle · M0Pin · M1Pin · M2Pin · Match_cycle · MultiBootMode · multipin_wakeup · next_config_addr · next_config_boot_mode · next_config_new_mode · next_config_register_write · OverTempPowerDown · PartialGCLK · PartialLeft 	<ul style="list-style-type: none"> · PartialMask0、 PartialMask1、 PartialMask2 · PartialRight · Persist · PowerdownPin · ProgPin · RdWrPin · ReadBack · reset_on_error · Security · SelectMAPAbort · StartCBC · StartupClk · sw_clk · sw_gts_cycle · sw_gwe_cycle · SPL_buswidth · TckPin · TdiPin · TdoPin · TIMER_CFG · TIMER_USR · TmsPin · UnusedPin · UserID · wakeup_mask
--	--	---

ActiveReconfig

コンフィギュレーション中に GHIGH および GSR がアサートされないように指定します。これは、パーシャル リコンフィギュレーションの追加機能です。

アーキテクチャ	Virtex®-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	No、Yes
デフォルト	No

Binary

プログラム データのみを含むバイナリ ファイルを作成します。プログラム データを抽出して表示する場合にこのオプションを使用します。ヘッダに変更を加えても、この抽出プロセスには影響しません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	No、Yes
デフォルト	No

BPI_1st_read_cycle

BPI コンフィギュレーションをフラッシュ デバイスのページ モード操作のタイミングと同期化します。最初のページの有効な読み出しに対するサイクル数を設定します。このオプションを使用するには、BPI_page_size を 4 または 8 に設定する必要があります。

アーキテクチャ	Virtex-5、Virtex-6
設定	1、2、3、4
デフォルト	1

BPI_page_size

BPI コンフィギュレーションで、フラッシュ メモリの各ページに必要な読み出し数に対応するページ サイズを指定します。

アーキテクチャ	Virtex-5、Virtex-6
設定	1、4、8
デフォルト	1

BusyPin

ピンをウィーク プルアップまたはプルダウンする内部抵抗を追加します。**Pullnone** に設定すると抵抗は追加されず、ピンはプルアップまたはプルダウンされません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

CclkPin

Cclk ピンに内部プルアップを追加します。**Pullnone** に設定すると、プルアップは接続されません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3
設定	Pullnone、Pullup
デフォルト	Pullup

Compress

ビットストリームの複数フレーム書き込み機能を使用して、BIT ファイルのみではなく、ビットストリームのサイズを縮小します。ただし、このオプションを使用しても、必ずしもビットストリームのサイズが縮小されるとは限りません。圧縮を有効にするには、**bitgen -g compress** オプションを指定します。このオプションを指定しない場合、圧縮は実行されません。

-r オプションを設定すると複数フレーム書き込み機能が自動的に使用され、生成されたパーシヤル BIT ファイルは圧縮されたビットストリームとなります。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	なし
デフォルト	オフ

ConfigFallBack

コンフィギュレーションが正しく実行できなかった場合の、デフォルトビットストリームの読み込みをイネーブルまたはディスエーブルにします。

アーキテクチャ	Virtex-5、Virtex-6
設定	Enable、Disable
デフォルト	Enable

ConfigRate

マスタ モードでコンフィギュレーションする際、内部オシレータを使用してコンフィギュレーションクロック Cclk が生成されます。この Cclk のレートを ConfigRate オプションを使用して選択します。

アーキテクチャ	設定	デフォルト
Virtex-4	4、5、7、8、9、10、13、15、20、26、30、34、41、45、51、55、60	4
Virtex-5	2、6、9、13、17、20、24、27、31、35、38、42、46、49、53、56、60	2
Virtex-6	2、4、6、10、12、16、22、26、33、40、50、66	2
Spartan-3	6、3、12、25、50	6
Spartan-3A	6、1、3、7、8、10、12、13、17、22、25、27、33、44、50、100	6
Spartan-3E	1、3、6、12、25、50	1
Spartan-6	2、4、6、10、12、16、22、26	2

CRC

ビットストリームの CRC (Cyclic Redundancy Check) 値の生成を制御します。Enable に設定すると、ビットストリームの内容に基づいて CRC 値が算出されます。算出された値がビットストリームの値と一致しない場合、デバイスはコンフィギュレーションされません。Disable に設定すると、CRC 値の代わりに定数値がビットストリームに挿入され、CRC 値は計算されません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Disable、Enable
デフォルト	Enable

CsPin

ピンをウィークプルアップまたはプルダウンする内部抵抗を追加します。**Pullnone** に設定すると抵抗は追加されず、ピンはプルアップまたはプルダウンされません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

DCIUpdateMode

DCI (デジタル制御インピーダンス) により I/O 規格に合わせてインピーダンスを調整する頻度を制御します。

アーキテクチャ	Virtex-4、Virtex-5、Spartan-3
設定	As required、Continuous、Quiet
デフォルト	As required

DCMShutdown

SHUTDOWN および AGHIGH コマンドがコンフィギュレーション ロジックに読み込まれたときに DCM (デジタル クロック マネージャ) をリセットするよう設定します。

アーキテクチャ	Spartan-3、Spartan-3E
設定	Disable、Enable
デフォルト	Disable

DebugBitstream

デバッグ ストリームを作成します。デバッグ ビットストリームのサイズは、標準的なビットストリームよりもかなり大きくなります。このオプションは、マスタ シリアルおよびスレーブ シリアル コンフィギュレーションでのみ使用可能です。バウンダリ スキャンおよびスレーブ パラレル /SelectMAP には使用できません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	No、Yes
デフォルト	No

デバッグ用ビットストリームには、標準ビットストリームに次の機能が追加されています。

- ・ 同期ワードの後、32 個の 0 を LOUT レジスタに書き込みます。
- ・ 各フレームを個別に読み込みます。
- ・ 各フレームの後、CRC (Cyclic Redundancy Check) を実行します。
- ・ 各フレームの後、列番号を LOUT レジスタに書き込みます。

DinPin

ピンをウィーク プルアップまたはプルダウンする内部抵抗を追加します。**Pullnone** に設定すると抵抗は追加されず、ピンはプルアップまたはプルダウンされません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

DONE_cycle

FPGA の Done 信号をアクティブにするスタートアップ フェーズを選択します。**DonePipe** を **Yes** に設定すると、Done が遅れます。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	1、2、3、4、5、6
デフォルト	4

DonePin

DONE ピンに内部プルアップを追加します。**Pullnone** に設定すると、プルアップは接続されません。**DonePin** オプションは、外部プルアップ抵抗をこのピンに接続する場合にのみ使用してください。このオプションを使用しない場合、内部プルアップ抵抗が自動的に接続されます。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Pullup、Pullnone
デフォルト	Pullup

DonePipe

CFG_DONE (DONE) ピンが High になった後、最初のクロック エッジが発生してから FPGA デバイスが **Done** ステートになるよう設定します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	No、Yes
デフォルト	No

drive_aware

AWAKE ピンをアクティブに駆動するか、オープンドレインにするかを指定します。オープンドレインにする場合は、オープン抵抗を使用して High にする必要があります。AWAKE ピンは、デバイスが SUSPEND モードであるかどうかを検出します。

アーキテクチャ	Spartan-3A、Spartan-6
設定	No、Yes
デフォルト	No

DriveDone

プルアップを使用せずに、DONE ピンを直接 High に駆動します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	No、Yes
デフォルト	No

en_por_b

SUSPEND ステートでのパワーオンリセット (POR) 検出をアクティブにするかどうかを指定します。デフォルトでは **Yes** に設定されており、**por_b** の検出は常にアクティブで、電圧が低すぎる場合に FPGA デバイスがリセットされます。

No に設定すると、次のようになります。

- ・ **por_b** の検出は SUSPEND ピンが Low の場合はイネーブル
- ・ **por_b** の検出は SUSPEND ピンが High の場合はディスエーブル

アーキテクチャ	Spartan-3A
設定	No、Yes
デフォルト	Yes

en_sw_gsr

FPGA が SUSPEND モードからウェークアップ状態になったときに、メモリ セルからフリップフロップの値を復元します。

アーキテクチャ	Spartan-3A、Spartan-6
設定	No、Yes
デフォルト	No

Encrypt

ビットストリームを暗号化します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6 アーキテクチャ、Spartan-6 デバイス (LX75/T 以上)
設定	No、Yes
デフォルト	No

暗号化については、<http://japan.xilinx.com/products/ipcenter/DES.htm> を参照してください。

EncryptKeySelect

AES 暗号キーの場所 (バッテリ バックアップ付き RAM (BBRAM) または eFUSE レジスタ) を指定します。

メモ: このサブオプションは Encrypt が Yes に設定されている場合のみ設定可能です。

アーキテクチャ	Virtex-6 アーキテクチャ、Spartan-6 デバイス (LX75/T 以上)
設定	bbram、efuse
デフォルト	bbram

暗号化については、<http://japan.xilinx.com/products/ipcenter/DES.htm> を参照してください。

ExtMasterCclk_en

すべてのマスター モードで外部クロックをコンフィギュレーション クロックとして使用できるように設定します。外部クロックは多目的 USERCCLK ピンに接続する必要があります。

アーキテクチャ	Spartan-6
設定	No、Yes
デフォルト	No

ExtMasterCclk_divide

外部マスタ コンフィギュレーション クロックを内部で分周するよう設定します。

メモ : このサブオプションは、**ExtMasterCclk_en** が **Yes** に設定されている場合のみ設定可能です。

アーキテクチャ	Spartan-6
設定	1、2 ~ 1022 の 2 の倍数
デフォルト	1

failsafe_user

GENERAL5 レジスタのアドレスを設定します。GENERAL5 は、フェイルセーフ機構に必要な追加情報を格納する 16 ビット レジスタです。

アーキテクチャ	Spartan-6
設定	4 桁の 16 進文字列
デフォルト	0x0000

Glutmask

コンフィギュレーション リードバックまたは SEU リードバック中に、LUTRAM フレームをマスクします。

アーキテクチャ	Spartan-3A
設定	No、Yes
デフォルト	Yes

golden_config_addr

ゴールデン コンフィギュレーション イメージ用の GENERAL3、4 でのアドレスを設定します。

アーキテクチャ	Spartan-6
設定	8 桁の 16 進文字列
デフォルト	0x00000000

GTS_cycle

I/O バッファに内部トライステート信号を解放するスタートアップ フェーズを選択します。**Done** に設定すると、**DoneIn** 信号が High のときに GTS がアサートされます。**DoneIn** は Done ピンの値で、**DonePipe** を **Yes** に設定している場合は遅延があります。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	1、2、3、4、5、6、Done、Keep
デフォルト	5

GWE_cycle

フリップフロップ、LUT RAM、シフトレジスタに対して内部ライト イネーブル信号をアサートするスタートアップ フェーズを選択します。BRAM もイネーブルになります。スタートアップ フェーズの前は、BRAM の書き込み/読み込みは無効です。**Done** に設定すると、**DoneIn** 信号が High のときに GWE がアサートされます。**DoneIn** は Done ピンの値で、**DonePipe** を **Yes** に設定している場合は遅延があります。**Keep** に設定すると、GWE 信号の現在の値が保持されます。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	1、2、3、4、5、6、Done、Keep
デフォルト	6

HKey

ビットストリームの暗号化用に HMAC 認証キーを設定します。Virtex-6 デバイスには、オンチップの HMAC (Keyed Hash Message Authentication Code) アルゴリズムがハードウェアにインプリメントされており、AES 暗号化のみの場合よりセキュリティが強化されています。Virtex-6 デバイスでは、ビットストリームの読み込み、変更、コピーを実行するのに AES キーと HMAC キーの両方が必要です。

pick に設定した場合、値がランダムに選択されます。このサブオプションを使用するには、**-g Encrypt:Yes** を設定する必要があります。

アーキテクチャ	Virtex-6
設定	Pick、16 進文字列
デフォルト	Pick

暗号化については、<http://japan.xilinx.com/products/ipcenter/DES.htm> を参照してください。

HswapenPin

Hswapen ピンにプルアップ、プルダウンを追加するかどうかを指定します。**Pullnone** に設定すると、プルアップにもプルダウンにも接続されません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

IEEE1532

IEEE 1532 コンフィギュレーション ファイルを作成します。StartUpClk を JTAG クロックに設定する必要があります。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	No、Yes
デフォルト	No

InitPin

INIT ピンにプルアップ抵抗を追加するか、未接続のままにするかを指定します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6
設定	Pullup、Pullnone
デフォルト	Pullup

JTAG_SysMon

システム モニタへの JTAG 接続をイネーブルまたはディスエーブルにします。

アーキテクチャ	Virtex-5、Virtex-6
設定	Enable、Disable
デフォルト	Enable

Virtex-5 デバイスでは、このオプションを Enabled に設定すると属性ビット sysmon_test_a[1] が 1 に設定されます。

Virtex-6 デバイスでは、このオプションを Enabled に設定すると属性ビット sysmon_test_e[2:0] が 3'b111 に設定されます。

Key0

ビットストリーム暗号化用に AES 暗号キーを設定します。**pick** に設定した場合、値がランダムに選択されます。このサブオプションを使用するには、**-g Encrypt:Yes** を設定する必要があります。

Virtex-6 デバイスでは、ビットストリームの読み込み、変更、コピーを実行するのに AES キーと HMAC キーの両方が必要です。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6 アーキテクチャ、Spartan-6 デバイス (LX75/T 以上)
設定	Pick、16 進文字列
デフォルト	Pick

暗号化については、<http://japan.xilinx.com/products/ipcenter/DES.htm> を参照してください。

KeyFile

入力する暗号化ファイルの名前 (拡張子は .nky) を指定します。このサブオプションを使用するには、**-g Encrypt:Yes** を設定する必要があります。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6 アーキテクチャ、Spartan-6 デバイス (LX75/T 以上)
設定	文字列
デフォルト	指定されない

暗号化については、<http://japan.xilinx.com/products/ipcenter/DES.htm> を参照してください。

LCK_cycle

DLL/DCM/PLL がロックするまで待機するスタートアップ フェーズを選択します。**NoWait** に設定すると、スタートアップ シーケンスで DLL/DCM/PLL がロックされるまで待機しません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定 (Virtex-6 および Spartan-6)	0、1、2、3、4、5、6、7、NoWait
設定 (その他のデバイス)	0、1、2、3、4、5、6、NoWait
デフォルト	NoWait

M0Pin

M0 ピンに内部プルアップまたはプルダウンを追加するかどうかを指定します。M0 ピンのプルアップ抵抗とプルダウン抵抗の両方を無効にするには、**Pullnone** に設定します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

M1Pin

M1 ピンに内部プルアップまたはプルダウンを追加するかどうかを指定します。M1 ピンのプルアップ抵抗とプルダウン抵抗の両方を無効にするには、**Pullnone** を選択します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

M2Pin

M2 ピンに内部プルアップまたはプルダウンを追加するかどうかを指定します。M2 ピンのプルアップ抵抗とプルダウン抵抗の両方を無効にするには、**Pullnone** を選択します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

Match_cycle

DCI (デジタル制御インピーダンス) の一致信号がアサートされるまで待機するスタートアップ フェーズを指定します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3
設定	0、1、2、3、4、5、6、Auto、NoWait
デフォルト	Auto

DCI の一致がこのオプションで設定したフェーズで開始するのではなく、DCI が一致するまでスタートアップ シーケンスが待機します。DCI が一致するまでにかかる時間にはさまざまな要素が関係するので、スタートアップ シーケンスを完了するまでにかかる CCLK サイクルはシステムによって異なります。DONE が High になるまで CCLK を駆動し続けるようにするのが理想的です。

Auto に設定すると、DCI の I/O 規格が検索されます。DCI に I/O 規格が設定されている場合は **Match_cycle:2** が使用され、設定されていない場合は **Match_cycle:NoWait** が使用されます。

MultiBootMode

Spartan-3E のマルチブート機構をイネーブルまたはディスエーブルにします。ディスエーブルにすると、スタートアップ ブロックの MBT の値が無視されます。

アーキテクチャ	Spartan-3E
設定	No、Yes
デフォルト	No

multipin_wakeup

システム コンフィギュレーション ポート (SCP) ピンをイネーブルにし、FPGA を SUSPEND モードから通常の動作状態に戻します。

アーキテクチャ	Spartan-6
設定	No、Yes
デフォルト	No

next_config_addr

マルチブート設定の次のコンフィギュレーションの開始アドレスを設定します。このアドレスは、General1 および General2 レジスタに格納されます。

アーキテクチャ	Spartan-3A、Spartan-6
設定	8 桁の 16 進文字列
デフォルト	0x00000000

next_config_boot_mode

マルチブート設定において、次のコンフィギュレーションのコンフィギュレーション モードを設定します。Spartan-6 では MSB を 0 にする必要があり、次の 2 ビットはモード ピン M[1:0] を表します。

アーキテクチャ	Spartan-3A、Spartan-6
設定	3 ビットの 2 進文字列
デフォルト	001

next_config_new_mode

モード ピンのモード値またはビットストリームで **next_config_boot_mode** により指定されているモード値のどちらを使用するかを選択します。**Yes** に設定すると、次のマルチブートコンフィギュレーションでモード ピンの値ではなく **next_config_boot_mode** で指定されたモード値が使用されます。

アーキテクチャ	Spartan-3A、Spartan-6
設定	No、Yes
デフォルト	No

next_config_register_write

ゴールデン ビットストリーム用にマルチブート ヘッダをイネーブルにします。このオプションはゴールデン イメージ用であり、マルチブート イメージのアドレスが含まれます。このヘッダを含むゴールデン イメージがデバイスに読み込まれると、next_config_addr で指定されたアドレスにジャンプし、マルチブート イメージが読み込まれます。マルチブート イメージのコンフィギュレーションが正しく完了しない場合、デバイスにゴールデン イメージが再び読み込まれ、ヘッダがスキップされます。

アーキテクチャ	Spartan-3A、Spartan-6
設定	Enable、Disable
デフォルト	Enable

OverTempPowerDown

システム モニタで温度が最高操作温度を超えたことが検出された場合に、デバイスをシャットダウンするよう設定します。このオプションを使用するには、システム モニタの外部回路設定をオンにする必要があります。

アーキテクチャ	Virtex-5、Virtex-6
設定	Disable、Enable
デフォルト	Disable

PartialGCLK

中央のグローバル クロック列のフレームをフレームのリストに追加し、パーシャル ビットストリームに出力します。このオプションは、**PartialMask0:1** と同じです。

アーキテクチャ	Spartan-3、Spartan-3A、Spartan-3E
設定	なし
デフォルト	指定なし (パーシャル マスクは使用されない)

PartialLeft

デバイスの左側のフレームをフレームのリストに追加し、パーシャル ビットストリームに出力します。CLB、IOB、および BRAM 列は含まれますが、中央のグローバル クロック列は含まれません。

アーキテクチャ	Spartan-3、Spartan-3A、Spartan-3E
設定	なし
デフォルト	指定なし (パーシャル マスクは使用されない)

PartialMask0、PartialMask1、PartialMask2

マスクで有効にされた値を持つブロック タイプ 0、1、または 2 の主なアドレスのみが含まれたビットストリームを生成します。ブロック タイプは、該当デバイスのブロック RAM ではない初期化データ フレームです。

アーキテクチャ	Spartan-3、Spartan-3A、Spartan-3E
設定	すべてのカラムがイネーブル、主なアドレスのマスク
デフォルト	指定なし (パーシャル マスクは使用されない)

PartialRight

デバイスの右側のフレームをフレームのリストに追加し、パーシャル ビットストリームに出力します。CLB、IOB、および BRAM 列は含まれますが、中央のグローバル クロック列は含まれません。

アーキテクチャ	Spartan-3、Spartan-3A、Spartan-3E
設定	なし
デフォルト	指定なし (パーシャル マスクは使用されない)

Persist

SelectMAP モード ピンをユーザー I/O として使用できないよう設定します。SelectMAP モードおよび関連ピンについては、データシートを参照してください。

SelectMAP モードのコンフィギュレーション ピンを使用するリードバックおよびパーシャル リコンフィギュレーションで必要であり、SelectMAP またはシリアル モードを使用する場合に使用する必要があります。影響を受けるのは SelectMAP ピンのみですが、コンフィギュレーション後に JTAG 以外のコンフィギュレーション ピンにアクセスするにはこのオプションを使用する必要があります。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	No、Yes
デフォルト	No

メモ： このコマンドの影響を受けるピンは、CONFIG_MODE 制約で選択されているモードによって異なります。詳細は、『[制約ガイド](#)』を参照してください。CONFIG_MODE 制約を設定していない場合は、Persist を Yes に設定すると、最初の 8 本の SelectMAP ピンが予約されます。

PowerdownPin

ピンの内部プルアップをイネーブルにするかどうかを指定し、ピンをスリープ モードにします。

アーキテクチャ	Virtex-4
設定	Pullup、Pullnone
デフォルト	Pullup

ProgPin

ProgPin ピンに内部プルアップを追加します。**Pullnone** に設定すると、プルアップは接続されません。ピン上のプルアップは、コンフィギュレーション後にアクティブになります。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Pullup、Pullnone
デフォルト	Pullup

RdWrPin

ピンをウィークプルアップまたはプルダウンする内部抵抗を追加します。**Pullnone** に設定すると抵抗は追加されず、ピンはプルアップまたはプルダウンされません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

ReadBack

必要なリードバック ファイルを作成してリードバックを実行できるようにします。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	なし
デフォルト	指定なし (リードバック ファイルは作成されない)

-g ReadBack オプションを指定すると、RBB、RBD、および MSD ファイルが作成されます。

-b オプションと共に **-g Readback** オプションを指定した場合、ASCII 形式のリードバック コマンド ファイル (*file_name.rba*) も生成されます。

reset_on_error

CRC エラーが検出された場合に FPGA デバイスをリセットします。マスタ モードのコンフィギュレーションにのみ適用されます。

アーキテクチャ	Spartan-3A、Spartan-6
設定	No、Yes
デフォルト	No

Security

リードバックおよびリコンフィギュレーションを無効にするかどうかを指定します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Level1、Level2、None
デフォルト	なし

Level1 に設定するとリードバックが無効になり、**Level2** に設定するとリードバックとリコンフィギュレーションが無効になります。

SelectMapAbort

SelectMAP の ABORT シーケンスをイネーブルまたはディスエーブルに設定します。ディスエーブルにすると、デバイスピン上の ABORT シーケンスは無視されます。

アーキテクチャ	Virtex-5
設定	Enable、Disable
デフォルト	Enable

SPI_buswidth

サードパーティの SPI フラッシュ デバイスからのマスタ SPI コンフィギュレーションにおいて、SPI バスを Dual (x2) または Quad (x4) モードに設定します。

アーキテクチャ	Spartan-6
設定	1、2、4
デフォルト	1

StartCBC

CBC (暗号ブロック連鎖) の開始値を設定します。**Pick** に設定した場合、値がランダムに選択されます。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6 アーキテクチャ、Spartan-6 デバイス (LX75/T 以上)
設定	Pick、32 ビットの 16 進文字列
デフォルト	Pick

StartupClk

デバイスのコンフィギュレーションに続くスタートアップ シーケンスは、Cclk、ユーザー クロック、JTAG クロックのいずれかに同期させることができます。デフォルトでは **Cclk** に設定されています。

- ・ **Cclk** : FPGA デバイスの内部クロックに同期させる場合は、Cclk に設定します。
- ・ **UserClk** : STARTUP シンボルの CLK ピンに接続されたユーザー定義信号に同期させる場合は、UserClk に設定します。
- ・ **JtagClk** : JTAG のクロックに同期させる場合は、JtagClk に設定します。このクロックは、JTAG の制御ロジックとなる TAP コントローラを制御します。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Cclk (ピン - メモを参照)、UserClk (ユーザー指定)、JtagClk
デフォルト	Cclk

メモ : Cclk が出力になるモードでは、このピンは内部オシレータによって駆動されます。

sw_clk

デバイスが SUSPEND モードからウェークアップする際のスタートアップ クロックを指定します。

アーキテクチャ	Spartan-3A、Spartan-6
設定	Startupclk、Internalclk
デフォルト	Startupclk

sw_gts_cycle

デバイスが SUSPEND モードからウェークアップする際に適用されます。1 ～ 1024 の値を設定できます。

アーキテクチャ	Spartan-3A、Spartan-6
設定	4、文字列
デフォルト	4

sw_gwe_cycle

デバイスが SUSPEND モードからウェークアップする際に適用されます。1 ～ 1024 の値を設定できます。

アーキテクチャ	Spartan-3A、Spartan-6
設定	5、文字列
デフォルト	5

TckPin

JTAG テスト クロックである TCK ピンに、プルアップまたはプルダウンを追加するかどうかを指定します。1 つの設定を選択するとそれが有効になり、ほかの設定は無効になります。**Pullnone** に設定すると、プルアップにもプルダウンにも接続されません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

TdiPin

すべての JTAG 命令と JTAG レジスタへのシリアル データ入力である TDI ピンに、プルアップまたはプルダウンを追加するかどうかを指定します。1 つの設定を選択するとそれが有効になり、ほかの設定は無効になります。**Pullnone** に設定すると、プルアップにもプルダウンにも接続されません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

TdoPin

すべての JTAG 命令とデータレジスタへのシリアル データ出力である TdoPin ピンに、プルアップまたはプルダウンを追加するかどうかを指定します。1 つの設定を選択するとそれが有効になり、ほかの設定は無効になります。**Pullnone** に設定すると、プルアップにもプルダウンにも接続されません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

TIMER_CFG

コンフィギュレーション モードのウォッチドッグ タイマの値を設定します。このオプションは、TIMER_USR と同時に使用することはできません。

アーキテクチャ	Virtex-5、Virtex-6、Spartan-6
設定 (Spartan-6)	4 桁の 16 進文字列
設定 (Virtex-5 および Virtex-6)	6 桁の 16 進文字列
デフォルト (Spartan-6)	0x0000
デフォルト (Virtex-5 および Virtex-6)	0xFFFF

TIMER_USR

ユーザー モードのウォッチドッグ タイマの値を設定します。このオプションは、TIMER_CFG と同時に使用することはできません。

アーキテクチャ	Virtex-5、Virtex-6
設定	6 桁の 16 進文字列
デフォルト	0x000000

TmsPin

TAP コントローラへのモード入力信号である TMS ピンに、プルアップまたはプルダウンを追加するかどうかを指定します。TAP コントローラは、JTAG のコントロール ロジックとなります。1 つの設定を選択するとそれが有効になり、ほかの設定は無効になります。**Pullnone** に設定すると、プルアップにもプルダウンにも接続されません。

アーキテクチャ	Virtex-4、Virtex-5、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pullup

UnusedPin

すべての JTAG 命令とデータレジスタの使用されていないデバイス ピンとシリアル データ出力 (TDO) に、プルアップまたはプルダウンを追加するかどうかを指定します。1 つの設定を選択するとそれが有効になり、ほかの設定は無効になります。**Pullnone** に設定すると、プルアップにもプルダウンにも接続されません。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	Pullup、Pulldown、Pullnone
デフォルト	Pulldown

UserID

インプリメンテーションのレビジョンを識別するために使用します。8 桁までの 16 進文字列を入力できます。

アーキテクチャ	Virtex-4、Virtex-5、Virtex-6、Spartan-3、Spartan-3A、Spartan-3E、Spartan-6
設定	8 桁の 16 進文字列
デフォルト	0xFFFFFFFF

wakeup_mask

SUSPEND モードからのウェークアップ用に 8 つの SCP ピンのうちどれをイネーブルにするかを指定します。

メモ : このサブオプションは `multipin_wakeup` が Yes に設定されている場合のみ設定可能です。

アーキテクチャ	Spartan-6
設定	2 桁の 16 進文字列
デフォルト	0x00

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle ise|xflow|silent

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-j (BIT ファイルを生成しない)

ビットストリーム (BIT) ファイルを作成しないよう指定します。このオプションは、ビットストリームを作成せずにレポートを生成する場合に使用します。たとえば、ビットストリーム ファイルを作成せずに DRC を実行する場合などに、**-j** オプションを使用できます。ただし、MSK ファイルと RBT ファイルは作成されます。

構文

-j

-l (ロジック アロケーション ファイルの作成)

選択したデザインに対して、ASCII 形式のロジック アロケーション ファイル (`design.ll`) を作成します。ロジック アロケーション ファイルには、ラッチ、フリップフロップ、IOB の入力および出力のビットストリーム位置が記述されます。

構文

-l

アプリケーションによっては、異なる時点における FPGA の内部レジスタの内容を調べる必要がある場合があります。-1 オプションで作成されるファイルは、現在のビットストリーム内のどのビットがフリップフロップとラッチの出力を表すかを識別するのに役立ちます。ビットは、フレームとフレーム内のビット番号によって参照されます。

iMPACT は、design.11 ファイルからリードバックビットストリーム内の信号値を検索します。

-m (マスク ファイルの作成)

マスク ファイルを作成します。このファイルは、検証時にリードバック データと比較するビットストリームのビット位置を指定します。

構文

-m

-r (パーシャル BIT ファイルの作成)

パーシャル ビットストリーム ファイル (BIT) を作成します。

BIT ファイルと指定した NCD ファイルが比較され、元の BIT ファイルとは異なる部分のみが出力されます。

構文

-r *bit_file*

-w (既存の出力ファイルの上書き)

既存の BitGen 出力ファイルを上書きします。

構文

-w

BitGen の出力ファイルについては、「[BitGen の概要](#)」を参照してください。

BSDLAnno

この章では、BSDLAnno について説明します。次のセクションが含まれています。

- ・ [BSDLAnno の概要](#)
- ・ [BSDLAnno の構文](#)
- ・ [BSDLAnno のオプション](#)
- ・ [BSDL ファイルの構成と BSDLAnno による変換](#)
- ・ [ザイリンクス デバイスのバウンダリ スキャン ビヘイビア](#)

BSDLAnno の概要

BSDLAnno は、コンフィギュレーション後のインターコネクト テスト用に BSDL ファイルを自動的に変更するコマンドライン ツールです。BSDLAnno には、次の機能があります。

- ・ 配線された NCD ファイル (FPGA デバイス) または PNX ファイル (CPLD デバイス) から必要なデザイン情報を読み込みます。
- ・ コンフィギュレーション後のバウンダリ スキャン アーキテクチャを反映した BSDL ファイルを生成します。

デバイスがコンフィギュレーションされる時点と、バウンダリ スキャン レジスタとパッド間の特定の接続が変更されることがあるため、これに伴ってバウンダリ スキャン アーキテクチャも変更されます。この変更は、コンフィギュレーション後の BSDL ファイルを介してバウンダリ スキャン テスタに伝搬されます。バウンダリ スキャン アーキテクチャの変更が BSDL ファイルに反映されていないと、バウンダリ スキャン テスタが正しく実行されません。

バウンダリ スキャン 記述言語 (BSDL) は、デバイスのバウンダリ スキャン アーキテクチャを定義する一般的な方法として IEEE 1149.1 で定義されています。ザイリンクスでは、コンフィギュレーション前のバウンダリ スキャン アーキテクチャを記述するファイルとして IEEE 1149.1 および 1532 の BSDL ファイルを提供しています。

ほとんどのデバイス ファミリーで、バウンダリ スキャン アーキテクチャがコンフィギュレーション後に変更されます。これは、バウンダリ スキャン レジスタが出力バッファと入力センスアンプの前にあるためです。

バウンダリ スキャン レジスタ → 出力バッファ/入力センス アンプ → パッド

ハードウェアは、指定された I/O 規格でバウンダリ スキャン ロジックが動作するようこのように配置されています。このため、I/O 規格の使用範囲内でバウンダリ スキャン テストを実施できます。

BitGen のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

入力ファイル

BSDLAAnno では、コンフィギュレーション後の BSDL ファイルを生成するのに次の 2 種類の入力ファイルが使用されます。

- ・ コンフィギュレーション前の BSDL ファイル。ザイリンクスのインストール ディレクトリから自動的に読み込まれます。
- ・ 配線済みの NCD ファイル (FPGA デバイス) または PNX ファイル (CPLD デバイス)。入力ファイルとして指定します。

ファイル	短縮形	拡張子	説明/メモ
Native Circuit Description	NCD	.ncd	ターゲット デバイスにマップ、配置配線されたデザインの物理的な記述 (FPGA デバイス用)。
バウンダリ スキャン記述言語	BSDL	.bsd	BSDL 出力ファイル名は、拡張子 .bsd も含めて 24 文字以内で指定してください。
外部ピン記述 (XDM フォーマット)	PNX	.pnx	CPLD デバイス用。

出力ファイル

BSDLAAnno から出力されるファイルは、ASCII 形式の BSDL ファイルであり、次を反映して変更されています。

- ・ 信号方向 (入力/出力/双方向)
- ・ 未使用の I/O
- ・ その他のデザイン固有のバウンダリ スキャン ビヘイビア

BSDLAAnno の構文

BSDLAAnno のコマンドライン構文は、次のとおりです。

```
bsdlanno [options] infile outfile[.bsd]
```

options: 「BSDLAAnno のオプション」にリストされているオプションを 1 つ以上指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

infile: デザインのソース ファイルです。

- ・ FPGA デバイスでは配線済み (PAR 後) の NCD ファイル
- ・ CPLD デバイスでは PNX ファイル

outfile: 出力 BSDL ファイルです。拡張子 .bsd の指定はオプションです。

BSDLAAnno のオプション

このセクションでは、BSDLAAnno のコマンド ライン オプションについて説明します。

- ・ `-intstyle` (統合スタイル)
- ・ `-s` (BSDL ファイルの指定)

`-intstyle` (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

```
-intstyle ise|xflow|silent
```

次のいずれかのモードに設定します。

- ・ **-intstyle ise**: プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow**: プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent**: 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ: Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

`-s` (BSDL ファイルの指定)

アノテートするコンフィギュレーション前の BSDL ファイルを指定します。

構文

```
-s [IEEE1149|IEEE1532]
```

IEEE 1149 または IEEE 1532 のいずれかに設定できます。ほとんどの場合、IEEE 1149 に設定する必要があります。

BSDL ファイルの構成と BSDLAAnno による変換

該当デバイスの BSDL ファイルは、JTAG 準拠デバイスの製造者から提供されます。BSDL ファイルには、JTAG 準拠デバイスのバウンダリ スキャン アーキテクチャが VHDL のサブセット言語で記述されています。このセクションでは、IEEE 1149 BSDL ファイルの主要な部分と、各セクションが BSDLAAnno によりどのように変更されるかについて説明します。

- ・ [BSDL ファイルのエンティティ宣言](#)
- ・ [BSDL ファイルのジェネリック パラメータ](#)
- ・ [BSDL ファイルの論理ポート記述](#)
- ・ [BSDL ファイルのパッケージ ピンのマップ](#)
- ・ [BSDL ファイルの use 文](#)
- ・ [BSDL ファイルのスキャン ポート識別](#)
- ・ [BSDL ファイルの TAP 記述](#)
- ・ [BSDL ファイルのバウンダリ レジスタ記述](#)
- ・ [シングルエンド ピンの BSDL ファイルの変更](#)
- ・ [差動ピンの BSDL ファイルの変更](#)
- ・ [DESIGN_WARNING セクション](#)
- ・ [ヘッダ コメント](#)

BSDL ファイルのエンティティ宣言

BSDL ファイルのエンティティ宣言は、デバイス名を定義する VHDL 構文です。

BSDL ファイルのジェネリック パラメータ

BSDL ファイルのジェネリック パラメータは、記述されているパッケージを指定します。

ジェネリックパラメータの例

```
generic (PHYSICAL_PIN_MAP : string := "FF324" );
```

BSDLAAnno ではジェネリック パラメータは変更されません。

BSDL ファイルの論理ポート記述

BSDL ファイルの論理ポート記述は、次を示します。

- ・ デバイスのすべての I/O
- ・ バウンダリ スキャンに使用するピンの方向 (入力、出力、双方向、または使用不可)

出力として設定されたピンは、出力としてのみ使用される場合でも、入力バウンダリ スキャンセルが接続されたままになるため、inout と記述されます。出力ピンを inout と定義することで、実際のバウンダリ スキャン機能が反映され、テストを効果的に実施できます。

パッケージによっては、一部の I/O が使用できないことがあります。使用されない I/O は、コンフィギュレーション前の BSDL ファイルで linkage bit (連鎖ビット) として定義されます。

BSDL ファイルの論理ポート記述例

```
port (  
  AVDD_H10: linkage bit;  
  AVSS_H9: linkage bit;  
  CCLK_N8: inout bit;  
  CS_B_R16: in bit;  
  DONE_P8: inout bit;  
  DOUT_BUSY_T6: out bit;  
  D_IN_R7: in bit;  
  GND: linkage bit_vector (1 to 44);  
  HSWAP_EN_T17: in bit;  
  INIT_B_M8: inout bit;
```

論理ポート記述は、コンフィギュレーション後のバウンダリ スキャン回路の機能に適合するよう BSDLAAnno により変更されます。ピンの変更は次のとおりです。

- ・ 専用ピン (JTAG、MODE、DONE など) は変更されず、inout bit のままです。
- ・ 双方向として定義されているピンは、inout bit のままです。
- ・ 入力として定義されているピンは、inout bit に変更されます。
- ・ 出力として定義されているピンは、inout bit のままです。
- ・ 使用されないピンは、変更されません。
- ・ 差動ペアの N ピンは、inout bit に変更されます。

BSDL ファイルのパッケージ ピンのマップ

BSDL ファイルのパッケージ ピンのマップは、デバイス チップ上のパッドとデバイス パッケージ上のピンの接続方法を示します。

BSDL ファイルのパッケージ ピンのマップ例

```
"AVDD_H10:H10," &  
"AVSS_H9:H9," &  
"CCLK_N8:N8," &  
"CS_B_R16:R16," &  
"DONE_P8:P8," &  
"DOUT_BUSY_T6:T6," &  
"D_IN_R7:R7," &
```

BSDLAAnno ではパッケージ ピンのマップは変更されません。

BSDL ファイルの use 文

use 文は、BSDL ファイルで参照される属性、タイプ、制約を含む VHDL パッケージを呼び出します。

構文

```
use vhdl_package ;
```

例

```
use STD_1149_1_1994.all;
```

BSDLAAnno では use 文は変更されません。

BSDL ファイルのスキャン ポート識別

スキャン ポート識別では、次の JTAG ピンが識別されます。

- ・ TDI
- ・ TDO
- ・ TMS
- ・ TCK
- ・ TRST

TRST はオプションの JTAG ピンであり、ザイリンクス デバイスでは使用されません。

BSDLAAnno ではスキャン ポート識別は変更されません。

BSDL ファイルのスキャン ポート識別の例

```
attribute TAP_SCAN_IN of TDI : signal is true;
attribute TAP_SCAN_MODE of TMS : signal is true;
attribute TAP_SCAN_OUT of TDO : signal is true;
attribute TAP_SCAN_CLOCK of TCK : signal is (33.0e6, BOTH);
```

BSDL ファイルの TAP 記述

TAP 記述は、デバイスの JTAG ロジックに関する次の追加情報を示します。

- ・ 命令レジスタ長
- ・ 命令 OPCODE
- ・ デバイス IDCODE

これらの特性はデバイスによって異なります。

BSDLAAnno では TAP 記述は変更されません。

BSDL ファイルの TAP 記述の例

```
-- Compliance-Enable Description

attribute COMPLIANCE_PATTERNS of test : entity is
    "(PROG_B) (1)";

-- Instruction Register Description

attribute INSTRUCTION_LENGTH of test : entity is 10;
```

BSDL ファイルのバウンダリ レジスタ記述

バウンダリレジスタ記述は、デバイス上のバウンダリ スキャン セルの構造を定義します。各ピンには、レジスタとラッチから構成されたバウンダリ スキャン セルが最低 3 つずつあります。バウンダリ スキャン テスト ベクタは、これらのレジスタに読み込まれるか、レジスタからスキャンされます。

BSDL ファイルのバウンダリ レジスタ記述の例

```
attribute BOUNDARY_REGISTER of test : entity is
-- cellnum (type, port, function, safe[, ccell, disval, disrslt])
" 0 (BC_1, *, internal, X)," &
" 1 (BC_1, *, internal, X)," &
" 2 (BC_1, *, internal, X)," &
" 3 (BC_1, *, internal, X)," &
" 4 (BC_1, *, internal, X)," &
" 5 (BC_1, *, internal, X)," &
" 6 (BC_1, *, internal, X)," &
```

各 IOB には、次の 3 つバウンダリ スキャン レジスタが関連付けられています。

- ・ 制御
- ・ 出力
- ・ 入力

バウンダリ レジスタ記述は、「[シングル エンド ピンの BSDL ファイルの変更](#)」および「[差動ピンの BSDL ファイルの変更](#)」で説明しているように、BSDLANno により変更されます。

シングル エンド ピンの BSDL ファイルの変更

このセクションでは、BSDLANno によるシングルエンドピンの BSDL ファイルの変更について説明します。次の内 容が含まれます。

- ・ シングルエンド ピンの BSDL ファイルの変更について
- ・ シングルエンド トライステート出力ピンの BSDL ファイル例
- ・ シングルエンド入力ピンの BSDL ファイル例
- ・ シングルエンド出力ピンの BSDL ファイル例
- ・ 設定されていないピンまたは使用されていないピンの BSDL ファイル例

シングルエンド ピンの BSDL ファイルの変更について

シングルエンドピンは、ピンを入力として設定した場合のみ変更されます。この場合、バウンダリ スキャン ロジックが出力ドライバから切断され、ピンを駆動することができなくなります。ピンを出力として設定した場合、バウンダリ スキャン入力レジスタはピンに接続されたままの状態になり、バウンダリ スキャン ロジックの機能はピンを双方向として設定した場合と同じになります。

シングルエンド トライステート出力ピンの BSDL ファイル例

ピン 57 をシングルエンドのトライステート出力ピンとして設定した場合、コードは変更されません。

```
-- TRISTATE OUTPUT PIN (three state output with an input component)
" 9 (BC_1, *, controlr, 1)," &
" 10 (BC_1, PAD57, output3, X, 9, 1, Z)," &
" 11 (BC_1, PAD57, input, X)," &
```

