

System Generator for DSP

入門ガイド

UG639 (v11.4) 2010 年 12 月 2 日



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

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

© 2009 Xilinx, Inc. XILINX, the Xilinx logo, Virtex, Spartan, ISE, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

本資料は英語版 (v.11.4) 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。

目次

このマニュアルについて

マニュアルの内容	7
System Generator の PDF マニュアル セット	7
その他のリソース	8
表記規則.....	8
書体.....	8
オンライン マニュアル.....	9

第 1 章：概要

ザイリンクス DSP ブロックセット	12
FIR フィルタの生成	13
MATLAB のサポート	14
システム リソースの予測	15
ハードウェア協調シミュレーション.....	16
システム統合プラットフォーム	17

第 2 章：インストール

ダウンロード	19
ハードウェア協調シミュレーションのサポート	19
システム要件および推奨事項	19
推奨ハードウェア.....	19
オペレーティング システム (OS) およびソフトウェア要件	20
ほかのツールとの互換性	21
ソフトウェア要件.....	21
ISE Design Suite インストーラの使用.....	21
Linux OS への System Generator のインストール.....	22
インストール後のタスク.....	29
ハードウェア協調シミュレーション用のインストール.....	29
ザイリンクス HDL ライブラリのコンパイル	30
System Generator キャッシュの設定	30
System Generator のバージョンの表示と切り替え	30

第 3 章：リリース情報

リリース ノート 11.4.....	33
System Generator の改善点	33
ザイリンクス ブロックセットの改善点.....	33
システム要件および推奨事項.....	35
既知の問題	36
リリース ノート 11.3.....	37
System Generator の改善点	37
ザイリンクス ブロックセットの改善点.....	37
使用できなくなった System Generator の機能	38
システム要件および推奨事項.....	38
既知の問題	40
リリース ノート 11.2.....	41
System Generator の改善点	41
ザイリンクス ブロックセットの改善点.....	43
システム要件および推奨事項.....	44
既知の問題	45

リリース ノート 11.1	46
System Generator の改善点	46
ザイリンクス ブロックセットの改善点	49
ザイリンクス基本ブロックの改善点	49
置き換えられたザイリンクス ブロック	50
システム要件および推奨事項	51
既知の問題	52
リリース ノート 10.1.3	53
ザイリンクス DSP ブロックセットの改善点	53
ツールフローとの互換性	54
既知の問題	54
リリース ノート 10.1.2	55
System Generator の改善点	55
ザイリンクス DSP ブロックセットの改善点	55
ツールフローとの互換性	56
既知の問題	56
リリース ノート 10.1.1	57
System Generator の改善点	57
ザイリンクス DSP ブロックセットの改善点	57
ツールフローとの互換性	58
既知の問題	58
リリース ノート 10.1	59
System Generator の改善点	59
ザイリンクス DSP ブロックセットの改善点	60
ツールフローとの互換性	60
既知の問題	60
ザイリンクス System Generator モデルのアップデート	61
V2.x 以前のモデルのアップデート	61
v3.x、v6.x、および v7.x モデルのアップデート	61
例	62

第 4 章：入門

概要	63
レッスン 1：デザイン作成の基礎	64
System Generator デザイン フロー	64
ザイリンクス DSP ブロックセット	65
FPGA の境界の定義	66
System Generator トークンの追加	67
DSP デザインの作成	68
HDL コードの生成	69
System Generator を使用したモデル ベースのデザイン	70
MATLAB を使用した入力ベクタの作成	71
レッスン 1 のまとめ	72
演習： Simulink の使用	72
演習： System Generator 入門	72
レッスン 2：固定小数点およびビット操作	73
固定小数点数値精度	73
System Generator 固定小数点量子化	74
オーバーフロー モードと量子化モード	75
ビット レベルの操作	76
Reinterpret ブロック	77
Convert ブロック	78
Concat ブロック	79
Slice ブロック	80
BitBasher ブロック	81
レッスン 2 のまとめ	82
演習： 信号配線	82

レッスン 3 : システム制御	83
DSP システムの制御	83
MCode ブロック	84
ザイリンクス xl_state データ型	85
ステート マシンの例	86
Expression ブロック	87
リセット ポートとイネーブル ポート	88
バースト データ	89
レッスン 3 のまとめ	90
演習 : システム制御	90
レッスン 4 : マルチレート システム	91
マルチレート システムの作成	91
Up Sample および Down Sample ブロック	92
レート変更ファンクション ブロック	93
Simulink でのレート変化の表示	94
ツールのデバッグ	95
サンプリング周期に関する規則	96
演習 : マルチレート システム	97
レッスン 5 : メモリの使用	98
ブロック RAM と分散 RAM	98
RAM および ROM の初期化	99
System Generator の RAM ブロック	100
System Generator の ROM ブロック	101
Delay ブロック	102
FIFO ブロック	103
Shared Memory ブロック	104
演習 : メモリの使用	105
レッスン 6 : フィルタの設計	106
概要	106
Virtex DSP48 スライス	107
FIR Compiler ブロック	108
FDATool を使用した係数の作成	109
FDATool の係数の使用	110
演習 : フィルタの設計	111
その他の例とチュートリアル	112
ブラック ボックスの例	112
ChipScope の例	112
DSP の例	112
MCode の例	113
プロセッサの例	114
共有メモリの例	114
タイミング解析の例	115
その他の例	115
System Generator デモ	115
索引	117

このマニュアルについて

このマニュアルでは、System Generator for DSP を紹介し、インストールおよびコンフィギュレーション手順、リリース情報、このツールの主な機能を使用する 6 つのトレーニング モジュールを示します。各モジュールは、8 ～ 10 枚のスライドを使用した重要な概念の説明と、30 分程度で終了する演習で構成されています。この入門トレーニングはツールの一部であるので、時間のあるときに独自のペースで進めることができます。

マニュアルの内容

このマニュアルには、次の内容が含まれています。

- 概要
- インストール
- リリース情報
- 入門
 - a. デザイン作成
 - b. 固定小数点およびビット操作
 - c. システム制御
 - d. マルチレート システム
 - e. メモリの使用
 - f. フィルタの設計
 - g. その他の例とチュートリアル

System Generator の PDF マニュアル セット

このマニュアルは System Generator の Help システムから参照でき、また System Generator の PDF マニュアル セットの一部です。この PDF マニュアル セットには、次のマニュアルが含まれています。

- System Generator for DSP 入門ガイド
- System Generator for DSP ユーザー ガイド
- System Generator for DSP リファレンス ガイド

メモ：これらのマニュアル間のハイパーリンクは、PDF ファイルが同じフォルダにある場合にのみ機能します。Adobe Reader でハイパーリンクをクリックした場合、Alt キーと左方向キー (←) を同時に押すと、前に参照していたページに戻ることができます。

