制約ガイド

UG625 (v. 13.4) 2012 年 1 月 18 日

該当するソフトウェア バージョン : ISE Design Suite 13.4 ～ 14.6

ISE Design Suite タイミング制約に関する情報は、『タイミング クロージャ ユーザー ガイド』 (UG612) を参照してください。

XILINX®

本資料は英語版 (v.13.4) 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。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com までお知らせください。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。

改訂履歴

日付	バージョン	改訂内容
2011 年 3 月 1 日	13.1	VCCAUX_IO および MARK_DEBUG 制約を追加
2011 年 6 月 22 日	13.2	該当するサポート デバイスのリストに Spartan®-6 を追加。 PULLUP (プルアップ) 制約に、NGDBuild で次が無視されることを記載· DEFAULT KEEPER = FALSE · DEFAULT PULLUP = FALSE · DEFAULT PULLDOWN = FALSE IODELAY_GROUP (IODELAY グループ) 制約の「LOC を使用した場合の制限」と「アーキテクチャ サポート」に情報を追加 エリア グループ (AREA_GROUP) 制約に次の注記を追加IOB および BUF 以外のコンポーネントにはすべて CLOCKREGION 範囲の制約を付けることができます。 CONFIG_MODE (コンフィギュレーション モード) 制約に、新規アーキテクチャ サポートと値を追加

日付	バージョン	改訂内容
		BEL 制約から VHDL 例を削除
2011 年 10 月 19 日	13.3	「RISING および FALLING キーワードは、TNM 制約と一緒に使用することもできます。」を削除 Virtex®-6 デバイスの DIFF_TERM サポートを追加 INPUT_JITTER および SYSTEM_JITTER のデフォルト単位を ps から ns に変更 OFFSET 制約に定義済みグループを使用できないという情報を追加 Spartan-6 デバイスの POST_CRC INIT フラグをアップデート 新しい Vcco Sense Mode (VCCOSENSEMODE) 制約を追加
2012 年 1 月 18 日	13.4	· タイミング制約の項目を『タイミング クロージャー ユーザー ガイド』(UG612) へ移動 · 「TNM_NET は通常、特定ネットを指定するため HDL デザインで使用するプロパティです。」という文を削除 · FSM スタイル (FSM_STYLE) 制約へのリファレンスを削除。XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices (UG687) に記述されています。 · 「パターン一致の UCF 構文例 2」のタイムグループ (TIMEGRP) 制約の構文例を修正

目次

XILINX®

第 1 章

制約のタイプ

この章では、さまざまな制約のタイプについて説明します。

注記： タイミング クロージャーを達成するためのタイミング制約の使用については、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

属性および制約

属性および制約は、同じ意味で使用される場合と、別の意味で使用される場合があります。また、構文で属性と指示子が使用される場合も、意味は類似していますが異なります。ザイリンクスで使用される「属性」と「制約」の定義は、次のとおりです。

属性

属性は、通常インスタンシエートされるコンポーネントのファンクションおよびインプリメンテーションに影響するデバイス アーキテクチャのプリミティブ コンポーネントに関連付けられるプロパティです。

属性は、次の方法で設定します。

- ジェネリック マップ (VHDL)
- プリミティブ コンポーネントのインスタンシエート中に渡される defparams またはインライン パラメーター (Verilog)

属性はすべて、ライブラリ ガイドのプリミティブ コンポーネントの説明に含まれています。

属性の例

- LUT4 コンポーネント上の INIT
- DCM 上の CLKFX_DIVIDE

インプリメンテーション制約

『制約ガイド』では、インプリメンテーション制約について記述しています。

インプリメンテーション制約では、FPGA インプリメンテーション ツールで FPGA デザインを処理する際に従うマップ、配置、タイミング、またはその他のガイドラインを指定します。

インプリメンテーション制約は、通常 UCF ファイルに記述されますが、次にも記述できます。

- HDL コード
- 合成制約ファイル

インプリメンテーション制約の構文例

- LOC (配置) 制約
- PERIOD (タイミング) 制約

CPLD Fitter

CPLD デバイスには、次の制約が適用されます。

BUFG (CPLD の場合)

COLLAPSE (コラプス)

COOL_CLK (クール クロック)

DATA_GATE (データ ゲート)

FAST

INREG (入力レジスター)

IOSTANDARD (I/O 規格)

KEEP

KEEPER

LOC (ロケーション)

MAXPT (最大積項)

NOREDUCE (削減なし)

OFFSET_IN (オフセット入力)

OFFSET_OUT (オフセット出力)

OPEN_DRAIN (オープン ドレイン)

PERIOD (周期)

PROHIBIT (禁止)

PULLUP (プルアップ)

PWR_MODE (電力モード)

REG (レジスター)

SCHMITT_TRIGGER (シュミット トリガー)

SLOW

TIMEGRP (タイミング グループ)

TIMESPEC (タイミング仕様)

TNM (タイミング名)

TSidentifier (タイミング指定識別子)

VREF

WIREAND (ワイヤード AND)

論理制約

論理制約とは、マップまたはフィットが実行される前にデザインのエレメントに設定する制約です。

- 論理制約を適用すると、予測されるワーストケース条件にデザインのパフォーマンスを適応させることができます。
- 論理制約は、次を実行すると物理制約に変換されます。
  1. ザイリンクスのアーキテクチャを選択
  2. デザインを配置配線またはフィット
- 論理制約を設定するには、ネットリスト制約ファイル (NCF) またはユーザー制約ファイル (UCF) に書き込まれた入力デザインの属性を使用します。
- 論理制約には、次の 3 種類があります。
  - 配置制約
  - 相対ロケーション制約 (RLOC 制約)

    FPGA デバイスの場合、相対ロケーション制約で次が指定できます。
    - ロジック エレメントを独立した集合にグループ化します。
    - デザイン全体での最終的な配置に関係なく、グループ内でのエレメント同士の位置を相対的に定義できます。
  - タイミング制約

    タイミング制約は、特定のパスまたはネットの集合に、最大許容遅延またはスキューを指定できます。

物理制約

注記 : この制約は、FPGA デバイスにのみ適用できます。

物理制約とは、物理デザインのエレメントに付ける制約です。

マップ

- 物理デザインとは、マップ後のデザインです。
- デザインをマップするときに、ネットリストと UCF ファイルにある論理制約は、物理制約 (特定のアーキテクチャに適用される制約) に変換されます。
- 物理制約はマップ中に作成される PCF (物理制約ファイル) で定義されます。

PCF

Physical Constraints File (PCF) の特徴 :

- マップで生成されるファイルです。
- 次の 2 つのセクションを含みます。
  - 回路図セクション

    ネットリストと UCF ファイルにある論理制約に基づいた物理制約が含まれます。
  - ユーザー セクション
    - 物理制約を追加するために使用します。
    - ユーザー生成の制約を設定する場合は、NCF または PCFではなく、UCFに入力することをお勧めします

マップ制約

マップ制約は、特定の動作を実行するように MAP に指示を与えます。

マップ制約

- AREA_GROUP (エリア グループ)
- BEL
- BLKNM (ブロック名)
- DCI_VALUE
- DRIVE (駆動電流)
- FAST
- HBLKNM (階層ブロック名)
- HLUTNM (階層ルックアップ テーブル名)
- HU_SET
- IOB
- IOBDELAY (I/O ブロック遅延)
- IOSTANDARD (I/O 規格)
- KEEP
- KEEPER
- LUTNM (ルックアップ テーブル名)
- MAP
- NODELAY (遅延なし)
- PULLDOWN (プルダウン)
- PULLUP (プルアップ)
- RLOC (相対位置)
- RLOC_ORIGIN (相対位置の原点)
- RLOC_RANGE (相対位置の範囲)
- SAVE_NET_FLAG (ネット フラグの保存)
- SLEW (スルー)
- U_SET
- USE_RLOC (RLOC の使用)
- XBLKNM

配置制約

ここでは、FPGA デザインの各種エレメントに設定する配置制約について説明します。エレメントには次のような種類があります。

- フリップフロップ
- ROM
- RAM
- BUFT
- CLB
- IOB
- I/O
- エッジ デコーダ
- グローバル バッファー

AND または OR のような論理ゲート :

- 制約の読み出し前に CLB ファンクション ジェネレーターにマップされます。
- 制約を付けることができません。

制約の指定

ほとんどの制約が次のいずれかで指定できます。

- HDL ソース コード
- ユーザー制約ファイル (UCF)

制約ファイルでは、配置制約は 1 つまたは複数のシンボルに適用されます。デザイン内のシンボルには個々に名前が付けられ、名前は入力ファイルで定義されます。シンボルに制約を設定するには、この名前を制約文で使用します。

大文字/小文字の区別

- UCF ファイルおよび NCF ファイルでは、大文字と小文字が区別されます。
- 識別子名 (ネット名などのデザイン内のオブジェクトの名前) は、ソース デザインのネットリストに記述されている名前と大文字/小文字が正確に一致しなければなりません。
- ザイリンクス キーワード (LOC、PROHIBIT、RLOC、BLKNM など) はすべて大文字かすべて小文字で入力できます。大文字と小文字は混用できません。

ネットリストのマップと配置制約

次の制約は、ネットリストのシンボルの配置やマップを制御するために使用します。

- BLKNM
- HBLKNM
- HLUTNM
- LOC
- LUTNM
- PROHIBIT
- RLOC
- RLOC_ORIGIN
- RLOC_RANGE
- XBLKNM

相対ロケーション制約 (RLOC 制約)

RLOC 制約は、ロジック エレメントを独立した集合にグループ化します。

- エレメントの位置は、デザイン全体の最終的な配置に関係なく、同じ集合内のほかのエレメントに相対的に定義できます。
- たとえば、1 列に並んだ 8 つのフリップフロップのグループに RLOC 制約を適用すると、マップは列の順番を保ったまま、フリップフロップのグループ全体を 1 つの単位として移動します。
- これに対して、絶対LOC 制約は、ほかのデザイン エレメントに関係なく、FPGA チップ上の特定の位置にデザイン エレメントを指定します。

配置制約

- AREA_GROUP
- BEL
- LOC
- LOCATE
- PROHIBIT
- RLOC
- RLOC_ORIGIN
- RLOC_RANGE
- USE_RLOC

配線制約

配線制約は、特定の動作を実行するように PAR に指示を与えます。

- AREA_GROUP
- CONFIG_MODE
- LOCK_PINS

合成制約

合成制約を使用すると、合成ツールによる特定のデザインまたは HDL コードの一部に対する最適化手法を制御できます。合成制約は、ソース コードに組み込まれるか、別の合成制約ファイルに含まれます。

次の制約が合成制約です。

- FROM-TO
- IOB
- KEEP
- MAP
- MARK_DEBUG
- OFFSET IN
- OFFSET OUT
- PERIOD
- TIG
- TNM
- TNM_NET

合成制約のマニュアル

XST 合成制約については、次を参照してください。

- 『XST ユーザー ガイド (Virtex-4、Virtex-5、Spartan-3、および CPLD デバイス用)』(UG627)
- 『XST ユーザー ガイド (Virtex-6、Spartan-6、および 7 シリーズ デバイス用)』(UG687)

その他の合成制約については、ソフトウェア ベンダーのマニュアルを参照してください。

タイミング制約

ザイリンクス ソフトウェアでは、グローバルまたはパス別にタイミング制約を使用することで正確なタイミング要件を指定することができます。

制約を定義するのに推奨される方法については、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

タイミング制約および関連のグループ制約は次のとおりです。

- ASYNC_REG
- DISABLE
- ENABLE
- FROM-THRU-TO
- FROM-TO
- MAXSKEW
- OFFSET_IN
- OFFSET_OUT
- PERIOD
- PRIORITY
- SYSTEM_JITTER
- TEMPERATURE
- TIG
- TIMEGRP
- TIMESPEC
- TNM
- TNM_NET
- TPSYNC
- TPTHRU
- TSidentifier
- VOLTAGE

コンフィギュレーション制約

- CONFIG_MODE
- DCI_CASCADE
- MCB_PERFORMANCE
- STEPPING
- POST_CRC
- POST_CRC_ACTION
- POST_CRC_FREQ
- POST_CRC_INIT_FLAG
- VCCAUX
- VREF
- INTERNAL_VREF_BANK

XILINX®

第 2 章

ザイリンクス制約の入力方法

この章では、ISE® Design Suite を使用して制約タイプを入力する方法を含む、ザイリンクス制約の入力方法について説明します。

制約の入力方法

次の表は、制約の入力に使用する ISE® Design Suite ソフトウェアを示したものです。

制約の入力方法

制約タイプ	ツール	デバイス
タイミング	Constraints Editor	すべての CPLD および FPGA
IO 配置制約およびエリア グループ制約	PlanAhead™ ソフトウェア	すべての FPGA
I/O 配置	PACE	すべての CPLD デバイス ファミリ
I/O 配置制約およびその他の配置制約	Schematic Editor/Symbol Editor	すべての CPLD および FPGA

制約の入力方法

次の表は、制約とその入力方法を示します。制約の詳細を参照するには、制約名をクリックしてください。

制約の入力方法

制約	回路図	VHDL Verilog	NCF	UCF	Constr-aints Editor	PCF	XCF	Plan-Ahead	PACE	FPGA Editor	ISE® Design Suite
AREA_GROUP	○		○	○	○			○			
ASYNC_REG		○	○	○	○						
BEL		○	○	○				○			
BLKNM	○	○	○	○			○				
BUFG (CPLD)	○	○	○	○			○				
CLOCK_DEDICATED_ROUTE			○	○							

制約	回路図	VHDL Verilog	NCF	UCF	Constr-aints Editor	PCF	XCF	Plan-Ahead	PACE	FPGA Editor	ISE® Design Suite
COLLAPSE	○	○	○	○							
COMPGRP						○					
CONFIG_MODE				○							
COOL_CLK	○	○	○	○							
DATA_GATE	○	○	○	○							
DEFAULT	○	○	○	○			○	○	○		
DCI_CASCADE			○	○		○					
DCI_VALUE			○	○							
DIRECTED_ROUTING			○	○						○	
DISABLE			○	○		○					
DRIVE	○	○	○	○			○	○	○	○	
ENABLE			○	○		○					
ENABLE_SUSPEND			○	○							
FAST	○	○	○	○			○	○	○		
FEEDBACK				○	○	○	○	○			
FILE	○	○									
FLOAT	○	○	○	○			○				
FROM-THRU-TO			○	○	○	○		○			
FROM-TO			○	○	○	○	○	○			
HBLKNM	○	○	○	○							
HLUTNM	○	○	○	○	○		○				
HU_SET	○	○	○	○			○				
IBUF_DELAY_VALUE	○	○	○	○							
IFD_DELAY_VALUE	○	○	○	○							
INREG	○			○							
IOB	○	○	○	○			○				○
IOBDELAY	○	○	○	○	○						
IODELAY_GROUP				○							
IOSTANDARD	○	○	○	○			○	○	○	○	
KEEP	○	○	○	○			○				
KEEPER	○	○	○	○	○		○			○	
KEEP_HIERARCHY	○	○	○	○			○				○

制約	回路図	VHDL Verilog	NCF	UCF	Constr-aints Editor	PCF	XCF	Plan-Ahead	PACE	FPGA Editor	ISE® Design Suite
LOC	○	○	○	○		○	○	○	○		
LOCATE						○				○	
LOCK_PINS		○	○	○							
LUTNM	○	○	○	○							
MAP	○		○	○							
MARK_DEBUG		○					○	○			
MAXDELAY	○	○	○	○	○	○				○	
MAX_FANOUT		○					○				○
MAXPT		○	○	○							
MAXSKEW	○	○	○	○	○	○				○	
NODELAY	○	○	○	○			○				
IODELAY_GROUP				○							
NOREDUCE	○	○	○	○			○				
OFFSET IN	○		○	○	○	○	○	○			
OFFSET OUT	○		○	○	○	○	○	○			
OPEN_DRAIN	○	○	○	○			○				
PERIOD	○	○	○	○	○	○	○	○		○	
PIN				○							
POST_CRC				○		○					
POST_CRC_ACTION				○		○					
POST_CRC_FREQ				○		○					
POST_CRC_INIT_FLAG				○		○					
PRIORITY			○	○		○					
PROHIBIT				○		○		○	○	○	
PULLDOWN	○	○	○	○			○	○	○	○	
PULLUP	○	○	○	○			○	○	○	○	
PWR_MODE	○	○	○	○			○				
REG	○	○	○	○			○				
RLOC	○	○	○	○			○	○			
RLOC_ORIGIN	○	○	○	○		○		○			
RLOC_RANGE	○	○	○	○		○	○				
SAVE NET FLAG	○	○	○	○			○				
SCHMITT_TRIGGER	○	○	○	○			○				

制約	回路図	VHDL Verilog	NCF	UCF	Constr-aints Editor	PCF	XCF	Plan-Ahead	PACE	FPGA Editor	ISE® Design Suite
SLEW	○	○	○	○			○	○	○	○	
SLOW	○	○	○	○			○	○	○	○	
STEPPING				○							
SUSPEND	○	○	○	○					○		
SYSTEM_JITTER	○	○	○	○			○				
TEMPERATURE			○	○	○	○					
TIG	○		○	○	○	○	○	○			
TIMEGRP			○	○	○	○	○	○			
TIMESPEC			○	○	○		○	○			
TNM			○	○	○		○	○			
TNM_NET	○		○	○	○		○	○			
TPSYNC	○		○	○							
TPTHRU	○		○	○	○						
TSidentifier			○	○	○	○	○			○	
U_SET	○	○	○	○			○				
USE_RLOC	○	○	○	○			○	○			
USE_INTERNAL_VREF		○	○	○			○				
VCCAUX			○	○							
VCCAUX_IO	○	○	○	○							
VCCOSENSEMODE			○	○							
VOLTAGE			○	○	○	○					
VREF	○		○	○							
WIREAND	○	○	○	○							
XBLKNM	○	○	○	○			○				

回路図デザイン

シンボルまたは回路図でザイリンクス制約を属性として設定するには、次のルールに従ってください。

- ネットに適用する場合、属性としてネットに設定します。
- インスタンスに適用する場合、属性としてインスタンスに設定します。
- PART や PROHIBIT のようなグローバル制約は設定できません。
- TIMESPEC や TIMEGRP などに設定されるタイミング要件も追加できません。
- 属性名および属性値は、大文字のみまたは小文字のみで入力します。大文字と小文字は一緒に使用できません。

属性の作成、変更、表示については、Schematic Editor/Symbol Editor ヘルプで手順を確認してください。

このマニュアルには、回路図に入力できる制約の構文例が記載されています。たとえば、BEL 制約の回路図構文は、「BEL」の「回路図からの設定」を参照してください。

VHDL 属性

VHDL コードの場合、制約は VHDL 属性で記述します。次のような構文で制約を宣言してから、制約を使用する必要があります。

attribute *attribute_name* : **string;**

例

attribute RLOC : string;

属性は、エンティティまたはアーキテクチャで宣言できます。

- エンティティで宣言すると、エンティティとアーキテクチャ本体の両方で属性を使用できます。
- アーキテクチャで宣言した場合は、その属性はエンティティ宣言では使用できません。

VHDL 属性を宣言した後、次のように指定します。

```
attribute attribute_name  of {component_name| label_name|entity_name|signal_name |variable_name|type_name}:
{component|label| entity|signal |variable |type} is    attribute_value;
```

使用できる属性値 *attribute_values* は、属性のタイプによって異なります。

例 1

attribute RLOC : string;

attribute RLOC of u123 : label is "R11C1.S0";

例 2

attribute bufg: string;

attribute bufg of my_clock: signal is "clk";

ザイリンクスで最もよく使用されるオブジェクトは、signal、entity、label です。label はコンポーネントのインスタンスを表します。

注記 : signal 属性は、出力ポートに使用される必要があります。

VHDL では大文字と小文字は区別されません。

ザイリンクス制約が VHDL キーワードである場合、VHDL 属性でその制約を使用できません。この問題を回避するには、制約エイリアスを使用します。各制約には、固有のエイリアスがあります。エイリアス名は、接頭辞 XIL_ に制約名を付けたものです。たとえば、RANGE 制約は VHDL 属性には使用できないので、XIL_RANGE を使用します。

Verilog 属性

Verilog 属性は、(*) で囲んで、次の構文のように記述する必要があります。

(* *attribute_name* = *attribute_value* ***)**

説明 :

- attribute を参照する信号、モジュール、またはインスタンスの宣言の前に記述する必要があります。
- attribute_value には、文字列を指定する必要があります。整数値やスカラ値は使用できません。
- attribute_value は、二重引用符 (″ ″) で囲む必要があります。
- デフォルト値は 1です。**(* attribute_name *)** は **(* attribute_name = ″1″ *)** と同じです。

Verilog 属性の構文例 1

```
(* clock_buffer = "IBUFG" *) input CLK;
```

Verilog 属性の構文例 2

```
(* INIT = "0000" *) reg [3:0] d_out;
```

Verilog 属性の構文例 3

```
always@(current_state or reset)
  begin (* parallel_case *) (* full_case *)
   case (current_state)
```

Verilog 属性の構文例 4

```
(* mult_style = "pipe_lut" *) MULT my_mult (a, b, c);
```

Verilog の制限

次の Verilog 属性は、サポートされていません。

- 信号の宣言
- ステートメント
- ポートの接続
- 論理演算子

Verilog のメタ コメント

メタ コメントを使用すると、制約を Verilog コードで指定することもできます。Verilog 形式がよく使用されますが、Verilogでもメタ コメントがサポートされています。次の構文を使用してください。

```
// synthesis attribute AttributeName [of] ObjectName [is] AttributeValue
```

Verilog のメタ コメントの例

```
// synthesis attribute RLOC of u123 is R11C1.S0
// synthesis attribute HU_SET u1 MY_SET
// synthesis attribute bufg of my_clock is "clk"
```

UCF (ユーザー制約ファイル)

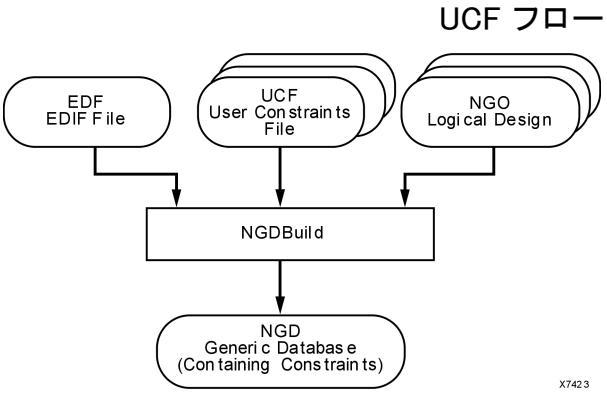
UCF ファイルは、論理デザインに制約を指定する ASCII 形式のファイルです。UCF ファイルは、次を使用すると、ユーザーが作成して制約を入力できます。

- すべてのテキスト エディター
- Constraints Editor

このファイルで指定する制約によって、論理デザインがターゲット デバイスでどのようにインプリメントされるかが決定されます。UCF ファイルは、デザイン入力中に指定した制約を上書きする際に使用されます。

UCF フロー

次の図に、UCF フローを示します。



UCF ファイルは NGDBuild の入力ファイルです (上図を参照)。UCF ファイルに記述された制約は、NGDBuild で生成される NGD ファイルの情報の一部となります。FPGA デバイスの場合、これらの制約はデザインのマップで使用されたり、MAP で生成される PCF (物理制約ファイル) に書き込まれます。

PCF ファイルの制約は、デザインのマップ後に実行する PAR やタイミング解析ツールなどのツールで使用されます。

手動でのタイミング制約の入力

Constraints Editor でタイミング制約を入力する方法以外に、タイミング仕様を制約として UCF ファイルに手動で入力することもできます。その後、NGDBuild を実行する際に、タイミング制約は NGD の一部としてデザインのデータベースに追加されます。Constraints Editor を使用しても UCF ファイルに制約を作成できます。

UCF ファイルが複数ある場合の制約の競合

ザイリンクスでは、HDL/NCF/UCF/PCF プロセスと同様、最後に入力した制約が優先される方式をとっています。現在のところ、UCF ファイルはプロジェクトに追加した順序で処理されます (ISE® Design Suite または Tcl コマンド)。タイムスタンプ、またはファイルが修正された順序は記録されません。

UCF および NCF の構文

論理制約は、次のファイルに記述されています。

- NCF (ネットリスト制約ファイル) : 合成プログラムにより生成される ASCII 形式のファイル
- UCF (ユーザー制約ファイル) : ユーザーが作成する ASCII 形式のファイル

ユーザー制約を設定する場合は、PCF ではなく、UCF または NCF に入力することをお勧めします。

UCF および NCF の一般規則

- UCF ファイルおよび NCF ファイルでは、大文字と小文字が区別されます。識別子名 (ネット名などのデザイン内のオブジェクトの名前) は、ソース デザインのネットリストに記述されている名前と大文字/小文字が正確に一致しなければなりません。ただし、LOC、PERIOD、HIGH、LOW などの制約キーワードは、大文字か小文字のいずれか、またはその両方を使用できます。
- 各制約文の終わりには、セミコロン (;) を付けます。
- 制約文の最後にセミコロンを付けるため、文が 2 行にまたがる場合でも行継続文字を使用する必要はありません。
- 似たようなブロックまたはコンポーネントのタイミング制約は個別に設定せず、まとめて 1 つのタイミング制約を設定します。
- UCF および NCF にコメントを追加するには、各コメント行の先頭にシャープ記号 (#) を付けます。例は次のとおりです。

```
# file TEST.UCF
# net constraints for TEST design
NET "$SIG_0 MAXDELAY" = 10;
NET "$SIG_1 MAXDELAY" = 12 ns;
```

- UCF および NCF では、文を特定の順序に並べる必要はありません。
- ネット名 (NET) およびインスタンス名 (INST) は、ダブル クォーテーションで囲むことをお勧めします。ただし、これはエラーを防ぐためであり、必須条件ではありません。
- ~OUTSIG1 のようなチルダを含む反転信号名は、必ずダブル クォーテーションで囲んでください。
- 指定したインスタンスに対して複数の制約を設定できます。詳細は、下記の「複数の制約の入力」を参照してください。

制約の競合

UCF/NCF ファイル、回路図、合成ファイルにそれぞれ記述された制約は、同等に適用されます。制約がどのファイルに入力されているかは問題ではありません。制約が重複した場合、UCF の制約が NCF や回路図/ネットリストの制約よりも優先されます。NCF の制約は回路図/ネットリストの制約よりも優先されます。

1 つの位置に複数のエレメントを誤ってロックしてしまった場合、マップにより競合が検知されてエラー メッセージが表示されるので、エラーを修正します。

構文

UCF ファイルでは、次のような基本的な構文がサポートされます。

{**NET**|**INST**|**PIN**} "*full_name*" *constraint*;

- *full_name* は、階層名が付いたオブジェクト名です。名前がピンを表す場合は、そのエレメントのインスタンス名も必要です。
- *constraint* は、回路図オブジェクトに属性を設定する場合に使用されるのと同じ形式の制約です。LOC=P38 や FAST などがその例です。

TIMEGRP および TIMESPEC 属性の指定

詳細は、『タイミング クロージャ ユーザー ガイド』(UG612) を参照してください。

複数制約の入力

指定したインスタンスに対して複数の制約を UCF ファイルに設定することができます。

INST *instanceName constraintName* **=** *constraintValue* | *constraintName* **=** *constraintValue*;

次に例を示します。

INST myInst LOC = P53 | IOSTANDARD = LVPECL33 | SLEW = FAST;

ファイル名

デフォルトで制約ファイルは NGDBuild によって読み出されます。制約ファイル名は入力デザイン名と同じで、拡張子は .ucf です。異なる制約ファイル名を指定するには、NGDBuild を実行する際に **-uc** オプションを使用します。EDIF 入力ファイルと同じベース名を持つ NCF が EDIF と同じディレクトリに含まれている場合、NCF が自動的に読み込まれます。

NGDBuild、MAP、PAR などのインプリメンテーション ツールを実行するには、コマンド ラインでファイル名の拡張子を .ucf などとすべて小文字で入力する必要があります。

インスタンスおよびブロック

次に、制約ファイルの構文で使用されるインスタンスとブロックを定義します。

- *instance* : 回路図上のシンボルです。
- *instance name* : EDIF ネットリストに含まれるシンボル名です。
- *block* : CLB または IOB のことです。
- *block name* : BLKNM、HBLKNM、または XBLKNM 属性を使用して指定できます。デフォルトでは、ブロックに関連する信号名に基づいてブロック名が割り当てられます。

物理制約ファイル (PCF)

デザイン ネットリストが ISE® Design Suite に読み込まれる際に生成される Native Generic Database (NGD) ファイルには、多数の論理制約が含まれます。これらの制約は次のいずれかをソースとしています。

- 回路図または HDL ファイルに記述された属性
- UCF の制約
- CAE ベンダー ツールにより生成された NCF (ネットリスト制約ファイル) の制約

NGD に記述された論理制約は、MAP により読み込まれます。一部の論理制約は、デザインのマップに使用され、物理制約に変換されます。物理制約はその後、物理制約ファイル (PCF) に書き込まれます。

PCF は、次の 2 つのセクションに分割された ASCII 形式のファイルです。

- マップにより作成された物理制約のセクション
- ユーザーが入力した物理制約のセクション

マップで生成された制約のセクションは、マップを実行するたびに書き込まれます。

ファイルでは、マップで生成された制約の後にユーザーが入力した制約が記述されています。制約の競合が発生しても、常にこの順番が維持され、ユーザーが入力した制約が最後に読み込まれてマップで生成された制約に上書きされます。

マップで生成されたセクションは、**SCHEMATIC START** で始まり、**SCHEMATIC END** で終了します。タイミング制約などのユーザー制約を入力する場合は、常に SCHEMATIC END の後に記述してください。

ユーザー制約はファイルに直接書き込むか、FPGA Editor を使用して入力します。FPGA Editor の制約情報については、FPGA Editor ヘルプを参照してください。

注記 : デザイン制約は、できる限り HDL、回路図、または UCF ファイルに書き込んでください。PCF ファイル以外のファイルに書き込むと、デザインを簡単に保存でき、デザイン ルール チェックが向上します。

PCF ファイルは、PAR、FPGA Editor、TRACE、NetGen および BitGen の入力ファイル (オプション) として使用されます。このファイルには制約およびコメントをいくつでも含めることができます。コメントは、シャープ記号 (#) またはダブル スラッシュ (//) で始めます。文字数に制限はありませんが、1 行以内におさめる必要があります。

PCF ファイルの構文は次のとおりです。

schematic start;

translated schematic and UCF and NCF constraints in PCF format

schematic end;

user-entered physical constraints

注意 : ユーザー制約は、schematic end 文の後に記述する必要があります。このセクションの前またはセクション内に記述された制約は、上書きされるか、無視される場合があります。

回路図制約には変更を加えないでください。この制約は、マップで新しい PCF ファイルが作成されるたびに上書きされます。

グローバル制約は、オブジェクトに設定せず、制約ファイルに入力してください。

各制約文の最後に、セミコロン (;) を付ける必要があります。

NCF、UCF、および PCF のすべての制約ファイルで、内部予約語と一致するインスタンス名または変数名をダブル クォーテーションで囲んでおかないと処理されません。したがって、すべての名前を二重引用符で囲むことをお勧めします。たとえば、net は予約語なので、次の構文は使用できません。

```
NET net FAST;
```

この場合は、次のように入力することをお勧めします。

```
NET "net" FAST;
```

ネットリスト制約ファイル (NCF)

NCF ファイルの構文ルールは UCF のルールと同じです。詳細は、「UCF および NCF のファイル構文」を参照してください。

XCF (ザイリンク制約ファイル)

Constraints Editor :

- タイミング制約の入力プロセスを単純化するグラフィカル ツールです。
- UCF 構文を理解していなくても制約を簡単に作成できます。
- 変換 (NGCBuild) 後のインプリメンテーション段階で使用されます。
- UCF を直接編集せずに制約を作成および変更できます。

Constraints Editor を使用して制約を作成または修正した場合は、NGCBuild を再実行する必要があります。この 2 回目の実行では、次が入力および出力されます。

- 新規 UCF とデザイン ソース ネットリスト ファイルが入力として使用されます。
- 新規 NGD ファイルが出力として生成されます。

Constraints Editor を使用できる制約およびデバイスについては、「制約の入力方法」を参照してください。Constraints Editor の実行方法の詳細は、ISE® Design Suite ヘルプを参照してください。

入力/出力

Constraints Editor には、次が必要です。

- UCF (ユーザー制約ファイル)
- Native Generic Database (NGD) ファイル

Constraints Editor は、グループ化する論理エレメントの名前を記述するのに NGD ファイルを使用します。出力には UCF が使用されます。

Constraints Editor を起動して、UCF ファイルを開きます。ここで UCF および NGD のベース名が異なる場合、正しい NGD ファイルを選択する必要があります。詳細は、Constraints Editor ヘルプを参照してください。

制約を設定すると、Constraints Editor により制約が UCF に出力されます。UCF ファイルは、NGCBuild (変換) によりデザイン ソースのネットリストと共に使用され、NGD ファイルが生成されます。NGD ファイルは MAP プログラムで読み出されます。MAP は、物理的デザインのデータベースを Native Circuit Description (NCD) ファイル形式で生成し、PCF (物理制約ファイル) も生成します。これらのファイルはインプリメンテーション ツールによって使用され、最終的にビットストリームが生成されます。

注記 : すべてのザイリンクス制約が Constraints Editorで設定できるわけではありません。

Constraints Editor の起動

Constraints Editor は PC および UNIX で実行でき、次のいずれかの方法で起動できます。

- ISE Design Suite から
- スタンドアロンとして
- コマンド ラインから

ISE Design Suite からの Constraints Editor の起動

ISE Design Suite の場合、[Processes] ペインから Constraints Editor を起動します。

1. [Sources] ペインでデザイン ファイルを選択します。
2. [Processes] ペイン→ [User Constraints] → [Create Timing Constraints] をダブルクリックします。

スタンドアロン ツールとしての Constraints Editor の起動

Constraints Editor をスタンドアロン型ツールとしてインストールしている場合は、次のいずれかに従って起動します。

- Windows デスクトップ上の [Constraints Editor] アイコンをクリックします。
- [スタート] → [プログラム] → [Xilinx ISE] → [アクセサリ] → [Constraints Editor] をクリックします。

データを読み込まずにコマンド ラインから Constraints Editor を起動する方法

データを読み込まずに Constraints Editor を起動するには、次のように入力します。

constraints_editor

NGD ファイルを読み込んでコマンド ラインから Constraints Editor を起動する方法

NGD を読み込んで Constraints Editor を起動するには、次のように入力します。

constraints_editor *ngdfile_name*

ngdfile_name は、NGD ファイル名です。

拡張子 `.ngd` を使用する必要があります。

NGD ファイルと同じベース名を持つ UCF がある場合、その UCF も一緒に読み込まれます。同じベース名のファイルがない場合は、UCF を指定するようメッセージが表示されます。

NGD および UCF を読み込んでコマンド ラインから Constraints Editor を起動する方法

NGD および UCF を読み込んで Constraints Editor を起動するには、次のように入力します。

constraints_editor *ngdfile_name* **-uc** *ucf_file_name*

- *ngdfile_name* は、NGD ファイル名です。
- *ucf_file_name* は、UCF ファイル名です。

拡張子 `.ucf` を使用する必要があります。

バックグラウンド プロセスとしてコマンド ラインから Constraints Editor を起動する方法

UNIX でバックグラウンド プロセスとして Constraints Editor を起動するには、次のように入力します。

constraints_editor &

ISE Design Suite

ISE® Design Suite でインプリメンテーション制約を設定するには

- FPGA デバイスを使用している場合、インプリメンテーションのプロセス プロパティで、デザインの変換、マップ、配置、および配線の方法を指定できます。複数のプロパティを設定して、インプリメンテーション プロセスを制御できます。
- CPLD デバイスを使用している場合、インプリメンテーションのプロセス プロパティで、デザインの変換やフィットの方法を指定できます。

詳細は、ISE Design Suite ヘルプの [Process Properties] ダイアログ ボックスに関するセクションを参照してください。

PlanAhead

PlanAhead™ ソフトウェアは、合成前と合成後のいずれかで使用できます。PlanAhead ソフトウェアでは、次のデバイスがサポートされます。

- Virtex®-4 以降のデバイス
- Spartan®-3 以降のデバイス

PlanAhead ソフトウェアでは、次のような配置制約をドラッグ アンド ドロップできます。

- ピン配置
- LOC (配置) 制約
- エリア

詳細は、ライブラリガイドを参照してください。

配置制約の割り当て

FPGA デバイスをターゲットにする場合は、PlanAhead ソフトウェアを使用して次を制御する配置制約を入力できます。

- I/O ピンおよびロジックの割り当て
- グローバル ロジックの配置
- エリア グループの割り当て

PlanAhead ソフトウェアは、ユーザーがデザインを解析したり、配置制約を適用したりできるように、デザイン プロセスのさまざまなプロセス段階で自動的に実行されます。PlanAhead の簡易版は ISE® Design Suite から起動され、選択したタスクを実行するのに必要な機能だけが使用可能になります。スタンドアロンの PlanAhead には、さらに多くの機能が含まれます。

PlanAhead ソフトウェアが ISE Design Suite から起動される場合、CPU プロセスは別で、その他一部のツールと同様、ISE Design Suite とリアルタイムで連動していません。PlanAhead を起動中に ISE Design Suite ソース ファイルをアップデートすると、データが一致しなかったり、同期エラーになったりすることがあります。

PlanAhead を起動中は、ソース ファイルをまず PlanAhead に渡してから、PlanAhead プロジェクトを作成してください。PlanAhead プロジェクトを保存すると、修正された UCF ファイルのみが ISE Design Suite に戻されて、プロジェクトがアップデートされます。入力ソース ファイルは、起動するプロセスによって異なります。

どの形式のファイルが渡されるかは、この後の「ピン配置」および「フロアプランおよび配置制約」セクションを参照してください。次のセクションでは、PlanAhead を使用した配置制約の入力について説明します。

I/O ピン コンフィギュレーションの定義

このセクションでは、I/O ピン コンフィギュレーションの定義について次の内容に分けて説明します。

- ピンの割り当ての概要
- I/O ピン データ情報の確認
- ピンの割り当て
- I/O Planner の資料

ピンの割り当ての概要

I/O Planner は、スタンドアロン ツールとして起動できるほか、ISE Design Suite からも起動できます。スタンドアロンの I/O Planner は、HDL ソースがまだ完成していないようなデザイン プロセスの早期段階で使用すると便利です。I/O ポートはこのツール内で手動で定義できるほか、CSV 形式のスプレッドシートや HDL ソースにもインポートできます。初期のピン配置を定義してから UCF ファイルをエクスポートして、ISE Design Suite フローで使用することもできます。

ISE Design Suite から I/O Planner を起動する場合は、UCF ファイルが必要です。UCF ファイルがないと、空のファイルが作成されます。ISE Design Suite から I/O Planner を起動する場合は、I/O ポートの手動作成や CSV スプレッドシートのインポートはできません。

I/O Planner は、さまざまな便利なビューや機能を含む I/O ピン割り当てのための環境です。I/O ポートのグループをさまざまな方法で別々にデバイスにドラッグ アンド ドロップできます。自動配置ルーチンも使用できます。デザイン ルール チェック (DRC) を使用することで、有効なピン配置の定義を確認することもできます。

I/O ピン データ情報の確認

I/O 規格を含むデバイス仕様については、データシートを参照してください。デバイス特有の I/O 規格情報については、ターゲットにするデバイスのデータ シートを参照してください。データシートに含まれるデータの多くは、I/O Planner ツールからも入手できます。入手可能な情報には、I/O 規格、クロック対応のピン、内部トレース遅延、差動ペア、クロック領域、I/O バンクの内容などがあります。グローバル/リージョナル クロック バッファー、I/O 遅延、遅延コントローラ、ギガビット トランシーバなどの I/O 関連のデバイス リソースについての情報も含まれます。

ピンの割り当て

PlanAhead を起動するには、Windows の PlanAhead のデスクトップ アイコンをクリックするか、Linux コマンド ラインで PlanAhead と入力します。ISE Design Suite から起動する場合は、次のいずれかの方法でピン割り当てプロセスを実行します。一番適切な方法を使用してください。

- [Floorplanning I/O – Pre-Synthesis]

  このコマンドまたはプロセスを使用すると、HDL ソース ファイルが PlanAhead ソフトウェアに渡され、最上位レベルの I/O ポート情報のみが抽出されます。UCF が ISE Design Suite プロジェクトに既に存在する場合は、それが PlanAhead ソフトウェアに渡されます。UCF ファイルがない場合は、作成することを促すメッセージが表示されます。UCF ファイルが複数存在する場合、新しい制約を追加するために使用するファイルを選択するように促すメッセージが表示されます。既存の制約は、それが含まれるファイルで修正されます。

  PlanAhead に含まれる I/O Planner の使用方法については、「I/O Planner のマニュアル」セクションを参照してください。

  I/O ピンの割り当てができたら、PlanAhead プロジェクトを保存して PlanAhead を終了します。これで ISE Design Suite プロジェクトの UCF ファイルがアップデートされ、プロジェクト ステータスもアップデートされます。PlanAhead を保存せずに閉じると、ISE Design Suite の UCF ソース ファイルまたはステータスが変更されません。

- [Floorplanning a Design – Post-Synthesis]

  このコマンドまたはプロセスを使用すると、合成済みのネットリスト ソース ファイルが PlanAhead に渡されます。UCF が ISE Design Suite プロジェクトに既に存在する場合は、それが PlanAhead ソフトウェアに渡されます。UCF ファイルがない場合は、作成することを促すメッセージが表示されます。UCF ファイルが複数存在する場合、新しい制約を追加するために使用するファイルを選択するように促すメッセージが表示されます。既存の制約は、それが含まれるファイルで修正されます。

  合成済みネットリストを入力ファイルとして使用すると、I/O Planner でクロックおよびクロック関連のロジックが認識されるようになるので、使える の機能がさらに増えます。I/O 配置機能および DRC が使用できるようになり、さらに知的なピン割り当てが可能になります。デザインの接続も解析できるようになり、I/O に関するデバイス リソースを最小限に抑えることができます。

  PlanAhead に含まれる I/O Planner の使用方法については、「I/O Planner のマニュアル」セクションを参照してください。

  I/O ピンの割り当てができたら、PlanAhead プロジェクトを保存して PlanAhead を終了します。これで Project Navigator プロジェクトの UCF ファイルがアップデートされ、プロジェクト ステータスもアップデートされます。PlanAhead を保存せずに閉じると、ISE Design Suite の UCF ソース ファイルまたはステータスが変更されません。

I/O 配置ストラテジ

『PlanAhead ユーザー ガイド』(UG632) の「I/O ピンの配置」には、デバイス リソースと I/O ピン割り当ての解析に関する情報が含まれます。

『PlanAhead ソフトウェア チュートリアル : I/O ピン配置』(UG674)、『ピン配置手法ガイド』(UG792) も参照してください。

フロアプランおよび配置制約

PlanAhead は、接続、集積度、タイミングなどを含むさまざまな側面からデザインを解析する総合的な環境を提供するツールです。配置制約を設定すると、結果がより良く、一貫性のあるものになるように、インプリメンテーション ツールを導くことができます。この配置制約には、特定のロジックをデバイスの特定のサイトに固定する LOC 制約やロジックのグループをデバイスの特定エリアに制約する AREA_GROUP 制約も含まれます。

配置制約 (LOC) の割り当て

PlanAhead を使用すると、どのロジックでも特定のデバイス サイトに固定できます。これは、BUFG、BRAM、MULT、PPC405、GT、DLL、DCM のようなグローバル ロジック オブジェクトに設定することもできます。

ロジック オブジェクトは、そのロジック オブジェクトを該当する PlanAhead のビューからドラッグして、ワークスペースの [Device] ビューにドロップするだけで配置できます。I/O ポートのような、そのオブジェクトの [General Properties] ビューに該当ロケーション サイトを入力することができるロジックもあります。

配置制約の割り当ての詳細は、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」の「配置制約の使用」のセクションを参照してください。

エリア グループの割り当て

エリアグループは、たとえば特定のクロック 領域内などのデバイスの特定の領域にロジックを配置する主な手段です。PlanAhead を使用するとさまざまな手段でエリア グループを作成できます。DRC を含め、接続、サイズ ロジック タイプ、範囲などが提供されるので、最適な AREA_GROUP プロパティの設定が可能です。

エリア グループ制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」セクションを参照してください。

PACE での制約の設定

CPLD の場合、制約は Pinout and Area Constraints Editor (PACE) で設定できます。PACE でピンの事前割り当て機能は、次の目的で使用します。

- IO へのロケーション制約の割り当て
- IO 規格などの IO プロパティの割り当て

PACE を使用できる制約およびデバイスについては、「制約の入力方法」を参照してください。PACE の起動および使用方法の詳細については、ISE® Design Suite ヘルプを参照してください。

パーシャル デザインのピンの事前割り当て

このセクションでは、パーシャル デザインにおけるピンの事前割り当てについて説明します。定義したピンに制約を追加すると HDL テンプレートが生成される、「ピンの事前割り当て」は PlanAhead™ または PACE で実行できます。詳細は、ISE® Design Suite ヘルプを参照してください。PACE は、CPLD 用、PlanAhead は FPGA 用です。

デザインが未完成でも、ピンの割り当て、電圧標準、バンク規則、その他のボードの条件などのレイアウト要件が整っていることがあります。ピンの事前割り当ての機能を使用すると、デザインのロジックが完成していなくてもデザインのピン配置ルールを決定できます。

PlanAhead または PACE でピンの事前割り当て機能を使用するには、次の手順に従います。

1. 最上位デザインのポートをすべて特定します。
2. I/O 制約を割り当てます。

入力ピンにロードがない、出力ピンにソースがないなどのように、デザイン内のどのロジックでも使用されていないポートも制約の適用を受け、インプリメントされます。

ほかの I/O ピンと同じように、LOC または IOSTANDARD 制約を UCF で割り当てます。これらの要件はデータベースでアノテートされます。PlanAhead および PACE を使用して、ピンのロケーション、バンク グループ、電圧標準を割り当て、DRC チェックを実行します。最終的な PAD レポートには、ロジック、または制約が関連付けられたピンが含まれます。

このデザイン インプリメンテーションは完全なものではなく、ハードウェアにはダウンロードできません。BitGen (ビットストリーム生成) の デザイン ルール チェックで次のようなエラーが発生します。

- ERROR: PhysDesignRules:368 – The signal <D_OBUF> is incomplete. The signal is not driven by any source pin in the design.
- ERROR: PhysDesignRules:10 – The network <D_OBUF> is completely unrouted.