シングルエンド入力ピンの BSDL ファイル例

ピン 57 をシングルエンドの入力として設定した場合、次のように変更されます。

```
-- PIN CONFIGURED AS AN INPUT
" 9 (BC_1, *, internal, 1)," &
" 10 (BC_1, *, internal, X)," &
" 11 (BC_1, PAD57, input, X)," &
```

シングルエンド出力ピンの BSDL ファイル例

ピン 57 をシングルエンドの出力として設定した場合、シングルエンドの双方向ピンと見なされます。

```
-- PIN CONFIGURED AS AN OUTPUT
" 9 (BC_1, *, controlr, 1)," &
" 10 (BC_1, PAD57, output3, X, 9, 1, Z)," &
" 11 (BC_1, PAD57, input, X)," &
```

設定されていないピンまたは使用されていないピンの BSDL ファイル例

ピン 57 を設定しない場合や使用しない場合は、変更されません。

```
-- PIN CONFIGURED AS "UNUSED"
" 9 (BC_1, *, controlr, 1)," &
" 10 (BC_1, PAD57, output3, X, 9, 1, PULL0)," &
" 11 (BC_1, PAD57, input, X)," &
```

差動ピンの BSDL ファイルの変更

このセクションでは、BSDLAAnno による差動ピンの BSDL ファイルの変更について説明します。次の内容が含まれます。

- ・ 差動ピンの BSDL ファイルの変更について
- ・ 差動出力ピン、差動トライステート出力ピン、差動双方向ピンの BSDL ファイル例
- ・ P 側の差動入力ピンの BSDL ファイル例
- ・ N 側の差動入力ピンの BSDL ファイル例

差動ピンの BSDL ファイルの変更について

差動ピン ペアとの通信は、P 側のピンに接続されたバウンダリ スキャン セルによって処理されます。差動ペアの値は、P 側の入力レジスタをスキャンして読み取ります。差動ペアに値を駆動するには、P 側の出力レジスタに値をシフトします。N 側のスキャン レジスタの値は、このピンには影響しません。

ほとんどのバウンダリ スキャン デバイスでは、差動ペアごとに 3 つのバウンダリ スキャン レジスタが使用され、バウンダリ スキャンで個別のピンが直接制御されるのではなく、差動ペアが制御されます。2 つのピンのペアで 1 ビットのデータが転送されるので、1 つの入力レジスタ、出力レジスタ、制御レジスタのみが必要です。

各ピンにバウンダリ スキャン セルが 3 つ、各差動ペアに 6 つのレジスタがあります。N 側のレジスタはバウンダリ スキャン レジスタ内に残りますが、ピンには接続されていないため、コンフィギュレーション後の BSDL ファイルで、内部レジスタとしてリストされます。N 側のピンのベヘビヤは、P 側のバウンダリ スキャン レジスタによって制御されます。たとえば、P 側の出力スキャン レジスタに値を配置し、出力をイネーブルにすると、出力ドライバによって反転値が N ピンに駆動されます。これは、バウンダリ スキャン ロジックとは独立した動作です。

差動出力ピン、差動トライステート出力ピン、差動双方向ピンの BSDL ファイル例

ピン 57 を差動出力、差動トライステート出力、または差動双方向として設定した場合、次のように変更されます。

```
" 9 (BC_1, *, controlr, 1)," &
" 10 (BC_1, PAD57, output3, X, 9, 1, Z)," &
" 11 (BC_1, PAD57, input, X)," &
```

P 側の差動入力ピンの BSDL ファイル例

ピン 57 を P 側の差動ピンとして設定した場合、次のように変更されます。

```
" 9 (BC_1, *, internal, 1)," &
" 10 (BC_1, *, internal, X)," &
" 11 (BC_1, PAD57, input, X)," &
```

N 側の差動入力ピンの BSDL ファイル例

ピン 57 を N 側の差動ピン (入力、出力、トライステート出力、双方向のすべてのタイプ) として設定した場合、次のように変更されます。

```
" 9 (BC_1, *, internal, 1)," &
" 10 (BC_1, *, internal, X)," &
" 11 (BC_1, *, internal, X)," &
```

DESIGN_WARNING セクション

BSDLAAnno は、BSDL ファイルに次の DESIGN_WARNING を追加します。

```
This BSDL file has been modified to reflect post-configuration"&
behavior by BSDLAAnno. BSDLAAnno does not modify the USER1,"&
USER2, or USERCODE registers. For details on the features and"&
limitations of BSDLAAnno, please consult the Xilinx Development"&
System Reference Guide.";
```

ヘッダ コメント

BSDLAAnno は、BSDL ファイルのヘッダに次のコメントを追加します。

- ・ BSDLAAnno Post-Configuration File for design [entity name]
- ・ BSDLAAnno [BSDLAAnno version number]

ザイリンクス デバイスのバウンダリ スキャン ビヘイビア

ザイリンクスの BSDL ファイルには、コンフィギュレーションされていないデバイスのバウンダリ スキャン ビヘイビアが反映されています。コンフィギュレーション後、このビヘイビアは変更される場合があります。たとえば、コンフィギュレーション前に双方向であった I/O ピンが入力のみになったりすることがあります。バウンダリ スキャン テスト ベクタは、通常 BSDL ファイルから読み取られるので、コンフィギュレーション済みのデバイスでバウンダリ スキャン テストを実施する場合は、コンフィギュレーション後のビヘイビアを反映させて BSDL ファイルを変更してください。

コンフィギュレーション前のデバイスでは、すべての I/O に双方向スキャン ベクタを使用でき、テストの対象となる範囲がより大きいという利点があるため、できる限りコンフィギュレーション前のデバイスでテストを実施してください。

ザイリンクス デバイスを使用したバウンダリ スキャン テストをコンフィギュレーション後に実施できるのは、次のような場合のみです。

- ・ コンフィギュレーションが避けられない場合
- ・ 差動信号標準を使用した場合。ただし、差動信号がデバイス間に配置されている場合は、両方のデバイスをコンフィギュレーション前にテストできます。この場合、差動ペアの両側がシングルエンドの信号として動作します。

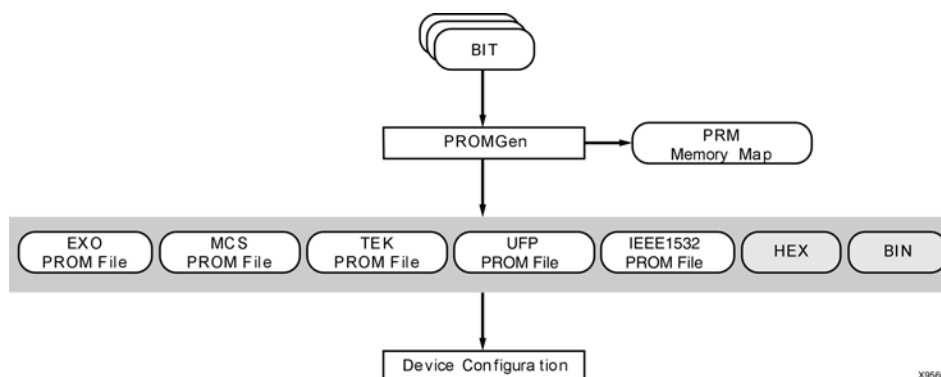
PROMGen

この章には、次のセクションが含まれています。

- ・ [PROMGen の概要](#)
- ・ [PROMGen の構文](#)
- ・ [PROMGen のオプション](#)
- ・ [PROM ファイルのビット スワップ](#)
- ・ [PROMGen の例](#)

PROMGen の概要

PROMGen プログラムは、BitGen により生成されたコンフィギュレーション ビットストリーム (BIT) ファイルを PROM フォーマットのファイルに変換します。PROM ファイルには、FPGA デバイスのコンフィギュレーションデータが含まれています。PROMGen は BIT ファイルを PROM フォーマットまたはマイクロプロセッサと互換性のあるフォーマット ([「-p \(PROM フォーマット\)」](#)を参照) に変換します。次に、PROMGenP プログラムの入力ファイルと出力ファイルを示します。



PROMGen には、機能的には同じ 2 つのバージョンがあります。オペレーティング システムのプロンプトからアクセスできるスタンドアロン バージョンと、Project Navigator から起動できるバージョン (iMPACT) です。iMPACT の詳細は、iMPACT ヘルプを参照してください。

PROMGen を使用すると、ビットストリーム ファイルをデジタイズ チェーン接続した FPGA 用に結合することも可能です。

メモ： デスティネーション PROM がザイリンクスのシリアル PROM で、ザイリンクスの PROM プログラマを使用しており、FPGA がデジタイズ チェーン接続されない場合は、PROM ファイルを作成する必要はありません。

PROMGen のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6

PROMGen の入力ファイル

PROMGen の入力ファイルは、次のとおりです。

- ・ **BIT ファイル** : FPGA デザインのコンフィギュレーション データが含まれます。
- ・ **ELF (MEM) ファイル** : BMM ファイルで指定されたブロック RAM の値を設定します (オプション)。
- ・ **RBT (ロービット) ファイル** : ビットストリーム ファイルのデータを ASCII 形式の 1 と 0 で表します。

PROMGen の出力ファイル

PROMGen の出力ファイルは、次のとおりです。

- ・ **PROM ファイル** : PROM コンフィギュレーション情報を含むファイル。詳細は、「[-p \(PROM フォーマット\)](#)」を参照してください。
- ・ **PRM ファイル** : 拡張子が .prm の PROM イメージ ファイル。このファイルには、出力 PROM ファイルのメモリ マップが含まれています。拡張子は .prm です。
- ・ **CFI ファイル** : XCFP PROM で使用するファイル。
- ・ **SIG ファイル** : 自動署名プログラム用にデバイスの署名を格納するファイル。

PROMGen の構文

PROMGen を実行するには、次の構文を使用します。

promgen [*options*]

options : 「[PROMGen のオプション](#)」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

メモ : **-r**、**-u**、**-d**、または **-ver** のうち少なくとも 1 つを使用する必要があります。

PROMGen のオプション

このセクションでは、PROMGen のコマンドライン オプションについて説明します。

- ・ `-b` (ビット スワップをオフ)
- ・ `-bd` (データ ファイルの指定)
- ・ `-bm` (BMM ファイルの指定)
- ・ `-bpi_dc` (シリアルまたはパラレル デイジー チェーン接続)
- ・ `-c` (チェックサム)
- ・ `-config_mode` (コンフィギュレーション モード)
- ・ `-d` (下方向に読み込み)
- ・ `-data_file` (データ ファイルを追加)
- ・ `-data_width` (PROM データ幅の指定)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-i` (初期バージョンを選択)
- ・ `-intstyle` (統合スタイル)
- ・ `-l` (レングス カウントのディスエーブル)
- ・ `-n` (BIT ファイルの追加)
- ・ `-o` (出力ファイル名)
- ・ `-p` (PROM フォーマット)
- ・ `-r` (PROM ファイルの読み込み)
- ・ `-s` (PROM のサイズ)
- ・ `-spi` (ビット スワップをオフ)
- ・ `-t` (テンプレート ファイル)
- ・ `-u` (上方向に読み込み)
- ・ `-ver` (バージョン)
- ・ `-w` (既存の出力ファイルの上書き)
- ・ `-x` (ザイリンクス PROM の指定)
- ・ `-z` (圧縮をイネーブル)

`-b` (ビット スワップをオフ)

HEX および BIN ファイルでのビット スワップをオフにします。

デフォルト (`-b` オプションなし) では、入力 BIT ファイルのビットがスワップされて HEX ファイルまたは BIN ファイルに書き込まれます。`-b` オプションを使用すると、ビットはスワップされません。ビットのスワップについては、「[PROM ファイルのビット スワップ](#)」を参照してください。

構文

`-b`

メモ: `-p` オプションで PROMGen の出力として HEX ファイルまたは BIN ファイルを指定した場合にのみ適用されます。

-bd (データ ファイルの指定)

出力 PROM ファイルに含めるデータ ファイルを指定します。ELF および MEM ファイルを使用できます。拡張子を指定しない場合は、ELF ファイルが使用されます。

構文

```
-bd filename [.elf|.mem] [start hexaddress]
```

データ ファイルには、開始アドレスがある場合があります。開始アドレスを指定すると、データ ファイルはそのアドレスから読み込まれます。開始アドレスを指定しない場合は、前のデータ ファイルの後に読み込まれます。

メモ：データ ファイルは、上方向に読み込まれます。BIT ファイルに適用されるメモリ サイズ チェックはデータ ファイルにも適用され、データ ファイルが指定の場所に収まるかどうかをチェックされます。

-bm (BMM ファイルの指定)

-bd オプションで指定したデータ ファイルのビット/バイト順を含むメモリ マップ ファイル (BMM) を指定します。

構文

```
-bm filename
```

-bpi_dc (シリアルまたはパラレル デイジー チェーン接続)

BPI または SelectMAP モードで接続された最初の FPGA から、シリアルまたはパラレル デイジー チェーン出力を選択します。

メモ：Spartan®-3 および Virtex®-4 デバイスではシリアル デイジー チェーンは使用できません。

構文

```
-bpi_dc serial|parallel
```

-c (チェックサム)

PRM ファイルのチェックサム値を生成します。この値は PROM プログラムのチェックサムと一致する必要があります。このオプションは、PROM に読み込むデータが正しいかどうかを検証するために使用します。

構文

```
-c
```

-config_mode (コンフィギュレーション モード)

SelectMAP コンフィギュレーション データ バス インターフェイスのサイズを 8、16、または 32 ビットに設定します。

構文

```
-config_mode selectmap8|selectmap16|selectmap32
```

-d (下方向に読み込み)

1 つ以上の BIT ファイルが開始アドレスから下方向に読み込まれるよう指定します。複数のファイルを指定すると、ファイルがデイジー チェーン接続されます。複数の **-d** オプションを使用すると、異なるアドレスにファイルを読み込むことができます。このオプションは、入力ビットストリーム ファイル名の前に入力してください。

構文

```
-d hexaddress0 filename filename
```

複数のファイルを指定するには、次のような構文を使用します。

```
promgen -d hexaddress0 filename filename
```

複数の **-d** オプションを指定するには、次のような構文を使用します。

```
promgen -d hexaddress1 filename -d hexaddress2 filename...
```

-data_file (データ ファイルを追加)

PROM ファイルにデータを追加する方法、開始アドレス、データ ファイル名を指定します。ファイルは PROM にそのまま追加され、フォーマットは変更されません。

構文

```
-data_file up|down hex_address file [ file ... ]
```

up : ファイルを指定のアドレスから上方向に読み込むよう指定します。

down : ファイルを指定のアドレスから下方向に読み込むよう指定します。

hex_address : ファイルの読み込みを開始するアドレスを 16 進数で指定します。

file : 読み込むファイルを指定します。複数のファイルを指定できます。ファイルを複数指定する場合は、スペースで区切ります。ファイルはリストされた順に読み込まれます。

-data_width (PROM データ幅の指定)

PROM のデータ幅を指定します。たとえば、「**-data_width 8**」と指定すると、PROM はバイト幅になります。

構文

```
-data_width 8|16|32
```

データ幅を 16 または 32 に設定すると、次のように処理されます。

- ・ PROM のアドレス空間が、指定したデータ幅 (16 または 32) に応じて 2 倍または 4 倍に拡張されます。
- ・ ビットストリームのビット順およびバイト順が、Virtex®-4、Virtex-5、Virtex-6、および Spartan®-6 デバイス ファミリー用の順序に変更されます。

デフォルトでは 8 に設定されています。

メモ : 拡張されたアドレス空間は、BIT ファイルおよびデータ ファイルの両方に適用されません。ビット順およびバイト順の変更は、一部の BIT ファイルのみに適用され、データ ファイルには適用されません。

各オプション値は、次のアーキテクチャで使用可能です。

- ・ **-data_width 8** : サポートされるすべての FPGA アーキテクチャで使用可能です。
- ・ **-data_width 16** : Virtex-5、Virtex-6、および Spartan-6 デバイスで使用可能です。
- ・ **-data_width 32** : Virtex-4、Virtex-5、Virtex-6、Spartan-6 デバイスで使用可能です。

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f \(コマンド ファイルの実行\)](#)」を参照してください。

-i (初期バージョンを選択)

ザイリンクス マルチバンク PROM の初期バージョンを指定します。

構文

-i *version*

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle *ise|xflow|silent*

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-l (レンジス カウントのディスエーブル)

FPGA ビットストリームのレンジス カウンタをディスエーブルにします。レンジス カウンタで指定されている 24 ビットの制限を超えるビットストリームを連結する場合に、このオプションを使用します。

構文

-l

-n (BIT ファイルの追加)

直前の読み込みからその次に読み込み可能なアドレスに 1 つ以上の BIT ファイルを読み込みます。**-n** オプションでは方向が指定できないので、最初の **-n** オプションの前に **-u** オプションまたは **-d** オプションを指定する必要があります。このオプションで指定したファイルは、以前のファイルにデジター チェーン接続されません。ファイルは、その前の **-d**、**-u**、**-n** オプションで設定された方向に読み込まれます。

構文

```
-n file1[.bit] file2[.bit]...
```

複数のファイルを指定するには、次の構文を使用します。この構文を使用すると、ファイルがデジター チェーン接続されます。

```
promgen -d hexaddress file0 -n file1 file2...
```

複数の **-n** オプションを使用するには、次の構文を使用します。この構文を使用すると、ファイルはデジター チェーン接続されません。

```
promgen -d hexaddress file0 -n file1 -n file2...
```

-o (出力ファイル名)

PROM の出力ファイル名にデフォルトとは異なる名前を指定します。出力ファイル名を指定しない場合は、PROM ファイルの名前は最初に読み込まれた BIT ファイルと同じ名前になります。

構文

```
-o file1[.ext] file2[.ext]
```

.ext は、該当する PROM フォーマットの拡張子を表します。

複数のファイル名を指定すると、複数のファイルに情報を分割できます。複数の PROM ファイルに対して名前が 1 つしか指定されていない場合は、出力 PROM ファイルの名前は *file_#.ext* となります。*file* はベース名、*#* は 0、1、などの番号、*.ext* は該当する PROM フォーマットの拡張子です。

```
promgen -d hexaddress file0 -o filename
```

-p (PROM フォーマット)

PROM のフォーマットを MCS (Intel MCS86)、EXO (Motorola EXORMacs)、TEK (Tektronix TEKHEX)、UFP (ユーザー フォーマット PROM)、または IEEE1532 のいずれかに指定します。

マイクロプロセッサのダウンロードに使用される HEX ファイル (コンフィギュレーション ビットストリームの 16 進表現) または BIN ファイル (コンフィギュレーション ビットストリームの 2 進表現) も作成できます。

構文

```
-p mcs|exo|tek|ufp|ieee1532|hex|bin
```

デフォルトでは MCS に設定されています。

IEEE1532 は、ISP (In-System Programmability) 規格です。PROMGen では、ヘッダを含み、IEEE1532 規格で指定されたデータ フォーマットで記述された IEEE1532 準拠のファイルが生成されます。

UFP (ユーザー フォーマット PROM) では、PROM ファイル テンプレート (PFT) ファイルでパラメータを定義できます。`default.pft` ファイルが `$XILINX/data` ディレクトリに含まれており、バイト順、ワードごとのバイト数、データ区切り文字など多数のパラメータを制御できます。

-r (PROM ファイルの読み込み)

BIT ファイルの代わりに既存の PROM ファイルが読み込まれるよう指定します。PROMGen のすべての出力オプションが使用できるので、**-r** オプションを使用して既存の PROM ファイルを複数の PROM ファイルに分割したり、別のフォーマットに変換できます。

構文

```
-r promfile
```

メモ : **-r** を使用する場合、**-d**、**-u**、および **-n** オプションは使用できません。

-s (PROM のサイズ)

PROM のサイズをキロバイト単位で指定します。PROM のサイズは、2 のべき乗で指定する必要があります。デフォルトでは 64 キロバイトに設定されています。**-s** オプションは、**-u**、**-d**、**-n** オプションの前に指定する必要があります。

構文

```
-s promsize1 [ promsize2 ... ]
```

promsize に複数のサイズを指定すると、PROM が複数の PROM ファイルに分割されます。

メモ : ソフトウェア ツールを使用してチェーンに含まれる PROM を設定し、PROM ファイルを作成して、PRM レポートでこれらのオプションがどのように使用されているかを確認してください。

-spi (ビット スワップをオフ)

SPI フラッシュ デバイスとの互換性のため、ビット スワップをオフにします。

構文

```
-spi
```

-t (テンプレート ファイル)

UFP (ユーザー フォーマット PROM) のテンプレート ファイルを指定します。ファイルを指定しない場合、デフォルトで `$XILINX/data/default.pft` が使用されます。UFP フォーマットを指定した場合、**-t** オプションを使用して制御ファイルを指定します。

構文

```
-t templatefile.pft
```


-u (上方向に読み込み)

1 つ以上の BIT ファイルが開始アドレスから上方向に読み込まれるよう指定します。複数のファイルを指定すると、ファイルがデイジー チェーン接続されます。複数の **-u** オプションを使用すると、異なるアドレスにファイルを読み込むことができます。

構文

```
-u hexaddress0 filename1 filename2...
```

このオプションは、入力ビットストリーム ファイルの直前に指定する必要があります。

-ver (バージョン)

指定した 16 進数アドレスから BIT ファイルを読み込むよう指定します。複数の BIT ファイルはデイジー チェーン接続され、1 つの PROM ロードになります。このデイジー チェーンは、PROM 内の特定バージョンに割り当てられます。

メモ: このオプションは、ザイリンクスのマルチバンク PROM のみに使用できます。

構文

```
-ver [version] hexaddress filename1.bit filename2.bit...
```

-w (既存の出力ファイルの上書き)

既存の出力ファイルを上書きします。出力ファイルが既に存在する場合は、このオプションを使用する必要があります。使用しないと、エラーが発生します。

構文

```
-w
```

-x (ザイリンクス PROM の指定)

PROM ファイルのターゲットとなるザイリンクスのシリアル PROM を指定します。使用するザイリンクス PROM がわかっている場合は、**-s** オプションの代わりにこのオプションを使用します。

構文

```
-x xilinx_prom1 [ xilinx_prom2 ... ]
```

複数の PROM (*xilinx_prom*) を指定すると、PROM が複数の PROM ファイルに分割されます。

メモ: ソフトウェア ツールを使用してチェーンに含まれる PROM を設定し、PROM ファイルを作成して、PRM レポートでこれらのオプションがどのように使用されているかを確認してください。

-z (圧縮をイネーブル)

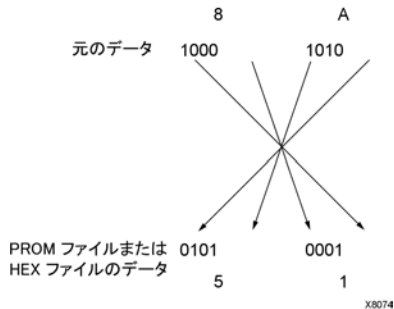
ザイリンクスのマルチバンク PROM の圧縮をイネーブルにします。バージョンを指定しない場合は、すべてのバージョンが圧縮されます。

構文

```
-z version
```

PROM ファイルのビット スワップ

PROMGen で生成される PROM ファイルでは、入力 BIT ファイルのバイト内のビットがスワップされています。このビット スワップはビット反転とも呼ばれ、次の図に示すように各バイト内のビットが入れ替えられます。



BIT ファイルに含まれるビットストリームでは、最下位ビット (LSB) は常にバイトの左側にありますが、PROM プログラムまたはマイクロプロセッサでは、バイトの右側を LSB と見なしてデータ バイトを読み込みます。そのため、PROM プログラムやマイクロプロセッサでビットストリームを正しく読み取るには、各バイトのビットをスワップする必要があります。

今リリースの ISE® Design Suite では、すべての出力フォーマット (MCS、EXO、TEK、UFP、IEEE1532、HEX、および BIN) でビットがスワップされます。HEX および BIN ファイルでは、デフォルトでビット スワップがオンになっていますが、PROMGen の **-b** オプションを使用してオフにできます。

PROMGen の例

ファイルを上方方向に読み込む

次の構文は、ファイル test.bit を MCS フォーマットのアドレス 0x0000 から上方方向に読み込みます。

```
promgen -u 0 test
```

ファイルをデジター チェーン接続

32K PROM と Motorola EXOrmacs フォーマットを使用して、アドレス 0x0000 からファイル test1.bit および test2.bit を上方方向に、アドレス 0x4000 からファイル test3.bit および test4.bit を上方方向にデジター チェーン接続するには、次のように指定します。

```
promgen -s 32 -p exo -u 00 test1 test2 -u 4000 test3 test4
```

ファイルを下方方向に読み込む

次の構文は、ザイリンクス XC1718D PROM を使用して、PROM プログラムにアドレス 0x400 から下方方向にファイル test.bit を読み込みます。

```
promgen -x xc1718d -u 0 test
```

デフォルト以外のファイル名を指定

次の構文は、デフォルトのファイル名とは異なる PROM ファイル名を指定します。

```
promgen options filename -o newfilename
```

IBISWriter

この章では、IBISWriter プログラムについて説明します。次のセクションが含まれています。

- ・ [IBISWriter の概要](#)
- ・ [IBISWriter の構文](#)
- ・ [IBISWriter のオプション](#)

IBISWriter の概要

IBIS (Input/Output Buffer Information Specification) はデバイス モデル規格で、デバイス インターコネクトの信号ビヘイビアを記述するビヘイビア モデルを開発できます。これらのモデルでは、SPICE シミュレーションなどで生成された構造モデルとは異なり、周辺の回路情報が保持されます。IBIS バッファ モデルは、測定または回路シミュレーションで得られた V/I 曲線データに基づいて作成されます。

IBIS モデルは各 IOB 規格に対して作成され、IBIS ファイルにはデバイスのすべての I/O 規格の IBIS モデルが含まれます。IBIS ファイルには、I/O 規格をサポートするためにコンフィギュレーションされた IOB に接続されているデバイス上の使用ピンのリストも含まれており、これらのピンと IBIS バッファ モデルとの関連性を示します。

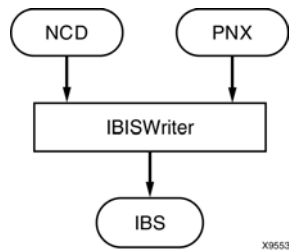
IBISWriter を使用すると、DCI (デジタル制御インピーダンス) をユーザーの選択した参照抵抗で使用できます。ユーザー入力すべてに対して IBIS モデルを含めることはできませんが、LVCMOS15 から LVCMOS33 の I/O 規格 (インピーダンス 40、50、および 65 Ω) の IBIS モデルがあります。デフォルトの抵抗値は 50 Ω です。

IBIS 規格は、出力情報ファイルのフォーマットを規定しており、出力ファイルはファイル ヘッダとコンポーネントの記述から構成されます。IBIS モデル ファイルの構文が IBIS データフォーマットに準拠しているかどうかを確認する Golden Parser というツールが、IBIS Open Forum Group (<http://www.eigroup.org/ibis>) により開発されています。

IBISWriter には、入力としてデザイン ソース ファイルを使用する必要があります。FPGA デザインの場合は、デザインの物理記述を含む NCD ファイル (拡張子 `.ncd`) を使用します。CPLD デザインの場合は、CPLDFit により生成されたファイル (拡張子 `.pnx`) を使用します。

IBISWriter から出力される IBS ファイルには、デザインで使用されるピン、これらのピンに接続されているデバイス内部の信号、およびピンに接続されている IOB に適用される IBIS バッファ モデルのリストが含まれます。

IBISWriter フロー



IBISWriter のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

IBISWriter の入力ファイル

IBISWriter には、入力としてデザイン ソース ファイルを使用する必要があります。

- ・ FPGA デザイン
デザインの物理記述を含む NCD ファイル (拡張子 `.ncd`) を使用します。
- ・ CPLD デザイン
CPLDFit により生成されたファイル (拡張子 `.pnx`) を使用します。

IBISWriter の出力ファイル

IBISWriter から出力される ASCII 形式の IBS ファイルには、デザインで使用されるピン、これらのピンに接続されているデバイス内部の信号、およびピンに接続されている IOB に適用される IBIS バッファ モデルのリストが含まれます。IBIS 出力ファイルのフォーマットは IBIS 規格で規定されており、ファイルのフォーマットがその仕様に準拠していることを確認するため、Golden Parser で検証する必要があります。

メモ： I/O 規格を使用するピンがあるためにバッファが使用されていないか、DCI プロパティがあるためにバッファ モデルが使用されていない場合、エラー メッセージが表示されます。エラー メッセージが表示されても作業は続行され、ほかにもエラーがないかどうかリストされた後、IBS 出力ファイルが作成されずに IBISWriter が終了します。このエラー メッセージ表示機能によって、エラーを事前に検知し、IBISWriter を再実行する前に問題を修正できます。

IBISWriter の構文

IBISWriter を実行するには、次の構文を使用します。

```
ibiswriter [options] infile outfile[.ibs]
```

- ・ *options*: 「IBISWriter のオプション」にリストされているオプションを 1 つ以上指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。
- ・ *infile*: デザインのソース ファイルです。FPGA デザインの場合、拡張子が `.ncd` のファイルを指定する必要があります。CPLD デザインの場合は、CPLDFit により生成されたファイル (拡張子 `.pnx`) を使用します。
- ・ *outfile*: デザインの IBIS ファイル名です。拡張子の `.ibs` は省略してもかまいません。IBIS ファイル名は、拡張子 `.ibs` も含めて 24 文字以内で指定してください。

IBISWriter のオプション

このセクションでは、IBISWriter のコマンドライン オプションについて説明します。

- ・ `-allmodels` (アーキテクチャに使用できるバッファ モデルをすべて含む)
- ・ `-g` (参照電圧の設定)
- ・ `-intstyle` (統合スタイル)
- ・ `-ml` (複数言語のサポート)
- ・ `-pin` (パッケージ寄生情報の生成)
- ・ `-truncate` (出力ファイルでの信号名の最大長を指定)
- ・ `-vccaux` (VCCAUX 電圧レベルを設定)

`-allmodels` (アーキテクチャに使用できるバッファ モデルをすべて含む)

IBISWriter は、IBS 出力ファイルのサイズを削減するため、デザイン特有のバッファ モデルのみを含む出力ファイル (アクティブ ピンリストにより決定) を生成します。このオプションを指定すると、使用可能なバッファ モデルすべてにアクセスできます。

構文

```
-allmodels
```

`-g` (参照電圧の設定)

サポートされるアーキテクチャとオプション値は、次のとおりです。

アーキテクチャ	オプション	値	説明
XC9500	VCCIO	LVTTL、TTL	I/O を 3.3V (LVTTL) または 5V (TTL) の VCCIO 参照電圧用にコンフィギュレーションします。 <code>-g</code> オプションは、必ず使用してください。
XC9500XL	VCCIO	LVC MOS2、LVTTL	出力を 3.3V (LVTTL) または 2.5V (LVC MOS2) の VCCIO 参照電圧用にコンフィギュレーションします。ユーザー ピンは、5V、3.3V、2.5V の入力に対応しています。 <code>-g</code> オプションは、必ず使用してください。

構文

```
-g option_value_pair
```

VCCIO オプションを LVTTTL に設定する例

```
-g VCCIO:LVTTTL design.ncd design.ibs
```

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

```
-intstyle ise|xflow|silent
```

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-ml (複数言語のサポート)

SPICE ファイルなど外部ファイルへ参照するため、複数の言語がサポートされるよう指定します。

構文

```
-ml
```

-pin (詳細なピンごとのパッケージ寄生情報の生成)

指定のデバイス/パッケージの組み合わせにピンごとのパッケージ寄生情報が存在する場合に、その情報を含めます。

このオプションを使用すると、抵抗、インピーダンス、およびキャパシタンスを表す RLC 寄生情報 (疎行列形式) を含むセクションが出力ファイルの各パッケージ ピンに追加されます。これにより、生成されたモデルにパッケージに関するより詳細な情報が含まれるようになるので、タイミング シミュレーションおよびシグナル インテグリティ シミュレーションの精度が向上します。

構文

```
-pin
```

このオプションは、デフォルトではディスエーブルになっています。

-truncate (出力ファイルでの信号名の最大長を指定)

生成されたモデルでの信号名の最大長を指定します。

IBIS 仕様では、当初の制限は 20 文字以内でしたが、現在では 40 文字まで許容されるようになっています。シグナル インテグリティ シミュレータでサポートされるバージョンに応じて、この設定を調整してください。デフォルトでは、IBIS バージョン 3.2 の仕様に準拠するよう信号名は 20 文字に切り詰められます。この際、バスの各信号のインデックスが保持されるなど、各信号名が固有なものになるように切り詰められます。

構文

-truncate [20|40|no]

20 (デフォルト) : 信号名が 20 文字に制限されます。

40 : 信号名が 40 文字に制限されます。

no : 信号名の長さは制限されません。

-vccaux (VCCAUX 電圧レベルを設定)

複数の電圧が許容されるファミリにおいて、VCCAUX 電源に供給する電圧を指定します。

メモ : このオプションは、Spartan®-3A、Spartan-3A DSP、および Spartan-6 デバイスでのみサポートされます。

構文

-vccaux [2.5|3.3|25|33]

デフォルト値は 2.5 です。

CPLDFit

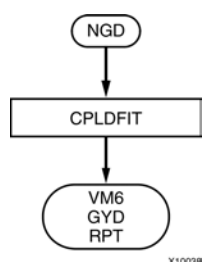
この章では、CPLDFit について説明します。次のセクションが含まれています。

- ・ [CPLDFit の概要](#)
- ・ [CPLDFit の構文](#)
- ・ [CPLDFit のオプション](#)

CPLDFit の概要

CPLDFit プログラムは、NGDBuild により生成された NGD (Native Generic Database) を入力ファイルとして読み込み、CPLD デバイスにデザインをフィットさせるコマンドライン プログラムです。

CPLDFit デザイン フロー



CPLDFit のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

CPLDFit の入力ファイル

CPLDFit に使用する入力ファイルは次のとおりです。

NGD ファイル：NGDBuild により生成されたネイティブ ジェネリック データベース ファイル。デザインが最初に作成されたときに使用された階層と、階層が分割される下位のザイリンクス プリミティブが論理的に記述されています。

CPLDFit の出力ファイル

CPLDFit は、次のファイルを出力します。

- ・ **VM6 ファイル** : CPLDFit のデフォルトの出力ファイルで、Hprep6 および TAEEngine の入力ファイルとして使用します。詳細は、「[Hprep6](#)」の章および「[TAEEngine](#)」の章を参照してください。
- ・ **GYD ファイル** : ピン フリーズや最後に実行したフィットに使用された内部論理式の配置などの情報が含まれたオプションのガイド ファイルです。
- ・ **RPT ファイル** : リソース サマリ、インプリメントされた論理式、デバイスのピン配置、CPLDFit で使用されるコンパイラ オプションが含まれたレポート ファイルです。
- ・ **XML ファイル** : HTML 形式のレポートの生成に使用します。
- ・ **PNX ファイル** : インプリメンテーション デザインの IBIS モデルを作成するため、[IBISWriter](#) で使用します。
- ・ **CXT ファイル** : 消費電力を計算および表示するため、XPower で使用します。
- ・ **MFD ファイル** : デザイン インプリメンテーションをグラフィックで表示するため、HTML レポートで使用します。

CPLDFit の構文

CPLDFit を実行するには、次の構文を使用します。

```
cpldfit infile .ngd [options]
```

infile.ngd : NGD ファイルの名前を指定します。

options : 「[CPLDFit のオプション](#)」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

CPLDFit のオプション

このセクションでは、CPLDFit のコマンドライン オプションについて説明します。

- ・ `-blkfanin` (ファンクション ブロックの最大ファンインを指定)
- ・ `-exhaust` (包括的なフィットを有効)
- ・ `-ignoredatagate` (DATA_GATE 属性を無視)
- ・ `-ignoretspec` (タイミング使用を無視)
- ・ `-init` (パワーアップ値を設定)
- ・ `-inputs` (最適化に使用する入力数)
- ・ `-iostd` (I/O 規格の指定)
- ・ `-keepio` (未使用の入力を最適化しない)
- ・ `-loc` (指定したロケーション制約を保持)
- ・ `-localfbk` (ローカル フィードバックを使用)
- ・ `-log` (ログ ファイルの指定)
- ・ `-nofbnand` (フォールドバック NAND の使用を無効)
- ・ `-nogclkopt` (グローバル クロックの最適化を無効)
- ・ `-nogstropt` (グローバル セット/リセットの最適化を無効)
- ・ `-nogtsopt` (グローバル出力イネーブルの最適化を無効)
- ・ `-noisp` (ISP ピンの予約を無効)
- ・ `-nomlopt` (複数レベルのロジック最適化を無効)
- ・ `-nouim` (FASTConnect/UIM 最適化を無効)
- ・ `-ofmt` (出力形式の指定)
- ・ `-optimize` (集積度/スピードに基づいて最適化)
- ・ `-p` (製品番号)
- ・ `-pinfbk` (ピンのフィードバックを使用)
- ・ `-power` (電力モードの設定)
- ・ `-pterm` (最適化に使用する積項数)
- ・ `-slew` (スルー レートの設定)
- ・ `-terminate` (終端モードに設定)
- ・ `-unused` (未使用 I/O の終端モードを設定)
- ・ `-wysiwyg` (最適化を実行しない)

メモ: 特に記述のない場合、これらのオプションはすべての CPLD ファミリで使用できます。

-blkfanin (ファンクション ブロックの最大ファンインを指定)

デバイスをフィットする際に使用するファンクション ブロック入力の最大数を指定します。この値が最大値に近い場合、ピン配置を変更せずにデザイン リビジョンをフィットできる可能性が低くなります。

構文

`-blkfanin [limit:4,40]`

最大値は次のとおりです (カッコ内の値はデフォルト)。

- ・ CoolRunner™ XPLA3 = 40 (38)
- ・ CoolRunner-II = 40 (36)

-exhaust (包括的なフィットを有効)

入力および積項の数は、デザインのフィットに影響します。デザインが最適にフィットされるよう、異なる値を試す必要があります。このオプションをオンにすると、適切なフィットが見つかるまで、自動的にさまざまな入力数および積項数の組み合わせを使用してフィットが繰り返し実行されます。デザインのサイズによっては、数時間かかる場合があります。デフォルトではオフです。

サポートされるアーキテクチャ : CoolRunner™ XPLA3 および CoolRunner-II

構文

-exhaust

-ignoredatagate (DATA_GATE 属性を無視)

CoolRunner™-II デバイスをフィットする際に、DATA_GATE を無視します。デフォルトではオフです。

サポートされるアーキテクチャ : CoolRunner-II

構文

-ignoredatagate

-ignoretspec (タイミング使用を無視)

CPLDFit は通常、タイミング制約に適合するようパスを最適化します。このオプションをオンにすると、この最適化が実行されないように指定できます。デフォルトではオフです。

構文

-ignoretspec

-init (パワーアップ値を設定)

すべてのレジスタに対するデフォルトのパワーアップ状態を指定します。レジスタに INIT 属性を設定すると、INIT 属性の設定が優先されます。fpga に設定すると、パワーアップ時に非同期リセットがあるレジスタは Low に、非同期プリセットがあるレジスタは High に、残りのレジスタは Low になります。デフォルトでは low に設定されています。

構文

-init [low|high|fpga]

-inputs (最適化に使用する入力数)

1 つの論理式に対する最大入力数を指定します。値が大きいくほど論理式で使用するリソースの数が多くなるため、1 つのファンクション ブロックで利用できる論理式の数が制限される場合があります。

構文

-inputs [*limit*]

指定可能な最大値は、CPLD アーキテクチャによって異なります。アーキテクチャ別の値は次のとおりです (カッコ内の値はデフォルト)。

- ・ XC9500 : 2 ~ 36 (36)
- ・ XC9500XL : 2 ~ 54 (54)
- ・ CoolRunner™ XPLA3 : 2 ~ 40 (36)
- ・ CoolRunner-II : 2 ~ 40 (36)

-iostd (I/O 規格の指定)

すべての I/O に対するデフォルトの電圧規格を指定します。すべての I/O にデフォルトの電圧規格を指定します。

メモ : このオプションは、CoolRunner™-II デバイスでのみ使用できます。

構文

-iostd *voltage_standard*

*voltage_standard*I/O に割り当てる電圧規格を指定します。有効な値は LVTTTL、LVCMOS18、LVCMOS18_ALL、LVCMOS25、LVCMOS33、SSTL2_I、SSTL3_I、HSTL_I、および LVCMOS15 で、デフォルト値は LVCMOS18 です。

-keepio (未使用の入力を最適化しない)

未使用の I/O ピンが最適化されないよう指定します。デフォルトでは、未接続の入力ピンは最適化で削除されます。

メモ : ほかのデバイスでは複数の I/O 規格がサポートされますが、特別なソフトウェア設定は必要ありません。

構文

-keepio

-loc (指定したロケーション制約を保持)

ロケーション制約の使用方法を指定します。

構文

-loc [*on|off|try*]

on (デフォルト) : ロケーション制約が適用されます。

off : ロケーション制約は無視されます。

try : フィットのエラーが発生しない限り、ロケーション制約が使用されます。

-localfbk (ローカル フィードバックを使用)

XC9500 マクロセルには、ローカル フィードバック パスが含まれています。このオプションは、このフィードバック パスをオンにします。デフォルトではオフです。

サポートされるアーキテクチャ : XC9500

構文

-localfbk

-log (ログ ファイルの指定)

エラー、警告、情報メッセージを含むログ ファイルを生成します。

構文

-log logfile

-nofbnand (フォールドバック NAND の使用を無効)