その他のリソース

追加の資料は、次の Web サイトから参照できます。

<http://japan.xilinx.com/support/documentation/index.htm>

シリコンやソフトウェア、IP に関するアンサー データベースを検索したり、テクニカル サポートのウェブ ケースを開く場合は、次の Web サイトにアクセスしてください。

<http://japan.xilinx.com/support/mysupport.htm>

表記規則

このマニュアルでは、次の表記規則を使用しています。各規則について、例を挙げて説明します。

書体

次の規則は、すべてのマニュアルで使用されています。

表記規則	使用箇所	例
Courier フォント	システムが表示するメッセージ、プロンプト、プログラム ファイルを表示します。	<code>speed grade: - 100</code>
Courier フォント (太字)	構文内で入力するコマンドを示します。	ngdbuild <i>design_name</i>
イタリック フォント	ユーザーが値を入力する必要のある構文内の変数に使用します。	<i>ngdbuild design_name</i>
二重/一重かぎカッコ『』、『』、『』	『』はマニュアル名を、『』はセクション名を示します。	詳細については、『開発システムリファレンス ガイド』の「PAR」を参照してください。
角カッコ []	オプションの入力またはパラメータを示しますが、 bus[7:0] のようなバス仕様では必ず使用します。また、GUI 表記にも使用します。	ngdbuild [<i>option_name</i>] <i>design_name</i> [File] → [Open] をクリックします。
中カッコ { }	1 つ以上の項目を選択するためのリストを示します。	lowpwr = { on off }
縦棒	選択するリストの項目を分離します。	lowpwr = { on off }
縦の省略記号 . . .	繰り返し項目が省略されていることを示します。	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
横の省略記号 ...	繰り返し項目が省略されていることを示します。	allow block <i>block_name loc1 loc2 ... locn;</i>

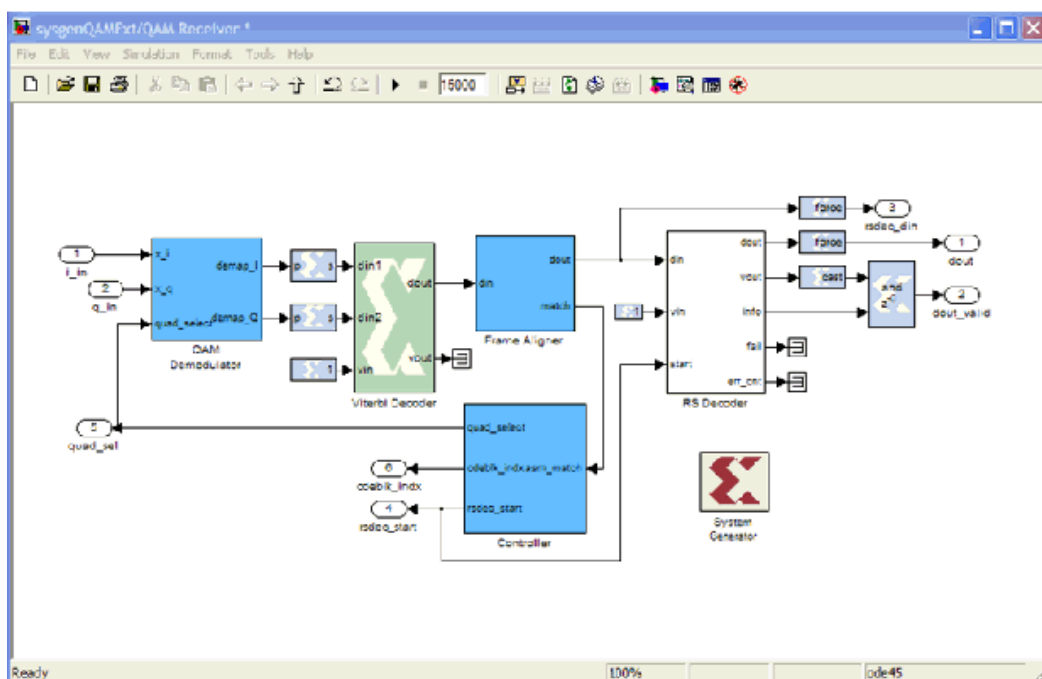
オンライン マニュアル

このマニュアルでは、次の規則が使用されています。

表記規則	使用箇所	例
青色の文字	マニュアル内の相互参照を示します。	詳細は、「 その他のリソース 」を参照してください。 詳細は、第 1 章「 タイトルフォーマット 」を参照してください。
赤色の文字	ほかのマニュアルへの相互参照を示します。	詳細は、『Virtex-4 Platform FPGA ユーザー ガイド』の 図 2-5 を参照してください。
青色の下線付き文字	Web サイト (URL) へのハイパーリンクです。	最新のスピード ファイルは、 http://japan.xilinx.com から入手できます。

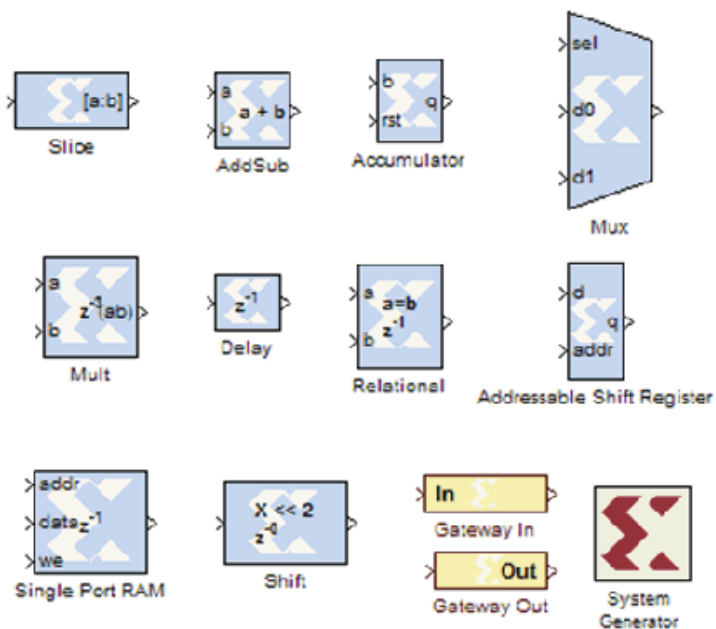
概要

System Generator はザイリンクスが提供する DSP デザイン ツールで、FPGA デザインに MathWorks モデルに基づくデザイン環境 Simulink® を使用できるようにしています。System Generator を使用する際、ザイリンクス FPGA または RTL デザイン設計の経験は必要ありません。デザインは、DSP 設計に適した Simulink モデル環境にザイリンクス専用のブロックセットを使用して表示されます。合成、配置配線などの FPGA インプリメンテーションプロセスは自動的に実行され、FPGA プログラム ファイルが生成されます。