デザインから使用されていないポートを取り除くには、関連付けられた制約を削除します。変換プロセス (NGDBuild) で使用されていないピンが削除されます。

以下の例では、最上位レベルに 6 つのポートがあり、clk、A、C の 3 つだけがデザインで使用されています。残りの 3 つのポートは、次のとおりです。

- B は LOC 制約があるので保持
- D は IOSTANDARD 制約があるので保持
- E は使用されておらず、制約もないのでデザインから削除

Verilog コード例

```
---------------
module design_top(clk, A, B, C, D, E);
input clk, A, B;
output reg C, D, E;

always@(posedge clk)
C <= A;

endmodule
```

UCF の例

```
--------

NET "A" LOC = "E2" ;
NET "B" LOC = "E3" ;
NET "C" LOC = "B15" ;
NET "D" IOSTANDARD = SSTL2_II ;
```

FPGA Editor

FPGA Editor を使用すると、Physical Constraints File (PCF) の一部の制約を追加または削除できます。FPGA Editor では、各ネットおよびコンポーネントのプロパティ フィールドとして、ネット、サイトおよびコンポーネント制約がサポートされます。プロパティは Setattr コマンドで設定でき、Getattr コマンドで読み込まれます。

LOCATE、LOCK, OFFSET IN、OFFSET OUT および PROHIBIT などを含むすべてのブール型制約の値は、**On** か **Off** になります。オフセット方向の値は **In** か **Out** になります。オフセット順序の値は **Before** か **After** になります。それ以外の制約には数値を指定します。制約を削除するには Off を指定します。値の指定には大文字と小文字を区別する必要はありません (On でも on でも使用できます)。

FPGA Editor で制約を作成する場合、デザインを保存するたびに PCF ファイルに制約が書き込まれます。FPGA Editor を使用して制約を削除し、デザイン ファイルに保存した場合、PCF ファイル内で制約が記述された行は自動的にコメントとして保持されます。次の表に、FPGA Editor でサポートされる制約を示します。

FPGA Editor でサポートされる制約

制約	制約を指定する場所
block paths	[Component Properties and Path Properties] プロパティ シート
define path	[Viewed with Path Properties] プロパティ シート
location range	[Component Properties Constraints] ページ
locate macro	[Macro Properties Constraints] ページ
lock placement	[Component Properties Constraints] ページ
lock routing of this net	[Net Properties Constraints] ページ
lock routing	[Net Properties Constraints] ページ
maxdelay allnets	[Main Properties Constraints] ページ
maxdelay allpaths	[Main Properties Constraints] ページ
maxdelay net	[Net Properties Constraints] ページ
maxdelay path	[Path Properties] プロパティ シート
maxskew	[Main Properties Constraints] ページ
maxskew net	[Net Properties Constraints] ページ
offset comp	[Component Properties Offset] ページ
penalize tilde	[Main Properties Constraints] ページ
period	[Main Properties Constraints] ページ
period net	[Net Properties Constraints] ページ
prioritize net	[Net Properties Constraints] ページ
prohibit site	[Site Properties] プロパティ シート

固定されたネットおよびコンポーネント

ネットがロックされている場合、ネット全体、ネットの一部、ピン、またはワイヤなどの配線を解除できません。ネットの配線を解除するには、まずロックを解除する必要があります。ロックされたネットには、ピンや配線を追加できます。

PCF ファイルで Lock Net [*net_name*] 制約が有効になっている場合、FPGA Editor でネットはロックされた状態で表示されます。[Net Properties] プロパティ シートを使用するとロック制約を削除できます。

コンポーネントがロックされると、次のいずれかの制約が PCF ファイルで設定されます。

lock comp [*comp_name*]

locate comp [*comp_name*]

lock macro [*macro_name*]

lock placement

コンポーネントが固定されている場合、配置の解除はできませんが、配線は解除できます。コンポーネントの配置を解除するには、まず固定を解除する必要があります。

制約間の相互関係

回路図制約は、MAP により PCF ファイルの最初に記述されます。このセクションは **SCHEMATIC START** で開始され、**SCHEMATIC END** で終了します。ユーザー制約は、SCHEMATIC END の後に入力する必要があります。

ユーザー制約は、回路図制約セクションの前に記述してもかまいませんが、競合する制約があった場合、回路図制約が優先されることがあります。

デザインのマップが実行されるたびに、PCF ファイルの回路図セクションが上書きされます。原則としてユーザー制約セクションはそのまま保持されますが、制約によっては無効になるものがあります。

ザイリンクス制約ファイル (XCF)

XST 制約は、ザイリンクス制約ファイル (XCF) で指定できます。制約ファイルの拡張子は .xcf です。詳細は、次を参照してください。

- ISE® Design Suite ヘルプ
- XST Constraint File (XCF)『XST ユーザー ガイド』の「XCF (ザイリンクス制約ファイル)」セクション

制約の優先順位

詳細は、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

場合によっては、2 つ以上のタイミング制約がデザイン内の同じパスに設定されることがあります。このような場合、そのパスに対して優先度の高い制約が適用され、優先度の低い制約はそのパスでは無視されるようにして、制約同士の競合が起こらないようにする必要があります。優先順序は、制約を指定した順序と制約の優先度の両方の条件によって異なります。優先順序の規則は、次のようになります。制約の優先度によって決まるほか、同じ優先度の制約が重なっている場合は、PCF ファイルでのどの制約が後に表示されているかによって決まります。たとえば、同じパスをカバーする 周期 (PERIOD) 制約が 2 つある場合、PCF ファイルで後の方に記述されている 周期 (PERIOD) 制約が使用され、これらのパスが解析されます。前の方の 周期 (PERIOD) 制約については、タイミング レポートで「0 items analyzed」と表示されます。このデフォルトの優先順序を変更するびは、**PRIORITY** キーワードを使用して優先度を設定する必要があります。

ファイルの優先順位

同じ優先度の制約により競合が発生する場合、指定する順序によってどちらの制約が優先されるかが決まります。同じ優先度の制約の場合は、最後に記述された制約が前に記述された制約を上書きします。この規則は、1 つの UCF ファイルだけでなく、複数の UCF ファイルで定義された制約にも適用されます。

次のリストは、同じ優先度の制約が別の制約ファイルで定義されている場合に、どのファイルの制約が優先されるかを表示しています。優先度の高いファイルの制約から順にリストしています。

- Physical Constraints File (PCF) の制約
- User Constraints File (UCF) の制約
- Netlist Constraints File (NCF) の制約
- 回路図の属性、またはネットリストに渡される HDL で指定された制約

XILINX®

第 3 章

ザイリンクス制約

ザイリンクス制約には、それぞれ次の内容が含まれます。

- アーキテクチャ サポート
- 適用可能エレメント
- 説明
- 適用ルール
- 構文
- 構文例
- 必要であれば、その他追加情報

基本的な VHDL 構文については、「VHDL 属性」を参照してください。

基本的な Verilog 構文については、「Verilog 属性」を参照してください。

制約に関する情報

各制約のセクションには、次の情報が記載されています。

- アーキテクチャ サポート

  どのデバイスで制約を使用できるかを示します。
- 適用可能エレメント

  制約を適用するエレメントを示します。
- 説明

  使用法およびビヘイビアーなど制約について説明します。
- 適用ルール

  どのように制約が適用されるかを示します。
- 構文例

  構文例をツールまたは手法ごとに示します。例で網羅されていないツールや手法には、制約に使用できないものがあります。
- その他の情報

  制約によっては、追加の情報が記載されています。

AREA_GROUP

AREA_GROUP (エリア グループ) 制約には、次の特徴があります。

- デザイン インプリメンテーション制約です。
- マップ、パック、配置配線のためにデザインを物理的な領域に分割できるようにします。
- デザインの論理ブロックに適用されます。

この制約の値には、MAP によりパックされる論理ブロックのグループや、PAR により指定範囲に配置される論理ブロックのグループ名を文字列で指定します。制約を階層ブロックに設定すると、その階層ブロック内のすべての下位ブロックは、同じグループに割り当てられます。

AREA_GROUP 制約を設定すると、この制約に関連したさまざまな制約を追加で使用して、インプリメンテーションを制御できます。詳細は、次の「制約の構文」セクションを参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

- ロジック ブロック
- タイミング グループ

詳細は、次の「タイミング グループによる定義」を参照してください。

適用ルール

- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。
- ネット、信号、またはピンには設定できません。

AREA_GROUP の UCF 構文

エリア グループを定義する UCF 構文は次のとおりです。

INST "*X* " **AREA_GROUP=***groupname* ;

エリア グループに制約を適用する UCF 構文は次のとおりです。

AREA_GROUP "*groupname*" **RANGE=***range;*

AREA_GROUP "*groupname*" **COMPRESSION=***percent;*

AREA_GROUP "*groupname*" **GROUP=**{**OPEN**|**CLOSED**}**;**

AREA_GROUP "*groupname* "**PLACE=**{**OPEN**|**CLOSED**}**;**

groupname は、グループを定義するため区画に割り当てられた名前です。

次に、AREA_GROUP に指定する属性について説明します。

RANGE

LOC 制約を使用した範囲の定義と同様で、AREA_GROUP 内のロジックを配置するデバイス リソースの範囲を定義します。

FPGA デバイスの場合、RANGE は次のようになります。

RANGE=SLICE_X# Y#:SLICE_X#Y#

RANGE=RAMB16_X#Y#:RAMB16_X#Y#

RANGE=MULT18X18_X #Y#:MULT18X18_X#Y#

すべての FPGA デバイスで、SLICE がサポートされます。AREA_GROUP にブロック RAM および SLICE エレメントの両方が含まれている場合、一方を BRAM に、もう一方を SLICE というふうに、2 つの異なる AREA_GROUP RANGE コンポーネントを指定できます。

FPGA デバイスの位置はすべて X 座標または Y 座標で指定されます。ワイルドカード文字は X 座標または Y 座標のいずれかに対して使用できます。

RANGE の値は、CLOCK REGION エレメントまたは複数の CLOCK REGION エレメントのセットとしても指定できます。この構文は、AREA_GROUP 制約で使用されるすべての INIT タイプでサポートされます。

FPGA デバイスの場合、AREA_GROUP は次のようなさまざまなクロック領域でサポートされます。

1 つの領域
AREA_GROUP "*groupname*" RANGE=CLOCKREGION_X#Y#;
長方形を形成するクロック領域の範囲
AREA_GROUP "*group_name*" RANGE=CLOCKREGION_X#Y#:CLOCKREGION_X#Y#;
クロック領域のリスト
AREA_GROUP "*groupname*" RANGE=CLOCKREGION_X#Y#,CLOCKREGION_X#Y#,...,;

X# および Y# の値は、デバイスによって異なります。

IOB および BUF 以外のコンポーネントにはすべて CLOCKREGION 範囲の制約を付けることができます。

カンマ区切りの RANGE 指定

1 行に RANGE をカンマで区切って指定することはできません。この場合、2 つ目の範囲が無視されます。RANGE は 1 行ごとに指定する必要があります。

次は、無効な構文例です。

```
INST "RM_data_control" AREA_GROUP = "RR_RM_data_control" ;
AREA_GROUP "RR_RM_data_control" RANGE = SLICE_X0Y44:SLICE_X27Y20, DSP48_X0Y25:DSP48_X0Y14;
```

次は、有効な構文例です。

```
INST "RM_data_control" AREA_GROUP = "RR_RM_data_control" ;
AREA_GROUP "RR_RM_data_control" RANGE = SLICE_X0Y44:SLICE_X27Y20;
AREA_GROUP "RR_RM_data_control" RANGE = DSP48_X0Y25:DSP48_X0Y14;
```

RANGE 制約の有効なロジック タイプ

範囲名	virtex4	virtex5	virtex6	spartan6
BSCAN_XnYn	X	X	X	X
BUFDS_XnYn		X		X

範囲名	virtex4	virtex5	virtex6	spartan6
BUFGCTRL_XnYn	X	X	X	
BUFGMUX_XnYn				X
BUFHCE_XnYn			X	
BUFH_XnYn				X
BUFIO2FB_XnYn				X
BUFIO2_XnYn				X
BUFIODQS_XnYn			X	
BUFIO_XnYn	X	X		
BUFO_XnYn			X	
BUFPLL_MCB_XnYn				X
BUFPLL_XnYn				X
BUFR_XnYn	X	X	X	
CAPTURE_XnYn			X	
CFG_IO_ACCESS_XnYn			X	
CRC32_XnYn		X		
CRC64_XnYn		X		
DCIRESET_XnYn			X	
DCI_XnYn	X	X	X	
DCM_ADV_XnYn	X	X		
DCM_XnYn				X
DNA_PORT_XnYn			X	
DPM_XnYn	X			
DSP48_XnYn	X	X	X	X
EFUSE_USR_XnYn			X	
EMAC_XnYn	X			
FIFO16_XnYn	X			
GLOBALSIG_XnYn	X	X	X	
GT11CLK_XnYn	X			
GT11_XnYn	X			
GTPA1_DUAL_XnYn				X
GTP_DUAL_XnYn		X		
GTXE1_XnYn			X	
GTX_DUAL_XnYn		X		
IBUFDS_GTXE1_XnYn			X	
ICAP_XnYn	X	X	X	X
IDELAYCTRL_XnYn	X	X	X	
ILOGIC_XnYn	X	X	X	X
IOB_XnYn	X	X	X	

範囲名	virtex4	virtex5	virtex6	spartan6
IODELAY_XnYn		X	X	X
IPAD_XnYn	X	X	X	X
MCB_XnYn				X
MMCM_ADV_XnYn			X	
MONITOR_XnYn	X			
OCT_CAL_XnYn				X
OLOGIC_XnYn	X	X	X	X
OPAD_XnYn	X	X	X	X
PCIE_XnYn		X	X	X
PCILOGIC_XnYn				X
PLL_ADV_XnYn		X		X
PMCD_XnYn	X			
PMVBRAM_XnYn		X	X	
PMVIOB_XnYn			X	
PMV_XnYn			X	
PPC405_ADV_XnYn	X			
PPC440_XnYn		X		
PPR_FRAME_XnYn			X	
RAMB16_XnYn	X			X
RAMB18_XnYn			X	
RAMB36_XnYn		X	X	
RAMB8_XnYn				X
SLICE_XnYn	X	X	X	X
STARTUP_XnYn			X	
SYSMON_XnYn		X	X	
TEMAC_XnYn		X	X	
TIEOFF_XnYn	X	X	X	X
USR_ACCESS_XnYn			X	

標準の X、Y 形式に従わないサイト

次は、標準の X、Y 形式に従わないサイトで、AREA_GROUP の RAMGE で使用できます。構文は次のとおりです。

```
AREA_GROUP "group" RANGE=site1;   AREA_GROUP "group" RANGE=site2;
```

サイト名	virtex4	virtex5	virtex6	spartan6
CAPTURE	X	X		
DCIRESET	X	X		
DNA_PORT				X
EFUSE_USR		X		
FRAME_ECC	X	X	X	
JTAGPPC	X	X		
KEY_CLEAR		X		
PAD				X
PMV	X	X		X
POST_CRC_INTERNAL				X
SLAVE_SPI				X
SPI_ACCESS				X
STARTUP	X	X		X
SUSPEND_SYNC				X
USR_ACCESS_SITE	X	X		

COMPRESSION

COMPRESSION は、AREA_GROUP 制約の圧縮係数を定義します。0 から 100 までの値を使用できます。グループに範囲を指定しない場合、使用できる値は 0 (圧縮なし) および 1 (最大圧縮) のみです。マップは RANGE を使用して AREA_GROUP 内の CLB コンポーネント数を計算し、指定されたパーセンテージでロジックを圧縮します。BRAM または DSP ブロック/乗算器には、圧縮は適用されません。

圧縮係数はマップの **-c** オプションに似ていますが、この係数はデザイン全体に有効ではなく、AREA_GROUP に有効です。AREA_GROUP の圧縮とマップの **-c** オプションの関係は次のとおりです。

- 圧縮係数が定義されているエリア グループは、**-c** オプションの影響を受けません(圧縮係数が定義されたエリア グループの場合、その AREA_GROUP に含まれないロジックはグループ化されたロジックにマージされません)。
- 圧縮係数が定義されていないエリア グループは、**-c** オプションの影響を受けます。マップは、圧縮係数が定義されていないエリア グループの一部であるロジックを、グループ化されていないロジックにマージしようとします。
- 2 つの異なるエリア グループのロジックをマージすることはありません。
- **-c** マップ オプションを使用しても、エリア グループ内のスライスの圧縮を強制的に実行できません。

マップ レポート (MRP) には、処理された AREA_GROUP の要約が記載されます。

エリア グループの一部となっているシンボルに LOC 制約を設定すると、そのシンボルはマップでエリア グループから削除され、LOC 制約が処理されます。

AREA_GROUP に含まれないロジックは、エリア グループに含まれるロジック (スライスを形成するロジックにパックまたはマージされるロジック) の領域に移動する場合があります。

AREA_GROUP 制約の COMPRESSION は、マップでタイミング ドリブン パックと配置 (-timing オプション) を使用している場合は使用できません。

AREA_GROUP 制約の COMPRESSION は、Virtex®-5 ではサポートされません。

GROUP

スライスなどロジックの物理コンポーネントへのパックを制御します。

- CLOSED

  エリア グループ外のロジックがエリア グループ内のロジックと共にパックされないようになります。

- OPEN

  エリア グループ外のロジックがエリア グループ内のロジックと共にパックされるようになります。

  デフォルト値は、GROUP=OPEN です。

PLACE

エリア グループの範囲内でのリソースの割り当てを制御します。

- CLOSED

  エリア グループ以外のロジックがエリア グループの範囲内に配置されません。

- OPEN

  エリア グループ以外のロジックがエリア グループの範囲内に配置されません。

  デフォルト値は、PLACE=OPENです。

AREA_GROUP の構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図の例

- AREA_GROUP=groupname を有効なインスタンスに設定します。
- RANGE=range を CONFIG シンボルに設定します。
- COMPRESSION=percent を CONFIG シンボルに設定します。
- GROUP={OPEN|CLOSED} を CONFIG シンボルに設定します。
- PLACE={OPEN|CLOSED} を CONFIG シンボルに設定します。
- CONFIG シンボルを設定します。

  TRUE、PLACE、GROUP の値は、CLOSED に指定する必要があります。

- 属性名 :
  - **AREA_GROUP**
  - **RANGE** *range*
  - **COMPRESSION** *percent*
  - **GROUP={OPEN|CLOSED}**
  - **PLACE={OPEN|CLOSED}**
- 属性値 :
  - groupname
  - 範囲
  - percent
  - **GROUP={OPEN|CLOSED}**
  - **PLACE={OPEN|CLOSED}**

UCF および NCF の構文例

```
INST "state_machine_X" AREA_GROUP=group1;AREA_GROUP "group1"RANGE=SLICE_X1Y1:SLICE_X10Y10;
```

- **state_machine_X** のすべての論理ブロックをエリア グループ **group1** に割り当てます。
- 次の間の物理エリアにロジックを配置します。
  - **SLICE row 1, column 1**

    および
  - **SLICE row 10, column 10**

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

Constraints Editor の設定

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

タイミング グループによる定義

タイミング グループを基にしたエリア グループを作成するには、次の UCF/NCF 構文を使用します。

TIMEGRP *timing_group_name* **AREA_GROUP =** *area_group_name* **;**

- timing_group_name

  定義済みのタイミング グループ名です。
- area_group_name

  そのタイミング グループを基にして定義するエリア グループ名です。

これは、タイミング グループの各メンバーに *area_group_name* を手動で割り当てることと同じです。この構文で定義されたエリア グループ名は、ほかのエリア グループ名と同様、RANGE 制約で使用できます。

TNM_NET グループ

エリア グループを定義する場合、通常はタイミング グループ (timing_group_name) 名を TNM_NET グループとして使用し、クロックのロードやその他の制御ネットを基にしてエリア グループを作成します。TIMEGRP 制約から AREA_GROUP を定義すると、クロック領域よりも多くのクロックを使用するデバイスで、異なるクロック ドメインを使用したデザインの配置を改善できます。

TNM および TIMEGRP グループ

次のいずれかも指定できます。

- TNM のグループ名

  または

- TIMEGRP 文で定義されたユーザー グループの名前

ただし、TIMEGRP で使用されるエッジ識別子は、エリア グループのメンバーを識別する際に無視されます。いずれの場合にも、エリア グループのメンバーは、タイミング グループがターゲット エレメントへ伝搬された後で識別されます。

TIMEGRP 制約には、同期エレメントとパッドのみを含めることができます。タイム グループから定義されたエリア グループにもこれらのエレメント タイプを含めることができます。AREA_GROUP がフリップフロップまたはラッチだけを含む TIMEGRP により定義されていると、そのエリア内でグループにまとめられていないロジックも使用できる場合にのみ、グループに RANGE を設定すると便利です。このため、このようなグループには COMPRESSION は定義しないようにしてください。

PERIOD 仕様

TNM_NET が PERIOD 仕様により使用され、DCM、PLL または MMCM にトレースされる場合、新しい TNM_NET グループと PERIOD 仕様が DCM、PLL または MMCM 出力で作成されます。エリア グループの定義にオリジナルの TNM_NET を使用し、DCM、PLL、または MMCM で複数のクロック タップを使用した場合、エリア グループはクロック タップごとに異なるグループに分割されます。

たとえば、次のような UCF 構文があるとします。

```
NET "clk" TNM_NET="clock";

TIMESPEC "TS_clk" = PERIOD "clock" 10 MHz;

TIMEGRP "clock" AREA_GROUP="clock_area";
```

ネット **clk** が DCM、PLL、MMCM へトレースされると、次が実行されます。

- 新しいグループと PERIOD 仕様が各クロック タップで作成されます。
- 各クロック タップで新しいエリア グループが作成されます。

クロック タップ名を示す接尾辞が付けられます。CLK0 または CLK2X タップが使用された場合、AREA_GROUP の clock_area_CLK0 および clock_area_CLK2X が自動的に定義されます。

このようにしてエリア グループの定義が分割されると、NGDBuild により新しいグループ名を示すメッセージが表示されます。元から指定されていた名前の代わりに、RANGE 制約でこれらの新しいグループ名を使用してください。

ASYNC_REG

ASYNC_REG (非同期レジスタ) 制約には、次の特徴があります。

- タイミング制約です。
- 非同期クロックをソースとしたデータのビヘイビアーをシミュレーション用に向上します。
- タイミング シミュレーション中に X が適用されないようにします。
  タイミング違反があった場合、不定値とはならず、以前の値が出力で保持されます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

ASYNC_REG (非同期レジスタ) 制約には、次が適用できます。

- この制約は、レジスタおよびラッチにのみ適用できます。
- 非同期入力 (D 入力または CE 入力) 付きのレジスタおよびラッチのみに設定できます。

適用ルール

設定されたレジスタまたはラッチに適用されます。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute ASYNC_REG  : string;
```

VHDL 制約を次のように指定します。

```
attribute ASYNC_REG of instance_name: label is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* ASYNC_REG = " {TRUE|FALSE}" *)
```

UCF および NCF 構文

```
INST "instance_name" ASYNC_REG =  {TRUE|FALSE};
```

デフォルトは FALSE です。

ブール属性値を使用しない場合、TRUE とみなされます。

Constraints Editor の設定

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

BEL

BEL 制約には次の特徴があります。

- 高度な配置制約です。
- 論理シンボルを次の BEL サイトにロックします。
  - スライス

    または
  - IOB
- コンポーネント レベルに指定するLOC 制約とは異なります。コンポーネント例には、次が含まれます。
  - SLICE
  - BRAM
  - ILOGIC
  - OLOGIC
  - IOB
- コンポーネントの使用される特定の BEL サイトまで指定できます。たとえば、特定の LUT または FF を SLICE 内で使用されるように指定できます。
- 常に LOC または RLOC 属性を使用する必要があります。

BEL 制約を IOB に設定すると、MAP によりレジスタが IOB コンポーネントにパックされないため、**-pr** オプションなどほかの機能を使用してパックする必要があります。レジスタが IOB にパックされると、BEL 制約により IOB 内で適正に配置されます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

レジスタ	ラッチ
LUT LUTRAM RAMB18	SRL

適用ルール

BEL 制約は有効な LOC または RLOC の付いたインスタンスにのみ付けることができます。

制約値

値	スライス内での識別
F、G	特定の LUT、SRL16、および分散 RAM コンポーネント
A6LUT、B6LUT、C6LUT、D6LUT	
A5LUT、B5LUT、C5LUT、D5LUT	
FFA、FFB、FFC、FFD、FFX、FFY	特定のフリップフロップ、ラッチ、およびその他のエレメント
XORF、XORG	XORCY エレメント

Virtex®-6 デバイスの場合は、次の値も設定できます。

- AFF
- BFF
- CFF
- DFF
- A5FF
- B5FF
- C5FF
- D5FF

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  BEL

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* BEL = " {value}" *)
```

UCF および NCF 構文

```
INST "instance_name " BEL={value};
```

RAMB BEL インスタンスの構文は、次のようになります。

```
INST "upper_BRAM_instance_name" LOC = RAMB36_XnYn|BEL = UPPER;
INST "lower_BRAM_instance_name" LOC = RAMB36_XnYn|BEL = LOWER;
```

UCF および NCF の構文例

```
INST "ramb18_inst0" LOC = RAMB36_X0Y2|BEL = UPPER; INST "ramb18_inst1" LOC = RAMB36_X0Y2|BEL = LOWER;
```

次の文は、スライス内の FFX サイトに xyzzy をロックするよう指定しています。

```
INST "xyzzy" BEL=FFX;
```

PlanAhead™ の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

BLKNM

BLKNM (ブロック名) 制約には、次の特徴があります。

- 高度なマップ制約です。
- ブロック名を適切なプリミティブおよびロジック エレメントに割り当てます。

同じ BLKNM を複数のインスタンスに割り当てた場合、ソフトウェアはそれらを同じブロックにマップしようと試みます。逆に、異なるBLKNM 名を持つ 2 つのシンボルが同じブロックにマップされることはありません。複数の類似した BLKNM 制約を、1 つのブロック内に収まらない複数のインスタンスに設定すると、エラーが発生します。

FMAP に同じ BLKNM を複数設定すると、関連するファンクション ジェネレータが 1 つのスライスにまとめられます。BLKNM を使用すると、スライスをデバイス上の物理的な位置に制約することなく、スライスを分割できます。

LOC 制約と同様、BLKNM はデザイン内で指定します。階層パスが接頭辞として割り当てられないため、デザイン内でそれぞれの SLICE に固有の BLKNM を設定する必要があります。

階層ブロック名の割り当てについては、「階層ブロック名 (HBLKNM)」を参照してください。

BLKNM を使用すると、別の BLKNM が指定されているエレメントを除き、すべてのエレメントを同じ物理的コンポーネントにマップできます。BLKNM が指定されていないエレメントは、BLKNM が指定されているエレメントとともにパックできます。

同じ XBLKNM を設定したエレメントだけを 1 つの物理コンポーネントにマップする場合は、「XBLKNM」を参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

この制約は、次のエレメントまたはデザイン エレメントのカテゴリを 1 つまたは複数使用して設定できます。すべてのデバイスですべてのエレメントがサポートされるわけではありません。デバイス別にどのデザイン エレメントが使用可能かを確認するには、そのデバイスのライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

- フリップフロップおよびラッチ プリミティブ
- すべての I/O エレメントまたはパッド
- FMAP
- ROM プリミティブ
- RAMS および RAMD プリミティブ
- キャリー ロジック プリミティブ
- ブロック RAM

BLKNM は、UCF ファイル内のパッド コンポーネントに接続されているネットにも設定できます。

ネットに設定した制約は、NGDBuild により NGD ファイル内のパッド インスタンスに挿入され、マップで処理されます。

次の構文を使用してください。

```
NET "net_name" BLKNM=property_value;
```

適用ルール

デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

制約値

block_name

そのシンボル タイプに対して有効なブロック名

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  BLKNM
- 属性値

  block_name

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute blknm : string;
```

VHDL 制約を次のように指定します。

```
attribute blknm of {component_name|signal_name|entity_name|label_name}: {component|signal|entity|label}
is "block_name";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* BLKNM = "blk_name" *)
```

UCF および NCF 構文

```
INST "instance_name" BLKNM=block_name;
```

階層ブロック名の割り当てについては、「HBLKNM」を参照してください。

次の文は、エレメント block1 のインスタンスをブロック U1358 に割り当てます。

```
INST "$1I87/block1" BLKNM=U1358;
```

XCF 構文

```
MODEL "entity_name" blknm = block_name;
 BEGIN MODEL "entity_name"
 INST "instance_name" blknm = block_name;
END;
```

BUFG

BUFG 制約には、次の特徴があります。

- 高度なフィッタ制約です。
- 合成制約です。
- 次に BUFG を適用すると、指定した信号がグローバル ネットにマップされます。
  - 入力バッファー

    または
  - 入力パッド ネット

内部ネットに設定した場合、指定された信号には次のいずれかが実行されます。

- 直接グローバルネットに配線される

  または
- グローバル ネットを駆動するためグローバル制御ピンに接続される

アーキテクチャ サポート

CPLD デバイスのみサポートされます。FPGA デバイスのサポートなし

適用可能エレメント

BUFG 制約は、次のいずれかに適用できます。

- 入力バッファー (IBUF)
- 入力パッド ネット
- CLK、OE、SR、DATA_GATE ピンを駆動する内部ネット

適用ルール

- ネットに使用されると、BUFG 制約はネットまたは信号形式になります。特別な適用ルールはありません。
- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

制約値

- CLK
  - グローバル クロック ピンを指定します。
  - すべての CPLD デバイスでサポートされます。
- OE

  次のいずれかを指定します。
  - グローバル トライステート制御ピン

    CoolRunner™-II および CoolRunner XPLA3 デバイスを除く CPLD デバイスすべてでサポートされます。
  - 内部グローバル トライステート制御ラインを指定します。

    CoolRunner-II デバイスのみをサポート
- SR
  - グローバル セット/リセット ピンを指定します。
  - CoolRunner-II および CoolRunner XPLA3 デバイスを除く CPLD デバイスすべてでサポートされます。
- DATA_GATE

  DATA_GATE ラッチ イネーブル制御線に割り当てます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- IBUF 入力に接続されている入力パッドの IBUF インスタンスに設定します。
- 属性名

  BUFG

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute BUFG: string;
```

VHDL 制約を次のように指定します。

```
attribute BUFG of signal_name : signal  is "{CLK|OE|SR|DATA_GATE} ";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* BUFG = "{CLK|OE|SR|DATA_GATE}" *)
```

UCF および NCF 構文

```
NET "net_name " BUFG={CLK|OE|SR|DATA_GATE};
```

```
INST "instance_name" BUFG={CLK|OE|SR|DATA_GATE};
```

次の文により、信号 **fastclk** がグローバル クロック ネットに割り当てられます。

```
NET “fastclk” BUFG=CLK;
```

XCF 構文

```
BEGIN MODEL “entity_name”
 NET "signal_name" BUFG = {CLK|OE|SR|DATA_GATE} ;
END;
```

CLOCK_DEDICATED_ROUTE

CLOCK_DEDICATED_ROUTE (クロック専用配線) 制約には、次の特徴があります。

- 高度な制約です。
- アーキテクチャ別にクロック配置規則に従うかどうかを指定します。

クロック配置規則

CLOCK_DEDICATED_ROUTE 制約が次のような場合、クロック配置規則に必ず従う必要があります。

- この制約が使用されていない場合、または
- TRUE に設定されている場合

従わない場合は、配置でエラーになりますす。

CLOCK_DEDICATED_ROUTE 制約が FALSE に設定されていると、ソフトウェアで次が実行されます。

- 特定のクロック配置規則は無視されます。
- 配置配線が続行されます。

可能であれば、すべてのクロック配置規則違反を修正し、最適なクロック パフォーマンスを達成できるようにします。この制約は、クロック配置規則に違反することが絶対に必要な場合にのみ使用してください。

特定のクロック配置規則の詳細は、『ハードウェア ユーザー ガイド』を参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

この制約は次に適用されます。

- クロック バッファー
- クロック マネージャー ブロック
- 高速 I/O ブロック

適用ルール

ネットまたはインスタンス ピンに設定できます。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  CLOCK_DEDICATED_ROUTE
- 属性値

  上記の「制約値」を参照してください。

UCF および NCF 構文

```
PIN "BEL_INSTANCE_NAME.PIN" CLOCK_DEDICATED_ROUTE = {TRUE|FALSE};
```

BEL_INSTANCE_NAME.PIN は制約を付ける特定のインスタンスの入力/出力ピン (例 : DCM インスタンスの CLKIN 入力ピン) です。

COLLAPSE

COLLAPSE 制約には、次の特徴があります。

- 高度なフィッタ制約です。
- 組み合わせノードがすべてのファンアウトにコラプスします。

アーキテクチャ サポート

CPLD デバイスのみサポートされます。FPGA デバイスのサポートなし

適用可能エレメント

すべての内部ネットに適用されます。

適用ルール

この制約はネット制約です。デザイン エレメントへは接続できません。

制約値

- YES
- NO
- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ロジック シンボルまたはその出力ネットに設定します。
- 属性名

  COLLAPSE
- 属性値
  - TRUE
  - FALSE

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute collapse: string;
```

VHDL 制約を次のように指定します。

```
attribute collapse of signal_name: signal is
"{YES|NO|TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* COLLAPSE = "{YES|NO|TRUE|FALSE}" *)
```

UCF および NCF 構文

```
NET "net_name" COLLAPSE;
```

次の文は、ネット $1N6745 をすべてのファンアウトにコラプスします。

```
NET "$1I87/$1N6745" COLLAPSE;
```

COMPGRP

COMPGRP (コンポーネント グループ) 制約には、次の特徴があります。

- 高度なグループ制約です。
- コンポーネントのグループを識別します。
- PCF にのみ設定できます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

コンポーネントのグループに適用します。

適用ルール

なし

制約値

comp_item は、次のいずれかです。

- **COMP** "*comp_name*"
- **COMPGRP** "*group_name*"

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

PCF 構文

```
COMPGRP "group_name"=comp_item1... comp_itemn [EXCEPT comp_group];
```

CONFIG_MODE

コンフィギュレーション モード制約を使用すると、どの多目的コンフィギュレーション ピンをを汎用 I/O として使用するかを PAR に伝えることができます。

PAR では CONFIG_MODE を使用して、必要な場合は次の多目的 I/O コンポーネントの使用を禁止します。

- S_SELECTMAP+READBACK
- M_SELECTMAP+READBACK

これらの I/O がリードバックおよびリコンフィギュレーションに使用されないようにするには **bitgen -g Persist** オプションを使用します。

次の場合、PAR では多目的 I/O コンポーネントが必要な場合にのみ汎用 I/O コンポーネントとして使用されます。

- CONFIG_MODE :S_SELECTMAP
- M_SELECTMAP

アーキテクチャ サポート

この制約は、次のデバイスでサポートされます。

- Spartan®-3
- Virtex®-4
- Virtex-5
- Virtex-6
- 7 series

適用可能エレメント

CONFIG シンボルに適用します。

適用ルール

多目的 I/O コンポーネントに適用されます。

制約値

次の文字列になります。

- M_SERIAL

  マスター シリアル モード (デフォルト値)

- S_SERIAL

  スレーブ シリアル モード

- B_SCAN

  バウンダリ スキャン モード

- B_SCAN+READBACK

  Persist の使用が予測されるバウンダリ スキャン モード

- M_SELECTMAP

  マスタ SelectMAP モード、8 ビット幅

- M_SELECTMAP+READBACK

  マスタ SelectMAP モード、8 ビット幅、Persist の使用が予測される

- S_SELECTMAP

  スレーブ SelectMAP モード、8 ビット幅

- S_SELECTMAP+READBACK

  スレーブ SelectMAP モード、8 ビット幅、Persist の使用が予測される

- S_SELECTMAP16

  スレーブ SelectMAP モード、16 ビット幅

- S_SELECTMAP16+READBACK

  スレーブ SelectMAP モード、16 ビット幅、Persist の使用が予測される

- S_SELECTMAP32

  スレーブ SelectMAP モード、32 ビット幅

- S_SELECTMAP32+READBACK

  スレーブ SelectMAP モード、Persist の使用が予測される

  S_SELECTMAP32 および S_SELECTMAP32+READBACK については、Virtex-5 デバイスの場合に S_SELECTMAP16 および S_SELECTMAP16+READBACK を選択すると、コンフィギュレーション後も維持する必要のあるデータ ピンの数が正しく設定できます。

7 シリーズの場合のみ、次の値が適用できます。

- SPIx1
- シリアル ペリフェラル インターフェイス、1 ビット幅
- SPIx2

  シリアル ペリフェラル インターフェイス、2 ビット幅
- SPIx4

  シリアル ペリフェラル インターフェイス、4 ビット幅
- BPI8

  バイト ペリフェラル インターフェイス (Parallel NOR)、8 ビット幅
- BPI16

  バイト ペリフェラル インターフェイス (Parallel NOR)、8 ビット幅

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG CONFIG_MODE=string;
```

COOL_CLK

CoolCLOCK (COOL_CLK) 制約には、次のような特徴があります。

- CoolRunner™-II デバイスでのみサポートされます。
- クロック分周器と DualEDGE 回路を組み合わせることで CPLD デバイス内のクロック電力を削減できます。

クロック電力の削減

クロック ネットは、かなりの電力を消費します。クロックの電力を抑えるには、次を実行します。

1. ネットを半分の周波数で駆動
2. DualEDGE で動作するマクロセルを使用してクロック レートを倍増

アーキテクチャ サポート

CoolRunner-II

適用可能エレメント

入力パッドまたはレジスタ クロックを駆動する内部信号に適用できます。

適用ルール

- クロック ネットに COOL_CLK を設定すると、次を実行したのと同じようになります。
  - 2 分周のクロック分周器 (CLK_DIV2) を介してクロックを渡します。
  - そのクロックで制御されるフリップフロップをすべて DualEDGE フリップフロップに置き換えます。
- COOL_CLK 属性を使用しても、デザイン全体の機能は変更できません。
- 立ち下がりエッジでフリップフロップを動作させるクロックに COOL_CLK は使用できません。CoolRunner™-II のクロック分周器は、クロック信号の立ち上がりエッジのみで動作します。
- デザイン ソースに DualEDGE フリップフロップがある場合、そのフリップフロップを制御するクロックは COOL_CLK として指定できません。
- デザイン ソースにクロック分周器が既にある場合は、COOL_CLK を使用できません。CoolRunner-II デバイスには、クロック分周器を 1 つのみ使用できます。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 入力パッドまたはレジスタ クロックを駆動する内部信号に適用できます。
- 属性名

  COOL_CLK
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute cool_clk: string;
```

VHDL 制約を次のように指定します。

```
attribute cool_clk of signal_name: signal is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* COOL_CLK = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

```
NET "signal_name" COOL_CLK;
```

DATA_GATE

DATA_GATE (データ ゲート) 制約には、次のような特徴があります。

- CoolRunner™-II デバイスのみがサポートされます。
- 消費電力を低減することができます。

電力消費の削減

各 I/O ピンはラッチを介して入力信号を送信します。デザインに関係のない遷移が発生した場合、このラッチで信号の遷移が伝搬されないようブロックすることができます。ラッチを通過しない場合、入力信号はデバイスのファンクション ブロック内で配線されるため、信号遷移がデザインの機能に影響を与えなくても電力が消費されます。

デバイスに DATA_GATE 制御の I/O ピンを配置すると、次のようになります。

- 選択した I/O ピンの入力にラッチが付けられます。
- それらのピンの外部信号遷移に伴う電力の消費が削減されます。

アーキテクチャ サポート

128 個以上のマクロセルを使用した CoolRunner-II デバイスにのみ適用できます。

適用可能エレメント

I/O パッドおよびピンに適用します。

適用ルール

I/O パッドに DATA_GATE 属性を設定すると、そのデバイス ピンに付けられた通過ラッチが DATA_GATE 制御ピンに反応します。

- クロック入力パッドを含む I/O パッド (DATA_GATE 制御の I/O ピンを除く) は、DATA_GATE 属性を使用してラッチを付けるようにコンフィギュレーションできます。
- DATA_GATE 属性が設定されていないその他の I/O パッドは常に、ラッチが付いていない状態になります。

DATA_GATE 制御信号自体は、次のように処理されます。

- DATA_GATE 制御の I/O ピンを介してオフチップから受信されます。
- ラッチが付かないままの入力 (DATA_GATE 属性が設定されていないパッド) を基づいて生成されます。

VHDL および Verilog デザインでの DATA_GATE の使用については「BUFG」を参照してください。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- I/O パッドおよびピンに設定します。
- 属性名

  DATA_GATE
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute DATA_GATE : string;
```

VHDL 制約を次のように指定します。

```
attribute DATA_GATE of signal_name: signal is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* DATA_GATE = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

```
NET "signal_name" DATA_GATE;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" data_gate={TRUE|FALSE};
END ;
```

DCI_CASCADE

Virtex®-5 および Virtex-6 デバイス ファミリでは、DCI 参照電圧を必要とする IO バンクを、ほかの DCI IO バンクとカスケード接続でき、1 組の VRN/VRP ピンで、複数の IO バンクに参照電圧を供給できます。この方法では使用される VR ピン数および DCI コントローラ数が少なくて済むため、使用可能なピン数が増え、消費電力量が減少します。DCI_CASCADE (DCI カスケード) 制約では、DCI マスタ バンクと対応するスレーブ バンクを指定します。複数のマスタ/スレーブの組み合わせを指定する場合は、デザイン内にこの制約のインスタンスが複数含まれます。この制約で指定した情報を基に BitGen で DCI コントローラがプログラムされ、バンクがカスケード接続されます。配置の際にもこの情報が使用され、スレーブ バンクの VR ピンをほかの用途に使用できるかを調べます。

DCI_CASCADE 制約の各インスタンスには、マスタ バンクが 1 つと、少なくとも 1 つのスレーブ バンクが含まれている必要があります。バンク間は、スペースで区切ります。最初に指定される値はマスタ バンクで、それ以降の値はスレーブ バンクです。スレーブ バンクには、マスタ バンクから DCI 参照電圧が供給されます。この制限は、Virtex-6 デバイスには該当しません。カスケード接続されたバンクは、同じ列にあり (左、中央、または右) VCCO 設定が同じである必要があります。詳細は、「UCF および NCF 構文」を参照してください。

アーキテクチャ サポート

Virtex-5 および Virtex-6 デバイスにのみ適用できます。

適用可能エレメント

最上位デザイン ブロックの DCI_CASCADE 属性。

適用ルール

CONFIG ブロックに属性として配置され、物理デザイン オブジェクトに伝播します。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

CONFIG DCI_CASCADE = "<*master*> <*slave1*> <*slave2*> ...**";**

説明 :

- <*master*> = [1...MAX_NUM_BANKS]
- <*slave1*> = [1...MAX_NUM_BANKS]
- <*slave2*> = [1...MAX_NUM_BANKS]
- 値はすべて、Virtex-5 デバイスで有効な IO バンクです。
- マスタ バンクには、DCI 参照電圧を必要とする IO 規格が適用された IOB が必要です。この制限は、Virtex-6 デバイスには該当しません。
- すべてのスレーブ バンクの VCCO 設定は、マスタ バンクと同じである必要があります。
- マスタとスレーブ間にその他のバンクが存在する場合は、正しい方向にカスケード接続されている必要があります。

例

CONFIG DCI_CASCADE = "11 13 15 17";

PCF 構文

CONFIG DCI_CASCADE = "<*master*>, <*slave1*>, <*slave2*>, ..."

説明 :

- <*master*> = [1...MAX_NUM_BANKS]
- <*slave1*> = [1...MAX_NUM_BANKS]
- <*slave2*> = [1...MAX_NUM_BANKS]

DCI_VALUE

DCI_VALUE (DCI 値) は、IBISWriter を使用して IBS ファイルを作成する際に、どのバッファーのビヘイビアー モデルがデザインの IOB に関連するかを指定します。

アーキテクチャ サポート

- Virtex®-4
- Virtex-5
- Virtex-6
- Spartan®-3

適用可能エレメント

この制約は IOB コンポーネントに適用します。

適用ルール

それが接続されている IOB に適用

制約値

integer

- integer の値は、25 ～ 100 (単位は Ω) に指定できます。
- デフォルト値は、50 オームです。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

```
INST pin_name DCI_VALUE = integer;
```

DEFAULT

この制約を使用すると、複数の制約に新しいデフォルト値を設定できます。

特定の制約を指定すると、DEFAULT 制約値が上書きされます。

アーキテクチャ サポート

DEFAULT 制約は、次の制約およびアーキテクチャに適用されます。

- KEEPER、PULLDOWN

  すべての FPGA および CoolRunner™-II CPLD デバイスにのみ適用されます。

- PULLUP

  すべての FPGA および CPLD デバイスの CoolRunner XPLA3 および CoolRunner-II に適用されます。

適用可能エレメント

DEFAULT でサポートされる制約に適用可能なエレメントについては、次を参照してください。

- KEEPER
- FLOAT
- PULLDOWN
- PULLUP

適用ルール

DEFAULT でサポートされる制約の適用ルールについては、次を参照してください。

- KEEPER
- FLOAT
- PULLDOWN
- PULLUP

制約値

- KEEPER
- FLOAT
- PULLDOWN
- PULLUP

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

DEFAULT 制約を回路図に適用する場合は、次の手順に従います。

- ネット、インスタンス、またはピンに設定します。
- 属性名

  DEFAULT *constraint_name*
- 属性値

  constraint_name によって決まります。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute attribute_name : string;
attribute KEEPER: string;
```

VHDL 制約を次のように指定します。

```
attribute  attribute_name of DEFAULT is attribute_value;
```

attribute_name に使用可能な値については、上記の「制約値」を参照してください。

attribute_values に使用できる値は、次の例に示すように属性タイプによって異なります。

```
attribute of DEFAULT KEEPER is “TRUE”;
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(*  CONSTRAINT_NAME = "constrant_value" *) DEFAULT
```

constraint_name に使用可能な値については、上記の「制約値」を参照してください。

constraint_value は、大文字/小文字が区別されます。

constraint_values に使用できる値は、次の例に示すように *constraint_name* によって異なります。

```
(* KEEPER = “TRUE” *) DEFAULT
```

UCF 構文

```
DEFAULT  constraint_name;
```

UCF 構文例

```
DEFAULT KEEPER = TRUE;
```

XCF 構文

```
BEGIN MODEL "entity_name"
DEFAULT constraint_name [attribute_value] ;
END;
```

使用できる属性値 *attribute_values* は、属性のタイプによって異なります。

XCF の構文例

```
BEGIN MODEL "my_design"
 DEFAULT keeper = TRUE;
END ;
```

NCF 構文

UCF と同じ構文を使用します。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PACE の構文

Pinout and Area Constraints Editor (PACE) ツール

- CPLD でのみサポートされます。
- 主としてロケーション制約を I/O に設定する際に使用します。
- I/O 規格など特定の I/O プロパティの設定に使用できます。
- ISE® Design Suite の [Processes] ウィンドウからアクセスできます。

詳細は、PACE ヘルプの「手順」セクションのエリア制約およびピンの編集に関する部分を参照してください。

DIFF_TERM

DIFF_TERM 制約には、次の特徴があります。

- 基本的なマップ制約です。
- ビルトインの 100W 終端抵抗のオン/オフを切り替えるために使用します。

アーキテクチャ サポート

Virtex®-6、Spartan®-6 デバイスのみをサポート

適用可能エレメント

IBUFDS_DIFF_OUT のような差動 I/O ブロックに適用します。

適用ルール

なし

制約値

- TRUE

  ビルトインの 100W 終端抵抗をオンにします。
- FALSE

  ビルトインの 100W 終端抵抗をオフにします。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図構文

- 属性名

  DIFF_TERM
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
Attribute DIFF_TERM: string;
```

VHDL 制約を次のように指定します。

```
attribute DIFF_TERM of block_name: signal is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* DIFF_TERM  = "{TRUE|FALSE }" *)
```

UCF および NCF 構文

次の文では、ビルトインの 100 オーム終端抵抗を使用するように I/O を指定しています。

```
INST "IO block name" DIFF_TERM = "{TRUE|FALSE}" ;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "block_name" DIFF_term=true;
END;
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

DIRECTED_ROUTING

DIRECTED_ROUTING (指定配線) は、少数のロードおよびソースの配線とタイミングを維持するために使用する制約です。

この制約を使用する場合は、次の制約を使用してロードとソースの相対位置をまったく同じに維持しておく必要があります。

- LOC
- RLOC
- BEL

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

それが接続されているネットに適用

適用ルール

なし

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

次の例は、説明用のもので、有効な実行ファイルではありません。指定配線制約を使用する場合は、ロードとソース コンポーネントを相対的に配置する必要があります。

FPGA Editor の設定

FPGA Editor でこの制約を生成するには、次をクリックします。

[Tools] → [Directed Routing Constraints]

FPGA Editor では、次を設定し、ソースおよびロード コンポーネントに自動的にこのタイプの配置制約が生成されるようにします。

- [Do Not Generate Placement Constraint Setting]
- [Use Relative Location Constraint Setting]
- [Use Absolute Location Constraint Setting]

[Do Not Generate Placement Constraint Setting]

これを設定すると、次が実行されます。

- 配線に対してのみ制約が生成されます。
- 既存の RPM コンポーネントと一緒に使用されます。

```
NET "net_name" ROUTE="{2;1;-4!-1;-53320;2920;14;90;200;30;13!0;- 2091;1480;24!0;16;-8!}";
```

[Use Relative Location Constraint Setting]

これを設定すると、この配線制約と一緒にソースおよびロード コンポーネントに対する RPM コンポーネントが生成されます。

この RPM コンポーネントをデバイスの周囲に再配置しておくと、配置ツールで最終的な配置を実行させることができます。

この例の場合、次のようになります。

- 各 RLOC リファレンス信号により、新しいインスタンスの開始が示されます。
- この例には 3 つのインスタンスが含まれます。

```
NET "net_name" ROUTE="{2;1;-4!-1;-53320;2920;14;90;200;30;13!0;- 2091;1480;24!0;16;-8!}";
INST "inst1" RLOC=X3Y0;
INST "inst1" RPM_GRID=GRID;
INST "inst1" U_SET=macro name;
INST "inst1" BEL="F";
INST "inst2" RLOC=X3Y0;
INST "inst2" U_SET=macro name;
INST "inst2" BEL="G";
```

[Use Absolute Location Constraint Setting]

この設定をすると、ターゲット ネットに接続されたソースおよびロード コンポーネントが RLOC 制約と RLOC_ORIGIN 制約で指定した場所に固定されます。

また、手動で LOC 制約を指定することもできます。

```
NET "net_name" ROUTE="{2;1;-4!-1;-53320;2920;14;90;200;30;13!0;- 2091;1480;24!0;16;-8!}";
INST "inst1" RLOC=X3Y0;
INST "inst1" RPM_GRID=GRID;
INST "inst1" RLOC_ORIGIN=X87Y200;
INST "inst1" U_SET=macro name;
INST "inst1" BEL="F";
INST "inst2" RLOC=X0Y1;
INST "inst2" U_SET=macro name;
INST "inst2" BEL="F";
INST "inst3" RLOC=X3Y0;
INST "inst3" U_SET=macro name;
INST "inst3" BEL="G";
```

DISABLE

DISABLE 制約には、次の特徴があります。

- タイミング制約です。
- 特定のパストレース制御を無効にします。

パストレース制御は、よくあるタイプのパスがタイミング パス解析でイネーブルになるかディスエーブルになるかを決定するために使用されます。

パストレース制御文は、ソースがネットリスト、UCF、NCF のいずれの場合でも、PCF に出力されます。

ただし、ネットリストの DISABLE を UCF で ENABLE に置き換えることはできません。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

制約ファイルでグローバルに適用します。

適用ルール

指定したブロックの遅延シンボルに対してタイミング解析が実行されないようにします。

制約値

delay_symbol_name

パストレースの標準ブロックの遅延シンボル名、またはデータシートに表示される遅延名です。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

DISABLE=*delay_symbol_name*;

これらのシンボルについては、次の表を参照してください。PCF ではコンポーネント遅延名も使用できます。

パス トレースの標準ブロック遅延シンボル

遅延シンボル名	パス タイプ	デフォルト
reg_sr_o	非同期セット/リセットから出力への伝搬遅延	ディスエーブル
reg_sr_r	非同期セット/リセットからのリカバリ パス	Virtex®-5 およびそれ以前のアーキテクチャではディスエーブル Virtex-6 および Spartan®-6 アーキテクチャではイネーブル
reg_sr_clk	同期セット/リセットからクロックへの設定およびホールド チェック	イネーブル
lat_d_q	データから出力へのトランスペアレント ラッチ遅延	ディスエーブル
lat_ce_q	クロック イネーブルから出力へのトランスペアレント ラッチ遅延	ディスエーブル
ram_we_o	RAM 書き込みイネーブルから出力への伝搬遅延	イネーブル
io_pad_i	I/O パッドから入力への伝搬遅延	イネーブル
io_t_pad	I/O トライステートからパッドへの伝搬遅延	イネーブル
io_o_i	I/O 出力から入力への伝搬遅延。トライステート IOB ではオフ	イネーブル
io_o_pad	I/O 出力からパッドへの伝搬遅延	イネーブル

PCF 構文

UCF および NCF 構文と同じ

DRIVE

DRIVE 制約には、次の特徴があります。

- 基本的なマップ制約です。
- サポートされる FPGA アーキテクチャすべての駆動電流の出力を選択します。
- 次のインターフェイス I/O 規格を使用する SelectIO™ バッファーの出力の駆動電流 (mA) を選択します。
  - LVTTL
  - LVCMOS12
  - LVCMOS15
  - LVCMOS18
  - LVCMOS25
  - LVCMOS33

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

この制約は、次のエレメントまたはデザイン エレメントのカテゴリを 1 つまたは複数使用して設定できます。すべてのデバイスですべてのエレメントがサポートされるわけではありません。デバイス別にどのデザイン エレメントが使用可能かを確認するには、そのデバイスのライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

- IOB 出力コンポーネント (OBUF および OFD など)
- 次を使用する SelectIO 出力バッファー
  - IOSTANDARD = LVTTL
  - LVCMOS15
  - LVCMOS18
  - LVCMOS25または
  - LVCMOS33
- ネット

適用ルール

- ネットまたは信号がパッドに接続されている場合を除いて、ネットや信号に設定できません。この場合、DRIVE はパッド インスタンスに設定されているものと見なされます。
- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効な IOB 出力コンポーネントに設定します。
- 属性名

  DRIVE
- 属性値

  制約値のリストは、次の「UCF および NCF 構文」セクションを参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute drive: string;
```