デザインをフィットする際に、フォールドバック NAND の使用を無効にします。デフォルトではオフです。

サポートされるアーキテクチャ : CoolRunner™ XPLA3

構文

-nofbnand

-nogclkopt (グローバル クロックの最適化を無効)

グローバル クロックの自動推論をオフにします。デフォルトではオフです。

構文

-nogclkopt

-nogsropt (グローバル セット/リセットの最適化を無効)

グローバル セット/リセットの自動推論をオフにします。このオプションをオフにした場合、グローバル バッファを UCF で宣言するか、HDL または回路図に直接インスタンスエートする必要があります。

構文

-nogsropt

-nogtsopt (グローバル出力インプットの最適化を無効)

グローバル トライステートの自動推論をオフにします。このオプションをオフにした場合、グローバル バッファを UCF で宣言するか、HDL または回路図に直接インスタンスエートする必要があります。

構文

-nogtsopt

-noisp (ISP ピンの予約を無効)

JTAG ピンを無効にし、I/O ピンとして使用できるようにします。デフォルトではオフです。

サポートされるアーキテクチャ : CoolRunner™ XPLA3

構文

-noisp

-nomlopt (複数レベルのロジック最適化を無効)

デザインをフィットする際に、複数レベルのロジックの最適化を無効にします。デフォルトではオフです。

構文

-nomlopt

-nouim (FASTConnect/UIM 最適化を無効)

XC9500 インターコネクト マトリックスを使用すると、複数の信号を結合して、ワイヤード AND を形成できます。このオプションは、その機能をオフにします。デフォルトではオフです。

サポートされるアーキテクチャ : XC9500

構文

-nouim

-ofmt (出力形式の指定)

インプリメントされた論理式を記述する際に、フィッターレポートに使用する言語を指定します。

構文

-ofmt [vhdl|verilog]

-optimize (集積度/スピードに基づいて最適化)

デザインの最適化で集積度またはスピードのどちらを優先するかを指定します。集積度を優先して最適化を実行すると、動作周波数は低くなりますが、リソースが共有されるため、より多くのロジックをデバイスにフィットできます。スピードを優先して最適化した場合、リソースが共有される率は低くなりますが、ロジックがフラットになるため、ロジックレベル数が減り、結果的に動作周波数が高くなります。デフォルトでは density (集積度) に設定されています。

構文

-optimize density|speed

-p (製品番号)

デザインをインプリメントするパーツを指定します。

構文

-p *part_number*

part_number は、「XC2C512-10-FT256」のように、デバイス-スピードグレード-パッケージの形式で指定します。デバイスが鉛フリー パッケージである場合は、XC2C512-10-FTG25 のように、パッケージ名に G が付きます。ソフトウェアでは通常のパッケージと鉛フリー パッケージは同一に処理されるので、パッケージを指定する際は G は必要ありません。

「XPLA3」のようにデバイス ファミリ名のみを入力した場合、適切なフィットが見つかるまで、そのファミリのすべての集積度のデバイスを使用してフィットが繰り返し実行されます。

-pinfbk (ピンのフィードバックを使用)

XC9500 アーキテクチャでは、I/O ピンを介してデバイスにフィードバックできます。このオプションは、このフィードバック機能をオンにします。デフォルトはオンです。

サポートされるアーキテクチャ : XC9500

構文

-pinfbk

-power (電力モードの設定)

マクロセルのデフォルトの電力モードを設定します。マクロセルに電力設定を明示的に割り当てると、その設定が優先されます。

メモ : サポートされるアーキテクチャ : XC9500、XC9500XL、XC9500XV

構文

-power [*std*|*low*|*auto*]

std (デフォルト) : 標準の高速モードを使用します。

low : 低消費電力モードを使用します。スピードは低くなります。

auto : タイミング制約に基づいて *std* または *low* が自動的に選択されます。

-pterm (最適化に使用する積項数)

1 つの論理式に対する最大積項数を指定します。値が大きくなるほど論理式に使用する積項リソースの数が多くなるため、1 つのファンクション ブロックに使用できる論理式の数が制限される場合があります。指定可能な最大値は、CPLD アーキテクチャによって異なります。

構文

-pterm [*limit*:*1*,*90*]

アーキテクチャ別の値は次のとおりです (カッコ内の値はデフォルト)。

- ・ XC9500 : 90 (25)
- ・ XC9500XL : 90 (25)
- ・ CoolRunner™ XPLA3 : 48 (36)
- ・ CoolRunner-II : 56 (36)

-slew (スルー レートの設定)

出力ピンのスルー レートを指定します。auto に設定すると、タイミング制約に基づいてスルー レートが選択されます。デフォルトでは fast に設定されています。

構文

```
-slew [fast|slow|auto]
```

-terminate (終端モードに設定)

すべての入力およびトライステート可能な出力を、指定したモードの終端にグローバルに設定します。使用可能な終端モードは、アーキテクチャによって異なります。

構文

```
-terminate [pullup|keeper|float]
```

使用可能な終端モードは、次のとおりです (カッコ内はデフォルト)。

- ・ XC9500XL : float、keeper (keeper)
- ・ CoolRunner™ XPLA3 : float、pullup (pullup)
- ・ CoolRunner-II : float、pullup、keeper、pulldown (float)

-unused (未使用 I/O の終端モードを設定)

未使用ピンの終端方法を指定します。指定可能な値は、アーキテクチャによって異なります。

構文

```
-unused [ground|pulldown|pullup|keeper|float]
```

指定可能なオプションは、次のとおりです (カッコ内はデフォルト)。

- ・ XC9500XL : float、ground (float)
- ・ CoolRunner™ XPLA3 : float、pullup (pullup)
- ・ CoolRunner-II : float、ground、pullup、keeper、pulldown (ground)

-wysiwyg (最適化を実行しない)

デザインにいかなる最適化も実行しないよう指定します。デフォルトではオフです。

構文

```
-wysiwyg
```


TSIM

この章では、TSIM について説明します。次のセクションから構成されています。

- ・ [TSIM の概要](#)
- ・ [TSIM の構文](#)

TSIM の概要

TSIM プログラムは、インプリメントされた CPLD デザイン ファイル (VM6) を読み込み、アノテートされた NGA ファイルを出力するコマンド ライン プログラムです。NetGen のタイミング シミュレーション フローでは、生成された NGA ファイルを使用して、バックアノテートされたタイミング ネットリストを生成します。詳細は、[「NetGen」の章](#)の「CPLD タイミング シミュレーション」を参照してください。

TSIM のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

TSIM の入力ファイル

TSIM では、VM6 ファイルを入力として使用します。これは CPLDFit により出力されるデータベース ファイルで、ターゲット CPLD へのユーザー デザインのマップ情報が含まれています。

TSIM の出力ファイル

TSIM の出力ファイルは NGA ファイルです。これはバックアノテートされた論理的なデザイン ファイルで、NetGen で実行されるタイミング シミュレーション フローで入力ファイルとして使用されます。

TSIM の構文

TSIM プログラムのコマンド ライン 構文は、次のとおりです。

```
tsim design.vm6 output.nga
```

design.vm6 : 入力デザイン ファイル (VM6) です。このファイルは、CPLDFit により生成されます。詳細は、[「CPLDFit」の章](#)を参照してください。

output.nga : 出力ファイルです。このファイルは、バックアノテートされたネットリストを生成するため NetGen のタイミング シミュレーション フローで入力ファイルとして使用されます。出力ファイル名を指定しない場合、入力デザイン ファイルのルート名に拡張子 *.nga* が付けられます。

TAEngine

この章では、TAEngine (Timing Analysis Engine) プログラムについて説明します。次のセクションが含まれています。

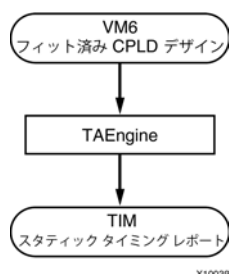
- ・ [TAEngine の概要](#)
- ・ [TAEngine の構文](#)
- ・ [TAEngine のオプション](#)

TAEngine の概要

TAEngine (Timing Analysis Engine) は、CPLDFit で生成されたインプリメント済みの CPLD デザイン ファイル (VM6) を読み込み、タイミング コンポーネントに対してスタティック タイミング解析を実行するコマンドライン プログラムです。スタティック タイミング解析の結果は、TAEngine レポート ファイル (TIM) にサマリまたは詳細のいずれかの形式で出力されます。

デフォルトでは、サマリ形式でレポートが出力され、すべてのタイミング パスとその遅延がリストされます。-detail (詳細レポートの生成) オプションを使用して生成した詳細レポートには、すべてのタイミング パスのリストと各パスのタイミング コンポーネントのサマリが表示されます。いずれのレポートにも、デザインに設定されたタイミング制約のパフォーマンスが表示されます。

TAEngine のフロー



TAEngine のデバイス サポート

このプログラムは、次のデバイス ファミリーで使用できます。

- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

TAEEngine の入力ファイル

TAEEngine の入力ファイルは、次のとおりです。

VM6 ファイル : CPLDFit により生成された、インプリメント済みの CPLD デザイン ファイルです。

TAEEngine の出力ファイル

TAEEngine の出力ファイルは、次のとおりです。

TIM ファイル : 拡張子が .tim の ASCII テキスト形式のタイミング レポート ファイル。タイミング制約に対するタイミング パスとパフォーマンスをリストします。このファイルは、サマリ (デフォルト) または詳細のいずれかの形式で出力できます。

TAEEngine の構文

TAEEngine を実行するには、次のコマンド ライン構文を使用します。

```
taengine -f design_name.vm6 [options]
```

-f design_name.vm6 : VM6 デザイン ファイルを指定します。

options : 「[TAEEngine のオプション](#)」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

TAEEngine のオプション

TAEEngine のコマンド ライン オプションについて説明します。

- ・ [-detail](#) (詳細レポートの生成)
- ・ [-iopath](#) (パスのトレース)
- ・ [-l](#) (出力ファイル名の指定)

-detail (詳細レポートの生成)

詳細なタイミング レポートを生成します。このレポートには、すべてのパスに対するスタティック タイミング解析の結果と、各パスの遅延が詳細に表示されます。

構文

```
-detail
```

-iopath (パスのトレース)

パスが双方向ピンを通過してトレースされるよう指定します。

構文

```
-iopath
```

-l (出力ファイル名の指定)

出力されるレポートファイルの名前を指定します。デフォルトでは、入力デザイン ファイルのルート名に拡張子 `.tim` が付けられます (`design_name.tim`)。

構文

```
-l output_file.tim
```


Hprep6

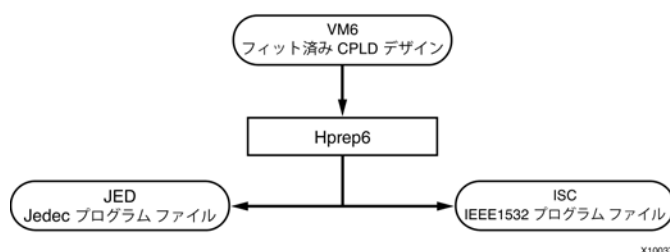
この章では、Hprep6 について説明します。次のセクションが含まれています。

- ・ [Hprep6 の概要](#)
- ・ [Hprep6 のオプション](#)

Hprep6 の概要

Hprep6 は、CPLDFit で生成されたインプリメント済みの CPLD デザイン (VM6) を入力ファイルとして読み込み、CPLD デバイスのコンフィギュレーションに使用するプログラム ファイルを生成するコマンドライン プログラムです。プログラム ファイルは、JEDEC (JED) 形式で生成されます。コマンドライン オプションで指定すると、オプションで ISC 形式でも生成できます。

Hprep6 デザイン フロー



Hprep6 のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

Hprep6 の構文

Hprep6 を実行するには、次の構文を使用します。

```
hprep6 -i design_name.vm6 [options]
```

-i design_name.vm6 : 入力デザイン ファイルを指定します。このオプションは必須です。

options : 「[Hprep6 のオプション](#)」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

Hprep6 の入力ファイル

Hprep6 に使用する入力ファイルは、次のとおりです。

VM6 : CPLDFit により生成された、インプリメント済みの CPLD デザイン ファイルです。詳細は、「[CPLDFit](#)」の章を参照してください。

Hprep6 の出力ファイル

Hprep6 は、次のファイルを出力します。

- ・ **JED ファイル** : CPLD のプログラムに使用する JEDEC ファイルです。
- ・ **ISC ファイル** : CPLD のプログラムに使用する IEEE 1532 形式のファイルです。

Hprep6 のオプション

このセクションでは、Hprep6 のコマンドライン オプションについて説明します。

- ・ [-autosig](#) (シグネチャの自動生成)
- ・ [-intstyle](#) (統合スタイル)
- ・ [-n](#) (リードバックのシグネチャ値を指定)
- ・ [-nopullup](#) (プルアップを無効)
- ・ [-s](#) (ISC ファイルの生成)
- ・ [-tmv](#) (テスト ベクタ ファイルの指定)

-autosig (シグネチャの自動生成)

自動的に生成したパターン特定のシグネチャを JEDEC ファイルに挿入します。このシグネチャは、iMPACT を使用してターゲット デバイスの USERCODE レジスタにプログラムできます。**-n** オプションを使用すると、**-autosig** は無視されます。

構文

```
-autosig
```

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

```
-intstyle ise|xflow|silent
```

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-n (リードバックのシグネチャ値を指定)

XC9500/XL デバイスでのみ使用できます。signature に指定した値は、CPLD 内にプログラムされ、その後 JTAG を介してリードバックされます。この機能は、デバイスにプログラムされたデザインのバージョンを識別するために使用します。

メモ : CoolRunner™ ファミリでもシグネチャ値を使用できますが、iMPACT やサードパーティのプログラム ツールを使用して値を入力する必要があります。

構文

-n [*signature*]

-nopullup (プルアップを無効)

空のファンクション ブロックのプルアップを無効にするよう指定します。デフォルトでは、リーク電流が低減して I/O がフロートしないよう、プルアップが有効になっています。

メモ : このオプションは、XC9500/XL デバイスでのみ使用できます。

構文

-nopullup

-s (ISC ファイルの生成)

IEEE 1532 形式 (ISC) のプログラム ファイルを追加で出力します。このファイルには、*design_name.isc* という名前が付けられます。

メモ : CoolRunner™ XPLA3 ファミリの場合、ISC IEEE 1532 形式のファイルは出力されません。

構文

-s *IEEE1532*

-tmv (テスト ベクタ ファイルの指定)

iMPACT ツールのファンクション テストに使用するテスト ベクタ (TMV) ファイルを指定します。TMV ファイルは ABEL フォーマットのファイルであり、JEDEC プログラム ファイルの最後にテスト ベクタを埋め込みます。

メモ : このオプションは、XC9500/XL デバイスでのみ使用できます。

構文

-tmv *filename*

XFLOW

この章では、XFLOW について説明します。次のセクションが含まれています。

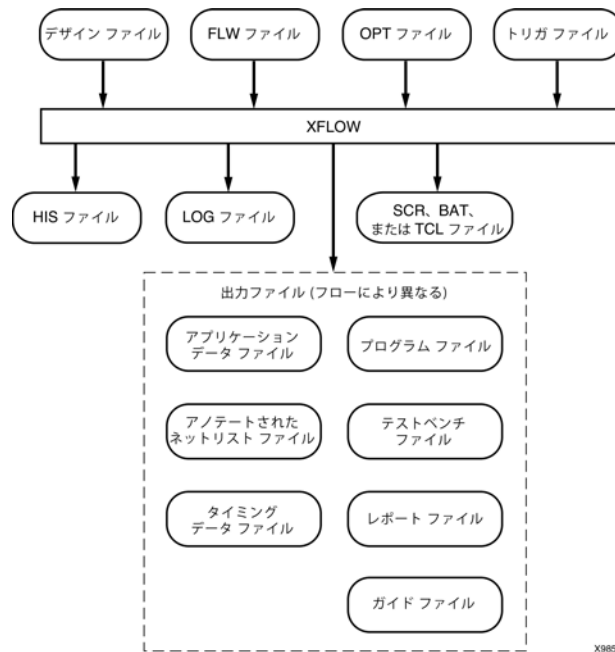
- ・ [XFLOW の概要](#)
- ・ [XFLOW のフロー タイプ](#)
- ・ [XFLOW のオプション ファイル](#)
- ・ [XFLOW のオプション](#)
- ・ [XFLOW の実行](#)

XFLOW の概要

XFLOW は、デザイン ファイルに加えてフロー ファイルとオプション ファイルを読み込み、ザイリンクスの合成、インプリメンテーション、シミュレーションのフローを自動化するコマンドライン プログラムです。ザイリンクスでは、ザイリンクス プログラムを特定のデザイン フローで実行できるようデフォルトのフロー ファイル セットを提供しています。たとえば、FPGA のインプリメンテーション フロー用には、NGDBuild、MAP、PAR、および TRACE を実行するフロー ファイルがあります。デフォルトのフロー ファイルは、そのまま、またはカスタマイズして使用できます。詳細は、「[XFLOW のフロー タイプ](#)」および「[フロー ファイル](#)」を参照してください。オプション ファイルは、フロー ファイルにリストされている各プログラムを実行する際のコマンドライン オプションを指定します。オプション ファイルには、ザイリンクスが提供するデフォルトのファイルを使用するか、ユーザーが独自でファイルを作成できます。詳細は、「[XFLOW のオプション](#)」を参照してください。

次の図に、XFLOW プログラムの入力ファイルと出力ファイルを示します。出力ファイルは、実行するフローによって異なります。

XFLOW デザイン フロー



XFLOW のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

XFLOW の入力ファイル

XFLOW に使用する入力ファイルは、次のとおりです。

デザイン ファイル (合成以外のフロー) : `-synth` を除くすべてのフロー タイプでは、入力デザインに EDIF 2 0 0 または NGC ネットリスト ファイル (XST の出力ファイル) を使用できます。フローの途中から開始する場合は、NGD、NGO、または NCD ファイルも指定できます。XFLOW では、次の拡張子が付いたファイルが識別されます。

拡張子	拡張子
EDIF	.sedif、.edn、.edf、.edif
NCD ファイル	.ncd
NGC ファイル	.ngc
NGD ファイル	.ngd
NGO	.ngo

デザイン ファイル (合成フロー) : **-synth** フロー タイプでは、Verilog または VHDL ファイルを入力デザインとして使用できます。VHDL または Verilog ファイルが複数ある場合は、これらのファイルが参照される PRJ または V ファイルを XFLOW の入力ファイルとして使用できます。PRJ または V ファイルの作成方法は、『[XST ユーザー ガイド](#)』または『[XST ユーザー ガイド \(Virtex-6 および Spartan-6 デバイス用\)](#)』を参照してください。また、Project Navigator を使用して生成した既存の PRJ ファイルも使用できます。XFLOW では、次の拡張子が付いたファイルが識別されます。

拡張子	拡張子
PRJ	.prj
Verilog	.v
VHDL	.vhd

メモ : Synplify 合成ツールを使用して複数のファイルを合成するには、**-g** オプションを使用する必要があります。詳細は、『[-synth](#)』を参照してください。

- ・ **FLW ファイル :** XFLOW でインプリメンテーション フローまたはシミュレーション フローを実行するために必要な情報を含む ASCII 形式のフロー ファイル。フロー タイプを指定すると、特定のフロー ファイルが呼び出されます (『[XFLOW のフロー タイプ](#)』を参照)。このファイルには、フローで起動する各プログラムに対するプログラム ブロックも含まれます。出力ファイルをコピーするディレクトリも、フロー ファイルで指定します。デフォルトのフロー ファイルは、そのまままたは変更して使用できます。詳細は、『[フロー ファイル](#)』を参照してください。
- ・ **OPT ファイル :** フロー ファイルに含まれる各プログラムのオプションを含む ASCII 形式のオプション ファイル。ザイリンクスが提供するデフォルトのファイルを使用するか、ユーザーが独自に作成できます。詳細は、『[XFLOW のオプション ファイル](#)』を参照してください。
- ・ **トリガ ファイル :** UCF、NCF、PCF、MFP ファイルなど、コマンド ライン プログラムで読み込む追加のファイルです。これらのファイルは、コマンド ラインで指定せずに、フロー ファイルの Triggers 行に記述する必要があります。詳細は、『[XFLOW のフロー タイプ](#)』を参照してください。

XFLOW の出力ファイル

XFLOW を実行すると、次のファイルが出力され、作業ディレクトリに保存されます。

- ・ **HIS ファイル** : フローを実行するために入力した XFLOW コマンド、使用されたフローファイルとオプション ファイル、実行されたプログラムのコマンドライン コマンド、フローの各プログラム用の入力ファイルリストを含む ASCII 形式のファイル (xflow.his)。
- ・ **LOG ファイル** : XFLOW の実行中に生成されたすべてのメッセージを含む ASCII 形式のファイル (xflow.log)。
- ・ **SCR、BAT、TCL ファイル** : フローで実行されたすべてのプログラムのコマンドライン コマンドを含むスクリプト ファイル。実行されたコマンドを確認したり、スクリプト ファイルを後で実行できるよう生成されます。拡張子は、使用しているプラットフォームによって異なります。**\$scripts_to_generate** 変数を使用してスクリプト ファイルを指定していても、Linux の場合は SCR ファイルが、PC の場合は BAT ファイルがデフォルトで出力されます。

上記のファイルに加え、次の表に示すファイルが出力されます。生成されるファイルは、フローファイルに含まれるプログラムやオプション ファイルに含まれるコマンドによって異なります。

メモ : デフォルトでは、レポートファイルは作業ディレクトリに保存されます。異なるディレクトリを指定するには、XFLOW の **-rd** (レポートファイルのコピー) オプションを使用するか、フローファイルでレポート ディレクトリ オプション (「フロー ファイル」を参照) を使用します。レポートファイルの形式は、すべて ASCII です。

次のファイルは、FPGA および CPLD の両方のデザインで生成されます。

XFLOW の出力ファイル (FPGA および CPLD)

ファイル名	説明	ファイルの生成方法
<i>design_name.bld</i>	入力ネットリストを NGD ファイルに変換する NGDBuild の実行に関する情報を含む	フロー ファイルに「 ngdbuild 」を含める (-implement または -fit フロー タイプを使用)
<i>time_sim.sdf</i> <i>func_sim.sdf</i>	デザインのタイミング データを含む標準遅延フォーマット ファイル	フロー ファイルに「 netgen 」を含める (-tsim または -fsim フロー タイプを使用) タイミング情報を含む NGA ファイルを入力として使用
<i>time_sim.tv</i> <i>func_sim.tv</i>	Verilog テスト ベンチ ファイル (オプション)	フロー ファイルに「 netgen 」を含める (-tsim または -fsim フロー タイプを使用)
<i>time_sim.tvhd</i> <i>func_sim.tvhd</i>	VHDL テスト ベンチ ファイル (オプション)	フロー ファイルに「 netgen 」を含める (-tsim または -fsim フロー タイプを使用)
<i>time_sim.v</i> <i>func_sim.v</i>	ザイリンクスのシミュレーション プリミティブで表現された Verilog シミュレーション ネットリスト。Verilog 入力ネットリストとは異なり、シミュレーションにのみ使用。インプリメンテーションには使用不可。	フロー ファイルに「 netgen 」を含める (-tsim または -fsim フロー タイプを使用)

ファイル名	説明	ファイルの生成方法
<i>time_sim.vhd</i> <i>func_sim.vhd</i>	ザイリンクスのシミュレーションプリミティブで表現された VHDL シミュレーション ネットリスト。VHDL 入力ネットリストとは異なり、シミュレーションにのみ使用。インプリメンテーションには使用不可。	フロー ファイルに「 netgen 」を含める (-tsim または -fsim フロー タイプを使用)

次のファイルは、FPGA デザインでのみ生成されます。

XFLOW の出力ファイル (FPGA)

ファイル名	説明	ファイルの生成方法
<i>design_name.bgn</i>	ザイリンクスのデバイス コンフィギュレーション用のビットストリームを生成する BitGen の実行に関する情報を含む	フロー ファイルに「 bitgen 」を含める (-config フロー タイプを使用)
<i>design_name.bit</i>	PROMGen または iMPACT を使用して FPGA デバイスにダウンロードするコンフィギュレーション データを含むビットストリーム ファイル	フロー ファイルに「 bitgen 」を含める (-config フロー タイプを使用)
<i>design_name.dly</i>	デザイン内の各ネットの遅延情報を含む	フロー ファイルに「 par 」を含める (-implement フロー タイプを使用)
<i>design_name.ll</i>	BIT ファイルでのラッチ、フリップフロップ、IOB 入出力の位置を含む ASCII 形式のファイル (オプション)	フロー ファイルに「 bitgen 」を含める (-config フロー タイプを使用) オプション ファイルに BitGen の -l オプションを含める
<i>design_name.mrp</i>	論理デザインをザイリンクスの FPGA デバイスにマップする MAP の実行に関する情報を含むファイル	フロー ファイルに「 map 」を含める (-implement フロー タイプを使用)
<i>design_name.ncd</i> (PAR) <i>design_name_map.ncd</i> (MAP)	ターゲット ザイリンクス デバイスのコンポーネントレベルで表現されたデザインの物理記述を含む NCD (ネイティブ回路記述) ファイル。ガイド ファイルとして使用可能。マップ済みの NCD ファイルまたは配置配線済みの NCD ファイルを使用可能。	フロー ファイルに「 map 」または「 par 」を含める (-implement フロー タイプを使用)

ファイル名	説明	ファイルの生成方法
<code>design_name.par</code>	配置配線のすべての実行に関するサマリ情報を含む	フロー ファイルに「 par 」を含める (-implement フロー タイプを使用)
<code>design_name.pad</code>	デザイン内で使用する I/O コンポーネントと関連のプライマリピンをリストしたレポート ファイル	フロー ファイルに「 par 」を含める (-implement フロー タイプを使用)
<code>design_name.rbt</code>	ビットストリーム ファイルのデータを 1 と 0 で表した ASCII 形式のロービット ファイル (オプション)	フロー ファイルに「 bitgen 」を含める (-config フロー タイプを使用) オプション ファイルに BitGen の -b オプションを含める
<code>design_name.twr</code>	NCD ファイルから計算されたタイミング データを含むファイル	フロー ファイルに「 trce 」を含める (-implement フロー タイプを使用)
<code>design_name.xpi</code>	デザインが配線されているか、タイミング仕様を満たしているかどうかを表すファイル	フロー ファイルに「 par 」を含める (-implement フロー タイプを使用)

次の表に示すファイルは、CPLD デザインでのみ生成されます。

XFLOW の出力ファイル (CPLD)

ファイル名	説明	ファイルの生成方法
<code>design_name.gyd</code>	ASCII 形式の CPLD ガイド ファイル	フロー ファイルに「 cpldfit 」を含める (-fit フロー タイプを使用)
<code>design_name.jed</code>	iMPACT を使用して CPLD デバイスにダウンロード可能なコンフィギュレーション データを含む ASCII 形式のファイル	フロー ファイルに「 hprep6 」を含める (-fit フロー タイプを使用)
<code>design_name.rpt</code>	CPLD デバイスに論理デザインをフィットさせるプログラム CPLDFit の実行に関する情報を含むレポート ファイル	フロー ファイルに「 cpldfit 」を含める (-fit フロー タイプを使用)
<code>design_name.tim</code>	タイミング データを含むレポート ファイル	フロー ファイルに「 taengine 」を含める (-fit フロー タイプを使用)

XFLOW の構文

XFLOW のコマンド ライン構文を次に示します。

```
xflow [-p partname] [flow type] [option file[.opt]] [xflow options] design_name
```

flow type: 「[XFLOW のフロー タイプ](#)」にリストされているフロー タイプを指定します。フロー タイプを指定することにより、特定のフロー ファイルが読み込まれます。1 つのコマンドラインに複数のフロー タイプを組み合わせて使用できますが、フロー タイプごとにオプション ファイルが必要になります。

option file: 指定したフロー タイプで使用可能なオプション ファイルをしています。詳細は、「[XFLOW のオプション ファイル](#)」を参照してください。オプション ファイルについては、該当するフロー タイプのセクションを参照してください。

xflow options: 「[XFLOW のオプション](#)」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

design_name: 処理する最上位デザインのファイル名を指定します。入力デザイン ファイルのフォーマットについては、「[XFLOW の入力ファイル](#)」を参照してください。

メモ: フロー タイプおよびオプション ファイルを指定せずにデザイン名のみを指定した場合、FPGA では **-implement** フロー タイプとオプション ファイル `fast_runtime.opt` が使用され、CPLD では **-fit** フロー タイプとオプション ファイル `balanced.opt` が使用されます。

オプション ファイルのパスを完全に指定する必要はありません。デフォルトでは、作業ディレクトリにあるオプション ファイルが使用されます。オプション ファイルが作業ディレクトリにない場合は、次の場所から検索され、作業ディレクトリにコピーされます。これらの場所でオプション ファイルが見つからない場合は、エラー メッセージが表示されます。

- ・ XIL_XFLOW_PATH で指定されているディレクトリ
- ・ 環境変数 XILINX で指定されているインストール ディレクトリ

メモ: デフォルトでは、XFLOW を起動したディレクトリが作業ディレクトリになります。異なるディレクトリを指定する場合は、**-wd** ([作業ディレクトリの指定](#)) オプションを使用してください。

XFLOW のフロー タイプ

フローとは、デザインを合成、インプリメント、シミュレーション、およびコンフィギュレーションするために実行する一連のプログラムのことをいいます。たとえば、FPGA デザインをインプリメントするには、NGDBuild、MAP、PAR プログラムを実行します。

フロー タイプは、フロー ファイルで指定されたフローを実行するよう XFLOW に指示します。フロー ファイルの詳細は、「[フロー ファイル](#)」を参照してください。必要なフローを達成するために、複数のフロー タイプをコマンドラインに入力できます。次のセクションでは、使用可能なフロー タイプについて説明します。

メモ: すべてのフロー タイプにオプション ファイルを指定する必要があります。オプション ファイルが指定されていない場合、エラーが表示されます。

-config (FPGA 用 BIT ファイルの生成)

配線済みのデザインから FPGA デバイス コンフィギュレーション用のビットストリームを生成します。フロー ファイル `fpga.flw` を呼び出し、BitGen を実行します。

構文

```
-config option_file
```

ザイリンクスでは、このフロー タイプで使用するオプション ファイル `bitgen.opt` を提供しています。

ネットリスト ファイルを入力として使用するには、**-config** フロー タイプと共に **-implement** フロー タイプを使用してください。

例

次に、FPGA のインプリメンテーションとコンフィギュレーションを実行するコマンド例を示します。

```
xflow -p xc5v1x30ff324-2 -implement balanced.opt -config
bitgen.opt testclk.edf
```

-implement フロー タイプを使用せずにこのフロー タイプを使用するには、配置配線済みの NCD ファイルを入力として使用する必要があります。

-ecn (等価チェック用ファイルの生成)

FPGA デザインのフォーマル検証に使用するファイルを生成します。フロー ファイル `fpga.flw` を呼び出し、NGDBuild と NetGen を実行して `netgen.ecn` ファイルを生成します。このファイルには、等価チェック用にデザインの Verilog ネットリスト記述が含まれます。

構文

-ecn *option_file*

ザイリンクスでは、このフロー タイプで利用できる次のオプション ファイルを提供しています。

-ecn フロー タイプのオプション ファイル

オプション ファイル	説明
<code>conformal_verilog.opt</code>	Conformal 等価チェック用のオプション ファイル
<code>formality_verilog.opt</code>	Formality 等価チェック用のオプション ファイル

-fit (CPLD のフィット)

CPLD デザインで、ロジックをマクロセルの物理的な位置に組み込みます。フロー ファイル `cp1d.flw` を呼び出し、NGDBuild と CPLDFit を実行して JED ファイルを作成します。

構文

-fit *option_file*

ザイリンクスでは、このフロー タイプで利用できる次のオプション ファイルを提供しています。これらのファイルを使用すると、さまざまなパラメータに基づいてデザインを最適化できます。

-fit フロー タイプのオプション ファイル

オプション ファイル	説明
balanced.opt	スピードと集積度のバランスが取られるよう最適化されています。
speed.opt	スピードを優先するよう最適化されています。
density.opt	集積度を優先するよう最適化されています。

例

```
xflow -p xc2c64-4-cp56 -fit balanced.opt -tsim generic_vhdl.opt
main_pcb.edn
```

この例は、デザインをフィットし、CPLD 用の VHDL タイミング シミュレーション ネットリストを生成します。

-fsim (論理シミュレーション用ファイルの生成)

FPGA または CPLD デザインの論理シミュレーションに使用するファイルを生成します。フロー ファイル `fsim.flw` を呼び出して、NGDBuild および NetGen を実行し、`func_sim.edn`、`func_sim.v`、または `func_sim.vhdl` ファイルを生成します。生成されたファイルには、ザイリンクスのシミュレーション プリミティブで表現されたデザインのネットリスト記述が含まれます。論理シミュレーション ファイルを使用すると、シミュレータによりバックエンド シミュレーションを実行できます。

メモ: 単独で、または **-synth** フロー タイプと共に使用できます。**-implement**、**-tsim**、**-fit**、**-config** のフロー タイプと組み合わせることはできません。

構文

```
-fsim option_file
```

ザイリンクスでは、このフロー タイプで使用するオプション ファイルをベンダー別に提供しています。

-fsim フロー タイプのオプション ファイル

オプション ファイル	説明
generic_vhdl.opt	汎用 VHDL
modelsim_vhdl.opt	ModelSimVHDL
generic_verilog.opt	汎用 Verilog
modelsim_verilog.opt	ModelSimVerilog
nc_verilog.opt	NC-Verilog
vcs_verilog.opt	VCS Verilog
nc_vhdl.opt	NC-VHDL

例

次に、FPGA デザインの Verilog 論理シミュレーション ネットリストを生成するコマンド例を示します。

```
xflow -p xc5v1x30ff324-2 -fsim generic_verilog.opt testclk.v
```

-implement (FPGA のインプリメンテーション)

デザインをインプリメントします。フロー ファイル `fpga.flw` を呼び出し、NGDBuild、MAP、PAR、TRACE を順に実行します。出力されるファイルは、配置配線済みの NCD ファイルです。

構文

```
-implement option_file
```

ザイリンクスでは、このフロー タイプで利用できる次のオプション ファイルを提供しています。これらのファイルを使用すると、さまざまなパラメータに基づいてデザインを最適化できます。

-implement フロー タイプのオプション ファイル

オプション ファイル	説明
<code>fast_runtime.opt</code>	デザインのパフォーマンスよりもランタイムの短縮を優先するよう最適化されています。 中速または低速デザインに適しています。
<code>balanced.opt</code>	ランタイムとエフォートレベルのバランスが取られるよう最適化されています。
<code>high_effort.opt</code>	高いエフォートを使用するよう最適化されています。ランタイムが長くなることがあります。 高速デザインに適しています。

例

次に、**-implement** フロー タイプを使用したコマンド例を示します。

```
xflow -p xc5v1x30ff324-2 -implement balanced.opt testclk.edf
```

-sta (スタティック タイミング解析用ファイルの生成)

FPGA デザインのスタティック タイミング解析に使用されるファイルを生成します。フロー ファイル `fpga.flw` を呼び出して NGDBuild と NetGen を実行し、スタティック タイミング解析ツールで使用される Verilog ネットリストを作成します。

このフロー タイプは、Spartan®-3、Spartan-3A、Spartan-3E デバイスでのみ使用できます。

構文

```
-sta option_file
```

ザイリンクスでは、このフロー タイプで使用するオプション ファイル `primetime_verilog.opt` を提供しています。

-synth

FPGA の場合はインプリメンテーション用に、CPLD の場合はフィット用に、論理シミュレーションの場合はコンパイル用にデザインを合成します。入力デザイン ファイルには、Verilog または VHDL ファイルを使用できます。

-synth フロー タイプは、単独で使用するか、**-implement**、**-fit**、**-fsim** のフロー タイプと共に使用できます。単独で使用すると、`fpga.flw` または `cp1d.flw` ファイルが呼び出され、XST を実行してデザインが合成されます。**-implement**、**-fit**、**-fsim** と組み合わせて使用すると、適切なフロー ファイルが呼び出され、XST を実行してデザインが合成され、次のセクションで説明するようにデザインが処理されます。

- ・ **-implement** (FPGA のインプリメンテーション)
- ・ **-fit** (CPLD のフィット)
- ・ **-fsim** (論理シミュレーション用ファイルの生成)

構文

-synth *option_file*

メモ : **-synth** フロー タイプを使用する際は、**-p** オプションを指定する必要があります。

-synth フロー タイプは、XST または Synplify 合成ツールを使用した合成に使用できます。使用される合成ツールは、オプション ファイルによって異なります。

ザイリンクスでは、このフロー タイプで利用できる次のオプション ファイルを提供しています。これらのファイルを使用すると、さまざまなパラメータに基づいてデザインを最適化できます。

オプション ファイル	説明
<code>xst_vhdl.opt</code> <code>synplicity_vhdl.opt</code>	ロジックのレベル数を減少し、デザインのスピードを向上するよう VHDL ソース ファイルを最適化します。
<code>xst_verilog.opt</code> <code>synplicity_verilog.opt</code>	ロジックのレベル数を減少し、デザインのスピードを向上するよう Verilog ソース ファイルを最適化します。
<code>xst_mixed.opt</code>	ロジックのレベル数を減少し、デザインのスピードを向上するよう VHDL と Verilog の混合ソース ファイルを最適化します。

XST を使用した 1 つのファイルの合成

```
xflow -p xc5vlx30ff324-2 -synth xst_verilog.opt mydesign.v
```

この例は、XST を使用して Verilog デザイン `mydesign.v` を合成します。

XST での複数のファイルの合成

VHDL または Verilog ファイルが複数ある場合、これらのファイルを参照する PRJ ファイルを入力として使用できます。

```
xflow -p xc5vlx30ff324-2 -synth xst_vhdl.opt mydesign.prj
```

この例は、`mydesign.prj` で指定したすべての VHDL ファイルを XST で合成します。

Synplify 合成ツールを使用した 1 つのファイルの合成

```
xflow -p xc5vlx30ff324-2 -synth synplicity_vhdl.opt mycdesign.vhd
```

この例は、Synplify 合成ツールを使用して VHDL デザイン Synplifymydesign.vhd

Synplify 合成ツールを使用した複数のファイルの合成

VHDL ファイルが複数ある場合、テキストファイルにすべてのソース ファイルをリストし (1 行につきファイルを 1 つずつ入力)、**-g** オプションを使用して XFLOW にその情報を渡す必要があります。ソース ファイルをリストしたファイルを使用する場合、構文は次のようになります。

```
xflow -p xc5vlx30ff324-2 -g srclist:filelist.txt -synth
synplicity_vhdl.opt mydesign.vhd
```

この例は、Synplify 合成ツールを使用して filelist.txt にリストされている VHDL ファイルを合成します。mydesign.vhd はこのプロジェクトの最上位デザインです。

XST を使用した合成およびインプリメント

```
xflow -p xc5vlx30ff324-2 -synth xst_vhdl.opt -implement
balanced.opt testclk.prj
```

次に、デザインを合成しインプリメントするコマンド例を示します。

-tsim (タイミング シミュレーション用ファイルの生成)

FPGA デザインのフォーマル検証に使用するファイルを生成します。フロー ファイル fpga.flw または cpld.flw (ターゲット デバイスによる) を呼び出し、FPGA の場合は NetGen を、CPLD の場合は TSIM および NetGen を実行し、ザイリンクスのシミュレーション プリミティブで表現されたデザインのネットリスト記述を含む time_sim.v または time_sim.vhdl ファイルを作成します。出力されたタイミング シミュレーション ファイルを使用すると、バックエンド シミュレーションを実行できます。

構文

```
-tsim option_file
```

ザイリンクスでは、このフロー タイプで使用するオプション ファイルをベンダー別に提供しています。

-tsim フロー タイプのオプション ファイル

オプション ファイル	説明
generic_vhdl.opt	汎用 VHDL
modelsim_vhdl.opt	ModelSimVHDL
generic_verilog.opt	汎用 Verilog
modelsim_verilog.opt	ModelSimVerilog
nc_verilog.opt	NC-Verilog
vcs_verilog.opt	VCS Verilog
nc_vhdl.opt	NC-VHDL

例

次に、CPLD のフィットと VHDL タイミング シミュレーションを実行する例を示します。

```
xflow -p xc2c64-4-cp56 -fit balanced.opt -tsim generic_vhdl.opt
main_pcb.vhd
```

フロー ファイル

コマンドラインでフロー タイプを指定すると、適切なフロー ファイルが呼び出され、フロー ファイルにリストされているプログラムが実行されます。フロー ファイルの拡張子は `.flw` です。プログラムは、フロー ファイルに指定されている順序で実行されます。

ザイリンクス フロー ファイル

ザイリンクスでは 3 つのフロー ファイルを提供しています。新しいプログラムを追加したり、デフォルトの設定を変更したり、独自のコマンドを追加して、これらのフロー ファイルを編集できます。ただし、フロー ファイルは新規作成できません。

フロー タイプ別に呼び出されるフロー ファイルは次のとおりです。

フロー タイプ	フロー ファイル	デバイス	フローの段階	実行されるプログラム
-synth	fpga.flw	FPGA	合成	XST Synplify 合成ツール
-implement	fpga.flw	FPGA	インプリメンテーション	NGDBuild、MAP、 PAR、TRACE
-tsim	fpga.flw	FPGA	タイミング シミュレーション	NGDBuild, NetGen
-ecn	fpga.flw	FPGA	等価チェック	NGDBuild, NetGen
-sta	fpga.flw	FPGA	スタティック タイミング解析	NGDBuild, NetGen
-config	fpga.flw	FPGA	コンフィギュレーション	BitGen
-synth	cpld.flw	CPLD	合成	XST Synplify 合成ツール
-fit	cpld.flw	CPLD	フィット	NGDBuild、 CPLDFit、 TAEngine、Hprep6
-tsim	cpld.flw	CPLD	タイミング シミュレーション	TSim、NetGen
-synth	fsim.flw	FPGA CPLD	合成	XST Synplify 合成ツール
-fsim	fsim.flw	FPGA CPLD	論理シミュレーション	NGDBuild, NetGen