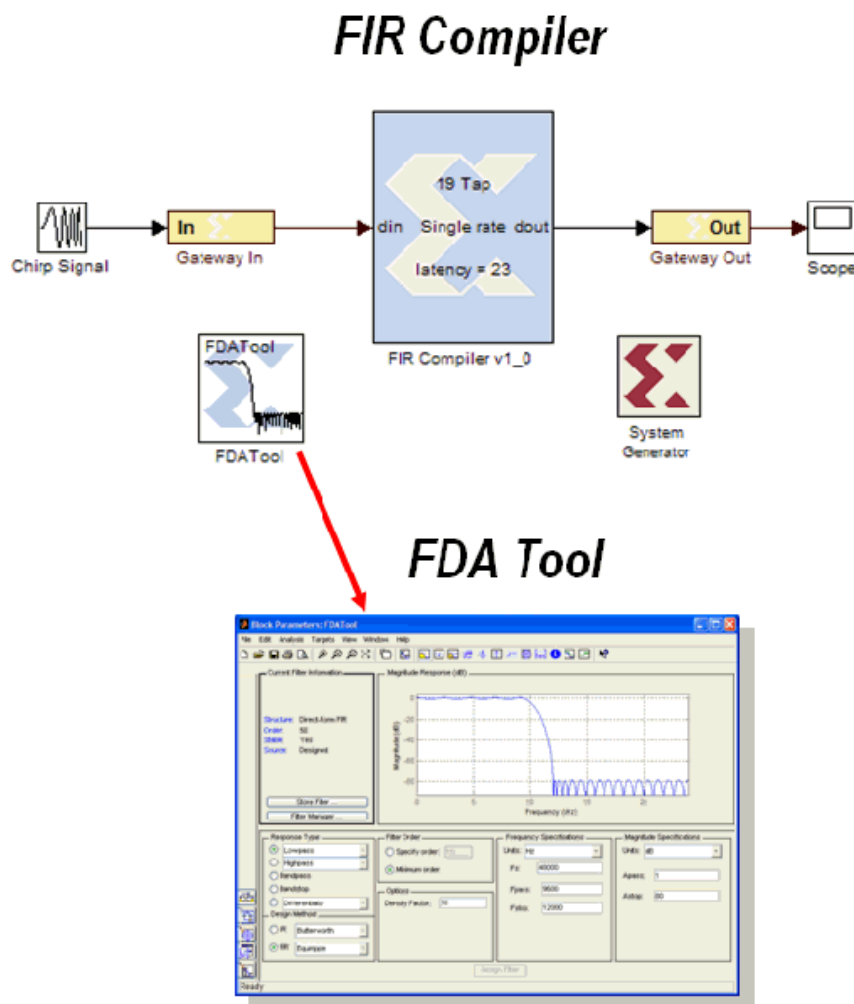
ザイリンクス DSP ブロックセット

Simulink 用のザイリンクス DSP ブロックセットには、90 個以上の DSP 機能ブロックが含まれています。加算器、乗算器、レジスタなどの一般的な DSP 機能ブロックや、順方向誤り訂正ブロック、FFT、フィルタ、メモリ などの複雑な DSP 機能ブロックがあります。これらのブロックは、ザイリンクス CORE Generator を使用しており、選択したデバイス用に最適化されたデザインを生成します。



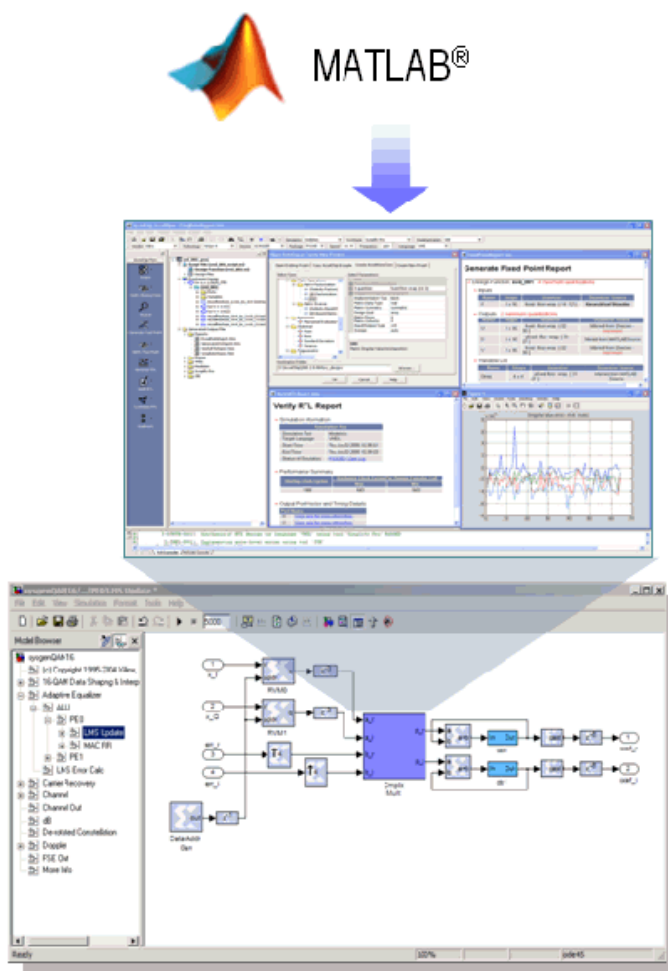
FIR フィルタの生成

System Generator には、Virtex®-4 および Virtex-5 に含まれる専用 DSP48 ハードウェア リソースを使用する FIR Compiler ブロックが含まれており、高度に最適化された、動作速度が 500MHz 以上のインプリメンテーションを作成できます。コンフィギュレーション オプションにより、直接型、多相デシメーション、多相補間、オーバーサンプリング インプリメンテーションを生成できます。fir2 などの標準 MATLAB 関数や MathWorks 社の FDATool を使用すると、ザイリンクス FIR Compiler 用の係数を作成できます。