VHDL 制約を次のように指定します。

```
attribute drive of {component_name|entity_name|label_name} : {component|entity|label} is "value";
```

制約値のリストは、次の「UCF および NCF 構文」セクションを参照してください。

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* DRIVE = "value" *)
```

制約値のリストは、次の「UCF および NCF 構文」セクションを参照してください。

UCF および NCF 構文

このセクションでは、UCF および NCF の例を示します。

- IOB 出力コンポーネント (UCF)
- SelectIO 出力コンポーネント

IOB 出力コンポーネント (UCF)

Spartan®-3 および Virtex®-4 以降のデバイスでは、次のように記述します。

```
INST "instance_name" DRIVE={2|4|6| 8|12|16 |24};
```

デフォルトでは 12mA に設定されています。

SelectIO 出力コンポーネント

このセクションは、次のコンポーネントに適用されます。

- IOBUF_SelectIO
- OBUF_SelectIO
- OBUFT_SelectIO

次は、Spartan-3 および Virtex-4 以降のデバイスを使用した場合の規格に対する構文例を示しています。それぞれのデフォルトは 12 mA です。

規格	構文
LVTTL	**INST** "*instance_name*" **DRIVE=**{**2**|**4**|**6**|**8**|**12**|**16**|**24**};
LVCMOS12 LVCMOS15 LVCMOS18	**INST** "*instance_name*" **DRIVE=**{**2**|**4**|**6**|**8**|**12**|**16**};
LVCMOS25 LVCMOS33	**INST** "*instance_name*" **DRIVE=**{**2**|**4**|**6**|**8**|**12**|**16**|**24**};

XCF 構文

```
MODEL "entity_name" drive={2|4|6|8|12|16|24};
 BEGIN MODEL "entity_name"
 NET "signal_name" drive={2|4|6|8|12|16|24};
END;
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

ENABLE

ENABLE 制約には、次の特徴があります。

- タイミング制約です。
- 特定のパストレース制御を有効にします。

パストレース制御は、よくあるタイプのパスがタイミング パス解析でイネーブルになるかディスエーブルになるかを決定するために使用されます。

詳細は、「DISABLE」を参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

制約ファイルでグローバルに適用します。

適用ルール

特定のパス遅延に対してタイミングが解析されます。

制約値

delay_symbol_name

- 次の表に示すパストレースの標準ブロックの遅延シンボル名です。

  または

- データシートに表示される特定の遅延名です。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

この制約は、グローバルな TIMESPEC 制約にのみ適用できます。

UCF のパストレース構文は次のようになります。

```
ENABLE= delay_symbol_name ;
```

パス トレースの標準ブロック遅延シンボル

遅延シンボル名	パス タイプ	デフォルト
reg_sr_o	非同期セット/リセットから出力への伝搬遅延	ディスエーブル
reg_sr_r	非同期セット/リセットからのリカバリ パス	Virtex®-5 およびそれ以前のアーキテクチャではディスエーブル Virtex-6 および Spartan®-6 アーキテクチャではイネーブル
reg_sr_clk	同期セット/リセットからクロックへの設定およびホールド チェック	イネーブル
lat_d_q	データから出力へのトランスペアレント ラッチ遅延	ディスエーブル
lat_ce_q	クロック イネーブルから出力へのトランスペアレント ラッチ遅延	ディスエーブル
ram_we_o	RAM 書き込みイネーブルから出力への伝搬遅延	イネーブル
io_pad_i	I/O パッドから入力への伝搬遅延	イネーブル
io_t_pad	I/O トライステートからパッドへの伝搬遅延	イネーブル
io_o_1	I/O 出力から入力への伝搬遅延。トライステート IOB ではオフ	イネーブル
io_o_pad	I/O 出力からパッドへの伝搬遅延	イネーブル

PCF 構文

ENABLE=*delay_symbol_name* ;

TIMEGRP *name* **ENABLE=***delay_symbol_name* ;

ENABLE_SUSPEND

ENABLE_SUSPEND 制約には、次の特徴があります。

- Spartan®-3A、Spartan-6 デバイスでのみサポートサポートされます。
- UCF にのみ設定できます。
- 電力削減モードの SUSPEND のビヘイビアーを定義します。

アーキテクチャ サポート

- Spartan-3A
- Spartan-6

適用可能エレメント

ENABLE_SUSPEND 制約には、次の特徴があります。

- グローバル制約です。
- 特定のエレメントには設定されません。

適用ルール

ENABLE_SUSPEND 制約には、次の特徴があります。

- グローバル制約です。
- デザイン全体にグローバルに適用されます。

制約値

- NO (デフォルト)

  この機能をオフにします。
- FILTERED
  - SUSPEND をオンにします。
  - グリッチ フィルターをオンにします。
  - オンにするには長めのパルス幅が必要です。
- UNFILTERED
  - SUSPEND をオンにします。
  - グリッチ フィルターをバイパスします。
  - SUSPEND をより速くオンにできます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG ENABLE_SUSPEND="{NO|FILTERED|UNFILTERED}";
```

UCF 構文例

```
CONFIG ENABLE_SUSPEND="FILTERED";
```

FAST

FAST 制約には、次の特徴があります。

- 基本的なマップ制約です。
- IOB 出力の速度を増加します。
- ノイズおよび消費電力が増加する可能性があります。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

- 出力プリミティブ
- 出力パッド
- 双方向パッド

UCF ファイル内のパッド コンポーネントに接続されているネットにも設定できます。ネットに設定した制約は、NGDBuild により NGD ファイル内のパッド インスタンスに挿入され、マップで処理されます。

次の構文を使用してください。

NET "*net_name*" **FAST;**

適用ルール

- ネットがパッドに接続されている場合を除き、ネットには設定できません。この場合、FAST はパッド インスタンスに設定されているものと見なされます。
- マクロ、モジュール、またはエンティティに設定すると、そのモジュールよりも下位の階層にあるすべての適用可能エレメントに伝播されます。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  FAST
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute FAST: string;
```

VHDL 制約を次のように指定します。

```
attribute FAST of signal_name: signal is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* FAST = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

- **INST "$1I87/y2" FAST;**

  エレメント y2 の出力速度を増加します。

- **NET "net1" FAST;**

  net1 が接続されているパッドの出力速度を増加します。

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" fast={TRUE|FALSE};
END;
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

FEEDBACK

FEEDBACK 制約は、DCM がボード スキュー アプリケーションで使用される場合に、外部の DCM フィードバック パスの遅延を定義するために使用します。

- この遅延は、ボード トレースの最大外部パス遅延として定義されます。
- 内部 FPGA パス遅延は含まれません。

FEEDBACK 制約を使用すると、タイミング ソフトウェアで次が実行できます。

- タイミング ツールで DCM/PLL/MMCM 位相シフトを正しく指定
- 関連する同期パスを解析

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

なし

適用ルール

次は、パッド ネットに対応している必要があります。

- input_feedback_clock_net

  入力パッドを指定する必要があります。
- output_clock_net

  出力パッドを指定する必要があります。

それ以外のネットに設定した場合、エラーが発生します。

制約値

- input_feedback_clock_net

  DCM へのフィードバックとして使用される入力パッド ネットの名前
- value

  ユーザーによって算出されたボード トレース遅延です。
- units
  - ns (デフォルト)
  - ps
- output_clock_net

  DCM で駆動される出力パッド ネット

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
NET output_clock_net FEEDBACK = value units NET input_feedback_clock_net;
```

XCF 構文

```
BEGIN MODEL "entity_name"

 NET output_clock_net FEEDBACK = value
units NET input_feedback_clock_net;

END;
```

PCF 構文

```
{BEL|COMP} output_clock_net FEEDBACK = value units {BEL|COMP} input_feedback_clock_net;
```

Constraints Editor の構文

ISE® Design Suite での Constraints Editor および ISE® Design Suite に関する詳細は、ISE Design Suite ヘルプを参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

FILE

ファイル (FILE) 制約を使用すると、別のネットリストの中のファイルの検出先を NGCBuild に伝えることができます。

別のネットリストに含まれるモジュールをインスタンシエートすると、このファイル名が NGCBuild によって検索されます。このため、ネットリストの名前はファイル内で定義されたモジュールの名前と同じにする必要があります。モジュール名とは異なる名前を付ける場合は、インスタンス宣言に FILE 制約を指定しておきます。これにより、NGCBuild によって指定したファイル内でモジュールが検索されます。

ザイリンクス制約が VHDL のキーワードにもなっている場合は、そのザイリンクス制約を使用できません。問題を回避するには、制約のエイリアスを使用します。制約にはそれぞれ固有のエイリアスがあります。エイリアス名には、制約名に XIL という接頭辞を付けます。たとえば、FILE 制約は VHDL で使用できないので、XIL_FILE という名前を使用します。XIL_FILE という名前を使用します。現在のところ、既存の XILFILE エイリアスがまだサポートされています。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

指定したファイル内で定義されるインスタンス宣言に適用されます。

適用ルール

インスタンスのみに適用されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  FILE
- 属性値

  file_name.extension

file_name は、制約が設定されたエレメントの基になるロジックを表すファイルの名前です。

ファイル タイプ例は、次のとおりです。

- **EDIF**
- EDN
- NGC
- NMC

VHDL 構文

VHDL 制約を次のように宣言します。

attribute xilfile: *string*;

VHDL 制約を次のように指定します。

attribute xilfile of {*instance_name*|*component_name*} **:** {**label**|**component**} **is** "*file_name*";

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

(* XIL_FILE = "*file_name*" ***)**

UCF および NCF 構文

INST <instance definition> **FILE**= <filename definition is located in>;

この構文は、UCF には適用できません。

FLOAT

FLOAT 制約には、次の特徴があります。

- 基本的なマップ制約です。
- トライステート パッドが駆動されていないときにフロートできます。

FLOAT 制約は、適用される I/O のデフォルトの終端が、ISE® Design Suite で次に設定されている場合に有用です。

- PULLUP
- PULLDOWN
- KEEPER

アーキテクチャ サポート

- CoolRunner™ XPLA3
- CoolRunner-II

適用可能エレメント

ネットまたはピンに設定できます。

適用ルール

設定されたネットまたはピンに適用されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  FLOAT
- 属性値
  - TRUE
  - FALSE
  - 必須ではありません。デフォルトで TRUE が設定されています。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute FLOAT: string;
```

VHDL 制約を次のように指定します。

```
attribute FLOAT of signal_name  : signal  is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* FLOAT = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

```
NET "signal_name" FLOAT;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" FLOAT;
END;
```

FROM-THRU-TO

FROM-THRU-TO 制約には、次の特徴があります。

- 高度なタイミング制約です。
- High または Low の時間を使用した Period 制約に関連しています。

同期パスの場合はセットアップ パスのみを設定でき、ホールド パスは設定できません。

この制約は、次のような特定のパスに適用されます。

1. ソース グループから開始
2. 中間点を通過
3. デスティネーション グループで終了

ソース グループおよびデスティネーション グループは、ユーザー グループまたは定義済みグループです。THRU を使用する前に TPTHRU を使用して、パスの中間点を定義する必要があります。

注記 : OFFSET 制約には定義済みグループは使用できません。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

定義済みのグループおよびユーザー定義のグループに適用されます。

適用ルール

この制約は、指定した FROM-THRU-TO パスにのみ適用されます。

制約値

- identifier

  文字とアンダースコアを組み合わせて入力
- source_group および destination_group

  ユーザー定義のグループまたは定義済みのグループ。

  注記 : OFFSET 制約には定義済みグループは使用できません。
- thru_pt1 および thru_pt2

  タイミング解析用のパスを定義する中間点です。
- value

  遅延時間
- units
  - ps
  - ms
  - ns
  - micro

FROM、THRU、TO すべてを必ずしも指定する必要はありません。次のように、ほとんどすべての組み合わせを使用することができます。

- FROM-TO
- FROM-THRU-TO
- THRU-TO
- TO
- FROM
- FROM-THRU-THRU-THRU-TO
- FROM-THRU

THRU ポイントの数に制限はありません。ソース、スルーポイント、デスティネーションは、次のいずれかになります。

- net
- bel
- comp
- macro
- pin
- timegroup

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

```
TIMESPEC "TSidentifier"=FROM "source_group" THRU "thru_pt1"...[THRU  "thru_pt2" ...]
 TO "destination_group" value [Units] [DATAPATHONLY];
```

DATAPATHONLY キーワード

- FROM-TO 制約でクロック スキューまたは位相情報が考慮されないことを示します。
- このキーワードを使用すると、制約が設定されているグループとタイミング解析されるグループのデータ パスのみが指定されます。

```
TIMESPEC TS_MY_PathB = FROM "my_src_grp" THRU "my_thru_pt" TO "my_dst_grp" 13.5 ns DATAPATHONLY;
```

Constraints Editor の構文

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PCF 構文

```
TSname=MAXDELAY FROM TIMEGRP "source" THRU TIMEGRP "thru_pt1" ...THRU "thru_ptn" TO
TIMEGRP "destination" [DATAPATHONLY];
```

FROM-TO

FROM-TO (From To) 制約には、次の特徴があります。

- 2 つのグループ間のタイミング制約を定義します。
- High または Low の時間を使用した Period 制約に関連しています。

この場合のグループは、ユーザー定義のグループまたは定義済みのグループです。

同期パスの場合、この制約はセットアップ パスのみを制御し、ホールド パスは制御しません。

Virtex®-5 の場合、FROM-TO 制約でセットアップ パスとホールド パスの両方が制御されます。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

定義済みのグループおよびユーザー定義のグループに適用されます。

適用ルール

2 つのグループ間で指定されたパスに適用されます。

制約値

- **TS***name*

  常に「TS」で始める必要があります。その後に英数字またはアンダースコアを付けます。
- group1

  オリジナル パス
- group2

  デスティネーション パス
- value
  - ns (デフォルト)
  - MHz
  - 次のようなタイムスペックもあります。
    - TS_C2S/2
    - TS_C2S*2

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

```
TIMESPEC TSname=FROM "group1" TO "group2" value [DATAPATHONLY];
```

DATAPATHONLY キーワード

- FROM-TO 制約でクロック スキューまたは位相情報が考慮されないことを示します。
- このキーワードを使用すると、制約が設定されているグループとタイミング解析されるグループのデータ パスのみが指定されます。

```
TIMESPEC TS_MY_PathA = FROM "my_src_grp" TO "my_dst_grp" 23.5 ns DATAPATHONLY;
```

XCF 構文

XST では FROM-TO がサポートされますが、次はサポートされません。

- FROM-THRU-TO
- 仕様の結合
- 定義済みグループの文字列の一致

```
TIMESPEC TS_1 = FROM FFS(machine/*) TO FFS 2 ns;
```

PCF 構文

```
TSname=MAXDELAY FROM TIMEGRP "group1" TO TIMEGRP "group2"value[DATAPATHONLY];
```

FROM、THRU、TO すべてを必ずしも指定する必要はありません。次のように、ほとんどすべての組み合わせを使用することができます。

- FROM-TO制約
- FROM-THRU-TO
- THRU-TO
- TO
- FROM
- FROM-THRU-THRU-THRU-TO
- FROM-THRU

THRU ポイントの数に制限はありません。ソース、スルーポイント、デスティネーションは、次のいずれかになります。

- net
- bel
- comp
- macro
- pin
- timegroup

Constraints Editor の構文

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

FSM_STYLE

FSM_STYLE (FSM スタイル) 制約の詳細については、『XST ユーザー ガイド (Virtex-4、Virtex-5、Spartan-3、および CPLD 用)』および『XST ユーザー ガイド (Virtex-6、Spartan-6、および 7 シリーズ デバイス用)』を参照してください。

HBLKNM

HBLKNM (階層ブロック名) 制約には、次の特徴があります。

- 高度なマップ制約です。
- 階層的なブロック名をロジック エレメントに割り当て、フラット化された階層デザインでのグループ化を行います。

  階層デザインの異なるレベルにあるエレメントに同じブロック名が付いていて、そのデザインがフラットになると、NGCBuild により階層パス名が接頭辞として HBLKNM 値に追加されます。

BLKNM 制約と同様、HBLKNM はファンクション ジェネレーターとフリップフロップを同じ CLB にマップします。同じ HBLKNM を持つシンボルは、可能であれば、同じ CLB にマップされます。

BLKNM ではなく、HBLKNM を使用した場合、変換中に階層パス名が追加されるという利点があります。同じデザイン エレメントの異なるインスタンス内のエレメントには、同じ HBLKNM 制約や値を使用できます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

この制約は、次のエレメントまたはデザイン エレメントのカテゴリを 1 つまたは複数使用して設定できます。すべてのデバイスですべてのエレメントがサポートされるわけではありません。どのデバイス ファミリにどのエレメントを使用できるかは、ライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

- レジスタ
- I/O エレメントおよびパッド
- FMAP
- PULLUP
- ACLK
- GCLK
- BUFG
- BUFG
- BUFGP
- ROM
- RAMS
- RAMD
- キャリー ロジック プリミティブ

HBLKNM は、UCF ファイル内のパッド コンポーネントに接続されているネットにも設定できます。ネットに設定した制約は、NGCBuild により NGD ファイル内のパッド インスタンスに挿入され、マップで処理されます。

次の構文を使用してください。

NET "*net_name*" **HBLKNM=***property_value*;

適用ルール

- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。
- ただし、NET に適用される場合、HBLKNM は PAD にのみ伝播されます。

制約値

block_name

そのシンボル タイプに対して有効なブロック名

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  HBLKNM
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute hblknm: string;
```

VHDL 制約を次のように指定します。

```
attribute hblknm
of {entity_name|component_name|signal_name|label_name}: {entity|component|signal|label}
is "block_name";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* HBLKNM = "block_name" *)
```

UCF および NCF 構文

NET “*net_name*” **HBLKNM=***property_value*;

INST “*instance_name*” **HBLKNM=***block_name*;

- **INST "$I13245/this_fmap" HBLKNM=group1;**

  エレメント this_fmap がブロック group1 に配置されます。

- **NET "net1" HBLKNM=$COMP_0;**

  HBLKNM 制約を net1 に接続されているパッドに設定します。

同じ HBLKNM があるエレメントは、可能であれば同じロジック ブロックに配置されます。配置できない場合は、エラーが発生します。

複数のエレメントに異なるブロック名を付けた場合は、1 つのブロックにまとめて配置されません。

HIODELAY_GROUP

HIODELAY_GROUP (HIODELAY グループ) 制約には、次の特徴があります。

- デザイン インプリメンテーション制約です。
- IDELAY/IODELAY の階層セットを IDELAYCTRL を組み合わせ、IDELAYCTRL を自動的に複製して配置させることができます。

  詳細は、デバイスのユーザー ガイドの「IDELAYCTRL」セクションを参照してください。

アーキテクチャ サポート

Virtex®-4 および Virtex-5 デバイスにのみ適用されます。Virtex-4 の場合、この制約は MAP の[Timing Driven Pack and Placement] オプションを使用した場合にのみサポートされます。

適用可能エレメント

IDELAY、IODELAY、IDELAYCTRL プリミティブ インスタンシエーション

適用ルール

HIODELAY_GROUP はデザイン エレメントにのみ適用できます。ネット、信号、またはピンには設定できません。2 つ以上の HIODELAY_GROUP 制約をマージする場合は、「MIODELAY_GROUP」を参照してください。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

VHDL 構文

VHDL 制約を次のように宣言します。

attribute HIODELAY_GROUP: string;

VHDL 制約を次のように指定します。

attribute HIODELAY_GROUP of {*component_name*|*label_name*}**:{component|label} is** **"***group_name***";**

group_name の詳細は、「UCF 構文」を参照してください。

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

(* HIODELAY_GROUP = "*group_name***" *)**

group_name の詳細は、「UCF 構文」を参照してください。

UCF および NCF 構文

INST "*instance_name***" HIODELAY_GROUP =** *group_name***;**

説明 :

group_name は、IDELAY/IODELAY と IDELAYCTRL のセットをグループとして定義するために割り当てる名前です。

HLUTNM

HLUTNM (階層ルックアップ テーブル名) 制約には、次の特徴があります。

- Virtex®-5 デバイスでのみサポートされます。
- 論理シンボルを Virtex-5 デバイスの LUT サイトにまとめます。
- 値は文字列で、条件を満たす 2 つのシンボルに設定されます。
- シンボルの階層内ではこれらのシンボルのみに使用されます。シンボルは、SLICE コンポーネント内で共有された LUT サイト内でインプリメントされます。
- 機能は HBLKNM 制約に似ています。

アーキテクチャ サポート

Virtex-5 デバイスにのみ適用できます。

適用可能エレメント

この制約は、次のものに適用できます。

- 次のような 2 つのシンボル :
  - 共通の階層を共有
  - 階層レベル内で重複していない
- 両方のシンボルの、重複しない入力ピン数の合計が 5 を超えない場合は、5 入力以下のファンクション ジェネレータ シンボル (LUT、SRL16) 2 つに設定できます。
- 次のような場合、6 入力の読み込み専用ファンクション ジェネレータ シンボル (LUT6) と 5 入力の読み込み専用シンボル (LUT5) に設定できます。
  - 両方のシンボルの、重複しない入力ピン数の合計が 6 入力を超えない場合
  - 6 入力シンボル プログラムの下位 32 ビットは 5 入力シンボル プログラムのの 32 ビットすべてと一致している場合

適用ルール

この制約は、次の条件の 2 つのシンボルに設定できます。

- 共通の階層を共有
- 階層レベル内で重複していない

制約値

- instance_name

  インスタンシエートされた LUT、または LUTRAM のインスタンス名です。

- string_value

  指定された階層レベル内の 2 つのシンボルのみに使用されます。
  - デフォルトの値はありません。
  - 空白の場合は、制約は無視されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なエレメントまたはシンボル タイプに設定します。
- 属性名

  HLUTNM
- 属性値

  <*user_defined*>

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute hlutnm:  string;
```

VHDL 制約を次のように指定します。

```
attribute hlutnm of instance_name : label is "string_value ";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* HLUTNM = "string_value" *)
```

UCF および NCF 構文

```
INST "symbol_name" HLUTNM=string_value ;
```

XCF 構文

```
MODEL "symbol_name" hlutnm = string_value ;
```

H_SET

H_SET の詳細は、「HU_SET」を参照してください。

HU_SET

HU_SET (HU セット) 制約には、次の特徴があります。

- 高度なマップ制約です。
- デザイン階層で定義されます。
- セットの名前を指定できます。
  - H_SETの場合は、1 つの階層エレメントに 1 つの H_SET 制約しか指定できませんが、
  - HU_SET の場合は、集合として名前を指定すると複数の HU_SET を指定できるようになります。

NGCBuild では、デザインをフラットにする際に、HU_SET 名に階層名が接頭語として付けられます。

HU_SET と H_SET の相違点

HU_SET	H_SET
HU_SET セットには、ユーザー定義のベース名の前に階層名が付けられる	デザインのフラット化によって自動的に階層名がベース名の前に付けられる
HU_SET が設定されたシンボルが冒頭にくる	RLOC 制約が設定されたシンボルの 1 レベル上にあるマクロのインスタンスが冒頭にくる

詳細は、「RLOC」を参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

この制約は、次のエレメントまたはデザイン エレメントのカテゴリを 1 つまたは複数使用して設定できます。すべてのデバイスですべてのエレメントがサポートされるわけではありません。デバイス別にどのデザイン エレメントが使用可能かを確認するには、そのデバイスのライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

- レジスタ
- FMAP
- マクロ インスタンス
- ROM
- RAMS
- RAMD
- MULT18X18S
- RAMB4_Sm_Sn
- RAMB4_Sn
- RAMB16_Sm_Sn
- RAMB16_Sn
- RAMB16
- DSP48

適用ルール

この制約はデザイン エレメント制約です。ネットへは接続できません。

制約値

set_name

- 集合の識別名です。
- 名前がデザイン内のほかの集合名と重複しないようにしてください。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  HU_SET
- 属性値

  set_name

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute HU_SET: string;
```

VHDL 制約を次のように指定します。

```
attribute HU_SET of {component_name|entity_name|label_name} : {component|entity|label} is "set_name";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* HU_SET = "set_name" *)
```

UCF および NCF 構文

```
INST "instance_name" HU_SET=set_name ;
```

UCF および NCF の構文例

```
INST "$1I3245/FF_1" HU_SET=heavy_set;
```

レジスタ FF_1 のインスタンスを heavy_set 集合に割り当てます。

XCF 構文

```
MODEL "entity_name" hu_set={yes|no};
 BEGIN MODEL "entity_name"
 INST "instance_name" hu_set=yes;
END;
```

IBUF_DELAY_VALUE

IBUF_DELAY_VALUE (入力バッファー遅延値) 制約には、次の特徴があります。

- マップ制約です。
- 追加のスタティック遅延を FPGA アレイの入力パスに追加します。
- クロックまたは IOB (入出力ブロック) レジスタを直接駆動しない入力信号または双方向信号に適用できます。

  クロックおよび IOB レジスタを駆動する信号に設定する制約については、「IFD_DELAY_VALUE」を参照してください。

アーキテクチャ サポート

- Spartan®-3A
- Spartan-3E

適用可能エレメント

最上位の I/O ポートに設定できます。

制約値

IBUF_DELAY_VALUE には 0 ～ 16 の整数を指定できます。

- 0
  - デフォルトは 0 です。
  - この場合、入力パスに遅延が追加されません。
- 1 ～16
  - この制約に 1 ～ 6 の値を設定すると、入力パスにその分遅延が追加されます。
  - これらの値は、遅延の時間単位ではなく、バッファーの追加遅延を指定するものです。

詳細は、データシートを参照してください。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 最上位のポートに新しいプロパティを設定します。
- 属性名

  IBUF_DELAY_VALUE
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

最上位のポートに、VHDL 属性を設定します。

```
attribute IBUF_DELAY_VALUE : string;

attribute IBUF_DELAY_VALUE of top_level_port_name:
signal is "value";
```

次の構文は、ネット DataIn1 に値 5 の IBUF_DELAY_VALUE 制約を設定します。

```
attribute IBUF_DELAY_VALUE :  string;

attribute IBUF_DELAY_VALUE of DataIn1: label is "5";
```

Verilog 構文

最上位のポートに Verilog 属性を設定します。

```
(* IBUF_DELAY_VALUE="value" *) input top_level_port_name;
```

次の構文は、ネット DataIn1 に値 5 の IBUF_DELAY_VALUE 制約を設定します。

```
(* IBUF_DELAY_VALUE="5" *) input DataIn1;
```

UCF および NCF 構文

```
NET "top_level_port_name" IBUF_DELAY_VALUE = value;
```

次の構文は、ネット DataIn1 に値 5 の IBUF_DELAY_VALUE 制約を設定します。

```
NET "DataIn1" IBUF_DELAY_VALUE = 5;
```

IFD_DELAY_VALUE

IFD_DELAY_VALUE (IFD 遅延値) 制約には、次の特徴があります。

- マップ制約です。
- 追加のスタティック遅延を FPGA アレイの入力パスに追加します。
- IOB (入出力ブロック) レジスタを駆動する入力信号または双方向信号に適用できます。

  IOB レジスタを駆動しない信号に設定する制約については、「IBUF_DELAY_VALUE」を参照してください。

アーキテクチャ サポート

- Spartan®-3A
- Spartan-3E

適用可能エレメント

最上位の I/O ポートに設定できます。

適用ルール

この制約は I/O シンボルに設定されますが、I/O コンポーネント全体に適用されます。

制約値

次に設定できます。

- 0 ～ 8 の整数
  - IFD_DELAY_VALUE を 0 に設定すると、データ パスに遅延が追加されません。
  - 1 ～ 8 の整数を設定すると、データ パスのその値の分だけ遅延が追加されます。

    これらの値は、遅延の時間単位ではなく、バッファーの追加遅延を指定するものです。

    詳細は、データシートを参照してください。
- AUTO (デフォルト)
  - 適切な遅延が自動的にデータ パスに追加されます。
  - デスティネーション レジスタの入力ホールド タイムが満たされるようにします。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットに設定します。
- 属性名

  IFD_DELAY_VALUE
- 属性値
  - 0 ～ 8
  - AUTO

VHDL 構文

最上位のポートに、VHDL 属性を設定します。

```
attribute IFD_DELAY_VALUE : string;
```

次の構文は、ネット DataIn1 に値 5 の IFD_DELAY_VALUE 制約を設定します。

```
attribute IFD_DELAY_VALUE : string;
attribute IFD_DELAY_VALUE of DataIn1: label is "5";
```

Verilog 構文

最上位のポートに Verilog 属性を設定します。

```
(* IFD_DELAY_VALUE="value" *) input top_level_port_name;
```

次の構文は、ネット DataIn1 に値 5 の IFD_DELAY_VALUE 制約を設定します。

```
(* IFD_DELAY_VALUE="5" *) input DataIn1;
```

UCF および NCF 構文

```
NET "top_level_port_name" IFD_DELAY_VALUE = value;
```

value

IBUF の遅延時間 (数値) です。

次の構文は、ネット DataIn1 に値 5 の IFD_DELAY_VALUE 制約を設定します。

```
NET "DataIn1" IFD_DELAY_VALUE = 5;
```

IN_TERM

IN_TERM 制約には、次のような特徴があります

- 基本的なマップ制約です。
- 入力終端抵抗のコンフィギュレーションを設定します。
- 次で使用できます。
  - 入力パッド NET
  - 入力パッド INST
  - デザイン全体

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

この制約は、次のエレメントまたはデザイン エレメントのカテゴリを 1 つまたは複数使用して設定できます。

- IOB 入力コンポーネント (例 : IBUF)
- 入力パッド ネット

すべてのデバイスですべてのエレメントがサポートされるわけではありません。デバイス別にどのデザイン エレメントが使用可能かを確認するには、そのデバイスのライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

適用ルール

ネットまたは信号がパッドに接続されている場合を除いて、ネットや信号に設定できません。この場合、IN_TERM はパッド インスタンスに設定されているものと見なされます。

制約値

- NONE
- TUNED_SPLIT
- UNTUNED_SPLIT_25
- UNTUNED_SPLIT_50
- UNTUNED_SPLIT_75

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図構文

- パッド ネットに設定します。
- 属性名

  IN_TERM
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
Attribute IN_TERM: string;
```

VHDL 制約を次のように指定します。

```
attribute IN_TERM of signal_name: signal is
"{NONE|TUNED_SPLIT|UNTUNED_SPLIT_25|UNTUNED_SPLIT_50|UNTUNED_SPLIT_75}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* IN_TERM  = "{NONE|TUNED_SPLIT|UNTUNED_SPLIT_25|UNTUNED_SPLIT_50|UNTUNED_SPLIT_75 }" *)
```

UCF および NCF 構文

```
NET "pad_net_name"IN_TERM = "{NONE|TUNED_SPLIT|UNTUNED_SPLIT_25|UNTUNED_SPLIT_50|UNTUNED_SPLIT_75 }" ;
```

次の文は、I/O で PULLUP を使用するよう指定します。

```
DEFAULT IN_TERM = TUNED_SPLIT;
```

次の文は、IN_TERM をグローバルに設定しています。

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" in_term=tuned_split;
END;
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

INREG

この制約は、入力パッドにより駆動される D 入力付きのレジスタおよびラッチのインスタンスか、そのレジスタおよびラッチの Q 出力ネットに設定します。デフォルトでは、CoolRunner™ XPLA3 または CoolRunner-II デザインに含まれる入力パッドにより駆動される D 入力付きのレジスタおよびラッチが、デバイスの高速入力パスを使用して自動的にインプリメントされます。ISE® Design Suite で [Use Fast Input for INREG for the Fit (Implement Design)] プロセスを使用しない場合、INREG 属性が設定されたレジスタおよびラッチのみが高速入力で最適化されます。

アーキテクチャ サポート

CoolRunner™ XPLA3、CoolRunner-II デバイスでのみサポートサポートされます。

適用可能エレメント

入力パッドにより駆動される D 入力付きのレジスタおよびラッチのインスタンスか、そのレジスタおよびラッチの Q 出力ネットに設定します。

適用ルール

制約が設定されたレジスタ/ラッチか、そのレジスタ/ラッチの Q 出力ネットに適用されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- レジスタ、ラッチ、またはネットに設定します。
- 属性名

  INREG
- 属性値

  なし (デフォルトは TRUE)

UCF 構文

NET “*signal_name*” INREG;

INST “*register_name*” INREG;

INTERNAL_VREF_BANK

INTERNAL_VREF_BANK 制約には、次のような特徴があります。

- 該当する I/O バンクの内部 Vref 機能に電圧値を割り当てることができます。
- 参照電圧を提供するファンクションから I/O バンクの Vref ピンを空ける際に便利です。
- I/O バンクに内部 Vref を使用すると、Vref ピンは別の用途に使用できます。

アーキテクチャ サポート

Virtex®-6、Kintex™-7 および Virtex®-7 デバイスにのみ適用されます。

適用可能エレメント

INTERNAL_VREF_BANK 制約には、次のような特徴があります。

- グローバル CONFIG 制約です。
- 特定のインスタンスや信号名には設定されません。

適用ルール

デザイン全体の特定バンクの I/O に適用されます。

制約値

- n

  バンク数
- v

  ターゲット電圧値
  - 0.0
  - 0.6
  - 0.675
  - 0.75
  - 0.9
  - 1.1
  - 1.25

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

```
CONFIG INTERNAL_VREF_BANKn=v;
```

UCF および NCF の構文例

```
CONFIG INTERNAL_VREF_BANK5=1.1;
```

IOB

IOB 制約には、次の特徴があります。

- 基本的なマップおよび合成制約です。
- IOB に移動可能なフリップフロップとラッチを指定します。
- マップの **-pr** コマンド ライン オプションよりも優先されます。
- LOC 制約よりは優先されません。

マップでは、次のコマンド ライン オプションがサポートされます。

-pr i|o|b|off

このオプションを使用すると、グローバルにフリップフロップまたはラッチのプリミティブを次のいずれかに移動できます。

- 入力 IOB (i)
- 出力 IOB (o)
- 入出力 IOB (b)

フリップフロップまたはラッチで指定する IOB 制約は、マップに対して、可能であればそのインスタンスを IOB タイプのコンポーネントにパックするよう指示します。

Xilinx® Synthesis Technology (XST) では、次が実行されます。

- IOB 制約は、インプリメンテーション制約と認識されます。
- 生成された NGC ファイルに伝搬されます。
- 出力バッファーのイネーブル ピンを駆動するフリップフロップおよびラッチのコピーを作成して、対応するフリップフロップおよびラッチが IOB にパックされるようにします。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

- INFF/OUTFF 以外のフリップフロップ、ラッチ プリミティブ
- レジスタ

適用ルール

それが接続されているデザイン エレメントに適用

制約値

- TRUE

  フリップフロップまたはラッチが IOB に移動し ます。

- FALSE

  フリップフロップまたはラッチは IOB に移動しません。

- AUTO (XST のみ)

  タイミング制約を考慮して、フリップフロップを IOB に移動するかどうかが自動的に決定されます。

- FORCE
  - フリップフロップまたはラッチが IOB に移動されます。
  - レジスタが次の状態の場合は、エラー メッセージが表示されます。
    - レジスタに I/O 接続があり、さらに
    - IOB にパックできない場合

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- フリップフロップまたはラッチのインスタンス、またはレジスタに設定します。
- 属性名

  IOB

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute iob: string;
```

VHDL 制約を次のように指定します。

```
attribute iob
of {component_name|entity_name|label_name|signal_name} : {component|entity |label|signal} is
"{TRUE|FALSE|AUTO|FORCE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* IOB = "{TRUE|FALSE|AUTO|FORCE}" *)
```

UCF および NCF 構文

INST "*instance_name*" **IOB=**{**TRUE**|**FALSE**|**FORCE**};

- **INST "foo/bar" IOB=TRUE;**

  マップでインスタンス foo/bar が IOB コンポーネントに配置されるように指定します。

- **NET "foo/bar" IOB=TRUE;**

  この構文は、UCF ではサポートされません。

- **INST "foo/bar" IOB=TRUE;**

  この構文は、UCF でサポートされます。

XCF 構文

BEGIN MODEL "*entity_name*"

NET "*signal_name*" **iob=**{**true**|**false**|**auto**|**force**};

INST " *instance_name*" **iob=**{**true**|**false**|**auto**|**force**};

END;

Constraints Editor の設定

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

IOBDELAY

IOBDELAY (入出力ブロック遅延) 制約には、次の特徴があります。

- 基本的なマップ制約です。
- デバイス内の入力パス遅延エレメントのプログラム方法を指定できます。

入力信号は、次の 2 つのいずれかに送信されます。

- ローカル IOB 入力のフリップフロップ
- IOB 外部のロード

ザイリンクス デバイスを使用すると、遅延エレメントはこれらデスティネーションのうち 1 つか、または両方への信号送信を遅らせることができます。

IOBDELAY は NODELAY と併用できません。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

次の I/O シンボルのいずれかに適用されます。

- I/O パッド
- I/O バッファー
- 入力パッド ネット

適用ルール

この制約は I/O シンボルに設定されますが、I/O コンポーネント全体に適用されます。

制約値

- NONE

  NONE に指定すると、IBUF パスと IFD パスの遅延が OFF に設定されます。

  - この文は、IBUF パスと IFD パスの遅延を OFF に設定します。

    INST "xyzzy" **IOBDELAY=NONE**

  - Spartan®-3 デバイスの場合、デフォルトは NONE に設定されているので、ホールド タイムを 0 にすることができます。

- BOTH

  IBUF パスと IFD パスの遅延が ON に設定されます。

- IBUF
  - I/O コンポーネント内部のレジスタで遅延をオフにします。
  - 入力バッファーが I/O コンポーネント外部にあるレジスタの D ピンを駆動する場合、I/O コンポーネント外部のレジスタで遅延をオンにします。
- IFD
  - I/O コンポーネント内部のレジスタで遅延をオンにします。
  - レジスタが I/O コンポーネントの入力側に配置されている場合、IOB=TRUE 制約がレジスタに設定されているかどうかにかかわらず、I/O コンポーネント外部のレジスタで遅延をオフに設定します。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- I/O シンボルに設定します。
- 属性名

  IOBDELAY

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute iobdelay: string;
```

VHDL 制約を次のように指定します。

```
attribute iobdelay of {component_name|label_name}: {component|label} is "{NONE|BOTH|IBUF|IFD}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* IOBDELAY = {NONE|BOTH|IBUF|IFD} *)
```

UCF および NCF 構文

```
INST "instance_name" IOBDELAY={NONE|BOTH|IBUF|IFD};
```

IODELAY_GROUP

IODELAY_GROUP (IODELAY グループ) 制約には、次の特徴があります。

- デザイン インプリメンテーション制約です。
- IDELAY/IODELAY のセットを IDELAYCTRL を組み合わせ、IDELAYCTRL を自動的に複製して配置させることができます。

詳細は、デバイスのユーザー ガイドの「IDELAYCTRL」セクションを参照してください。

LOC を使用した場合の制限

- LOC 制約を使用しないで 1 つの IDELAYCTRL を複数のバンクに対して複製する場合は、IODELAY_GROUP のみを使用します。
- IODELAY_GROUP を LOC 制約の設定された IDELAYCTRL インスタンスと一緒に使用することはできません。
- デザインには IDELAYCTRL を 1 つしかインスタンシエートできません。
- LOC 制約は指定しないでください。
- IDELAYCTRL を IODELAY_GROUP に入れる必要のある IODELAY 制約はすべてグループにまとめます。
- 各バンクに 1 グループ作成します。

アーキテクチャ サポート

Virtex®-4、Virtex-5、Virtex-6、7 シリーズ デバイスでのみサポートされます

- Virtex-4 デバイスの場合、この制約は MAP の [Timing Driven Pack and Placement] オプションを使用した場合にのみサポートされます。
- IODELAY_GROUP は Virtex-4 および Virtex-5 デバイスでサポートされますが、IDELAYCTRL を複製する方法としては推奨されません。推奨される方法については、デバイスのユーザー ガイドを参照してください。
- IODELAY_GROUP は、Virtex-6 および 7 シリーズ デバイスでは IDELAYCTRL プリミティブを複製する方法として推奨されています。

適用可能エレメント

IODELAY_GROUP 制約は次のエレメントに適用します。

- IDELAY
- IODELAY
- IODELAYE1
- IDELAYE2
- IODELAYE2
- IDELAYCTRL

適用ルール

- IODELAY_GROUP はデザイン エレメントにのみ適用できます。
- ネット、信号、またはピンには適用できません。
- 2 つ以上の IODELAY_GROUP 制約をマージする場合は、「MIODELAY_GROUP」を参照してください。

制約値

group_name

IDELAY/IODELAY 制約と IDELAYCTRL のセットをグループとして定義するために割り当てる名前です。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute IODELAY_GROUP: string;
```

VHDL 制約を次のように指定します。

```
attribute IODELAY_GROUP of {component_name|label_name}: {component|label} is "group_name";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* IODELAY_GROUP = "group_name" *)
```

UCF 構文

```
INST "instance_name" IODELAY_GROUP = group_name;
```

IOSTANDARD

IOSTANDARD (I/O 規格) 制約には、次の特徴があります。

- 基本的なマップ制約です。
- 合成制約です。

FPGA デバイスの場合

IOSTANDARD を使用して、I/O 規格を I/O プリミティブに割り当てます。

IOSTANDARD が設定されたコンポーネントは、SelectIO™ コンポーネントと同じ配置規則 (バンク規則) に従う必要があります。

- 各アーキテクチャのバンク規則については、該当するライブラリ ガイドを参照してください。
- サポートされている I/O 規格の詳細は、該当するデバイスのデータシートを参照してください。

Spartan®-3、Spartan-3A、Spartan-3E、Virtex®-4、Virtex-5 デバイスの場合 :

- コンポーネントの SelectIO を使用するよりも、バッファー コンポーネントに IOSTANDARD を設定することをお勧めします。

  たとえば、IBUF_HSTL_III ではなく、IOSTANDARD=HSTL_III が設定された IBUF を使用します。
- 差動信号標準は IBUF および OBUF には適用されず、IBUFDS、IBUFGDS、OBUFDS、OBUFTDS のみに適用されます。

CPLD デバイスの場合

CoolRunner™-II デバイスの I/O パッドに IOSTANDARD 制約を設定し、入力しきい値および出力 VCCIO 電圧を指定できます。サポートされる値については、該当するデバイスのデータシートを参照してください。

ロケーション制約が指定されていない場合は、IOSTANDARD が設定された出力が CPLDFitter により自動的に同じバンクにまとめられます。

アーキテクチャ サポート

- すべての FPGA デバイス
- CoolRunner-II CPLD デバイス

適用可能エレメント

どのデバイス ファミリにどのエレメントを使用できるかは、ライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

- IBUF、IBUFG、OBUF、OBUFT
- IBUFDS、IBUFGDS、OBUFDS、OBUFTDS
- 出力電圧バンク

適用ルール

- ネットまたは信号がパッドに接続されている場合を除いて、ネットや信号に設定できません。
- パッドに接続されている場合、IOSTANDARD 制約は次の IOB インスタンスに設定されているものと見なされます。
  - IBUF
  - OBUF
  - IOB FF
- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

制約値

iostandard_name

デバイスのデータシートで指定されている I/O 規格の名前になります。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- I/O プリミティブに設定します。
- 属性名

  IOSTANDARD
- 属性値

  iostandard_name

詳細は、次の「UCF および NCF 構文」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute iostandard: string;
```

VHDL 制約を次のように指定します。

```
attribute iostandard of {component_name|label_name}: {component|label} is "iostandard_name ";
```

詳細は、次の「UCF および NCF 構文」を参照してください。

CPLD デバイスの場合、パッド信号にも適用できます。

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

iostandard_name については、UCF セクションを参照してください。

CPLD デバイスの場合、パッド信号にも適用できます。

UCF および NCF 構文

```
INST " instance_name" IOSTANDARD= iostandard_name;
NET "pad_net_name" IOSTANDARD=iostandard_name;
```

XCF 構文

```
BEGIN MODEL "entity_name "
 INST "instance_name" iostandard=string ;
 NET "signal_name" iostandard=string ;
END;
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

Pinout and Area Constraints Editor (PACE) の設定

CPLD でのみサポートされます。FPGA ではサポートされません。

次からアクセスできます。

ISE® Design Suite の [Processes] ペイン

PACE は次のために使用します。

- I/O コンポーネントへのロケーション制約の割り当て
- I/O 規格など特定の I/O プロパティの設定

詳細は、PACE ヘルプで次を参照してください。

「手順」のピンおよびエリアの編集に関するセクション

KEEP

KEEP 制約には、次の特徴があります。

- 高度なマップ制約です。
- 合成制約です。
- ネットがロジック ブロックに吸収されないようにします。
- FPGA の場合、KEEP は内部制約 NOMERGE に変換されます。

  インプリメンテーション ツールのメッセージには、KEEP ではなく、システム プロパティ NOMERGE が表示されます。

ネットの論理ブロックへの吸収

デザインのマップ時に、一部のネットが論理ブロックに含まれることがあります。ブロックに吸収されたネットは、物理的なデザイン データベースには存在しなくなります。これは、ネットの各サイドに接続されたコンポーネントが同じ論理ブロックにマップされる場合などに発生することがあります。この後、ネットはそのコンポーネントを含むブロックに吸収されます。KEEP を使用すると、ネットが吸収されないようにできます。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

信号に適用

適用ルール

それが接続されている信号に適用

制約値

- TRUE (または XCF でのみ YES)
- FALSE (または XCF でのみ NO)
- SOFT (XST のみ)
  - 指定したネットを保持するように XST に命令します。
  - 合成後のネットリストでこのネットに NOMERGE 制約が付かないようになります。

この結果、ネットが合成で保持されても、インプリメンテーション ツールではどのようにでも処理できるようになります。つまり、合成の場合にのみ KEEP=TRUE に設定され、インプリメンテーション ツールでは KEEP=FALSE に設定されるようなものです。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットに設定します。
- 属性名

  KEEP
- 属性値
  - TRUE
  - FALSE
  - SOFT (XST のみ)

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute keep : string;
```

VHDL 制約を次のように指定します。

```
attribute keep of signal_name: signal is "{TRUE|FALSE|SOFT}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* KEEP = "{TRUE|FALSE|SOFT}" *)
```

UCF および NCF 構文

```
INST "instance_name" KEEP={TRUE|FALSE};
```

次の構文で、ネット $SIG_0 が常に認識されるように指定します。

```
NET "$1I3245/$SIG_0" KEEP;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" keep={yes|no|true|false};
END;
```

XCF ファイルでは、KEEP 制約の値はオプションで二重引用符 (" ") で囲むことができます。値が SOFT の場合は、次のように必ず二重引用符 (" ") で囲む必要があります。

```
BEGIN MODEL "entity_name"
 NET "signal_name" keep="soft";
END;
```

KEEP_HIERARCHY

KEEP_HIERARCHY は、合成およびインプリメンテーション制約です。

デザイン階層が合成で維持された場合、インプリメンテーション ソフトウェアでこの制約が使用され、次が実行されます。

- インプリメンテーション プロセスを通して階層が保持されます。
- 指定した階層を使用してシミュレーション ネットリストが作成できるようになります。

XST では、よい結果を得るために、エンティティおよびモジュールすべてを最適化してデザインをフラットにすることがあります。この制約を true に設定すると、生成されたネットリストが階層構造であるため、デザインのエンティティとモジュールの階層およびインターフェイスを維持できます。

この制約は、HDL 合成で推論されたマクロではなく、HDL デザインで指定された階層ブロック (VHDL のエンティティ、Verilog のモジュール) に適用されます。