フロー ファイルのフォーマット

フロー ファイルは、次の情報が含まれた ASCII 形式のファイルです。

メモ : Input、Triggers、Export、および Report 行のファイル名には、変数を使用できます。たとえば、「**Input: <design>.vhd**」と入力すると、作業ディレクトリにある VHDL ファイルが入力ファイルとして自動的に読み込まれます。

- ・ **ExportDir** : フローに含まれるプログラムの出力ファイルをコピーするディレクトリを指定します。デフォルトでは、作業ディレクトリに設定されています。

メモ : エクスポート ディレクトリは、**-ed** コマンドライン オプションでも指定できます。フロー ファイルで指定した ExportDir よりも、コマンドライン オプションで指定した ExportDir が優先されます。

- ・ **ReportDir** : フローに含まれるプログラムで生成されたレポート ファイルのコピー先のディレクトリを指定します。デフォルトでは、作業ディレクトリに設定されています。

メモ : レポート ディレクトリは、**-rd** コマンドライン オプションでも指定できます。フロー ファイルで指定した ReportDir よりも、コマンドライン オプションで指定した ReportDir が優先されます。

- ・ **ユーザー指定のグローバル変数** : 次の例に示すように、グローバル変数の値を指定できます。

```
Variables
$simulation_output = time_sim;
End variables
```

フロー ファイルには、フローで起動する各プログラムのプログラム ブロックが含まれます。プログラム ブロックには、次の情報が含まれます。

- **Program** *program_name*

プログラム ブロック名を指定します。 *program_name* には、**ngdbuild** のようにコマンドラインに入力するプログラム名も使用できます。Program をブロックの最初の行として指定します。

- **Flag:** **ENABLED** | **DISABLED**

- **ENABLED** : オプション ファイル内にオプションがある場合、XFLOW でプログラムが実行されます。
- **DISABLED** : ファイルにオプションがあっても、プログラムは実行されません。

- **Input:** *filename*

プログラムの入力ファイル名を指定します。たとえば、NGDBuild プログラム ブロックの場合、*design.edn* を指定します。

- **Triggers:**

プログラムで読み込む必要のある追加のファイルを指定します。たとえば、NGDBuild プログラム ブロックの場合、*design.ucf* を指定します。

- **Exports:**

エクスポートするファイル名を指定します。たとえば、NGDBuild プログラム ブロックの場合、*design.ngd* を指定します。

- **Reports:**

生成されるレポート ファイル名を指定します。たとえば、NGDBuild プログラム ブロックの場合、*design.bld* を指定します。

- **Executable:** *executable_name*

オプションで、1 つのプログラムに複数のプログラム ブロックを作成できます。この場合は、Program 行にプログラム名以外の名前を入力する必要があります。たとえば、「**trce**」ではなく「**post_map_trace**」と入力します。Executable 行には、「**trce**」などコマンドラインに入力するプログラム名を入力します。

たとえば、TRACE を MAP 終了後と PAR 終了後に実行する場合、MAP 後の TRACE および PAR 後の TRACE のプログラム ブロックは、次のようになります。

```
Program post_map_trce
Flag: ENABLED;
Executable: trce;
Input: <design>_map.ncd;
Exports: <design>.twr, <design>.tsi;
End Program post_map_trce
```

```
Program post_par_trce
Flag: ENABLED;
Executable: trce;
Input: <design>.ncd;
Reports: <design>.twr, <design>.tsi;
End Program post_par_trce
```

メモ : オプション ファイルに対応するプログラム ブロックがある場合、Program 行はフロー ファイルの Program 行と一致している必要があります (post_map_trace など)。

End Program *program_name*

プログラム ブロックの終点を示します。 *program_name* は、プログラム ブロックの開始行と同じ名前です。

ユーザー コマンド ブロック

独自のプログラムを実行するには、フロー ファイルにユーザー コマンド ブロックを追加します。ユーザー コマンド ブロック構文は次のとおりです。

```
UserCommand
  Cmdline: <user_cmdline>;
End UserCommand
```

次にコマンド例を示します。

```
UserCommand
  Cmdline: myscript.csh;
End UserCommand
```

メモ : コマンド ラインでは、アスタリスク (*)、ドル記号 (\$)、かっこ () は使用できません。

XFLOW のオプション ファイル

オプション ファイルには、フローで実行するすべてのプログラムのオプションが含まれています。「[XFLOW のフロー タイプ](#)」で説明したように、各フロー タイプごとに拡張子が .opt のオプション ファイルが提供されています。ユーザーが独自のオプション ファイルを作成することも可能です。

メモ : 独自のオプション ファイルを作成する場合は、既存のファイルをコピーして名前を変更し、そのファイルに変更を加えるのが簡単で確実な方法です。

オプション ファイルのフォーマット

オプション ファイルは、フロー ファイル内のプログラムに対応するプログラム ブロックが含まれた ASCII 形式のファイルです。オプション ファイルのプログラム ブロックには、プログラム実行のオプションを指定します。プログラムのオプションを指定するには、コマンドライン オプションまたはパラメータ ファイルを使用します。

- ・ コマンドライン オプション

プログラムのコマンドライン オプションについては、該当するプログラムの章を参照するか、コマンドラインでプログラム名の後に **-h** オプションを入力してください。オプションによっては、ファイル名や値を指定する必要があります。

- ・ パラメータ ファイル

プログラムのパラメータを指定します。パラメータは、指定のファイルに記述します。たとえば、XST では、コマンドライン オプションの実行にスクリプト ファイルが使用されます。

```
Program xst
  -ifn <design>_xst.scr;
  -ofn <design>_xst.log;
  ParamFile: <design>_xst.scr
    "run";
    "-ifn <synthdesign>";
    "-ifmt Verilog";
    "-ofn <design>.ngc";
  .
  .
  .
  End ParamFile
End Program xst
```

メモ： オプション ファイルに記述するファイル名には、変数を使用できます。たとえば、入力ファイルとして *design_name.vhd* を指定すると、作業ディレクトリにある VHDL ファイルが入力ファイルとして自動的に読み込まれます。

XFLOW のオプション

このセクションでは、XFLOW のコマンドライン オプションについて説明します。これらのオプションは、どのフロー タイプでも使用できます。

- ・ `-config` (FPGA 用 BIT ファイルの生成)
- ・ `-ecn` (等価チェック用ファイルの生成)
- ・ `-ed` (エクスポート ディレクトリにファイルをコピー)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-fit` (CPLD のフィット)
- ・ `-fsim` (論理シミュレーション用ファイルの生成)
- ・ `-g` (グローバル変数の指定)
- ・ `-implement` (FPGA のインプリメンテーション)
- ・ `-log` (ログ ファイルの指定)
- ・ `-norun` (スクリプト ファイルのみを生成)
- ・ `-o` (出力ファイル名の変更)
- ・ `-p` (製品番号)
- ・ `-rd` (レポート ファイルのコピー)
- ・ `-sta` (スタティック タイミング解析用ファイルの生成)
- ・ `-synth`
- ・ `-tsim` (タイミング シミュレーション用ファイルの生成)
- ・ `-wd` (作業ディレクトリの指定)

`-ed` (エクスポート ディレクトリにファイルをコピー)

フロー ファイルの Export 行に記述されたファイルを、指定されたディレクトリにコピーします。このオプションを使用しない場合、ファイルは作業ディレクトリにコピーされます。フロー ファイルの Export 行については、「[フロー ファイル](#)」を参照してください。

構文

`-ed export_directory`

`-ed` オプションを `-wd` オプションと共に使用して、エクスポート ディレクトリの絶対パスを指定しない場合、エクスポート ディレクトリは作業ディレクトリの下に作成されます。

例

次の例では、sub3 ディレクトリの下に export3 ディレクトリが作成されます。

```
xflow -implement balanced.opt -wd sub3 -ed export3 testclk.vhd
```

エクスポート ディレクトリを作業ディレクトリのサブディレクトリに指定しない場合は、次のように、絶対パスを指定します。

```
xflow -implement balanced.opt -wd sub3 -ed /usr/export3  
testclk.vhd
```

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f\(コマンド ファイルの実行\)](#)」を参照してください。

-g (グローバル変数の指定)

フロー ファイルまたはオプション ファイルの変数に値を割り当てます。この値は、グローバルに適用されます。

構文

-g *variable:value*

例

コマンド ラインでグローバル変数を指定するには、次のように入力します。

```
xflow -implement balanced -g $simulation_output:time_sim calc
```

メモ: グローバル変数をコマンド ラインとフロー ファイルの両方で指定した場合、コマンド ラインの値が優先されます。

-log (ログ ファイルの指定)

ログ ファイルを指定します。ログ ファイルは、プログラム実行後に作業ディレクトリに書き込まれます。デフォルトのログ ファイル名は `xflow.log` です。

構文

-log

-norun (スクリプト ファイルのみを生成)

デフォルトでは、フロー ファイルで指定したプログラムが実行されます。このオプションを使用すると、プログラムを実行せずにスクリプト ファイル (SCR、BAT、TCL) を生成できます。該当するフロー ファイルおよびオプション ファイルが作業ディレクトリにコピーされ、これらのファイルに基づいてスクリプト ファイルが作成されます。このオプションは、フローの実行前にスクリプトでプログラムとオプションを確認する場合に便利です。

構文

-norun

例

次にコマンド例を示します。

```
xflow -implement balanced.opt -norun testclk.edf
```

この例では、balanced.opt および fpga.flw ファイルが作業ディレクトリにコピーされ、次のスクリプト ファイルが作成されます。

```
#####
# Script file to run the flow
#
#####
#
# Command line for ngdbuild
#
ngdbuild -p xc5v1x30ff324-2 -nt timestamp /home/
xflow_test/testclk.edf testclk.ngd
#
# Command line for map
#
map -o testclk_map.ncd testclk.ngd testclk.pcf
#
# Command line for par
#
par -w -ol high testclk_map.ncd testclk.ncd
testclk.pcf
#
# Command line for post_par_trce
#
trce -e 3 -o testclk.twr testclk.ncd testclk.pcf
```

-o (出力ファイル名の変更)

出力ファイルのベース名を変更します。このオプションを指定しない場合、出力ファイルのベース名は、入力ファイルのベース名と同じになります。

構文

-o *output_filename*

例

次の例では、出力ファイルのベース名が「testclk」から「newname」に変更されます。

```
xflow -implement balanced.opt -o newname testclk.edf
```

-p (製品番号)

デザインをインプリメントするパーツを指定します。

構文

-p *part_number*

メモ：構文の詳細および例は、「はじめに」の章の「[-p \(製品番号\)](#)」を参照してください。

デフォルト (**-p** オプションなし) では、入力デザイン ファイルから製品番号が検索されます。製品番号が見つかった場合、その番号がデザインのターゲット デバイスとして使用されます。入力デザイン ファイルに製品番号が見つからない場合、製品番号が不明であることを示すエラー メッセージが表示されます。

FPGA デバイスの場合、製品番号と共にパッケージ名を指定する必要があります。パッケージ名を指定しない場合、MAP でエラーが発生し、パッケージを指定する必要があることを示すメッセージが表示されます。PARTGen の **-i** オプションを使用すると、インストールされているデバイスのパッケージ名が表示されます。詳細は、「PARTGen」の章の「[-i \(デバイス、パッケージ、スピードの一覧の表示\)](#)」を参照してください。

CPLD デバイスの場合は、製品番号またはファミリ名を指定します。

例

次に、**-p** オプションで Virtex®-5 デバイスを指定するコマンド例を示します。

```
xflow -p xc5v1x30ff324-2 -implement high_effort.opt testclk.edf
```

-rd (レポート ファイルのコピー)

XFLOW で出力されたレポート ファイルを、作業ディレクトリから指定ディレクトリにコピーします。オリジナルのレポート ファイルは、作業ディレクトリにそのまま保持されます。

構文

-rd *report_directory*

このオプションを使用する前にレポートのディレクトリを作成できますが、ディレクトリが存在していなくても、このオプションでディレクトリ名を指定すれば、自動的に作成されます。レポート ディレクトリの絶対パス名を指定しない場合は、作業ディレクトリ内に指定のレポート ディレクトリが作成されます。

例

次に、レポート ディレクトリ (*reportdir*) を作業ディレクトリ (*workdir*) 内に作成するコマンド例を示します。

```
xflow -implement balanced.opt -wd workdir -rd reportdir  
testclk.edf
```

レポート ディレクトリを作業ディレクトリのサブディレクトリにしない場合は、次のように、絶対パス名を指定します。

```
xflow -implement balanced.opt -wd workdir -rd /usr/reportdir  
testclk.edf
```

-wd (作業ディレクトリの指定)

デフォルト (**-wd** オプションなし) では、XFLOW の起動ディレクトリが作業ディレクトリとなります。**-wd** オプションを使用すると、異なるディレクトリを作業ディレクトリとして指定できます。フロー ファイル、オプション ファイル、入力ファイルは、作業ディレクトリから検索されます。すべてのサブプログラムはこのディレクトリから起動され、出力ファイルが保存されます。

構文

-wd *working_directory*

メモ : **-wd** オプションを使用した場合に、UCF を入力ファイルとして読み込むには、UCF を作業ディレクトリにコピーする必要があります。

ディレクトリのパスを指定しない場合、作業ディレクトリは現在のディレクトリに作成されます。

例

たとえば、次のように指定すると、ディレクトリ sub1 は現在のディレクトリに作成されます。

```
xflow -fsim generic_verilog.opt -wd sub1 testclk.v
```

また、次のように、作業ディレクトリの絶対パスを指定できます。既存のディレクトリを指定することも可能ですが、ディレクトリが存在していなくても、パスを指定すると自動的に作成されます。

```
xflow -fsim generic_verilog.opt -wd /usr/project1 testclk.v
```

XFLOW の実行

XFLOW の一般的な使用法について説明します。

XFLOW のフロー タイプを組み合わせる使用

XFLOW コマンドラインでフロー タイプを組み合わせると、複数のフローを実行できます。

フロー タイプを組み合わせ、デザインをインプリメンテーションし、FPGA デバイス コンフィギュレーションのビットストリームを作成し、FPGA デザイン testclk の EDIF タイミング シミュレーション ネットリストを生成するには、次のように指定します。

```
xflow -p xc5v1x30ff324-2 -implement balanced -tsim  
generic_verilog -config bitgen testclk
```

フロー タイプを組み合わせ、デザインをフィットし、CPLD デザイン main_pcb の VHDL タイミング シミュレーション ネットリストを生成するには、次のように指定します。

```
xflow -p xc5v1x30ff324-2 -fit balanced -tsim generic_vhdl  
main_pcb
```

Smart Flow

Smart Flow は、入力ファイルの変更を自動的に検出し、適切な段階からフローを実行します。デザイン ファイル、フロー ファイル、オプション ファイル、トリガ ファイルに変更があった場合、その変更が検出されます。中断されたフローも検出され、再実行されます。Smart Flow を実行するには、XFLOW コマンドラインに入力デザイン ファイルを拡張子を付けずに入力します。入力ファイルが自動的に検知され、適切な段階からフローが開始されます。

たとえば、次のコマンドを入力すると、calc.edf ファイルの変更が検出され、フロー ファイルとオプション ファイルのプログラムがすべて実行されます。

```
xflow -implement balanced.opt calc
```

SCR、BAT、TCL ファイルの使用

XFLOW を実行すると、実行されたすべてのプログラムのコマンドを含むスクリプト ファイルが作成されます。このファイルは、次の用途に使用できます。

- ・ 実行されたコマンドの確認
- ・ スクリプト ファイルの実行

デフォルトではこのファイルには `xflow_script.bat` (PC) または `xflow_script.scr` (Linux) というファイル名が付けられますが、`$scripts_to_generate` オプションを使用してスクリプト ファイルのタイプを指定できます。コマンドラインに「`xflow_script.bat`」、「`xflow_script.scr`」、または「`xflow_script.tcl`」と入力すると、スクリプト ファイルを実行できます。

XFLOW の代わりにスクリプト ファイルを実行した場合、Smart Flow の機能は使用されません。XFLOW を実行するとファイルの変更内容に基づいて適切な段階からフローが開始されますが、スクリプト ファイルを実行するとファイルに記述されたコマンドがすべて実行されます。また、スクリプト ファイルではエラー メッセージは表示されません。たとえば、NGDBuild の実行中にエラーが発生した場合、XFLOW ではエラーが検出されてフローが停止しますが、スクリプト ファイルではフローが続行され、MAP が実行されてしまいます。

XIL_XFLOW_PATH 環境変数の使用

この環境変数は、チームでデザインを開発する場合に便利です。デフォルトでは、作業ディレクトリからフロー ファイルとオプション ファイルが検索されますが、環境変数を使用すると、フロー ファイルとオプション ファイルを共通の場所に保存してチーム メンバーがそれぞれローカル ディレクトリにコピーして使用することにより、一貫性を保つことができます。

環境変数を使用するには

1. 必要に応じて、フロー ファイルとオプション ファイルに変更を加えます。
2. フロー ファイルとオプション ファイルを共通ディレクトリに保存し、保存先をチーム メンバーに連絡します。
3. チーム メンバーは、作業ディレクトリから次のように入力します。

```
set XIL_XFLOW_PATH=name_of_central_directory
```

メンバーが XFLOW を実行すると、フロー ファイルとオプション ファイルが共通ディレクトリからローカル ディレクトリにコピーされます。

チーム メンバーが共通ディレクトリのファイルを変更した場合、ローカル ディレクトリのファイルを上書きするには、ローカル ディレクトリのフロー ファイルとオプション ファイルをまず削除して、環境変数を設定してから、XFLOW を再実行します。

NGCBuild

この章では、NGCBuild ユーティリティについて説明します。次の各セクションが含まれています。

- ・ [NGCBuild の概要](#)
- ・ [NGCBuild の構文](#)
- ・ [NGCBuild のオプション](#)

NGCBuild の概要

NGCBuild ユーティリティは、次の操作を実行します。

- ・ 複数のソース ネットリスト (EDIF および NGC ファイル) を 1 つの NGC ファイルにコンパイルし、アトミック エンティティ (インクリメンタル リンケージとも呼ぶ) を作成します。
- ・ ユーザー制約ファイル (UCF) を既存のネットリストまたはネットリストのコレクションにアノテートします。

NGCBuild のほとんどの機能は、NGDBuild の機能のサブセットです。NGCBuild には、次の機能があります。

1. 最上位の EDIF または NGC ネットリストを開く。
2. 最上位ネットリストのデザイン階層を再帰的に移動し、同じディレクトリまたは **-sd** コマンドライン オプションで指定されたディレクトリにあるほかのネットリストへの参照をチェックする。
3. UCF ファイルをリンクされたデザイン階層にアノテートする (オプション)。
4. 結果のデザイン階層を、コマンドラインで指定したように新しい NGC ファイルに記述する。

NGCBuild のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

設計フローでの NGCBuild の使用

NGCBuild は、スタンドアロン ユーティリティとして使用するか、またはさまざまなフローで使用できます。

- ・ NGCBuild は次のために使用します。
 - 複数のデザイン ソースを 1 つにまとめ、IP (部分的なデザイン) を 1 つのファイルとして配布する場合
 - 新しい制約を既存の IP に追加する場合
- ・ NGC シミュレーションを実行する場合は、NGCBuild を使用してデザインの異なる部分 (EDIF および NGC ファイル) を 1 つのユニットにまとめると、デザイン全体を UNISIM ライブラリを使用してシミュレーションできます。

ほかのフローでも NGCBuild を使用しますが、上記の 2 つのケースが主な使用方法です。

NGCBuild の入力ファイル (<infile[.ext]>)

入力ファイルには、最上位 EDIF、NGC、または NGO ファイルを指定します。次の拡張子のファイルを指定できます。

- ・ .edn
- ・ .edf
- ・ .ngc

拡張子を指定しない場合は、NGCBuild により該当するファイルが検索されます (必要に応じて EDIF2NGD も使用)。

NGCBuild の出力ファイル (<outfile[.ngc]>)

出力ファイルは、<outfile[.ngc]> という形式で指定します。拡張子 .ngc の指定はオプションです。

出力ファイルは、必ず指定する必要があります。

入力ファイルの拡張子が .ngc である場合に入力ファイルが上書きされるのを避けるため、<infile[.ext]> と <outfile[.ngc]> を異なる名前にしてください。パス名が異なれば、同じファイル名を使用できます。

NGCBuild での NGC ファイルの検証

ライブラリを展開しない状態では重要なチェックはほとんど実行できないので、NGCBuild ではデザイン ルール チェック (DRC) は実行されません。NGCBuild で NGC ファイルが正常に生成されても、そのファイルの NGCBuild でのチェックでエラーが検出されないとは限りません。生成された NGC ファイルを検証するには、NGCBuild 以降の標準フローをそのファイルのみまたはテストベンチに対して実行する必要があります。

NGCBuild のメッセージとレポート

NGCBuild では、NGDBuild で作成される BLD ファイルと類似した BLC ファイルが作成されます。BLC ファイルの内容は、次のとおりです。

- ・ コンパイルされたネットリストまたはデザイン階層に読み込まれたネットリストに関するレポート
- ・ NGDBuild と同様のデザイン サマリ セクション
- ・ 警告およびエラー メッセージ (DRC は実行されないので少数)

NGCBuild の構文

NGCBuild を起動するには、次のコマンドを実行します。

```
ngcbuild [options] infile[.ext]outfile[.ngc]
```

このコマンドは、次の操作を実行します。

1. NGCBuild を開きます。
2. デザインを読み込みます。
3. デザインを NGC ファイルに変換します。

NGCBuild のオプション

NGCBuild オプションは NGDBuild オプションのサブセットであり、同じ機能があります。

- ・ `-aul` (不一致の LOC を許可)
- ・ `-dd` (保存先ディレクトリ)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-i` (UCF ファイルを無視)
- ・ `-insert_keep_hierarchy` (KEEP_HIERARCHY 制約の挿入)
- ・ `-intstyle` (統合スタイル)
- ・ `-filter` (フィルタ ファイルの指定)
- ・ `-nt` (ネットリスト変換タイプ)
- ・ `-p` (製品番号)
- ・ `-quiet` (警告およびエラーのみを表示)
- ・ `-r` (LOC 制約の無視)
- ・ `-sd` (指定したディレクトリの検索)
- ・ `-uc` (ユーザー制約ファイル)
- ・ `-ur` (ユーザー ルール ファイルの読み込み)
- ・ `-verbose` (すべてのメッセージの表示)

`-aul` (不一致の LOC を許可)

デフォルトでは、UCF または NCF ファイルでピン名、ネット名、インスタンス名に指定した制約がデザインに含まれていない場合、エラーが発生し、NGD ファイルは生成されません。このオプションを使用すると、LOC 制約に対してエラーではなく警告が表示され、NGD ファイルが生成されます。

構文

-aul

HDL または回路図で定義していないピン名、ネット名、インスタンス名に設定したロケーション制約が制約ファイルに含まれている場合、**-aul** オプションを使用してプログラムを実行します。これにより、作成中のデザインと完成デザインの両方に、同じ制約ファイルを適用できます。

メモ : このオプションを使用する場合、デザインに含まれるネット名およびインスタンス名にスペルミスがないことを確認してください。スペルミスがあると、配置配線が正しく実行されない場合があります。

-dd (保存先ディレクトリ)

中間ファイル (デザイン NGO ファイルとネットリスト ファイル) を保存するディレクトリを指定します。このオプションを指定しない場合、ファイルは作業ディレクトリに保存されます。

構文

-dd *NGOoutput_directory*

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f\(コマンド ファイルの実行\)](#)」を参照してください。

-i (UCF ファイルを無視)

UCF ファイルを無視します。デフォルトでは、最上位デザインのディレクトリに入力デザイン ファイルとベース名が同じで拡張子が *.ucf* のファイルが存在する場合、この UCF ファイルに記述された制約が自動的に読み込まれます。

構文

-i

メモ : このオプションを使用する場合は、**-uc** オプションを使用しないでください。

-insert_keep_hierarchy (KEEP_HIERARCHY 制約の挿入)

各入力ネットリストに KEEP_HIERARCHY 制約が自動的に追加されます。このオプションは、ネットリストが階層ごとに生成される、ボトムアップの合成フローを使用する場合にのみ使用できます。このオプションを使用する場合は、『[合成/シミュレーション デザイン ガイド](#)』で説明している設計手法を使用してください。

構文

-insert_keep_hierarchy

メモ : コアを含むデザインでこのオプションを使用する場合、コアは階層を保持するよう記述されていない可能性があるため、注意が必要です。

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

```
-intstyle ise|xflow|silent
```

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-filter (フィルタ ファイルの指定)

フィルタ ファイルを指定します。フィルタ ファイルには、プログラムの実行中に生成されるメッセージを保存およびフィルタ処理するための設定が含まれています。

構文

```
-filter [filter_file]
```

デフォルトのフィルタ ファイル名は `filter.filter` です。

-nt (ネットリスト変換タイプ)

ネットリスト ラウンチャによるタイムスタンプの処理方法を決定します。タイムスタンプは、ファイルが作成された日時を示す情報です。

構文

```
-nt timestamp|on|off
```

timestamp (デフォルト) : ネットリスト ラウンチャにより通常のタイムスタンプ チェックが実行され、タイムスタンプに応じて NGO がアップデートされます。

on : タイムスタンプにかかわらず、ネットリストが変換されます (すべての NGO を再構築)。

off : タイムスタンプにかかわらず、既存の NGO は再構築されません。

-p (製品番号)

デザインをインプリメントするパーツを指定します。

構文

```
-p part_number
```

メモ : 構文の詳細および例は、「はじめに」の章の「[-p \(製品番号\)](#)」を参照してください。

-quiet (警告およびエラーのみを表示)

警告メッセージおよびエラーメッセージのみを表示します。

構文

-quiet

-r (LOC 制約の無視)

入力ネットリストまたは UCF に記述されたすべてのロケーション制約 (LOC=) が無視されます。あるアーキテクチャ内のロケーションが別のアーキテクチャ内のロケーションと一致しない場合があるので、異なるデバイスまたはアーキテクチャに移行する際は、このオプションを使用します。

構文

-r

-sd (指定したディレクトリの検索)

ファイル参照 (回路図で FILE=*filename* プロパティを使用して指定されたファイル) を解決する際、およびネットリスト、NGO、NGC、NMC、MEM の各ファイルを検索する際に、検索するディレクトリのリストに指定したパス (*search_path*) を追加します。最上位デザイン ネットリストのディレクトリは、NGCBuild により自動的に検索されるので、検索パスとして指定する必要はありません。

構文

-sd {*search_path*}

-sd オプションと *search_path* の間には、スペースまたはタブを挿入します。たとえば、「**-sddesigns**」ではなく、「**-sd designs**」と指定します。各パスの前に **-sd** を付けて、1 つのコマンドラインに複数の検索パスを指定できます。1 つの **-sd** の後に複数の検索パスを指定することはできません。たとえば、2 つの検索パスを指定するには、次のように指定します。

-sd /home/macros/counter -sd /home/designs/pal2

次のように指定することはできません。

-sd /home/macros/counter /home/designs/pal2

-uc (ユーザー制約ファイル)

ネットリストラウンチャで読み込む UCF を指定します。UCF には、論理デザインのターゲットデバイスへのインプリメント方法に影響するタイミング制約とレイアウト制約が含まれています。

構文

-uc *ucf_file* [.ucf]

この UCF ファイルの拡張子には、**.ucf** を使用する必要があります。拡張子を指定しない場合、拡張子 **.ucf** が追加されます。拡張子が **.ucf** 以外のファイル名を指定すると、エラーメッセージが表示され、NGCBuild は実行されません。

-uc オプションを使用しない場合でも、入力デザイン ファイルと同じベース名で拡張子が **.ucf** の UCF が存在すれば、UCF に記述された制約が自動的に読み込まれます。

UCF ファイルの詳細は、『[制約ガイド](#)』を参照してください。

メモ : 入力ファイルとして使用できる UCF は 1 つのみです。-uc オプションは複数指定できません。

メモ : このオプションを使用する場合は、-i オプションを使用しないでください。

-ur (ユーザー ルール ファイルの読み込み)

ネットリスト ラウンチャがアクセスするユーザー ルール ファイルを指定します。このファイルは、有効なネットリスト入力ファイル、これらのファイルを読み込むネットリストリーダ、およびネットリストリーダのデフォルトのオプションを指定します。また、デザインを処理するサードパーティのコマンドも指定できます。

構文

```
-ur rules_file [.urf]
```

このファイルの拡張子は .urf にする必要があります。拡張子を付けずにユーザー ルール ファイル名を指定すると、ファイル名に .urf の拡張子が追加されます。.urf 以外の拡張子の付いたファイル名を指定すると、エラー メッセージが表示され、NGCBuild は実行されません。

詳細は、付録 B の「[ユーザー ルール ファイル \(URF\)](#)」を参照してください。

-verbose (すべてのメッセージの表示)

NGCBuild、ネットリスト ラウンチャ、ネットリストリーダの実行により出力されるすべてのメッセージを画面に表示します。このオプションは、ツールの実行に関する詳細情報が必要な場合に便利です。

構文

```
-verbose
```


Compxlib

この章では、Compxlib について説明します。次のセクションが含まれています。

- ・ [Compxlib の概要](#)
- ・ [Compxlib の構文](#)
- ・ [Compxlib のオプション](#)
- ・ [Compxlib のコマンド ラインの例](#)
- ・ [ランタイム オプションの指定](#)
- ・ [コンフィギュレーション ファイルの例 \(Windows\)](#)

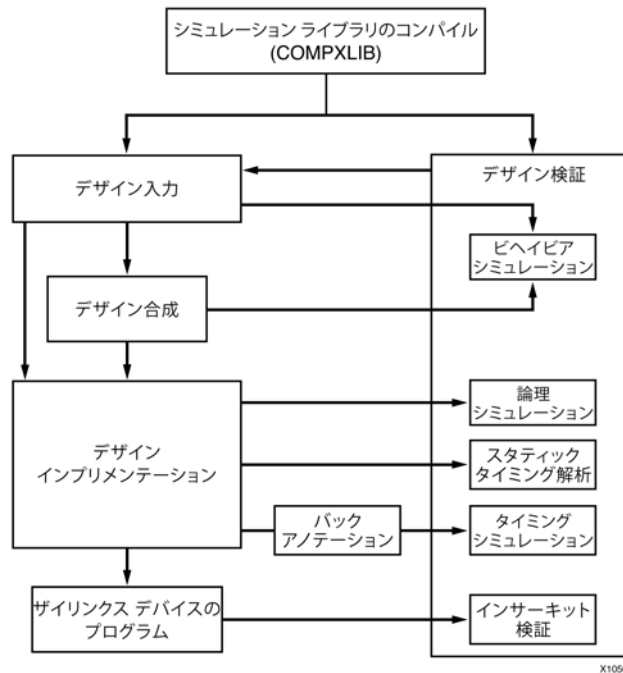
Compxlib の概要

Compxlib は、ザイリンクスの HDL ベースのシミュレーション ライブラリをサードパーティのシミュレータ用にコンパイルするツールです。ライブラリは通常、シミュレータの新しいバージョン、ISE の新しいバージョン、新しいサービス パックをインストールするたびに、コンパイルまたは再コンパイルする必要があります。

デザインの論理シミュレーションを実行する前に、ザイリンクス シミュレーション ライブラリを使用するシミュレータ用にコンパイルする必要があります。ザイリンクスでは、コンパイル ツールとして Compxlib を提供しています。

メモ： ModelSim XE (Xilinx Edition) または ISim では使用しないでください。これらのシミュレータでは、あらかじめコンパイルされたザイリンクス ライブラリが提供されています。

デザイン フロー



メモ: デザイン サイクル中に新しいシミュレータ、ISE® Design Suite の新しいバージョンまたはアップデータをインストールした場合は、Compplib を再実行する必要があります。

Compplib のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3, Spartan-3A, Spartan-3E, Spartan-6
- ・ Virtex®-4, Virtex-5, Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

Compplib の構文

コマンド ラインからシミュレーション ライブラリをコンパイルするには、次のように入力します。

compplib [*options*]

options: 「Compplib のオプション」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。

たとえば、次のコマンドを実行すると、ModelSim SE シミュレータ用に Virtex®-6 デバイス ファミリのザイリンクス Verilog ライブラリがすべてコンパイルされます。

compplib -s mti_se -arch virtex4 -l verilog

コンパイルされたライブラリは、デフォルトのディレクトリ \$XILINX/verilog/mti_se/6.5c/lin に保存されます。

Compplib のオプションおよび構文の詳細は、「Compplib のオプション」を参照してください。

Compplib のヘルプを表示するには、コマンドラインに「**compplib -help** <value>」と入力します。

value には、Compplib の特定のオプションまたはデバイス ファミリを指定します。詳細は、「[Compplib のコマンドラインの例](#)」を参照してください。

メモ : Project Navigator でシミュレーションライブラリをコンパイルする方法については、ISE® ヘルプの「HDL シミュレーションライブラリのコンパイル」を参照してください。Project Navigator の [Process Properties] ダイアログ ボックスで、さまざまなオプションを指定できます。Project Navigator では、そのデザイン フローに関連するオプションのみが表示されます。たとえば、Virtex-6 のプロジェクトでは、Virtex-6 デザインをシミュレーションするのに必要なライブラリのみが表示されます。プロセスの終了後に Project Navigator の [Design] パネルの [Processes] ペインで [View Compilation Log] をダブルクリックすると、`compplib.log` ファイルが開き、コンパイルの結果を表示できます。

Compplib のオプション

このセクションでは、Compplib のコマンドライン オプションについて説明します。

- ・ `-arch` (デバイス ファミリ)
- ・ `-cfg` (コンフィギュレーション ファイルの作成)
- ・ `-dir` (出力ディレクトリ)
- ・ `-e` (既存のディレクトリ)
- ・ `-exclude_deprecated` (廃止予定の EDK ライブラリを除外)
- ・ `-exclude_sublib` (EDK サブライブラリを除外)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-info` (コンパイル済みライブラリの情報の表示)
- ・ `-l` (言語)
- ・ `-lib` (コンパイルするライブラリの名前の指定)
- ・ `-log` (ログ ファイル)
- ・ `-p` (シミュレータのパス)
- ・ `-s` (ターゲット シミュレータ)
- ・ `-source_lib` (ソース ライブラリ)
- ・ `-verbose` (詳細メッセージ)
- ・ `-w` (コンパイル済みライブラリの上書き)

`-arch` (デバイス ファミリ)

指定したデバイス ファミリのライブラリをコンパイルします。

構文

-arch {*device_family* | *all*}

-arch オプションを指定しない場合、エラー メッセージが表示され、ライブラリはコンパイルされません。all を指定すると、すべてのデバイス ファミリ用のライブラリが生成されます。

device_family に指定可能な値は、次のとおりです。

- ・ acr2 (オートモーティブ CoolRunner™-II)
- ・ aspartan3 (オートモーティブ Spartan®-3)
- ・ aspartan3a (オートモーティブ Spartan-3A)
- ・ aspartan3adsp (オートモーティブ Spartan-3A DSP)
- ・ aspartan3e (オートモーティブ Spartan-3E)
- ・ aspartan6 (オートモーティブ Spartan-6)
- ・ qrvirtex4 (QPro™ Virtex®-4 Rad Tolerant)
- ・ qvirtex4 (QPro Virtex-4 Hi-Rel)
- ・ qvirtex5 (QPro Virtex-5 Hi-Rel)
- ・ qspartan6 (QPro Spartan-6 Hi-Rel)
- ・ qvirtex6 (QPro Virtex-6 Hi-Rel)
- ・ spartan3 (Spartan-3)
- ・ spartan3a (Spartan-3A)
- ・ spartan3adsp (Spartan-3A DSP)
- ・ spartan3e (Spartan-3E)
- ・ spartan6 (Spartan-6)
- ・ virtex4 (Virtex-4)
- ・ virtex5 (Virtex-5)
- ・ virtex6 (Virtex-6)
- ・ virtex6l (Virtex-6 低消費電力)
- ・ xa9500xl (オートモーティブ XC9500XL)
- ・ xbr (CoolRunner-II)
- ・ xc9500 (XC9500)
- ・ xc9500xl (XC9500XL)
- ・ xpla3 (CoolRunner XPLA3)

-cfg (コンフィギュレーション ファイルの作成)

コンフィギュレーション ファイルをデフォルト設定で作成します。デフォルトでは、`compxlib.cfg` ファイルが現在のディレクトリに存在しない場合に、このファイルが生成されます。ファイル名を指定することもできます。

コンフィギュレーション ファイルは、ライブラリのコンパイル中にランタイム オプションを Compxlib に渡すために使用します。コンフィギュレーション ファイルの詳細は、「[ランタイム オプションの指定](#)」を参照してください。

構文

-cfg [*cfg_file*]

-dir (出力ディレクトリ)

ライブラリをコンパイルするディレクトリのパスを指定します。デフォルトでは、次の表に示すディレクトリでライブラリがコンパイルされます。

Compxlib のデフォルトの出力ディレクトリ

OS	デフォルトの出力ディレクトリ
Linux	\$XILINX/language/target_simulator/version/lin
Windows	%XILINX%\language\target_simulator\version\{nt nt64}

構文

-dir *dir_path*

-e (既存のディレクトリ)

Compxlib で以前にコンパイルされたライブラリのディレクトリを指定します。

構文

-e *existing_directory*

existing_directory: Compxlib で以前にコンパイルされたライブラリのディレクトリを指定します。

-exclude_deprecated (廃止予定の EDK ライブラリを除外)

廃止予定の EDK ライブラリをコンパイルしないよう指定します。廃止予定のライブラリに関する情報は、『エンベデッド システム ツール リファレンス マニュアル』を参照してください。

構文

-exclude_deprecated

-exclude_sublib (EDK サブライブラリを除外)

PAO ファイルで定義された EDK サブライブラリをコンパイルしないよう指定します。どのライブラリがサブライブラリかについては、『エンベデッド システム ツール リファレンス マニュアル』を参照してください。

構文

-exclude_sublib

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f\(コマンド ファイルの実行\)](#)」を参照してください。

-info (コンパイル済みライブラリの情報の表示)

コンパイル済みのライブラリの情報を表示します。指定したディレクトリに含まれるライブラリの情報が表示されます。

構文

```
-info dir_path
```

-l (言語)

ライブラリをコンパイルする言語を指定します。

構文

```
-l {all|verilog|vhdl}
```

デフォルトでは、**-s** オプションから言語が検出されます。シミュレータで Verilog と VHDL の両方がサポートされる場合は、次のように処理されます。

- ・ **-l** オプションが **all** に設定されます。
- ・ Verilog と VHDL ライブラリの両方がコンパイルされます。

シミュレータで Verilog と VHDL の両方がサポートされない場合は、次のように処理されます。

- ・ シミュレータでサポートされる言語が検出されます。
- ・ **-l** オプションの値が検出された言語に設定されます。

-l オプションを指定すると、指定した言語でライブラリがコンパイルされます。

メモ : XILINX_EDK 環境変数が指定されており、EDK コンパイルが選択されている場合は、このオプションは無視され、VHDL および Verilog のライブラリがコンパイルされます。

-lib (コンパイルするライブラリの名前の指定)

コンパイルするライブラリの名前を指定します。このオプションを使用しない場合、または **all** を指定した場合は、すべてのライブラリがコンパイルされます。

構文

```
-lib [library|all]
```

library に有効な値は、次のとおりです。

- ・ **unisim** (または **u**)
- ・ **simplim** (または **s**)
- ・ **uni9000** (または **n**)
- ・ **xilinxcorelib** (または **c**)
- ・ **coolrunner** (または **r**)
- ・ **edk** (または **e**)

複数のライブラリを指定する場合は、次の例のように **-lib** オプションをスペースで区切って指定します。

```
.. -lib unisim -lib simplim ..
```

メモ: EDK ライブラリを指定した場合 (**-lib edk**)、EDK ライブラリは UNISIM および SIMPRIM に依存しているため、すべての ISE® ライブラリがコンパイルされます。

-log (ログ ファイル)

ログ ファイルを指定します。

構文

-log *log_file*

log_file : ログ ファイルの名前を指定します。

-p (シミュレータのパス)

シミュレータの実行ファイルが配置されているディレクトリのパスを指定します。デフォルトでは、`$PATH` または `%PATH%` 環境変数から自動的にパスが検出されます。このオプションは、ターゲットシミュレータが `$PATH` または `%PATH%` 環境変数で指定されていない場合、またはこれらの環境変数で指定されているのとは異なるディレクトリパスを指定する場合に使用します。

構文

-p *dir_path*

-s (ターゲット シミュレータ)

ライブラリをコンパイルする際のターゲットシミュレータを指定します。

-s オプションを指定しない場合、ライブラリはコンパイルされません。

構文

-s *simulator*

simulator に指定可能な値は、次のとおりです。

- ・ **mti_se**
- ・ **mti_pe**
- ・ **mti_de**
- ・ **questa**
- ・ **ncsim**
- ・ **riviera**
- ・ **vcs_mx**

-source_lib (ソース ライブラリ)

ライブラリソースファイルを検索する際に、環境変数 `XILINX` (ISE®) または `XILINX_EDK` (EDK) で指定されているデフォルトパスの前に検索するディレクトリを指定します。

メモ: ザイリンクス テクニカル サポートから指示があった場合以外は、このオプションを使用しないでください。

構文

-source_lib *dir_path*

dir_path : ライブラリ ソース ファイルの検索を開始するディレクトリの名前を指定します。

-verbose (詳細メッセージ)

ログ ファイルに詳細なプログラム実行メッセージを記述します。

構文

-verbose

-w (コンパイル済みライブラリの上書き)