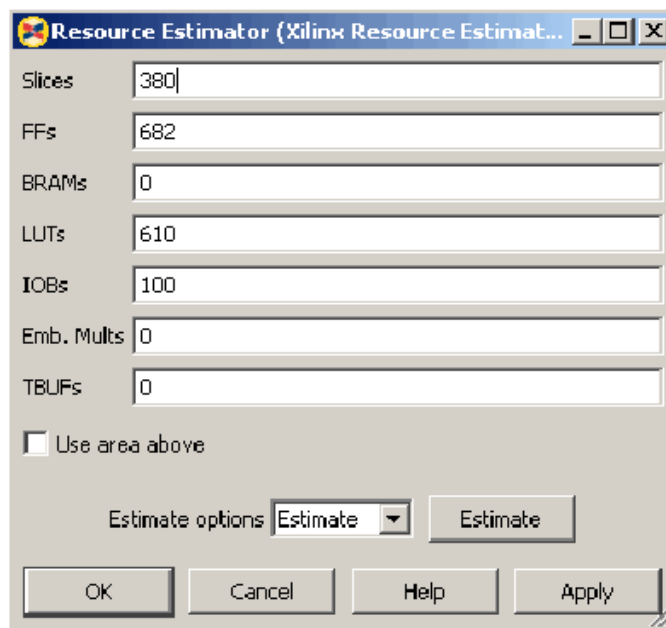
MATLAB のサポート

アルゴリズム MATLAB モデルは、AccelDSP™ を使用して System Generator に組み込むことができます。AccelDSP には高度なアルゴリズム合成が含まれており、入力された浮動小数点 MATLAB から System Generator 用の完全にスケジュールされた固定小数点モデルを生成します。浮動小数点から固定小数点への変換、自動 IP 挿入、デザイン解析、アルゴリズム スケジュールなどの機能が 있습니다。System Generator には MCode ブロックも含まれており、単純な制御操作のモデリングおよびインプリメンテーションに非アルゴリズム MATLAB を使用できます。



システム リソースの予測

System Generator には Resource Estimator ブロックが含まれており、配置配線の前にデザインのエリアを予測できます。エリア予測により、Virtex-5 デバイスに含まれる 640 個の乗算/累積 (または DSP) ブロックなど、FPGA リソースを最大限に利用できるのもので、ハードウェアとソフトウェアの分割プロセスに有益です。



Resource Estimator (Xilinx Resource Estimat...

Slices	380
FFs	682
BRAMs	0
LUTs	610
IOBs	100
Emb. Mults	0
TBUFs	0

☐ Use area above

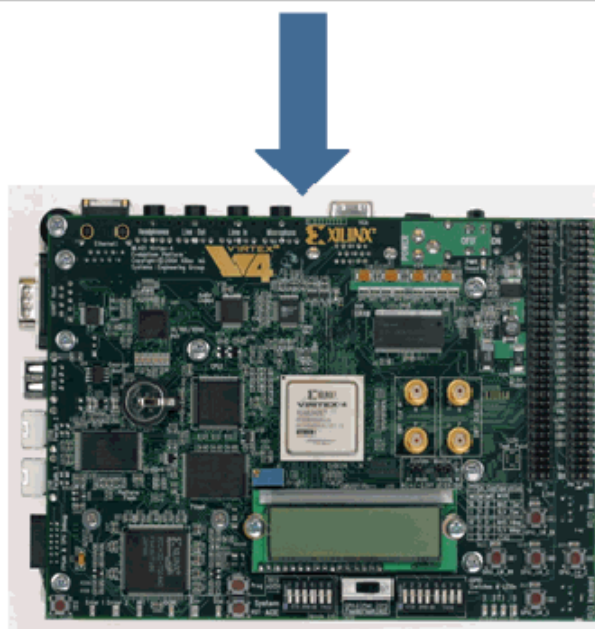
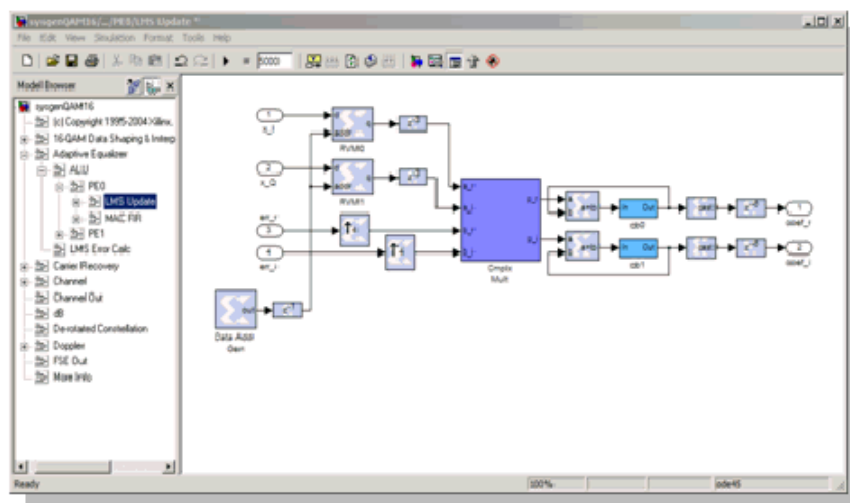
Estimate options: Estimate