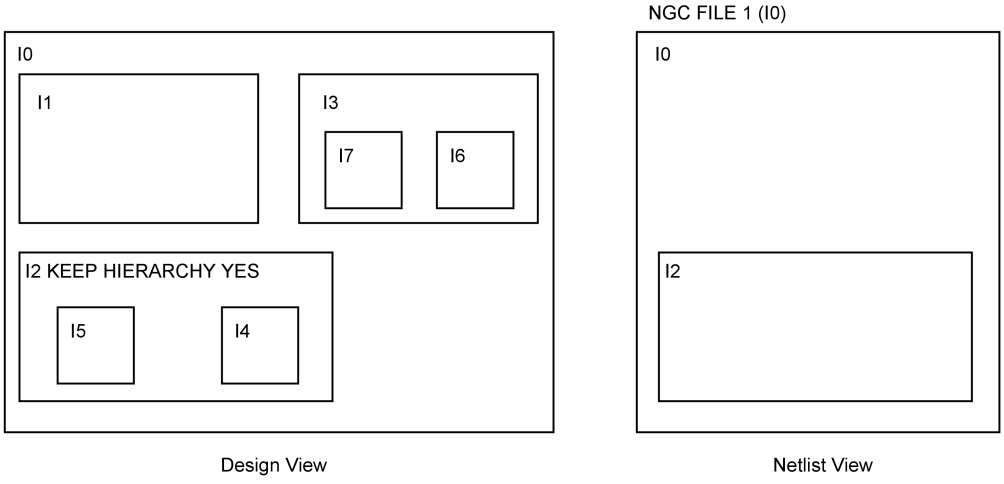
一般的に、HDL デザインは階層ブロックの集合です。より単純な階層で個別に最適化が行われるため、デザイン階層を保持すると処理速度が向上します。また、コラプスや因数分解などの最適化のプロセスはロジック全体にグローバルに適用されるため、階層ブロックのマージによりフィットの結果が向上します (積項およびデバイス マクロセルの少量化、周波数の向上など)。

KEEP_HIERARCHY を設定すると、ユーザー定義のデザイン ユニットの階層をフラットに設定できます。属性値は、true または false です。デフォルトで、ユーザーの階層は維持されます。

KEEP_HIERARCHY 制約の例

たとえば、エンティティまたはモジュール **I2** に制約を設定した場合、次のようになります。

- I2 の階層は維持されたままで最後のネットリストに含まれます
- I2 の下にある I4 および I5 はフラットになります。
- I1、I3、I6、I7 も同様にフラットになります。



アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

階層ブロックまたはシンボル ブロックを含む論理ブロックに適用します。

適用ルール

それが接続されているエンティティ、モジュールまたは信号に適用

制約値

- true (CPLD デバイスの場合デフォルト)

  HDL プロジェクトで記述されたデザイン階層を保持します。この値を合成に適用した場合、インプリメンテーションにも適用されます。

- false (FPGA デバイスの場合デフォルト)

  階層ブロックが 最上位モジュールにマージされます。

- soft

  デザイン階層は合成で維持されますが、インプリメンテーションで制約は適用されません。

注記 : この制約は、XST では yes、true、no、false、および soft に設定できます。コマンド ラインで使用する場合は、yes、no、および soft のみが使用できます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- エンティティまたはモジュール シンボルに設定します。
- 属性名

  KEEP_HIERARCHY
- 属性値
  - TRUE
  - FALSE

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute keep_hierarchy : string;
```

VHDL 制約を次のように指定します。

```
attribute keep_hierarchy
of architecture_name: architecture is {TRUE|FALSE|SOFT};
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* KEEP_HIERARCHY = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

```
INST "instance_name" KEEP_HIERARCHY={TRUE|FALSE};
```

XCF 構文

XST では、KEEP_HIERARCHY に次の値を使用できます。

- yes
- true
- no
- false
- soft

コマンド ラインで使用する場合は、yes、no、および soft のみが使用できます。

```
MODEL "entity_name" keep_hierarchy={yes|no|soft};
```

ISE Design Suite の構文

グローバルに設定する場合 :

[Process Properties] ダイアログ ボックス → [Synthesis Options] → [Keep Hierarchy]

[Sources] ペインでデザイン ファイルを選択した場合 :

[Processes] → [Synthesize] → [Process Properties] → [Property display level] → [Advanced] をクリックします。

KEEPER

KEEPER 制約には、次の特徴があります。

- 基本的なマップ制約です。
- 出力ネットに対して指定すると、その出力ネットの値が保持されます。

たとえば、ネットに対してロジック 1 を駆動すると、KEEPER はそのネットに対してウィーク/抵抗 1 を駆動します。その後、ネットドライバーがトライステートになっても、KEEPER はウィーク/抵抗値 1 を駆動し続けます。

KEEPER 制約は、KEEPER コンポーネントと同じバンク規則に従う必要があります。バンク規則の詳細は、該当するデバイスのライブラリ ガイドを参照してください。

KEEPER、PULLUP および PULLDOWN は、パッド ネットにのみ使用でき、INST には使用できません。

CoolRunner™-II デバイスでは、KEEPER と PULLUP を共に使用できません。

アーキテクチャ サポート

- すべての FPGA デバイス
- CoolRunner-II CPLD デバイス

適用可能エレメント

トライステート入力/出力パッド ネットに適用されます。

適用ルール

ネットまたは信号がパッドに接続されている場合を除いて、ネットや信号に設定できません。この場合、KEEPER はパッド インスタンスに設定されているものと見なされます。

制約値

- YES
- NO
- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 出力パッド ネットに設定します。
- 属性名

  KEEPER
- 属性値
  - TRUE
  - FALSE

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute keeper: string;
```

VHDL 制約を次のように指定します。

```
attribute keeper of signal_name : signal is "{YES|NO|TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* KEEPER = " {YES|NO|TRUE|FALSE}" *)
```

UCF および NCF 構文

次の文は、ネットに対して KEEPER を使用するように I/O を設定しています。

```
NET "pad_net_name" KEEPER;
```

次の文は、KEEPER をグローバルに設定しています。

```
DEFAULT KEEPER = TRUE;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" keeper={yes|no|true |false};
END;
```

LOC

LOC (ロケーション) 制約には、次の特徴があります。

- 基本的な配置制約です。
- 合成制約です。

LOC (ロケーション) 制約の詳細は、「LOC の詳細」を参照してください。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

どのデバイス ファミリにどのエレメントを使用できるかは、ライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

適用ルール

- ネットまたは信号には設定できませんが、ネットまたは信号がパッドに接続されている場合は例外です。この場合、LOC はパッド インスタンスに設定されていると見なされます。
- CPLD の場合、ネットまたは信号を駆動するすべての適用可能なエレメントに設定できます。
- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

制約値

location

パーツ タイプに対して有効な位置

構文例

次は 1 つのロケーションの場合の構文です。

```
INST "instance_name"  LOC=location;
```

FPGA デザインで各種ロジック エレメントに設定できる配置制約の例は、この制約の FPGA デバイス用の構文か、RLOC 制約の項目を参照してください。有効なエレメントには、次が含まれます。

- フリップフロップ
- ROM
- RAM
- ブロック RAM
- FMAP
- BUFT
- CLB
- IOB
- I/O
- エッジ デコーダー
- グローバル バッファー

詳細は、「LOC の制約例」を参照してください。

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- インスタンスに設定します。
- 属性名

  LOC
- 属性値

  value

有効な値については、この制約の FPGA デバイスおよび CPLD デバイスの構文をそれぞれ参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute loc: string;
```

VHDL 制約を次のように指定します。

```
attribute loc of {signal_name| label_name}: {signal |label} is "location";
```

バスに LOC を設定するには、次のように指定します。

```
attribute loc of  bus_name : signal is " location_1
location_2 location_3...";
```

CPLD デバイスの場合、バスの一部のみに LOC を設定するには、次のように指定します。

```
attribute loc of bus_name  : signal is "* * location_1
 * location_2...";
```

location の詳細は、この制約の FPGA デバイスおよび CPLD デバイスの構文をそれぞれ参照してください。

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* LOC = " location" *)
```

バスに LOC を設定するには、次のように指定します。

```
(* LOC = "location_1 location_2 location_3... " *)
```

CPLD デバイスの場合、バスの一部のみに LOC を設定するには、次のように指定します。

```
(* LOC = " * *location_1 location_2..."  *)
```

location の詳細は、この制約の FPGA デバイスおよび CPLD デバイスの構文をそれぞれ参照してください。

UCF および NCF 構文

- **INST "/FLIP_FLOPS/*" LOC=SLICE_X*Y8;**

  次の文は、FLIP_FLOPS の下にある各インスタンスを列 8 のいずれかの CLB に配置するように指定します。

- **INST "MUXBUF_D0_OUT" LOC=P110;**

  次の文は、MUXBUF_D0_OUT のインスタンスを IOB の位置 P110 に配置するように指定します。

- **NET "DATA<1>" LOC=P111;**

  次の文は、NET DATA<1> を IOB の P111 位置からパッドに接続するように指定します。

XCF 構文

```
BEGIN MODEL " entity_name"
 PIN "signal_name"  loc=string ;
 INST "instance_name" loc=string ;
END;
```

PCF 構文

LOC を指定すると、LOCATE 制約が PCF ファイルに書き込まれます。詳細は、「LOCATE」を参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PACE の設定

主としてロケーション制約を I/O コンポーネントに設定する際に使用します。I/O 規格など特定の I/O プロパティの設定にも使用できます。PACE は、Project Navigator の [Processes] ウィンドウからアクセスできます。

CPLD でのみサポートされます。FPGA ではサポートされません。

詳細は、PACE ヘルプの「手順」セクションのエリア制約およびピンの編集に関する部分を参照してください。

LOCATE

LOCATE (ロケート) 制約には、次の特徴があります。

- 基本的な配置制約です。
- 次のいずれかを指定します。
  - 単一の位置
  - 複数の位置
  - 位置の範囲

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

- CLB
- IOB
- DCM
- クロック ロジック
- マクロ

適用ルール

- マクロに設定されている場合、そのマクロのすべてのエレメントに適用されます。
- プリミティブに設定されている場合は、プリミティブ全体に適用されます。

制約値

- site_name

  コンポーネント サイト :
  - CLB の位置

    または
  - IOB の位置
- site_item
  - **SITE** "*site_name*"

    または
  - **SITEGRP** "*site_group_name*"
- LEVEL *n* の *n* は次のいずれかになります。

  0、1、2、3、4

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

単一の位置または複数の位置の PCF 構文

```
COMP "comp_name" LOCATE=[SOFT] "site_item1"... "site_itemn" [LEVEL n];
COMPGRP "group_name" LOCATE=[SOFT] "site_item1"... "site_itemn" [LEVEL n];
MACRO name LOCATE=[SOFT] "site_item1" "site_itemn" [LEVEL n];
```

位置の範囲の PCF 構文

```
COMP "comp_name" LOCATE=[SOFT] SITE "site_name" : SITE "site_name" [LEVEL n];
COMPGRP "group_name" LOCATE=[SOFT] SITE "site_name" : SITE "site_name" [LEVEL n];
MACRO "macro_name" LOCATE=[SOFT] SITE "site_name" : SITE "site_name" [LEVEL n];
```

LOC の詳細

LOC (ロケーション) 制約には、次の特徴があります。

- 基本的な配置制約です。
- 合成制約です。

FPGA デバイスの場合

LOC は、FPGA チップ上のデザイン エレメントの絶対配置を指定します。絶対位置には、単一の位置、位置の範囲、または複数の位置のリストを指定できます。LOC はデザイン ファイルから指定できるほか、制約ファイルの構文を使用して直接配置することもできます。

同じシンボルに対して複数の位置を指定するには、各位置をカンマで区切ります。これにより、指定された位置のいずれにもシンボルを配置できるようになります。また、デザインエレメントまたはデザイン エレメントのグループを配置する領域も指定できます。

PlanAhead™ または FPGA Editor を使用すれば、有効なサイト名を簡単に識別できます。有効となる名前は、ターゲットとなるデバイス タイプのファンクションです。ターゲット ロケーションを指定する構文を検索するには、FPGA Editor に空のデバイスをロードします。いずれかのブロックにカーソルを合わせてクリックすると、FPGA Editor のヒストリ エリア内で特定ブロックの位置を表示できます。位置に .I、.O、.T などのピン名を含めることはできません。

LOC 制約は、複数の CLB、IOB、ソフト マクロなどのシンボルを使用するロジックに適用できます。LOC 制約をソフト マクロ シンボルに使用すると、位置の情報が下位レベルのロジックに渡されます。ロケーション制約は、LOC が有効な下位レベルのすべてのブロックに自動的に適用されます。

FPGA デバイスでは、スライス レベルでデカルト座標をベースとした XY 指定を使用します。スライスで位置を指定するには、次の形式を使用します。

SLICE_X*m*Y*n*

チップの左下にある CLB を XY 座標の基点 X0Y0 とします。X 値および Y 値は、CLB ごとに 2 つずつあります。X 値は 0 で始まり、CLB の行を右方向に向かって増えていきます。Y 値も 0 で始まり、CLB の列を上方向に向かって増えていきます。

XY 座標でスライスを指定する構文例については、下記の「単一の位置を指定する LOC 制約の例」を参照してください。

FPGA のブロック RAM、乗算器を指定する場合にも SLICE とは異なる独自の仕様になります。したがって、location の値は SLICE、RAMB、または MULT で始める必要があります。

- RAMXB16_X2Y3 にあるブロック RAM は、SLICE_X2Y3 にあるフリップフロップと同じサイトには配置されません。
- また、MULT18X18_X2Y3 にある乗算器は、SLICE_X2Y3 にあるフリップフロップや RAMB16_X2Y3 にあるブロック RAM と同じサイトには配置されません。

グローバル バッファー (BUFG) と DCM エレメントの位置の値は、配置できる位置の特定の物理的なサイト名となります。

LOC 制約を使用したピンの割り当ては、OPAD8 などのバス パッド シンボルではサポートされていません。

FPGA デバイスのロケーション指定の種類

エレメント タイプ	位置の例	説明
IOB	P12	IOB の位置 (チップ キャリア)
	A12	IOB の位置 (ピン グリッド)
	B、L、T、R	Spartan®-3、Spartan-3A、Spartan-3E デバイスの場合、IOB に適用され、エッジの位置 (下、左、上、右) を示します。
	LB、RB、LT、RT、BR、TR、BL、TL	Spartan-3、Spartan-3A、Spartan-3E デバイスの場合、IOB に適用され、ハーフ エッジの位置 (左下、右下など) を示します。
	Bank#	IOB コンポーネントに適用され、すべての FPGA のバンクを示します。
スライス	SLICE_X22Y3	すべての FPGA の SLICE_X22Y3 スライスの位置
ブロック RAM	RAMB16_X2Y56	Spartan®-3、Spartan-3A、Spartan-3E デバイスのブロック RAM の位置
	RAMB36_X2Y56	Virtex®-5 デバイスのブロック RAM の位置
乗算器	MULT18X18_X#Y#	Spartan-3 および Spartan-3A デバイスの乗算器の位置
	DSP48_X#Y#	Virtex-4 および Virtex-5 デバイスの乗算器の位置
デジタル クロック マネージャー	DCM_X#Y#	Spartan-3、Spartan-3A、Spartan-3E デバイスのデバイスのデジタル クロック マネージャ
	DCM_ADV_X#Y#	Virtex-4 および Virtex-5 デバイスのデバイスのデジタル クロック マネージャ
位相ロック ループ	PLL_ADV_X#Y#	すべての FPGA の位相ロック ループ

次のように、ワイルドカード文字を使用すると、位置の範囲を指定できます。

SLICE_X*Y5	Y 座標が 5 にある FPGA デバイスのすべてのスライス

FPGA のグローバル バッファー、グローバル パッド、DCM の位置を表すワイルドカード文字はサポートされていません。

CPLD デバイスの場合

CPLD デバイスでは、LOC=**pin_name** 制約を PAD シンボルまたはパッド ネットに設定し、信号を特定ピンに割り当てます。PAD シンボルは、次のとおりです。

- IPAD
- OPAD
- IOPAD
- UPAD

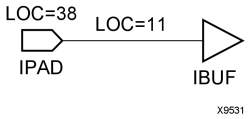
インスタンスがコラプスされていない場合、そのインスタンスまたはその出力ネットに対して **LOC=FB***nn* 制約を使用すると、特定のファンクション ブロックまたはマクロセルにロジックまたはレジスタを割り当てることができます。

すべての内部インスタンスまたは出力パッドの **LOC=FB** *nn_mm* 制約は、対応するロジックを CPLD 内の特定のファンクション ブロックまたはマクロセルに割り当てます。LOC が、マクロセルにマップされていないシンボルに設定されていたり、最適化により削除された場合、LOC は無視されます。

LOC の優先順位

隣接する 2 つの LOC を入力パッドとそれに隣接するネットに設定する場合、ネットに設定された LOC が優先されます。たとえば、LOC=11 は LOC=38 よりも優先されます。

LOC の優先順位の例



LOC の制約例

このセクションでは、LOC 制約例を具体的に示します。

- 単一の位置を指定する LOC 制約の例
- DCM (デジタル クロック マネージャー) の LOC 制約の例
- フリップフロップの LOC 制約の例
- I/O の LOC 制約の例
- IOB の LOC 制約の例
- マップ の LOC 制約の例 (FMAP)
- ROM の LOC 制約の例
- ブロック RAM の LOC 制約の例
- スライスの LOC 制約の例

単一の位置を指定する LOC 制約の例

制約 (UCF 構文)	デバイス	説明
INST “*instance_name*” LOC=P12;		I/O を位置 P12 に配置します。
INST “*instance_name* “ LOC=SLICE_X3Y2;	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5	スライスの XY 座標上にあるスライス X3Y2 にロジックを配置します。
INST “*instance_name* “ LOC=RAMB16_X0Y6;	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4	RAMB の XY 座標上にある RAMB16_X0Y6 にあるブロック RAM にロジックを配置します。
INST “*instance_name* “ LOC=MULT18X18_X0Y6;	Spartan-3 および Spartan-3A	MULT の XY 座標上にある MULT18X18_X0Y6 にある乗算器にロジックを配置します。
INST “*instance_name* “ LOC=FIFO16_X0Y15;	Virtex-4	FIFO の XY 座標上の FIFO16_X0Y15 にある FIFO にロジックを配置します。
INST “*instance_name* “ LOC=IDELAYCTRL_X0Y3;	Virtex-4 および Virtex-5	IDELAYCTRL_X0Y3 にある IDELAYCTRL にロジックを 配置します。

次は、複数ロケーションの場合の構文です。

LOC= *location1*,*location2* ,...,*locationx*

個々の制約をカンマで区切ることで、1 つのエレメントに複数の位置を指定できます。複数の位置を指定した場合、PAR は指定された位置のいずれかを使用することもできます。複数の LOC 制約の例は、次のとおりです。

複数の位置を指定する LOC 制約の例

制約	デバイス	説明
INST “*instance_name* “ LOC=SLICE_X2Y10, SLICE_X1Y10;	FPGA	スライスの XY 座標上にある SLICE_X2Y10 または SLICE_X1Y10 にロジックを配置します。

現在のところ、単一の制約を使用して、複数のエレメントを単一の位置または複数の位置に配置できません。

次はロケーション範囲の構文です。

INST " *instance_name*" **LOC=***location* **:***location* {**SOFT** };

ボックスの 2 隅を指定することで、範囲を定義できます。Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 以外のアーキテクチャの場合、ロジックが配置される領域の左上隅および右下隅を指定します。FPGA デバイスの場合、左下隅と右上隅を指定します。指定した 2 隅は、コロン (:) で区切ります。

シンボルで表したロジックは、ボックス内のいずれかの位置に配置されます。デフォルトでは、制約は「ハード」条件と解釈され、ボックス内に配置されます。SOFT を指定すると、ボックス外の位置の方がより良い結果が得られるような場合、PAR は制約を別の位置に配置します。範囲を指定する LOC 制約の例は、次のとおりです。

位置の範囲を指定する LOC 制約の例

制約	デバイス	説明
INST "*instance_name* " LOC=SLICE_X3Y5:SLICE_X5Y20;	FPGA	スライスの XY 座標上の SLICE_X3Y5 (左下隅) または SLICE_X5Y20 (右上隅) で囲まれた矩形領域にあるいずれかのスライスにロジックを配置します。

LOC の範囲には、SOFT キーワードも使用できます。AREA_GROUP とは異なり、LOC はシンボルのパックに影響を与えません。厳密に言えば、LOC は PAR で使用される配置制約です。

次は、CPLD デバイスの LOC 構文です。

INST "*instance_name*" **LOC=***pin_name*;

または

INST " *instance_name*" **LOC=FB***ff* ;

または

INST "*instance_name*" **LOC=FB** *ff_mm*;

- *pin_name* は、数値のピン名の場合は P*nn*、または行/列のピン名の場合は *rc* です。
- *ff* は、ファンクション ブロックの番号です。
- *mm* は、ファンクション ブロック内のマクロセルの番号です。

FB*ff* および **FB***ff_mm* の 2 つの制約フォーマットは、出力および双方向ピンにのみ適用でき、入力ピンには適用できません。

1 つ目の制約フォーマットは次のようになります。

INST "*instance_name***" LOC=***pin_name***;**

このフォーマットはすべての IO タイプに適用できます。

DCM (デジタル クロック マネージャー) の LOC 制約の例

このセクションは、すべての FPGA デバイスに該当します。

DCM は UCF ファイルでロックできます。構文は、次のようになります。

INST "*instance_name*" **LOC = DCM_XY***B*;

すべての Spartan デバイス

INST "*instance_name*" LOC = DCM_ADV_XY*B*;

Virtex-4 および Virtex-5 デバイス

A は左下隅の 0 から始まる X 軸で、デバイスの右に向かって値は増加します。

B は左下隅の 0 から始まる Y 軸で、デバイスの上に向かって値は増加します。

例

```
INST "myinstance" LOC = DCM_X0Y0;
```

フリップフロップの LOC 制約の例

フリップフロップ制約は、回路図または UCF ファイルで割り当てることができます。

回路図の場合は、フリップフロップに LOC 制約を設定します。制約は EDIF ネットリストに渡され、デザインのマップ後 PAR によって読み出されます。

次に、LOC 制約を回路図および UCF に適用する方法を示します。UCF の例にある /top-12/fdrd および /top-54/fdsd は、フリップフロップのインスタンス名です。

スライスをベースとした XY 座標で指定する場合

スライスをベースとした XY 座標で指定できるのは Spartan-3 および Virtex-4 以降のアーキテクチャのみです。

フリップフロップ制約は、特定のスライス、スライスの範囲、スライスの行および列に設定されます。

フリップフロップの LOC 制約の例 1

回路図	LOC=SLICE_XlY5
UCF	INST “/top-12/fdrd” LOC=SLICE_X1Y5;

フリップフロップを SLICE_X1Y5 に配置します。SLICE_X0Y0 はデバイスの左下隅になります。

フリップフロップの LOC 制約の例 2

左下隅の SLICE_X1Y1 と右上隅の SLICE_X5Y7 で囲まれる矩形領域にフリップフロップを配置します。

回路図	LOC=SLICE_R1C1:SLICE_R5C7
UCF	INST “/top-12/fdrd” LOC=SLICE_X1Y1:SLICE_X5Y7;

フリップフロップの LOC 制約の例 3

Y 座標が 3 のスライスのすべての行にフリップフロップを配置します。X 値または Y 値の代わりにワイルドカード文字を使用すると、スライスの行全体または列全体を指定できます。

回路図	LOC=SLICE_X*Y3
UCF	INST “/top-12/fdrd/top-54/fdsd” LOC=SLICE_X*Y3;

フリップフロップの LOC 制約の例 4

SLICE_X2Y4 または SLICE_X7Y9 にフリップフロップを配置します。

回路図	LOC=SLICE_X2Y4,SLICE_X7Y9
UCF	INST “/top-54/fdsd” LOC=SLICE_X2Y4, SLICE_X7Y9;

例 4 の場合、各 LOC 制約をカンマで区切りながら繰り返して使用することで、1 つのエレメントに複数の位置を指定できます。複数の位置を指定した場合、PAR は指定された位置のいずれかを使用することもできます。

フリップフロップの LOC 制約の例 5

X 座標が 5 のスライスの列にフリップフロップを配置しないようにしてください。X 値または Y 値の代わりにワイルドカード文字を使用すると、スライスの行全体または列全体を指定できます。

回路図	PROHIBIT=SLICE_X5Y*
UCF	CONFIG PROHIBIT=SLICE_X5Y*;

I/O の LOC 制約の例

I/O 制約は、特定の IOB に設定できます。I/O 制約は、回路図から設定するか、または UCF ファイルを介して設定できます。

回路図の場合は、ターゲットのパッド シンボルに LOC 制約を設定します。制約はネットリスト ファイルに渡され、マップ後に PAR によって読み出されます。

UCF ファイルの場合は、一意なインスタンス名でパッドを識別します。次に、回路図と UCF (ユーザー制約ファイル) に LOC 制約を設定する方法を示します。この例では、I/O のインスタンス名は /top-102/data0_pad と /top-117/q13_pad です。さらに、ピン番号を使用して 1 つのピンにロックしています。

回路図	LOC=P17
UCF	INST “/top-102/data0_pad” LOC=P17;

ピン 17 の IOB に I/O を配置します。ピン グリッド アレイの場合は、B3 や T1 などのピン名を使用します。

IOB の LOC 制約の例

I/O パッド、バッファー、レジスタは、個々の IOB の位置に割り当てることができます。IOB の位置は、対応するパッケージ ピンの指定によって識別されます。

IOB 制約は、次のような形式で使用できます。LOC= にピンの位置を指定してください。INFF8 などのように、シンボルがソフト マクロを表し、そのマクロに I/O エレメントのみが含まれる場合、マクロに含まれるすべての I/O エレメントに LOC 制約が適用されます。指定した I/O エレメントが指定した位置に収まらない場合は、エラーが発生します。

次の文は、I/O エレメントを位置 P13 に配置するように指定します。PGA パッケージの場合は、B3 などのように文字と数字で指定します。

INST “ *instance_name* ” LOC=P13;

マップで特定の IOB を使用しないように設定できます。これにより、半専用のコンフィギュレーション ピンからユーザー I/O 信号を分離させることもできます。このような PROHIBIT 制約は、UCF ファイルでのみ割り当てることができます。

次の例に示すように、IOB コンポーネントは PROHIBIT 制約の前に CONFIG キーワードを付けて禁止できます。

回路図	None
UCF	CONFIG PROHIBIT=p36, p37, p41;

IOB コンポーネントのピン 36、37、41 にはユーザー I/O を配置しません。ピン グリッド アレイの場合は、D14、C16、H15 などのピン名を使用します。

マップ の LOC 制約の例 (FMAP)

マップ制約は、CLB コンポーネントへのロジックのマップを制御します。この制約は 2 種類に分類されます。1 つ目は、回路図に配置する FMAP で、2 つ目は、回路図または制約ファイルに配置できる LOC 制約です。

FMAP は、ファンクション ジェネレーターへのロジックのマップを制御します。このシンボルは回路図上のロジックを定義しませんが、回路図上の別の場所にあるロジック部分をファンクション ジェネレーターにマップする方法を指定します。

FMAP シンボルは、4 入力の F ファンクション ジェネレーターへのマップを定義します。

FMAP シンボルの場合、CLBMAP プリミティブと同じように、LOC 制約のほかに MAP=PUC および MAP=PUO がサポートされています (現在、ピンのロックはサポートされていません。MAP=PLC と MAP=PLO は、それぞれ MAP=PUC と MAP=PUO に変換されます)。

マップ制約 (FMAP) 例 1

回路図	LOC=SLICE_X7Y3
UCF	INST “$1I323”LOC=SLICE_X2Y4, SLICE_X3Y4;

FMAP シンボルを行 7、列 3 の SLICE に配置します。

マップ制約 (FMAP) 例 2

回路図	LOC=SLICE_X2Y4, SLICE_X3Y4
UCF	INST “top/dec0011” LOC=CLB_R2C4,CLB_R3C4;

FMAP シンボルを行 2、列 4 または行 3、列 4 のいずれかにある SLICE に配置します。

マップ制約 (FMAP) 例 3

回路図	LOC=SLICE_X5Y5:SLICE_X10Y8
UXCF	INST “$3I27” LOC=SLICE_X5Y5:SLICE_X10Y8;

FMAP シンボルを左上隅の SLICE X5Y5 と右上隅の SLICE X10Y8 で囲まれた領域に配置します。

乗算器の LOC 制約の例

このセクションは、すべての FPGA デバイスに該当します。

乗算器制約は、回路図または UCF ファイルで割り当てることができます。回路図の場合は、LOC 制約を乗算器シンボルに設定します。制約はネットリスト ファイルに渡され、マップ後に PAR によって読み出されます。LOC 制約の適用方法の詳細は、そのアプリケーションのユーザー ガイドを参照してください。制約ファイルの場合、乗算器は一意なインスタンス名によって識別されます。

FPGA の乗算器は、スライス、ブロック RAM の場合とは異なる XY 座標仕様で指定します。

- Spartan-3、Spartan-3A、および Spartan-3E デバイスは、MULT18X18_X#Y# を使用して指定されます。
- Virtex-4 および Virtex-5 デバイスは、DSP48_X#Y# を使用して指定されます。ここでは、XY 座標の値が乗算器のグリッド アレイに相当します。

MULT18X18_X0Y1 にある乗算器は、SLICE_X0Y1 にあるフリップフロップや RAMB16_X0Y1 にあるブロック RAM と同じサイトには配置されません。

たとえば、2 列の乗算器があるデバイスがあるとします。 各列には 2 つの乗算器が含まれ、一方の列はチップの右側に、もう一方の列は左側にあります。左下隅にある乗算器は MULT18X18_X0Y0 です。乗算器は 2 列しかないため、右上隅にある乗算器は MULT18X18_X1Y1 です。

回路図	LOC=MULT18X18_X0Y0
UCF	INST “/top-7/rq” LOC=MULT18X18_X0Y0;

ROM の LOC 制約の例

メモリ制約は、回路図または UCF ファイルで割り当てることができます。

回路図の場合は、LOC 制約をメモリ シンボルに設定します。制約はネットリスト ファイルに渡され、マップ後に PAR によって読み出されます。LOC 制約の適用方法の詳細は、そのアプリケーションのユーザー ガイドを参照してください。

制約ファイルの場合、メモリは一意なインスタンス名によって識別されます。タイプが ROM のメモリ インスタンスを入力ファイルで 1 つ以上指定できます。16 X 1 または 32 X 1 より大きいすべてのメモリ マクロは、ネットリスト ファイル内でこれらの基本エレメントに分割されます。

次の例では、ROM プリミティブのインスタンス名は /top-7/rq です。

スライスをベースとした XY 座標で指定する場合

スライスをベースとした XY 座標で指定できるのは Spartan-3 および Virtex-4 以降のアーキテクチャです。ROM 制約は、特定のスライス、スライスの範囲、スライスの行または列に設定できます。

ROM の LOC 制約の例 1

回路図	LOC=SLICE_X1Y1
UCF	INST “/top-7/rq” LOC=SLICE_X1Y1;

SLICE_X1Y1 にメモリを配置します。SLICE_X1Y1 はデバイスの左下隅になります。16 X 1 または 32 X 1 メモリには、SLICE 制約を 1 つだけ設定できます。

ROM の LOC 制約の例 2

回路図	LOC=SLICE_X2Y4, SLICE_X7Y9
UCF	INST “/top-7/rq” LOC=SLICE_X2Y4, SLICE_X7Y9;

SLICE_X2Y4 または SLICE_X7Y9 のいずれかにメモリを配置します。

ROM の LOC 制約の例 3

回路図	PROHIBIT SLICE_X5Y*
UCF	CONFIG PROHIBIT=SLICE_X5Y*;

X 座標が 5 のスライスの列にメモリを配置しません。X 値または Y 値の代わりにワイルドカード文字を使用すると、スライスの行全体または列全体を指定できます。

ブロック RAM の LOC 制約の例

このセクションは、すべての FPGA デバイスに該当します。

ブロック RAM 制約は、回路図または UCF ファイルで割り当てることができます。回路図の場合は、LOC 制約をブロック RAM シンボルに設定します。制約は、ネットリスト ファイルに渡され、マップ後に PAR によって読み込まれます。LOC 制約の適用方法の詳細は、そのアプリケーションのユーザー ガイドを参照してください。制約ファイルの場合、メモリは一意なインスタンス名によって識別されます。

Spartan-3 以降のデバイス

FPGA のブロック RAM は、スライス、乗算器の場合とは異なる XY 座標仕様で指定します。具体的には、RAMB16_X*m*Y*n* を使用します。ここでは、XY 座標の値がブロック RAM のグリッド アレイに相当します。RAMB16_X0Y1 にあるブロック RAM は、SLICE_X0Y1 にあるフリップフロップと同じサイトには配置されません。

たとえば、2 列のブロック RAM があるデバイスがあるとします。 各列には 2 つのブロック RAM が含まれ、一方の列はチップの右側に、もう一方の列は左側にあります。左下隅にあるブロック RAM は RAMB16_X0Y0 です。ブロック RAM は 2 列しかないため、右上隅にあるブロックは RAMB16_X1Y1 になります。

回路図	LOC=RAMB16_X0Y0 (Virtex-5 以外のすべての FPGA デバイスの場合) LOC=RAMB36_X0Y0 (Virtex-5 デバイスの場合)
UCF	INST “/top-7/rq” LOC=RAMB16_X0Y0;

スライスの LOC 制約の例

このセクションは、すべての FPGA に適用されます。現在のところ、スライスをベースとした XY 座標で指定できるのはこれらのみです。

単一のスライス位置、スライス位置のリスト、または スライス位置の矩形ブロックにソフト マクロとフリップフロップを割り当てることができます。

スライスの位置は、固定位置または位置の範囲で表すことができます。固定位置を表すには、次の構文を使用します。

SLICE_X*m*Y *n*

説明 :

m および *n* は、それぞれ X 座標と Y 座標の値です。

また、ターゲットとなるデバイスで、値はスライス番号以下でなければなりません。一番上から一番下までの位置の範囲を表すには、次の構文を使用します。

SLICE_X *m*Y*n*:**SLICE_X***m*Y *n*

スライス制約の形式

スライス制約は、次のような形式で指定します。LOC= にスライスの位置を指定してください。ターゲットとなるシンボルがソフト マクロを表す場合、そのマクロに含まれるすべての該当シンボル (フリップフロップ、マップ) に対して LOC 制約が適用されます。指定したロジックが指定したブロックに収まらない場合は、エラーが発生します。

スライス制約の例 1

次の UCF 構文は、指定されたスライスにロジックを配置します。

INST “*instance_name*” LOC=SLICE_X133Y10;

スライス制約の例 2

次の UCF 構文は、スライスの最初の列にロジックを配置するように指定します。アスタリスク (*) は、ワイルドカードです。

INST “*instance_name*” LOC=SLICE_X0Y*;

スライス制約の例 3

次の UCF 構文は、指定された 3 つのスライスのすべてにロジックを配置するように指定します。LOC 制約の構文の順番に意味はありません。

INST “ *instance_name*” LOC=SLICE_X0Y3, SLICE_X67Y120, SLICE_X3Y0;

スライス制約の例 4

次の UCF 構文は、最初に指定される左下隅のスライスと 2 番目に指定される右上隅のスライスによって定義される矩形ブロック内にロジックを配置します。

INST “ *instance_name*” LOC=SLICE_X3Y22:SLICE_X10Y55;

スライス禁止制約

特定のスライスや、スライスの範囲、あるいはスライスの行または列を PAR が使用しないように指定できます。このような PROHIBIT 制約は、UCF ファイルでのみ割り当てることができます。次の例に示すように、スライスを禁止するにはデザイン レベルで PROHIBIT 制約を指定します。

スライス禁止制約の例 1

SLICE_X0Y0 にロジックを配置しません。SLICE_X0Y0 はデバイスの左下隅になります。

回路図	なし
UCF	CONFIG PROHIBIT=SLICE_X0Y0;

スライス禁止制約の例 2

左下隅の SLICE_X2Y3 と右上隅の SLICE_X10Y10 で囲まれた矩形領域にロジックを配置しません。

回路図	なし
UCF	CONFIG PROHIBIT=SLICE_X2Y3:SLICE_X10Y10;

スライス禁止制約の例 3

X 座標の 3 にあるスライスにロジックを配置しません。これは、禁止されたスライスの列を指定しています。X 座標または Y 座標にワイルドカード文字を使用すると、スライスの行全体または列全体を指定できます。

回路図	なし
UCF	CONFIG PROHIBIT=SLICE_X3Y*;

スライス禁止制約の例 4

SLICE_X2Y4 または SLICE_X7Y9 のいずれにもロジックを配置しません。

回路図	なし
UCF	CONFIG PROHIBIT=SLICE_X2Y4, SLICE_X7Y9;

LOCK_PINS

LOCK_PINS (ロック ピン) 制約には、次の特徴があります。

- インプリメンテーション ツールで LUT シンボルに付いているピンがスワップしないように指定します。
- CPLD デザインで既存のピン配置を維持するために使用する ISE® Design Suite のピン固定機能とは異なります。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

LUT シンボルの特定インスタンスのみに設定できます。

適用ルール

単一の LUT インスタンスのみに適用されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

VHDL 構文

VHDL 制約を次のように宣言します。

attribute lock_pins:*string*;

VHDL 制約を次のように指定します。

attribute lock_pins of *{component_name|label_name}* **:{component|label} is "all";**

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

(* LOCK_PINS = "all" *)

Verilog 構文の詳細は、「Verilog 属性」を参照してください。

UCF および NCF 構文

- 識別子を使用しない場合

  INST "XSYM1" LOCK_PINS;
- すべての属性を使用する場合

  INST "XSYM1" LOCK_PINS='ALL';
- ピン割り当てリストを使用する場合

  INST I_589 LOCK_PINS=I0:A2;

  INST I_894 LOCK_PINS=I3:A1,I2:A4;

  INST tvAgy LOCK_PINS=I0:A4,I1:A3,I2:A2,I3:A1;

LUTNM

LUTNM (ルックアップ テーブル名) 制約には、次の特徴があります。

- 論理シンボルのグループ化の制御に使用されます。グループ化されたシンボルは、Virtex®-5 デバイスの LUT サイトにまとめられます。
- 値は文字列で、条件を満たす 2 つのシンボルに設定されます。
- デザイン内では 2 つのシンボルのみに使用されます。シンボルは、SLICE コンポーネント内で共有された LUT サイト内でインプリメントされます。
- この制約の機能は BLKNM 制約に類似しています。

アーキテクチャ サポート

Virtex-5

適用可能エレメント

次に設定できます。

- デザイン内で重複しない 2 つのシンボル
- 両方のシンボルの、重複しない入力ピン数の合計が 5 を超えない場合は、5 入力以下のファンクション ジェネレータ シンボル (LUT、ROM または RAM) 2 つに設定できます。
- 次のような場合、6 入力の読み込み専用ファンクション ジェネレータ シンボル (LUT6、ROM64) と 5 入力の読み込み専用シンボル (LUT5、ROM32) に設定できます。
  - 両方のシンボルの、重複しない入力ピン数の合計が 6 入力を超えない場合および
  - 6 入力シンボル プログラムの下位 32 ビットは 5 入力シンボル プログラムのの 32 ビットすべてと一致している場合

適用ルール

LUTNM 制約はデザイン内で特定の 2 つのシンボルのみに適用されます。

制約値

value

2 つのエレメントのグループに選択した任意の名前になります。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なエレメントまたはシンボル タイプに設定します。
- 属性名

  LUTNM
- 属性値

  <*user_defined*>

VHDL 構文

LUTNM は、使用前にアーキテクチャ宣言と最上位 VHDL ファイルの begin 文の間で次のように宣言します。

```
attribute LUTNM: string;
```

VHDL 制約を次のように指定します。

```
attribute LUTNM of{ LUT5_instance_name}: label is "value";
```

```
architecture MY_DESIGN of top is
attribute LUTNM: string;
   attribute LUTNM of LUT5_inst1: label is "logic_group1";
   attribute LUTNM of LUT5_inst2: label is "logic_group1";
begin
-- LUT5: 5-input Look-Up Table
   -- Virtex-5
   -- Xilinx HDL Libraries Guide version 8.2i
LUT5_inst1 : LUT5
   generic map (
      INIT => X"a49b44c1")
   port map (
      O => aout, -- LUT output (1-bit)
      I0 => d(0), -- LUT input (1-bit)
      I1 => d(1), -- LUT input (1-bit)
      I2 => d(2), -- LUT input (1-bit)
      I3 => d(3), -- LUT input (1-bit)
      I4 => d(4) -- LUT input (1-bit)
   );
-- End of LUT5_inst1 instantiation
-- LUT5: 5-input Look-Up Table
   -- Virtex-5
   -- Xilinx HDL Libraries Guide version 8.2i
LUT5_inst2 : LUT5
   generic map (
      INIT => X"649d610a")
   port map (
      O => bout, -- LUT output (1-bit)
      I0 => d(0), -- LUT input (1-bit)
      I1 => d(1), -- LUT input (1-bit)
      I2 => d(2), -- LUT input (1-bit)
      I3 => d(3), -- LUT input (1-bit)
      I4 => d(4) -- LUT input (1-bit)
   );
-- End of LUT5_inst2 instantiation
END MY_DESIGN;
```

Verilog 構文

最上位の Verilog コードの port 宣言の前に、次のように記述します。

```
(* LUTNM = "value" *)
```

```
// LUT5: 5-input Look-Up Table
// Virtex-5
// Xilinx HDL Libraries Guide version 8.2i
(* LUTNM="logic_group1" *) LUT5 #(
   .INIT(32'ha49b44c1)
) LUT5_inst1 (
   .O(aout), // LUT output (1-bit)
   .I0(d[0]), // LUT input (1-bit)
   .I1(d[1]), // LUT input (1-bit)
   .I2(d[2]), // LUT input (1-bit)
   .I3(d[3]), // LUT input (1-bit)
   .I4(d[4]) // LUT input (1-bit)
);
// End of LUT5_inst1 instantiation
// LUT5: 5-input Look-Up Table
// Virtex-5
// Xilinx HDL Libraries Guide version 8.2i
(* LUTNM="logic_group1" *) LUT5 #(
   .INIT(32'h649d610a)
) LUT5_inst2 (
   .O(bout), // LUT output (1-bit)
   .I0(d[0]), // LUT input (1-bit)
   .I1(d[1]), // LUT input (1-bit)
   .I2(d[2]), // LUT input (1-bit)
   .I3(d[3]), // LUT input (1-bit)
   .I4(d[4]) // LUT input (1-bit)
);
// End of LUT5_inst2 instantiation
```

UCF および NCF 構文

出力ポートまたは双方向ポートに設定します。

```
INST "LUT5_instance_name" LUTNM="value";
```

```
INST "LUT5_inst1" LUTNM="logic_group1";
```

```
INST "LUT5_inst2" LUTNM="logic_group1";
```

MAP

MAP (マップ) 制約には、次のような特徴があります。

- 高度なマップ制約です。
- FMAP に適用すると、マップでロジックにほかのファンクションのマージを許可するか、ピンの交換を許可するかを指定できます。

ほかのファンクションのマージが許可されると、スペースが許す限り、CLB 内にほかのロジックも配置できます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

FMAP に適用されます。

適用ルール

設定したデザイン エレメントに適用されます。

制約値

値	CLB ピン	CLB	ソフトウェアで CLB のピン間の信号交換が可能か	ソフトウェアで CLB からロジックを追加または削除可能か
PUC	アンロック (U)	クローズ (C)	○	×
PUO (デフォルト)	アンロック (U)	オープン (O)	○	○
PLC	ロック (L)	クローズ (C)	×	×
PLO	ロック (L)	オープン (O)	×	○

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

INST " *instance_name*" **MAP=**[**PUC**|**PUO**|**PLC**|**PLO**] ;

現時点では、PUC と PUO のみが有効です。PLC は PUC に、PLO は PUO に変換されます。

次の構文では、ピンのスワップを許可し、オリジナルのマップで定義されたロジック以外はファンクション ジェネレーターにマップされないように設定しています。

```
INST "$1I3245/map_of_the_world" map=puc;
```

MARK_DEBUG

MARK_DEBUG (マーク デバッグ) 制約には、次のような特徴があります。

- 合成制約です。
- ChipScope™ ツールを使用したデバッグでネットをマークするために使用されます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

設定されたネットに適用されます。

適用ルール

ネットがバスの場合、MARK_DEBUG はそのバスを構成する各信号に伝播されます。

制約値

- true

  ネットは、次のように処理されます。

  - 最適化から保護されます。
  - ChipScope ツールを使用したデバッグ用にマークされます。
- false

  制約は無視されます。
- soft (XST のみ)

  ネットは XST 合成中に最適化で削除されない場合にのみ、デバッグ用にマークされます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

VHDL の構文例

VHDL 制約を次のように宣言します。

```
attribute mark_debug : string;
```

VHDL 制約を次のように指定します。

```
attribute mark_debug of signal_name : signal is "{TRUE|FALSE|SOFT}";
```

Verilog の構文例

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* mark_debug = "{TRUE|FALSE|SOFT}" *) wire wire_name;
```

XCF の構文例

```
BEGIN MODEL "entity_name"
 NET "signal_name" mark_debug = "{TRUE|FALSE|SOFT}" ;
END;
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

デバッグ用にマークされたネットは自動的に次にリストされます。

- ChipScope ツールの Unassigned Nets フォルダー
- Set Up ChipScope Wizard

MAX_FANOUT

MAX_FANOUT (最大ファンアウト) を使用すると、ネットまたは信号のファンアウト数を制限できます。

- 制約の値により、XST および MAP でネットのファンアウトが制限されます。
- この値は整数 (XST のみ) または REDUCE (MAP のみ) のいずれかにできます。

XST の MAX_FANOUT

XST のデフォルト整数値は、次の表を参照してください。MAX_FANOUT は、XST でグローバルにもローカルにも設定できます。

MAX_FANOUT のデフォルト値

デバイス	デフォルト値
Spartan®-3、Spartan-3E、Spartan-3A、Spartan-3A D	500
Virtex®-4	500
Virtex-5	100000 (10 万)

ファンアウトが大きいと配線で問題を生じることがあるため、XST ではゲートを複製したり、バッファーを挿入することでファンアウト数が制限されます。これは技術面での限界ではなく、XST の基準です。この制限は、特に 30 未満に設定されている場合などは、厳密に適用されるとは限りません。

ほとんどの場合、ファンアウトが大きいネットを駆動するゲートを複製することでファンアウト数が制限されます。ゲートを複製できない場合は、バッファーが挿入されます。これらのバッファーは、NGC ファイルで キープ (KEEP) 属性が定義されていれば、インプリメンテーションのロジック トリミングにより削除されることはありません。レジスタの複製オプションを no に設定している場合は、バッファーのみを使用してフリップフロップおよびラッチのファンアウト数が制限されます。

MAX_FANOUT は、グローバルに設定できますが、エンティティやモジュール、指定した信号ごとに設定して、最大ファンアウトを制御できます。

実際のネット ファンアウトが MAX_FANOUT 値よりも小さい場合は、MAX_FANOUT の設定方法によって XST のビヘイビアーが異なります。

- MAX_FANOUT の値を ISE® Design Suite またはコマンド ラインを使用して設定するか、特定の階層ブロックに適用した場合、XST ではこの値が基準として解釈されます。
- MAX_FANOUT を特定のネットに設定した場合は、ロジックは複製されません。ネットに設定した場合は、XST で最適なタイミング最適化が行われないことがあります。

たとえば、実際のファンアウトが 80 で、MAX_FANOUT 値が 100 に設定されたネットをクリティカル パスが通過しているとします。MAX_FANOUT を ISE Design Suite で設定している場合は、XST がタイミングを向上しようとしてネットを複製することがあります。MAX_FANOUT をネットに設定している場合は、ロジックは複製されません。

MAP の MAX_FANOUT

MAX_FANOUT 制約を使用すると、レジスタまたはゲートの両方ともまたはいずれかを複製することで、マップでファンアウトを制限することができます。このためには、次を実行する必要があります。

- マップのレジスタ複製オプション (-register_duplication) をイネーブルにします。
- MAX_FANOUT 制約をネットに個別に設定します。

マップ中に使用する場合、値には REDUCE しか使用できません。

- **MAX_FANOUT = "REDUCE"** の場合、フィットで問題を引き起こすこともなく、パフォーマンスに改良があると判断される場合に、マップでファンアウトが制限されます。
- **MAX_FANOUT = "REDUCE"** でファンアウトの削減が実際に行われたかどうかは、マップで生成される物理合成レポート (*PSR) を確認するとわかります。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

- 値が整数の場合は、グローバルに適用するか、VHDL エンティティ、Verilog モジュール、または信号に適用します。
- 値が REDUCE の場合は、信号にのみ適用します。

適用ルール

それが接続されているエンティティ、モジュールまたは信号に適用

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute max_fanout: string;
```

VHDL 制約を次のように指定します。

```
attribute max_fanout of {signal_name|entity_name}: {signal|entity} is "integer";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* max_fanout = "integer" *)
```

XCF の構文例 1

```
MODEL "entity_name" max_fanout=integer;
```

XCF の構文例 2

```
BEGIN MODEL "entity_name"
 NET "signal_name" max_fanout=integer;
END;
```

XST コマンド ライン構文

run コマンドの -max_fanout コマンド ライン オプションでグローバルに定義できます。

```
-max_fanout integer
```

ISE Design Suite の構文

ISE Design Suite で次のようにグローバルに定義します。

[Process Properties] ダイアログ ボックスの [Xilinx Specific Options] ページにある [Max Fanout]

UCF 構文

マップのレジスタ複製オプションと共に使用される場合、MAX_FANOUT 制約は UCF ファイルで次のように指定します。

```
NET "signal_name" max_fanout=REDUCE;
```

MAXDELAY

MAXDELAY (最大遅延) 属性では、ネットの最大許容遅延を定義します。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

それが接続されているネットに適用

適用ルール

ネットに適用されます。

制約値

- value

  正の整数
- units
  - ps
  - ns (デフォルト)
  - micro
  - ms
  - GHz
  - MHz
  - kHz

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットに設定します。
- 属性名MAXDELAY
- 属性値
  - value

    遅延時間
  - units
    - micro
    - ms
    - ns
    - ps

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute maxdelay: string;
```

VHDL 制約を次のように指定します。

```
attribute maxdelay of signal_name: signal is "value [units]";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(*MAXDELAY = "value [units]" *)
```

UCF および NCF 構文

```
NET "net_name" MAXDELAY=value units;
```

次の文は、10 ナノ秒の最大遅延をネット $SIG_4 に指定します。