コンパイル済みのライブラリを上書きします。デフォルトでは、上書きされません。

構文

-w

Compplib のコマンド ラインの例

このセクションでは、Compplib のコマンド ラインの例を示します。

システム管理者としてのライブラリのコンパイル

システム管理者としてライブラリをコンパイルする場合は、すべてのユーザーがアクセス可能なデフォルトのディレクトリでライブラリをコンパイルする必要があります。

次のコマンド例は、ModelSim SE シミュレータ用にすべてのデバイスおよびすべての言語に対してライブラリをコンパイルする方法を示します。

compplib -s mti_se -arch all

ModelSim SE 6.4b でシミュレーションを実行するのに必要なライブラリがコンパイルされます。次の表に、ライブラリのコンパイル ディレクトリを示します。

ModelSim SE ライブラリのディレクトリ

	VHDL	Verilog
Linux	\$XILINX/vhdl/mti_se/6.4b/lin	\$XILINX/verilog/mti_se/6.4b/lin
Windows	%XILINX%\vhdl\mti_se\6.4b\nt または %XILINX%\vhdl\mti_se\6.4b\nt64	%XILINX%\verilog\mti_se\6.4b\nt または %XILINX%\verilog\mti_se\6.4b\nt64

ユーザーとしてのライブラリのコンパイル

ユーザーとして Compplib を実行する場合は、プロジェクトごとにライブラリをコンパイルすることをお勧めします。サイリンクス デバイスを 1 つのみ使用するプロジェクトの場合、そのデバイスのライブラリのみをコンパイルしてください。

次の例は、NCSim (VHDL) 用に Virtex®-5 デザインの UNISIM および SIMPRIM ライブラリをコンパイルする方法を示します。

```
compplib -s ncsim -arch virtex5 -lib unisim -lib simprim -lang
vhdl -dir ./
```

ライブラリは現在作業中のディレクトリにコンパイルされます。

システム管理者がすべてのライブラリをデフォルトのディレクトリにコンパイルしている場合、ユーザーは必要に応じてこれらのライブラリにマップできます。ライブラリへのマップはプロジェクトごとに行い、プロジェクト ディレクトリでの不要なライブラリ マップを最小限に抑えることをお勧めします。

次の例は、ModelSim PE 用に Virtex-5 デザインのコンパイル済みの UNISIM および XilinxCoreLib ライブラリにマップする方法を示します。

```
compplib -s mti_pe -arch virtex5 -lib unisim -lib xilinxcorelib
```

コンパイル済みのディレクトリにマップする場合、**-w** オプションは指定しないでください。デフォルトのディレクトリにコンパイル済みのライブラリがない場合は、ライブラリがコンパイルされます。

Compplib のその他の使用例

タスク	コマンド
Compplib のヘルプを表示する	compplib -h
特定のオプションのヘルプを表示する	compplib -h <option>
-arch オプションのヘルプを表示する	compplib -h arch
ModelSim SE シミュレータ用に Virtex-5 デバイス (UNISIM、SIMPRIM、および XilinxCoreLib) の Verilog ライブラリすべてをコンパイルし、\$XILINX/verilog/mti_se に上書きする	compplib -s mti_se -arch virtex5 -l verilog -w
ModelSim PE シミュレータ用に Verilog の UNISIM、Uni9000、および SIMPRIM ライブラリをコンパイルし、\$MYAREA ディレクトリに保存する	compplib -s mti_pe -arch all -lib uni9000 -lib simprim -l verilog -dir \$MYAREA
Synopsys 社 VCS および VCS MX シミュレータ用に Virtex-5 デバイスの Verilog の XilinxCoreLib ライブラリをコンパイルし、デフォルトの \$XILINX/verilog/vcs ディレクトリに保存する	compplib -s vcs_mx -arch virtex5 -lib xilinxcorelib
Synopsys 社 VCS および VCS MX シミュレータ用に Verilog の CoolRunner ライブラリをコンパイルし、そのライブラリを現在のディレクトリに保存する	compplib -s vcs_mx -arch coolrunner -lib -dir ./
%XILINX%\xilinxlibs にあるコンパイル済みのライブラリの情報を表示する	compplib -info %XILINX%\xilinxlibs
\$XILINX ディレクトリにある ModelSim SE シミュレータ用にコンパイルされたライブラリ情報を表示する	compplib -info \$XILINX/mti_se/
デフォルトのオプションで compplib.cfg を生成する	compplib -cfg

ランタイム オプションの指定

Compplib のランタイム オプションは、`compplib.cfg` ファイルで指定できます。デフォルトでは、このファイルは現在のディレクトリに生成されます。**-cfg** オプションを使用すると、このファイルをデフォルト設定で自動的に作成できます。詳細は、「[-cfg \(コンフィギュレーション ファイルの作成\)](#)」を参照してください。

次に、コンフィギュレーション ファイルで指定可能なランタイム オプションを示します。

EXECUTE

EXECUTE: on|off

デフォルトでは **on** です。

on : ライブラリをコンパイルします。

off : コンパイルを実行せず、コンパイル コマンドのリストを含む `compplib.log` ファイルのみを生成します。

EXTRACT_LIB_FROM_ARCH

EXTRACT_LIB_FROM_ARCH: on|off

このオプションは、早期アクセス デバイスをサポートします。変更しないでください。

LOCK_PRECOMPILED

LOCK_PRECOMPILED: on|off

デフォルトでは **off** です。

off : 依存ライブラリがコンパイルされていない場合に自動的にコンパイルします。

on : コンパイル済みのライブラリはコンパイルしません。

たとえば、XilinxCoreLib ライブラリをコンパイルする場合、この値に応じて依存ライブラリである UNISIM ライブラリをコンパイルするかが決定されます。

LOG_CMD_TEMPLATE

LOG_CMD_TEMPLATE: on|off

デフォルトでは **off** です。

off : コンパイル コマンドを `compplib.log` ファイルに記述しません。

on : コンパイル コマンドを `compplib.log` ファイルに記述します。

HIER_OUT_DIR

HIER_OUT_DIR: on|off

デフォルトでは **off** です。

off : すべてのライブラリを **-dir** オプションで指定されたディレクトリに保存します。

on : 各シミュレータのライブラリ用に階層出力ディレクトリを作成します。

PRECOMPILED_INFO

PRECOMPILED_INFO: on|off

デフォルトでは **on** です。

on : コンパイルされた日付を含むコンパイル済みライブラリの情報を表示します。

off : この情報を表示しません。

BACKUP_SETUP_FILES

BACKUP_SETUP_FILES: on|off

デフォルトでは **on** です。

on : 前回の実行時に記述されたシミュレータのセットアップ ファイル (ModelSim の場合は modelsim.ini、NCSim の場合は cds.lib および hdl.var、Synopsys VCS および VCS MX の場合は synopsys_sim.setup) すべてのバックアップ ファイルを作成します。

off : セットアップ ファイルのバックアップ ファイルを生成しません。

FAST_COMPILE

FAST_COMPILE: on|off

デフォルトでは **on** です。

on : 指定したライブラリを高速でコンパイルする手法を使用します。

off : この手法を使用せず、通常の手法でコンパイルします。

ABORT_ON_ERROR

ABORT_ON_ERROR: on|off

デフォルトでは **off** です。

off : エラーが発生してもコンパイルを続行します。

on : エラーが検出されるとコンパイルが停止します。

ADD_COMPILATION_RESULTS_TO_LOG

ADD_COMPILATION_RESULTS_TO_LOG: on|off

デフォルトでは **on** です。

on : コンパイル結果を **-log** オプションで指定されたログ ファイルに記述します。

off : **-log** オプションを無視します。

USE_OUTPUT_DIR_ENV

USE_OUTPUT_DIR_ENV: empty|<NAME_OF_ENVIRONMENT_VARIABLE>

デフォルトでは値は指定されません。

値が指定されていない場合、出力ディレクトリの環境変数は検索されず、**-o** オプションで指定されたディレクトリが使用されます。

<NAME_OF_ENV_VAR> の場合は、システム上でこの名前の環境変数が検索され、コンパイルされたライブラリがそのフォルダに出力されます。次に、この例を示します。

CFG ファイル	USE_OUTPUT_DIR_ENV:MY_LIBS
システム設定	setenv MY_LIBS /my_compiled_libs
ライブラリをコンパイルするフォルダ	/my_compiled_libs

OPTION

OPTION

シミュレータ言語のコマンドライン オプションを指定します。

OPTION:Target_Simulator:Language:Command_Line_Options

デフォルトでは、Command_Line_Options で指定されているシミュレータのコンパイル コマンドが使用されます。

コンパイル要件に応じて、Command_Line_Options にオプションを追加したり、指定されているオプションを削除できます。

コンフィギュレーション ファイルの例 (Windows)

次に、デフォルト設定で生成された compxlib.cfg ファイルの例を示します。

```
#####
# $XILINX/bin/lin/unwrapped/compxlib configuration file - compxlib.cfg
# Mon Apr 19 19:36:33 2010
#
# Important :-
# All options/variables must start from first column
#
#####

#
RELEASE_VERSION:12.1
#
RELEASE_BUILD:M.55
#
# set current simulator name
SIMULATOR_NAME:
#
# set current language name
LANGUAGE_NAME:all
#
# set compilation execution mode
EXECUTE:on
#
# print compilation command template in log file
LOG_CMD_TEMPLATE:off
#
# Hierarchical Output Directories
HIER_OUT_DIR:off
#
# print Pre-Compiled library info
PRECOMPILED_INFO:on
#
# create backup copy of setup files
BACKUP_SETUP_FILES:on
#
# use enhanced compilation techniques for faster library compilation
# (applicable to selected libraries only)
FAST_COMPILE:on
#
# save compilation results to log file with the name specified with -log option
```



```

ADD_COMPILATION_RESULTS_TO_LOG:on
#
# abort compilation process if errors are detected in the library
ABORT_ON_ERROR:off
#
# compile library in the directory specified by the environment variable if the
# -dir option is not specified
OUTPUT_DIR_ENV:
#
#////////////////////////////////////
# Setup file name: ModelSim SE
SET:mti_se:MODELSIM=modelsim.ini
#
# ModelSim SE options for VHDL Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# vcom -work <library> <OPTION> <file_name>
#
OPTION:mti_se:vhdl:u:-source -93 -novopt
OPTION:mti_se:vhdl:s:-source -93 -novopt
OPTION:mti_se:vhdl:c:-source -93 -novopt -explicit
OPTION:mti_se:vhdl:r:-source -93 -novopt
OPTION:mti_se:vhdl:i:-source -93 -novopt
OPTION:mti_se:vhdl:e:-93 -novopt -quiet
#
# ModelSim SE options for VERILOG Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# vlog -work <library> <OPTION> <file_name>
#
OPTION:mti_se:verilog:u:-source -novopt
OPTION:mti_se:verilog:s:-source -novopt
OPTION:mti_se:verilog:n:-source -novopt
OPTION:mti_se:verilog:c:-source -novopt
OPTION:mti_se:verilog:r:-source -novopt
OPTION:mti_se:verilog:i:-source -novopt
OPTION:mti_se:verilog:e:-novopt -quiet
#
#////////////////////////////////////
# Setup file name: ModelSim PE
SET:mti_pe:MODELSIM=modelsim.ini
#
# ModelSim PE options for VHDL Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# vcom -work <library> <OPTION> <file_name>
#
OPTION:mti_pe:vhdl:u:-source -93
OPTION:mti_pe:vhdl:s:-source -93
OPTION:mti_pe:vhdl:c:-source -93 -explicit
OPTION:mti_pe:vhdl:r:-source -93
OPTION:mti_pe:vhdl:i:-source -93
OPTION:mti_pe:vhdl:e:-93 -novopt -quiet
#
# ModelSim PE options for VERILOG Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# vlog -work <library> <OPTION> <file_name>
#
OPTION:mti_pe:verilog:u:-source
OPTION:mti_pe:verilog:s:-source
OPTION:mti_pe:verilog:n:-source
OPTION:mti_pe:verilog:c:-source
OPTION:mti_pe:verilog:r:-source
OPTION:mti_pe:verilog:i:-source

```

```

OPTION:mti_pe:verilog:e:-novopt -quiet
#
#////////////////////////////////////
# Setup file name: ModelSim DE
SET:mti_de:MODELSIM=modelsim.ini
#
# ModelSim DE options for VHDL Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#              r (coolrunner)
# vcom -work <library> <OPTION> <file_name>
#
OPTION:mti_de:vhdl:u:-source -93 -novopt
OPTION:mti_de:vhdl:s:-source -93 -novopt
OPTION:mti_de:vhdl:c:-source -93 -novopt -explicit
OPTION:mti_de:vhdl:r:-source -93 -novopt
OPTION:mti_de:vhdl:i:-source -93 -novopt
OPTION:mti_de:vhdl:e:-93 -novopt -quiet
#
# ModelSim DE options for VERILOG Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#              r (coolrunner)
# vlog -work <library> <OPTION> <file_name>
#
OPTION:mti_de:verilog:u:-source -novopt
OPTION:mti_de:verilog:s:-source -novopt
OPTION:mti_de:verilog:n:-source -novopt
OPTION:mti_de:verilog:c:-source -novopt
OPTION:mti_de:verilog:r:-source -novopt
OPTION:mti_de:verilog:i:-source -novopt
OPTION:mti_de:verilog:e:-novopt -quiet
#
#////////////////////////////////////
# Setup file name: QuestaSim
SET:questa:MODELSIM=modelsim.ini
#
# QuestaSim options for VHDL Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#              r (coolrunner)
# vcom -work <library> <OPTION> <file_name>
#
OPTION:questa:vhdl:u:-source -93 -novopt
OPTION:questa:vhdl:s:-source -93 -novopt
OPTION:questa:vhdl:c:-source -93 -novopt -explicit
OPTION:questa:vhdl:r:-source -93 -novopt
OPTION:questa:vhdl:i:-source -93 -novopt
OPTION:questa:vhdl:e:-93 -novopt -quiet
#
# QuestaSim options for VERILOG Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#              r (coolrunner)
# vlog -work <library> <OPTION> <file_name>
#
OPTION:questa:verilog:u:-source -novopt
OPTION:questa:verilog:s:-source -novopt
OPTION:questa:verilog:n:-source -novopt
OPTION:questa:verilog:c:-source -novopt
OPTION:questa:verilog:r:-source -novopt
OPTION:questa:verilog:i:-source -novopt
OPTION:questa:verilog:e:-novopt -quiet
#
#////////////////////////////////////
# Setup file name: ncvhdl
SET:ncsim:CDS=cds.lib
SET:ncsim:HDL=hdl.var
#

```

```

# ncvhdl options for VHDL Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# ncvhdl -work <library> <OPTION> <file_name>
#
OPTION:ncsim:vhdl:u:-MESSAGES -v93 -RELAX -NOLOG
OPTION:ncsim:vhdl:s:-MESSAGES -v93 -RELAX -NOLOG
OPTION:ncsim:vhdl:c:-MESSAGES -v93 -RELAX -NOLOG
OPTION:ncsim:vhdl:r:-MESSAGES -v93 -RELAX -NOLOG
OPTION:ncsim:vhdl:i:-MESSAGES -v93 -RELAX -NOLOG
OPTION:ncsim:vhdl:e:-MESSAGES -v93 -RELAX -NOLOG
#
# ncvhdl options for VERILOG Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# ncvlog -work <library> <OPTION> <file_name>
#
OPTION:ncsim:verilog:u:-MESSAGES -NOLOG
OPTION:ncsim:verilog:s:-MESSAGES -NOLOG
OPTION:ncsim:verilog:n:-MESSAGES -NOLOG
OPTION:ncsim:verilog:c:-MESSAGES -NOLOG
OPTION:ncsim:verilog:r:-MESSAGES -NOLOG
OPTION:ncsim:verilog:i:-MESSAGES -NOLOG
OPTION:ncsim:verilog:e:-MESSAGES -NOLOG
#
# //////////////////////////////////////
# Setup file name: vlogan script version
SET:vcs_mx:SYNOPSIS_SIM=synopsys_sim.setup
#
# vlogan script version options for VHDL Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# vhdlan -work <library> <OPTION> <file_name>
#
OPTION:vcs_mx:vhdl:u:-nc
OPTION:vcs_mx:vhdl:s:-nc
OPTION:vcs_mx:vhdl:c:-nc
OPTION:vcs_mx:vhdl:r:-nc
OPTION:vcs_mx:vhdl:i:-nc
OPTION:vcs_mx:vhdl:e:
#
# vlogan script version options for VERILOG Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# vlogan -work <library> <OPTION> <file_name>
#
OPTION:vcs_mx:verilog:u:+v2k -nc
OPTION:vcs_mx:verilog:s:+v2k -nc
OPTION:vcs_mx:verilog:n:+v2k -nc
OPTION:vcs_mx:verilog:c:+v2k -nc
OPTION:vcs_mx:verilog:r:+v2k -nc
OPTION:vcs_mx:verilog:i:+v2k -nc
#
# //////////////////////////////////////
# Setup file name: Aldec
SET:riviera:LIBRARY=library.cfg
#
# Aldec options for VHDL Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# vcom -work <library> <OPTION> <file_name>
#
OPTION:riviera:vhdl:u:-93 -quiet -nowarn ELAb1_0026

```

```
OPTION:riviera:vhdl:s:-93 -quiet -nowarn ELAB1_0026
OPTION:riviera:vhdl:c:-93 -quiet -nowarn ELAB1_0026
OPTION:riviera:vhdl:r:-93 -quiet -nowarn ELAB1_0026
OPTION:riviera:vhdl:i:-93 -quiet -nowarn ELAB1_0026
OPTION:riviera:vhdl:e:-93 -quiet -nowarn ELAB1_0026
#
# Aldec options for VERILOG Libraries
# Syntax:-
# OPTION:<simulator_name>:<language>:<library>:<options>
# <library> :- u (unisim) s (simprim) c (xilinxcorelib)
#             r (coolrunner)
# vlog -work <library> <OPTION> <file_name>
#
OPTION:riviera:verilog:u:-v2k5 -quiet -msg 0
OPTION:riviera:verilog:s:-v2k5 -quiet -msg 0
OPTION:riviera:verilog:n:-v2k5 -quiet -msg 0
OPTION:riviera:verilog:c:-v2k5 -quiet -msg 0
OPTION:riviera:verilog:r:-v2k5 -quiet -msg 0
OPTION:riviera:verilog:i:-v2k5 -quiet -msg 0
OPTION:riviera:verilog:e:-v2k5 -quiet -msg 0
#////////////////////////////////////
# End
```

XWebTalk

この章では、WebTalk によるデータ収集を制御する XWebTalk コマンド ライン ユーティリティについて説明します。次のセクションが含まれています。

- ・ [XWebTalk の概要](#)
- ・ [XWebTalk の構文](#)
- ・ [XWebTalk のオプション](#)

WebTalk の概要

ISE® Design Suite の WebTalk 機能を使用すると、ザイリンクス FPGA デバイス、ソフトウェア、および IP の使用に関する統計をザイリンクスに送信できます。WebTalk で収集および送信される情報は、ザイリンクスがカスタマに重要な機能を向上することに焦点を当てて開発活動を行い、カスタマの現在および将来のニーズにより迅速に対応できるように活用させていただきます。

イネーブルにすると、Project Navigator、PlanAhead™、Platform Studio、System Generator、XFLOW、またはコマンド ラインを使用してビットストリームを生成した後、および iMPACT を閉じたときに WebTalk により情報がザイリンクスに送信されます。

WebTalk のインストール プリファレンスは、ISE Design Suite のインストール中に設定できます。WebTalk のユーザープリファレンスは、ISE Design Suite、iMPACT、および PlanAhead で設定できます。WebTalk のユーザー プリファレンスを設定するには、次のいずれかを実行します。

- ・ Project Navigator で [Edit] → [Preferences] → [WebTalk] をクリックします。
- ・ iMPACT で [Edit] → [Preferences] → [iMPACT] → [WebTalk] をクリックします。
- ・ PlanAhead™ で [Tools] → [Options] → [General] をクリックします。

XWebTalk を使用すると、コマンド ラインからインストール プリファレンスとユーザー プリファレンスの両方を設定できます。

メモ： WebPACK を使用している場合は、WebTalk は常にイネーブルになっています。WebPACK ライセンスを使用してビットストリームを生成する場合、ユーザーおよびインストールのプリファレンスは無視されます。WebPACK に含まれるデバイスを使用して WebPACK ライセンスが存在する場合、常に WebPACK ライセンスが使用されます。これを変更するには、[アンサー 34746](#) を参照してください。

ビットストリーム生成フローでの WebTalk のデータ送信動作

次の表に、ISE Design Suite ライセンスと WebTalk のインストールおよびユーザー プリファレンス設定に基づく、ビットストリーム生成後の WebTalk によるザイリンクスへのデータ送信動作を示します。

デザイン フロー	ISE Design Suite ライセンス	WebTalk のインストール プリファレンス	WebTalk のユーザー プリファレンス	WebTalk によるザイリンクスへのデータ送信
ビットストリーム生成	WebPACK	無視	無視	送信
	Logic Edition	イネーブル	イネーブル	送信
		イネーブル	ディスエーブル	送信しない
		ディスエーブル	無視	送信しない

iMPACT からの WebTalk のデータ送信動作

次の表に、WebTalk のインストールおよびユーザー プリファレンス設定に基づく、iMPACT からの WebTalk によるザイリンクスへのデータ送信動作を示します。イネーブルにすると、iMPACT の各セッションの終了後 (iMPACT を閉じたとき) に iMPACT により WebTalk を使用して使用統計が送信されます。

デザイン フロー	WebTalk のインストール プリファレンス	WebTalk のユーザー プリファレンス	WebTalk によるザイリンクスへのデータ送信
iMPACT	イネーブル	イネーブル	送信
	イネーブル	ディスエーブル	送信しない
	ディスエーブル	無視	送信しない

XWebTalk の構文

XWebTalk のコマンド ライン構文は、次のとおりです。

xwebtalk [*options*]

options : 「XWebTalk のオプション」にリストされているオプションを指定できます。

XWebTalk のオプション

このセクションでは、XWebTalk のコマンド ライン オプションについて説明します。

- ・ **-user** (ユーザーごとの設定)
- ・ **-install** (インストールごとの設定)
- ・ **-info** (WebTalk 設定の確認)

-user (ユーザーごとの設定)

WebTalk のイネーブル/ディスエーブルをユーザーごとに設定します。

構文

-user on|off

on : 現在のユーザーに対して WebTalk をイネーブルにします。

off : 現在のユーザーに対して WebTalk をディスエーブルにします。

ユーザー設定は、次のディレクトリに保存されます。

- ・ **Windows** : %APPDATA%\Xilinx\Common\version\webtalk (%APPDATA% は C:\Documents and Settings\user\Application Data)
- ・ **Linux** : /home/user/.Xilinx/Common/version/webtalk

例

```
xwebtalk -user on
```

現在のユーザーに対して WebTalk をイネーブルにします。

-install (インストールごとの設定)

WebTalk のイネーブル/ディスエーブルをインストールごとに設定します。

構文

```
-install on|off
```

on : インストールで WebTalk をイネーブルにします。

off : インストールで WebTalk をディスエーブルにします。

インストール設定は、次のディレクトリに保存されます。

- ・ **Windows** : %XILINX%\data\reports\webtalksettings
- ・ **Linux** : \$XILINX/data/reports/webtalksettings

メモ : インストール ディレクトリに書き込むには、管理者権限が必要です。

例

```
xwebtalk -install on
```

インストールで WebTalk をイネーブルにします。

-info (WebTalk 設定の確認)

WebTalk の現在の設定を表示します。

構文

```
-info
```

例

```
xwebtalk -info
```

現在の WebTalk 設定をリストします。

ISE Design Suite のファイル

ザイリンクス ISE® Design Suite および関連するコマンドライン ツールで使用されるファイルをアルファベット順に示します。

名前	タイプ	作成コマンド およびツール	説明
BIT	データ	BitGen	NCD からのコンフィギュレーション情報を含む、デバイスにダウンロードされるビットストリーム ファイル
BGN	ASCII	BitGen	BitGen の実行に関する情報を含むレポート ファイル
BLD	ASCII	NGDBuild	NGDBuild により実行されるサブプロセスも含めた NGDBuild の実行に関する情報を含むレポート ファイル
DATA	C ファイル	TRACE	TRACE を -stamp オプションを使用して実行すると生成されるタイミング モデル情報を含むファイル
DC	ASCII	Synopsys FPGA Compiler	ISE Design Suite および関連のコマンドライン ツールで使用される制約を含む Synopsys 社の設定ファイル
DLY	ASCII	PAR	デザイン内の各ネットの遅延情報を含むファイル
DRC	ASCII	BitGen	BitGen により作成される DRC (デザイン ルール チェック) ファイル
EDIF (各種の拡張子)	ASCII	CAE ベンダー の EDIF 2 0 0 ネットリストライ タ	EDIF ネットリスト (ISE Design Suite および関連のコマンドライン ツールには EDIF 2 0 0 レベル 0 のネットリスト ファイルが読み込まれる)
EDN	ASCII	NGD2EDIF	EDIF 2 0 0 ネットリスト ファイルのデフォルトの拡張子
ELF	ASCII	NetGen で使用	BMM ファイルで指定されたブロック RAM の情報を含むファイル
EPL	ASCII	FPGA Editor	FPGA Editor コマンドのログ ファイル (実行したコマンドと生成された出力をすべて表示、ツールが停止した場合にセッションの回復に使用)

名前	タイプ	作成コマンド およびツール	説明
EXO	データ	PROMGen	Motorola 社の EXORMAT フォーマットの PROM ファイル
FLW	ASCII	ソフトウェアで 提供	XFLOW プログラムのコマンド シーケンス を含むファイル
INI	ASCII	ザイリンクス ソ フトウェア	FPGA Editor の起動時に、実行する FPGA Editor コマンドを決定するスクリプト
GYD	ASCII	CPLDFit	CPLD のガイド ファイル
HEX	16 進数	PROMGen	ビットストリームを 16 進数で表示した PROMGen の出力ファイル
IBS	ASCII	IBISWriter	IBISWriter の出力ファイル (デザインで使用 するピンのリスト、ピンに接続されたデバイス 内部の信号、ピンに接続された IOB に適用 される IBIS バッファ モデルを含む)
JED	JEDEC	CPLDFit	デバイスにダウンロードされるプログラム ファ イル
LOG	ASCII	XFLOW TRACE	XFLOW (xflow.log) および TRACE (macro.log) の実行中に出力されるすべ てのメッセージを含むログ ファイル
LL	ASCII	BitGen	拡張子が .11 のオプションのロジック アロ ケーション ファイル (ラッチ、フリップフロッ プ、IOB の入力/出力のビットストリーム位 置を表す)
MEM	ASCII	テキスト エディ タ	ROM の内容を定義した、編集可能なメモ リ ファイル
MCS	データ	PROMGen	Intel 社の MCS-86 フォーマットの PROM ファイル
MDF	ASCII	MAP	デザインのマップでロジックがどのように分 割されたかを記述したファイル (ガイド マッ プに使用)
MOD	ASCII	TRACE	TRACE を -stamp オプションを使用して実 行すると生成されるタイミング モデル情報 を含むファイル
MRP	ASCII	MAP	MAP コマンドの実行に関する情報を含むレ ポート ファイル
MSK	データ	BitGen	動作するデバイスに含まれたコンフィギュ レーション データをリードバックする際、ビッ ト位置の比較に使用するファイル

名前	タイプ	作成コマンド およびツール	説明
NAV	XML	NGDBuild	サブプロセスも含め、NGDBuild の実行に関する情報を含むレポート ファイル (リンクされたネット名またはインスタンス名をクリックすると、ソース デザインのネットまたはインスタンスに移動可能)
NCD	データ	MAP、PAR、 FPGA Editor	元のデザインに戻るために NGD の物理情報に関連づけられた、フラットな物理デザイン データベース
NCF	ASCII	CAE ベンダー ツールセット	ベンダー固有の論理制約ファイル
NGA	データ	NetGen	バックアノテートされたマップ済みの NCD ファイル
NGC	バイナリ	XST	制約情報を含むネットリスト ファイル
NGD	データ	NGDBuild	Native Generic Database (NGD) ファイル (最初のデザイン作成に使用した階層と、階層を分解した下位のザイリンクス プリミティブの両方で表現したデザインの論理記述を含む)
NGM	データ	MAP	入力 NGD 内のすべてのデータと、マップにより生成された物理的デザインの情報を含むファイル (バックアノテーションに使用)
NGO	データ	ネットリストリー ダ	元のコンポーネントと階層でデザインを論理的に記述したファイル
NKY	データ	BitGen	暗号化キー ファイル
NLF	ASCII	NetGen	NetGen の実行に関する情報を含むログ ファイル
NMC	バイナリ	FPGA Editor	ザイリンクスの物理マクロ ライブラリ ファイル (デザインにインスタンシエーションできる物理的なマクロ定義を含む)
OPT	ASCII	XFLOW	XFLOW で使用されるオプション ファイル
PAD	ASCII	PAR	デザイン内で使用される I/O コンポーネントと、関連のプライマリ ピンのリストを含むファイル
PAR	ASCII	PAR	PAR コマンドの実行に関する情報を含むレポート ファイル (配置配線完了に至るまでのステップを記述)
PCF	ASCII	MAP、FPGA Editor	デザイン入力で回路図に設定した物理制約とユーザー定義の制約が含まれたファイル
PIN	ASCII	NetGen	Cadence 社の信号からピンへのマップ ファイル

名前	タイプ	作成コマンド およびツール	説明
PNX	ASCII	CPLDFit	IBISWriter でインプリメント済みのデザインの IBIS モデルを生成するために使用されるファイル
PRM	ASCII	PROMGen	読み込まれた BIT ファイルに対して PROM の開始および終了アドレスを表す PROM ファイルのメモリマップを含むファイル
RBT	ASCII	BitGen	ビットストリーム ファイルのデータを 1 と 0 で表したロービット ファイル
RPT	ASCII	PIN2UCF	競合する制約が検出された際に生成されるレポート ファイル (pinlock.rpt)
RCV	ASCII	FPGA Editor	FPGA Editor のリカバリ ファイル
SCR	ASCII	FPGA Editor または XFLOW	FPGA Editor または XFLOW のコマンド スクリプト ファイル
SDF	ASCII	NetGen	デザインのタイミング データを含む標準遅延フォーマット ファイル
SVF	ASCII	NetGen	Formality 等価チェックツール用に記述されたアサーション ファイル
TCL	ASCII	テキスト エディタ	Tcl スクリプト ファイル
TDR	ASCII	DRC	物理的 DRC のレポート ファイル
TEK	データ	PROMGen	Tektronix 社の TEKHEX フォーマットの PROM ファイル
TV	ASCII	NetGen	Verilog テストベンチ ファイル
TVHD	ASCII	NetGen	VHDL テストベンチ ファイル
TWR	ASCII	TRACE	TRACE で生成されるタイミング レポート ファイル
TWX	XML	TRACE	TRACE で生成されるタイミング レポート ファイル (リンクされたネット名またはインスタンス名をクリックすると、ソース デザインのネットまたはインスタンスに移動可能)
UCF	ASCII	テキスト エディタ	ユーザー定義の論理制約ファイル
URF	ASCII	テキスト エディタ	ユーザー定義のルール ファイル (使用可能なネットリスト入力ファイル、ネットリストリーダー、ネットリストリーダー オプションに関する情報を含む)
V	ASCII	NetGen	Verilog ネットリスト
VHD	ASCII	NetGen	VHDL ネットリスト
VM6	デザイン	CPLDFit	CPLDFit の出力ファイル

名前	タイプ	作成コマンド およびツール	説明
VXC	ASCII	NetGen	Conformal-LEC 等価チェック ツール用に記 述されたアサーション ファイル
XCT	ASCII	PARTGen	アーキテクチャとデバイスの詳細情報を含 むファイル
XTF	ASCII	旧リリースのザ イリンクス ソフ トウェア	ザイリンクスのネットリスト フォーマット ファイル
XPI	ASCII	PAR	PAR の実行サマリを含むファイル

EDIF2NGD と NGDBuild

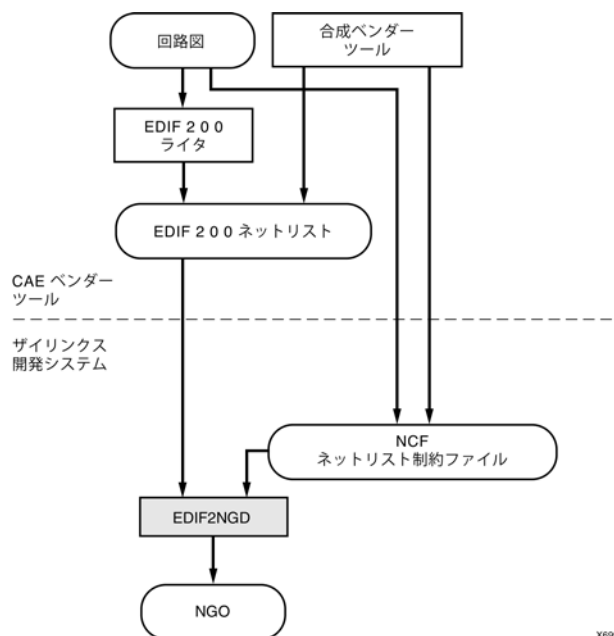
この付録では、ネットリストリーダー プログラム EDIF2NGD と、その NGDBuild との関係について説明します。この付録は、次のセクションから構成されています。

- ・ [EDIF2NGD の概要](#)
- ・ [EDIF2NGD のオプション](#)
- ・ [NGDBuild](#)
- ・ [ネットリスト ラウンチャ](#)
- ・ [NGDBuild で使用するファイルの名前と保存先](#)

EDIF2NGD の概要

EDIF2NGD プログラムを使用すると、ザイリンクスのツールセットに EDIF (Electronic Data Interchange Format) 2 0 0 ファイルを読み込むことができます。EDIF2NGD は、業界標準の EDIF ネットリストをザイリンクス固有のフォーマットである NGO ファイルに変換します。EDIF ファイルには、入力回路図の階層が含まれています。出力される NGO ファイルは、入力デザイン ファイルで指定したコンポーネントと階層に基づいてデザインを記述したバイナリ データベースです。EDIF ファイルを NGO ファイルに変換した後、NGDBuild を実行して NGD ファイルを作成すると、デザインが展開され、ザイリンクス プリミティブで記述されます。

EDIF2NGD デザイン フロー



X6994

EDIF2NGD のデバイス サポート

このプログラムは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

EDIF2NGD の構文

EDIF ネットリストを読み込み、NGO ファイルに変換するには、次のコマンドを使用します。

edif2ngd [*options*] *edif_file* *ngo_file*

- ・ *options*: 「EDIF2NGD のオプション」にリストされているオプションをいくつでも指定できます。オプションは、特定の順序で入力する必要はありません。オプションの前にはハイフン (-) を付け、複数のオプションはスペースで区切ります。
- ・ *edif_file*: 変換する EDIF200 入力ファイルです。拡張子を付けずにファイル名を入力すると、入力したファイル名に拡張子 **.edn** が付いたファイルが使用されます。**.edn** 以外の拡張子がファイルに付いている場合は、その拡張子を *edif_file* の一部として入力する必要があります。

メモ: EDIF2NGD で Mentor Graphics 社の EDIF ファイルを読み込む場合、あらかじめシステム上に Mentor Graphics のソフトウェア コンポーネントをインストールしておいてください。同様に、Cadence 社の EDIF ファイルを読み込む場合も、あらかじめ Cadence 社のソフトウェア コンポーネントをインストールしておく必要があります。

- ・ *ngo_file*: NGO フォーマットの出力ファイルです。出力ファイルの名前、拡張子、ディレクトリは、次のように決定されます。
 - 出力ファイル名を指定しない場合、出力ファイルのベース名は入力ファイルと同じで、拡張子 **.ngo** が付けられます。
 - 拡張子を付けずに出力ファイル名を指定すると、ファイル名に拡張子 **.ngo** が付けられます。
 - 拡張子が **.ngo** 以外のファイル名を指定すると、エラー メッセージが表示され、EDIF2NGD は実行されません。
 - 完全パス名を指定しない場合、出力ファイルが EDIF2NGD を実行したディレクトリ内に保存されます。既に出力ファイルが存在している場合、そのファイルは新規ファイルにより上書きされます。

EDIF2NGD の入力ファイル

EDIF2NGD に使用する入力ファイルは、次のとおりです。

- ・ **EDIF ファイル**: EDIF 2 0 0 ネットリスト ファイル。EDIF200 仕様で定義されているように、レベル 0 の EDIF ネットリスト ファイルを使用する必要があります。ザイリンクスのツールセットは、次のライブラリからのコンポーネントを使用して EDIF ファイルを識別します。
 - ザイリンクス ユニファイド ライブラリ (ライブラリ ガイドを参照)
 - XSI (Xilinx Synopsys Interface) ライブラリ
 - ユーザーが作成したザイリンクスの物理マクロ

メモ: ザイリンクスのツールでは、マクロとして定義されたザイリンクス ユニファイド ライブラリのコンポーネントは識別されず、ライブラリのプリミティブのみが識別されます。サードパーティの EDIF ライタには、すべてのマクロの定義が含まれている必要があります。

- ・ **NCF ファイル**: ネットリスト制約ファイル。ベンダーのツールセットを使用して設定した制約がこのファイルに出力されます。EDIF2NGD は、このファイルに記述された制約を読み込み、NGO ファイルに出力します。

NCF ファイルに入力 EDIF ファイルと同じルート名と拡張子 **.ncf** が付いている場合、NCF ファイルの制約が読み込まれます。NCF のファイル名を EDIF2NGD コマンドラインに入力する必要はありません。

EDIF2NGD の出力ファイル

EDIF2NGD から出力されるファイルは、NGO です。このファイルは、オリジナルのコンポーネントと階層で表現したデザインの論理記述を含むバイナリ ファイルです。

EDIF2NGD のオプション

このセクションでは、EDIF2NGD のコマンドライン オプションについて説明します。

- ・ `-a` (最上位ポート信号への PAD の追加)
- ・ `-aul` (不一致の LOC を許可)
- ・ `-f` (コマンド ファイルの実行)
- ・ `-intstyle` (統合スタイル)
- ・ `-l` (検索するライブラリ)
- ・ `-p` (製品番号)
- ・ `-r` (LOC 制約の無視)

`-a` (最上位ポート信号への PAD の追加)

すべての最上位ポート信号に PAD プロパティを追加します。このオプションは、PAD シンボルがポートに変換されている EDIF ファイルを入力ファイルとして読み込む場合に使用する必要があります。EDIF ファイルに `-a` オプションを指定しない場合、EDIF ファイル内に PAD インスタンスがないため、デザインが正しく読み込まれません。その結果、MAP でこれらのロジックが未使用であると解釈され、削除されます。

構文

`-a`

Mentor Graphics 社と Cadence 社の EDIF ファイルでは、PAD シンボルがポートに変換されます。これらのベンダーの EDIF ファイルを使用する場合、`-a` オプションは自動的に設定されているため、EDIF2NGD コマンドラインに `-a` オプションを入力する必要はありません。

`-aul` (不一致の LOC を許可)

EDIF2NGD をデフォルト (`-aul` オプションなし) で実行した場合、NCF ファイルでピン名、ネット名、インスタンス名に設定した制約がデザイン内で識別されないと、エラーが発生します。このエラーが発生すると、NGO ファイルは生成されません。`-aul` オプションを使用すると、LOC 制約に対してエラーではなく警告が表示され、NGO ファイルが生成されます。

HDL または回路図で定義していないピン名、ネット名、インスタンス名に設定したロケーション制約が制約ファイルに含まれている場合、`-aul` オプションを使用して NGDBuild を実行します。これにより、作成中のデザインと完成デザインの両方に、同じ制約ファイルを適用できます。

構文

`-aul`

メモ： このオプションを使用する場合、デザインに含まれるネット名およびインスタンス名にスペルミスがないことを確認してください。スペルミスがあると、配置配線が正しく実行されない場合があります。

-f (コマンド ファイルの実行)

指定したコマンド ファイル (*command_file*) 内のコマンド ライン引数を実行します。

構文

-f *command_file*

-f オプションの詳細は、「はじめに」の章の「[-f\(コマンド ファイルの実行\)](#)」を参照してください。

-intstyle (統合スタイル)

実行している統合スタイルによって、画面に出力されるメッセージを制限します。

構文

-intstyle *ise* | *xflow* | *silent*

次のいずれかのモードに設定します。

- ・ **-intstyle ise** : プログラムが統合デザイン環境の一部として実行されます。
- ・ **-intstyle xflow** : プログラムが統合バッチ フローの一部として実行されます。
- ・ **-intstyle silent** : 画面への出力を、警告およびエラー メッセージのみに制限します。

メモ : Project Navigator や XFLOW などの統合環境を使用している場合、このオプションは自動的に設定されます。

-l (検索するライブラリ)

デザイン構築に使用されたライブラリ コンポーネントを識別する際に、検索するライブラリを指定します。この情報は、NGDBuild でコンポーネントをザイリンクス プリミティブに変換する際、コンポーネント ソースを識別するために必要です。

構文

-l *libname*

このオプションは、1 つのコマンド ラインに複数回入力できますが、ライブラリ名の前にそれぞれ **-l** を付ける必要があります。たとえば、「**-l xilinxun -l synopsys**」とは指定できませんが、「**-l xilinxun synopsys**」とは指定できません。

libname に使用できる値は、次のとおりです。

- ・ **xilinxun** (ザイリンクス ユニファイド ライブラリ)
- ・ **synopsys**

メモ : このオプションを使用して **xilinxun** を指定する必要はありません。ザイリンクス ツールは、これらのライブラリに自動的にアクセスします。EDIF ネットリストに「Synopsys」という文字列を使用した文が含まれている場合、**-l** オプションで **synopsys** を指定する必要はありません。この場合、デザインが Synopsys からであることが自動的に識別されます。

-p (製品番号)