Estimate

OK Cancel Help Apply

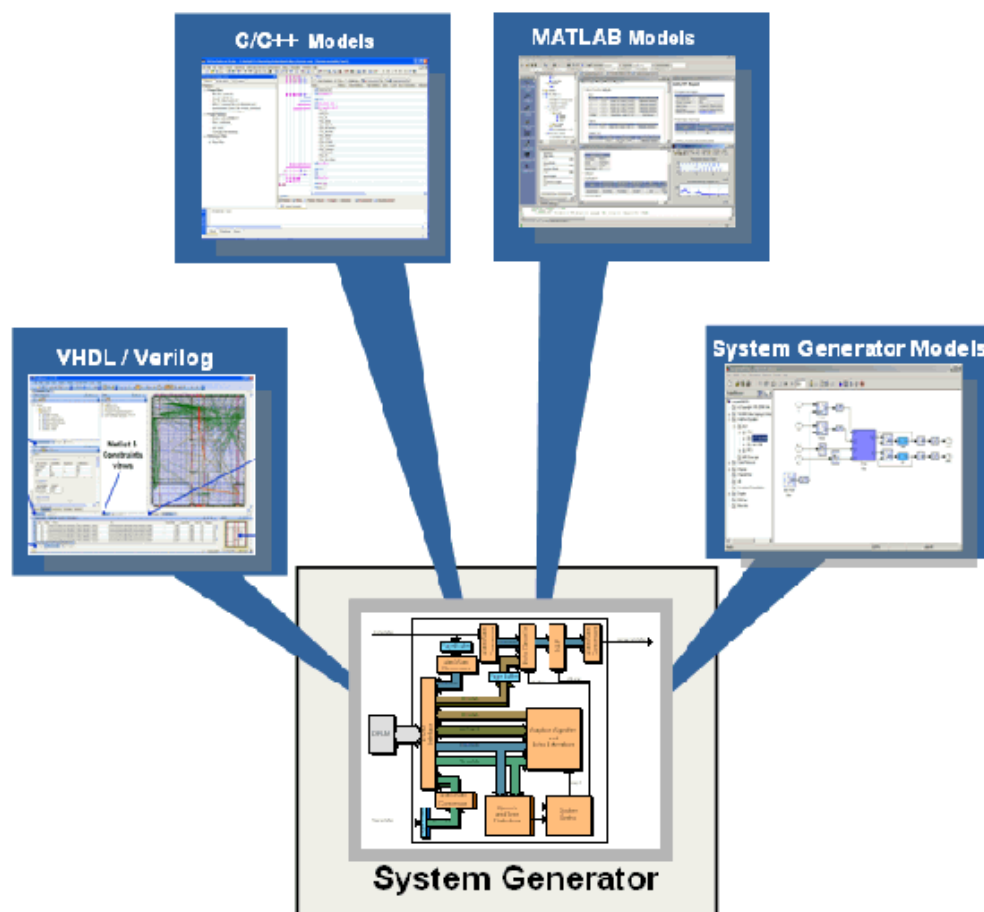
ハードウェア協調シミュレーション

System Generator では、ハードウェア協調シミュレーションを使用した高速シミュレーションが提供されています。ザイリックス DSP ブロックセットで取り込まれたデザインのハードウェア シミュレーション トークンが自動的に作成され、このトークンをサポートされる 20 以上のハードウェア プラットフォームのいずれかで実行できます。このハードウェアは、**Simulink** システムの残りの部分と共に協調シミュレーションされ、シミュレーション パフォーマンスは最大 1000 倍向上します。



システム統合プラットフォーム

System Generator は DSP FPGA の設計用にシステム統合プラットフォームを提供しており、DSP システムの RTL、Simulink、MATLAB、C/C++ コンポーネントを 1 つのシミュレーションおよびインプリメンテーション環境で操作できます。System Generator では、Simulink に RTL をインポート可能なブラックボックス ブロックがサポートされており、ModelSim またはザイリンクス ISE® Simulator を使用して協調シミュレーションできます。また、C/C++ プログラムを実行する MicroBlaze® エンベデッド プロセッサを含めることも可能です。



インストール

ダウンロード

System Generator は ISE® Design Suite に含まれており、ザイリンクスの Web サイトからダウンロードできます。次のサイトから購入、登録、ダウンロードできます。

<http://japan.xilinx.com/tools/sysgen.htm>

メモ：場合によっては、CD での送付も可能です。Web サイトからソフトウェアをダウンロードできない場合は、ザイリンクス販売代理店にご連絡ください。

ハードウェア協調シミュレーションのサポート

FPGA 開発ボードを使用する場合は、Simulink® によるシミュレーションとの FPGA ハードウェア協調シミュレーション機能を利用できます。System Generator では、XtremeDSP™ 開発キット、MicroBlaze™ マルチメディア デモ ボード、MVI ハードウェア プラットフォーム、ML402 Virtex®-4 プラットフォーム、ML506 Virtex-5 プラットフォーム、および Spartan®-3A DSP 1800 スタータ プラットフォーム/3400 開発プラットフォームがサポートされています。また、その他の System Generator ボード サポート パッケージでは、別のハードウェア協調シミュレーション プラットフォームがサポートされています。System Generator ボード サポート パッケージは、次のサイトからダウンロードできます。

http://japan.xilinx.com/products/boards_kits/index.htm

システム要件および推奨事項

推奨ハードウェア

表 2-1 : Windows ベースの推奨ハードウェア

推奨	メモ
2.00GB の RAM	
600MB のハード ディスク容量	最低必要条件
ザイリンクス ハードウェア協調シミュレーション プラットフォーム	ハードウェア協調シミュレーション フローに必要

表 2-2 : Linux ベースの推奨ハードウェア

推奨	メモ
4 GB の RAM	
600MB のハード ディスク容量	最低必要条件
ザイリンクス ハードウェア協調シミュレーションプラットフォーム	ハードウェア協調シミュレーション フローに必要

オペレーティング システム (OS) およびソフトウェア要件

表 2-3 : Windows ベースの OS およびソフトウェア要件

必要条件	メモ
Windows XP 32 ビット オペレーティング システム SP2 (英語版および日本語版)	
ザイリンクス ISE Design Suite 11.1 リリース	
MathWorks MATLAB®, Simulink (Fixed-Point Toolbox 含む) バージョン 2008b および 2008b	<p>MATLAB のインストール ディレクトリのパス名には、C:\MATLAB\R2008b のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>

表 2-4 : Linux ベースの OS およびソフトウェア要件

必要条件	メモ
Red Hat Linux 4.7、32 および 64 ビット オペレーティング システム (英語版のみ)	
ザイリンクス ISE Design Suite 11.1 リリース	
MathWorks MATLAB, Simulink (Fixed-Point Toolbox 含む) バージョン 2008b または 2008b	<p>MATLAB のインストール ディレクトリのパス名には、C:\MATLAB\R2008b のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>

ほかのツールとの互換性

System Generator は、統合フローに含まれるその他のソフトウェア ツールと共に動作するように設計されています。System Generator は、次のツールと互換性があります。

表 2-5：ほかのツールとの互換性

ツール	バージョン
Mentor Graphics 社 ModelSim SE および PE	6.4b
Synplcity 社 Synplify Pro	8.9

ソフトウェア要件

System Generator の一部の機能を動作させるには、次のソフトウェアをインストールする必要があります。

- 論理合成ツール。System Generator は、ザイリンクス XST (ISE Foundation に含まれる) および Synopsys 社の Synplify Pro v8.9 と完全に互換性があります。
- ハードウェア記述言語 (HDL) シミュレータ。System Generator を使用して Simulink 内の HDL モジュールの協調シミュレーションを実行する場合にのみ必要です。System Generator の HDL 協調シミュレーションのインターフェイスは、ザイリンクス ISE Simulator および Mentor Graphics 社の ModelSim SE と互換性があります。

メモ：Microsoft Windows 環境変数 \$XILINX に ISE のインストールディレクトリを指定する必要があります。

ISE サービス パックは、次のダウンロード センタのページからダウンロードできます。

http://japan.xilinx.com/xlnx/xil_sw_updates_home.jsp

ISE Design Suite インストーラの使用

ISE Design Suite インストーラを実行する前に、MATLAB のすべてのインスタンスを閉じてください。これらのインスタンスを閉じたらインストーラを起動して、画面に表示される指示に従ってください。

Windows OS にインストールする場合の MATLAB バージョンの選択

Windows での System Generator のインストールの最後の手順で、このバージョンの System Generator と関連付ける MATLAB バージョンのチェック ボックスをオンにし、[Apply] をクリックします。

ネットワーク デバイスにインストールされているなどの理由で、有効な MATLAB バージョンがリストされていない場合は、[Add Version] ボタンをクリックし、MATLAB のルート ディレクトリを選択して [Add] をクリックします。このバージョンの MATLAB を System Generator と関連付ける場合は、追加した MATLAB のバージョンのチェック ボックスをオンにし、[Apply] をクリックします。