```
NET "$1I3245/$SIG_4" MAXDELAY=10 ns;
```

PCF 構文

```
item MAXDELAY = maxvalue [PRIORITY integer];
```

- item
  - ALLNETS
  - NET *name*
  - TIMEGRP *name*
  - ALLPATHS
  - PATH *name*
  - *path specification*
- maxvalue
  - 時間を表す数字 (単位 : micro、ms、ps、ns)
  - 周波数を表す数字 (単位 : GHz、MHz、KHz)
  - TSidentifier

Constraints Editor の設定

Constraints Editor を起動するには、次を実行します。

[Processes] → [User Constraints] → [Exceptions] → [Nets] をクリックします。

FPGA Editor の設定

すべてのパスまたはネットに制約を設定するには、次を実行します。

[File] → [Main Properties] → [Global Physical Constraints] をクリックします。

選択したパスまたはネットに制約を設定するには、配線済みネットを選択して、次を実行します。

[Edit] → [Properties of Selected Items] → [Physical Constraints] をクリックします。

MAXPT

MAXPT (最大積項) 制約には、次の特徴があります。

- 高度な制約です。
- CPLD デバイスのみサポートされます。
- 制約を設定したノード内でロジックをコラプスする際にフィルターで使用できる積項の最大数を指定できます。
- ISE® Design Suite の [Collapsing Pterm Limit] で設定されている値より優先されます。

アーキテクチャ サポート

CPLD デバイスのみサポートされます。FPGA デバイスのサポートなし

適用可能エレメント

信号に適用

適用ルール

それが接続されている信号に適用

制約値

integer

正の整数

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute maxpt: integer;
```

VHDL 制約を次のように指定します。

```
attribute maxpt of signal_name : signal is "integer";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* MAXPT = "integer" *)
```

UCF および NCF 構文

```
Net "signal_name" maxpt=integer;
```

MAXSKEW

MAXSKEW (最大スキュー) 制約には、次の特徴があります。

- タイミング制約です。
- ネット上の最大スキューを制御するために使用します。
- 次のスキューを制御します。
  - ローカル クロック
  - グローバル クロック ネットワークに含まれないクロック
- グローバル クロック ネットワークには必要ありません。

  グローバル クロック ネットワークには MAXSKEW 制約を使用しないようにしてください。

スキュー

「スキュー」とは、ネットで駆動されるすべてのロードの遅延の差です。

この制約では、ネットで駆動されるすべてのロードが識別されるので、論理的な接続のないロード間のスキューがレポートされることもあります。

ネットで許容される最大スキューを制御するには、MAXSKEW をネットに直接適用します。

詳細は、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

ネットに適用されます。

適用ルール

それが接続されているネットに適用

制約値

- allowable_skew

  タイミング要件
- units
  - ms
  - micro
  - ns (デフォルト)
  - ps

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットに設定します。
- 属性名

  MAXSKEW
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute maxskew: string;
```

VHDL 制約を次のように指定します。

```
attribute maxskew of signal_name  : signal is
"allowable_skew  [units]";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* MAXSKEW = "allowable_skew [units] " *)
```

UCF および NCF 構文

```
NET " net_name" MAXSKEW=allowable_skew [units];
```

次の文は、3ns の最大スキューをネット $SIG_6 に指定します。

```
NET "$1I3245/$SIG_6" MAXSKEW=3 ns;
```

Constraints Editor の構文

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

FPGA Editor の構文

FPGA Editor で制約は次の方法で設定できます。

[Edit] → [Properties of Selected Items] をクリックします。

配線済みのネットを選択し、[Physical Constraints] タブから MAXSKEW を設定します。

MCB_PERFORMANCE

MCB_PERFORMANCE (MCB パフォーマンス) 制約には、次のような特徴があります。

- Spartan®-6 デバイスにのみ適用できます。
- UCF にのみ設定できます。
- メモリ コントローラー ブロック (MCB) のパフォーマンス レベルを指定します。

Spartan-6 メモリ コントローラー ブロック (MCB)

- Spartan-6 の メモリ コントローラー ブロック (MCB) では、V_{CCINT} 電源の電圧設定と状況によって 2 つのパフォーマンスをターゲットにできます。
- パフォーマンス ターゲットの詳細は、デバイスのデータシートを参照してください。
- ザイリンクスの ISE® Design Suite ツールで MCB パフォーマンスを指定するには、UCF ファイルで MCB_PERFORMANCE コンフィギュレーション制約を指定します。

VCCINT 電圧設定

- UCF とタイミング ツールの V_{CCINT} 電圧設定、および ISE Design Suite ツールのレポートの VCCINT 電圧設定は、この設定の影響を受けません。
- MCB_PERFORMANCE 設定と異なる電圧がレポートされることがあります。
- タイミング解析をするには、電圧要件をこの属性と電圧設定の両方に基づいて正しく設定する必要があります。

アーキテクチャ サポート

この制約は、Spartan-6 デバイスにのみ適用できます。

適用ルール

なし

制約値

- なし

  MCB_PERFORMANCE が指定されない場合、デフォルトは STANDARD です。
- STANDARD

  デバイスのデータシートで示されるように、MCB を標準パフォーマンスにし、V_{CCINT} を全電圧範囲に設定するには、UCF で STANDARD を指定する必要があります。

  ```
  CONFIG MCB_PERFORMANCE= STANDARD;
  ```
- EXTENDED
  - MCB をより高速のパフォーマンスにするには、UCF で EXTENDED を指定します。

    ```
    CONFIG MCB_PERFORMANCE=EXTENDED;
    ```
  - EXTENDED を使用する場合は、決まった電圧要件に従う必要があります。
  - 詳細は、デバイスのデータシートを参照してください。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG MCB_PERFORMANCE=[STANDARD|EXTENDED];
```

MIODELAY_GROUP

MIODELAY_GROUP (MIODELAY グループ) 制約には、次の特徴があります。

- デザイン インプリメンテーション制約です。
- 2 つ以上の IODELAY_GROUP を 1 つのマスタ IODELAY_GROUP にまとめ、IDELAYCTRL を自動的に複製して配置させることができます。

アーキテクチャ サポート

MIODELAY_GROUP は Virtex®-4 および Virtex-5 デバイスに適用されます。Virtex-4 の場合、この制約は MAP の[Timing Driven Pack and Placement] オプションを使用した場合にのみサポートされます。

適用可能エレメント

MIODELAY_GROUP は、定義済みの複数の IODELAY_GROUP に適用できます。

適用ルール

MIODELAY_GROUP は、既存の IODELAY_GROUP に適用されます。元の IODELAY_GROUP に属するデザイン エレメントすべてに適用されます。ネット、信号、またはピンには設定できません。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

MIODELAY_GROUP "*master_group_name*" = *iodelay_group1 iodelay_group2* ... ;

説明 :

- *master_group_name*
  - 定義されたマスタ グループを表しています。
  - *iodelay_group1* と *iodelay_group2* のエレメントがすべて含まれます。
- *iodelay_group1* と *iodelay_group2* は、既に定義済みの IODELAY グループです。

NODELAY

NODELAY (遅延なし) 制約には、次の特徴があります。

- 高度なマップ制約です。
- 入力遅延を削除します。
- 次に適用できます。
  - I/O シンボル
  - 次の特殊ファンクションにアクセスするシンボル :
    - TDI
    - TMS
    - TCK

入力遅延の削除

IOB フリップフロップのデフォルト コンフィギュレーションには、入力データ パスで外部ホールド タイムになる入力遅延が含まれます。この遅延を削除するには、入力フリップフロップまたはラッチに NODELAY 制約を設定します。この場合、セットアップ タイムは小さくなりますが、ホールド タイムは正の値になります。

Spartan®-3、Spartan-3A、Spartan-3E では、デフォルトのコンフィギュレーションに入力遅延エレメントが含まれています。

NODELAY 制約を適用するには、**IOBDELAY=NONE** を使用することをお勧めします。

詳細は、「IOBDELAY」を参照してください。

アーキテクチャ サポート

- Spartan-3
- Spartan-3A
- Spartan-3E

適用可能エレメント

- 入力レジスタに適用されます。
- NODELAY は、UCF ファイル内のパッド コンポーネントに接続されているネットにも設定できます。

  ネットに設定した制約は、NGDBuild により NGD ファイル内のパッド インスタンスに挿入され、マップで処理されます。
- これには、次の UCF 構文を使用します。

```
NET " net_name" NODELAY;
```

適用ルール

- ネットまたは信号がパッドに接続されている場合を除いて、ネットや信号に設定できません。この場合、NODELAY はパッド インスタンスに設定されているものと見なされます。
- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  NODELAY
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute nodelay: string;
```

VHDL 制約を次のように指定します。

```
attribute nodelay of {component_name|signal_name|label_name} : {component|signal|label}
is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* NODELAY = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

- **INST "$1I87/inreg67" NODELAY;**

  IOB レジスタ inreg67 に入力遅延を含めないように指定しています。
- **NET "net1" NODELAY;**

  net1 に接続されているパッドに入力遅延を含めないように指定しています。

XCF 構文

```
BEGIN MODEL "entity_name "
 NET  "signal_name" nodelay=true;
 INST "instance_name" nodelay=true;
END;
```

NOREDUCE

NOREDUCE (削減なし) には、次の特徴があります。

- フィッタ制約および合成制約です。
- ロジック ハザードやレース コンディションを避けるためにデザインに含まれている冗長な論理記述が最小化されないように指定します。
- 適正なマップが実行されるように組み合わせフィードバック ループの出力ノードが識別されます。

デザイン内に組み合わせフィードバック ラッチを作成するときは、次を実行します。

- 必ず NOREDUCE をラッチの出力ネットに適用します。
- レース コンディションの回避に必要となる冗長な論理記述を含めます。

アーキテクチャ サポート

CPLD デバイスのみサポートされます。FPGA デバイスのサポートなし

適用可能エレメント

それが接続されているネットに適用

適用ルール

この制約はネット制約です。デザイン エレメントへは接続できません。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットに設定します。
- 属性名

  NOREDUCE
- 属性値
  - TRUE
  - FALSE

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute NOREDUCE: string;
```

VHDL 制約を次のように指定します。

```
attribute NOREDUCE of signal_name: signal is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* NOREDUCE = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

次の文は、ネット $SIG_12 以降にブール ロジックの減少やロジックのコラプスがないように指定します。

```
NET "$SIG_12" NOREDUCE;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" noreduce={true|false};
END;
```

OFFSET IN

注記 : OFFSET 制約には定義済みグループは使用できません。

OFFSET IN (オフセット入力) 制約には、次の特徴があります。

- FPGA デバイスへの入力インターフェイスのタイミング要件を指定します。
- FPGA デバイスの外部パッドにおけるクロックとデータのタイミング関係を指定します。

OFFSET_IN 制約を指定すると、制約の付いた同期エレメントすべてのセットアップ タイムとホールド タイムの要件が確認されます。

OFFSET IN 制約は、クロック ネット名を使用して指定します。OFFSET IN 制約の付いたクロック ネットは、外部クロック パッドになります。この制約で指定されるのは、FPGA の外部パッドのクロックとデータのタイミング関係であるため、内部クロック ネットを使用して制約を指定することはできません。ただし、同期エレメントがキャプチャされてセットアップ タイムとホールド タイムの要件が解析される際に、DCM、PLL、MMCM、IDELAY などのコンポーネントがあると、クロック パスの位相または遅延が自動的に考慮されます。また、この制約はクロック ネットワークを介して伝搬され、元の外部クロックから派生したすべてのクロックに自動的に適用されます。

OFFSET IN 制約は、デフォルトではグローバルに適用されます。この場合、特定のクロック ネットからのクロックが使用され、外部データをキャプチャする同期エレメントすべてにこの制約が適用されます。この制約が適用される同期エレメントは、入力データ パッドのサブセットのタイム グループや、同期エレメントををキャプチャするサブセットのタイム グループ、またはその両方のグループを指定することで制限できます。

詳細は、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

- グローバル
- 特定のネット
- パッド タイム グループ

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

注記 : ここでは、UCF の構文例も提供していますが、OFFSET IN 制約の指定には Constraints Editor の使用をお勧めします。

グローバルに指定する場合

OFFSET IN 制約は、デフォルトでグローバルに設定されます。この場合、入力データをキャプチャし、指定したクロック信号でトリガされる同期エレメントすべてに適用されます。

UCF 構文例 (グローバル)

```
OFFSET = IN "offset_time" [units] [VALID <datavalid_time> [UNITS]] {BEFORE|AFTER}
"clk_name" [{RISING|FALLING}];
```

PCF 構文例 (グローバル)

```
OFFSET = IN "offset_time" [units] [VALID <datavalid_time> [UNITS]] {BEFORE|AFTER}
COMP "clk_iob_name" [{RISING|FALLING}];
```

- offset_time [units]
  - キャプチャ クロック エッジとデータの開始時点のタイミングの違いです。
  - この時間の単位は、設定してもしなくてもかまいません。
  - 単位を指定しない場合、デフォルトは ns (ナノ秒) になります。
  - 指定できる値は、次のいずれかです。
    - ps
    - ns
    - micro
    - ms
- VALID <datavalid_time> [UNITS]
  - データがキャプチャされるまでの時間です。
  - この値は、入力インターフェイスの正確なホールド タイム違反を確認するために必要です。
  - この単位は、設定してもしなくてもかまいません。
  - 単位を指定しない場合、デフォルトは ns (ナノ秒) になります。
  - 指定できる値は、次のいずれかです。
    - ps
    - ns
    - micro
    - ms
- BEFORE|AFTER
  - データの開始時点とクロック エッジのタイミング関係を定義します。
  - クロックとデータの関係を定義するには、BEFORE を使用するのが最適な方法です。
  - BEFORE は、クロック エッジに対して、データが有効になる時間を示します。
    - BEFORE に正の値を指定すると、データはキャプチャ クロック エッジの前に、
    - 負の値を指定すると、後に開始されます。
  - OFFSET = IN は、**RISING** または **FALLING** 修飾子が使用されていない場合にのみ AFTER オプションと共に使用できます。
- clk_name

  入力クロック パッド ネットの階層名すべてを定義します。
- **RISING**|**FALLING**
  - クロック エッジを定義するオプションのキーワードで、データがクロックの立ち上がりエッジまたは立ち下がりエッジのどちらでキャプチャされるかが定義できます。
  - また、これらのキーワードを使用すると、DDR (デュアル データ レート) インターフェイスの立ち上がりエッジ レジスタと立ち下がりエッジ レジスタが自動的に別のグループに分けられて、解析されるようになります。

- **RISING|FALLING** キーワードは、OFFSET IN 制約の BEFORE タイプと共にのみ使用できます。

入力グループを使用する方法

同じクロックでキャプチャされる入力グループが同じタイミング要件を持つ場合、入力同士がグループになり、1 つのタイミング制約が作成されます。入力は、パッド グループを使用して入力信号名別、またはレジスタ グループを使用して同期エレメント別にグループ化できます。別々の信号を 1 つのタイム グループにすると、インプリメンテーション ツールのメモリやランタイムが削減され、タイミング レポートにバス ベースのスキューやクロックのセンタリング情報などが含まれるようになります。

UCF 構文例 (入力グループ)

```
[TIMEGRP "pad_groupname"] OFFSET = IN "offset_time" [units] [VALID <datavalid_time> [UNITS]]
{AFTER "clk_name" [TIMEGRP "reg_groupname"] | BEFORE "clk_name" [TIMEGRP "reg_groupname"]
[{RISING|FALLING}]};
```

PCF 構文例 (入力グループ)

```
[TIMEGRP "inputpad_grpname"] OFFSET = IN "offset_time" [units] [VALID <datavalid_time>
[UNITS]] {AFTER COMP "clk_iob_name" [TIMEGRP "reg_groupname"] | BEFORE
COMP "clk_iob_name" [TIMEGRP "reg_groupname"] [{RISING|FALLING}]};
```

- [**TIMEGRP** *"pad_groupname"*]

  オプションの入力パッドのタイム グループです。このタイム グループを使用すると、OFFSET IN 制約の適用範囲をタイム グループに含まれる入力パッド ネットから接続された同期エレメントのみに制限することができます。

- [**TIMEGRP** "*reg_groupname*"]

  オプションの同期エレメントのタイム グループです。このタイム グループを使用すると、OFFSET IN 制約の適用範囲を、指定クロック付き入力データをキャプチャする同期エレメントのみに制限することができます。

ネットにのみ適用する方法

OFFSET IN 制約は、回路図のデータ ネットや UCF ファイルの入力パッド ネットや PCF ファイルの入力コンポーネントにも使用できます。

回路図構文

```
OFFSET = IN "offset_time" [units] [VALID <datavalid_time> [UNITS]] {BEFORE|AFTER}
"clk_name" [TIMEGRP "reg_groupname"] [{RISING|FALLING}];
```

UCF 構文例 (ネットのみに適用)

```
NET "pad_net_name" OFFSET = IN "offset_time" [units] [VALID <datavalid_time> [UNITS]] {BEFORE|AFTER}
"clk_name" [TIMEGRP "reg_groupname"] [{RISING|FALLING}];
```

PCF 構文例 (ネットのみに適用)

```
COMP "pad_net_name" OFFSET = IN "offset_time" [units] [VALID <datavalid_time> [UNITS]] {BEFORE|AFTER}
COMP "clk_iob_name" [TIMEGRP "reg_groupname"] [{RISING|FALLING}];
```

- pad_net_name

  パッドに接続された入力データ ネットの名前です。

  その他の変数やキーワードの定義は、前述の「グローバルに指定する方法」を参照してください。

- PCF ファイルでは、ネット (NET) ではなく、I/O ブロック (COMP) を使用します。

  PCF ファイルで IOB COMP 名が、UCF ファイルで NET 名が指定されていない場合、OFFSET IN 制約はグローバルに指定されているものと見なされます。

回路図

- 特定のネットに設定します。
- 属性名

  OFFSET
- 属性値
  - IN|OUT
  - BEFORE|AFTER *clk_pad_netname*

XCF 構文

UCF と同じ構文を使用します。ただし、XCF 構文では OFFSET IN BEFORE しかサポートされません。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

Constraints Editor の構文

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

OFFSET_OUT

注記 : OFFSET 制約には定義済みグループは使用できません。

OFFSET OUT (オフセット出力) 制約には、次の特徴があります。

- FPGA デバイスからの出力インターフェイスのタイミング要件を指定します。
- FPGA の入力ピンのクロック エッジから FPGA の出力ピンでデータが有効になるまでの時間を指定します。
- クロック ネット名を使用して指定されます。

OFFSET OUT 制約の付いたクロック ネットは、外部クロック パッドです。この制約で指定されるのは、FPGA の入力ピンのクロック エッジから FPGA の出力ピンのデータまでなので、内部クロック ネットを使用して制約を指定することはできません。ただし、出力のタイミング要件が解析される際に、DCM、PLL、MMCM、IDELAY などのコンポーネントがあると、クロック パスの位相または遅延が自動的に考慮されます。また、この制約はクロック ネットワークを介して伝搬され、元の外部クロックから派生したすべてのクロックに自動的に適用されます。

OFFSET OUT 制約は、デフォルトではグローバルに適用されます。この場合、特定のクロック ネットからのクロックが使用され、外部データを送信する同期エレメントすべてにこの制約が適用されます。この制約が適用される同期エレメントは、出力データ パッドのサブセットのタイム グループや、同期エレメントをを送信するサブセットのタイム グループ、またはその両方のグループを指定することで制限できます。

詳細は、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

- グローバル
- ネット
- タイム グループ

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF の構文例も提供していますが、OFFSET OUT 制約の指定には Constraints Editor の使用をお勧めします。

グローバルに指定する場合

OFFSET OUT 制約は、デフォルトでグローバルに設定されます。この場合、出力データを送信し、指定したクロック信号でトリガされる同期エレメントすべてに適用されます。

注記 : BEFORE とキーワード RISING および FALLING を一緒に使用できますが、REFERENCE_PIN キーワードが使用される場合は AFTER キーワードを使用する必要があります。この場合、BEFORE キーワードは使用できません。

UCF 構文例

```
OFFSET = OUT "offset_time" [units] {BEFORE "clk_name"|AFTER "clk_name" [REFERENCE_PIN
"ref_pin"]} [{RISING|FALLING}];
```

PCF 構文例

```
OFFSET = OUT "offset_time" [units] {BEFORE COMP "clk_iob_name"|AFTER COMP
"clk_iob_name" [REFERENCE_PIN "ref_pin"]} [{RISING|FALLING}];
```

- offset_time [units]

  FPGA の入力ピンのクロック エッジからデータ出力ピンでデータが有効になるまでの時間を定義するオプションのパラメーターです。

  - *offset_time* 値が指定される場合 :
    - タイミング制約がこれらのパスに適用されます。
    - その制約に対するエラーがレポートされます。
  - *offset_time* 値が指定されない場合 :
    - タイミング制約は生成されません。
    - インターフェイスの出力タイミングとバス スキューがレポートされます。

  このレポートのみを表示するオプションは、出力バスのスキューの方がクロックから出力への時間より重視されるソース同期インターフェイスで使用すると便利です。

- **BEFORE|AFTER**

  クロック エッジからデータの開始時点までのタイミング関係を定義します。

  クロックとデータの要件を定義するには、AFTER を使用するのが最適な方法です。AFTER は、FPGA のピンのクロック エッジ後にデータが有効になるまでの時間を示します。

- clk_name には、入力クロック パッド ネットの階層名すべてを定義します。

- **REFERENCE_PIN**

  - クロックが再生成され、データと共に送信されるようなソース同期出力インターフェイスでよく使用されるオプションのキーワードです。
  - ref_pin 信号に接続された出力信号のバス スキューを解析できるようになります。

  REFERENCE_PIN を指定しない場合、バス スキューのレポートには出力遅延への最小クロック付きの信号が記述されます。

- **RISING|FALLING**

  - データを送信する同期エレメントの送信クロック エッジを定義するオプションのキーワードです。
  - DDR (デュアル データ レート) インターフェイスの立ち上がりエッジ レジスタと立ち下がりエッジ レジスタが自動的に別のグループに分けられて、解析されるようになります。

  RISING および FALLING キーワードの詳細は、『タイミング制約ユーザー ガイド』を参照してください。

出力グループを使用する方法

同じクロックで送信される出力グループが同じタイミング要件を持つ場合、出力同士がグループになり、1 つのタイミング制約が作成されます。出力は、パッド グループを使用して出力信号名別、またはレジスタ グループを使用して同期エレメント別にグループ化できます。別々の信号を 1 つのタイム グループにすると、インプリメンテーション ツールのメモリやランタイムが削減され、タイミング レポートにバス ベースのスキューやクロックのセンタリング情報などが含まれるようになります。

UCF 構文例

```
[TIMEGRP "pad_groupname"] OFFSET = OUT "offset_time" [units] {BEFORE|AFTER} "clk_name"[REFERENCE_PIN
"ref_pin"] [TIMEGRP "reg_groupname"] [{RISING|FALLING}];
```

PCF 構文例

```
[TIMEGRP "pad_groupname"] OFFSET = OUT "offset_time" [units] {BEFORE|AFTER} COMP
"clk_iob_name"[REFERENCE_PIN "ref_pin"][TIMEGRP "reg_groupname"][{RISING|FALLING}];
```

- グループ別の方法は、次に示す一般的な方法と同じです。その他の変数やキーワードの定義は、前述の「グローバルに指定する方法」を参照してください。
- [TIMEGRP *"pad_groupname"*] : オプションの出力パッドのタイム グループです。このタイム グループを使用すると、OFFSET OUT 制約の適用範囲をタイム グループに含まれる出力パッド ネットから接続された同期エレメントのみに制限することができます。
- [TIMEGRP "*reg_groupname*"] : オプションの同期エレメントのタイム グループです。このタイム グループを使用すると、OFFSET OUT 制約の適用範囲を、指定クロック付き出力データを送信する同期エレメントのみに制限することができます。

ネットにのみ適用する方法

OFFSET OUT 制約は、回路図のデータ ネットや UCF ファイルの出力パッド ネットや PCF ファイルの出力コンポーネントにも使用できます。

ネットに設定した場合の回路図構文

```
OFFSET = OUT "offset_time" [units] {BEFORE|AFTER} "clk_name" [TIMEGRP "reg_groupname"]
[REFERENCE_PIN "ref_pin"] [{RISING|FALLING}];
```

UCF 構文

```
NET "pad_net_name" OFFSET = OUT "offset_time" [units] {BEFORE|AFTER} "clk_name" [TIMEGRP
"reg_groupname"] [REFERENCE_PIN "ref_pin"] [{RISING|FALLING}];
```

PCF 構文

```
COMP "pad_net_name" OFFSET = OUT "offset_time" [units] {BEFORE|AFTER} "clk_name" [TIMEGRP
"reg_groupname"] [REFERENCE_PIN "ref_pin"] [{RISING|FALLING}];
```

- グループ別の方法は、次に示す一般的な方法と同じです。その他の変数やキーワードの定義は、前述の「グローバルに指定する方法」を参照してください。
- *"pad_net_name"* : パッドに接続された出力データ ネットの名前です。
- PCF ファイルでは、ネット (NET) ではなく、I/O ブロック (COMP) を使用します。
- PCF ファイルで IOB COMP 名が、UCF ファイルで NET 名が指定されていない場合、OFFSET OUT 制約はグローバルに指定されているものと見なされます。

ソース同期 DDR の UCF 例

この例に含まれるインターフェイスでは、クロックが FPGA 内で再生成され、データと共に送信され、デバイスがキャプチャされます。DDR インターフェイスでは、データは立ち上がりエッジと立ち下がりエッジの両方のクロック エッジで送信されます。データを送信する立ち上がりおよび立ち下がりクロック エッジのレジスタ別に OFFSET OUT 制約を定義してください。OFFSET OUT 制約に RISING および FALLING キーワードを使用すると、このタスクが簡単になります。また、再生成されたクロックに対するバス スキューを解析するために、**REFERENCE_PIN** キーワードが使用されます。

この例では、clock というクロック信号が FPGA に入力され、データ出力の同期エレメントをトリガします。また、TxClock という再生成されたクロックが作成されて、データと共に送信されます。これはソース同期インターフェイスなので、クロックから出力までの絶対時間は必要ないので、OFFSET OUT AFTER 値を指定しないで、レポートのみの制約を生成します。

UCF 構文

```
NET “clock” TNM_NET = CLK;
TIMESPEC TS_CLK = PERIOD CLK 5.0 ns HIGH 50%;
OFFSET = OUT AFTER clock REFERENCE_PIN “TxClock” RISING;
OFFSET = OUT AFTER clock REFERENCE_PIN “TxClock” FALLING;
```

システム同期 SDR の UCF 例

この例に含まれるインターフェイスでは、入力クロックが受信デバイスへデータを送信するために使用されます。SDR インターフェイスでは、データはクロック サイクルごとに送信されます。この場合、インターフェイスに制約を付けるため、OFFSET OUT が 1 つ必要になります。

この例では、clock というクロック信号が FPGA に入力され、データ出力の同期エレメントをトリガします。これはシステム同期インターフェイスなので、クロックから出力までの絶対時間は必要ありません。この場合、再生成されたクロックは存在しないので、**REFERENCE_PIN** キーワードを指定しないで、デフォルトのスキュー レポートを出力させます。

UCF 構文

```
NET “clock” TNM_NET = CLK;
TIMESPEC TS_CLK = PERIOD CLK 5.0 ns HIGH 50%;
OFFSET = OUT 5 ns AFTER “clock”;
```

回路図

- 特定のネットに設定します。
- 属性名

  OFFSET
- 属性値

  OUT *offset_time* **BEFORE|AFTER** *clk_pad_netname*

XCF 構文

XST Constraint File (XCF) 構文は、次のように指定します。

- UCF と同じ構文を使用します。
- OFFSET OUT AFTER しかサポートされません。

Constraints Editor の構文

ISE® Design Suite での Constraints Editor およびその設定に関する詳細は、ISE Design Suite ヘルプを参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

OPEN_DRAIN

OPEN_DRAIN (オープン ドレイン) 制約には、次のような特徴があります。

- CoolRunner™-II デバイスでのみサポートされます。
- トライステート以外の出力 (常にアクティブ) に適用されます。

出力をオープン ドレインとして設定

- CoolRunner-II デバイスの出力は、プライマリ マクロセル出力をそのピンのオープン ドレイン出力信号として駆動するよう設定できます。
- 出力構造をオープン ドレインとして設定すると、出力信号のステートが 1 の場合に、電圧が High に駆動されず、デバイス ピンでハイ インピーダンスになります。
- このオープン ドレインに関連したハイ インピーダンスは、論理シミュレーションでは表示されませんが、フィット後のタイミング シミュレーションで正確に表示されます。

オープン ドレインに代わる手段

- OPEN_DRAIN 制約を使用する代わりに、元の出力パッド信号をトライステート ディスエーブルとして使用する方法もあります。この方法を使用すると、出力データ値が定数 0 になります。
- CPLDFitter では、データ値が定数 0 のすべてのトライステート出力が自動的に最適化され、デバイスのオープン ドレイン機能が利用されます。

アーキテクチャ サポート

CoolRunner-II

適用可能エレメント

この制約は、次に適用されます。

- 出力パッド
- パッド ネット

適用ルール

ネットまたは信号に設定します。マクロ、エンティティ、モジュールには適用できません。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 出力パッド ネットに設定します。
- 属性名

  OPEN_DRAIN
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute OPEN_DRAIN: string;
```

VHDL 制約を次のように指定します。

```
attribute OPEN_DRAIN of signal_name : signal is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* OPEN_DRAIN = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

```
NET "mysignal" OPEN_DRAIN;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" OPEN_DRAIN=true;
END;
```

OUT_TERM

OUT_TERM 制約には、次の特徴があります。

- 基本的なマップ制約です。
- 出力終端抵抗のコンフィギュレーションを設定します。
- 次で使用できます。
  - 出力パッド NET
  - 出力パッド INST
  - デザイン全体

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

この制約は、次のエレメントまたはデザイン エレメントのカテゴリを 1 つまたは複数使用して設定できます。

- IOB 入力コンポーネント (例 : IBUF)
- 出力パッド ネット

すべてのデバイスですべてのエレメントがサポートされるわけではありません。デバイス別にどのデザイン エレメントが使用可能かを確認するには、そのデバイスのライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

適用ルール

ネットまたは信号がパッドに接続されている場合を除いて、ネットや信号に設定できません。この場合、OUT_TERM はパッド インスタンスに設定されているものと見なされます。

制約値

- NONE
- TUNED
- UNTUNED_25
- UNTUNED_50
- UNTUNED_75

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図構文

- パッド ネットに設定します。
- 属性名

  OUT_TERM
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
Attribute OUT_TERM: string;
```

VHDL 制約を次のように指定します。

```
attribute OUT_TERM of signal_name: signal is
"{NONE|TUNED|UNTUNED_25|UNTUNED_50|UNTUNED_75}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* OUT_TERM = "{NONE|TUNED|UNTUNED_25|UNTUNED_50|UNTUNED_75 }" *)
```

UCF および NCF 構文

- ```
  NET "pad_net_name" OUT_TERM =
  "{NONE|TUNED|UNTUNED_25|UNTUNED_50|UNTUNED_75}" ;
  ```

  I/O で PULLUP を使用するよう指定しています。
- ```
  DEFAULT OUT_TERM = TUNED;
  ```

  OUT_TERM をグローバルに設定しています。

XCF 構文

```
BEGIN MODEL "entity_name"

 NET "signal_name" out_term=tuned;

END;
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PERIOD

PERIOD は、基本的なタイミング制約および合成制約です。

PERIOD 制約を指定すると、デスティネーション エレメントのグループで定義されているクロック ドメイン内で、すべての同期エレメント間のタイミングが確認されます。クロックが別のクロックの関数として定義されている場合、グループには複数のクロック ドメインを通過するパスが含まれます。

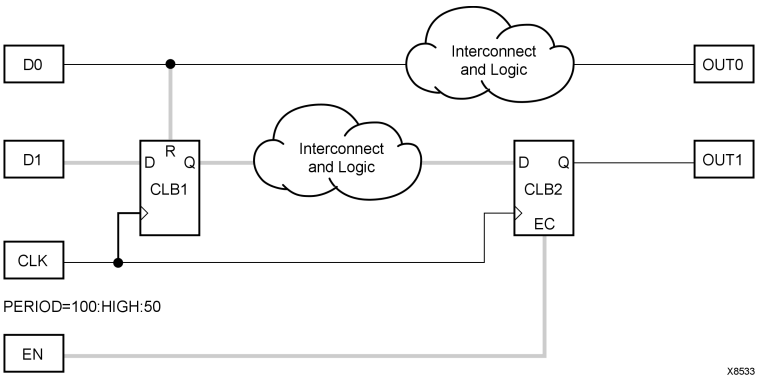
派生周期制約は、それらの参照制約と同じ単位で定義されます。

PERIOD 制約は、クロック ネットに設定されます。タイミング解析ツールは、レジスタのクロック ピンにおけるクロック ネットの反転や固定位相を自動的に考慮し、同期エレメントのすべてのタイプを解析の対象とするため、クロック周期の定義は FROM-TO とは異なります。また、ホールド タイム違反があるかどうかもチェックされます。

クロック ネットに設定された PERIOD 制約は、クロック ネットに対応するセットアップまたはホールドのタイミング制約が設定されたピンが終端となるすべてのパスで、遅延を確認します。イネーブルがクロックと同期化している場合、このパスには CLB1.Q から CLB2.D までのパス、EN から CLB2.EC までのパスが含まれることがあります。

詳細は、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

PERIOD 制約のパス



設定要件によって必然的に決定される pad-to-register パスは、タイミング ツールでチェックされません。たとえば、D1 から CLB1 のピン D までのパスは PERIOD 制約には含まれません。また、clock-to-out パスも同様にチェックされません。また、clock-to-out パスも同様にチェックされません。

DLL、DCM、PLL、MMCM で、PERIOD 制約と共に TNM または TNM_NET を使用する場合は、特別なルールが適用されます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

フリップフロップのクロック ピンを駆動するためのネットに適用します。

適用ルール

設定された信号に適用されます。

TIMESPEC PERIOD の使用

この方法は、単純なクロック周期だけでなく、より複雑な派生関係も定義できるので一番お勧めします。

TIMESPEC PERIOD の UCF 構文

関連するクロック ネットに TNM 制約が設定されるとともに、**TIMESPEC** キーワードを使用して次のように設定されます。

```
TIMESPEC "TSidentifier"=PERIOD "TNM_reference" period {HIGH | LOW}
[high_or_low_time] INPUT_JITTER value;
```

- *identifier*

  一意な名前を持つ参照識別名です。

- *TNM_reference*

  PERIOD 制約を適用するエレメントのグループを識別します。通常では、TNM_NET という名前がクロック ネットに付けられていますが、それ以外にも同期エレメントのみを含む TNM グループまたはユーザー グループ (TIMEGRP) も使用できます。

次の規則が適用されます。

- *period* は、必要なクロック周期です。
- デフォルトで *period* の単位はナノ秒ですが、ps、ns、micro、ms なども指定できます。また。MHz、GHz、または kHZ などの単位を使って周波数で指定することもできます。
- 単位の前に空白を入れても入れなくてもかまいません。
- 単位で大文字/小文字の区別をする必要はありません。
- HIGH|LOW キーワードは、周期の最初のパルスの高低を表し、オプションの *high_or_low_time* は最初のパルスの極性を表します。これらのキーワードは、最初のクロック エッジを定義し、OFFSET 制約で使用されます。ロジック レベルを指定しない場合は、デフォルトの High になります。
- 実際の時間を指定する場合は、周期より小さい値にする必要があります。
- *high_or_low_time* を指定しない場合、デフォルトのデューティ サイクルは 50% です。
- デフォルトで *high_or_low_time* の単位は ns ですが、% も使用できます。 単位には ps、ns、micro、ms なども選択できます。
- INPUT_JITTER は、入力クロックの peak-to-peak ジッタです。デフォルトで、単位は ps に設定されています。

TIMESPEC PERIOD の UCF 構文の例

クロック ネット sys_clk には、制約 tnm=master_clk が設定され、次のように TIMESPEC に設定されます。

```
TIMESPEC TS_master = PERIOD "master_clk" 50 HIGH 30 INPUT_JITTER
0.050;
```

PERIOD 制約はネット master_clk に適用され、最初の High 状態の時間 30ns でクロック周期 50ns、INPUT_JITTER は 50ps で定義します。

```
TIMESPEC TS_clkinA = PERIOD "clkinA" 21 ns LOW 50% INPUT_JITTER
500 ps;

TIMESPEC TS_clkinB = PERIOD "clkinB" 21 ns HIGH 50% INPUT_JITTER
500 ps;
```

NET PERIOD の使用

注意 : これは 2 つ目の方法で、推奨はされません。

この方法では、レジスタのクロック ピンを駆動するパスで、制約をネットに直接設定します。

NET PERIOD の 回路図構文

```
PERIOD = period  {HIGH|LOW} [ high_or_low_time] INPUT_JITTER value;
```

NET PERIOD の UCF 構文

```
NET "net_name" PERIOD = period  {HIGH|LOW} [ high_or_low_time] INPUT_JITTER value;
```

- period は、必要なクロック周期です。デフォルトの単位はナノ秒ですが、ps、ns、micro、ms なども指定できます。また。MHz、GHz、または kHZ などの単位を使って周波数で指定することもできます。
- 単位の前に空白を入れても入れなくてもかまいません。
- 単位で大文字/小文字の区別をする必要はありません。
- HIGH|LOW キーワードは、周期の最初のパルスの高低を表し、オプションの *high_or_low_time* は最初のパルスのデューティ サイクルを表します。ロジック レベルを指定しない場合は、デフォルトの High になります。
- 実際の時間を指定する場合は、周期より小さい値にする必要があります。
- high_or_low_time を指定しない場合、デフォルトのデューティ サイクルは 50% になります。
- デフォルトで *high_or_low_time* の単位は ns ですが、% も使用できます。 単位には ps、ns、micro、ms なども選択できます。

PERIOD 制約は TNM の場合とまったく同じ方法で順方向にトレースされ、到達したすべての同期エレメントに設定されます。ゲートを介したクロックをデザイン内で使用する場合など、複雑な形式のトレースが必要な場合は、特定ネットに PERIOD を設定するか、次の「推奨する方法」を使用する必要があります。

派生クロックの指定

「推奨する方法」では、識別名を使用して、別のクロック周期指定が参照できるようにしています。派生したPERIOD 制約にはマスター PERIOD 制約と同じ HIGH/LOW キーワードを使用してください。マスター PERIOD 制約に HIGH キーワードが付いているか、マスタ PERIOD 制約がデフォルトの場合、同じ HIGH キーワードを派生の PERIOD 制約に使用してください。派生クロックの場合に関係を定義するには、次の構文を使用します。

派生クロックを指定する UCF 構文

```
TIMESPEC "TSidentifier"=PERIOD "timegroup_name" "TSidentifier" [* or /]
factor PHASE [+ |-] phase_value [units];
```

説明 :

- identifier は、一意な名前を持つ参照識別名です。
- *factor* は、浮動小数点数です。

  注記 : 定義される値が参照される値と同じで位相だけが異なる場合、(*) または (/) の引数を省略できます。これは (* 1) を使用するのと同じことになります。
- *phase_value* は、浮動小数点数です。
- *units* は、ps、ms、micro、ns (デフォルト) などの単位です。

次の規則が適用されます。

- 実際の時間を指定する場合は、周期より小さい値にする必要があります。
- *high_or_low_time* を指定しない場合、デフォルトのデューティ サイクルは 50% です。
- デフォルトで *high_or_low_time* の単位は ns ですが、% も使用できます。 単位には ps、ns、micro、ms なども選択できます。

派生クロックを使用したプライマリ クロックの例

プライマリ クロックの周期

```
TIMESPEC "TS01" = PERIOD "clk0" 10.0 ns;
```

180° 順方向へ位相シフトしたクロックの周期

```
TIMESPEC "TS02" = PERIOD "clk180" TS01 PHASE + 5.0 ns;
```

90° 逆方向へ位相シフトしたクロックの周期

```
TIMESPEC "TS03" = PERIOD "clk90" TS01 PHASE - 2.5 ns;
```

180° 順方向 (TS01 に対して 90°) に位相シフトしたり、倍増したクロックの周期

```
TIMESPEC "TS04" = PERIOD "clk180" TS01 / 2 PHASE + 2.5 nS;
```

回路図

- ネットに設定します。構文は、次のようになります。
- 属性名 :**PERIOD**
- 属性値 :*period* [*units*] [{**HIGH**|**LOW**} [*high_or_low_time* [*hi_lo_units*]]

VHDL 構文

XST の場合、PERIOD は特定のクロック信号のみに適用されます。

注記 : ソース コード (VHDL または Verilog) の PERIOD 制約は、ネットリストへ伝搬されません。

VHDL 制約を次のように宣言します。

```
attribute period:  string;
```

VHDL 制約を次のように指定します。

```
attribute period of signal_name : signal is "period [units]";
```

- period は、必要なクロック周期です。
- units は、オプションでクロック周期の単位を指定します。デフォルトの単位はナノ秒 (ns) ですが、ps、ns、micro なども指定できます。

Verilog 構文

XST の場合、PERIOD は特定のクロック信号のみに適用されます。

注記 : ソース コード (VHDL または Verilog) の PERIOD 制約は、ネットリストへ伝搬されません。

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* PERIOD = "period [units]" *)
```

- period

  必要なクロック周期です。

- units

  オプションでクロック周期の単位を指定します。デフォルトの単位はナノ秒 (ns) ですが、ps、ns、micro なども指定できます。

UCF および NCF 構文

次に、UCF と NCF 構文の例を示します。

- TIMESPEC PERIOD の使用 (推奨)
- NET PERIOD の使用 (推奨されない)

TIMESPEC PERIOD の使用 (推奨)

これは推奨される方法です。

```
TIMESPEC "TSidentifier"=PERIOD "TNM_reference period" [units]
[{HIGH | LOW} [high_or_low_time [hi_lo_units]]]
INPUT_JITTER value [units];
```

- identifier

  一意な名前を持つ参照識別名です。

- TNM_reference

  TNM 制約や TNM_NET 制約を使用してクロック ネットまたはクロック パスのネットに適用するときの識別名です。

DLL、DCM、PLL、MMCM コンポーネントの CLKIN 入力に TNM_NET 制約がトレースされる場合、新たに PERIOD を DLL/DCM/PLL/MMCM 出力に指定できます。この場合、その指定で使用する TNM_NET グループも新たに生成されます。

新しい TNM_NET グループには、対応する DLL/DCM/PLL/MMCM の出力ネットと同じ名前 (*outputnetname*) が付けられます。新しい PERIOD 指定は、TS_*outputnetname*=PERIOD *outputnetname value units* になります。

新しい TNM_NET グループは、DLL/DCM/PLL/MMCM の出力ネットからトレースされ、クロック信号で制御されるすべての同期エレメントを指定します。新しいグループと指定は、タイミング解析レポートに出力されます。

次の規則が適用されます。

- period

  必要なクロック周期です。

- units

  オプションでクロック周期の単位を指定します。デフォルトはナノ秒 (ns) ですが、ps、ms、micro、% なども指定できます。

- HIGH または LOW

  最初のパルスを High にするか、Low にするかを指定します。

- high_or_low_time

  High または Low になっている時間を指定します (オプション)。High か Low かは、この前のキーワードによって指定します。実際の時間を指定する場合は、周期より小さい値にする必要があります。*high_or_low_time* を指定しない場合、デフォルトのデューティ サイクルは 50% になります。

- hi_lo_units

  デューティ サイクルの単位を指定します (オプション)。デフォルトはナノ秒 (ns) です。*high_or_low_time* が実際の計測値である場合、High または Low の時間を示す値の後に ps、micro、ms、% を付けて単位を指定できます。

次の文は、クロック周期 40ns をネット $SIG_24 に設定します。最初のパルスは High で、その継続時間は 25ns です。

```
NET "CLOCK" PERIOD=40 HIGH 25;
```

NET PERIOD の使用 (推奨されない)

注意 : これは 2 つ目の方法で、推奨はされません。

```
NET "net_name" PERIOD=period [units] [{HIGH|LOW}
[high_or_low_time [hi_lo_units]]];
```

- period

  必要なクロック周期です。

- units

  オプションでクロック周期の単位を指定します。デフォルトの単位はナノ秒 (ns) ですが、ps、ns、micro なども指定できます。

- HIGH または LOW

  最初のパルスを High にするか、Low にするかを指定します。

- hi_lo_units

  ns (デフォルト)、ps、micro にできます。

次の規則が適用されます。

- high_or_low_time は、High または Low になっている時間を指定します (オプション)。High か Low かは、この前のキーワードによって指定します。
- 実際の時間を指定する場合は、周期より小さい値にする必要があります。
- *high_or_low_time* を指定しない場合、デフォルトのデューティ サイクルは 50% になります。
- hi_lo_units は、デューティ サイクルの単位を指定します (オプション)。
- デフォルトはナノ秒 (ns) です。*high_or_low_time* が実際の計測値である場合、High または Low の時間を示す値の後に ps、micro、ms、% を付けて単位を指定できます。

Constraints Editor の構文

Constraints Editor を起動するには

1. ISE® Design Suite の [Processes] ペインで [Create Timing Constraints] をダブルクリックします。
2. [Constraint Type] ウィンドウの [Timing Constraints] の下の [Clock Domains] をダブルクリックします。

XCF 構文

UCF と同じ構文を使用します。

単純な方法および推奨する方法の両方がサポートされていますが、HIGH/LOW の値はタイミング概算および最適化では使用されず、WRITE_TIMING_CONSTRAINTS=yes の場合に最終のネットリストに含められるのみです。

PCF 構文

"TS*identifier***"=PERIOD** *perioditem periodvalue* **INPUT_JITTER** *value;*

- *perioditem* は、次のいずれかです。
  - NET *name*
  - TIMEGRP *name*
- *periodvalue* は、次のいずれかです。
  - TS*identifier* PHASE [+ | -] *time*
  - TS*identifier* PHASE *time*
  - TS*identifier* PHASE [+ | -] *time* [LOW | HIGH] *time*
  - TS*identifier* PHASE *time* [LOW | HIGH] *time*
  - TS*identifier* PHASE [+ | -] *time* [LOW | HIGH] *percent*
  - TS*identifier* PHASE *time* [LOW | HIGH] *percent*

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

FPGA Editor の構文

制約を設定するには、FPGA Editor のメイン ウィンドウで [Edit] → [Properties of Selected Items] をクリックします。PERIOD 制約を設定するには、ネットを選択した状態で [Edit] → [Properties of Selected Items] をクリックします。制約は、[Physical Constraints] タブから設定できます。

CLKDLL、DCM、PLL、MMCM での PERIOD 仕様

詳細は、『タイミング クロージャ ユーザー ガイド』(UG612) を参照してください。

PIN

PIN (UCF 制約) には、次の特徴があります。

- LOC 制約と一緒に使用してネットの位置 (ロケーション) を定義します。
- デザイン フローを作成する際に使用します。
- PCF ファイルで COMP/LOCATE 制約に変換されます。

  次の「PCF 構文」セクションを参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

ネットに適用

適用ルール

なし

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

PIN "*module*.*pin* " LOC=*location* ;

PIN mod.pin TIG;

PCF 構文

COMP "*name*" LOCATE = SITE "*location* ";

この制約は、モジュールでピンに対して作成された疑似コンポーネントがサイトの位置に配置されるように指定します。疑似ロジックは、ネットがあるモジュールのピンから別のモジュールのピンに接続される場合にのみ、作成されます。

POST_CRC

POST_CRC 制約には、次のような特徴があります。

- コンフィギュレーション ロジック CRC エラー検出機能をオンまたはオフにする制約です。コンフィギュレーション メモリの変更は、この機能により知ることができます。
- Spartan®-3A の場合は、この制約によりコンフィギュレーション CRC エラーの信号送信用に多用途 INIT ピンが予約されます。

また、PlanAhead™、PAR、および BitGen によって使用されるバンク規則で、INIT ピンを駆動する IOB が使用されないように指定することもできます。INIT ピンは、コンフィギュレーション中は通常通りに作動し、POST_CRC 解析がオンの場合は、コンフィギュレーション後、CRC ステータス ピンとして動作します。リアルタイムに計算された CRC が事前に計算された CRC と異なると、コンフィギュレーション メモリの変更が検出され、INIT ピンは Low に駆動されます。

詳細は、デバイスのデータシートを参照してください。

アーキテクチャ サポート

- Virtex®-5
- Virtex-6
- Spartan-3A
- Spartan-6

適用可能エレメント

デザイン全体に適用

適用ルール

デザイン全体に適用

制約値

値	説明
ENABLE	Post CRC の検出機能をオンにします。
DISABLE	Post CRC の検出機能をオフにします (デフォルト)。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG POST_CRC = {ENABLE|DISABLE|ONESHOT};
```

PCF 構文

```
CONFIG POST_CRC = {ENABLE|DISABLE|ONESHOT};
```

POST_CRC_ACTION

POST_CRC_ACTION 制約には、次のような特徴があります。

- コンフィギュレーション ロジック CRC エラーの検出モードです。
- コンフィギュレーション ビットストリームのあらかじめ算出された CRC が、コンフィギュレーション メモリ セルの周期的なリードバックに基づいて内部ロジックで算出された CRC と比較されます。
- CRC の不一致が検出されたときに検出を続けるか、処理を停止するかを指定します。
- POST_CRC が ENABLE に設定されているときにのみ使用できます。

アーキテクチャ サポート

- Spartan®-3A
- Spartan-6
- Virtex®-6

適用可能エレメント

- デバイス全体に適用されます。
- 特定のデザイン エレメントには適用できません。

適用ルール

デザイン/デバイス全体に適用されます。

制約値

値	説明
HALT	CRC の不一致が検出されると、ビットストリームのリードバック、比較 CRC の計算、および事前に計算された CRC との比較が停止されます (のデフォルト)。
CONTINUE	CRC の不一致が検出されても、ビットストリームのリードバック、比較 CRC の計算、事前に計算された CRC との比較が続行されます (のデフォルト)。
CORRECT_AND_CONTINUE	CRC の不一致が検出されると、それが修正され、ビットストリームのリードバック、CRC の計算、事前に計算された CRC との比較が続行されます。
CORRECT_AND_HALT	CRC の不一致が検出されると、それが修正され、ビットストリームのリードバック、CRC の計算、事前に計算された CRC との比較が停止されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG POST_CRC_ACTION = [HALT|CONTINUE];
```

PCF 構文