デザインをインプリメントするパーツを指定します。

メモ : EDIF2NGD を実行する際にパーツを指定しない場合は、NGDBuild を実行する際に指定する必要があります。

構文

-p *part number*

part number デバイス、パッケージ、スピードを含む完全な製品番号を指定する必要があります (例 : xc4vlx60-10- ff256)。

メモ : 構文の詳細および例は、「はじめに」の章の「**-p (製品番号)**」を参照してください。

-r (LOC 制約の無視)

デザインのすべてのロケーション制約 (LOC) を無視します。出力ファイルが既に存在する場合、そのファイルは新規ファイルで上書きされます。

構文

-r

NGDBuild

NGDBuild は、EDIF ネットリスト ファイルを読み込み、論理デザインが記述された NGD ファイルを生成するのに必要なすべてのステップを実行します。NGDBuild により生成される NGD ファイルには、NGD プリミティブで表現されたデザインの論理記述と、入力ネットリストからの階層で表現された記述が含まれます。出力された NGD ファイルは、使用するデバイス ファミリーにマップできます。

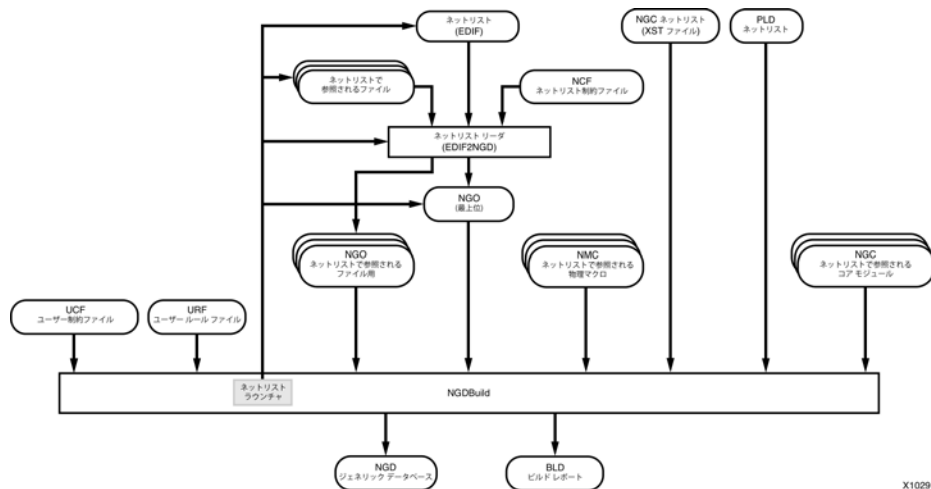
このプログラムは、次のファミリーで使用できます。

- ・ Virtex®-4
- ・ Virtex-5
- ・ Spartan®-3
- ・ Spartan-3A
- ・ Spartan-3E
- ・ CoolRunner™ XPLA3
- ・ CoolRunner-II
- ・ XC9500 シリーズ

ネットリストから NGD ファイルへの変換

次に、NGDBuild の変換プロセスを示します。

NGDBuild およびネットリスト リーダ



NGDBuild は、次の手順でネットリストを NGD ファイルに変換します。

1. ソース ネットリストを読み込みます。

NGDBuild の一部であるネットリスト ラウンチャが起動します。このプログラムは、入力ネットリストの種類を識別し、適切なネットリストリーダを起動します。入力ネットリストが EDIF フォーマットであれば EDIF2NGD が起動します。入力ネットリストがそれ以外のフォーマットであっても、ネットリスト ラウンチャで識別されるフォーマットであれば、適切なプログラムで EDIF フォーマットに変換してから、EDIF2NGD を起動します。EDIF2NGD は、最上位ネットリストに対して NGO ファイルを生成します。

最上位ネットリスト内でサブファイル (PAL 記述ファイルや別の回路図ファイルなど) が参照されている場合、ネットリスト ラウンチャがサブファイルごとに適切なネットリストリーダを起動し、参照ファイルを NGO ファイルに変換します。

ネットリスト ラウンチャについては、「[ネットリスト ラウンチャ](#)」を参照してください。ネットリストリーダ プログラムについては、「[EDIF2NGD の概要](#)」を参照してください。

2. デザイン内のすべてのコンポーネントを NGD プリミティブに置き換えます。

NGDBuild は、参照された NGO ファイルを検索し、ほかのファイルを参照するコンポーネントを結合します。また、適切なシステムライブラリのコンポーネント、物理的マクロ (NMC ファイル)、ビヘイビアモデルを検索します。

3. 変換されたデザインで論理的 DRC (デザイン ルール チェック) を実行し、デザインを検証します。

論理的 DRC は、論理デザインで実行する一連のテストです。詳細は、「[論理的デザイン ルール チェック](#)」の章を参照してください。

4. NGD ファイルを出力します。

NGDBuild は、ソース ネットリストを読み込む際に、前回の NGDBuild の実行後に変更されたファイルやデザインの一部を検出し、次のようにファイルをアップデートします。ファイルは、次のようにアップデートされます。

- ・ 入力デザインを変更した場合、変更の影響を受けたすべてのファイルがアップデートされ、アップデートされたファイルを使用して新たに NGD ファイルが生成されます。
ネットリスト ラウンチャは、ネットリスト ファイルと中間ファイル (NGO) のタイムスタンプをチェックします。NGO のタイムスタンプがネットリストよりも古い場合、NGO がアップデートされ、新しい NGD ファイルが生成されます。
- ・ すべてまたは一部の中間ファイルが既に存在している場合、これらのファイルを使用して NGD が生成されます。中間ファイルは、NGDBuild の実行前にネットリストリーダを実行した場合、存在していることがあります。NGDBuild は、これらの既存ファイルを使用して、NGD ファイルの生成に必要なファイルを生成します。

メモ: ネットリスト ファイルの NGO が最新であれば、ネットリストと同じルート名を持つ NCF ファイルがネットリスト ディレクトリで検索され、NGO と NCF のタイムスタンプが比較されます。NCF ファイルの方が新しい場合は、EDIF2NGD が再実行されます。ただし、前回の NGDBuild の実行で存在していた NCF ファイルを削除してしまうと、EDIF2NGD が実行されません。この場合は、**-nt on** オプションを使用して EDIF2NGD を再実行する必要があります。EDIF2NGD のオプションを変更した場合にも、**-nt on** オプションを使用して再実行してください。

NGDBuild の構文、ファイル、オプションについては、「[NGDBuild](#)」の章を参照してください。

バスの一致

あるネットリストのインスタンスが別のネットリストで検出された場合、上位のインスタンスに指定された各ピンと、下位のネットリストのピン (またはポート) が一致している必要があります。一致するためには、2 つのピンの名前が同じである必要があります。この条件は、NGDBuild でサポートされているすべての FPGA および CPLD に適用されます。

2 つのネットリスト間のインターフェイスでピンをバスとして使用している場合は、これらのピンをスカラピンに展開してから、ピンの一致が確認されます。たとえば、ピン A[7:0] は、A[7] から A[0] までの名前を持つ 8 つのピンに展開されます。両方のネットリストで同じ命名規則 (区切り文字) を使用していれば、バスピンを展開した際に、スカラピンの名前が正確に一致します。ただし、2 つのネットリストが異なるベンダー ツールで生成され、異なる区切り文字が使用されている場合は、展開されたスカラピンの名前が正確に一致しません。

スカラピンの名前が正確に一致しない場合、NGDBuild は両方のネットリストのピン名を解析して、展開されたバスピンの名前を識別しようとします。バスピン名を識別できたら、もう 1 つのネットリストのバス命名規則で一致する名前がないかどうかを確認し、2 つのネットリストをマージしようとします。たとえば、あるネットリストに A(3) という名前のピンを検出した場合、もう 1 つのネットリストに A(3)、A[3]、A<3>、A3 という名前のピンがないかどうかを確認します。

NGDBuild により識別されるバス命名規則は次のとおりです。

バスの命名規則

命名規則	例
<i>busname(index)</i>	DI(3)
<i>busname<index></i>	DI<3>
<i>busname[index]</i>	DI[3]
<i>busnameindex</i>	DI3

サードパーティのネットリストライターでバス命名規則を指定できる場合は、NGDBuild 実行時にピンの不一致エラーが発生しないよう上記の表記のいずれかを使用してください。サードパーティの EDIF ライターで EDIF の array 文を使用してバスピンが保持される場合、バスピンは EDIF2NGD によりかっこ付きで展開されます。この命名は、NGDBuild でサポートされます。

メモ： NGDBuild では、上記の命名規則によりバスピンが認識されるだけで、2 つのネットリストファイルでバスピンに異なる数値範囲を使用していると、ネットリストを結合できません。たとえば、あるネットリストの A[7:0] を、別のネットリストの A[15:8] と一致させることはできません。

ザイリンクス UnifiedPro ライブラリでは、ブロック RAM プリミティブのピンの一部がバスとして使用されています。サードパーティのネットリストライターで上記のバス命名規則が使用されていたり、EDIF の array 文が使用されている場合、NGDBuild によりそのプリミティブが正しく識別されます。ほかの命名規則が使用されている場合は、NGDBuild 実行時に、ブロックが展開できないことを示すエラーが発生することがあります。

ネットリスト ラウンチャ

ネットリスト ラウンチャは、NGDBuild の一部で、EDIF ネットリストを NGO ファイルに変換します。NGO ファイルは、NGD ファイルを生成するため NGDBuild で使用されます。

メモ： NGC ネットリスト ファイルは、NGO と同等であるため、ネットリスト ラウンチャで処理する必要はありません。

NGDBuild を起動すると、ネットリスト ラウンチャが次の処理を実行します。

1. 各ネットリストに対応するネットリストリーダを判断し、各ネットリストリーダの起動に使用するオプションを決定するルールに従って、ネットリスト ラウンチャを初期化します。

これらのルールは、システム ルール ファイル ([「システム ルール ファイル」](#)を参照) およびユーザー ルール ファイル ([「ユーザー ルール ファイル \(URF\)」](#)を参照) に含まれています。

2. 最上位ネットリストのディレクトリを、ネットリスト ラウンチャの検索パスリストの最初に設定します。
3. 最上位デザインと最上位デザインで参照される各ファイルに対応する NGO ファイルがあるかどうかを確認します。
4. 各 NGO ファイルに対して次の処理を実行します。

- ・ NGO ファイルのソース ネットリストを検出する。

ルール データベース内で有効なネットリストの拡張子を検索し、ソース ネットリストを識別します。作業ディレクトリを含む検索パス内を検索し、NGO ファイルと同じ名前で、有効な拡張子の付いたネットリスト ファイルを検出します。

- ・ NGO ファイルを検索する。

-dd オプションで指定したディレクトリ (ディレクトリを指定しない場合は作業ディレクトリ) 内を検索します。ディレクトリで NGO を検出できず、検索パスでソース ネットリストを検出できなかった場合は、検索パスで NGO を検索します。

- ・ NGO ファイルを生成またはアップデートする必要があるかどうかを判断する。

ネットリスト ソース ファイルと NGO のいずれも検出できなかった場合、エラーが発生して NGDBuild が終了します。

ネットリスト ソース ファイルは検出したが、対応する NGO を検出できなかった場合は、適切なネットリストリーダを実行して NGO ファイルを生成します。

ネットリスト ソース ファイルを検出できず、対応する NGO を検出した場合は、ネットリスト ラウンチャが NGO の存在を NGDBuild に伝え、NGDBuild が NGO を使用します。

ネットリスト ソース ファイルと対応する NGO の両方が検出された場合は、ネットリスト ファイルと NGO のタイムスタンプが比較されます。NGO のタイムスタンプがネットリストよりも新しい場合、ネットリスト ラウンチャは「found」ステータスを NGDBuild に返します。NGO のタイムスタンプがネットリストよりも古い場合、NGO が予期された位置に存在しない場合は、ルール ファイルで指定されているネットリストリーダを実行してネットリスト ソース ファイルから NGO を生成します。

メモ： タイムスタンプの確認は、NGDBuild コマンドライン オプションで変更できます。**-nt on** オプションを使用すると、タイムスタンプにかかわらず、既存のすべての NGO がアップデートされます。**-nt off** オプションを使用すると、タイムスタンプにかかわらず、既存の NGO はアップデートされません。

5. ネットリスト ラウンチャは、NGO が検出され、NGO の処理が可能なことを NGDBuild に伝えます。

ネットリスト ラウンチャのルール ファイル

ネットリスト ラウンチャの動作は、システム ルール ファイルとユーザー ルール ファイルで定義されたルールによって決まります。これらのルールは、次のことを決定します。

- ・ どのネットリスト ソース ファイルを使用できるか
- ・ ネットリスト ファイルをどのネットリスト リーダで読み込むか
- ・ ネットリスト リーダにどのデフォルトのオプションを使用するか

システム ルール ファイルには、ザイリンクスが提供するデフォルトのルールが含まれています。ユーザー ルール ファイルを使用すると、ルールを追加したり、システム ルールより優先させることができます。

ユーザー ルール ファイル (URF)

ユーザー ルール ファイルを使用すると、ルールを追加したり、システム ルールより優先させることができます。ユーザー ルール ファイルの保存先は、**-ur** オプションを使用して指定します。このファイルの拡張子は **.urf** にする必要があります。詳細は、この章の「[-ur \(ユーザー ルール ファイルの読み込み\)](#)」を参照してください。

ユーザー ルールとシステム ルール

ユーザー ルールは、次のように処理されます。

- ・ ユーザー ルールとシステム ルールで同じソース ファイルとターゲット ファイルを指定している場合、ユーザー ルールが優先されます。
- ・ ユーザー ルールのターゲット ファイルがシステム ルールのソース ファイルと同じであるか、ユーザー ルールのソース ファイルがシステム ルールのターゲット ファイルと同じである場合、ユーザー ルールはシステム ルールの補足となります。
- ・ ユーザー ルールのソース ファイルがシステム ルールのターゲット ファイルと同じで、ユーザー ルールのターゲット ファイルがシステム ルールのソース ファイルと同じ場合、このユーザー ルールはループとなるため無効になります。

ユーザー ルールのフォーマット

ユーザー ルールのフォーマットは、次のとおりです。

```
RuleName = <rulename1>;  
<key1>   = <value1>;  
<key2>   = <value2>;  
.  
.  
.  
<keyn>   = <valuen>;
```

キーとその値は、次のとおりです。

メモ： キーの値のタイプについては、「キー ステートメントの値のタイプ」を参照してください。

- ・ **RuleName** : ルールの開始を示します。ルールに関連するエラー メッセージにも使用されます。値には、RULENAME を指定する必要があります。
- ・ **NetlistFile** : ネットリストリーダに読み込まれるネットリストまたはネットリストのクラスを指定します。ルールを識別するため TargetExtension と共に使用されます。値には、FILENAME または EXTENSION を指定する必要があります。ファイルを指定する場

合、パスを指定しても無視されるので、パスを指定せずにファイル名のみを指定します。値は必須です。

- ・ **TargetExtension** : ネットリストリーダで生成されるファイルのクラスを指定します。ルールを識別するため NetlistFile の拡張子と共に使用されます。値には、EXTENSION を指定する必要があります。
- ・ **Netlister** : ネットリストまたはネットリストのクラスをターゲットファイルに変換するために使用するネットリストリーダを指定します。ネットリストまたはネットリストのクラスは NetlistFile で指定し、ターゲットファイルのクラスは TargetExtension で指定します。値には、EXECUTABLE を指定する必要があります。
- ・ **NetlisterTopOptions** : 最上位デザインをコンパイルするときにネットリストリーダで使用するオプションを指定します。値には、OPTIONS または NONE キーワードを指定します。文字列には、\$INFILE と \$OUTFILE キーワードを含める必要があります。キーワードは、それぞれ入力ファイルと出力ファイルに置き換えられます。また、次のキーワードが表示される場合があります。
 - **\$PART : -p** オプションにより NGDBuild に渡されたパーツに置き換えられます。これには、アーキテクチャ、デバイス、パッケージ、スピードの情報を含めることができます。\$PART の構文は、「はじめに」の章の「**-p (製品番号)**」を参照してください。
 - **\$FAMILY : -p** オプションにより NGDBuild に渡されたファミリに置き換えられます。値はオプションです。
 - **\$DEVICE : -p** オプションにより NGDBuild に渡されたデバイスに置き換えられます。値はオプションです。
 - **\$PKG : -p** オプションにより NGDBuild に渡されたパッケージに置き換えられます。値はオプションです。
 - **\$SPEED : -p** オプションにより NGDBuild に渡されたスピードに置き換えられます。値はオプションです。
 - **\$LIBRARIES** : NGDBuild に渡されるライブラリ。値はオプションです。
 - **\$IGNORE_LOCS** : NGDBuild コマンドライン構文に **-r** オプションが含まれている場合、EDIF2NGD の **-r** オプションに置き換えられます。
 - **\$ADD_PADS** : NGDBuild コマンドライン構文に **-a** オプションが含まれている場合、EDIF2NGD の **-a** オプションに置き換えられます。

NetlisterTopOptions 行のオプションは、ダブルクォーテーション (") で囲む必要があります。
- ・ **NetlisterOptions** : サブデザインをコンパイルするときにネットリストリーダで使用するオプションを指定します。値には、OPTIONS または NONE キーワードを指定します。文字列には、\$INFILE と \$OUTFILE キーワードを含める必要があります。キーワードは、それぞれ入力ファイルと出力ファイルに置き換えられます。NetlisterTopOptions に使用できるキーワードは、NetlisterOptions にも使用できます。

NetlisterOptions 行のオプションは、ダブルクォーテーション (") で囲む必要があります。

- ・ **NetlisterDirectory** : ネットリストリーダを実行するディレクトリを指定します。ネットリストラウンチャは、ネットリストリーダを実行する前に、このディレクトリに移動します。このキーには、DIR 値またはキーワード \$SOURCE、\$OUTPUT、NONE のいずれかを指定します。\$SOURCE はソース ネットリストに、\$OUTPUT は **-dd** オプションで指定したディレクトリに、NONE は作業中のディレクトリに置き換えられます。値はオプションです。
- ・ **NetlisterSuccessStatus** : ネットリストリーダが正常に実行された場合に返されるリターンコードを指定します。値には、NUMBER または NONE キーワードをオプションで

指定します。数値の前には、「=」、「<」、「>」、「!」のいずれかを付けることができます。値はオプションです。

キー ステートメントの値のタイプ

キー ステートメントで使用する値のタイプは、次のとおりです。

- ・ **RULENAME** : セミコロン (;) と空白 (スペース、タブ、改行など) 以外の文字を使用した任意の文字列
- ・ **EXTENSION** : ペリオド (.) の後にプラットフォームの要件に適合する拡張子が付いたもの
- ・ **FILENAME** : プラットフォームの要件に適合するファイル名
- ・ **EXECUTABLE** : プラットフォームの要件に適合する実行ファイル名。完全パスも含めた実行ファイル名か、実行ファイル名のみを指定します。ファイル名の場合は、実行ファイルの検索に \$PATH 環境変数が使用されます。
- ・ **DIR** : プラットフォームの要件に適合するディレクトリ名
- ・ **OPTIONS** : 実行ファイルに有効なオプション文字列
- ・ **NUMBER** : 数値
- ・ **文字列** : ダブルクォーテーション (") で囲まれた文字列

システム ルール ファイル

システム ルールは次のとおりです。システム ルール ファイルは ASCII 形式のファイルではありませんが、ルールを説明するため、ここではユーザー ルール ファイルと同じ構文を使用し、記述しています。ユーザー ルール ファイルの構文は、「[ユーザー ルール ファイル \(URF\)](#)」を参照してください。

メモ : ルール属性が指定されていない場合は、NONE と見なして処理されます。

システム ルール ファイル

```
#####
# edif2ngd rules
#####

RuleName = EDN_RULE;
NetlistFile = .edn;
TargetExtension = .ngo;
Netlister = edif2ngd;
NetlisterTopOptions = "[$IGNORE_LOCS] [$ADD_PADS] [$QUIET] [$AUL] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterOptions = "-noa [$IGNORE_LOCS] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterDirectory = NONE;
NetlisterSuccessStatus = 0;

RuleName = EDF_RULE;
NetlistFile = .edf;
TargetExtension = .ngo;
Netlister = edif2ngd;
NetlisterTopOptions = "[$IGNORE_LOCS] [$ADD_PADS] [$QUIET] [$AUL] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterOptions = "-noa [$IGNORE_LOCS] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterDirectory = NONE;
NetlisterSuccessStatus = 0;

RuleName = EDIF_RULE;
NetlistFile = .edif;
TargetExtension = .ngo;
Netlister = edif2ngd;
NetlisterTopOptions = "[$IGNORE_LOCS] [$ADD_PADS] [$QUIET] [$AUL] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterOptions = "-noa [$IGNORE_LOCS] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterDirectory = NONE;
NetlisterSuccessStatus = 0;

RuleName = SYN_EDIF_RULE;
NetlistFile = .sedif;
TargetExtension = .ngo;
Netlister = edif2ngd;
NetlisterTopOptions = NONE;
NetlisterOptions = "-l synopsys [$IGNORE_LOCS] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterDirectory = NONE;
NetlisterSuccessStatus = 0;
```

ルール ファイルの例

このセクションでは、システム ルールとユーザー ルールの例を示します。最初の例は、それ以降のユーザー ルールの例を理解するために示しています。

例 1 : EDF_RULE システム ルール

「システム ルール ファイル」に示すように、EDF_RULE システム ルールは次のように定義されます。

```
RuleName = EDF_RULE;
NetlistFile = .edf;
TargetExtension = .ngo;
Netlister = edif2ngd;
NetlisterTopOptions = "[$IGNORE_LOCS] [$ADD_PADS] [$QUIET] [$AUL] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterOptions = "-noa [$IGNORE_LOCS] {-1 $LIBRARIES} $INFILE $OUTFILE";
NetlisterDirectory = NONE;
NetlisterSuccessStatus = 0;
```

この EDF_RULE システム ルールでは、EDIF2NGD を使用して EDIF ファイルを NGO ファイルに変換するよう定義しています。最上位ネットリスト ファイルの変換には、NetlisterTopOptions で定義されたオプションが使用され、下位ネットリスト ファイルの変換には、NetlisterOptions で定義されたオプションが使用されます。NetlisterDirectory は NONE に設定されているため、EDIF2NGD は NGDBuild を起動したときの作業ディレクトリで実行されます。正常に変換された場合、リターン コード 0 が返されます。それ以外の値が返された場合は、変換できなかったことを示します。

例 2 : ユーザー ルール

```
// URF Example 2
RuleName = OTHER_RULE; // end-of-line comments are also allowed
NetlistFile = .oth;
TargetExtension = .edf;
Netlister = other2edf;
NetlisterOptions = "$INFILE $OUTFILE";
NetlisterSuccessStatus = 1;
```

ユーザー ルール OTHER_RULE は、仮想の OTH ファイルから EDIF ファイルへの変換を定義しています。変換には、other2edf プログラムが使用されます。NetlisterOptions が指定されていないため、ネットリストが最上位か下位かにかかわらず、NetlisterOptions で定義されたオプションがすべての OTH ファイルの変換に使用されます。正常に変換された場合、リターンコード 1 が返されます。それ以外の値が返された場合は、変換できなかったことを示します。

OTHER_RULE を使用して other2edf を実行し、EDIF ファイルを生成した後、EDF_RULE システム ルール (例 1) を使用して EDIF ファイルを NGO ファイルに変換します。

例 3 : ユーザー ルール

```
// URF Example 3
RuleName = EDF_LIB_RULE;
NetlistFile = .edf;
TargetExtension = .ngo;
NetlisterOptions = "-l xilinxun $INFILE $OUTFILE";
```

ユーザー ルールの NetlistFile と TargetExtension で指定した拡張子は、EDF_RULE システム ルール (例 1) で指定した拡張子と同じなので、EDF_LIB_RULE が優先されます。EDF_LIB_RULE で設定が定義されていない場合は、EDF_RULE と同じ設定が使用されます。EDF_LIB_RULE では、ネットリスト ライタ (EDIF2NGD)、最上位オプション、ディレクトリに EDF_RULE と同じものを使用するので、EDF_RULE を使用したときと同じ結果が得られます。下位のネットリストを変換する場合、使用するオプションは「**-l xilinxun \$INFILE \$OUTFILE**」のみです。実際には、EDIF2NGD に **-l xilinxun** を使用する必要はありません。ここでは、説明のため使用しています。

例 4 : ユーザー ルール

```
// URF Example 4
RuleName = STATE_EDF_RULE;
NetlistFile = state.edf;
TargetExtension = .ngo;
Netlister = state2ngd;
```

NetlistFile には完全なファイル名が指定されていますが、ユーザー ルールと EDF_RULE システム ルール (例 1) の NetlistFile と TargetExtension の拡張子と同じであるため、2 つのルールは一致しています。state.edf が存在すると仮定すると、ファイル state.ngo の生成には EDF_RULE システム ルールではなく、このルールが使用されます。例 3 のように、設定が指定されていない場合は、一致するシステム ルールの設定が使用されます。ただしこの場合は、EDIF2NGD ではなく、架空のプログラム state2ngd が使用されます。

メモ : EDF_LIB_RULE (例 3) とこの規則の両方がユーザールールファイルに含まれている場合は、EDF_LIB_RULE による変更が STATE_EDF_RULE に含まれます。このため、下位の state.edf ファイルは、**-l xilinxun** オプションを使用して state2ngd を実行することで変換されます。

NGDBuild で使用するファイルの名前と保存先

NGDBuild で使用するファイル名に関しては、次の点に注意してください。

- ・ 中間ファイルには、そのファイルが生成されたデザインと同じルート名が付いています。中間ファイルは、ネットリストを NGO に変換するため複数のネットリストリーダーが必要な場合に生成されます。
- ・ 検索パス内のネットリスト ファイルのルート名が、別のファイル名と重複していないことを確認してください。たとえば、state.edn というデザインを作成した場合、検索パスで指定したディレクトリ内に state という名前が付いた別のデザインを含めることはできません。
- ・ NGDBuild とネットリスト ラウンチャでは、ファイル名をダブルクォーテーション (") で囲んで指定します。ファイル名をダブルクォーテーションで囲むと、通常は使用できない特殊文字 (スペースなど) を含んだファイル名を指定できます。
- ・ NGDBuild の実行に指定した出力ディレクトリに書き込み権がない場合は、NGDBuild でエラーが発生します。

Tcl リファレンス

この章では、ザイリンクス Tcl コマンド言語に関する基本的な情報を示します。次のセクションが含まれています。

- ・ [Tcl の概要](#)
- ・ [Tcl の基礎](#)
- ・ [プロジェクト プロパティとプロセス プロパティ](#)
- ・ [一般用法のザイリンクス Tcl コマンド](#)
- ・ [アドバンス スクリプト用のザイリンクス Tcl コマンド](#)
- ・ [Tcl スクリプト例](#)

Tcl の概要

ツール コマンド言語 (Tcl) は使用が簡単なスクリプト言語で、EDA 業界でよく使用されています。

ザイリンクス Tcl コマンドは、ISE® グラフィカル ユーザー インターフェイス (GUI) を補足し、拡張するために設計されています。GUI を使用すると、プロジェクトの設定、初期インプリメンテーションの実行、さまざまなオプションの試行、制約の設定、デザインの可視化などを簡単に行うことができ、新規ユーザーの場合や新しいプロジェクトを作成する際に便利です。使用するオプションおよびインプリメンテーション手順が確定している場合は、Tcl コマンドのバッチ インターフェイスを使用して、同じスクリプトおよび操作を繰り返し実行できます。ザイリンクス Tcl コマンドの構文は、GUI とできる限り一致するように作成されており、GUI からスクリプトまたはバッチ モードへスムーズに移行できます。

Tcl のデバイス サポート

ザイリンクス Tcl コマンドは、次のデバイス ファミリで使用できます。

- ・ Spartan®-3、Spartan-3A、Spartan-3E、Spartan-6
- ・ Virtex®-4、Virtex-5、Virtex-6
- ・ CoolRunner™ XPLA3 および CoolRunner-II
- ・ XC9500 および XC9500XL

ザイリンクス Tcl シェル

コマンド ラインから xtclsh にアクセスするには、「**xtclsh**」と入力します。

```
> xtclsh
%
```


コマンド ライン構文は、Tcl コマンドおよびサブコマンドで構成されています。

```
% tcl_command subcommand optional_arguments
```

tcl_command : ザイリンクス Tcl コマンドです。

subcommand : ザイリンクス Tcl コマンドのサブコマンドです。

optional_arguments : サブコマンドの引数です。

ザイリンクス Tcl コマンド、サブコマンド、および引数の構文例は、この章の「[一般用法のザイリンクス Tcl コマンド](#)」および「[アドバンス スクリプト用のザイリンクス Tcl コマンド](#)」を参照してください。

ザイリンクス Tcl コマンドのヘルプの表示

help コマンドを使用すると、ザイリンクス Tcl コマンドの説明を表示できます。xtclsh プロンプト (%) で「**help**」と入力すると、ザイリンクス Tcl コマンドとその簡単な説明がリストされます。特定の Tcl コマンドに関する情報を表示するには、次のように入力します。

```
% help <tcl_command>
```

Tcl コマンドの後にサブコマンドを入力すると、このサブコマンドに関する情報が表示されます。たとえば、次のように入力すると、新規 ISE プロジェクトを作成するコマンドの詳細が表示されます。

```
% help project new
```

help : Tcl の情報を表示するコマンドです。

project : ザイリンクス Tcl コマンドです。

new : project のサブコマンドです。

メモ : Tcl の **help** コマンドでは、大文字と小文字が区別されます。xtclsh または [Tcl Console] パネルに小文字の「**help**」ではなく大文字で「**HELP**」と入力すると、OS で使用可能なコマンドが表示されます。

Tcl の基礎

各 Tcl コマンドは複数の単語で構成されており、最初の単語がコマンド名です。ザイリンクス Tcl コマンドでは、コマンド名は名詞 (project など) または動詞 (search など) です。コマンド名が名詞の場合、2 番目の単語は動詞になります (project open など)。2 番目の単語は、サブコマンドと呼ばれます。

その後の単語は、コマンドの追加パラメータです。ザイリンクス Tcl コマンドでは、必須のパラメータはサブコマンドの後に決まった順序で入力する必要があります。必須のパラメータの後には、追加パラメータを入力できます。追加パラメータはマイナス記号 (-) で開始しており (**-instance <instance-name>** など)、入力順は問いません。

Tcl では、大文字と小文字が区別されます。ザイリンクス Tcl コマンドは、すべて小文字です。コマンド名が 2 つの単語で構成されている場合、アンダースコア () で接続されます。Tcl では大文字と小文字が区別されますが、ほとんどのデザイン データ (インスタンス名など)、プロパティ名、プロパティ値では区別されません。コマンド プロンプトでの入力を簡略化するため、サブコマンドを入力する際に固有の接頭辞も認識されるので、コマンドの最初の数文字を入力するだけで済みます。

この Tcl リファレンスを活用するため、標準 Tcl コマンドについて理解しておくことが有益です。

- ・ **set** : 変数およびプロパティに値を割り当てます。set コマンドには 2 つの引数があり、1 つ目の引数で変数名、2 つ目の引数でその値を指定します。Tcl 変数にはデータ型はないので、事前に宣言する必要はありません。

```
% set fruit apple; # fruit という変数に apple という値を割り当てる
```
- ・ **\$ (ドル記号)** : 変数を値に置換します。たとえば、先ほどの fruit という変数を考えた場合、\$ を使用するかしらないかで次のような違いがあります。

```
% puts fruit; # fruit という単語を出力
```

```
% puts $fruit; # fruit という変数の値である apple を出力
```
- ・ **[] (角かっこ)** : 1 つのコマンドの結果を別のコマンドに直接代入できます。角かっこ ([]) を使用すると、コマンドをネストできます。角かっこで囲まれた部分が解釈され、その結果が代入されます。
- ・ **その他の代入** : Tcl では、スペースや特殊文字を含む文字列の引用、代入の制御に複数の方法があります。ダブル クォーテーション (") を使用すると、一部の特殊文字 ([] および \$) を代入に使用できます。波かっこ ({ }) では、代入は実行されません。
- ・ **バックスラッシュ (\)** : バックスラッシュは、Tcl では特別な意味があります。そのため、バックスラッシュを含む DOS 形式のパス名を Tcl コマンドに挿入しても、予測どおりに認識されません。Tcl コマンドおよびスクリプトでパス名を指定する場合は、スラッシュ (/) を使用してください。

Tcl は、ネストしたコマンドおよびスクリプト記述で使用する、その真価が発揮されます。コマンドの結果は変数として保存でき、変数 (または角かっこを使用して代入したコマンド結果) は別のコマンドの入力としてネストできます。

Tcl に関する一般的な情報は、Tcl Developer Xchange の Web サイト (<http://www.tcl.tk/doc/>) などから Tcl のマニュアルを参照してください。標準 Tcl コマンドを使用して記述されたスクリプトの例は、この章の「Tcl スクリプト例」で「標準 Tcl スクリプト例」を参照してください。Tcl Developer Xchange のサイトからも、チュートリアルと例を参照できます (<http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html>)。

ザイリンクス名前空間

ザイリンクス Tcl コマンドは、Tcl 名前空間 ::xilinx:: の一部です。別の Tcl パッケージでザイリンクス特有の Tcl コマンド名と同じコマンド名が使用されている場合、ザイリンクス Tcl コマンドを指定するにはザイリンクス名前空間を使用する必要があります。たとえば、次のように入力すると、ザイリンクス Tcl コマンドを使用して新規 ISE プロジェクトが作成されます。

```
% xilinx::project new <project_name>
```

ザイリンクス名前空間を指定する必要があるのは、複数の名前空間がインストールされている場合のみです。

プロジェクト プロパティとプロセス プロパティ

このセクションでは、Tcl コマンド のオプションとして使用可能なプロジェクト プロパティとプロセス プロパティを示します。

最初の表ではプロジェクトに適用されるプロジェクト プロパティをリストし、残りの表ではプロセス プロパティ (サポートされるバッチ ツール オプション) をソフトウェア プロセス別にリストします。

メモ：ここにリストされているプロパティには、ほかのプロパティに依存しており、関連するプロパティを設定しないと設定できないものがあります。このようなプロパティを設定しようとして設定が不可能な場合は、ほかのプロパティに依存していることを示す警告メッセージが表示されます。

プロジェクト プロパティ

プロジェクト プロパティ

プロパティ名	説明
family	デザインをインプリメントするデバイス ファミリ。
device	プロジェクトで使用するデバイス (指定したデバイス ファミリに含まれるもの)。
package	プロジェクトで使用するパッケージ (指定したデバイスに使用可能なもの)。
speed	デバイスのスピード グレード。
"Top-Level Source Type" または top_level_module_type	デザインの最上位モジュールのソース タイプ (HDL、EDIF、Schematic、または NGC/NGO)。
"Synthesis Tool"	デザインの合成に ISE® Design Suite で使用する合成ツール。デフォルトのツールは XST です。インストールされているサードパーティの合成ツールも使用可能です。
Simulator	ISE Design Suite に統合されたシミュレータ (ISim または ModelSim XE)、または ISE Design Suite で生成されたシミュレーション ネットリストとファイルを使用可能な外部シミュレータを指定します。
"Preferred Language"	シミュレーション ネットリストおよびその他の中間ファイルを生成する際に ISE Design Suite で使用する HDL 言語。合成ツールおよびシミュレータでサポートされている言語が 1 つのみの場合は、その言語がデフォルトになります。
Top	デザイン階層で最上位モジュールであるソース ファイルを指定します。
name	プロジェクト名。
"Use SmartGuide"	SmartGuide™ 機能をイネーブル (TRUE) またはディスエーブル (FALSE) にします。
"SmartGuide Filename"	最後に生成された配置配線済み NCD ファイル (デフォルト) 以外のファイルをガイド ファイルとして指定する場合に使用します。配置配線済み NCD ファイルを指定する必要があります。このプロパティは、"Use SmartGuide" プロパティを TRUE に設定しないと使用できません。

プロセス プロパティ - [Synthesize - XST] プロセス

次に、**-process "Synthesize - XST"** オプションを指定した場合に **project set** および **project get** コマンドで使用可能な XST プロセス プロパティを示します。

[Synthesize - XST] プロセス プロパティ

メモ： この表に示す値は、Virtex-5 デバイスをターゲットとした場合の値です。ほかのデバイスでは、値が異なる場合があります。

メモ： [コマンドライン オプション] 列は、シェル コマンドライン構文を示すものではなく、このガイドのほかのセクションで該当するオプションを参照するためにリストしています。

プロパティ名	データ型	値	デフォルト値	対応する XST コマンドライン オプション
"Add I/O Buffers"	ブール型	TRUE、FALSE	TRUE	-iobuf
"Automatic BRAM Packing"	ブール型	TRUE、FALSE	FALSE	-auto_bram_packing
"BRAM Utilization Ratio"	範囲	1 ~ 100	100	-bram_utilization_ratio
"Bus Delimiter"	リスト	<>、□、∅、0	<>	-bus_delimiter
"Case Implementation Style"	リスト	None、Full、Parallel、Full-Parallel	None	-vlgcase
"Case"	リスト	Maintain、Lower、Upper	Maintain	-case
"Cores Search Directories"	ファイル名			-sd
"Cross Clock Analysis"	ブール型	TRUE、FALSE	FALSE	-cross_clock_analysis
"Custom Compile File List"	ファイル名			-hdl_compilation_order
"Decoder Extraction"	ブール型	TRUE、FALSE	TRUE	-decoder_extract
"DSP Utilization Ratio"	範囲	1 ~ 100	100	-dsp_utilization_ratio
"Equivalent Register Removal"	ブール型	TRUE、FALSE	TRUE	-equivalent_register_removal
"FSM Encoding Algorithm"	リスト	Auto、One-Hot、Compact、Sequential、Gray、Johnson、User、Speed1、None	Auto	-fsm_extract -fsm_encoding
"FSM Style"	リスト	LUT、Bram	LUT 数	-fsm_style
"Generate RTL Schematic"	リスト	Yes、No、Only	Yes	-rtlview
"Generics, Parameters"	文字列			-generics
"Global Optimization Goal"	リスト	AllClockNets、Inpad To Outpad、Offset In Before、Offset Out After、Maximum Delay	AllClockNets	-glob_opt
"HDL INI File"	ファイル名			-xsthdpini
"Hierarchy Separator"	リスト	/、_	/	-hierarchy_separator

プロパティ名	データ型	値	デフォルト値	対応する XST コマンドライン オプション
"Keep Hierarchy"	リスト	No、Yes、Soft	No	-keep_hierarchy
"Library Search Order"	ファイル名	.iso ファイル		-lso
"Logical Shifter Extraction"	ブール型	TRUE、FALSE	TRUE	-shift_extract
"LUT Combining"	リスト	No、Auto、Area	No	-lc
"LUT-FF Pairs Utilization Ratio" (Virtex-5)	範囲	-1 ~ 100	100	-slice_utilization_ratio
"Max Fanout"	範囲	0 ~ 10000+	100000 (Virtex-5)	-max_fanout
"Move First Flip-Flop Stage"	ブール型	TRUE、FALSE	ほかのプロパティに依存	-move_first_stage
"Move Last Flip-Flop Stage"	ブール型	TRUE、FALSE	ほかのプロパティに依存	-move_last_stage
"Mux Extraction"	リスト	Yes、No、Force	Yes	-mux_extract
"Mux Style"	リスト	Auto、MUXF、MUXCY	Auto	-mux_style
"Netlist Hierarchy"	リスト	As Optimized、Rebuilt	As Optimized	-netlist_hierarchy
"Number of Clock Buffers" (Virtex-4 以外)	範囲	0 - 32	32	-bufg
"Number of Global Clock Buffers" (Virtex-4)	範囲	0 ~ 32	32	-bufg
"Number of Regional Clock Buffers" (Virtex-4)	範囲	0 ~ 16	16	-bufr
"Optimization Effort"	リスト	Normal、High	Normal	-opt_level
"Optimization Goal"	リスト	Speed、Area	Speed	-opt_mode
"Optimize Instantiated Primitives"	ブール型	TRUE、FALSE	FALSE	-optimize_primitives
"Other XST Command Line Options"	テキスト文字列	ほかのプロパティで設定されていない有効なコマンドライン オプション	なし	なし
"Pack I/O Registers into IOBs"	リスト	Auto、Yes、No	Auto	-iob
"Power Reduction"	ブール型	TRUE、FALSE	FALSE	-power

プロパティ名	データ型	値	デフォルト値	対応する XST コマンドライン オプション
"Priority Encoder Extraction"	リスト	Yes、No、Force	Yes	-priority_extract
"RAM Extraction"	ブール型	TRUE、FALSE	TRUE	-ram_extract
"RAM Style"	リスト	Auto、Distributed、Block (デバイスによって異なる)	Auto	-ram_style
"Read Cores"	リスト	Yes、No、Optimize	Yes	-read_cores
"Reduce Control Sets"	リスト	No、Auto	No	-reduce_control_sets
"Register Balancing"	リスト	No、Yes、Forward、Backward	No	-register_balancing
"Register Duplication"	ブール型	TRUE、FALSE	TRUE	-register_duplication
"Resource Sharing"	ブール型	TRUE、FALSE	TRUE	-resource_sharing
"ROM Extraction"	ブール型	TRUE、FALSE	TRUE	-rom_extract
"ROM Style"	リスト	Auto、Distributed、Block	Auto	-rom_style
"Safe Implementation"	リスト	No、Yes	No	-safe_implementation
"Shift Register Extraction"	ブール型	TRUE、FALSE	TRUE	-shreg_extract
"Slice Packing"	ブール型	TRUE、FALSE	TRUE	-slice_packing
"Slice Utilization Ratio" (Virtex-5 以外)	範囲	-1 ~ 100	100	-slice_utilization_ratio
"Synthesis Constraints File"	ファイル名			-uc
"Use Clock Enable"	リスト	Auto、Yes、No	Auto	-use_clock_enable
"Use DSP Block"	リスト	Auto、Yes、No	Auto	-use_dsp48
"Use Synchronous Reset"	リスト	Auto、Yes、No	Auto	-use_sync_reset
"Use Synchronous Set"	リスト	Auto、Yes、No	Auto	-use_sync_set
"Use Synthesis Constraints File"	ブール型	TRUE、FALSE	TRUE	-iuc
"Verilog 2001"	ブール型	TRUE、FALSE	TRUE	-verilog2001
"Verilog Include Directories"	ファイル名			-vlgincdir
"Verilog Macros"	テキスト文字列			-define で使用
"Work Directory"	ファイル名		./xst	-xsthdpdir