MATLAB がインストールされていない場合は、[Choose Later] をクリックしてインストールを続行します。MATLAB をインストールした後に MATLAB のバージョンと System Generator を関連付けることができます。この場合は、Windows で [スタート] → [すべてのプログラム] → [Xilinx ISE Design Suite 11.1] → [DSP Tools] → [Select MATLAB version for

Xilinx System Generator] をクリックするか、または Linux で sysgen ツリーの bin ディレクトリに含まれている sg_config コマンドを実行します。

Linux OS への System Generator のインストール

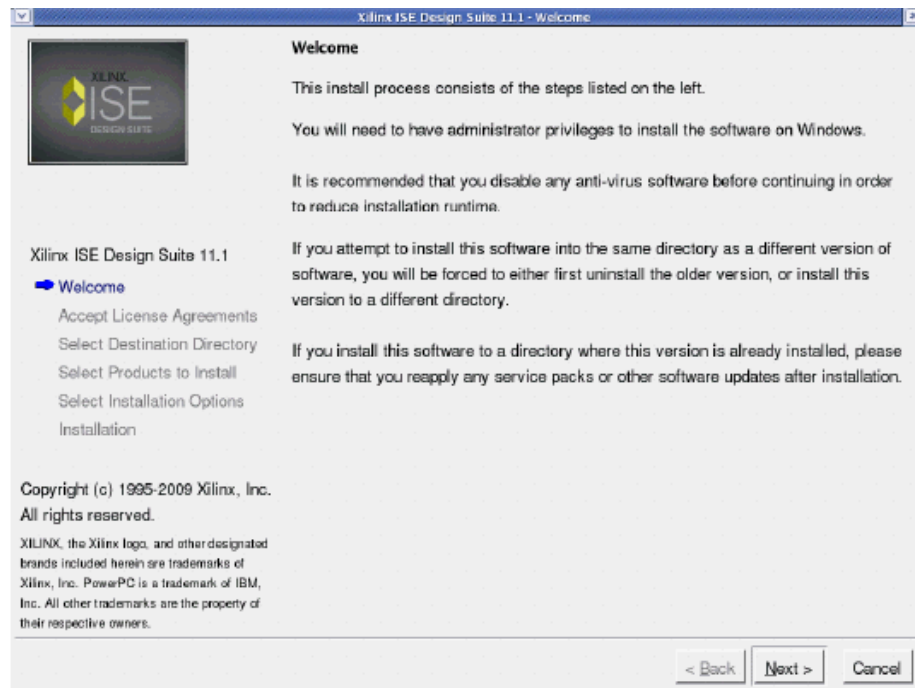
1. 32 ビットまたは 63 ビット Linux マシンで xsetup を実行します。プロンプトに「xsetup」を入力して、次のインストール手順に従います。

メモ：DSP Tools インストーラでは、OS タイプに一致するコードのみがインストールされます。

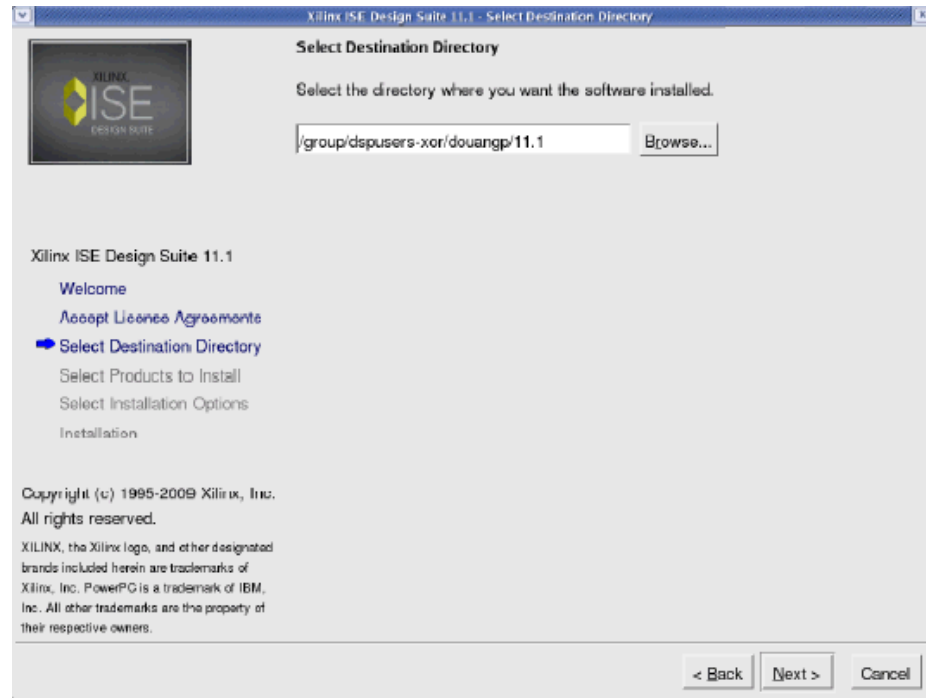
- 32 ビット Windows マシンでは、win32 インストールのみがインストールされます。
- 32 ビット Linux マシンでは、lin32 インストールのみがインストールされます。
- 64 ビット Linux マシンでは、lin64 インストールのみがインストールされます。