```
CONFIG POST_CRC_ACTION = [HALT|CONTINUE];
```

POST_CRC_FREQ

POST_CRC_FREQ (ポスト CRC 周波数) 制約には、次の特徴があります。

- コンフィギュレーション ロジック CRC エラーの検出モードです。
- コンフィギュレーション ビットストリームのあらかじめ算出された CRC が、コンフィギュレーション メモリ セルの周期的なリードバックに基づいて内部ロジックで算出された CRC と比較されます。
- コンフィギュレーション CRC チェックがこの制約でサポートされるすべてのデバイスで実行される周波数を制御します。
- POST_CRC が ENABLE に設定されているときにのみ使用できます。

アーキテクチャ サポート

Spartan®-3A、Spartan-6、Virtex®-6 デバイスのみをサポート

適用可能エレメント

デバイス全体に適用され、特定のデザイン エレメントには指定されません。

適用ルール

デザイン全体に適用

制約値

デバイス	周波数範囲 (MHz)	ステップ (MHz)	デフォルト値 (MHz)
Spartan-3A	1 ～ 100	1、3、6、7、8、10、12、13、17、22、25、27、33、44、50、100	1
Spartan-6	1 ～ 100	1、2、4、6、10、12、16、22、26、33、40、50、66	1
Virtex-6	1 ～ 50	1、2、3、6、13、25、50	1

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG POST_CRC_FREQ =
[1|3|6|7|8|10|12|13|17|22|25|27|33|44|50|100];
```

PCF 構文

```
CONFIG POST_CRC_FREQ =
[1|3|6|7|8|10|12|13|17|22|25|27|33|44|50|100];
```

POST_CRC_INIT_FLAG

POST_CRC_INIT_FLAG (ポスト CRC INIT フラグ) 制約には、次の特徴があります。

- ロジック CRC エラーの検出モードです。
- Virtex®-5 デバイス デザインにのみ使用可能であった POST_CRC_SIGNAL に代わるものです。
- POST_CRC が ENABLE に設定されているときにのみ使用できます。

ロジック CRC エラーの検出モード

このモードでは、コンフィギュレーション ビットストリームのあらかじめ算出された CRC が、コンフィギュレーション メモリ セルの周期的なリードバックに基づいて内部ロジックで算出された CRC と比較されます。

POST_CRC_INIT_FLAG では、INIT_B ピンを SEU (Single Event Upset) エラー信号の出力としてイネーブルにするかどうかが指定されます。

- Spartan®-6 デバイスにはエラー状況も含む POST_CONFIG_INTERNAL サイトがあります。これは、POST_CRC_INIT_FLAG の設定に関係なく使用できます。
- Virtex-5 および Virtex-6 の場合、エラー状況は FRAME_ECC_VIRTEX5 および FRAME_ECC_VIRTEX6 サイトから常に知ることができます。

制約値

値	説明
DISABLE	• Virtex-5、Virtex-6 デバイス INIT_B ピンの使用をオフにします。FRAME_ECC サイトが CRC エラー信号の唯一のソースとなります。 • Spartan-6 デバイス INIT_B ピンのステータス出力としての使用をディスエーブルにします。POST_CONFIG_INTERNAL サイトが CRC エラー信号の唯一のソースとなります。また、INIT_B ピンは保存され、ユーザー I/O として使用できないようになります。
ENABLE	• INIT_B ピンをCRC エラー信号のソースとして使用します • デフォルトは ENABLE です。

アーキテクチャ サポート

この制約は、次のデバイスでサポートされます。

- Virtex-5
- Virtex-6
- Spartan-6

適用可能エレメント

デバイス全体に適用され、特定のデザイン エレメントには指定されません。

適用ルール

デザイン全体またはデバイス全体に適用

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG POST_CRC_INIT_FLAG = [DISABLE|ENABLE];
```

PCF 構文

```
CONFIG POST_CRC_INIT_FLAG = [DISABLE|ENABLE];
```

POST_CRC_SIGNAL

POST_CRC_SIGNAL (ポスト CRC 信号) 制約には、次のような特徴があります。

- Virtex®-5 デバイスでのみサポートされます。
- コンフィギュレーション ロジック CRC エラーの検出モードです。
- コンフィギュレーション ビットストリームのあらかじめ算出された CRC が、コンフィギュレーション メモリ セルの周期的なリードバックに基づいて内部ロジックで算出された CRC と比較されます。
- POST_CRC_SIGNAL では、INIT_B ピンを SEU (Single Event Upset) エラー信号の出力としてイネーブルにするかどうかが指定されます。この場合でも、エラー ステータスは FRAME_ECC_VIRTEX5 サイトから取得できます。
- この制約は、POST_CRC が ENABLE に設定されているときにのみ使用できます。

アーキテクチャ サポート

Virtex-5

適用可能エレメント

- デバイス全体に適用されます。
- 特定のデザイン エレメントには適用できません。

適用ルール

デザイン全体に適用されます。

制約値

値	説明
FRAME_ECC_ONLY	INIT_B ピンの使用をディスエーブルにします。FRAME_ECC サイトが CRC エラー信号の唯一のソースとなります。
INIT_AND_FRAME_ECC	INIT_B ピンを CRC エラー信号のソースとしてイネーブルのままにします (デフォルト)。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG POST_CRC_SIGNAL = [FRAME_ECC_ONLY|INIT_AND_FRAME_ECC];
```

PCF 構文

```
CONFIG POST_CRC_SIGNAL = [FRAME_ECC_ONLY|INIT_AND_FRAME_ECC];
```

POST_CRC_SOURCE

POST_CRC_SOURCE (ポスト CRC ソース) 制約は、コンフィギュレーション ロジック CRC エラー検出機能がコンフィギュレーション メモリに変更があった可能性を通知するのに使用される場合に、その CRC 値のソースを指定するために使用されます。

アーキテクチャ サポート

この制約は、次のデバイスでサポートされます。

- Virtex®-5
- Virtex-6
- Spartan®-6

適用可能エレメント

デザイン全体に適用されます。

適用ルール

デザイン全体に適用されます。

制約値

- PRE_COMPUTED

  事前に計測されたビットストリーム CRC が使用されます。
- FIRST_READBACK

  最初に計測された CRC が使用されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

```
CONFIG POST_CRC_SOURCE = {PRE_COMPUTED|FIRST_READBACK};
```

PCF 構文

UCF および NCF 構文と同じ

PRIORITY

PRIORITY (優先度) 制約には、次の特徴があります。

- 高度なタイミング制約です。
- 同じパスを持つ 2 つのタイミング制約間で競合が発生する場合に、優先順位を付けるため使用します。

小さい数字から優先順位が高くなります。

この値は、次のようになります。

- 配置配線が実行されるパスの順番には影響しません
- 優先順位が同じ 2 つの制約を同じパスに設定した場合、パスに適用される制約の選択に影響します。

詳細は、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

TIMESPEC 制約に適用されます。

適用ルール

なし

制約値

タイミング制約はすべて UCF に書き込まれ次第優先度 0 になります。タイミング制約がその他すべての制約よりも優先される場合は、PRIORITY キーワードの前に負の数値が必要となります。

- normal_timespec_syntax

  有効なタイミング仕様です。

- integer
  - 優先度を示します。
  - 数字には、正の数、負の数、またはゼロを使用できます。
  - この数字は、ほかの PRIORITY 値と比較する場合にのみ使用されます。
  - 小さい数字から優先順位が高くなります。
  - PRIORITY キーワードの付いた制約は、常にそれ以外の制約よりも優先されます。

```
TIMESPEC "TS01"=FROM "GROUPA" TO "GROUPB" 40 PRIORITY 4;
```

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

normal_timespec_syntax **PRIORITY** *integer*;

PCF 構文

UCF および NCF 構文と同じ

PROHIBIT

PROHIBIT (禁止) 制約には、次の特徴があります。

- 基本的な配置制約です。
- 次でサイトが使用されないよう指定します。
  - PAR
  - FPGA Editor
  - CPLD フィッタ

FPGA デバイスのロケーション指定の種類

エレメントの物理的な位置を定義するには、次のように指定します。

エレメント のタイプ	位置の指定	説明
IOB	P12	IOB の位置 (チップ キャリア)
	A12	IOB の位置 (ピン グリッド)
	T、B、L、R	Spartan®-3、Spartan-3A、Spartan-3E、Virtex®-4、Virtex-5 デバイスの場合、IOB に適用され、エッジの位置 (下、左、上、右) を示します。
	LB、RB、LT、RT、BR、TR、BL、TL	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスの場合、IOB に適用され、ハーフ エッジの位置 (左下、右下など) を示します。
	Bank 0、Bank 1、Bank 2、Bank 3、Bank 4、Bank 5、Bank 6、Bank 7	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスの場合、IOB に適用され、ハーフ エッジ (バンク) を示します。
スライス	SLICE_X22Y3	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスのサイトの位置
ブロック RAM	RAMB16_X2Y56	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスのブロック RAM の位置
乗算器	MULT18X18_X55Y82	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスの乗算器の位置
グローバル クロック	BUFGMUX0P	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスのグローバル クロック バッファーの位置
デジタル クロック マネージャ (DCM)	DCM_X[A]Y[B]	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスのデジタル クロック マネージャ
位相ロック ループ (PLL)	PLL_X[A]Y[B]	Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 の位相ロック ループ

エレメント のタイプ	位置の指定	説明
混合モード クロック マネージャ (MMCM)	MMCM_X[A]Y[B]	Virtex-6 デバイスの混合モード クロック マネージャ

次の例のように、ワイルドカード文字 (*) を使用すると、単一の位置を範囲に置き換えることができます。

SLICE_X*Y5

Y 座標が 5 にある FPGA デバイスのすべてのスライス

次の文字は、サポートされていません。

- 範囲を指定する拡張子

  例

  LOC=SLICE_X3Y5:SLICE_X5Y7.G

- Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 のグローバル バッファー、グローバル パッド、DLL の位置を表すワイルドカード文字

CPLD デバイスのロケーション指定の種類

CPLD デバイスでは、位置の種類である *pin_name* しかサポートされません。

pin_name は次のいずれかの形式になります。

- ピン名 (数値) は P*nn*
- 行/列のピン名は *rc*

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

サイトに適用されます。

適用ルール

ネット、信号、エンティティ、モジュール、マクロには設定できません。

制約値

- location

  パーツ タイプに対して有効な位置

- site_group
  - **SITE** "*site_name*"

    または

  - **SITEGRP** "*site_group_name*"
- site_name
  - コンポーネント サイト

    または

  - CLB または IOB の位置

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

UCF ファイルでは、PROHIBIT の前に CONFIG キーワードを付ける必要があります。

- 詳細は、「FPGA デバイスおよび CPLD デバイスの位置指定の制約の箇所」を参照してください。
- 位置指定の例は「LOC」を参照してください。

位置	例
単一の位置	**CONFIG PROHIBIT=***location*;
複数の位置	**CONFIG PROHIBIT=***location1*, *location2*, ... ,*locationn*;
位置の範囲	**CONFIG PROHIBIT=***location1*:*location2*;

CPLD デバイスでは、「位置の範囲」はサポートされません。

UCF 構文例

- **CONFIG PROHIBIT=P45;**

  サイト P45 の使用を禁止します。

- **CONFIG PROHIBIT=SLICE_X6Y8;**

  SLICE_X6Y8 のサイトにあるスライスの使用を禁止します。

PCF 構文

次は、さまざまな位置に対する PCF 構文の例です。

位置	例
単一位置または複数の単一位置	**COMP** "*comp_name*" **PROHIBIT = [SOFT]** "*site_group*"..."*site_group*"; **COMPGRP** "*group_name*" **PROHIBIT = [SOFT]** "*site_group*"..."*site_group*"; **MACRO** "*name*" **PROHIBIT = [SOFT]** "*site_group*"..."*site_group*";
位置の範囲	**COMP** "*comp_name*" **PROHIBIT = [SOFT]** "*site_group*"... "*site_group*"; **COMPGRP** "*group_name*" **PROHIBIT = [SOFT]** "*site_group*"... "*site_group*"; **MACRO** "*name*" **PROHIBIT = [SOFT]** "*site_group*"..."*site_group*";

PlanAhead™ の構文

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PACE の設定

Pinout and Area Constraints Editor (PACE) では、CPLD デバイスのみの PROHIBIT 制約がサポートされます。

詳細は、PACE ヘルプの [Prohibit Mode] コマンドに関するセクションを参照してください。

FPGA Editor の構文

FPGA Editor では、PROHIBIT 制約がサポートされます。

FPGA Editor を使用すると、PROHIBIT 制約は PCF ファイルに書き込まれます。

詳細は、FPGA Editor ヘルプの「禁止制約」の項目を参照してください。

PULLDOWN

PULLDOWN (プルダウン) 制約には、次の特徴があります。

- 基本的なマップ制約です。
- ロジック レベルが Low になるので、トライステートで駆動されるネットが駆動していない状態のときでもフロートしなくなります。

KEEPER、PULLUP および PULLDOWN 制約は、次の条件で使用できます。

- パッド ネットにのみ使用できます。
- INST には使用できません。

アーキテクチャ サポート

- すべての FPGA デバイス
- CoolRunner™-II

適用可能エレメント

- 入力
- トライステート出力
- 双方向のパッド ネット

適用ルール

この制約はネット制約です。デザイン エレメントへは接続できません。

制約値

- YES
- NO
- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- パッド ネットに設定します。
- 属性名

  PULLDOWN
- 属性値
  - TRUE
  - FALSE

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute PULLDOWN: string;
```

VHDL 制約を次のように指定します。

```
attribute PULLDOWN of signal_name: signal is "{YES|NO|TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* PULLDOWN = "{YES|NO|TRUE|FALSE}" *)
```

UCF および NCF 構文

- **`NET "pad_net_name" PULLDOWN;`**

  I/O で PULLDOWN を使用するように指定します。

- **`DEFAULT PULLDOWN = TRUE;`**

  PULLDOWN をグローバルに指定しています。

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" pulldown=true;
END;
```

PlanAhead™ の構文

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PULLUP

PULLUP (プルアップ) 制約には、次の特徴があります。

- 基本的なマップ制約です。
- ロジック レベルが High になるので、トライステートで駆動されるネットが駆動していない状態のときでもフロートしなくなります。

KEEPER、PULLUP および PULLDOWN 制約は、次の条件で使用できます。

- パッド ネットにのみ使用できます。
- INST には使用できません。

CoolRunner™-II デザインの場合は、KEEPER と PULLUP を一緒に使用できません。

NGDBuild では、次が無視されます。

- DEFAULT KEEPER = FALSE
- DEFAULT PULLUP = FALSE
- DEFAULT PULLDOWN = FALSE

アーキテクチャ サポート

- すべての FPGA デバイス
- CoolRunner XPLA3
- CoolRunner-II

適用可能エレメント

- 入力
- トライステート出力
- 双方向のパッド ネット

適用ルール

この制約はネット制約です。デザイン エレメントへは接続できません。

制約値

- YES
- NO
- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- パッド ネットに設定します。
- 属性名

  PULLUP
- 属性値
  - TRUE
  - FALSE

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute PULLUP: string;
```

VHDL 制約を次のように指定します。

```
attribute PULLUP of signal_name: signal is "{YES|NO|TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* PULLUP = "{YES|NO|TRUE|FALSE}" *)
```

UCF および NCF 構文

- **NET "pad_net_name" PULLUP;**

  I/O で PULLUP 制約を使用するよう指定しています。
- **DEFAULT PULLUP = TRUE;**

  I/O で PULLUP を使用するよう指定しています。

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" pullup=true;
END;
```

PlanAhead™ の構文

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PWR_MODE

PWR_MODE (電力モード) 制約には、次の特徴があります。

- 高度なフィッタ制約です。
- 指定したエレメントをインプリメントするマクロセルのモードを Low または High (標準) に定義します。
- 指定したファンクションがファンアウト分割されている場合、PWR_MODE は適用されません。

アーキテクチャ サポート

XC9500 デバイスでのみサポートされます。

適用可能エレメント

- ネット
- すべてのインスタンス

適用ルール

- ネットに設定されている場合、ネットを駆動するすべての適用可能エレメントに適用されます。
- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

制約値

- LOW
- STD

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットまたはインスタンスに設定します。
- 属性名

  PWR_MODE
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute PWR_MODE: string;
```

VHDL 制約を次のように指定します。

```
attribute PWR_MODE of {signal_name|component_name|label_name}: {signal|component|label} is "{LOW|STD}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* PWR_MODE = "{LOW|STD} *)
```

UCF および NCF 構文

```
NET "$1187/$SIG_0" PWR_MODE=LOW;
```

ネット $SIG_0 をインプリメントするマクロセルを Low モードに指定しています。

XCF 構文

```
BEGIN MODEL "entity_name"
 NET  "signal_name" PWR_MODE={LOW|STD};
 INST "instance_name" PWR_MODE={LOW|STD};
END;
```

REG

REG (レジスタ) 制約には、次の特徴があります。

- 基本的なフィッタ制約です。
- CPLD マクロセルでレジスタがどのようにインプリメントされるかを指定します。

アーキテクチャ サポート

CPLD デバイスのみサポートされます。FPGA デバイスのサポートなし

適用可能エレメント

レジスタに適用

適用ルール

デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

制約値

- CE
  - CE 入力付きのフリップフロップ プリミティブに適用すると、CE 入力はマクロセルのクロック イネーブル積項を使用してインプリメントされます。
  - CE 入力のすべてのロジックが 1 つのクロック イネーブル積項でインプリメントできる場合、通常レジスタの CE 入力が使用されます。それ以外の場合は、REG=CE が指定されない限り、CE 入力は D または T ロジック表現に分解されます。
  - XC9500 デバイスではクロック イネーブル積項は使用できず、REG=CE は無視されます。XC9500XL デバイスでは、クロック イネーブル積項は CLR 入力と PRE 入力のいずれも使用しないレジスタでのみ使用できます。
- TFF
  - TFF を指定すると、レジスタは CPLD のマクロセルに T フリップフロップとしてインプリメントされます。
  - D フリップフロップ プリミティブに適用すると、D 入力が T 入力に変換され、T フリップフロップとしてインプリメントされます。
  - 通常、D フリップフロップと T フリップフロップの自動変換は、CPLD フィッタで実行されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- フリップフロップのインスタンスまたはフリップフロップを含むマクロに設定します。
- 属性名

  REG

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute REG: string;
```

VHDL 制約を次のように指定します。

```
attribute REG of signal_name: signal is "{CE|TFF}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* REG = {CE|TFF} *)
```

UCF および NCF 構文

```
INST "instance_name" REG = {CE|TFF};
```

次の文は、XC9500XL のマクロセルのクロック イネーブル積項を使用して CE ピン入力がインプリメントされるように指定します。

```
INST "Q1" REG=CE;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" REG={CE|TFF};
END;
```

RLOC

RLOC (相対ロケーション) 制約には、次のような特徴があります。

- 基本的なマップおよび配置制約です。
- 合成制約です。
- ロジック エレメントを独立した集合にグループ化します。
- デザイン全体での最終的な配置に関係なく、グループ内でのエレメント同士の位置を相対的に定義できます。
- ロジック ブロックを相対的に配置するように指定できるため、速度が向上し、チップ リソースを効率的に使用できます。
- FPGA チップ上で絶対的な配置を指定しなくても、関連するデザイン エレメントに順序と構造が指定されます。
- 既存のハード マクロに相当する、直接シミュレーション可能なマクロを作成できます。

グリッド システム

すべての FPGA アーキテクチャに、RLOC 制約を定義する際に使用できる 2 つの座標システムがあります。

- 元からあるグリッド システム

  すべてのコンポーネント タイプに対して共通の座標システムを使用しない
- RPM グリッド システム

  すべてのコンポーネント タイプに対して共通の座標システムを使用する

RPM グリッド システムを使用すると、ブロック RAM やスライス コンポーネントのような、さまざまなタイプのコンポーネントを含む再配置可能な RPM マクロを作成できます。

ユニファイド ライブラリでの RLOC の使用

ユニファイド ライブラリでは、次に RLOC 制約を使用できます。

- RLOC 制約を BUFT および CLB 関連のプリミティブ (FMAP)
- 非プリミティブのマクロ シンボル

BUFT シンボルで RLOC 制約を使用するには、いくつかの制限があります。詳細は、次の「集合の修正子」セクションを参照してください。RLOC 制約は、デコーダ、クロックには使用できません。

LOC 制約は次のようなすべてのプリミティブに使用できます。

- BUFT
- CLB
- CLB
- デコーダー
- クロック

RLOC 制約は、ロジック ブロックの相対的配置の制御に使用されますが、この制約を設定しても、同じ配線リソースがロジック ブロックを接続するためにインプリメンテーション間で使用されるとは限りません。使用される配線を制御するには、DIRECTED_ROUTING 制約を使用してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

どのデバイス ファミリにどのエレメントを使用できるかは、ライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

- レジスタ
- ROM
- RAMS、RAMD
- BUFT

  関連付けられている RPM に RLOC_ORIGIN が設定されており、RPM の RLOC 値が LOC 値に変換される場合のみ使用可能。
- LUT、MUXF5、MUXF6、MUXCY、XORCY、MULT_AND、SRL16、SRL16E、MUXF7

  Spartan®-3、Spartan-3A、Spartan-3E デバイスのみ
- MUXF8

  FPGA デバイスのみ
- ブロック RAM
- 乗算器
- DSP48

適用ルール

この制約はデザイン エレメント制約です。ネットへは接続できません。デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

NGDBuild は、デザイン階層の下位に LOC 制約を伝搬し、集合の要素ではない適切なオブジェクトに LOC 制約を追加します。RLOC 制約の伝搬は集合内に制限されますが、LOC 制約は開始点から下位階層にあるすべてのエレメントに適用されます。

デザインをフラット化すると、エレメントに設定された RLOC 制約の行番号と列番号は、そのエレメントより下位にある集合エレメントの RLOC 制約の行番号と列番号に追加されます。この機能を使用すると、プリミティブ シンボルに割り当てられている RLOC 値を変更せずに、サブモジュールやマクロの RLOC 値を修正できます。

制約の構文

スライスをベースとした XY 座標を使用して RLOC 制約を指定します。

RLOC=X*m*Y*n*

説明 :

- *m* は X 軸を表す整数です。
- *n* は Y 軸を表す整数です。

RPM グリッドの使用

相対配置 (RLOC) 制約は標準の RPM と同じように論理デザインのシンボルに適用されますが、グリッド値は異なります。RPM グリッドの座標は、該当するサイトを FPGA Editor でクリックして、履歴ウィンドウのグリッド座標を読み込むと指定されます。たとえば、下位の一番左のスライス サイトを選択すると、次のようになります。

site "SLICE_X0Y0", type = SLICE (RPM grid X3Y4)

元のグリッド システムのスライス X0Y0 が RPM グリッド システムでは X3Y4 と表示されます。このスライス用のシンボルにはすべて次の制約が適用されているはずです。

RLOC = X3Y4

FPGA Editor は、通常特定デバイスのグリッド値を探すために使用します。マクロの 1 つのシンボルには、RLOC 制約だけでなく、次の制約が適用されているはずです。

RPM_GRID = GRID

すべての合成ツールがこの RPM_GRID 属性をサポートするわけではないので、UCF 制約を使用してこの属性を割り当てる必要のあることもあります。

INST "*instance_name*" RPM_GRID = GRID

説明 :

instance_name はシンボル名への完全な階層パスになります。

集合の修正子

修正子は、デザイン エレメントに関連した RLOC 制約を修正します。修正子は集合に属するすべてのエレメントの RLOC 制約を修正するので、すべてのエレメントに簡単かつ直接的に伝搬されるように適用する必要があります。このため、集合の RLOC 修正子は、その集合の開始点に配置します。

RLOC 制約には、次のような集合修正子が適用されます。

- RLOC (相対位置)

  集合の階層内でそのエレメントの下位にあるほかの RLOC 制約の値を修正します。

  集合の種類に関係なく、エレメントの RLOC 値 (行、列、拡張子、XY 値) は、常に階層の下位に伝搬され、下位の階層レベルで同じ集合に含まれるエレメントの RLOC 制約に追加されます。

- RLOC_ORIGIN (相対位置の原点)

  集合のエレメントをチップ上の特定の位置に固定します。この制約を使用すると、RLOC 値を集合の構造を保った絶対的な LOC 制約に変更できます。

  RLOC_ORIGIN 制約から LOC 制約への変換は、NGCBuild により行われます。RLOC_ORIGIN の行と列の値は、下位階層のエレメントに RLOC 値を加えて行と列の値を変更した後、集合の各エレメントに追加されます。この後最終的な値が各プリミティブの LOC 制約になります。

- RLOC_RANGE (相対位置の範囲)

  集合の要素をチップ上の特定の範囲に制限します。

  この場合、集合は最終的に配置されるまで、1 つの単位としてその範囲内を移動できます。集合のすべての要素がその範囲内に収まる必要があるので、集合を含む大きさのエリアを指定する必要があります。

  BUFT シンボルを含む集合には、RLOC_RANGE 制約を使用できません。

- USE_RLOC (RLOC の使用)

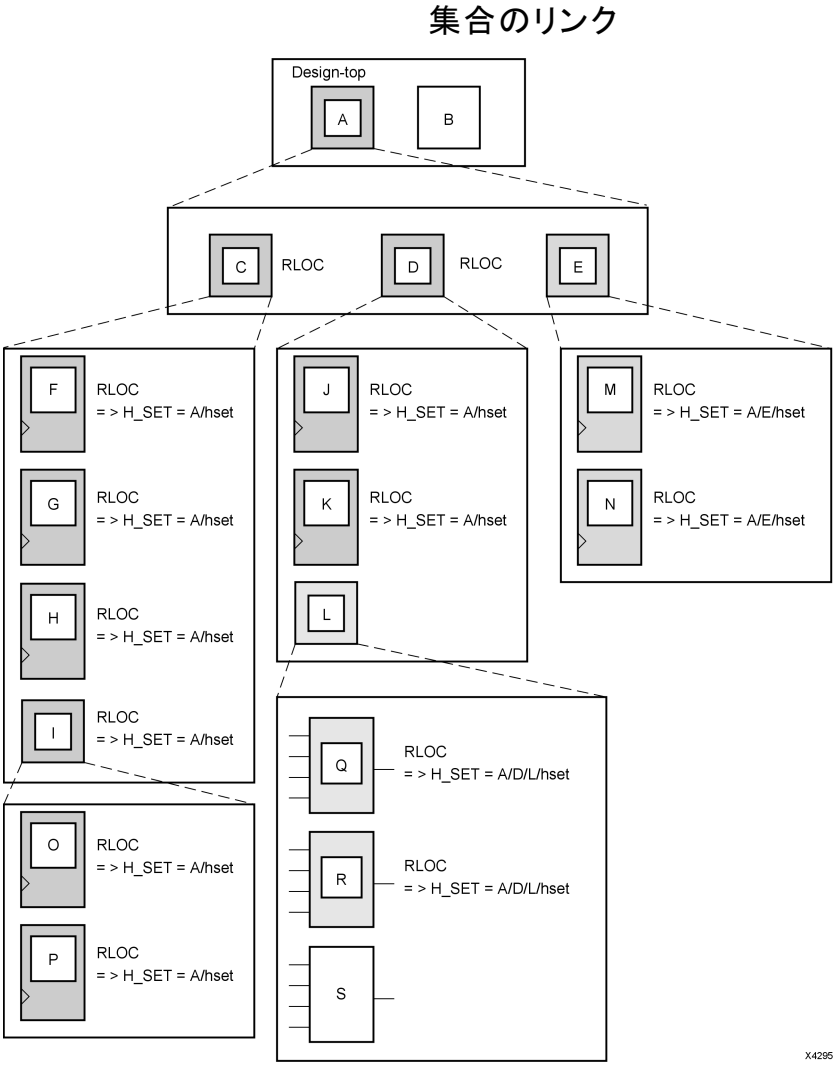
  集合の特定のエレメントまたは一部に対して、RLOC 制約を適用するかどうかを指定します。使用可能な値は、TRUE または FALSE です。

  階層に基づいて適用します。あるエレメントに設定された USE_RLOC 制約は、そのエレメントの下位にあり、同じ集合の要素であるすべてのエレメントに適用されます。集合の開始を定義するシンボルに適用されると、その制約が集合内の下位エレメントすべてに適用されます。

  USE_RLOC=FALSE を指定すると、RLOC および集合制約が NCD ファイルの影響のあるシンボルから削除されます。このプロセスは、RLOC_ORIGIN 制約の適用プロセスとは異なります。RLOC_ORIGIN の場合、すべての集合の制約および RLOC 制約に加えて LOC 制約が生成され、PCF ファイルに出力されます。USE_RLOC=FALSE が設定されていると、その後のプログラムではオフになるので、マップでは元の制約が維持されません。

  USE_RLOC 制約をプリミティブ シンボルに直接設定すると、そのシンボルのみに制約が適用されます。

集合のリンク



この例は、デザイン階層の異なるエレメントをリンクするプロセスを示しています。RLOC を指定するには、RLOC=R *m*C*n* か RLOC=X*m*X*n* を使用する必要があります。

注記 : このセクションの図では、わかりやすいように集合に影をつけています。

デザイン階層の 1 つのノードにある RLOC 制約が設定されたデザイン エレメントはすべて、RLOC_ORIGIN や RLOC_RANGE などのほかの種類の集合制約を設定しない限り、同じ H_SET 集合に属すると見なされます。この図では、プリミティブおよび非プリミティブの C、D、F、G、H、I、J、K、M、N、O、P、Q、R に RLOC 制約が設定され、B、E、L、S には RLOC 制約は設定されていません。マクロ C と D はノード A で RLOC 制約が付いているので、RLOC 値の付いた C と D よりも下位のプリミティブはすべて 1 つの H_SET 集合にまとめられます。

この H_SET 集合の名前は、この集合がノード A で開始されるので **A/h_set** になります。H_SET 集合の開始はその HU_SET 集合の RLOC 制約の付いたすべてのエレメントに共通の親で、最も下位にあるものです。

エレメント E には RLOC 制約が設定されていないため、A/h_set 集合にはリンクされません。RLOC 制約が設定されているエレメント M と N は、エレメント E の下位にあるので、別の H_SET 集合の要素になります。その H_SET の開始は、名前が A/E/h_set だとすると A/E です。

同様に、プリミティブ Q と R は、エレメント L がほかのデザイン エレメントにリンクされていないため、別の H_SET 集合になります。これらの H_SET 集合の最下位にある共通の親は、名前が A/D/L/h_set だとすると L です。フラットにされると、NGDBuild では次の表に示すプリミティブに集合が設定されます。

集合	プリミティブ
H_SET=A/h_set	F、G、H、O、P、J、K
H_SET=A/D/L/h_set	Q、R
H_SET=A/E/h_set	N

集合がデザインの最上位で作成された場合を考えます。マクロ A にも RLOC 制約が付く場合、A はデザインの最上位であり、親は存在しないので、最下位の共通の親というものがありません。この場合、ベース名の **h_set** に階層名が頭に付くことはなく、H SET 集合の名前は単純に **h_set** となります。

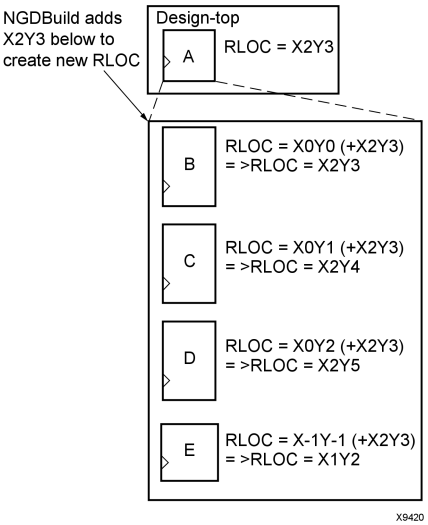
集合の編集

RLOC 制約は、プリミティブに RLOC 値 (行と列の番号で指定し、オプションで拡張子あり) を割り当て、集合に含まれるエレメントを指定し、別の階層レベルにあるエレメントを一緒にリンクするために使用できます。「3 つの H_SET 集合」の例で、マクロ C と D に設定された RLOC 制約は、これらマクロの下位にある RLOC 制約が設定されたすべてのオブジェクトをリンクします。RLOC 制約は、階層のそれよりも下位にある制約の RLOC 値を変更するために使用することもできます。つまり、エレメントの RLOC 値は、デザイン階層で特定のエレメントより下位にある同じ H_SET 集合のほかのエレメントの RLOC 値に影響を与えます。

デザインをフラット化すると、エレメントの RLOC 制約の XY 値は、そのエレメントより下位の階層にある集合エレメントの RLOC 制約の XY 値に加えられます。この機能を使用すると、プリミティブ シンボルに割り当てられている RLOC 値を変更せずに、サブモジュールやマクロの RLOC 値を修正できます。

次のセクションでは、集合の編集をすることで階層に及ぼす影響について説明します。

下位階層に RLOC 値を加算する方法 (スライスをベースとした XY 座標)

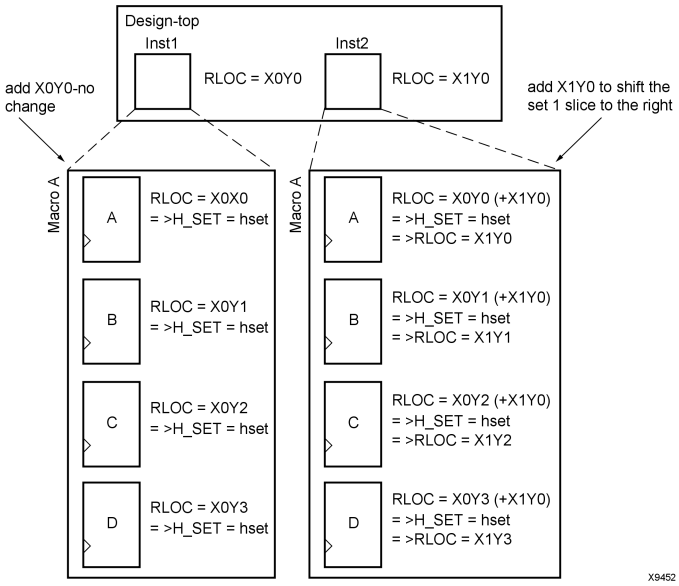


この例では、RLOC 値を下位階層に追加するプロセスを示します。括弧内の行と列の値は、マップで実行される追加ファンクションを表します。先頭に => が付いている斜体文字は、デ

ザイン決定プロセスのときに自動的に追加され、ユーザーが追加したオリジナルの RLOC 制約の代わりに使用されます。

同じマクロの RLOC 値の修正と 1 つの集合としてのリンク

下位階層の RLOC 値を修正する機能は、同じマクロを何度もインスタンシエートする際に特に便利です。通常、マクロはインスタンシエート時に、修正された RLOC 制約を使用してデザインされます。

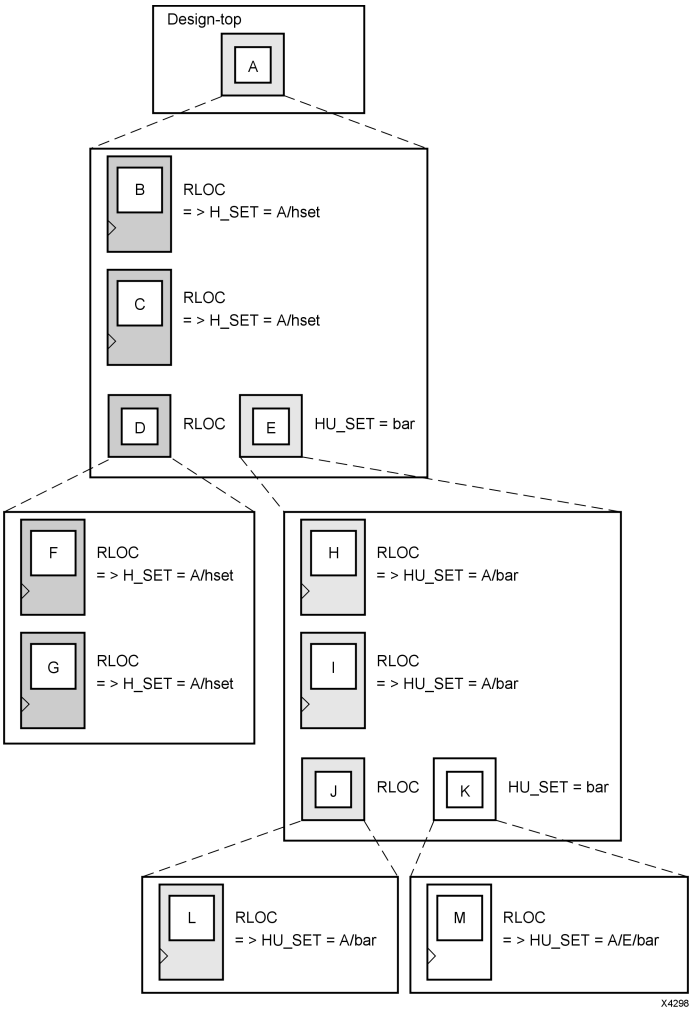


この例は、前の例のバリエーションです。Inst1 および Inst2 の RLOC 制約は、今度は 1 つの H_SET 集合にすべてのオブジェクトをリンクしています。

Inst1 マクロの RLOC=X0Y0 修正子はそれより下位のオブジェクトに影響しないので、H_SET がオブジェクトに追加されるだけで、RLOC 値はそのままです。Inst2 マクロの RLOC=X0Y1 修正子の場合は、斜体文字で示すように、H_SET が追加されるだけでなく、それより下位にあるオブジェクトの RLOC 値が変更されます。

H_SET 集合からのエレメントの分離

HU_SET (HU 集合) 制約は、H_SET (階層集合) のバリエーションです。HU_SET 制約は、新しい集合の開始点を定義します。HU_SET は H_SET と同様、デザイン階層で定義されますが、HU_SET を使用するとユーザー定義の名前を集合に割り当てることができます。



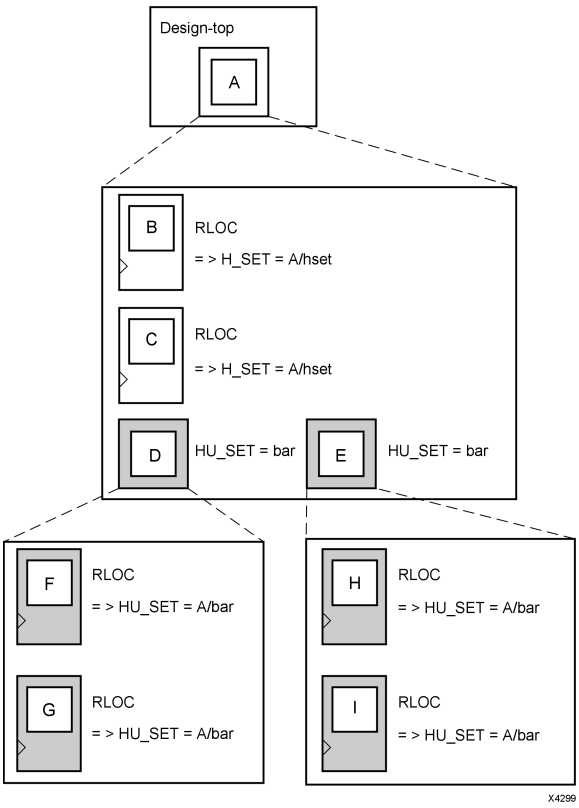
この例は、HU_SET 制約で集合要素としてエレメントを指定する方法、階層内の RLOC 制約の付いたエレメント間のリンクを解除して H_SET 集合から分離させる方法、これらの集合の識別子として名前を生成する方法などを示しています。

E に設定されたユーザー定義の HU_SET 制約は、その下位にあるデザイン エレメント H、I、J、K、L、M を、プリミティブ要素 B、C、F、G を含む H_SET=A/h_set から分離します。E で定義された HU_SET 集合には、エレメント H、I および J を介した L が含まれます。A は HU_SET 集合に含まれるすべてのエレメントの共通の親です。

HU_SET 集合のすべてのエレメントに共通の親で最も下位にあるのが A であるため、MAP はエレメント E の名前「bar」に階層修飾子を付けて「A/bar」とし、これをプリミティブ H、I、L に追加します。K の HU_SET 制約は M を含む別の集合を開始し、この集合にはマップ処理後に HU_SET=A/E/bar が追加されます。

2 つの HU_SET 制約には同じ名前のフィールドが使用されますが、異なる階層レベルにあるシンボルに適用されているため、別々の集合として定義されます。

2 つの HU_SET 集合のリンク

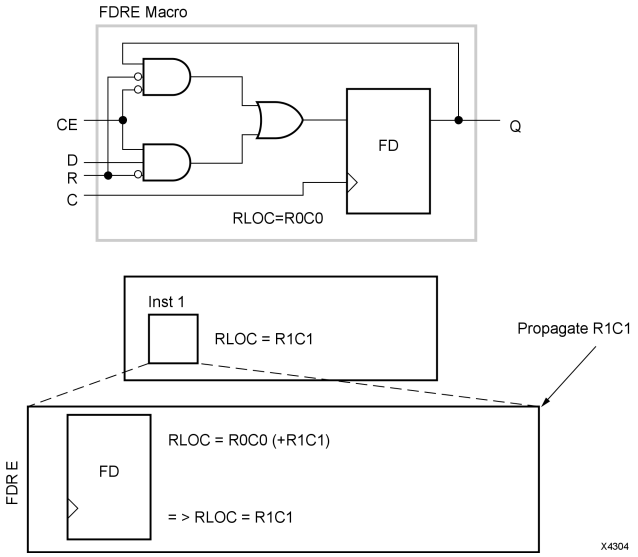


この例は、同じ識別子の付いた名前を付けることで、同じノードのエレメントを HU_SET 制約がどのようにリンクしているかを示しています。2 つのエレメント (D および E) には同じ名前 (bar) が付いているので、D および E よりも下位の RLOC 制約の付いたエレメントは同じ HU_SET の一部になります。

ザイリンクス マクロでの RLOC 制約の使用

ザイリンクスが提供するフリップフロップ マクロには、下位のプリミティブに RLOC=R0C0 制約が含まれるため、マクロ シンボルに RLOC を設定できます。このシンボルは、下位プリミティブをそのマクロ シンボルを含む集合にリンクします。

単に、ザイリンクスのフリップフロップ マクロのインスタンシエーションに適切な RLOC 制約を付けてください。マップでは下位プリミティブに指定した RLOC 値が追加され、必要な値が設定されます。



この例では、マクロのインスタンス (Inst1) に RLOC = R1C1 制約が設定されています。これはマクロ内のフリップフロップに設定された RLOC 制約の R0C0 値に追加され、新しい RLOC 値 (R1C1) が得られます。

RLOC=X1Y1 制約をマクロの Inst1 に設定した場合、これがマクロ内のフリップフロップに適用された RLOC 制約の X0Y0 値に追加され、新しい RLOC 値 (X1Y1) が得られます。

フリップフロップ マクロ シンボルに RLOC 制約を付けない場合、下位プリミティブ シンボルは集合の唯一の要素となります。マップでは、集合の唯一の要素であるプリミティブや下位に RLOC オブジェクトがないマクロから RLOC 制約が取り除かれます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- インスタンスに設定します。
- 属性名

  RLOC
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute rloc: string;
```

VHDL 制約を次のように指定します。

```
attribute rloc of {component_name|entity_name| label_name}
: {component |entity|label} is "[element]X mYn[ .extension]";
```

有効な値については、「相対位置を指定するためのガイドライン」を参照してください。

次のコード例は、VHDL の generate 文で RLOC 制約を使用する方法を示しています。このコードは、複数のインスタンシエート済み FDE コンポーネントの RLOC 制約が自動生成されるようにする単純な例です。この方法は、どのプリミティブに対しても使用できます。

注記 : ユーザーは、**itoa** ファンクションを作成する必要があります。

```
LEN:for i in 0 to bits-1 generate
 constant row :natural:=((width-1)/2)-(i/2);
 constant column:natural:=0;
 constant slice:natural:=0;
 constant rloc_str : string := "R" & itoa(row) & "C" & itoa(column) & ".S" & itoa(slice);
 attribute RLOC of U1: label is rloc_str;
begin
 U1: FDE port map (
   Q=> dd(j),
   D=> ff_d,
   C=> clk,
   CE =>lcl_en(en_idx));
end generate LEN;
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* RLOC = "[element ] XmY n[ .extension ] " *)
```

有効な値については、「相対位置を指定するためのガイドライン」を参照してください。

UCF および NCF 構文

FPGA デバイスの場合、次の文は、X 座標の +4 と Y 座標の +4 にあるスライスに FF1 のインスタンスが配置されるように指定します。

```
INST "/V2/design/FF1" RLOC=X4Y4;
```

XCF 構文

Virtex®-4 および Virtex-5 デバイスの場合 :

```
BEGIN MODEL "entity_name "
 INST "instance_name " rloc=[element]XmYn [ .extension] ;
END;
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

相対ロケーション制約の指定方法

相対位置にエレメントを割り当てるスライス ベースの座標システムでは、次の一般的な構文が使用されます。

RLOC=X*m* **Y***n*

- m

  X 軸 (左/右) の相対値

- n

  Y 軸 (上/下) の相対値

- X および Y は次のいずれかの値にできます。
  - 0
  - 正の整数
  - 負の整数

整数

相対ロケーション (RLOC) 制約の X 値と Y 値はデザイン エレメント間の順序と関係を指定するだけで、絶対的なチップ位置を定義するわけではないので、番号に負数を使用できます。

相対ロケーション制約には任意の整数を使用できますが、簡単でわかりやすいように、小さい数の使用をお勧めします。

絶対値および相対値

RLOC の指定で重要なのは、X 座標と Y 座標の絶対値ではなく、相対値 (差) です。

たとえば、デザイン エレメント A に RLOC=X3Y4 という制約を、デザイン エレメント B に RLOC=X6Y7 という制約を指定した場合、X 座標の絶対値 (3 と 6) は重要ではありません。重要なのはこれらの値の差であり、この場合は 6 から 3 を引いた値 3 で、デザイン エレメント B がデザイン エレメント A から 3 スライス離れるように指定されます。

この情報を得るため、値は正規化されます。デザイン エレメント B は、デザイン エレメント A から上方向に 3 列 (7 から 4 を引いた値) 離れた位置に指定されます。上記の例では、正規化によってデザイン エレメント A の RLOC が X0Y0 に、デザイン エレメント B の RLOC が X3Y3 になります。

高レベル デバイスのスライス番号

Spartan®-3 および Virtex®-4 以降のデバイスの場合、XY 軸を使用してチップの左下隅からスライスに番号が付けられています。

- X 軸では右に移動するに従って増加し、
- Y 軸では上に移動するに従って増加します。

相対ロケーション制約はデカルト座標の変換方式に従います。

4 つのフリップ フロップ プリミティブの RLOC 指定

次の図は、相対ロケーション制約の使用例を示しています。

図 (a) では、A、B、C、D という 4 つのフリップに相対ロケーション制約を割り当てています。これらの RLOC 制約では、各フリップフロップは異なるスライスに配置され、図のように下から A、B、C、D という順に並びます。

2 つ以上のフリップフロップ プリミティブを 1 つのスライスに配置するには、図に示すように RLOC 制約を指定します。この場合、A と B は 1 つのスライスに配置され、そのすぐ右側のスライスに C と D が配置されます。

相対ロケーション (RLOC) 集合

相対ロケーション (RLOC) 制約では、関連するデザイン エレメントの順序および構造を指定します。

相対ロケーション (RLOC) 集合とは、RLOC 制約の付いた関連するデザイン エレメントのグループのことです。

- 集合に含まれるエレメントは、同じ集合内のほかのエレメントと RLOC 制約で関連付けられます。
- 集合内のエレメント同士を関連付けるため、集合内の各エレメントに RLOC 制約を設定する必要があります。
- ユーザーは複数の集合を作成することができますが、1 つのデザイン エレメントは 1 つの集合にしか含めることができません。

たとえば、RLOC 仕様の異なる 4 つのフリップフロップ プリミティブは RLOC 制約で関連付けられ、1 つの集合を形成します。

集合の定義

RLOC 集合は次のいずれかの方法で定義できます。

- 集合パラメーターを使用して明示的に定義

  または

- デザイン階層の構造を使用して暗示的に定義

集合の規則

次の規則は、各 RLOC 集合に関連するものです。

- 定義規則

  集合のメンバーの要件を定義します。

- リンク規則

  1 つの集合を形成するためのエレメント同士のリンク方法を指定します。

- 修正規則

  集合に含まれるメンバーすべての RLOC 値を修正するパラメーターの指定方法を決定します。

- 命名規則

  集合の命名方法を指定します。

集合制約

エレメントには、相対ロケーション (RLOC) 制約と次の集合制約のいずれかの両方を付ける必要があります。

- U_SET
- H_SET
- HU_SET

U_SET

U_SET 制約を使用すると、デザイン階層に分散される RLOC 制約を付けることで、デザイン エレメントを 1 つの集合にまとめることができます。U_SET の U は、この集合がユーザー定義であることを示します。

U_SET 制約を使用すると、エレメントがデザイン階層で直接関連付けられていなくてもエレメントを一緒のグループにできます。U_SET 制約をデザイン エレメントに付けると、集合のメンバーを明示的に定義できます。