プロパティ名	データ型	値	デフォルト値	対応する XST コマンドライン オプション
"Write Timing Constraints"	ブール型	TRUE、FALSE	FALSE	-write_timing_constraints
"XOR Collapsing"	ブール型	TRUE、FALSE	TRUE	-xor_collapse

プロセス プロパティ - [Translate] プロセス

次に、**-process Translate** オプションを指定した場合に **project set** および **project get** コマンドで使用可能な変換 (NGDBuild) プロセス プロパティを示します。

[Translate] プロセス プロパティ

メモ：この表に示す値は、Virtex-5 デバイスをターゲットとした場合の値です。ほかのデバイスでは、値が異なる場合があります。

メモ：[コマンドライン オプション] 列は、シェル コマンドライン構文を示すものではなく、このガイドのほかのセクションで該当するオプションを参照するためにリストしています。

プロパティ名	データ型	値	デフォルト値	対応する NGDBuild コマンドライン オプション
"Allow Unexpanded Blocks"	ブール型	TRUE、FALSE	FALSE	-u
"Allow Unmatched LOC Constraints"	ブール型	TRUE、FALSE	FALSE	-aul
"Create I/O Pads from Ports"	ブール型	TRUE、FALSE	FALSE	-a
"Macro Search Path"	ファイル名	パイプ文字 () で区切ったファイル名		-sd
"Netlist Translation Type"	リスト	Timestamp、On、Off	Timestamp	-nt
"Other NGDBuild Command Line Options"	テキスト文字列	ほかのプロパティで設定されていない有効なコマンドライン オプション	なし	なし
"Preserve Hierarchy on Sub Module"	ブール型	TRUE、FALSE	FALSE	-insert_keep_hierarchy
"Use LOC Constraints"	ブール型	TRUE、FALSE	TRUE	-r が FALSE に対応
"User Rules File for Netlister Launcher"	ファイル名	--	--	-ur

プロセス プロパティ - [Map] プロセス

次に、**-process Map** オプションを指定した場合に **project set** および **project get** コマンドで使用可能なマップ プロセス プロパティを示します。

[Map] プロセス プロパティ

メモ：この表に示す値は、Virtex®-5 デバイスをターゲットとした場合の値です。ほかのデバイスでは、値が異なる場合があります。

メモ：[コマンドライン オプション] 列は、シェル コマンドライン構文を示すものではなく、このガイドのほかのセクションで該当するオプションを参照するためにリストしています。

プロパティ名	データ型	値	デフォルト値	対応する MAP コマンドライン オプション
"Allow Logic Optimization Across Hierarchy"	ブール型	TRUE、FALSE	FALSE	-ignore_keep_hierarchy
"CLB Pack Factor Percentage"	範囲	1 ~ 100	100	-c
"Combinatorial Logic Optimization"	ブール型	TRUE、FALSE	FALSE	-logic_opt
"Disable Register Ordering"	ブール型	TRUE、FALSE	FALSE	-r
"Equivalent Register Removal"	ブール型	TRUE、FALSE	FALSE	-equivalent_register_removal
"Extra Effort" (ほかのプロパティに依存)	リスト	None、Normal、Continue on Impossible	None	-xe
"Generate Detailed MAP Report"	ブール型	TRUE、FALSE	FALSE	-detail
"Global Optimization"	ブール型	TRUE、FALSE	FALSE	-global_opt
"Ignore User Timing Constraints" (「Timing Mode」も参照)	ブール型	TRUE、FALSE	FALSE	-ntd -x (Virtex-5 デバイス)
"LUT Combining"	リスト	Off、Auto、Area	off	-lc
"Map Effort Level" (ほかのプロパティに依存)	リスト	Standard、High	Standard	-ol
"Map Slice Logic into Unused Block RAMs"	ブール型	TRUE、FALSE	FALSE	-bp
"Map to Input Functions"	リスト	4、5、6、7、8	6	-k
"Maximum Compression"	ブール型	TRUE、FALSE	FALSE	-c

プロパティ名	データ型	値	デフォルト値	対応する MAP コマンドライン オプション
"Optimization Strategy (Cover Mode)"	リスト	Area、Speed、Balanced、Off	Area	-cm
"Other Map Command Line Options"	テキスト文字列	ほかのプロパティで設定されていない有効なコマンドライン オプション	なし	なし
"Pack I/O Registers/Latches into IOBs"	リスト	For Inputs and Outputs、For Inputs Only、For Outputs Only、Off	Off	-pr
"Perform Timing-Driven Packing and Placement"	ブール型	TRUE、FALSE	FALSE	-timing
"Placer Effort Level"	リスト	Standard、High	Standard	-ol
"Placer Extra Effort" (ほかのプロパティに依存)	リスト	None、Normal、Continue on Impossible	None	-xe
"Register Duplication"	ブール型	TRUE、FALSE	FALSE	-register_duplication
"Replicate Logic to Allow Logic Level Reduction"	ブール型	TRUE、FALSE	TRUE	-l
"Retiming"	ブール型	TRUE、FALSE	FALSE	-retiming
"Starting Placer Cost Table (1-100)"	範囲	1 ~ 100	1	-t
"Timing Mode" ("Ignore User Timing Constraints" に依存)	リスト			「 -ntd 」および「 -x 」を参照
"Trim Unconnected Signals"	ブール型	TRUE、FALSE	TRUE	-u
"Use RLOC Constraints"	ブール型	TRUE、FALSE	TRUE	-ir
"Use Timing Constraints"	ブール型	TRUE、FALSE	TRUE	-x

プロセス プロパティ – [Place & Route] プロセス

次に、**-process "Place & Route"** オプションを指定した場合に **project set** および **project get** コマンドで使用可能な配置配線 (PAR) プロセス プロパティを示します。

[Place & Route] プロセス プロパティ

メモ： この表に示す値は、Virtex®-4 デバイスをターゲットとした場合の値です。ほかのデバイスでは、値が異なる場合があります。

メモ： [コマンドライン オプション] 列は、シェル コマンドライン構文を示すものではなく、このガイドのほかのセクションで該当するオプションを参照するためにリストしています。

プロパティ名	データ型	値	デフォルト値	対応する PAR コマンドライン オプション
"Extra Effort (Highest PAR level only)" ("Place & Route Effort Level (Overall)" が High に設定されている場合のみ)	リスト	None、Normal、"Continue on Impossible"	None	-xe
"Generate Asynchronous Delay Report"	ブール型	TRUE、FALSE	FALSE	-delay (ReportGen)
"Generate Clock Region Report"	ブール型	TRUE、FALSE	FALSE	-clock_regions (ReportGen)
"Generate Post-Place & Route Simulation Model"	ブール型	TRUE、FALSE	FALSE	NetGen
"Generate Post-Place & Route Static Timing Report"	ブール型	TRUE、FALSE	TRUE	TRACE
"Ignore User Timing Constraints" (「Timing Mode」も参照)	ブール型	TRUE、FALSE	FALSE	-ntd -x (Virtex-5 デバイス)
"Other Place & Route Command Line Options"	テキスト文字列	ほかのプロパティで設定されていない有効なコマンドライン オプション	なし	なし
"Place & Route Effort Level (Overall)"	リスト	Standard、High	Standard	-ol
"Place and Route Mode" (デバイスによって値が異なる)	リスト	Normal Place and Route、Place Only、Route Only、Reentrant Route	"Normal Place and Route"	設定によって、対応するオプション (-r 、 -p 、 -k) が異なる。 デバイスによって異なる。

プロパティ名	データ型	値	デフォルト値	対応する PAR コマンド ライン オプション
"Placer Effort Level (Overrides Overall Level)"	リスト	None、Standard、High	None	-pl (Spartan®-3、Spartan-3A、Spartan-3E、Virtex®-4 デバイスのみ)
"Power Activity File"	ファイル名			-activityfile
"Power Reduction"	ブール型	TRUE、FALSE	FALSE	-power
"Router Effort Level (Overrides Overall Level)"	リスト	None、Standard、High	None	-rl
"Starting Placer Cost Table (1-100)"	範囲	1 ~ 100	1	-t (Spartan®-3、Spartan-3A、Spartan-3E、Virtex®-4 デバイスのみ)
"Timing Mode" ("Ignore User Timing Constraints" に依存)	リスト			「 -ntd 」および「 -x 」を参照
"Use Bonded I/Os"	ブール型	TRUE、FALSE	FALSE	-ub
"Use Timing Constraints"	ブール型	TRUE、FALSE	TRUE	-x

プロセス プロパティ – [Generate Programming File] プロセス

次に、**-process "Generate Programming File"** オプションを指定した場合に **project set** および **project get** コマンドで使用可能なプログラム ファイル生成 (BitGen) プロセス プロパティを示します。

[Generate Programming File] プロセス プロパティ

メモ： このプロセスのプロパティは、デバイスによって異なります。スペースが限られているため、ここではプロパティ名およびそのプロパティが適用される一部のデバイス ファミリと、1 つのデバイス (該当する場合は Virtex®-5) に対する値を示します。この表には、デバイス特定の情報は掲載されていません。詳細は、「BitGen」の章の「[BitGen のオプション](#)」で該当するオプションに関する説明を参照してください。

メモ： [コマンドライン オプション] 列は、シェル コマンドライン構文を示すものではなく、このガイドのほかのセクションで該当するオプションを参照するためにリストしています。

プロパティ名	データ型	値	デフォルト値	対応する BitGen コマンド ライン オプション
"AES Key (Hex String)"	文字列		なし	-g Key0
"Allow SelectMAP Pins to Persist"	ブール型	TRUE、FALSE	FALSE	-g Persist
"BPI Reads Per Page"	リスト	1、4、8	1	-g BPI_page_size

プロパティ名	データ型	値	デフォルト値	対応する BitGen コマンド ライン オプション
"Configuration Clk (Configuration Pins)"	リスト	"Pull Up"、Float	"Pull Up"	-g CclkPin
"Configuration Pin Busy"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g BusyPin
"Configuration Pin CS"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g CsPin
"Configuration Pin DIn"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g DinPin
"Configuration Pin Done"	リスト	"Pull Up"、Float	"Pull Up"	-g DonePin
"Configuration Pin Init"	リスト	"Pull Up"、Float	"Pull Up"	-g InitPin
"Configuration Pin M0"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g M0Pin
"Configuration Pin M1"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g M1Pin
"Configuration Pin M2"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g M2Pin
"Configuration Pin Powerdown"	リスト	"Pull Up"、Float	"Pull Up"	-g PowerdownPin
"Configuration Pin Program"	リスト	"Pull Up"、Float	"Pull Up"	-g ProgPin
"Configuration Pin RdWr"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g RdWrPin
"Configuration Rate"	リスト	2、6、9、13、17、20、24、27、31、35、38、42、46、49、53、56、60	2	-g ConfigRate
"Create ASCII Configuration File"	ブール型	TRUE、FALSE	FALSE	-b
"Create Binary Configuration File"	ブール型	TRUE、FALSE	FALSE	-g Binary
"Create Bit File"	ブール型	TRUE、FALSE	TRUE	-j
"Create IEEE 1532 Configuration File"	ブール型	TRUE、FALSE	FALSE	-g IEEE1532
"Create Mask File"	ブール型	TRUE、FALSE	FALSE	-m
"Create ReadBack Data Files"	ブール型	TRUE、FALSE	FALSE	-g Readback
"JTAG to System Monitor Connection"	ブール型	Enable、Disable	Enable	-g JTAG_SysMon

プロパティ名	データ型	値	デフォルト値	対応する BitGen コマンド ライン オプション
"Cycles for First BPI Page Read"	リスト	1、2、3、4	1	-g BPI_1st_read_cycle
"DCI Update Mode"	リスト	"As Required"、Continuous、Quiet(Off)	"As Required"	-g DCIUpdateMode
"Done (Output Events)"	リスト	"Default (4)"、1、2、3、4、5、6	"Default (4)"	-g DONE_cycle
"Drive Awake Pin During Suspend / Wake Sequence"	ブール型	TRUE、FALSE	FALSE	-g Drive_aware
"Drive Done Pin High"	ブール型	TRUE、FALSE	FALSE	-g DriveDone
"Enable BitStream Compression"	ブール型	TRUE、FALSE	FALSE	-g Compress
"Enable Cyclic Redundancy Checking (CRC)"	ブール型	TRUE、FALSE	TRUE	-g CRC
"Enable Debugging of Serial Mode BitStream"	ブール型	TRUE、FALSE	FALSE	-g DebugBitstream
"Enable Filter on Suspend Input"	ブール型	TRUE、FALSE	TRUE	-g Suspend_filter
"Enable Internal Done Pipe"	ブール型	TRUE、FALSE	FALSE	-g DonePipe
"Enable Outputs (Output Events)"	リスト	"Default (5)"、1、2、3、4、5、6、Done、Keep	"Default (5)"	-g DONE_cycle
"Enable Power-On Reset Detection"	ブール型	TRUE、FALSE	TRUE	-g en_porb
"Enable Suspend/Wake Global Set/Reset"	ブール型	TRUE、FALSE	FALSE	-g en_sw_gsr
"Encrypt Bitstream"	ブール型	TRUE、FALSE	FALSE	-g Encrypt
"Fallback Reconfiguration"	リスト	Enable、Disable	Enable	-g ConfigFallback
"FPGA Start-Up Clock"	リスト	CCLK、"User Clock"、"JTAG Clock"	CCLK	-g StartupClk
"GTS Cycle During Suspend / Wakeup Sequence"	範囲	1 - 1024	4	-g sw_gts_cycle
"GWE Cycle During Suspend / Wakeup Sequence"	範囲	1 - 1024	5	-g sw_gwe_cycle

プロパティ名	データ型	値	デフォルト値	対応する BitGen コマンド ライン オプション
"HMAC Key (Hex String)"	文字列		なし	-g HKey
"JTAG Pin TCK"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g TckPin
"JTAG Pin TDI"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g TdiPin
"JTAG Pin TDO"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g TdoPin
"JTAG Pin TMS"	リスト	"Pull Up"、Float、"Pull Down"	"Pull Up"	-g TmsPin
"Match Cycle"	リスト	Auto、0、1、2、3、4、5、6、NoWait	Auto	-g Match_cycle
"Other BitGen Command Line Options"	テキスト文字列	ほかのプロパティで設定されていない有効なコマンドライン オプション	なし	なし
"Power Down Device if Over Safe Temperature"	ブール型	TRUE、FALSE	FALSE	-g OverTempPowerDown
"Release DLL (Output Events)"	リスト	"Default (NoWait)"、0、1、2、3、4、5、6、"NoWait"	"Default (NoWait)"	-g LCK_cycle
"Release Write Enable (Output Events)"	リスト	"Default (6)"、1、2、3、4、5、6、Done、Keep	"Default (6)"	-g GWE_cycle
"Reset DCM if SHUTDOWN & AGHIGH performed"	ブール型	TRUE、FALSE	FALSE	-g DCMSshutdown
"Retry Configuration if CRC Error Occurs"	ブール型	TRUE、FALSE	ほかのプロパティに依存	-g Reset_on_err
"Run Design Rules Checker (DRC)"	ブール型	TRUE、FALSE	TRUE	-d
"Security"	リスト	"Enable Readback and Reconfiguration"、"Disable Readback"、"Disable Readback and Reconfiguration"	"Enable Readback and Reconfiguration"	-g Security
"SelectMAP Abort Sequence"	リスト	Enable、Disable	Enable	-g SelectMAPAbort
"Starting CBC Value (Hex)"	文字列	16 進文字列	ランダムに選択	-g StartCBC
"Starting Key"	リスト	None、0、3	None	-g StartKey
"Unused IOB Pins"	リスト	"Pull Down"、Float、"Pull Up"	"Pull Down"	-g UnusedPin

プロパティ名	データ型	値	デフォルト値	対応する BitGen コマンド ライン オプション
"UserID Code (8 Digit Hexadecimal)"	文字列	8 桁の 16 進数	0xFFFFFFFF	-g UserID
"Wakeup Clock"	リスト	"Startup Clock"、"Internal Clock"	"Startup Clock"	-g Sw_clk

プロセス プロパティ – [Generate Post-Place & Route Simulation Model] プロセス

次に、**-process "Generate Post-Place & Route Simulation Model"** オプションを指定した場合に **project set** および **project get** コマンドで使用可能な配置配線後のシミュレーション モデル (NetGen) プロセス プロパティを示します。

[Generate Post-Place & Route Simulation Model] プロセス プロパティ

メモ： この表に示す値は、Virtex®-5 デバイスをターゲットとした場合の値です。ほかのデバイスでは、値が異なる場合があります。

メモ： [コマンドライン オプション] 列は、シェル コマンドライン構文を示すものではなく、このガイドのほかのセクションで該当するオプションを参照するためにリストしています。

プロパティ名	データ型	値	デフォルト値	対応する NetGen コマンド ライン オプション
"Automatically Insert glbl Module in the Netlist"	ブール型	TRUE、FALSE	TRUE	-insert_glbl
"Bring Out Global Set/Reset Net as a Port"	ブール型	TRUE、FALSE	FALSE	-gp
"Bring Out Global Tristate Net as a Port"	ブール型	TRUE、FALSE	FALSE	-tp
"Device Speed Grade/Select ABS Minimum"	リスト	-3、-2、-1、"Absolute Min"	-3	-s
"Generate Architecture Only (No Entity Declaration)"	ブール型	TRUE、FALSE	FALSE	-a
"Generate Multiple Hierarchical Netlist Files"	ブール型	TRUE、FALSE	FALSE	-insert_glbl
"Global Set/Reset Port Name"	文字列		GSR_PORT	-gp で使用
"Global Tristate Port Name"	文字列		GTS_PORT	-tp で使用

プロパティ名	データ型	値	デフォルト値	対応する NetGen コマンドライン オプション
"Include sdf_annotate task in Verilog File"	ブール型	TRUE、FALSE	TRUE	-sdf_anno
"Other NetGen Command Line Options"	テキスト文字 列	ほかのプロパティで設定され ていない有効なコマンドライ ン オプション	なし	なし
"Output Extended Identifiers"	ブール型	TRUE、FALSE	FALSE	-extid
"Retain Hierarchy"	ブール型	TRUE、FALSE	TRUE	-fn

一般用法のザイリンクス Tcl コマンド

各コマンドに対し、説明、構文、例、および戻り値を示します。これらの例のほとんどでは、**project new** コマンドを使用してプロジェクトが作成されているか、**project open** コマンドを使用してプロジェクトが開いていることを前提としています。プロジェクト ファイルを追加するには、**xfile add** コマンドを使用します。

ザイリンクス Tcl コマンドを実際にどのように使用するかは、この章の最後にある「[Tcl スクリプト例](#)」の例を参照してください。

次の表に、一般用法のザイリンクス Tcl コマンドを示します。

コマンド	サブコマンド
lib_vhdl (VHDL ライブラリの制御)	add_file get delete new properties
process (プロジェクト プロセスの実行と制御)	get properties run set
project (プロジェクトの作成と制御)	archive clean close get get_processes new open properties set
xfile (ISE ソース ファイルの制御)	add get properties remove set

lib_vhdl (VHDL ライブラリの制御)

lib_vhdl コマンドは、ISE® プロジェクトの VHDL ライブラリを制御します。

VHDL ライブラリの作成および削除、VHDL ライブラリへのファイルの追加、VHDL ライブラリ情報の取得を実行できます。

構文

```
% lib_vhdl subcommand
```


使用可能なサブコマンドは、次のとおりです。

- ・ `add_file` (ソース ファイルのライブラリへの追加)
- ・ `delete` (ライブラリの削除)
- ・ `get` (ライブラリのプロパティ値の表示)
- ・ `new` (新規ライブラリの作成)
- ・ `properties` (ライブラリ プロパティのリスト表示)

追加情報

サブコマンドの詳細は、次のように入力してください。

```
% help lib_vhdl subcommand
```

lib_vhdl add_file (ソース ファイルのライブラリへの追加)

現在の ISE® プロジェクトに含まれるソース ファイルを現在のプロジェクトの既存のライブラリに追加します。

構文

```
% lib_vhdl add_file library_name file_name
```

lib_vhdl : ザイリンクス Tcl コマンドです。

add_file : サブコマンドです。

library_name : VHDL ライブラリの名前を指定します。

file_name : プロジェクト ソース ファイルの名前を指定します。

例

```
% lib_vhdl add_file mylib top.vhd
```

ソース ファイル `top.vhd` を `mylib` ライブラリに追加します。

戻り値

ファイルが正しく追加された場合は `true`、されなかった場合はエラー メッセージを表示。

追加情報

```
% help lib_vhdl
```

lib_vhdl delete (ライブラリの削除)

現在の ISE® プロジェクトからライブラリを削除します。

構文

```
% lib_vhdl delete library_name
```

lib_vhdl : ザイリンクス Tcl コマンドです。

delete : サブコマンドです。

library_name : 削除するライブラリの名前を指定します。

例

```
% lib_vhdl delete mylib
```

現在のプロジェクトから mylib ライブラリを削除します。

戻り値

ライブラリが正しく削除された場合は true、されなかった場合はエラー メッセージを表示。

追加情報

```
% help lib_vhdl
```

lib_vhdl get (ライブラリのプロパティ値の表示)

指定したライブラリ プロパティの値を表示します。

すべてのライブラリ プロパティのリストを表示するには、[lib_vhdl properties](#) (ライブラリ プロパティのリスト表示) を使用します。

構文

```
% lib_vhdl get library_name property_name
```

lib_vhdl : ザイリンクス Tcl コマンドです。

get : サブコマンドです。

library_name : ライブラリの名前を指定します。

property_name : ライブラリ プロパティの名前を指定します。有効なプロパティ名は name と files です。

例 1

```
% lib_vhdl get mylib name
```

mylib ライブラリの名前を表示します。

例 2

```
% lib_vhdl get mylib files
```

mylib ライブラリに含まれるファイルのリストを表示します。

戻り値

プロパティ値。正しく取得できなかった場合はエラー メッセージを表示。

追加情報

```
% help lib_vhdl
```

lib_vhdl new (新規ライブラリの作成)

現在の ISE® プロジェクトに新規ライブラリを作成します。

構文

```
% lib_vhdl new library_name
```

lib_vhdl : ザイリンクス Tcl コマンドです。

new : サブコマンドです。

library_name : 作成するライブラリの名前を指定します。

例

```
% lib_vhdl new mylib
```

mylib という VHDL ライブラリを作成し、現在のプロジェクトに追加します。

戻り値

ファイルが正しく作成された場合は true、されなかった場合はエラー メッセージを表示。

追加情報

```
% help lib_vhdl
```

lib_vhdl properties (ライブラリ プロパティのリスト表示)

ライブラリ プロパティをリスト表示します。

特定のライブラリ プロパティの値を表示するには、[lib_vhdl get](#) (ライブラリのプロパティ値の表示) を使用します。

構文

```
% lib_vhdl properties
```

lib_vhdl : ザイリンクス Tcl コマンドです。

properties : サブコマンドです。

例

```
% lib_vhdl properties
```

ライブラリ プロパティをリスト表示します。

戻り値

ライブラリ プロパティのリスト。正しく取得できなかった場合はエラー メッセージを表示。

追加情報

```
% help lib_vhdl
```

process (プロジェクト プロセスの実行と制御)

現在の ISE® プロジェクトに対するプロセスを実行および制御します。

構文

```
% process subcommand
```

使用可能なサブコマンドは、次のとおりです。

- ・ [get](#) (プロセスの指定したプロパティの値を表示)
- ・ [properties](#) (プロセス プロパティのリスト表示)
- ・ [run](#) (プロセス タスクの実行)
- ・ [set](#) (プロセスの指定したプロパティの値を設定)

追加情報

サブコマンドの詳細は、次のように入力してください。

```
% help process subcommand
```

process get (プロセスの指定したプロパティの値を表示)

指定したプロセス タスクのステータスを表示します。

メモ : 選択したソース ファイルによって、実行可能なプロセスは異なります。**project get_processes** コマンドを使用すると、実行可能なプロセスのリストを表示できます。このコマンドの使用法は、「**help project get_processes**」と入力すると表示されます。

構文

```
% process get process_task property_name
```

process : ザイリンクス Tcl コマンドです。

get : サブコマンドです。

process_task : プロパティ値を表示するプロセス タスクの名前を指定します。Project Navigator の [Design] パネルの [Processes] ペインに表示されるプロセス タスク名を指定する必要があります。選択したソース ファイルによって、実行可能なプロセスは異なります。**project get_processes** コマンドを使用すると、実行可能なプロセスのリストを表示できます。このコマンドの使用法は、「**help project get_processes**」と入力すると表示されます。

property_name : プロパティ名を指定します。指定可能なプロパティ名は、status および name です。

例 1

```
% process get "Map" status
```

[Map] プロセスの現在のステータスを表示します。

例 2

```
% process get "place" name
```

文字列「place」で開始するプロセスの完全な名前を表示します。この例では、「Place & Route」が返されます。

戻り値

指定したプロパティの値 (テキスト文字列)。

追加情報

```
% help process
```

process properties (プロセス プロパティのリスト表示)

プロセス プロパティをリスト表示します。次の 2 つのプロパティがサポートされています。

- ・ name : ISE® のプロセス名を表示するのに使用します。
- ・ status : プロセスのステータス情報を管理するのに使用します。

構文

```
% process properties
```

process : ザイリンクス Tcl コマンドです。

properties : サブコマンドです。

例

```
% process properties
```

プロセス プロパティのリストを表示します。

戻り値

プロパティ プロパティのリスト。

追加情報

```
% help process
```

process run (プロセス タスクの実行)

現在の ISE® プロジェクトに対して指定したプロセス タスクを実行します。

メモ : 選択したソース ファイルによって、実行可能なプロセスは異なります。 **project get processes** コマンドを使用すると、実行可能なプロセスのリストを表示できます。このコマンドの使用法は、「**help project get processes**」と入力すると表示されます。

構文

```
% process run process_task [-instance instance_name ]  
[-force { rerun | rerun_all }]
```

process : ザイリンクス Tcl コマンドです。

run : サブコマンドです。

process_task : 実行する プロセス タスクを指定します。Project Navigator の [Design] パネルの [Processes] ペインに表示されるプロセス タスク名を指定する必要があります。

-instance : プロセスの検索を指定したインスタンスに制限します。

instance_name : プロセスの検索を制限するインスタンスの名前を指定します。デフォルトは、最上位インスタンスです。

-force : プロセスの現在のステートにかかわらず、指定したプロセスを再実行します。

rerun : 指定したプロセスと、そのプロセスに関連するプロセスでアップデートが必要なものを再実行します。

rerun_all : 指定したプロセスとそのプロセスに関連するすべてのプロセスを再実行します。アップデートされているかどうかにかかわらず、すべてのプロセスが実行されます。

例 1

```
% process run "Translate"
```

[Translate] プロセスを実行します。

例 2

```
% process run "Implement Design" -force rerun_all
```

ソース ファイルがアップデートされているかどうかにかかわらず、デザイン全体をすべて再インプリメントします。

戻り値

プロセスが正常に実行された場合は true、されなかった場合は false。

追加情報

```
% help process
```

process set (プロセスの指定したプロパティの値を設定)

指定したプロセスのプロパティに値を設定します。

メモ : 選択したソース ファイルによって、実行可能なプロセスは異なります。 **project get_processes** コマンドを使用すると、実行可能なプロセスのリストを表示できます。このコマンドの使用法は、「**help project get_processes**」と入力すると表示されます。

構文

```
% process set process_task property_name property_value
```

process : ザイリンクス Tcl コマンドです。

set : サブコマンドです。

process_task : プロパティ値を設定するプロセス タスクの名前を指定します。Project Navigator の [Design] パネルの [Processes] ペインに表示されるプロセス タスク名を指定する必要があります。

property_name : プロパティ名を指定します。現在のところ、サポートされているプロパティは *status* のみです。

property_value : プロパティ値を指定します。指定可能なプロパティ値は *up_to_date* です。

例

```
% process set "Map" status up_to_date
```

[Map] プロセスのステータスを *up_to_date* (アップデート済み) に変更します。[Map] プロセスのステータスが何かの理由でアップデート必要になっている場合に、このコマンドで強制的にアップデート済みに変更できます。このコマンドを実行すると、ISE® Project Navigator でプロセス名の横に緑色のチェックマークが表示されます。

戻り値

設定したプロパティの値 (テキスト文字列)。

追加情報

```
% help process
```

project (プロジェクトの作成と制御)

ISE® プロジェクトを作成および制御します。プロジェクトには、デザインに関連するすべてのファイルおよびデータが含まれます。プロジェクトを使用して、デザイン ファイルを管理し、異なるプロセスを実行します。

構文

```
% project subcommand
```

使用可能なサブコマンドは、次のとおりです。

- ・ `archive` (ISE プロジェクトのアーカイブを作成)
- ・ `clean` (システムで生成されたプロジェクト ファイルの削除)
- ・ `close` (ISE プロジェクトを閉じる)
- ・ `get` (プロジェクト プロパティ値の表示)
- ・ `get_processes` (プロジェクトのプロセスの表示)
- ・ `new` (新規 ISE プロジェクトの作成)
- ・ `open` (ISE プロジェクトを開く)
- ・ `properties` (プロジェクト プロパティのリスト表示)
- ・ `set` (プロジェクト プロパティ、値、オプションの設定)
 - － `set device` (デバイスの設定)
 - － `set family` (デバイス ファミリの設定)
 - － `set package` (デバイス パッケージの設定)
 - － `set speed` (デバイス スピードの設定)
 - － `set top` (最上位モジュール/エンティティの設定)

追加情報

サブコマンドの詳細は、次のように入力してください。

```
% help project subcommand
```

project archive (ISE プロジェクトのアーカイブを作成)

テンポラリ ファイル、システムにより生成されたファイル、HDL ソース ファイルなど、現在の ISE® プロジェクトに含まれるファイルのアーカイブを作成します。HDL ソース ファイルなどがリモート ディレクトリにあり、**xfile add -copy** コマンドでプロジェクト ディレクトリにコピーされていない場合は、アーカイブを復元したときにこれらのファイルは元のディレクトリに自動的にコピーされないため、手動でコピーする必要があります。

構文

```
% project archive archive_name
```

project : ザイリンクス Tcl コマンドです。

archive : サブコマンドです。

archive_name : アーカイブの名前を指定します。通常、アーカイブ ファイルには拡張子 `.zip` が使用されます。拡張子を指定しない場合は、`.zip` が使用されます。

注意 : 指定したアーカイブ名のファイルが既に存在する場合は、そのファイルが上書きされます。

例

```
% project archive myarchive.zip
```

現在のプロジェクトに含まれるすべてのファイルをアーカイブします。アーカイブ ファイル名は `myarchive.zip` です。

戻り値

正常にアーカイブされた場合は true、正常にアーカイブされなかった場合は false。

追加情報

% help project

project clean (システムで生成されたプロジェクト ファイルの削除)

現在の ISE® プロジェクトにある一時ファイルおよびシステムで生成されたファイルをすべて削除します。Verilog または VHDL などのソース ファイルや、ユーザーが変更したファイルは削除されません。たとえば、NCD (.ncd) やマップ レポート (.mpr) などのシステムで生成されたファイルやレポートファイルは、ユーザーにより変更されていない限り削除されます。

構文

% project clean

project : ザイリンクス Tcl コマンドです。

clean : サブコマンドです。

注意 : project clean コマンドでは、システムで生成されたすべてのファイルが現在のプロジェクトから恒久的に削除されます。これらのファイルには、NGD、NGA、NCD など、インプリメンテーション ツールで生成されるファイルも含まれます。

例

% project clean

現在のプロジェクトをクリーンアップします。一時ファイルおよびシステムで生成されたファイルが削除されます。

戻り値

クリーンアップが正常に実行された場合は true、されなかった場合は false。

追加情報

% help project

project close (ISE プロジェクトを閉じる)

現在開いている ISE® プロジェクトを閉じます。

構文

% project close

project : ザイリンクス Tcl コマンドです。

close : サブコマンドです。

例

% project close

現在のプロジェクトを閉じます。

戻り値

プロジェクトが正常に閉じた場合は true、閉じなかった場合は false。

追加情報

% help project

project get (プロジェクト プロパティ値の表示)

指定したプロジェクトのプロパティまたはバッチ アプリケーション オプションの値を表示します。

構文

```
% project get {option_name/property_name} [-process  
process_name] [-instance instance_name]
```

project : ザイリンクス Tcl コマンドです。

get : サブコマンドです。

option_name : 値を表示するバッチ アプリケーション オプションの名前を指定します ("Map Effort Level" など)。バッチ アプリケーション オプションは、ダブル クォーテーション (") で囲んで指定します。Project Navigator に表示されるオプション (プロパティ) 名全体またはその一部を入力できます。オプション名の一部を入力すると、完全なオプション名が検索され、1 つのオプション名を特定できない場合は、エラー メッセージが表示されます。

property_name : 値を表示するプロパティの名前を指定します。有効なプロパティ名は、family、device、generated_simulation_language、package、speed、および top です。

-process : *option_name* の検索を指定のプロセスに制限します。デフォルトでは、すべての合成およびインプリメンテーション プロセスのプロパティが検索されます。all を指定すると、すべてのプロジェクト プロセスのプロパティが表示されます。

process_name : *option_name* の値を取得するプロセスの名前を指定します。

-instance : *option_name* の検索を指定した *instance_name* に制限します。

instance_name : *option_name* を検索するインスタンスの名前を指定します。あるインスタンスに関連するプロセスのみに検索を制限する場合に指定します。インスタンスの完全な階層名を指定する必要があります。デフォルトは、最上位インスタンスです。

例

```
% project get speed
```

project set speed コマンドで設定したスピード グレードの値を表示します。

戻り値

プロパティ値 (テキスト文字列)。

追加情報

% help project

project get_processes (プロジェクトのプロセスの表示)

指定したインスタンスで実行可能なプロセスの名前をリスト表示します。

構文

```
% project get_processes [-instance instance_name]
```

project : ザイリンクス Tcl コマンドです。

get_processes : サブコマンドです。

-instance : 指定したインスタンスに関するプロパティのみを表示します。インスタンスを指定しない場合は、最上位インスタンスが指定されます。

instance_name : プロセスを表示するインスタンスの名前を指定します。

例

```
% project get_processes -instance /stopwatch/Inst_dcm1
```

/stopwatch/Inst_dcm1 インスタンスに対して実行可能なすべてのプロセスをリストします。

戻り値

実行可能なプロセス (Tcl リスト)。

追加情報

```
% help project
```

project new (新規 ISE プロジェクトの作成)

新規 ISE® プロジェクトを作成します。

構文

```
% project new project_name
```

project : ザイリンクス Tcl コマンドです。

new : サブコマンドです。

project_name : 作成するプロジェクトの名前を指定します。拡張子を指定しない場合は、.ise が使用されます。

例

```
% project new watchver.ise
```

watchver.ise という新規プロジェクトを作成します。

戻り値

新規プロジェクト名。

追加情報

```
% help project
```

project open (ISE プロジェクトを開く)

既存の ISE® プロジェクトを開きます。プロジェクトが存在しない場合は、エラー メッセージが表示され、project new コマンドで新規プロジェクトを作成する必要があることが示されます。既にプロジェクトが開いている場合は、現在開いているプロジェクトが閉じ、指定したプロジェクトが開きます。

構文

```
% project open project_name
```

project : ザイリンクス Tcl コマンドです。

open : サブコマンドです。

project_name : 開くプロジェクトの名前を指定します。拡張子を指定しない場合は、.ise が使用されます。

例

```
% project open watchver.ise
```

現在のディレクトリにある watchver.ise プロジェクトを開きます。

戻り値

開いたプロジェクトの名前。

追加情報

```
% help project
```

project properties (プロジェクト プロパティのリスト表示)

指定したプロセスまたはインスタンスのプロパティをすべてリスト表示します。

構文

```
% project properties [-process process_name] [-instance instance_name]
```

project : ザイリンクス Tcl コマンドです。

properties : サブコマンドです。

-process process_name : 指定したプロセスに関するプロパティのみを表示します。デフォルトでは、すべての合成およびインプリメンテーション プロセスのプロパティが検索されます。all を指定すると、すべてのプロジェクト プロセスのプロパティが表示されます。

-instance instance_name : 指定したインスタンスに関するプロパティのみを表示します。インスタンス名を指定しない場合は、最上位インスタンスのプロパティが表示されます。top を指定すると、最上位インスタンスが指定されます。top 以外を指定する場合は、インスタンスの完全な階層名を指定する必要があります。

メモ : 特定のインスタンスのプロセスを表示するには、**project get_processes** コマンドを使用してください。デバイス ファミリ、デバイス、スピード グレードなど特定のプロパティの値を表示するには、**project get** コマンドを使用してください。

例

```
% project properties -process all
```

現在のプロジェクトに対して実行可能なすべてのプロセスのプロパティをリストします。

戻り値

プロパティのリスト。

追加情報

```
% help project
```

project set (プロジェクト プロパティ、値、オプションの設定)

現在の ISE® プロジェクトに対してプロパティとその値を設定します。指定できるプロパティは、device、generated_simulation_language、family、package、speed、synthesis_tool、および top_level_module_type です。

ファミリおよびデバイス特定のプロパティに加え、XST、NGDBuild、MAP、PAR、TRACE、BitGen などのバッチ アプリケーション ツールのオプションも設定できます。set コマンドには、2 つの引数が必要です。1 つ目の引数はプロパティまたは変数の名前、2 つ目はその値を指定します。-process および -instance はオプションで、指定したプロセスまたはインスタンスでのみプロパティを設定します。

構文

```
% project set property_name property_value [-process  
process_name] [-instance instance_name]
```

project : ザイリンクス Tcl コマンドです。

set : サブコマンドです。

property_name : プロパティ、変数、またはバッチ アプリケーション オプションの名前を指定します。

property_value : プロパティ、変数、またはバッチ アプリケーション オプションの値を指定します。

-process *process_name* : 指定したプロセスに関するプロパティのみを設定します。デフォルトでは、すべての合成およびインプリメンテーション プロセスのプロパティが検索されます。

-process all を指定すると、すべてのプロセスのプロパティが設定されます。

-instance *instance_name* : 指定したインスタンスに関するプロパティのみを設定します。インスタンス名を指定しない場合は、最上位インスタンスのプロパティが表示されます。**top** を指定すると、最上位インスタンスが指定されます。インスタンスの完全な階層名を指定する必要があります。

メモ : バッチ アプリケーション オプションには、ほかのオプションを指定しないと機能しないものもあります。たとえば、XST の [Synthesize Constraints File] オプションは、[Use Synthesis Constraints File] オプションを指定しないと機能しません。バッチ アプリケーション オプションは、ダブルクォーテーション (") で囲んで指定します。Project Navigator に表示されるオプション (プロパティ) 名全体またはその一部を入力できます。オプション名の一部を入力すると、完全なオプション名が検索され、1 つのオプション名を特定できない場合は、エラーメッセージが表示されます。

メモ : VHDL ソースの場合は、アーキテクチャ名およびエンティティ名を使用して最上位ソースを設定します。次の例を参照してください。

例 1

```
% project set top /stopwatch/sixty
```

/stopwatch/sixty というインスタンスを最上位ソースに設定します。

例 2

```
% project set top inside cnt60
```

アーキテクチャ名が inside、entity 名が cnt60 のインスタンスを最上位ソースに設定します。

例 3

```
% project set "Map Effort Level" High
```

[Map Effort Level] (マップのエフォートレベル) を High に設定します。

戻り値

新しく設定したオプションの値。

追加情報

```
% help project
```

xfile (ISE ソース ファイルの制御)

ISE® プロジェクト内のソース ファイルを制御するために使用します。xfile コマンドを使用すると、現在のプロジェクトに含まれるソース ファイルを追加または削除したり、ファイルに関する情報を表示したりできます。

構文

```
% xfile subcommand
```

使用可能なサブコマンドは、次のとおりです。

- add (プロジェクトへのファイルの追加)
- get (プロジェクト ファイルのプロパティ値の表示)
- properties (ファイル プロパティのリスト表示)
- remove (プロジェクトからのファイルの削除)
- set (ファイルの指定したプロパティの値を設定)

追加情報

サブコマンドの詳細は、次のように入力してください。

```
% help xfile subcommand
```

xfile add (プロジェクトへのファイルの追加)

現在の ISE® プロジェクトに指定のファイルを追加します。**-copy** を使用した場合、ファイルを現在のプロジェクト ディレクトリにコピーしてからプロジェクトに追加します。ファイルは特定の順序で追加する必要はありません。ワイルドカードも使用できます。

このコマンドは、VHDL ライブラリに 1 つ以上のファイルを追加するために使用できます。

デフォルトでは、ファイルはすべてのデザイン フェーズに関連付けられます。**-view** オプションを使用すると、特定のデザイン フェーズに関連付けることができます。

構文

```
% xfile add file_name [-copy] [-lib_vhdl library_name] [-view view_type] [-include_global]
```

xfile : ザイリンクス Tcl コマンドです。

add : サブコマンドです。

file_name : プロジェクトに追加するソース ファイルの名前を指定します。ワイルドカードを使用して、複数のファイルを指定できます。Tcl コマンドでは、* および ? などのワイルドカードがサポートされています。ワイルドカードの詳細は、Tcl のマニュアルを参照してください。