Linux 用のインストール ファイル数は Windows 用と比べて少ないため、プロセス時間は多少短くなります。

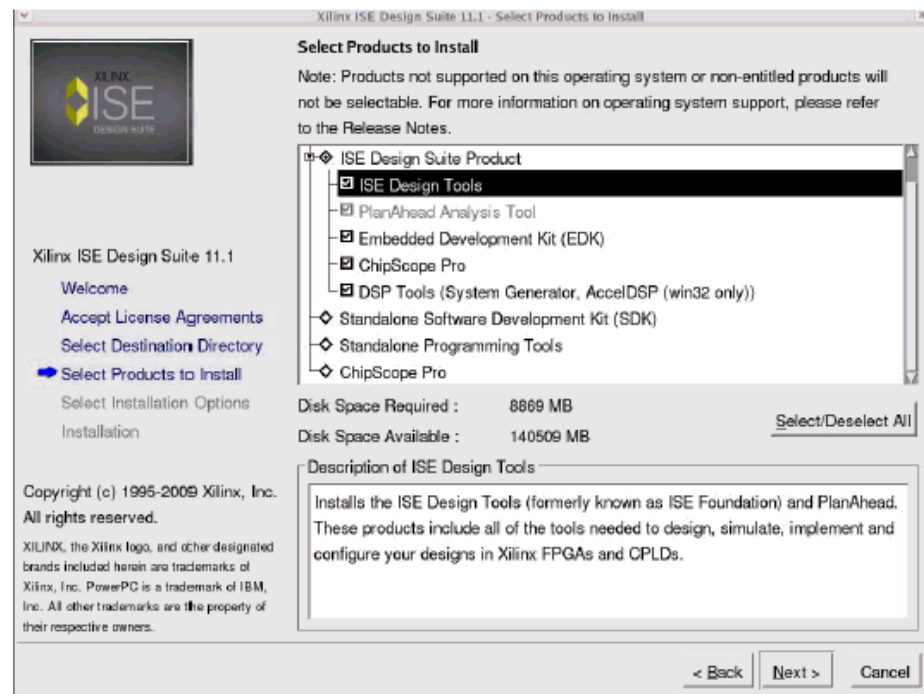
- a. [Next] をクリックしてインストールを開始します。



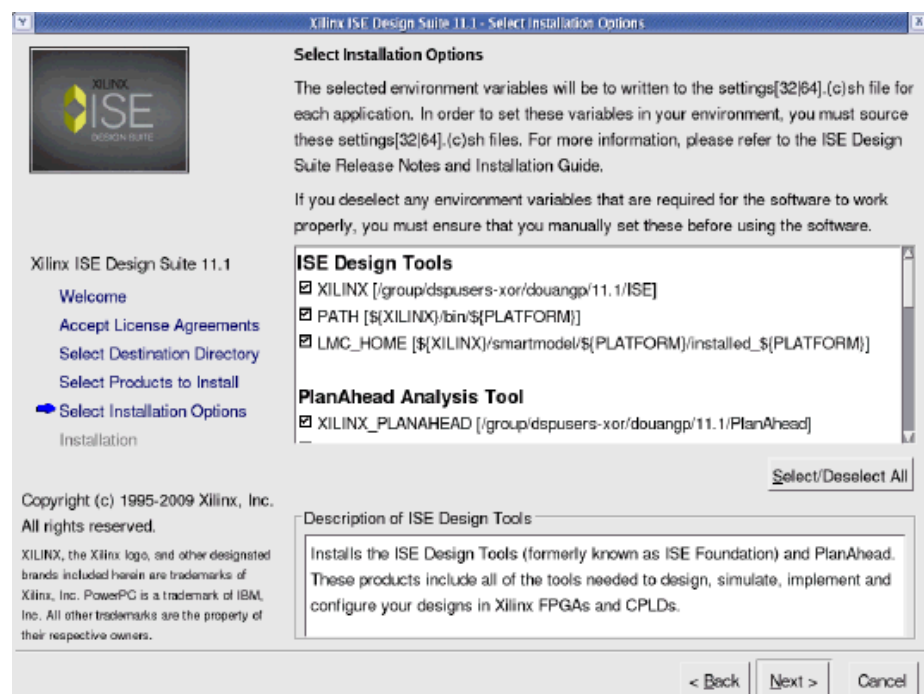
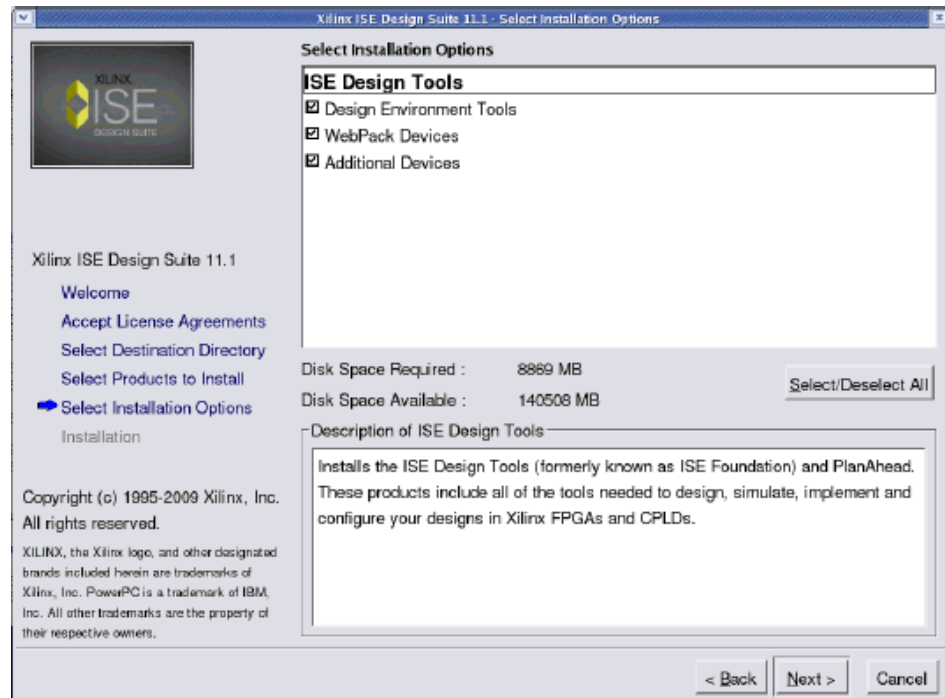
- b. 次の 2 ページでソフトウェア ライセンスを承諾して [Next] をクリックします。
- c. ISE Design Tools および DSP Tools のインストール ディレクトリを選択します。

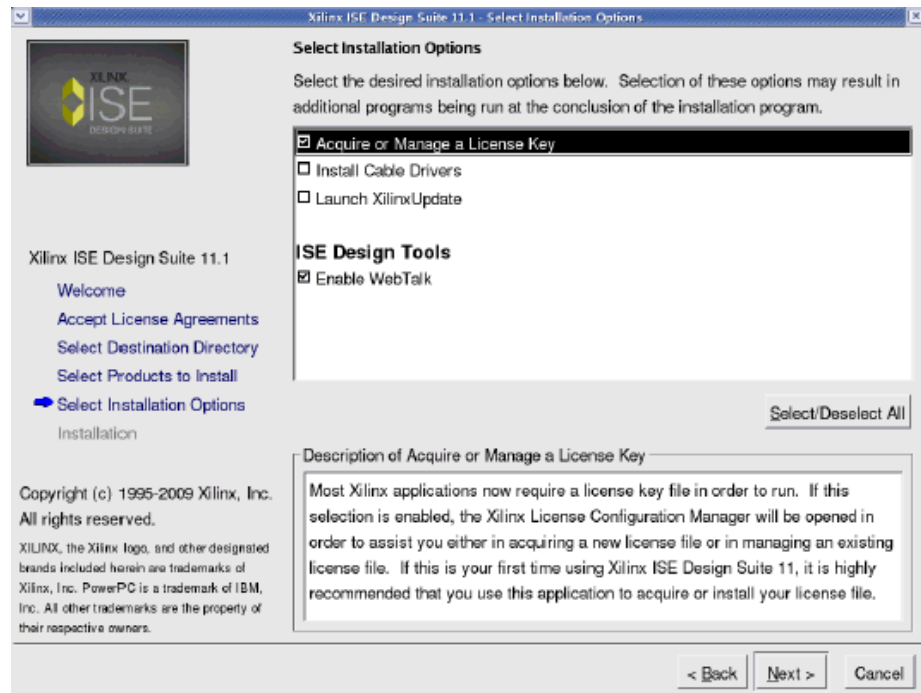


d. [DSP Tools] をオンにして [Next] をクリックします。



- e. 次の 3 つの [Installation Options] ページで任意にオプションを選択します。





メモ：環境設定の画面では、System Generator ライブラリおよび ISE ライブラリのみの LD_LIBRARY_PATH を設定できます。その他のザイリンクス ツールのライブラリ パスは後でこの LD_LIBRARY_PATH に追加できます。