U_SET 制約の付いたデザイン エレメントは、デザイン階層のどの部分にでも配置できます。U_SET 制約は、プリミティブ、非プリミティブ シンボルのどちらにも設定でき非プリミティブ シンボルに設定した場合、U_SET 制約は、そのシンボルの下位階層にあり、相対ロケーション (RLOC) 制約が設定されているプリミティブ シンボルにも適用されます。

U_SET の構文は次のようになります。

U_SET=*set_name*

set_name はユーザーが指定する集合の識別名です。

制約が指定されているデザイン エレメントで、同じ U_SET 制約を設定したものはすべて、同じ集合に属します。そのため、名前がデザイン内のほかの集合名と重複しないようにしてください。

H_SET

U_SET 制約はデザイン エレメントに設定して明示的に定義できますが、H_SET (階層集合) 制約はデザイン階層によって暗示的に定義されます。H_SET 集合 (階層集合)は次を組み合わせて定義されます。

- デザイン階層

  および

- エレメントの相対ロケーション (RLOC) 制約

帰属関係を表すために、制約をデザイン エレメントに設定する必要はありません。集合はデザイン階層によって自動的に定義されます。

デザイン階層の 1 つのノードにある RLOC 制約が設定されたデザイン エレメントはすべて、RLOC_ORIGIN や RLOC_RANGE などのほかの種類の集合制約を設定しない限り、同じ H_SET 集合に属すると見なされます。

次のいずれかの制約をエレメントに設定すると、そのエレメントは H_SET 集合から除外されます。

- RLOC_ORIGIN (相対位置の原点)
- RLOC_RANGE (相対位置の範囲)
- U_SET
- HU_SET

H_SET 制約は相対配置マクロの基本的なメカニズムであるため、ほとんどのデザインには H_SET 制約のみが含まれます。RLOC_ORIGIN および RLOC_RANGE 制約の詳細は、「集合の修正子」を参照してください。

NGDBuild には、次のような機能があります。

1. 暗示的に定義された H_SET 集合を認識
2. その名前 (識別名) を導き出す
3. H_SET 制約を集合のエレメントに設定
4. レジスタ ファイルに書き込む

HU_SET

HU_SET は暗示的に定義される H_SET のパターンの 1 つです。HU_SET は H_SET と同様、デザイン階層で定義されますが、HU_SET を使用するとユーザー定義の名前を集合に割り当てることができます。

HU_SET の構文は次のようになります。

HU_SET=*set_name*

- *set_name* は、集合の識別名です。
- *set_name* には、デザイン内のほかの集合名とは異なる名前を指定する必要があります。

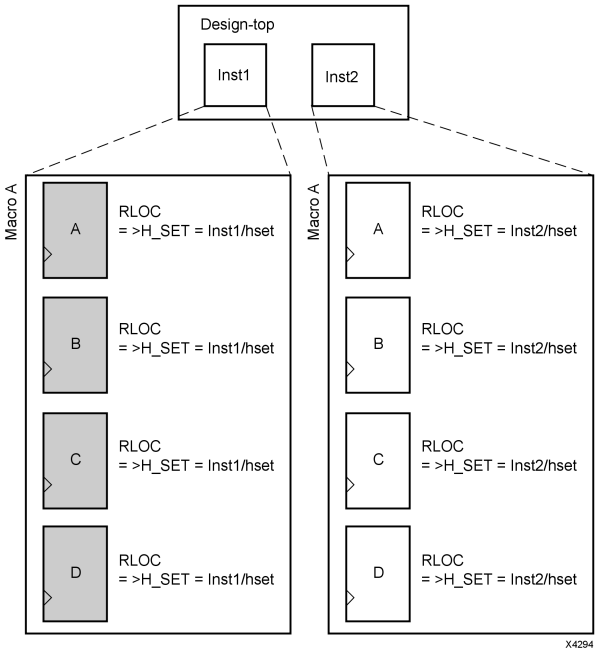
このユーザー定義の名前は、HU_SET 集合のベース名になります。H_SET 集合の場合、集合エレメントの共通の親で最も下位にあるものの階層名がベース名「h_set」の前に付けられます。集合の場合も同様で、集合エレメント共通の親で最も下位にあるものの階層名がユーザー定義のベース名の前に付けられます。

集合に階層で識別された固有の名前が付けられるようにするため、デザインがマップされ階層名が接頭辞として付けられる前に、ベース名を定義する必要があります。

HU_SET 制約は、新しい集合の開始点を定義します。同じノードにあるデザイン エレメントで、HU_SET 制約のユーザー定義の値が同じであるものは、すべて同じ HU_SET 集合のエレメントになります。HU_SET 制約と共に RLOC 制約もエレメントに設定できます。

H_SET 制約と RLOC 制約が設定されたエレメントは、階層の上位と下位にあるエレメントで RLOC が設定されているものにリンクされます。HU_SET 制約の場合は、RLOC 制約を一緒に設定しても、デザイン エレメントは、同じ階層レベルまたはそれより上位にある RLOC 制約が設定されたほかのエレメントに自動的にはリンクされません。

2 度インスタンシエートされるマクロ A



注記 : この図と次のセクションで示す図では、=> の後の斜体文字は、デザイン フラット化の際に によって追加されます。ほかのすべての文字は、ユーザーが入力します。

この図は、間接的に定義した H_SET を使用したところを示しています。この図には、制約の最初の RLOC 部分のみが表示されています。

実際のデザインでは、RLOC 制約は次のように指定します。

RLOC=R *m*C*n*

Spartan®-3 以降および Virtex®-4 以降のデバイスの場合、RLOC 制約は次のように指定します。

RLOC=X*m*Y*n*

この例では、マクロ **A** が元々、次の 4 つのフリップフロップの RLOC 制約を使用して設計されています。

- A
- B
- C
- D

この後、マクロは次のように 2 度インスタンシエートされます。

- Inst1
- Inst2

RLOC 制約が設定されたエレメントが 2 つの異なる階層レベルに含まれているため、デザインのフラット化の際に 2 つの異なる H_SET 集合が認識されます。NGDBuild では、は、集合の各エレメントに該当する H_SET 制約が設定されます。

- **Inst1** にインスタンシエートされるマクロの場合は **H_SET=Inst1/h_set**
- **Inst2** にインスタンシエートされるマクロの場合は **H_SET=Inst2/h_set**

デザイン インプリメンテーション プログラムは、RLOC 制約で指定された集合内の相対的な順序を使用して、2 つの集合をそれぞれ 1 つの単位として別々に配置します。ただし、2 つの集合は、互いに完全に独立していると見なされます。

H_SET 集合の名前は、すべての RLOC エレメントを含む階層のシンボルまたはノードから取得されます。**Inst1** は図の左側に示した RLOC が設定された 4 つのフリップフロップ エレメントを含むノード (インスタンシエーションされるマクロ) なので、したがって、この H_SET 集合の名前には、階層名「Inst1」の後に「h_set」を付けます。

Inst1 シンボルは H_SET 集合の開始点と考えられ、これを H_SET 全体の名前として H_SET 全体を修正する制約を設定します。集合を修正する制約については、「ネット フラグの保存 (SAVE NET FLAG)」制約を参照してください。

この図は、階層の 1 つのレベルに制限される集合について、簡単な使用例を示しています。リンクと修正を使用すると、複数の階層レベルにわたってリンクされた H_SET 集合も作成できます。

リンクを使用すると、異なる階層レベルにあるエレメントをリンクし、集合を形成できます。修正を使用すると、複数の階層にまたがる集合のエレメントの RLOC 値を修正できます。

RLOC 集合のサマリ

RLOC 集合のタイプと集合のエレメントを識別する制約を、次の表に要約します。

集合タイプのサマリ

タイプ	定義	命名	リンケージ	修正
U_SET= name	ユーザーが指定する U_SET 値が同じエレメントは、すべて同じ U_SET 集合のエレメントになります。	集合名は、階層の修飾子を除いたユーザー定義の名前と同じです。	U_SET は、U_SET 制約の値が同じエレメント同士をリンクします。	U_SET は、U_SET を指定したエレメントのいずれか、多くとも 1 つに RLOC_ORIGIN または RLOC_RANGE 制約を適用すると、修正されます。
HU_SET= name	階層の修飾子が付いた名前が同じエレメントは、すべて同じ集合の要素となります。	エレメントの共通の親をユーザー定義名の前に付け、集合名とします。	同じノードにあり、HU_SET 値が同じエレメントをリンクします。また、階層の下位にある RLOC が設定されたエレメントもリンクします。	集合の開始点は、同じノードにある HU_SET 値が同じエレメントで構成されます。HU_SET 集合の開始エレメントのいずれか、多くとも 1 つに、RLOC_ORIGIN または RLOC_RANGE 制約を適用できます。

RLOC_ORIGIN

RLOC_ORIGIN 制約には、次の特徴があります。

- 配置制約です。
- チップ上の特定の位置に集合エレメントを固定します。
- 単一の位置を指定する必要があり、位置の範囲や複数の位置のリストは指定できません。

  詳細は、「RLOC」の「集合の修正子」を参照してください。
- BUFT シンボルを含む集合に必要です。
- BUFT インスタンスには設定できません。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

集合の要素であるインスタンスまたはマクロに適用されます。

適用ルール

- この制約はマクロの制約なので、ネットには適用できません。
- RLOC_ORIGIN 制約を H_SET (階層集合) と共に使用する場合は、H_SET 集合の開始点となるエレメント、すなわち集合の全エレメントに共通の親に設定する必要があります。
- RLOC_ORIGIN 制約を HU SET 制約に適用する場合は、HU_SET 集合の開始点となるエレメント、すなわち HU_SET 制約が設定されているエレメントに配置します。
- 同じノードで HU_SET 制約を使用して複数のエレメントをリンクする場合は、複数の RLOC_ORIGIN 制約が HU_SET 集合に適用されないようにするため、RLOC_ORIGIN 制約は 1 つのエレメントのみに設定してください。
- 同様に、RLOC_ORIGIN 制約を U_SET 制約と共に使用する場合、U_SET 制約が設定されているエレメントの 1 つにのみ設定します。RLOC 制約のみが設定されたエレメントに RLOC_ORIGIN 制約を設定すると、そのエレメントの帰属関係はすべての集合で削除され、そのエレメントが、RLOC_ORIGIN 制約が設定された H_SET 集合の開始点と見なされます。

制約の構文

RLOC 集合に 1 つの基点を指定するには、次の構文を使用します。これは、回路図で RLOC_ORIGIN 制約を配置する場合と同じです。

set_name **RLOC_ORIGIN=X***m* **Y***n*

- set_name

  次の RLOC 集合のタイプ名になります。
  - U_SET
  - HU_SET
  - システムで生成された H_SET
- 基点は、RLOC=X0Y0 にあるエレメントの位置を示す X 値と Y 値で表されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 集合の要素であるインスタンスに設定します。
- 属性名

  RLOC_ORIGIN
- 属性値

  制約値のリストは、次の「UCF および NCF 構文」セクションを参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute rloc_origin: string;
```

VHDL 制約を次のように指定します。

```
attribute rloc_origin of {component_name|entity_name|label_name} : {component|entity |label}
is "value";
```

Spartan®-3、Spartan-3A、Spartan-3E、Virtex®-4、Virtex-5 デバイスの場合、*value* は **X** *m***Y***n* となります。

制約値のリストは、次の「UCF および NCF 構文」セクションを参照してください。

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* RLOC_ORIGIN = " value" *)
```

Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスの場合、*value* は **X** *m***Y***n* となります。

制約値のリストは、次の「UCF および NCF 構文」セクションを参照してください。

スライスをベースとして XY 座標を使用する UCF および NCF 構文

この制約は、Spartan-3、Spartan-3A、Spartan-3E、Virtex-4 および Virtex-5 デバイスに適用できます。

RLOC_ORIGIN=X *m***Y***n*

- m

  相対的な X 座標を表す次のいずれかになります。

  - 0
  - 正の整数
  - 負の整数

- n

  相対的な Y 座標を表す次のいずれかになります。

  - 0
  - 正の整数
  - 負の整数

次の文は、集合の要素である FF1 のインスタンスが、X4Y4 にあるスライスを基準として配置されるように指定します。

```
INST "/archive/designs/FF1" RLOC_ORIGIN=X4Y4;
```

たとえば、FF1 に RLOC=X0Y2 を設定した場合、FF1 のインスタンスは、次の位置にあるスライスに配置されます。

- X4 から右に 0 列
- Y4 から上に 2 行 (X4Y6)

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

RLOC_RANGE

RLOC_RANGE (相対ロケーション範囲) 制約には、次のような特徴があります。

- 配置制約です。
- RLOC_ORIGIN 制約と似ていますが、集合の要素をチップ上の特定の範囲に制限するという点で異なります。

  ロケーションの範囲や複数の位置のリストは、集合の基点だけでなく、RLOC が設定されている適用可能なすべてのエレメントに適用されます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

集合の要素であるインスタンスまたはマクロに適用されます。

適用ルール

- この制約はマクロの制約なので、
- ネットには設定できません。
- 矩形領域は、集合の基点だけでなく、相対配置マクロのすべてのエレメントに適用されます。
- RLOC_RANGE 制約の値は、エレメントの RLOC 値に加えられません。
- RLOC_RANGE (相対ロケーション範囲) 制約には、次のような特徴があります。
  - 下位エレメントの RLOC 制約の値は変更しません。
  - MAP により集合の全エレメントに自動的に設定される追加の制約です。
  - RLOC_RANGE 制約は、RLOC_ORIGIN 制約とまったく同じ方法でデザイン エレメントに設定されます。
  - RLOC_RANGE 制約の値は、RLOC_ORIGIN の値と同様にチップの位置に対応するので、0 以外の正の数を指定する必要があります。
- デザイン ネットリストと UCF ファイルの両方で RLOC_ORIGIN 制約または RLOC_RANGE 制約が RLOC 集合に設定されている場合、UCF ファイルの制約がネットリストの制約よりも優先されます。

制約の構文

RLOC_RANGE=X*m1* **Y***n1***:X** *m2***Y***n2*

相対的な X 値 (m1、m2) とY 値 (n1、n2) には、次を指定できます。

- 0 以外の正の整数
- ワイルドカード (*)

この構文では、次のような 3 種類の範囲指定が可能です。

- **X***m1*Y*n1*:X*m2* Y*n2*

  X*m1*Y*n1* と **X***m2* Y*n2* で囲まれた矩形領域

- **X*Y***n1*:X*Y*m2*

  Y 軸の *n1* から *n2* までの領域 (X の値は問わない)

- **X***m1*Y*:X*m2*

  X 軸の *m1* から *m2* までの領域 (Y の値は問わない)

ワイルドカードを使用する 2 番目と 3 番目の指定の場合、区切り記号のコロンの前後でワイルドカード文字 (*) を異なる位置に適用すると、エラーが発生します。たとえば、X*Y1:X2Y* と指定すると、コロンの左側では X 値に、コロンの右側では Y 値にワイルドカードが適用されているため、エラーになります。

範囲を指定するには、次の構文を使用します。これは、回路図で RLOC_RANGE 制約を配置するのと同じです。

set_name **RLOC_RANGE=X** *m1***Y***n1* **:X***m2***Y** *n2*

範囲は、長方形のエリアが識別されます。範囲を指定する X 値または Y 値の代わりに、ワイルドカード文字 (*) を使用できます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 集合の要素であるインスタンスに設定します。
- 属性名

  RLOC_RANGE

- 属性値
  - 正の整数 (0 を含む)
  - ワイルドカード (*)

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute rloc_range: string;
```

VHDL 制約を次のように指定します。

```
attribute rloc_range
of {component_name|entity_name|label_name}: {component|entity|label}
is "value";
```

Spartan®-3、Spartan-3A、Spartan-3E、Virtex®-4、Virtex-5 デバイスの場合、*value* は次のようになります。

X*m1***Y***n1***:X***m2***Y***n2*

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

(* RLOC_RANGE = "*value*" *)

Spartan-3、Spartan-3A、Spartan-3E、Virtex-4、Virtex-5 デバイスの場合、*value* は次のようになります。

X*m1***Y***n1***:X***m2***Y***n2*

UCF および NCF 構文

このセクションは、Spartan-3 および Virtex-4 以降のデバイスにのみ適用されます。

RLOC_RANGE=X*m1***Y***n1***:X***m2***Y***n2*

相対的な X 値 (m1、m2) とY 値 (n1、n2) には、次を指定できます。

- 正の整数 (0 を含む)
- ワイルドカード (*)

次の文は、マクロ MACRO4 のインスタンシエーションが、集合内のほかのエレメントに相対して、次で囲まれた矩形領域に配置されるように指定します。

- 左下角の X4Y4
- 右上角の X10Y10

```
INST "/archive/designs/MACRO4" RLOC_RANGE=X4Y4:X10Y10;
```

XCF 構文

```
MODEL "entity_name" rloc_range=value;
 BEGIN MODEL "entity_name"
 INST "instance_name" rloc_range=value;
END;
```

SAVE_NET_FLAG

SAVE NET FLAG 制約には、次のような特徴があります。

- 基本的なマップ制約です。
- ネットまたは信号に設定すると、未接続の信号が削除されなくなり、デザインのマップや配置配線に影響します。
- ロードのない信号やドライバーのない信号が削除されないようにします。
  - ロードのない信号の場合、SAVE NET FLAG 制約がその信号に接続された疑似 OBUF ロードの働きをします。
  - ドライバーのない信号では、SAVE NET FLAG 制約はその信号に接続された疑似 IBUF ドライバーの働きをします。
- S NET FLAG に省略できます。

ネットに SAVE NET FLAG 制約が設定されていない場合、I/O プリミティブへのパスを介して確認または制御されない信号はすべて削除されます。

SAVE NET FLAG 制約により、信号に接続されたロジックがトリムされなくなる場合もあります。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

- ネット
- 信号

適用ルール

- ネットまたは信号に設定します。デザイン エレメントには適用できません。
- 未接続の信号が削除されないようにします。ネットに S 制約が設定されていない場合、ロジックや I/O プリミティブに接続されていない信号はすべて削除されます。

制約値

- YES
- NO
- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットまたは信号に設定します。
- 属性名

  S (SAVE NET FLAG)
- 属性値
  - TRUE
  - FALSE

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute S: string;
```

VHDL 制約を次のように指定します。

```
attribute S of signal_name : signal is "{YES|NO|TRUE|FALSE }";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* S = {YES|NO|TRUE|FALSE} *)
```

UCF および NCF 構文

```
NET $SIG_9 S;
```

ネットまたは信号 $SIG_9 が削除されないように指定しています。

XCF 構文

```
BEGIN MODEL entity_name
 NET "signal_name" s=true;
END;
```

SCHMITT_TRIGGER

SCHMITT_TRIGGER (シュミット トリガー) 制約には、次のような特徴があります。

- 入力パッドがシュミット トリガー (ヒステリシス) でコンフィギュレーションされるよう設定します。
- 入力パッドに適用されます。

アーキテクチャ サポート

CoolRunner™-II デバイスでのみサポートされます。

適用可能エレメント

すべての入力パッドおよびパッド ネットに適用されます。

適用ルール

ネットまたは信号に設定します。マクロ、エンティティ、モジュールには適用できません。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットに設定します。
- 属性名

  SCHMITT_TRIGGER
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute SCHMITT_TRIGGER: string;
```

VHDL 制約を次のように指定します。

```
attribute SCHMITT_TRIGGER of signal_name : signal is
"{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* SCHMITT_TRIGGER = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

```
NET "mysignal" SCHMITT_TRIGGER;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" SCHMITT_TRIGGER=true;
END;
```

SIM_COLLISION_CHECK

SIM_COLLISION_CHECK 制約は、ブロック RAM のメモリ ロケーションに対して読み出しと書き込みの競合が発生した場合のシミュレーション モデル ビヘイビアーを指定するために使用します。

アーキテクチャ サポート

この制約は、Virtex®-4 およびそれ以降のデバイスにのみ適用できます。

適用可能エレメント

ブロック RAM プリミティブ エレメントに適用できます。

適用ルール

ネットまたはマクロには設定できません。

制約値

- ALL

  シミュレーション中、出力に警告メッセージと X の両方を生成します。

- NONE

  シミュレーション中、予測不可能な結果になる競合を無視します。

- WARNING_ONLY

  Virtex-4 ブロック RAM のメモリ ロケーションで読み出し/書き込みの競合があると、シミュレーション中に警告メッセージを生成します。

- GENERATE_X_ONLY

  シミュレーション中、出力に X を生成します。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図構文

- ブロック RAM のプリミティブに設定します。
- 属性名

  SIM_COLLISION_CHECK

- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute sim_collision_check: string;
```

VHDL 制約を次のように指定します。

```
attribute sim_collision_check of {component_name|label_name}: {component|label}
is "sim_collision_check_value";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

```
// synthesis attribute sim_collision_check [of] {module_name|instance_name}
[is] "sim_collision_check_value";
```

UCF および NCF 構文

次の構文では、I/O プリミティブ エレメントの y2 のインスタンシエーションに SIM_COLLISION_CHECK 制約を設定しています。

```
INST "$1187/y2 SIM_COLLISION_CHECK={WARNING_ONLY|GENERATE_X_ONLY|ALL|NONE};
```

SLEW

SLEW 制約には、次のような特徴があります。

- 各出力のスルー レート (遷移レート) ビヘイビアーをデバイスに定義します。
- 出力または双方向ポートに適用して、次のようにポートのスルー レートを指定します。
  - SLOW (デフォルト)
  - FAST
  - QUIETIO (Spartan®-3A デバイスのみ)

一番遅い SLEW 属性を使用しますが、シグナル インテグリティの問題を最小限に抑えるために I/O タイミング要件が満たされるように設定します。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

- 出力プリミティブ
- 出力パッド
- 双方向パッド

UCF ファイル内のパッド コンポーネントに接続されているネットにも設定できます。

ネットに設定した制約は、NGCBuild により NGD ファイル内のパッド インスタンスに挿入され、マップで処理されます。

次の構文を使用してください。

```
NET "net_name" slew={FAST|SLOW};
```

適用ルール

最上位の出力ポートまたは双方向ポートのみに設定します。

制約値

- FAST
- SLOW
- QUIETIO (Spartan-3A デバイスのみ)

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

出力ポートまたは双方向ポートに設定します。

- 属性名

  SLEW

- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

SLEW 制約は、使用前にアーキテクチャ宣言と最上位 VHDL ファイルの begin 文の間で次のように宣言します。

```
attribute SLEW: string;
```

VHDL 制約を次のように指定します。

```
attribute SLEW of {top_level_port_name}: signal is "value";
```

VHDL の構文例

```
entity top is
port (FAST_OUT: out std_logic);
end top;
architecture MY_DESIGN of top is
attribute SLEW: string;
attribute SLEW of FAST_OUT: signal is "FAST";
begin
```

Verilog 構文

最上位の Verilog コードの port 宣言の前に、次のように記述します。

```
(* SLEW="value" *)
```

Verilog の構文例

```
module top (
(* SLEW="FAST" *) output FAST_OUT
);
```

UCF および NCF 構文

出力ポートまたは双方向ポートに設定します。

```
NET "top_level_port_name" SLEW="value";
```

UCF および NCF の構文例

```
NET "FAST_OUT" SLEW="FAST";
```

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PACE の構文

PACE でこの制約を設定するには、[Design Objects] ウィンドウでピンの値を選択します。

PACE では、CPLD デバイスのみがサポートされます。

SLOW

SLOW 制約には、次のような特徴があります。

- 基本的なフィッタ制約です。
- スルー レートの制限付き制御をイネーブルにします。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

- 出力プリミティブ
- 出力パッド
- 双方向パッド

UCF ファイル内のパッド コンポーネントに接続されているネットにも設定できます。

ネットに設定した制約は、NGCBuild により NGD ファイル内のパッド インスタンスに挿入され、マップで処理されます。

これには、次の UCF 構文を使用します。

```
NET "net_name" SLOW;
```

適用ルール

- ネットがパッドに接続されている場合を除き、ネットには設定できません。この場合、SLOW 制約はパッド インスタンスに設定されているものと見なされます。
- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

制約値

- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  SLOW
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute SLOW : string;
```

VHDL 制約を次のように指定します。

```
attribute SLOW of {signal_name|entity_name}: {signal|entity}is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* SLOW = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

- **INST "$1I87/y2" SLOW;**

  エレメント y2 のインスタンスに低速スルー レートを設定しています。

- **NET "net1" SLOW;**

  net1 が接続されているパッドに低速スルー レートを設定しています。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

STEPPING

STEPPING 制約には、デバイス上に記載されたステップ レベルに一致する値が割り当てられます。ステップ レベルよりデバイスの性能が判断できます。

この制約を使用し、デザインにステップ レベルを設定することをお勧めします。設定がない場合は、デフォルトのターゲット デバイスが使用されます。

STEPPING の詳細は、ザイリンクスのアンサー #20947 の「ステッピングに関するよくある質問」を参照してください。

アーキテクチャ サポート

- CoolRunner™-II
- Spartan®-3A
- Spartan-3E
- Virtex®-4
- Virtex-5

適用可能エレメント

STEPPING 制約には、次のような特徴があります。

- グローバル CONFIG 制約です。
- 特定のインスタンスや信号名には設定されません。

適用ルール

デザイン全体に適用

制約値

n

次のようなターゲットのステップ レベルを示します。

- ES
- SCD1
- 1、2、3 ...

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF 構文

```
CONFIG STEPPING="n";
```

```
CONFIG STEPPING="1";
```

SUSPEND

SUSPEND 制約には、次のような特徴があります。

- Spartan®-3A、Spartan-6 デバイスでのみサポートサポートされます。
- FPGA が電力削減モードの SUSPEND に 設定されている場合に、各出力のビヘイビアーを定義します。
- 出力または双方向ポートに適用して、次のようにポートを指定します。
  - トライステート (3STATE)
  - High へのプルアップ (3STATE_PULLUP) または Low へのプルダウン (3STATE_PULLDOWN)
  - 最後の値(3STATE_KEEPER または DRIVE_LAST_VALUE) へ駆動

アーキテクチャ サポート

- Spartan-3A
- Spartan-6

適用可能エレメント

Spartan-3A または Spartan-6 デバイスをターゲット デバイスとする、最上位レベルの出力ポートまたは双方向ポートのみに設定できます。

適用ルール

最上位の出力ポートまたは双方向ポートのみに設定します。

制約値

- DRIVE_LAST_VALUE
- 3STATE (デフォルト)
- 3STATE_PULLUP
- 3STATE_PULLDOWN
- 3STATE_KEEPER

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

出力ポートまたは双方向ポートに設定します。

- 属性名

  SUSPEND

- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

SUSPEND は、使用前にアーキテクチャ宣言と最上位 VHDL ファイルの begin 文の間で次のように宣言します。

```
attribute SUSPEND: string;
```

SUSPEND 制約は宣言後、次のように指定します。

```
attribute SUSPEND of {top_level_port_name} : signal is "value";
```

```
entity top is
port (STATUS: out std_logic);
end top;architecture MY_DESIGN of top is
attribute SUSPEND: string;
attribute SUSPEND of STATUS: signal is "DRIVE_LAST_VALUE";
begin
```

Verilog 構文

最上位の Verilog コードの port 宣言の前に、次のように記述します。

```
(* SUSPEND="value" *)
```

```
module top ( (* SUSPEND="DRIVE_LAST_VALUE" *) output STATUS );
```

UCF および NCF 構文

出力ポートまたは双方向ポートに設定します。

```
NET "top_level_port_name" SUSPEND="value";
```

```
NET "STATUS" SUSPEND="DRIVE_LAST_VALUE";
```

PACE の構文

Pinout and Area Constraints Editor (PACE) でこの制約を設定するには、[Design Objects] ウィンドウでピンの値を選択します。

SYSTEM_JITTER

SYSTEM_JITTER (システム ジッタ) 制約には、次のような特徴があります。

- デザインのシステム ジッタを指定します。
- 次のようなデザイン条件に依存します。
  - 1 度に変更されるフリップフロップ数
  - 変更される I/O 数
- デザインのクロックすべてにグローバルに適用されます。
- 次と併用すると、タイミング レポートでクロック誤差の値が表示されるようになります。
  - PERIOD 制約の INPUT_JITTER キーワード
  - クロック ネットワークのすべてのジッタまたは位相エラー

詳細は、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

デザイン全体に適用

適用ルール

ありません。

制約値

value

- 数値を指定します。
- デフォルトは ns です。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  SYSTEM_JITTER
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute SYSTEM_JITTER: string;
```

VHDL 制約を次のように指定します。

```
attribute SYSTEM_JITTER
of {component_name|signal_name|entity_name|label_name}: {component|signal|entity|label} is "value  ns";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* SYSTEM_JITTER = "value ns" *)
```

UCF および NCF 構文

```
SYSTEM_JITTER= value ns;
```

XCF 構文

```
MODEL "entity_name" SYSTEM_JITTER = value ns;
```

TEMPERATURE

TEMPERATURE (温度) 制約には、次の特徴があります。

- タイミング制約です。
- 動作ジャンクション温度を指定します。
- 指定した温度に基づいてデバイスの遅延特性を比例配分できます。

比例配分は、既存のスピード ファイルの遅延に対して行われ、すべての遅延に対してグローバルに適用されます。新しいデバイスでは、タイミング情報 (スピード ファイル) が Production ステータスになるまで温度の比例配分がサポートされない場合があります。

アーキテクチャにはそれぞれ、有効な動作温度範囲があります。入力した温度がこの範囲外の場合、TEMPERATURE 制約は無視され、そのアーキテクチャのワーストケースの値が使用されます。これに関するエラー メッセージは、スタティック タイミングで出力されます。

アーキテクチャ サポート

- Spartan®-3A
- Spartan-3E
- Virtex®-4
- Virtex-5

適用可能エレメント

デザイン全体にグローバルに適用

適用ルール

この制約はデザイン エレメント制約です。ネットへは接続できません。

構文

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

TEMPERATURE=*value* **[C |F| K];**

説明 :

value

- は温度を指定する実数です。
- C、K、F は、温度の単位です。
  - F : 華氏
  - K : ケルビン
  - C : 摂氏 (デフォルト)

次のように指定すると、スピード ファイル遅延に関連した解析でのジャンクション温度は 25℃ になります。

TEMPERATURE=25 C;

Constraints Editor の設定

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

TIG

TIG (タイミング無視) 制約には、次のような特徴があります。

- タイミング制約および合成制約です。
- TIG を適用したポイント (点) 以降のパスは、インプリメンテーション中、タイミング解析において存在しないものとして処理されます。
- 特定のタイムスペックに関連して設定できます。
- 次のいずれかの値に指定できます。
  - 値なし (すべてのパスに適用されるグローバル TIG)
  - 1 つの TSid からブロック
  - カンマで区切った TSid からブロックのリスト (次の例を参照)
- XST で完全にサポートされます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

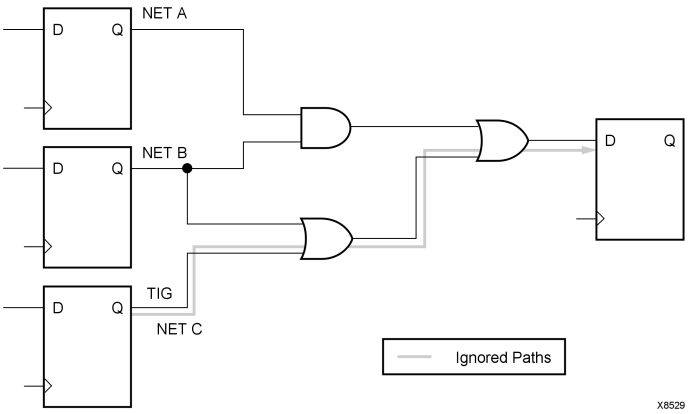
- ネット
- ピン
- インスタンス

適用ルール

TIG をネット、プリミティブ ピン、またはマクロ ピンに設定した場合、インプリメンテーション中、タイミング解析においては、制約が適用されるポイント以降のパスはすべて存在しないものとして処理されます。次の図を参照してください。

- NET C は無視されます。
- 2 つの OR ゲートを通る NET B の下位パスは無視されません。

TIG の例



次の制約をネットに設定すると、タイミング解析において、タイムスペック TS43 でそのネットを通るパスが無視されます。

回路図	UCF 構文
TIG = TS43	**NET** " *net_name*" **TIG = TS43;**

組み合わせループがある場合、パスの解析はできません。そのため、タイミング解析では特定の接続を無視し、組み合わせループを切断します。TIG 制約を使用すると、特定ネットまたはロード ピンを無視するようにタイミング ツールに命令でき、ループの接続を制御できます。

制約値

- identifier

  無視するタイムスペック

- item

  次のいずれかを指定します。

  - PIN *name*
  - PATH *name*
  - *path specification*
  - NET *name*
  - TIMEGRP *name*
  - BEL *name*
  - COMP *name*
  - MACRO *name*

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

注記 : TIG 制約を使用しても、XST レポートの一番下のタイミング レポートには影響ありません。TIG 制約が適用されるのは、Timing Analyzer でレポートされるタイミングのみです。

回路図

- ネットまたはピンに適用します。
- 属性名

  TIG
- 属性値

  value

UCF および NCF 構文

```
NET " net_name" TIG;

PIN "ff_inst.RST" TIG=TS_1;

INST "instance_name " TIG=TS_2;

TIG=TS identifier1 . . . TS identifiern
```

適用先	TIG 制約
instance	そのインスタンスの出力ピンにプッシュされます。
net	そのネットの駆動ピンにプッシュされます。
pin	ピンに適用されます。

次の文は、RESET ネットから順方向に広がるすべてのパスでタイムスペックの TS_*fast* および TS_*even_faster* が無視されるように指定しています。

```
NET "RESET" TIG=TS_fast, TS_even_faster;
```

XCF 構文

UCF と同じ構文を使用します。

TIG は、XST で完全にサポートされています。TIG は、次の CORE Generator ファイルに含まれるネットに適用できます

- Electronic Data Interchange Format (EDIF)
- Native Generic Database (NGD)

Constraints Editor の設定

ISE® Design Suite の Constraints Editor および Constraints Editor の設定に関する詳細は、ISE Design Suite ヘルプを参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PCF 構文

item **TIG;**

item **TIG =;**

item **TIG = TS***identifier* **;**

TIMEGRP

TIMEGRP (タイムグループ) 制約には、次のような特徴があります。

- TNM 識別子を使用し、タイミング解析用にデザイン エレメントをグループ化します。
- 次が実行できます。
  - ほかのグループに相対的にグループを指定できます。
  - 既存グループを組み合わせたグループを作成
  - UCF または NCF で TIMEGRP 制約を設定

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

- デザイン エレメント
- ネット

適用ルール

グループ内のすべてのエレメントまたはネットに適用されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

複数のグループを 1 つのグループにまとめる

複数のグループをまとめて 1 つのグループを定義します。

UCF の構文例 1 (複数グループをまとめる例)

次に、2 つのグループを 1 つにまとめる構文例を示します。

```
TIMEGRP "big_group"="small_group" "medium_group";
```

small_group および *medium_group* は、TNM または TIMEGRP で定義された既存のグループです。

UCF の構文例 2 (複数グループをまとめる例)

次のように定義が循環していると、NGCBuild でデザインを実行する際にエラーが発生します。

```
TIMEGRP "many_ffs"="ffs1" "ffs2";

TIMEGRP "ffs1"="many_ffs" "ffs3";
```

除外によって新しいグループを作成する方法

あるグループから別のグループに属するエレメントを除外することにより、グループを定義できます。次に、その構文例を示します。

UCF の構文例 1 (除外してグループを作成する例)

```
TIMEGRP "group1"="group2" EXCEPT "group3";
```

- group1

  定義されたグループを表しています。これには、*group2* のエレメントすべてが含まれますが、*group3* にも属するエレメントは除外されています。

- group2 および group3

  次のいずれかです。

  - 有効な TNM
  - 定義済みのグループ

    注記 : OFFSET 制約には定義済みグループは使用できません。

  - TIMEGRP 属性

UCF の構文例 2 (除外してグループを作成する例)

次に示すように、新しいグループを作成する際に、含めるグループまたは除外するグループを複数指定できます。

```
TIMEGRP "group1"=" "group2" "group3" EXCEPT "group4" "group5";
```

この例では、*group1* は *group2* および *group3* のエレメントを含みますが、*group4* または *group5* に属するエレメントは除外されます。キーワード EXCEPT の前のグループはすべて含まれ、キーワードの後のグループは除外されます。

クロック エッジによるフリップフロップのサブグループの定義

キーワード RISING および FALLING を使用して、立ち上がりエッジおよび立ち下がりエッジでトリガされるフリップフロップをグループ化し、サブグループを作成できます。

UCF の構文例 1 (クロック エッジでまとめる例)

```
TIMEGRP "group1"=RISING FFS;

TIMEGRP "group2"=RISING "ffs_group";

TIMEGRP "group3"=FALLING FFS;

TIMEGRP "group4"=FALLING "ffs_group";
```

- group1 ～ group4

  新しく定義されるグループです。

- ffs_group

  フリップフロップのみを含むグループである必要があります。

EXCEPT、RISING、FALLING などのキーワードには、大文字/小文字のいずれを使用してもかまいませんが、すべて大文字かすべて小文字にします。ただし、大文字と小文字を組み合わせて入力することはできません。

UCF の構文例 2 (クロック エッジでまとめる例)

次に、クロックの立ち下がりエッジでオンになるフリップフロップのグループを定義します。

```
TIMEGRP "falling_ffs"=FALLING FFS;
```

ゲートによるラッチのサブグループの定義

LATCHES のグループは、どのように定義されていても、透過 High および透過 Low のサブグループに分類できます。これにはキーワード TRANSHI および TRANSLO を使用します。これらのキーワードは、フリップフロップ グループのキーワード RISING および FALLING と同様、TIMEGRP 文で使用します。

UCF の構文例 1 (ゲートでまとめる例)

```
TIMEGRP "lowgroup"=TRANSLO "latchgroup";

TIMEGRP "highgroup"=TRANSHI "latchgroup";
```

文字列の指定によるグループの作成

特定の文字列に一致するネット名を持つシンボルをグループ化できます。文字列の指定には、ワイルドカード文字を使用します。通常、この方法は、ネット名を指定した回路図デザインで使用します。合成には、INST/TNM 構文を使用します。詳細は、「TNM」を参照してください。

ワイルドカードを使用したネット名の指定

ネット名に次のワイルドカード文字を使用すると、出力ネット名が特定の文字列またはパターンに一致するシンボルのグループを選択できます。

- アスタリスク (*)

  任意の数の文字列

- 疑問符 (?)

  任意の 1 文字

次に例を示します。

- DATA*

  次のような DATA で始まるネット名を示します。

  - DATA
  - DATA1
  - DATA22
  - DATABASE

- NUMBER?

  次のような NUMBER で始まり、その後に 1 文字が付いているネット名を示します。

  - NUMBER1
  - NUMBERS
  - 次は認識されません。
    - NUMBER
    - NUMBER12

ワイルドカード文字は、複数使用できます。次に例を示します。

- *AT?

  次のようなネット名を示します。

  - AT の前に何文字か含む
  - AT の後に 1 文字で終わる
    - BAT1
    - CAT2
    - THAT5

- *AT*

  次のようなネット名を示します。

  - AT の前に何文字か含む
  - AT の後に何文字か含む
    - BAT11
    - CAT26
    - THAT50

UCF の構文例 1 (ワイルドカード)

次に、文字列の指定によってグループを作成するための構文を示します。

```
TIMEGRP "group_name"=predefined_group("pattern");
```

- predefined_group

  次の定義済みグループのいずれかになります。

  - FF
  - LATCH
  - PAD
  - RAM
  - HSIOS
  - DSP
  - BRAM_PORTA
  - BRAM_PORTB
  - MULT

  注記 : OFFSET 制約には定義済みグループは使用できません。

  これらのグループの定義の詳細は「TNM_NET」の「UCF および NCF 構文」を参照してください。

  注記 : 該当アーキテクチャで乗算器が使用できない場合、定義済みのタイプ MULTS は使用できません。

- pattern

  1 つまたは複数のワイルドカード文字を使用した文字列です。

ネット名を指定する場合、PAR がフラット化されたデザインでネットを検索できるように、完全な階層パス名を使用する必要があります。

- 次の場合、出力ネット名を指定します。
  - FF
  - RAM
  - LATCH
  - PAD
  - CPU
  - DSP
  - HSIOS
  - MULT
- パッドの場合は、外部ネット名を指定します。

UCF の構文例 2 (ワイルドカード)

次の例では、名前が $1I3/FRED で始まるネットを持つフリップフロップを含むグループを作成します。

```
TIMEGRP "group1"=FFS("$1I3/FRED*");
```

UCF の構文例 3 (ワイルドカード)

次の例では、出力ネット名が特定の文字列に一致するフリップフロップを除くグループを作成します。

```
TIMEGRP "this_group"=FFS EXCEPT FFS("a*");
```

this_group には、名前が a で始まる出力ネットを持つフリップフロップを除く、すべてのフリップフロップが含まれます。

UCF の構文例 4 (ワイルドカード)

次の例では、some_latches という名前のグループを定義します。

```
TIMEGRP "some_latches"=latches("$113/xyz*");
```

some_latches グループには、名前が「$1I3/xyz」で始まる出力ネットを持つ入力ラッチがすべて含まれます。

その他のパターン一致情報

文字列の指定は、タイミング グループ作成時だけではなく、定義済みグループを指定する場所であればどこででも使用できます。次の構文では、タイムスペックで文字列を指定する方法を示します。

UCF の構文例 1 (パターン一致)

```
TIMESPEC "TSidentifier"=FROM predefined_group("pattern") TO predefined_group("pattern") value;
```

UCF の構文例 2 (パターン一致)

文字列を 1 つ指定する代わりに、コロンで区切った文字列のリストを指定できます。

```
TIMEGRP "some_ffs"=FFS("a*":b?":c*d");
```

グループ *some_ffs* には、名前が次のいずれかのルールに一致する出力ネットを持つフリップフロップが含まれます。

パターン	説明
a*	a で始まる任意の文字数
b?	最初の文字が「b」で 2 文字
c*d	最初の文字が「c」で、最後の文字が「d」

タイムグループを使用したエリア グループの定義

詳細は、「AREA_GROUP」の「タイミング グループによる定義」を参照してください。

UCF の構文例 1 (タイムグループ)

```
TIMEGRP "newgroup"="existing_grp1" "existimg_grp2" ["existing_grp3" ...];
```

newgroup は、新しく作成するグループです。このグループは、次のいずれかです。

- TNM で作成された既存のグループ
- 定義済みグループ

  注記 : OFFSET 制約には定義済みグループは使用できません。
- ほかの TIMEGRP 属性

UCF の構文例 2 (タイムグループ)

```
TIMEGRP "GROUP1" = "gr2" "GROUP3";
TIMEGRP "GROUP3" = FFS except "grp5";
```

XCF 構文

XST での TIMEGRP の使用には、次のような制限があります。

- 除外によるグループの作成はサポートされていません。
- 文字列の一致を使用して別のユーザー グループを基にグループを定義する場合
  - **TIMEGRP TG1 = FFS (machine*);**

    サポートあり
  - **TIMEGRP TG2 = TG1 (machine_clk1*);**

    サポートなし

Constraints Editor の構文

Constraints Editor での Constraints Editor およびその構文に関する詳細は、ISE Design Suite ヘルプを参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

PCF 構文

```
TIMEGRP name;
TIMEGRP name = list of elements;
```

TIMESPEC

TIMESPEC (タイミング仕様) 制約には、次のような特徴があります。

- 基本的なタイミングに関連する制約です。
- TS 属性で定義するタイムスペックのプレースホルダの役割を果たします。

TS 属性

TS 属性には、次の特徴があります。

- TS という文字で始まります。
- 次を含むことのできる独自の識別子で終了します。
  - 文字
  - 数値
  - アンダースコア (_)

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

TS 識別子に適用します。

適用ルール

なし

制約の構文

value は、属性の最大遅延を定義します。デフォルトの単位はナノ秒 (ns) ですが、ピコ秒やメガヘルツなど、ほかの単位も使用できます。

このマニュアルでは FROM、TO、TS などのキーワードは大文字で表記されていますが、TIMESPEC には、大文字または小文字のどちらでも使用できます。TIMESPEC には、大文字または小文字のどちらでも使用できます。ただし、キーワードの文字は大文字または小文字で統一する必要があります。

次の例は、キーワードとして使用できます。

- FROM
- PERIOD
- TO
- from
- to

次の例は、キーワードとして使用できません。

- From
- TO
- fRoM
- tO

TSidentifier 名

TSidentifier 名がプロパティ値で参照されている場合は、大文字で入力する必要があります。たとえば次の例で示すように、2 番目の制約名 TSID1 は、1 番目の制約名 TSID1 と一致させるため大文字で入力する必要があります。

```
TIMESPEC "TSID1" = FROM "gr1" TO "gr2" 50;

TIMESPEC "TSMAIN" = FROM "here" TO "there" TSID1 /2;
```

区切り文字

タイムスペックでは、区切り文字として、スペースの代わりにコロンを使用できます。

FROM-TO 構文

次の UCF 構文を使用し、特定の終了地点間のタイムスペックを指定します。

```
TIMESPEC "TSidentifier"=FROM "source_group" TO
"dest_group" value units;

TIMESPEC "TSidentifier"=FROM "source_group" value units;

TIMESPEC "TSidentifier"=TO "dest_group" value units;
```

2 番目と 3 番目の構文のように、**FROM** または **TO** を指定しない場合は、すべての点であることを示します。

注記 : FROM または TO を指定せずにすべての点を示すことはできますが、THRU を指定せずにすべての点を示すことはできません。

FROM TO 文は、TIMESPEC プリミティブ内に存在する TS 属性です。パラメーター *source_group* および *dest_group* は、次のいずれかです。

- 定義済みグループ

  注記 : OFFSET 制約には定義済みグループは使用できません。

- 作成済みの TNM 識別子
- TIMEGRP シンボルで定義されたグループ
- TPSYNC グループ

定義済みのグループには、次が含まれます。

- FFS
- LATCHES
- RAMS
- PADS
- CPUS
- DSPS
- HSIOS
- BRAMS_PORTA
- BRAM_PORTB
- MULTS

これらのグループの定義については TNM_NET 制約の「UCF および NCF 構文」セクションを参照してください。

このマニュアルでは FROM、TO、TS などのキーワードは大文字で表記されていますが、TIMESPEC には、大文字または小文字のどちらでも使用できます。TIMESPEC には、大文字または小文字のどちらでも使用できます。ただし、大文字と小文字を組み合わせて入力することはできません。

value は、属性の最大遅延を定義します。TS 属性の遅延時間を指定するデフォルトの単位は、ナノ秒です。ピコ秒やメガヘルツなど、ほかの単位も使用できます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

FROM TO を使用した TIMESPEC の構文例

```
TIMESPEC "TS_master"=PERIOD "master_clk" 50 HIGH 30;
TIMESPEC "TS_THIS"=FROM FFS TO RAMS 35;
TIMESPEC "TS_THAT"=FROM PADS TO LATCHES 35;
```

UCF 構文例

TS 属性は、デザインのパスに対して許容される遅延を定義します。次に、TS 属性の基本的な構文を示します。

```
TIMESPEC "TSidentifier"=PERIOD "timegroup_name" value [units];
```

- TS*identifier*

  TS 属性の名前

- value

  数値になります。

- units

  ms、micro、ps、ns

```
TIMESPEC "TSidentifier"=PERIOD "timegroup_name" "TSidentifier" [* or /]
factor PHASE [+ |-] phase_value [units];
```

Constraints Editor の構文

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

TNM

タイミング名 (TNM) 制約には、次の特徴があります。

- 基本的なグループ制約です。
- タイミング仕様で使用されるグループを構成するエレメントを指定します。
- 次のような特定のエレメントをグループの要素として指定すると、そのグループに対するタイミング仕様の適用を簡略化できます。
  - FF
  - RAM
  - LATCH
  - PAD
  - CPU
  - HSIOS
  - MULT
- この制約は、キーワード **RISING** および **FALLING** と一緒に使用できます。

TNM とパーティション

- パーティションに関連する PAD 名に基づいた TNM は、サポートされません。
- パーティション内のネット名に基づいた TNM は、サポートされます。

TNM および TNM_NET

- TNM は同じネットに駆動されるフリップフロップ、ラッチ、RAM またはパッドのネットグループに付けます。
- TNM は IBUF または BUFG コンポーネントを通らず、入力パッドまで到達します。
- TNM_NET 制約は IBUF およびグローバル クロック バッファーを通ります。
- ザイリンクスでは、次を推奨しています。
  - インスタンスおよびマクロ (階層ブロック) をグループにするには TNM を使用します。
  - 入力パッドをグループにするには、パッドに駆動されるネットに TNM を使用します。
  - 同じネットに駆動されるクロック、クロック イネーブル、チップ イネーブル、読み出し/書き込み、リセットなどの複数のロジック エレメントをグループにするには、TNM_NET を使用します。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

- ネット、エレメント ピン、プリミティブ、またはマクロに設定できます。
- TNM 制約は、UCF ファイル内で、パッド コンポーネントに接続されているネットにも設定できます。ネットに設定した制約は、NGCBuild により NGD ファイル内のパッド インスタンスに挿入され、マップで処理されます。これには、次の UCF 構文を使用します。