-copy : ファイルを現在のプロジェクトにコピーします (オプション)。

-lib_vhdl : ファイルを既存の VHDL ライブラリに追加します。

library_name : VHDL ライブラリの名前を指定します。

-view : ソース ファイルに **view_type** で指定したデザイン フェーズに関連付けます。

view_type : 関連付けるデザイン フェーズを指定します。指定可能な値は、“All”、“Implementation”、“Simulation”、および “None” です。

-include_global : プロジェクトにソースを追加するたびにコンパイル順シーケンス ID をインクリメントします。

例 1

```
% xfile add alu.vhd processor.vhd alu.ucf
```

現在のプロジェクトに、alu.vhd、processor.vhd、および alu.ucf ファイルを追加します。

例 2

```
% xfile add *.v
```

現在のプロジェクトに、現在のディレクトリに含まれるすべての Verilog ファイルを追加します。

例 3

```
% xfile add test.vhd -lib_vhdl mylib
```

test.vhd ソース ファイルを現在のプロジェクトに追加し、mylib ライブラリにも追加します。

例 4

```
% xfile add test_tb.vhd -view "Simulation"
```

現在のプロジェクトに test_tb.vhd ソース ファイルを追加し、シミュレーション フェーズにのみ関連付けます。

戻り値

ファイルが正常に追加された場合は true、追加されなかった場合は false。

追加情報

```
% help xfile
```

xfile get (プロジェクト ファイルのプロパティ値の表示)

指定したプロジェクト ファイルとそのプロパティに関する情報を表示します。このコマンドでは、name および timestamp の 2 つのプロパティがサポートされています。

構文

```
% xfile get file_name {name|timestamp|include_global}
```

xfile : ザイリンクス Tcl コマンドです。

get : サブコマンドです。

file_name : 名前またはタイムスタンプ情報を取得するソース ファイルの名前を指定します。

name : 指定したファイルの完全パスを表示します。

timestamp : **xfile add** コマンドを使用してファイルをプロジェクトに最初に追加したときのタイムスタンプを表示します。

include_global コンパイル順タグのステータスを表示します。ファイルがコンパイル順リストに含まれている場合は true、含まれていない場合は false を返します。

例

```
% xfile get stopwatch.vhd timestamp
```

stopwatch.vhd ファイルのタイムスタンプ情報を表示します。

戻り値

指定したプロパティの値 (テキスト文字列)。

追加情報

```
% help xfile
```

xfile properties (ファイル プロパティのリスト表示)

使用可能なすべてのファイル プロパティをリスト表示します。このコマンドでは、name および timestamp の 2 つのプロパティがサポートされています。

構文

```
% xfile properties
```

xfile : ザイリンクス Tcl コマンドです。

properties : サブコマンドです。

メモ : プロジェクトに含まれるファイルのリストを表示するには、search コマンドを使用してください。

例

```
% xfile properties
```

現在のプロジェクトに含まれるファイルで使用可能なプロパティを表示します。

戻り値

使用可能なファイル プロパティのリスト。

追加情報

詳細は、次のように入力してください。

- ・ **% help xfile**
- ・ **% help search**

xfile remove (プロジェクトからのファイルの削除)

ISE® プロジェクトから指定したファイルを削除します。

メモ : ファイルはプロジェクトから削除されるだけで、物理的な場所 (ディスク) からは削除されません。

構文

```
% xfile remove file_name
```

xfile : ザイリンクス Tcl コマンドです。

remove : サブコマンドです。

file_name : プロジェクトから削除するファイルの名前を指定します。ワイルドカードはサポートされていません。例 3 に示すように Tcl リストを使用してください。

例 1

```
% xfile remove stopwatch.vhd
```

現在のプロジェクトから stopwatch.vhd ファイルを削除します。

例 2

```
% xfile remove alu.vhd processor.vhd
```

現在のプロジェクトから alu.vhd および processor.vhd ファイルを削除します。

例 3

```
% xfile remove [ search *memory*.vhd -type file ]
```

現在のプロジェクトからファイル名に「memory」を含む VHDL ファイルをすべて削除します。

- ・ 角かっこ内のコマンドでは、ワイルドカードを使用して名前に「memory」を含むファイルの Tcl リストを作成します。
- ・ 作成した Tcl リストを使用して、プロジェクトからこれらのファイルを削除します。

例 4

```
% set file_list [ list alu.v processor.v ]
```

```
% xfile remove $file_list
```

現在のプロジェクトから alu.v および processor.v ファイルを削除します。

- ・ 1 行目で、alu.v および processor.v ファイルを含む file_list という Tcl リストを作成します。
- ・ 2 行目では、この Tcl リストに含まれるファイルをプロジェクトから削除します。

戻り値

ファイルが正常に削除された場合は true、されなかった場合は false。

追加情報

```
% help xfile
```

xfile set (ファイルの指定したプロパティの値を設定)

現在の ISE® プロジェクトに含まれる指定のファイルに対してプロパティ値を設定します。

現在のところ、サポートされているプロパティは lib_vhdl のみです。

構文

```
% xfile set file_name property_name property_value
```

xfile : ザイリンクス Tcl コマンドです。

set : サブコマンドです。

file_name : プロパティ値を設定するソースファイルの名前を指定します。

property_name : プロパティ名を指定します。

property_value : プロパティ値を指定します。

例 1

```
% xfile set stopwatch.vhd lib_vhdl mylib
```


stopwatch.vhd ファイルの lib_vhdl プロパティを mylib に設定します。

例 2

```
% xfile set stopwatch.vhd include_global true
```

コンパイル順リストに stopwatch.vhd ファイルを追加します。このファイルをリストから削除するには、**include_global false** を使用します。

戻り値

指定したプロパティの値 (テキスト文字列)。

追加情報

```
% help xfile
```

アドバンス スクリプト用のザイリンクス Tcl コマンド

アドバンス スクリプト用のザイリンクス Tcl コマンドでは、オブジェクトおよびコレクションが使用されます。オブジェクトには、インスタンス、ファイル、プロセスなど、ISE® プロジェクトのエレメントを指定できます。コレクションは、オブジェクトおよびコレクション変数に指定した値に基づくオブジェクトのグループです。

各コマンドに対し、説明、構文、例、および戻り値を示します。これらの例のほとんどでは、**project new** コマンドを使用してプロジェクトが作成されているか、**project open** コマンドを使用してプロジェクトが開いていることを前提としています。プロジェクト ファイルを追加するには、**xfile add** コマンドを使用します。

次の表に、アドバンス スクリプト用のザイリンクス Tcl コマンドを示します。

コマンド	サブコマンド
globals (ザイリンクス グローバル データの操作)	get properties set unset
collection (コレクションの作成と制御)	append_to copy equal foreach get index properties remove_from set sizeof
object (オブジェクト情報の表示)	get name properties type
search (デザイン オブジェクトの検索)	

globals (ザイリンクス グローバル データの操作)

ザイリンクス グローバル データを操作します。

構文

```
% globals subcommand
```

使用可能なサブコマンドは、次のとおりです。

- ・ **get** (グローバル プロパティ/データの表示)
- ・ **set** (グローバル プロパティ/データの設定)
- ・ **properties** (グローバル プロパティ/データのリスト表示)
- ・ **unset** (グローバル プロパティ/データの削除)

追加情報

サブコマンドの詳細は、次のように入力してください。

```
% help globals subcommand
```

globals get (グローバル プロパティ/データの表示)

指定したグローバル プロパティの値を表示します。

構文

```
% globals get property_name
```

globals : ザイリンクス Tcl コマンドです。

get : サブコマンドです。

property_name : グローバル プロパティ/データの名前を指定します。

例

```
% globals get display_type
```

グローバル プロパティ `display_type` の値を表示します。

戻り値

指定したプロパティの値。

追加情報

```
% help globals
```

globals properties (グローバル プロパティ/データのリスト表示)

使用可能なグローバル プロパティをリスト表示します。

構文

```
% globals properties
```

globals : ザイリンクス Tcl コマンドです。

properties : サブコマンドです。

例

```
% globals properties
```

使用可能なグローバル プロパティをリスト表示します。

戻り値

使用可能なグローバル プロパティのリスト。

追加情報

```
% help globals
```

globals set (グローバル プロパティ/データの設定)

指定したグローバル プロパティの値を設定します。指定したプロパティが存在しない場合は作成されます。

構文

```
% globals set property_name property_value
```

globals : ザイリンクス Tcl コマンドです。

set : サブコマンドです。

property_name : グローバル プロパティ/データの名前を指定します。

property_value : プロパティ値を指定します。

例

```
% globals set display_type 1
```

グローバル プロパティ `display_type` の値を 1 に設定します。

戻り値

指定したプロパティの値。

追加情報

```
% help globals
```

globals unset (グローバル プロパティ/データの削除)

指定したグローバル プロパティを削除します。

構文

```
% globals unset property_name
```

globals : ザイリンクス Tcl コマンドです。

unset : サブコマンドです。

property_name : グローバル プロパティ/データの名前を指定します。

例

```
% globals unset display_type
```

グローバル プロパティ `display_type` を削除します。

戻り値

指定したプロパティの値。

追加情報

```
% help globals
```

collection (コレクションの作成と制御)

コレクションは、Tcl オブジェクトのグループで、Tcl インターフェイスにエクスポートされます。
`collection` コマンドを使用すると、コレクションを作成したり、指定したコレクションに含まれるオブジェクトを制御できます。

コレクションは、Tcl ではコレクション変数として参照されます。コレクション変数は、**collection set** コマンドで定義します。コレクション変数の値は、コレクションです。

構文

```
% collection subcommand
```

使用可能なサブコマンドは、次のとおりです。

- ・ [append_to](#) (コレクションへのオブジェクトの追加)
- ・ [copy](#) (コレクションのコピー)
- ・ [equal](#) (コレクションの比較)
- ・ [foreach](#) (コレクション内の各オブジェクトで繰り返し)
- ・ [get](#) (コレクション プロパティ値の表示)
- ・ [index](#) (コレクションのオブジェクトの抽出)
- ・ [properties](#) (使用可能なコレクション プロパティのリスト表示)
- ・ [remove_from](#) (コレクションからのオブジェクトの削除)
- ・ [set](#) (コレクションのプロパティの設定)
- ・ [sizeof](#) (コレクションに含まれるオブジェクト数の表示)

追加情報

サブコマンドの詳細は、次のように入力してください。

```
% help collection subcommand
```

collection append_to (コレクションへのオブジェクトの追加)

コレクションにオブジェクトを追加します。このコマンドでは、コレクション変数で指定したコレクションに、search コマンドで検索されたオブジェクト、または別のコレクションからのオブジェクトを追加します。指定したコレクション変数が存在しない場合、コマンドが実行される際に作成されます。

構文

```
% collection append_to collection_variable objects_to_append  
[-unique]
```

collection : ザイリンクス Tcl コマンドです。

append_to : サブコマンドです。

collection_variable : コレクションを参照するコレクション変数を指定します。指定したコレクション変数が存在しない場合、コマンドが実行される際に作成されます。

objects_to_append : コレクションに追加するオブジェクトまたはオブジェクトのコレクションを指定します。

-unique : コレクションに含まれていないオブジェクトのみを追加します。このオプションを使用しない場合、オブジェクトが重複して追加される可能性があります。

例

```
% collection append_to colVar [search * -type instance]
```

colVar というコレクション変数を作成します。ネストされた search コマンドでは、現在のデザインに含まれるすべてのインスタンスが検索されます。これらのインスタンスがオブジェクトとしてコレクションに追加され、colVar コレクション変数で参照されます。

戻り値

オブジェクトのコレクション。

追加情報

- ・ `% help collection`
- ・ `% help object`
- ・ `% help search`

collection copy (コレクションのコピー)

既存のコレクションのコピーを作成します。コレクションの個別のコピーが 2 つ必要な場合にのみ使用するようにしてください。例 1 に、コレクションのコピーの作成方法を示します。

コレクションをコピーする代わりに、1 つのコレクションを参照するコレクション変数を複数作成することも可能です。通常は、コレクションを参照する変数をもう 1 つ作成するだけで十分です。変数が同じコレクションを参照していることを確認してください。例 2 に、2 つの変数で 1 つのコレクションを参照する方法を示します。

構文

collection copy *collection_variable*

collection : ザイリンクス Tcl コマンドです。

copy : サブコマンドです。

collection_variable : コピーするコレクションの名前を指定します。

例 1 : 別のコレクションの作成

```
% set colVar_2 [collection copy $colVar_1]
```

コレクション変数 colVar_2 を作成します。ネストされた collection copy コマンドで colVar_1 コレクションのコピーを作成し、colVar_2 コレクション変数に割り当てます。この例では、独立した別のコレクションが作成されます。

例 2 : 2 つの変数で 1 つのコレクションを参照

```
% set colVar_1 [search * -type instance]
```

```
% set colVar_2 $colVar_1
```

コレクション colVar_1 を参照するコレクション colVar_2 を作成します。

- ・ 1 行目でコレクションを作成し、コレクション変数 colVar_1 に割り当てます。
- ・ 2 行目で colVar_1 の値を参照する 2 つ目のコレクション変数 colVar_2 を作成します。

メモ : この場合、参照されているコレクションは 1 つです。colVar_1 または colVar_2 に加えた変更は、両方のコレクション変数に反映されます。

戻り値

新規コレクション。

追加情報

- ・ `% help collection`
- ・ `% help object`
- ・ `% help search`

collection equal (コレクションの比較)

2 つのコレクションの内容を比較します。同じオブジェクトが含まれている場合、コレクションは等しいと判断されます。両方のコレクションに含まれているオブジェクトが同じ場合は 1 が返され、異なる場合は 0 が返されます。デフォルトでは、オブジェクトの順序は考慮されません。**-order_dependent** オプションを指定すると、オブジェクトの順序も考慮されます。

構文

```
% collection equal colVar_1 colVar_2 [-order_dependent]
```

collection : ザイリンクス Tcl コマンドです。

equal : collection のサブコマンドです。

colVar_1 : 比較の基準となるコレクションを指定します。

colVar_2 : 基準コレクションと比較するコレクションを指定します。

-order_dependent : コレクションを比較する際に、オブジェクトの順序も考慮します (オプション)。

メモ : 2 つの空のコレクションを比較すると、等しいと判断され、1 が返されます。

例

```
% set colVar_1 [search * -type instance]
```

```
% set colVar_2 [search /top/T* -type instance]
```

```
% collection equal $colVar_1 $colVar_2
```

2 つのコレクションの内容を比較します。

- ・ 1 行目で、インスタンスのコレクションをコレクション変数 *colVar_1* に割り当てます。
- ・ 2 行目で、指定の文字列に一致するインスタンスのコレクションをコレクション変数 *colVar_2* に割り当てます。
- ・ 3 行目で、2 つのコレクションを比較します。コレクション変数の値は、ドル記号 (\$) を使用して参照します。この例では、*colVar_1* と *colVar_2* の値が等しいかどうかと比較されます。

戻り値

コレクションが等しい場合は 1、異なる場合は 0。

追加情報

- ・ `% help collection`
- ・ `% help object`
- ・ `% help search`

collection foreach (コレクション内の各オブジェクトで繰り返し)

コレクションの各オブジェクトに対して、反復子変数で指定した処理を繰り返します。反復子変数は、処理を実行するコレクションと、各オブジェクトに対して実行するコマンドまたはスクリプトを指定します。

構文

```
% collection foreach iterator_variable collection_variable  
{body}
```

collection : ザイリンクス Tcl コマンドです。

foreach : サブコマンドです。

iterator_variable : 反復子変数の名前を指定します。

collection_variable : 処理を実行するコレクションの名前を指定します。

body : 各繰り返しで実行するコマンドまたはスクリプトを指定します。

注意 : 標準の Tcl に含まれる foreach コマンドでは、コレクションに対して繰り返し処理を実行できません。ザイリンクス Tcl の collection foreach コマンドを使用してください。標準の Tcl の foreach コマンドを使用すると、コレクションが削除される可能性があります。

例

```
% set colVar [search * -type instance]
```

```
% collection foreach itr $colVar {puts [object name $itr]}
```

コレクションの各オブジェクトに対して繰り返し処理を実行します。

- ・ 1 行目で、インスタンスのコレクションをコレクション変数 colVar に割り当てます。
- ・ 2 行目で、colVar コレクション内の各オブジェクトに対して繰り返し処理を実行します。itr は、反復子変数の名前です。波かっこ {} で囲まれた部分は、各繰り返しで実行するスクリプトです。波かっこ内にネストされている object name コマンドは、反復子変数の値を返します。この例の場合はインスタンスです。

戻り値

スクリプトが実行された回数。

追加情報

- ・ `% help collection`
- ・ `% help object`
- ・ `% help search`

collection get (コレクション プロパティ値の表示)

指定したコレクション プロパティの値を表示します。コレクション プロパティとその値は、**collection set** コマンドで指定します。

構文

```
% collection get property_name
```

collection : ザイリンクス Tcl コマンドです。

get : サブコマンドです。

property_name : 値を表示するプロパティの名前を指定します。有効なプロパティは、**display_line_limit** および **display_type** です。

メモ : 「[collection set \(コレクションのプロパティの設定\)](#)」も参照してください。

例

```
% collection get display_type
```

display_type プロパティの値を表示します。

戻り値

指定したプロパティの値。

追加情報

- ・ `% help collection`
- ・ `% help object`
- ・ `% help search`

collection index (コレクションのオブジェクトの抽出)

コレクションから指定のインデックスにあるオブジェクトを抽出します。元のコレクションは変更されません。

インデックスの範囲は、0 ~ (コレクションのサイズ - 1) です。コレクションのサイズは、**collection sizeof** コマンドで確認できます。

構文

```
% collection index collection_variable index_value
```

collection : ザイリンクス Tcl コマンドです。

index : サブコマンドです。

collection_variable : オブジェクトを抽出するコレクションの名前を指定します。

index_value : インデックスを指定します。インデックスの範囲は、0 ~ (コレクションのサイズ - 1) です。コレクションのサイズは、**collection sizeof** コマンドで確認できます。

メモ : ザイリンクス Tcl コマンドでコレクションを作成する際、含まれるオブジェクトの順序は指定されませんが、オブジェクトは一定のルールに従って生成されるので、常に予測される順序になります。コレクションの並べ替えをサポートするアプリケーションでは、コレクションのオブジェクトを特定の順序に並べることができます。

例

```
% set colVar [search * -type instance]
```

```
% set item [collection index $colVar 2]
```

```
% object name $item
```

インスタンスのコレクションの 3 番目のオブジェクトを抽出します。

- ・ 1 行目では、colVar というコレクション変数を作成します。ネストされた **search** コマンドでは、現在のデザインに含まれるすべてのインスタンスが検索され、その結果がコレクションの値となります。
- ・ 2 行目では、item というコレクション変数を作成します。ネストされた **collection index** コマンドは、3 番目のオブジェクトを抽出します。
- ・ 3 行目は、item 変数の値 (インデックスで指定したオブジェクト) を返します。

戻り値

インデックスで指定したオブジェクト。

追加情報

- ・ `% help collection`
- ・ `% help object`
- ・ `% help search`

collection properties (使用可能なコレクション プロパティのリスト表示)

現在の ISE® プロジェクトに含まれるすべてのコレクションで使用可能なプロパティをリスト表示します。プロパティ値は、**collection set** コマンドで指定します。

構文

% collection properties

collection : ザイリンクス Tcl コマンドです。

properties : サブコマンドです。

コレクション プロパティには、display_line_limit と display_type があります。これらのプロパティは、**collection get** および **collection set** コマンドで使用できます。

メモ : 使用可能なプロパティのリストは、「[collection get \(コレクション プロパティ値の表示\)](#)」を参照してください。

例

% collection properties

使用可能なコレクション プロパティのリストを表示します。display_line_limit および display_type を返します。

戻り値

使用可能なコレクション プロパティのリスト。

追加情報

- ・ `% help collection`
- ・ `% help object`
- ・ `% help search`

collection remove_from (コレクションからのオブジェクトの削除)

指定したコレクションからオブジェクトを削除します。これにより、既存のコレクションが変更されます。既存のコレクションを変更せずに残す場合は、**collection copy** コマンドを使用してコピーを作成してからこのコマンドを使用してください。

構文

```
% collection remove_from collection_variable objects_to_remove
```

collection : ザイリンクス Tcl コマンドです。

remove_from : サブコマンドです。

collection_variable : コレクション変数の名前を指定します。

objects_to_remove : コレクションから削除するオブジェクトまたはオブジェクトのコレクションを指定します。

例

```
% set colVar_1 [search * -type instance]
```

```
% set colVar_2 [search /stopwatch/s* -type instance]
```

```
% set colVar_3 [collection remove_from colVar_1 $colVar_2]
```

colVar_1 から colVar_2 のオブジェクトを削除します。

- ・ 1 行目では、colVar_1 というコレクション変数を作成します。
- ・ 2 行目では、colVar_2 というコレクション変数を作成します。
- ・ 3 行目では、colVar_1 から colVar_2 に含まれるオブジェクトを削除し、コレクション変数 colVar_3 に割り当てます。

戻り値

オブジェクトが削除された元のコレクション。

追加情報

- ・ [% help collection](#)
- ・ [% help object](#)
- ・ [% help search](#)

collection set (コレクションのプロパティの設定)

現在の ISE® プロジェクトに含まれるすべてのコレクションに対して、指定したプロパティを設定します。

構文

```
% collection set property_name property_value
```

collection : ザイリンクス Tcl コマンドです。

set : サブコマンドです。

property_name : プロパティ名を指定します。

property_value : プロパティ値を指定します。

次の 2 つのプロパティを設定できます。

- ・ **display_line_limit** : コレクション変数で表示可能な行数を指定します。このプロパティは、多数のオブジェクトを含む大きなコレクションで有益です。デフォルト値は 100 です。設定可能な最小値は 0 で、上限はありません。
- ・ **display_type** : 指定したコレクション変数のオブジェクトの表示にオブジェクト タイプを含めます。有効な値は true または false です。デフォルトでは false に設定されており、オブジェクト タイプは表示されません。次の例を参照してください。

例

```
% collection set display_type true
```

プロジェクトに含まれるすべてのコレクション変数にプロパティを設定します。プロパティ名は display_type、プロパティ値は true となります。

戻り値

プロパティ値。

追加情報

- ・ [% help collection](#)
- ・ [% help object](#)
- ・ [% help search](#)

collection sizeof (コレクションに含まれるオブジェクト数の表示)

指定したコレクションに含まれるオブジェクトの数を表示します。

構文

```
% collection sizeof collection_variable
```

collection : ザイリンクス Tcl コマンドです。

sizeof : サブコマンドです。

collection_variable : コレクションの名前を指定します。

例

```
% collection sizeof $colVar
```

コレクション変数 colVar で参照されるコレクションのサイズを表示します。

戻り値

指定したコレクションに含まれるオブジェクトの数。

追加情報

- ・ [% help collection](#)
- ・ [% help object](#)
- ・ [% help search](#)

object (オブジェクト情報の表示)

現在の ISE® プロジェクトに含まれるオブジェクトの名前、タイプ、プロパティ情報を表示します。

1 つのオブジェクトまたはコレクションのオブジェクトを指定できます。

構文

```
% object subcommand
```

使用可能なサブコマンドは、次のとおりです。

- ・ `get` (オブジェクトのプロパティ値の表示)
- ・ `name` (オブジェクト名の表示)
- ・ `properties` (オブジェクト プロパティのリスト表示)
- ・ `type` (オブジェクト タイプの表示)

追加情報

サブコマンドの詳細は、次のように入力してください。

```
% help object subcommand
```

object get (オブジェクトのプロパティ値の表示)

指定したプロパティの値を表示します。

構文

```
% object get obj property_name
```

object : ザイリンクス Tcl コマンドです。

get : サブコマンドです。

obj : プロパティを表示するオブジェクトの名前を指定します。

property_name : 表示するプロパティを指定します。

プロパティは、オブジェクトのタイプによって異なります。**object properties** コマンドを使用すると、指定のオブジェクトのプロパティがリスト表示されます。

例

```
% set colVar [search * -type instance]
% collection foreach obj $colVar {
    set objProps [object properties $obj]
    foreach prop $objProps {
        puts [object get $obj $prop]
    }
}
```

1 行目では、すべてのインスタンスを含むコレクションを作成します。2 行目では、コレクションに含まれる各オブジェクトに対して繰り返し処理を実行します。各オブジェクトに対し、**object properties** コマンドを使用してオブジェクトに設定可能なプロパティのリストを取得し、各プロパティの値を返します。

戻り値

指定したプロパティの値。

追加情報

- ・ `% help object`
- ・ `% help collection`
- ・ `% help search`

object name (オブジェクト名の表示)

オブジェクトの名前を表示します。

構文

```
% object name obj
```

object : ザイリンクス Tcl コマンドです。

name : サブコマンドです。

obj : 名前を表示するオブジェクトを指定します。

例

```
% set colVar [search * -type instance]
```

```
% object name [collection index $colVar 1]
```

colVar コレクションの 2 番目のオブジェクトの名前を表示します。

- ・ 1 行目では、colVar というコレクション変数を作成します。ネストされた **search** コマンドで、現在のプロジェクトに含まれるすべてのインスタンスが検索され、その結果がコレクションの値となります。
- ・ 2 行目では、コレクションの 2 番目のオブジェクトの名前を取得します。**collection index** コマンドでは、オブジェクトの含まれるコレクションを \$colVar、オブジェクトのインデックスを 1 に指定しています。インデックスの値は 0 から開始するので、この例ではコレクションの 2 番目のオブジェクトの名前が表示されます。

メモ: 詳細は、「[collection index \(コレクションのオブジェクトの抽出\)](#)」を参照してください。

戻り値

オブジェクトの名前 (テキスト文字列)。

追加情報

- ・ `% help object`
- ・ `% help collection`
- ・ `% help search`

object properties (オブジェクト プロパティのリスト表示)

指定したオブジェクトで設定可能なプロパティを表示します。

構文

```
% object properties obj [-descriptors]
```

object : ザイリンクス Tcl コマンドです。

properties : サブコマンドです。

obj : プロパティを表示するオブジェクトの名前を指定します。

-descriptors : プロパティ ディスクリプタのコレクションを返します。このコレクションに対して反復処理を実行することにより、各プロパティの詳細情報を取得できます。このオプションを指定しない場合、プロパティ名のリストが返されます。

例 1

```
% set colVar [search * -type partition]
% collection foreach obj $colVar {
    set objProps [object properties $obj]
    foreach prop $objProps
        puts [object get $obj $prop]
    }
}
```

1 行目では、**search** コマンドで検索されたオブジェクトを含むコレクションを作成します。2 行目では、コレクションに含まれる各オブジェクトに対して繰り返し処理を実行します。各オブジェクトに対し、**object properties** コマンドを使用してオブジェクトに設定可能なプロパティのリストを取得し、各プロパティの値を返します。

例 2

```
% set colVar [search * -type partition]
% set partition [collection index $colVar 1]
% set propertyDescriptors [object properties $partition -descriptors]
% collection foreach propDescr $propertyDescriptors {
    puts "name           : [object get $propDescr name]"
    puts "type           : [object get $propDescr type]"
    puts "is_read_only     : [object get $propDescr is_read_only]"
    puts "allowable_values : [object get $propDescr allowable_values]"
    puts "default          : [object get $propDescr default]"
    puts "units            : [object get $propDescr units]"
    puts "drivers           : [object get $propDescr drivers]"
    puts "description        : [object get $propDescr description]"
}
```

プロパティ ディスクリプタのコレクションを返します。プロパティ ディスクリプタは、プロパティを説明するオブジェクトです。

これらのプロパティ ディスクリプタに対して反復処理を実行し、プロパティに関する情報を取得できます。

プロパティ ディスクリプタから、次の情報を取得できます。

- ・ プロパティ名 (name を指定)
- ・ プロパティのタイプ (type を指定)
- ・ プロパティが読み出し専用かどうか (is_read_only を指定)
- ・ 設定可能なプロパティ値 (allowable_values を指定)
- ・ プロパティのデフォルト値 (default を指定)
- ・ プロパティの単位仕様 (units を指定)
- ・ 指定のプロパティが依存するプロパティの名前 (drives を指定)
- ・ プロパティの説明 (description を指定)

戻り値

-descriptors オプションを使用した場合はプロパティ ディスクリプタのコレクション、使用しない場合はプロパティ名のリスト。

追加情報

- ・ `% help object`
- ・ `% help collection`
- ・ `% help search`

object type (オブジェクト タイプの表示)

オブジェクトのタイプを表示します。

構文

```
% object type obj
```

object : ザイリンクス Tcl コマンドです。

type : サブコマンドです。

obj: タイプを表示するオブジェクトを指定します。オブジェクト名は、常に Tcl 変数で指定します。Tcl 変数は、次の例に示すように、**set** コマンドを使用して作成します。

例

```
% set colVar [search * -type instance]
```

```
% object type [collection index $colVar 1]
```

colVar コレクションの 2 番目のオブジェクトのタイプを表示します。

- ・ 1 行目では、colVar というコレクション変数を作成します。ネストされた **search** コマンドで、現在のプロジェクトに含まれるすべてのインスタンスが検索され、その結果がコレクションの値となります。
- ・ 2 行目では、コレクションの 2 番目のオブジェクトの名前を取得します。**collection index** コマンドでは、オブジェクトの含まれるコレクションを \$colVar、オブジェクトのインデックスを 1 に指定しています。インデックスの値は 0 から開始するので、この例ではコレクションの 2 番目のオブジェクトのタイプが表示されます。

メモ: 詳細は、「[collection index \(コレクションのオブジェクトの抽出\)](#)」を参照してください。

戻り値

オブジェクトのタイプ (テキスト文字列)。

追加情報

- ・ `% help object`
- ・ `% help collection`
- ・ `% help search`

search (デザイン オブジェクトの検索)

指定したパターンに一致するオブジェクトを検索します。

構文

```
% search {pattern|expression} [[-matchcase] [-exactmatch]
[-regexp]] | [-exp] [-type object_type] [-in
{project|collection}]
```

search : ザイリンクス Tcl コマンドです。

pattern または *expression* : 検索文字列を指定します。**-exp** を使用した場合、ザイリンクス検索式構文を使用した検索条件 (*expression*) を指定できます。**-exp** を使用しない場合、オブジェクト名に一致する文字列パターン (*pattern*) を指定します。

-matchcase : **-exp** を使用しない場合にのみ有効です。指定した文字列と大文字/小文字の区別まで一致するオブジェクト名を検索します。

-exactmatch : **-exp** を使用しない場合にのみ有効です。指定した文字列と完全に一致するオブジェクト名を検索します。

-regexp : **-exp** を使用しない場合にのみ有効です。正規表現を指定する場合に使用します。デフォルトでは、検索文字列は単純な文字列として扱われ、「*_ccir_*」のようにワイルドカードを含めることもできます。

-exp : 検索条件を検索式構文を使用して指定します。検索式構文を使用すると、オブジェクトをプロパティで検索できます。

-type object_type : 検索するオブジェクトのタイプを指定します。プロジェクトが読み込まれている場合、有効なタイプは file、instance、および lib_vhdl です。デバイスが読み込まれている場合、有効なタイプは belsite、io_standard、site、および tile です。

-in {project|collection} : 検索範囲を指定します。**-in** または **-in project** を使用すると現在のプロジェクト内で検索が実行され、**-in collection** を使用すると指定のコレクション内で検索が実行されます。

例 1

```
% search "/stopwatch" -type instance
```

デザインに含まれるすべてのインスタンスから「/stopwatch」が含まれるものを検索します。

例 2

```
% search * -type file
```

プロジェクトに含まれるすべてのソース ファイルを検索します。

戻り値

検索条件に一致するオブジェクトのコレクション。一致するオブジェクトがない場合は、空のコレクションが返されます。

追加情報

search コマンドに関する詳細情報は、複数のセクションに分かれています。特定のセクションに関する詳細は、次のように入力してください。

```
% help search section
```

section には、次を指定できます。

- ・ examples : search コマンドの使用例
- ・ expression : 検索式の概要
- ・ operator : 検索表現でサポートされる演算子のリスト
- ・ function : 検索表現でサポートされる関数のリスト
- ・ approx : approx 関数の概要
- ・ contains : contains 関数の概要
- ・ exists : exists 関数の概要
- ・ glob : glob 関数の概要
- ・ property : property 関数の概要
- ・ quote : quote 関数の概要
- ・ regexp : regexp 関数の概要
- ・ size : size 関数の概要
- ・ type : type 関数の概要
- ・ contains_usage : contains 関数の詳細な使用法
- ・ glob_usage : glob 関数の詳細な使用法
- ・ regexp_usage : regexp 関数の詳細な使用法

Tcl スクリプト例

このセクションでは、次の Tcl スクリプト例を示します。

- ・ [標準 Tcl スクリプト例](#)
- ・ [一般用法の Tcl スクリプト例](#)
- ・ [その他のザイリンクス Tcl スクリプト例](#)

これらの Tcl スクリプト例は、次の方法で実行できます。

- ・ xtclsh プロンプト (%) に 1 文ずつ入力します。各コマンドを理解し、出力を監視する場合に適した方法です。
- ・ xtclsh プロンプト (%) にアクセスするには、コマンドラインで「**xtclsh**」と入力するか、Project Navigator の [Tcl Console] パネルを使用します。
- ・ 拡張子が .tcl のスクリプト ファイルに保存します。このスクリプト ファイルを実行するには、xtclsh プロンプト (%) に次のように入力します。

```
% source <script_name> .tcl
```

- ・ Tcl シェルのインスタンスの後にスクリプト名を入力して実行することにより、コマンドラインから直接スクリプトを実行することも可能です。

```
% xtclsh <script_name> .tcl
```

標準 Tcl スクリプト例

次の Tcl スクリプト例では、標準 Tcl 言語の基本的な機能を示します。これらの例は、基本的な Tcl スクリプト記述を習得中の初心者向けです。標準 Tcl を理解すると、ザイリンクス Tcl スクリプトを特定のデザイン用にカスタマイズするのに役立ちます。これらのスクリプト例は、標準 Tcl シェルまたはザイリンクス xtclsh から実行できます。

これらのスクリプト例には、プロシージャとして定義されているものがあります。プロシージャを定義すると、プロシージャ名を入力するだけでそのプロシージャを実行できるようになります。たとえば、最初のスクリプト例は **proc Factorial{n}** というプロシージャですが、これを Tcl に入力 (または **source** コマンドを使用してスクリプトを入力) すると、次のように名前を入力するだけでプロシージャを実行できるようになります。

% Factorial <number>; # <number> は関数への入力

最初のスクリプト例は、階乗を求める Factorial というプロシージャです。proc 文に {n} があることから、入力を 1 つ使用することがわかります。proc 文の最後の波かっこは、プロシージャを構成するコマンドの開始を示します。このプロシージャを最後まで見ると、最終結果は solution という変数であることがわかります。

このプロシージャでは、変数 multiplier を 1 から n まで増分しながらループを n 回実行します。ループが実行されるたびに、1 から n までの値が乗算されて solution の値が増加し、最終的に $1 * 2 * 3 * \dots * n$ が求められます。

```
proc Factorial{n} {  
  set solution 1;  
  set multiplier 1;  
  while {$multiplier <= $n} {  
    set solution [expr $multiplier * $solution];  
    set multiplier [expr $multiplier + 1];  
  }  
  return $solution;  
}
```

上記の関数を、次のように再帰的に記述することも可能です。

```
proc Factorial{n} {  
  if {$n <= 1}  
  {return 1;}  
  else { return [expr $n * [Factorial [expr $n - 1]]]  
  }  
}
```

次のスクリプト例は、2 つの引数を使用するプロシージャです。これは、ファイルの内容から特定のパターンに一致する文字列を検索する Linux のコマンドライン プログラム **grep** を簡単に表現したものです。1 つの引数は検索パターンを指定し、もう 1 つの引数は検索するファイルを指定します。

```
proc greppy {pattern fileexp} {# procedure with {arguments}
# use glob: to access filenames that match a pattern
foreach filename [glob $fileexp] {
  if {[file type $filename] eq "file"} {# file or directory?
    puts "---- Reading $filename ----"
# opens the filename and returns its file handle.
# You need the file handle to read from a file and/or write into it.
    set fh [open $filename]
# reads in the whole file into a variable!# Illustration of a benefit of Tcl's typeless variables
    set file_contents [read $fh]
    close $fh# close the file - by using its file handle.
# look for \n (end of line), and split up the lines based on
# the newlines. One line at a time is assigned to the variable "line"
    foreach line [split $file_contents \n] {
# evaluate regular expression, comparing the pattern you passed in on the command line to each line in the file.
      if [regexp $pattern $line] {
        puts $line
      }
    }
  }
}
```

次のスクリプト例は、絶対パス名からファイル名のみを取り出すプロシージャです。このスクリプトでは、Tcl の文字列操作関数のいくつかが使用されています。このプロシージャの記述方法は複数ありますが、ここでは次の文字列操作関数を使用した方法を示します。

[string last "/" \$fullfile]; # 文字列の最後のスラッシュの位置を取得

[string length \$fullfile]; # 文字列の長さを取得

[string range \$fullfile a b]; # 位置 a から b までの文字列を返す

fullfile の値として C:/designs/proj1/top.vhd を入力するとします。

プロシージャを絶対パス名を引数として使用して実行するには、次のように入力します。

% getfname C:/designs/proj1/top.vhd

この結果、ファイル名 top.vhd が返されます。

```
proc getfname {fullfile}{
  set start [expr [string last "/" $fullfile] + 1]
  set end [string length $fullfile]
  return [string range $fullfile $start $end]
}
```

中間の変数割り当てを省くことにより、これらの 3 つのコマンドを 1 つにまとめることもできます。

```
proc getfname {fullfile}{
  return [string range $fullfile \
[expr [string last "/" $fullfile] + 1] [string length $fullfile]]
}
```

一般用法の Tcl スクリプト例

次のスクリプト例は、一般的なデザイン プロセスを実行します。まず新規プロジェクトを作成し、デバイスおよびファミリなどのプロジェクトレベルのプロパティを指定します。その後、ソース デザイン ファイルを追加してインプリメンテーション ツールのオプションを制御するプロセス プロパティを設定し、インプリメンテーション プロセスを実行します。スクリプトのコメントに、どの操作を実行するかが説明されています。

```
# create and open the project and set project-level properties
project new watchvhd.xise
project set family spartan3e
project set device xc3s100e
project set package vq100
project set speed -5
# add all the source HDLs and ucf
xfile add stopwatch.vhd statmatch.vhd cnt60.vhd dcml.vhd decode.vhd smallcntr.vhd
xfile add tenths.vhd hex2led.vhd
xfile add watchvhd.ucf
# set batch application options :
# 1. set synthesis optimization goal to speed
# 2. ignore any LOCs in ngdbuild
# 3. perform timing-driven packing
# 4. use the highest par effort level
# 5. set the par extra effort level
# 6. pass "-instyle xflow" to the par command-line
# 7. generate a verbose report from trce
# 8. create the IEEE 1532 file during bitgen
project set "Optimization Goal" Speed
project set "Use LOC Constraints" false
project set "Place & Route Effort Level (Overall)" High
project set "Extra Effort (Highest PAR level only)" "Continue on Impossible"
project set "Report Type" "Verbose Report" -process "Generate Post-Place & Route Static Timing"
project set "Create IEEE 1532 Configuration File" TRUE
# run the entire xst-to-trce flow
process run "Implement Design"
# close project
project close
# open project again
project open watchvhd
# close project
project close
```

その他のザイリンクス Tcl スクリプト例

次に、ザイリンクス Tcl コマンドを使用した短い単純なスクリプト例を示します。ISE® プロジェクトを開いた状態で、ザイリンクス xtclsh からこれらのプロシージャを実行してください。

最初のスクリプトは、現在のデザイン プロパティのリストを画面またはファイルに出力するのに便利です。まず、プロパティをリストするプロセスの名前を含む Apps_list を作成します。次に、options.tcl というファイルを書き込み用に開き、1 つ目のループで Apps_list の各プロセスに対してプロパティリストを取得します。2 つ目のループでは、各プロパティの値を取得し、ファイルに書き込みます。ファイルを閉じたら、options.tcl をテキスト エディタで開くか、プロパティとその値をレポートとして印刷できます。

```
set Apps_list {"Synthesize - XST"\
"Translate"\
"Map"\
"Generate Post-Map Static Timing"\
"Generate Post-Map Simulation Model"\
"Place & Route"\
"Generate Post-Place & Route Static Timing"\
"Generate Post-Place & Route Simulation Model"\
"Back-Annotate Pin Locations"\
"Generate Programming File"
}
set fp [open "options.tcl" "w"]
foreach ISE_app $Apps_list {
puts $fp "# ***** Properties for < $ISE_app > *****"
foreach prop [project properties -process $ISE_app] {
set val [project get "$prop" -process "$ISE_app"]
if {$val != ""} {
puts $fp "project set \"$prop\" \"$val\" -process \"$ISE_app\""
}
}
}
close $fp
```

次のスクリプト例では、標準 Tcl コマンド **catch** を使用して、Tcl シェルでエラーが検出される前にエラーを見つける方法を示します。スクリプトの中間ステップにエラーが含まれていても、長いスクリプトを停止せずに実行する必要がある場合があります。このスクリプトでは、Tcl コマンド **time** を使用して、プロセスの経過時間も記録しています。

```
# Run XST, catch any errors, and record the runtime
if { [catch { time {process run "Synthesize - XST"} } synthTime ] } {
puts "Synthesis failed"
}
# or else, XST was successful. Write out the runtime.
else {
puts "Synthesis ran in $synthTime"
}
```

デザインをインプリメントする際、Tcl スクリプトに次のコマンドを追加すると便利な場合があります。

```
# Regenerate Core for a particular instance
process run "Regenerate Core" -instance myCore
# Set up properties to generate post place static timing report
project set "report type" "Verbose Report" \ process "Generate Post-Place & Route Static Timing"
# Set up properties to create the source control friendly version # of the bit file: the .bin file
# The .bin file has the same internals, but no header so a # simple diff works.
project set "Create Bit File" "true" project set "Create Binary Configuration File" "true"
```