```
NET "net_name" TNM="property_value";
```

適用ルール

- ネットまたは信号に TNM を設定すると、そのネットにより駆動されるすべての同期エレメントおよび PADS に伝搬されます。特別な伝搬はありません。
- デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。
- TNM をパッド ネットに適用すると、IBUF を介して順方向には伝搬されず、外部パッドに適用されます。これには、IFD の D 入力に接続されているネットも含まれます。TNM を入力パッド ネットから順方向に伝搬させる方法については、「TNM_NET」を参照してください。
- IBUF インスタンスには設定できません。
- IBUF の出力ピンに設定すると、その次の適切なエレメントに適用されます。
- IBUF エレメントに適用した場合、TNM はそのエレメントに設定されます。
- クロック パッド ネットに設定すると、クロック バッファー以降には適用されません。
- マクロに設定すると、マクロ内のすべてのエレメントにそのタイミング名が適用されます。
- TNM は入力パッド ネットに適用すると、IBUF コンポーネントを介して伝搬されません。

ネットに設定する場合

TNM 制約は、デザイン内のどのネットにも設定でき、指定したネット以降のパスから信号が供給される有効なエレメントすべてに TNM 値が付加されます。

この制約は、次のエレメントまで適用されます。

- FF
- RAM
- LATCH
- PAD
- CPU
- HSIOS
- MULT

マクロまたはプリミティブ ピンに設定する場合

デザイン入力パッケージでプリミティブ ピンに制約を設定できる場合、コンポーネント ピンに TNM 制約を設定できます。指定されたピン以降のパスから信号が供給される有効なエレメントすべてに TNM 値が付加されます。

この制約は、次のエレメントまで適用されます。

- FF
- RAM
- LATCH
- PAD
- CPU
- HSIOS
- MULT

UCF 構文は次のとおりです。

```
PIN "pin_name" TNM="FLOPS";
```

プリミティブ シンボルに設定する場合

各インスタンスに制約を設定すると、ロジック プリミティブをグループ化できます。

TNM が設定されたフリップフロップからグループ「FLOPS」が作成されます。設定されていないフリップフロップは、このグループには含まれません。UCF の構文例を参照してください。

TNM は、各シンボル、ドライバー ピン、またはマクロ ドライバー ピンに 1 つずつしか設定できません。

UCF 構文

```
INST "instance_name" TNM=FLOPS;
```

ネットまたはピンに設定してフリップフロップおよびラッチをグループ化する場合

クロック ネットやイネーブル ネットなど共通の入力ネットに TNM を設定すると、フリップフロップ、ラッチを簡単にグループ化できます。TNM をネットまたはドライバー ピンに設定した場合、TNM はそのネットまたはピン以降のパス上にあるすべてのフリップフロップおよび入力ラッチに適用されます。つまり、TNM 制約は、パス上にあるゲートまたはバッファーを介して、フリップフロップまたは入力ラッチに到達するまで順方向に伝搬され、そのフリップフロップまたはラッチが指定された TNM グループに追加されます。

ネットまたはピンに設定する TNM パラメーターには、修飾子を含めることができます。たとえば、UCF ファイルは次のように記述します。

```
{NET|PIN} "net_or_pin_name" TNM=FFS data;
{NET|PIN} "net_or_pin_name" TNM=RAMS fifo;
{NET|PIN} "net_or_pin_name" TNM=RAMS capture;
```

修飾子が付いた TNM は、次のような最初の記憶エレメントまで伝搬されます。

- FF
- RAM
- LATCH
- PAD
- CPU
- HSIOS
- MULT

記憶エレメントの種類が修飾子と一致すると、そのエレメントに TNM 値が適用されます。一致の有無に関係なく、TNM が記憶エレメントを介して伝搬されることはありません。

ネットまたはピンに設定された TNM 制約は、次の記憶エレメント以降には適用されません。

- FF
- RAM
- LATCH
- PAD
- CPU
- HSIOS
- MULT

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

{**NET**|**INST**|**PIN**} "*net_or_pin_or_inst_name*" **TNM=** [*predefined_group*] *identifier*;

- predefined_group
  - 次のキーワードを使用した定義済みグループのすべての要素にできます。
    - FF

      ファンクション ジェネレーターから作成されたフリップフロップを除くすべての CLB および IOB フリップフロップ
    - RAM

      LUT RAM およびブロック RAM を含むすべての RAM コンポーネント
    - PAD

      すべての I/O パッド
    - LATCH

      ファンクション ジェネレーターから作成されたラッチを除くすべての CLB および IOB ラッチ
    - Spartan®-3、Spartan-3A、Spartan-3E のレジスタ付き乗算器のグループ。
  - *predefined_group* のエレメントのサブセットにできます。

predefined_group (name_qualifier1...name_qualifiern)

name_qualifiern は、英数字、アンダースコアを自由に組み合わせて指定できます。*name_qualifier* のタイプ (ネットまたはインスタンス) は、TNM が配置されているエレメントのタイプによって決まります。TNM が NET に設定される場合、*name_qualifier* はネット名になり、インスタンス (INST) に設定される場合、インスタンス名になります。

例

```
NET clk TNM = FFS (my_flop) Grp1;INST clk TNM =
FFS (my_macro) Grp2;
```

- identifier
  - 英数字、アンダースコアを自由に組み合わせて指定できます。
  - identifier には、次の予約語は使用できません。
    - FF
    - RAM
    - LATCH
    - PAD
    - CPU
    - HSIOS
    - MULT
    - RISING
    - FALLING
    - TRANSHI
    - TRANSLO
    - EXCEPT

また、次にリストされている予約語も *identifier* には使用できません。

予約語 (制約)

ADD	ALU	ASSIGN
BEL	BLKNM	CAP
CLKDV_DIVIDE	CLBNM	CMOS
CYMODE	DECODE	DEF
DIVIDE1_BY	DIVIDE2_BY	DOUBLE
DRIVE	DUTY_CYCLE_CORRECTION	FAST
FBKINV	FILE	F_SET
HBLKNM	HU_SET	H_SET
INIT	INIT_OX	INTERNAL
IOB	IOSTANDARD	LIBVER
LOC	LOWPWR	MAP

MEDFAST	MEDSLOW	MINIM
NODELAY	OPT	OSC
RES	RLOC	RLOC_ORIGIN
RLOC_RANGE	SCHNM	SLOW
STARTUP_WAIT	SYSTEM	TNM
TRIM	TS	TTL
TYPE	USE_RLOC	U_SET

デザインのパフォーマンス要件を記述するのに必要なだけ、終端地点のグループを指定できます。次の目的のため、グループの数を最小限に抑えるようにしてください。

- 指定プロセスを単純化
- 配置配線時間を削減

XCF 構文

「UCF およびNCF 構文」を参照してください。

Constraints Editor の構文

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

PlanAhead の設定

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

TNM_NET

TNM_NET (タイミング名ネット) 制約には、次のような特徴があります。

- 基本的なグループ制約です。
- タイミング仕様で使用されるグループを構成するエレメントを指定します。
- 入力パッド ネットに設定する場合を除き、ネットに設定した TNM と基本的に同等です。
- TNM_NET で指定されたダウンストリームの同期エレメントおよびパッドは、すべてグループと見なされます。
- 特定の同期エレメント、パッド、ラッチをグループ化することにより、タイミング仕様の適用を簡略化するために使用できます。

  TNM 制約とは異なり、NGCBuild で、制約が設定されたネットから入力パッドに TNM_NET が伝搬されることはありません。

DLL、DCM、PLL、MMCM コンポーネント

次のコンポーネントでは、TNM に PERIOD 制約を付けて使用する場合、特別なルールが適用されます。

- DLL
- DCM
- PLL
- MMCM

詳細は、『タイミング クロージャー ユーザー ガイド』(UG612) を参照してください。

TNM および TNM_NET

TNM は同じネットに駆動されるフリップフロップ、ラッチ、RAM またはパッドのネット グループに付けます。

TNM は IBUF または BUFG コンポーネントを通らず、入力パッドまで到達します。

TNM_NET 制約は IBUF およびグローバル クロック バッファーを通ります。

ザイリンクスでは、次を推奨しています。

- インスタンスおよびマクロ (階層ブロック) をグループにするには TNM を使用します。
- 入力パッドをグループにするには、パッドに駆動されるネットに TNM を使用します。
- 同じネットに駆動されるクロック、クロック イネーブル、チップ イネーブル、読み出し/書き込み、リセットなどの複数のロジック エレメントをグループにするには、TNM_NET を使用します。

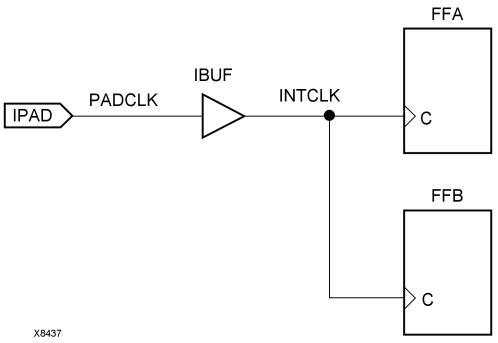
TNM_NET のルール

- パッド ネットに設定すると、IBUF、OBUF、または組み合わせロジックを介して同期ロジックまたはパッドに伝搬されます。
- クロック パッド ネットに設定すると、クロック バッファーを介して伝搬されます。
- Virtex®-4 および Virtex-5 の DLL、DCM、および PLL で、PERIOD 制約と共に TNM_NET を使用する場合は、特別なルールが適用されます。

TNM 制約では適切に記述できないタイプのネットを定義する場合は TNM_NET を使用します。

たとえば、次のようなデザインがあるとします。

IPAD に設定した TNM



このデザインで、IPAD シンボルに TNM を設定すると、タイミング解析グループには PAD シンボルのみが含まれます。たとえば、次の UCF 構文では、IPAD シンボルのみを含むタイムグループが作成されます。

```
NET "PADCLK" TNM= "PADGRP";
```

一方、PADCLK ネットのタイムグループを定義するために TNM を使用すると、空のタイムグループが作成されます。

```
NET "PADCLK" TNM=FFS "FFGRP";
```

パッドに適用されるプロパティはすべて、ネットから PAD シンボルに転送されます。TNM はネットから PAD シンボルに転送されるので、修飾子「FFS」は PAD シンボルと一致しません。

回路図デザインで TNM を使用する際にこの問題を回避するには、INTCLK ネットのタイムグループを作成します。

```
NET "INTCLK" TNM=FFS FFGRP;
```

HDL デザインの場合は、意味を持つネット名のみがパッドに直接接続されます。この場合は、TNM_NET を使用して FFGRP タイムグループを作成します。

```
NET PADCLK TNM_NET=FFS FFGRP;
```

TNM の場合とは異なり、NGDBuild は TNM_NET 制約をネットから IPAD に転送しません。

TNM_NET は、入力ネットリスト (EDIF または NGC) 内のネットに設定されたプロパティとして、NCF または UCF ファイル内で使用できます。TNM_NET は、PCF ファイルではサポートされていません。

TNM_NET は、ネットまたはインスタンスに使用できます。ピンやシンボルなどほかのオブジェクトに使用した場合は、警告が表示され、TNM_NET 定義は無視されます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

ネットに適用されます。

適用ルール

デザイン エレメントには設定できません。

制約値

- predefined_group

  注記 : OFFSET 制約には定義済みグループは使用できません。

  - 定義済みグループのすべての要素には、次のキーワードが使用されます。
    - FF

      すべての CLB および IOB のフリップフロップ。ファンクション ジェネレータから作成されたフリップフロップを除きます。
    - LATCH

      すべての CLB または IOB のラッチ。ファンクション ジェネレータから作成されたラッチは除きます。
    - PAD

      すべての I/O パッド
    - RAM

      LUT RAM およびブロック RAM を含むすべての RAM コンポーネント
    - HSIOS
    - DSP

      Virtex-4 DSP48 のような DSP エレメントをグループにする DSP コンポーネント
    - BRAM_PORTA
    - BRAM_PORTB
    - MULT

      Spartan®-3、Spartan-3A、Spartan-3E のレジスタ付き乗算器のグループ
  - *predefined_group* のエレメントのサブセットを指定するには、次の構文を使用します。

    predefined_group (*name_qualifier1*...*name_qualifiern*)

    name_qualifiern は、英数字、アンダースコアを自由に組み合わせて指定できます。*name_qualifier* のタイプ (ネットまたはインスタンス) は、TNM が配置されているエレメントのタイプによって決まります。TNM_NET が NET に設定される場合、name_qualifier はネット名になり、TNM_NET が NET に設定される場合、*name_qualifier* はネット名になり、インスタンス (INST) に設定される場合、インスタンス名になります。

    例

    ```
    NET clk TNM_NET = FFS (my_flop) Grp1;
    INST clk TNM_NET = FFS (my_macro) Grp2;
    ```

- identifier
  - 英数字、アンダースコアを自由に組み合わせて指定できます。
  - identifier には、次の予約語は使用できません。FF、RAM、LATCHE、PAD、CPU、HSIOS、MULTS、RISING、FALLING、TRANSHI、TRANSLO、または EXCEPT
  - 「TNM」に表示されている予約語も使用できません。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネットに設定します。
- 属性名

  TNM_NET
- 属性値

  identifier

  上記の「制約値」を参照してください。

UCF および NCF 構文

{**NET**|**INST**} "*net_name*" **TNM_NET=**[*predefined_group*:]*identifier*;

次の文は、PADCLK ネット以降のパス上にあるすべてのフリップフロップをタイミング グループ GRP1 として指定します。

```
NET "PADCLK" TNM_NET=FFS "GRP1";
```

XCF 構文

XST で TIME_NET を使用する場合、あらかじめ定義されているグループに対して 1 つのパターンしかサポートされません。

注記 : OFFSET 制約には定義済みグループは使用できません。

- XST では、次のコマンド ライン構文がサポートされます。

  ```
  NET "PADCLK" TNM_NET=FFS "GRP1";
  ```
- XST では、次のコマンド ライン構文はサポートされません。

  ```
  NET "PADCLK" TNM_NET = FFS(machine/*:xcounter/*) TG1;
  ```

Constraints Editor の設定

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

PlanAhead™ の構文

PlanAhead™ ソフトウェアを使用して制約を作成する方法については、『PlanAhead ユーザー ガイド』(UG632) の「デザインのフロアプラン」を参照してください。このマニュアルの「PlanAhead」では、次について説明しています。

- 配置制約の定義
- 配置制約の割り当て
- I/O ピン コンフィギュレーションの定義
- フロアプランおよび配置制約

TPSYNC

TPSYNC (タイミング ポイント同期) 制約には、次のような特徴があります。

- グループ制約です。
- 特定のポイントまたはポイントの集合に識別子を指定し、タイミング仕様で使用できるようにします。

  同じ識別名を複数のポイントに指定すると、それらのポイントはタイミング解析でグループとして扱われます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

- ネット
- インスタンス
- ピン

適用ルール

同期エレメントまたは I/O パッド以外のポイントに関してデザインのタイミングを指定する場合、TPSYNC タイミング ポイントを次のいずれに設定するかで、それぞれの規則が適用されます。

- ネット

  ネットのソースがタイミング仕様のソースまたはデスティネーションとして指定されます。
- マクロ ピン

  制約が設定されたピンを駆動するマクロ内のすべてのソースが、タイミング仕様のソースまたはデスティネーションとして指定されます。マクロ ピンが入力ピンの場合 (ピンのソースがマクロ内にない場合) は、マクロ内のすべてのロード ピンが同期点として指定されます。
- プリミティブの出力ピン

  出力がタイミング仕様の潜在的なソースまたはデスティネーションとして指定されます。
- プリミティブの入力ピン

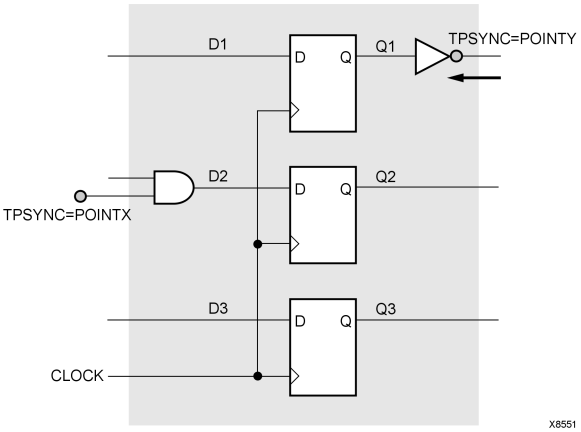
  プリミティブの入力がタイミング仕様のソースまたはデスティネーションとして指定されます。
- インスタンス

  そのエレメントの出力がタイミング仕様のソースまたはデスティネーションとして指定されます。
- プリミティブ シンボル

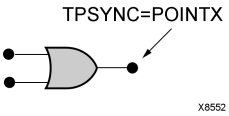
  設定されたエレメントの出力が、タイミング仕様のソースまたはデスティネーションとして指定されます。詳細は、次の図を参照してください。

マクロ ピンに設定した TPSYNC

POINTY がインバータに適用されます。

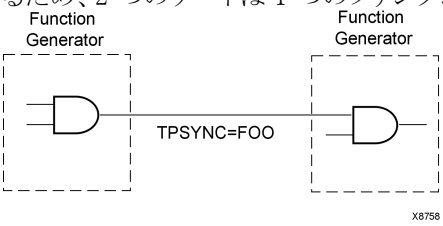


プリミティブシンボルに指定した場合



2 つのゲートに指定した場合

デザインでの同期点の定義に TPSYNC タイミング ポイントを使用すると、指定したポイントをファンクション ジェネレーターに結合できません。次の図の例では、TPSYNC が定義されているため、2 つのゲートは 1 つのファンクション ジェネレータにインプリメントされません。



制約値

identifier

グループと同様にタイミング仕様で使用する名前です。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネット、インスタンス、またはピンに設定します。
- 属性名

  TPSYNC
- 属性値

  上記の「制約値」を参照してください。

UCF および NCF 構文

NET "*net_name*" **TPSYNC=***identifier*;

INST "*instance_name*" **TPSYNC=***identifier*;

PIN "*pin_name*" **TPSYNC=***identifier*;

指定されたすべてのポイントは、TPSYNC 識別名が使用される仕様のソースまたはデスティネーションのいずれか、あるいは両方に使用されます。

identifier には、ほかの TNM または TNM_NET の identifier とは異なる名前を指定する必要があります。

次の文は、ネット logic_latch のタイミング仕様のソースまたはデスティネーションとしてラッチを指定します。

```
NET "logic_latch" TPSYNC=latch;
```

Constraints Editor の構文

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

TPTHRU

TPTHRU (タイミング スルー ポイント) 制約には、次のような特徴があります。

- グループ制約です。
- 特定のポイントまたはポイントの集合に識別子を指定し、タイミング仕様で使用できるようにします。
  - 同じ識別名を複数のポイントに指定すると、それらのポイントはタイミング解析でグループとして扱われます。
  - 詳細は、「TIMESPEC」を参照してください。
- 仕様を適用するパス上に中間点を定義します。

  詳細は、「TSidentifier」を参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

- ネット
- ピン
- インスタンス

適用ルール

なし

制約値

- identifier

  次の文字を含む ASCII 文字列 :

  - A ～ Z
  - a ～ z
  - 0 ～ 9
  - アンダースコア (_)

- source_group および dest_group

  - ユーザー定義のグループ

    または

  - 定義済みのグループ

    注記 : OFFSET 制約には定義済みグループは使用できません。

    または

  - TPSYNC

- thru_point

  パスを指定するために使用する中間点で、TPTHRU 制約で定義します。

- allowable_delay

  タイミング要件です。

- units

  オプションで許容遅延の単位を指定できます。

  - デフォルトの単位はナノ秒です。
  - 単位には、次のいずれかを使用することもできます。
    - ps
    - ns
    - micro
    - ms
    - GHz
    - MHz
    - KHz

identifier 名は、TNM 制約に使用される identifier とは重複しないようにする必要があります。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- ネット、インスタンス、またはピンに適用します。
- 属性名

  TPTHRU
- 属性値

  identifier

  詳細は、次の「UCF および NCF 構文」を参照してください。

UCF および NCF 構文

```
NET "net_name" TPTHRU=identifier;

INST "instance_name" TPTHRU=identifier;

PIN "instance_name.pin_name" TPTHRU="thru_group_name";
```

FROM-TO 制約との併用

仕様を適用するパス上に中間点を定義します。

- 最大許容遅延を定義します。
- 次のセクションに構文を示します。

TIMESPEC を併用した UCF 構文

```
TIMESPEC "TSidentifier"=FROM "source_group" THRU
"thru_point" [THRU "thru_point"] TO
"dest_group" allowable_delay [units];

TIMESPEC "TSidentifier"=FROM "source_group" THRU
"thru_point" [THRU "thru_point"] allowable_delay [units];
```

次の例は、回路図で TPTHRU 制約と THRU 制約を併用する方法を示しています。

UCF の構文は次のとおりです。

```
INST "FLOPA" TNM="A";

INST "FLOPB" TNM="B";

NET "MYNET" TPTHRU="ABC";

TIMESPEC "TSpath1"=FROM "A" THRU "ABC" TO "B" 30;
```

- **`NET "on_the_way" TPTHRU="here";`**

  ネット on_the_way を、タイミング仕様 here が適用されるパス上の中間点として指定しています。
- **`TIMESPECT "TS_1"=THRU "Thru_grp" 30.0`**

  NCF 構文はサポートされません。

Constraints Editor の設定

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

PCF 構文

```
PATH "name"=FROM "source" THRU "thru_pt1" THRU "thru_ptn" TO
"destination";
```

FROM、THRU、TO すべてを必ずしも指定する必要はありません。

次のように、ほとんどすべての組み合わせを使用することができます。

- FROM-TO
- FROM-THRU-TO
- THRU-TO
- TO
- FROM
- FROM-THRU-THRU-THRU-TO
- FROM-THRU

THRU ポイントの数に制限はありません。

ソース、スルーポイント、デスティネーションは、次のいずれかになります。

- ネット
- BEL
- コンポーネント
- マクロ
- ピン
- タイムグループ

TSidentifier

TS*identifier* (タイミング仕様識別子) は、基本的なタイミング制約です。

- この「TS」という文字で始まる TS*identifier* プロパティは、UCF で TIMESPEC と共に使用します。
- TS*identifier* の値は特定のタイミング仕様に対応し、パスに適用できます。

アーキテクチャ サポート

すべての FPGA および CPLD デバイスに適用されます。

適用可能エレメント

TIMESPEC キーワードに適用します。

適用ルール

ネット、信号、デザイン エレメントには設定できません。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

次の構文では、区切り文字としてスペースを使用していますが、コロン (:) も使用できます。

最大許容遅延の定義

```
TIMESPEC "TSidentifier"=FROM "source_group" TO "dest_group" allowable_delay [units];
```

中間点の定義 (UCF)

```
TIMESPEC "TSidentifier"=FROM "source_group" THRU "thru_point" [THRU "thru_point1"...
"thru_pointn"] TO "dest_group" allowable_delay [units];
```

- identifier

  次の文字を含む ASCII 文字列 :

  - A ～ Z
  - a ～ z
  - 0 ～ 9
  - アンダースコア (_)

- source_group および dest_group

  ユーザー定義のグループまたは定義済みのグループ

  注記 : OFFSET 制約には定義済みグループは使用できません。

- thru_point

  パスを指定するために使用する中間点で、TPTHRU 制約で定義します。

- allowable_delay

  タイミング要件です。

- units

  - オプションで許容遅延の単位を指定できます。
  - デフォルトの単位はナノ秒です。
  - ps、ns、micro、ms、GHz、MHz、または kHz のいずれかを指定できます。

仕様の結合の定義

ある仕様で使用されているタイミングの値を別の仕様に結合できます。

```
TIMESPEC "TSidentifier"=FROM "source_group" TO "dest_group" another_TSid [/|*] number;
```

- identifier

  次の文字を含む ASCII 文字列 :

  - A ～ Z
  - a ～ z
  - 0 ～ 9
  - アンダースコア (_)

- source_group および dest_group

  ユーザー定義のグループまたは定義済みのグループ。

  注記 : OFFSET 制約には定義済みグループは使用できません。

- another_Tsid

  別のタイムスペックの名前

- number

  浮動小数点数

クロック周期の定義

単純なクロック周期だけでなく、より複雑な派生関係も定義できます。

```
TIMESPEC "TSidentifier"=PERIOD "TNM_reference" value [units] [{HIGH | LOW}
[high_or_low_time [hi_lo_units]]] INPUT_JITTER value;
```

- identifier

  参照識別名

- TNM_reference

  TNM 制約を使用してクロック ネット (またはクロック パスのネット) に設定した識別名です。

- value

  必要なクロック周期です。

- units
  - オプションで許容遅延の単位を指定できます。
  - デフォルトは ns です。
  - ps、ns、micro、ms、GHz、MHz、または kHz のいずれかを指定できます。
- HIGH または LOW

  オプションの HIGH または LOW を使用すると、最初のパルスを High または Low に指定できます。

- high_or_low_time
  - High または Low になっている時間を指定します (オプション)。High か Low かは、この前のキーワードによって指定します。
  - 実際の時間を指定する場合は、周期より小さい値にする必要があります。
  - high_or_low_time を指定しない場合、デフォルトのデューティ サイクルは 50% です。
- hi_lo_units
  - デューティ サイクルの単位を指定します (オプション)。
  - デフォルトは ns です。
  - High または Low の時間が実際の時間である場合は、ps、micro、ms、ns、% のいずれかを指定できます。

派生クロックの指定

```
TIMESPEC "TSidentifier"=PERIOD "TNM_reference" "another_PERIOD_identifier" [/|*] number [{HIGH|LOW}
[high_or_low_time [hi_lo_units]]] INPUT_JITTER value;
```

- TNM_reference

  TNM 制約を使用してクロック ネット (またはクロック パスのネット) に設定した識別名です。

- another_PERIOD_identifier

  別の周期の仕様で使用されている識別名

- number

  浮動小数点数

- HIGH または LOW

  オプションの HIGH または LOW を使用すると、最初のパルスを High または Low に指定できます。

- high_or_low_time

  High または Low になっている時間を指定します (オプション)。High か Low かは、この前のキーワードによって指定します。実際の時間を指定する場合は、周期より小さい値にする必要があります。high_or_low_time を指定しない場合、デフォルトのデューティ サイクルは 50% です。

- hi_lo_units
  - デューティ サイクルの単位を指定します (オプション)。
  - デフォルトは ns です。
  - High または Low の時間が実際の時間である場合は、ps、micro、ms、% のいずれかを指定できます。

パスの無視

注記 : この形式は、CPLD デバイスではサポートされません。

ネット、インスタンス、インスタンス ピンを通るすべてのパスが、タイミング仕様の観点からは重要でない場合、特定のネットを通るパスを無視するよう指定できます。

```
TIMESPEC "TSidentifier"=FROM "source_group" TO "dest_group" TIG;
```

または

```
TIMESPEC "TSidentifier"=FROM "source_group" THRU "thru_point" [THRU "thru_point1"...
"thru_pointn"] TO "dest_group" TIG;
```

- identifier

  次の文字を含む ASCII 文字列 :
  - A ～ Z
  - a ～ z
  - 0 ～ 9
  - アンダースコア (_)

- source_group および dest_group

  ユーザー定義のグループまたは定義済みのグループ。

  注記 : OFFSET 制約には定義済みグループは使用できません。

- thru_point

  パスを指定するために使用する中間点で、TPTHRU 制約で定義します。

パスの無視の例

- **TIMESPEC "TS_35"=FROM "here" TO "there" 50;**

  タイミング仕様 TS_35 で、グループ here と there の間の最大許容遅延が 50ns に指定しています。

- **TIMESPEC "TS_70"=PERIOD "clock_a" 25 high 15;**

  タイミング仕様 TS_70 で、clock_a のクロック周期が 25ns で、最初のパルスが High、その継続時間が 15ns になるよう指定しています。

詳細は、次を参照してください。

- 論理制約
- 物理制約

Constraints Editor の設定

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

Item	入力方法
クロック周期制約	クロック ドメイン エントリー
入力セットアップ タイム	入力エントリー
clock-to-output 遅延	出力エントリー
pad-to-pad 遅延	[Exceptions] → [Paths] から入力します。

PCF 構文

UCF 構文と同じですが、TIMESPEC キーワードは使用しません。

FPGA Editor の設定

構文も含めた FPGA Editor での制約設定に関する詳細は、FPGA Editor ヘルプを参照してください。

U_SET

U_SET 制約には、次のような特徴があります。

- 高度なマップ制約です。
- デザイン階層全体に分散されているデザイン エレメントに RLOC 制約を設定し、1 つの集合にグループ化できるようにします。

U_SET 集合のエレメントは、デザイン階層のどこにあってもかまいません。デザイン階層に関係なく、任意のオブジェクトを選択して、それらを U_SET 集合のエレメントとして指定できます。

詳細は、「RLOC の集合」を参照してください。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

この制約は、次のエレメントまたはデザイン エレメントのカテゴリを 1 つまたは複数使用して設定できます。すべてのデバイスですべてのエレメントがサポートされるわけではありません。デバイス別にどのデザイン エレメントが使用可能かを確認するには、そのデバイスのライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

- レジスタ
- マクロ インスタンス
- FMAP
- ROM
- RAMS
- RAMD
- BUFT
- MULT18X18S
- RAMB4_*Sm_Sn*
- RAMB4_*Sn*
- RAMB16_*Sm_Sn*
- RAMB16_*Sn*
- RAMB16
- DSP48

適用ルール

この制約はマクロの制約なので、ネットには適用できません。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  U_SET
- 属性値

  name は、集合の識別名です。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute U_SET: string;
```

VHDL 制約を次のように指定します。

```
attribute U_SET of {component_name|label_name}:{component|label}is name;
```

name

集合の識別名

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* U_SET =  *) name
```

name

集合の識別名

UCF および NCF 構文

```
INST "instance_name" U_SET= name;
```

name

- 集合の識別名です。
- この名前は絶対名です。
- 名前の前に階層修飾子は付けません。

次の文は、デザイン エレメント ELEM_1 が集合 JET_SET に属するように指定しています。

```
INST "$1I3245/ELEM_1" U_SET=JET_SET;
```

XCF 構文

```
BEGIN MODEL entity_name
 INST "instance_name" U_SET=uset_name;
END;
```

USE_INTERNAL_VREF

USE_INTERNAL_VREF 制約には、次のような特徴があります。

- 該当する I/O バンクの内部 Vref 機能に電圧値を割り当てることができます。
- 参照電圧を提供するファンクションから I/O バンクの Vref ピンを空けます。
- VREF ピンを次のいずれかのために指定できます。
  - VREF 用
  - 別の使用目的

アーキテクチャ サポート

Virtex®-6 デバイスにのみ適用できます。

適用可能エレメント

この制約は、インスタンス、コンポーネント、ネットに使用できます。

適用ルール

- 基本的にはネットに設定できませんが、ネットがパッドに接続されている場合は例外です。この場合、パッド インスタンスに設定されていると見なされます。
- デザイン エレメントに使用されると、そのエンティティに適用されます。

制約値

- TRUE

  特定エレメントに制約が適用されます。
- FALSE

  特定エレメントに制約が適用されません。
- DONT_CARE

  VREF ピンの使用はツールで決定されます。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図構文

- 属性名

  USE_INTERNAL_VREF
- 属性値

  上記の「制約値」を参照してください。

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* USE_INTERNAL_VREF = "{TRUE|FALSE|DONT_CARE}" *)
```

デフォルトは DONT_CARE です。

UCF および NCF 構文

```
INST "instance_name" USE_INTERNAL_VREF={TRUE|FALSE|DONT_CARE};
```

デフォルトは TRUE です。

XCF 構文

```
MODEL "entity_name" use_internal_vref ={true|false|dont_care}
```

デフォルトは TRUE です。

USE_LUTNM

USE_LUTNM (ルックアップ テーブル名の使用) 制約には、次の特徴があります。

- 高度なマップおよび配置制約です。
- 集合の特定のエレメントまたは一部に対して、LUTNM (ルックアップ テーブル名) 制約を適用するかどうかを指定します。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

集合の要素であるインスタンスまたはマクロに適用されます。

適用ルール

ネットには設定できません。

制約値

- TRUE (デフォルト)

  特定エレメントに制約が適用されます。
- FALSE

  特定エレメントに制約が適用されません。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図構文

- 集合のエレメントに設定します。
- 属性名

  USE_LUTNM
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute USE_LUTNM: string;
```

VHDL 制約を次のように指定します。

```
attribute USE_LUTNM of entity_name: entity is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* USE_LUTNM = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

```
INST "instance_name" USE_LUTNM={TRUE|FALSE};
```

XCF 構文

```
MODEL "entity_name" use_lutnm={true|false};
```

USE_RLOC

USE_RLOC (相対配置使用) 制約には、次のような特徴があります。

- 高度なマップおよび配置制約です。
- 集合の特定のエレメントまたは一部に対して、RLOC 制約を適用するかどうかを指定します。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

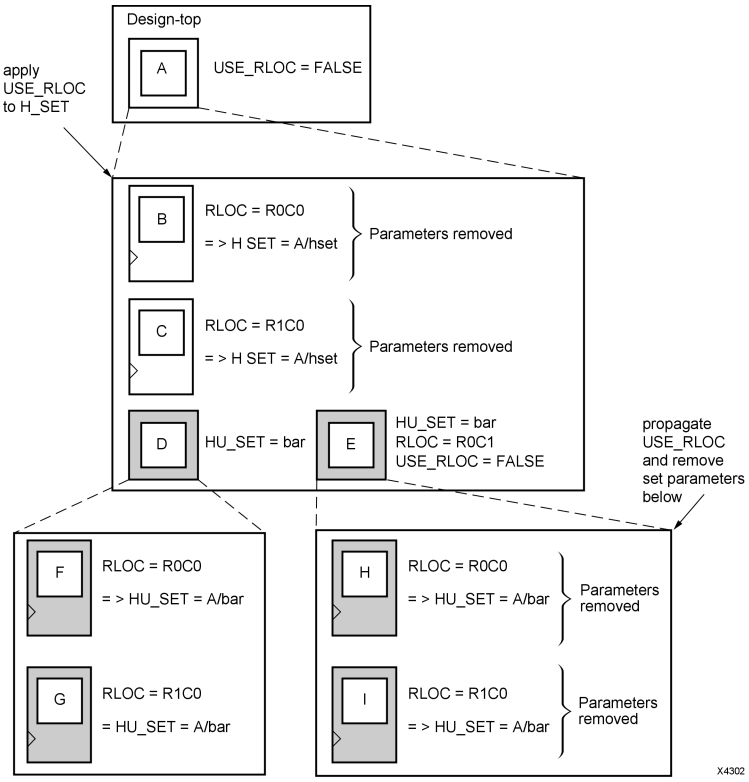
集合の要素であるインスタンスまたはマクロに適用されます。

適用ルール

ネットまたはマクロには設定できません。

デザイン エレメントに設定すると、そのデザイン エレメントの階層にあるすべての適用可能エレメントに適用されます。

USE_RLOC 制約を使用した H_SET および HU_SET 集合の RLOC の制御

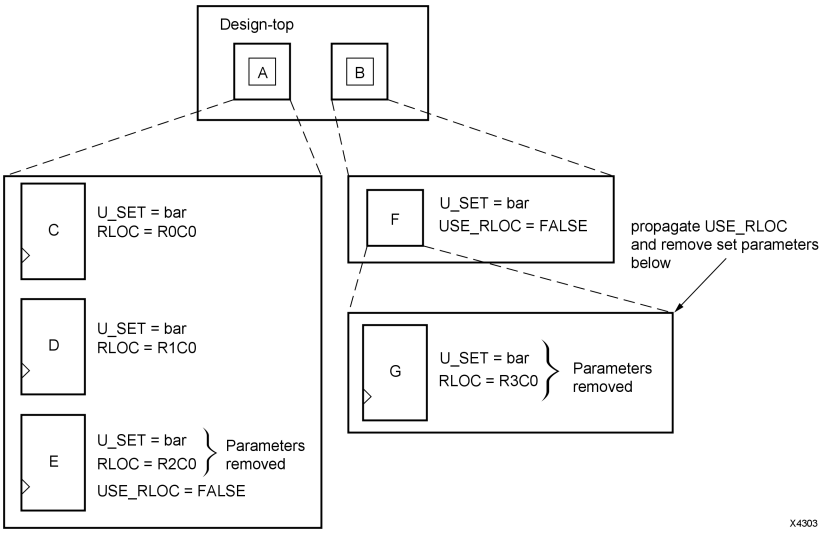


U_SET 集合には階層がないので、U_SET 集合に USE_RLOC 制約を適用するのは特殊なケースです。USE_RLOC 制約は階層を基に厳密に伝搬されるため、デザインの異なる階層レベルにある U_SET 集合のエレメントには、別々に USE_RLOC 制約を設定する必要があります。特定の USE_RLOC 制約は、異なる階層レベルにある集合のエレメントには伝搬されません。

インスタンシエーション マクロを使用して U_SET 集合を作成する場合は、そのマクロに USE_RLOC 制約を設定して、集合の全エレメントに階層を通して伝搬できます。

このようなマクロを作成するには、マクロに U_SET 制約を設定し、そのマクロの下位にある RLOC 制約が設定されているすべてのシンボルに対して、その制約が伝搬されるようにします。

USE_RLOC 制約を使用した U_SET 集合の RLOC の制御



- プリミティブ E の USE_RLOC=FALSE によって U_SET 集合から RLOC 制約が削除されます。
- エレメント F の USE_RLOC=FALSE がプリミティブ G に伝搬され、U_SET 集合から RLOC 制約が削除されます。

USE_RLOC 制約を伝搬するときに、上位階層に既に USE_RLOC 制約が設定されているエレメントがあると、それより下位の USE_RLOC 制約が無視されます。たとえば、USE_RLOC=FALSE 制約を伝搬する場合、USE_RLOC=TRUE 制約が下位のエレメントに設定されていても、その TRUE 制約は無視されます。

制約値

- TRUE (デフォルト)

  指定したエレメントにRLOC 制約が適用されます。
- FALSE

  指定したエレメントにRLOC 制約が適用されません。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 集合のエレメントに設定します。
- 属性名

  USE_RLOC
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute USE_RLOC: string;
```

VHDL 制約を次のように指定します。

```
attribute USE_RLOC of entity_name: entity is "{TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* USE_RLOC = "{TRUE|FALSE}" *)
```

UCF および NCF 構文

```
INST "instance_name" USE_RLOC={TRUE|FALSE};
```

XCF 構文

```
MODEL "entity_name" use_rloc={true|false};
```

USELOWSKEWLINES

ロー スキュー ラインの使用 (USELOWSKEWLINES) 制約には、次の特徴があります。

- 配線制約です。
- ネットでロー スキュー配線リソースを使用するよう指定します。

これらのリソースは、内部で生成される信号と外部で生成される信号の両方に使用できます。外部で生成される信号は、IOB コンポーネントで駆動されます。

ネットに USELOWSKEWLINES を設定すると、次のようになります。

- そのネットはロー スキュー リソースの 1 つに配線されます。
- タイミング解析では、ロー スキュー リソースを使用するレジスタ間のパスにあるスキューが自動的に考慮され、レポートされます。

この制約は、4 つのプライマリ グローバル クロックがすべて使用されている場合のみ、使用可能です。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

ネットに適用できます。

適用ルール

適用されたネットにのみ適用されます。

制約値

- YES
- NO
- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図構文

- 出力ネットに設定します。
- 属性名

  USELOWSKEWLINES
- 属性値
  - TRUE
  - FALSE

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute USELOWSKEWLINES: string;
```

VHDL 制約を次のように指定します。

```
attribute USELOWSKEWLINES of signal_name: signal is
"{YES|NO|TRUE|FALSE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* USELOWSKEWLINES = "{YES|NO|TRUE|FALSE}" *)
```

UCF および NCF 構文

次の文は、ネット $1I87/1N6745 を、デバイスのロー スキュー配線リソースのうち 1 つに配線するよう指定します。

```
NET "$1I87/$1N6745" USELOWSKEWLINES;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 NET "signal_name" uselowskewlines={yes|true};
END;
```

Constraints Editor の設定

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

PCF 構文

UCF および NCF 構文と同じ

VCCAUX

VCCAUX 制約には、次のような特徴があります。

- Spartan®-3A および Spartan-6 デバイスの VCCAUX ピンの電圧値を定義する制約です。
- 次の I/O 配置のバンク規則に影響します。
  - 自動配置プログラム
  - PACE (I/O ピン配置ソフトウェア)
- デバイスのビットストリームに影響します。

アーキテクチャ サポート

この制約は、次のデバイスでサポートされます。

- Spartan-3A
- Spartan-6

適用可能エレメント

- グローバル属性です。
- 特定のエレメントには設定されません。

適用ルール

なし

制約値

- 2.5
- 3.3

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

```
CONFIG VCCAUX="value";
CONFIG VCCAUX=3.3;
```

VCCAUX_IO

VCCAUX_IO (補助 I/O) 電源レールには、次のような特徴があります。

- HP I/O バンク専用です。
- シングルエンド回路および差動入力バッファー回路を含めた HP バンクの I/O 回路の一部に電源を供給するために使用されます。

HP I/O バンクには、次が含まれます。

- VCCAUX_IO ピン
- さまざまな内部ブロック機能に電源を供給する標準の VCCAUX ピン

7 シリーズ FPGA デバイス パッケージの場合、VCCAUX_IO ピンは 3 ～ 4 の I/O バンクのグループ内で接続されます。

- VCCAUX_IO ピンが一緒のグループになった I/O バンクの数は、7 シリーズのパーツおよびパッケージの組み合わせよって異なります。
- 各パーツおよびパッケージの組み合わせのグループ化されるバンクについては、『7 シリーズ パッケージおよびピン配置ガイド』を参照してください。

VCCAUX および VCCAUX_IO 電源は、VCCO 電源にりも前にオンにしておく必要があります。電源要件に関する詳細は、『7 シリーズ FPGA データシート』を参照してください。

アーキテクチャ サポート

- Kintex™-7
- Virtex®-7

適用可能エレメント

詳細は、『7 シリーズ I/O ユーザー ガイド』を参照してください。

適用ルール

VCCAUX_IO 制約の詳細は、『7 シリーズ I/O ユーザー ガイド』を参照してください。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

属性名

VCCAUX_IO

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute VCCAUX_IO: string;
```

VHDL 制約を次のように指定します。

```
attribute VCCAUX_IO of {component_name|label_name}:
{component|label} is "{NORMAL|HIGH|DONTCARE}";
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* VCCAUX_IO = {NORMAL|HIGH|DONTCARE}*)
```

UCF および NCF 構文

```
NET "net_name" VCCAUX_IO=(0|NORMAL|HIGH|DONTCARE);
INST "instance_name" VCCAUX_IO=(NORMAL|HIGH|DONTCARE);
```

VOLTAGE

VOLTAGE (電圧) 制約には、次のような特徴があります。

- タイミング制約です。
- 動作電圧を指定できます。

比例配分

- 指定した電圧に基づいてデバイスの遅延特性を比例配分できるように、動作電圧を指定します。
- 比例配分は、既存のスピード ファイルの遅延に対して行われ、すべての遅延に対してグローバルに適用されます。
- 新しいデバイスでは、タイミング情報 (スピード ファイル) が Production ステータス (製品版) になるまで電圧の比例配分がサポートされない場合があります。

サポートされる電圧の範囲

アーキテクチャにはそれぞれ、サポートされる電圧の範囲があります。入力した電圧がサポートされる範囲外の場合は、次のようになります。

- 制約は無視されます。
- アーキテクチャ別のデフォルト値が使用されます。
- エラー メッセージがスタティック タイミング中に表示されます。

アーキテクチャ サポート

- Spartan®-3A
- Spartan-3E
- Virtex®-4
- Virtex-5

適用可能エレメント

デザイン全体にグローバルに適用

適用ルール

この制約はデザイン エレメント制約です。ネットへは接続できません。

制約値

- value

  電圧を指定する実数

- V

  ボルト (デフォルトの電圧の単位)

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

VOLTAGE=*value* [**V**]**;**

次の文は、スピード ファイル遅延に関するすべての解析で、動作電圧が 5 ボルトになるように指定します。

```
VOLTAGE=5;
```

Constraints Editor の構文

構文も含めた Constraints Editor での制約設定に関する詳細は、Constraints Editor ヘルプを参照してください。

PCF 構文

UCF および NCF 構文と同じ

VCCOSENSEMODE

VCCOSENSEMODE (VSSO センス モード) 制約には、次のような特徴があります。

- HR (3.3V) I/O バンクで特殊なハードウェアをプログラムできるようになります。この特殊なハードウェアには、次のような特徴があります。
  - VCCO 電圧がユーザー プログラムの電圧とは異なるバンクに適用されるかどうかが検出されます。
  - ユーザー プログラムの電圧を上書きし、検出された電圧に基づいてバンクを高電圧モードに変更できます。
- HR バンクをほかのバンクとは別にプログラムできます。

アーキテクチャ サポート

- Artix™-7 デバイス
- Kintex™-7 デバイス

適用可能エレメント

グローバル CONFIG 制約です。特定のインスタンスや信号名には設定されません。

適用ルール

デザイン全体の特定バンクの I/O に適用されます。

制約値

- *n*

  バンク数
- OFF

  バンクの過電圧検出回路がディスエーブルになります。
- ALWAYSACTIVE

  バンクの過電圧検出回路が常にイネーブルになります。
- FREEZE

  バンクの過電圧検出回路がコンフィギュレーション シーケンス中にのみイネーブルになります。

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

UCF および NCF 構文

```
CONFIG VCCOSENSEMODEn=[OFF|ALWAYSACTIVE|FREEZE];
```

UCF および NCF の構文例

```
CONFIG VCCOSENSEMODE5=OFF;
```

VREF

VREF 制約には、次の特徴があります。

- グローバル属性として適用されます。
- エレメントには直接適用されません。
- リストされたピンが VREF の供給ピンとして、SSTL または HSTL I/O 標準のいずれかで指定された I/O ピンと共に使用されます。
- CoolRunner™-II デバイスの場合、どの I/O も VREF として使用できます。
- VREF ピンとして使用するピンを指定できます。

  注記 : レポート ファイル (RPT) でピン割り当てを確認してください。

次のような場合、フィッターにより自動的に必要な VREF が割り当てられます。

- 差動 I/O 規格の HSTL および SSTL に VREF ピンを指定しない場合
- 差動 I/O ピンの近くに必要な VREF ピンが指定されていない場合

アーキテクチャ サポート

128 個以上のマクロセルを使用した CoolRunner-II デバイスにのみ適用できます。

適用可能エレメント

デザイン全体にグローバルに適用

適用ルール

リストされたピンが VREF の供給ピンとして、SSTL または HSTL I/O 規格のいずれかで指定された I/O ピンと共に使用されます。

制約値

- P*nn*

  nn

  ピン番号
- *rc*
  - *r*

    行を表すアルファベット
  - *c*

    列を表す数値

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

VREF=*value_list*

CONFIG シンボルに適用します。

UCF および NCF 構文

CONFIG VREF=*value_list*;

```
CONFIG VREF=P12,P13;
```

WIREAND

WIREAND (ワイヤード AND) 制約には、次のような特徴があります。

- 高度なフィッタ制約です。
- 指定したノードをワイヤード AND ファンクションとしてインターコネクト (UIM および Fastconnect) にインプリメントします。

アーキテクチャ サポート

XC9500 デバイスでのみサポートされます。

適用可能エレメント

すべてのネットに適用されます。

適用ルール

この制約はネット制約です。デザイン エレメントへは接続できません。

制約値

- YES
- NO
- TRUE
- FALSE

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

XBLKNM

XBLKNM 制約には、次のような特徴があります。

- 高度なマップ制約です。
- ブロック名を適切なプリミティブおよびロジック エレメントに割り当てます。

同じ XBLKNM 制約が複数のインスタンスに設定されると、同じブロック名のロジックが 1 つまたは複数のスライスにパックされます。逆に、2 つのシンボルの XBLKNM 名が異なる場合は、同じブロックにはマップされません。1 つのブロックに収まらない複数のインスタンスに同じ XBLKNM を設定すると、エラーが発生します。

FMAP シンボルに同じ XBLKNM を指定すると、関連するファンクション ジェネレーターが 1 つのスライスにグループ化されます。XBLKNM を使用すると、スライスをデバイス上の物理的な位置に制約することなく、スライスを分割できます。

XBLKNM には階層パスは接頭辞として追加されないため、各スライスには固有の XBLKNM 名を割り当てる必要があります。

BLKNM を使用すると、別の BLKNM が指定されているエレメントを除き、すべてのエレメントを同じ物理的コンポーネントにマップできます。XBLKNM 属性の場合は、同じ XBLKNM が設定されているエレメントのみが、同じ物理コンポーネントにマップされます。XBLKNM が設定されていないエレメントは、XBLKNM が設定されているエレメントと同じ物理コンポーネントにはマップできません。

この制約は、ブロック RAM とも使用できます。

アーキテクチャ サポート

すべての FPGA に適用されますが、CPLD デバイスには適用されません。

適用可能エレメント

どのデバイス ファミリにどのエレメントを使用できるかは、ライブラリ ガイドを参照してください。詳細は、デバイスのデータシートを参照してください。

適用ルール

それが接続されているデザイン エレメントに適用

制約値

block_name

そのシンボル タイプに対して有効なブロック名

構文例

このセクションの構文例は、特定のツールまたは手法でこの制約を使用する方法について示しています。ここにリストされていないツールまたは手法は、この制約には使用できません。

回路図

- 有効なインスタンスに設定します。
- 属性名

  XBLKNM
- 属性値

  上記の「制約値」を参照してください。

VHDL 構文

VHDL 制約を次のように宣言します。

```
attribute XBLKNM: string;
```

VHDL 制約を次のように指定します。

```
attribute XBLKNM
of {component_name|label_name}: {component|label} is block_name;
```

Verilog 構文

Verilog 制約をモジュールまたはインスタンシエーション文の直前に記述します。

Verilog 制約を次のように指定します。

```
(* XBLKNM = "block_name" *)
```

UCF および NCF 構文

```
INST "instance_name" XBLKNM=block_name;
```

次の文は、エレメント flip_flop2 のインスタンシエーションをブロック U1358 に割り当てています。

```
INST "$1I87/flip_flop2" XBLKNM=U1358;
```

XCF 構文

```
BEGIN MODEL "entity_name"
 INST "instance_name" xblknm=xblknm_name;
END;
```

XILINX®

付録

その他のリソース

- ザイリンクス用語集：
  http://japan.xilinx.com/support/documentation/sw_manuals/glossary.pdf
- ザイリンクス マニュアル：
  http://japan.xilinx.com/support/documentation
- ザイリンクス サポート：
  http://japan.xilinx.com/support