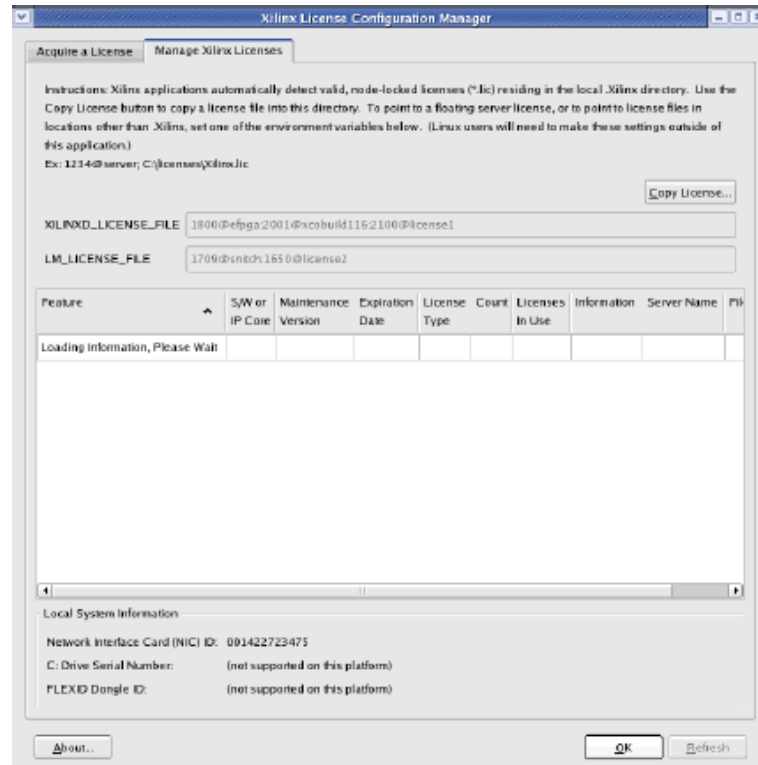
PATH には \$XILINX_DSP/common/bin を含める必要があります。このディレクトリには MATLAB で System Generator を起動するための sysgen スクリプトが含まれています。

- [Installation Options] の画面には、インストール後のスクリプトを実行しないオプションが表示されません。デフォルトはオンです。
- MATLAB Configuration GUI は起動されません。Linux のみのインストール中はディスプレイにされません。
- sysgen_startup.m スクリプトが作成され、\$XILINX_DSP/sysgen/util ディレクトリに保存されます。このスクリプトでは、MATLAB セッションに直接 System Generator が関連付けられるので、MATLAB インストールを変更する必要がありません。

f. [Options Summary] ページですべての設定オプションを確認してから [Install] をクリックします。

メモ：「Successful completion of post installation.」および「installation complete」というメッセージを含むダイアログ ボックスが表示されます。

g. [Acquire or manage License Key] オプションを選択した場合は、[Xilinx License Configuration Manager] ダイアログ ボックスが表示されます。ライセンスを設定して [OK] をクリックします。



メモ：これで、「Install has complete」というメッセージを示すダイアログ ボックスが表示されます。

2. 次のディレクトリ/ファイルは、Linux の DSP_Tools インストール ディレクトリに含まれています。ここでは、Windows でのインストール ディレクトリは示しません。

```
<install_dir>/DSP_Tools/<OS>/install_logfiles/postinstall_<OS>.log
<install_dir>/DSP_Tools/<OS>/sysgen/util/sysgen_startup.m
<install_dir>/DSP_Tools/<OS>/common/bin/sysgen
```

3. インストール後に System Generator により <install_dir>/11.1/DSP_Tools ディレクトリに生成される settings32.csh および settings32.sh (32 ビット Linux マシンの場合) または settings64.csh および settings64.sh (64 ビットの Linux マシンの場合) シェル スクリプトを編集し、\$PATH 環境変数に XILINXD_LICENSE_FILE、MATLAB、およびサードパーティ ツールを含めます。

settings32.csh または settings64.csh

```
#-----\
# Add your actual MATLAB path and any other network license servers
#-----/
setenv XILINXD_LICENSE_FILE <add your network license servers here>
setenv MATLAB_ROOT <add your MATLAB path here>
```

settings32.sh または settings64.sh

```
#-----\
# Add your actual MATLAB path and any other network license servers
#-----/
XILINXD_LICENSE_FILE=<add your network license servers here>
export XILINXD_LICENSE_FILE
MATLAB_ROOT=<add your MATLAB path here>
export MATLAB
```

ユニファイド インストーラでは、2 つのレベルの settings32[64].csh および settings32[64].sh シェル スクリプトが生成されます。1 つは共通スクリプトで、もう 1 つは各アプリケーション用のスクリプトです。インストール ディレクトリはユーザーによっては異なる可能性があります。ディレクトリ構造はデフォルトで次のような構造になっています。

```
<install_dir>/Xilinx/11.1/ISE
<install_dir>/Xilinx/11.1/DSP_Tools
```

次のディレクトリにスクリプトが含まれています。共通スクリプトのディレクトリを次に示します。

```
<install_dir>/Xilinx/11.1
```

次のディレクトリに含まれる settings32[64].csh[sh] ファイルをソースにします。

```
<install_dir>/Xilinx/11.1
```

シェル コマンド ラインに「source settings32[64].csh[sh]」と入力します。

メモ：このスクリプトにより、ユニファイド インストール中にインストールを選択したすべてのザイリンクス ツールがソースされます。

- Linux ターミナル ウィンドウ プロンプトで「which sysgen」と入力して、\$PATH 環境変数に System Generator が正しく含まれていることを確認します。次のパスが表示されます。

```
< install_dir >/Xilinx/11.1/DSP_Tools/<OS>/common/bin/sysgen
```

- 「sysgen」と入力すると、System Generator を起動できます。

メモ：MATLAB が起動され、その MATLAB セッションに System Generator が直接追加されます。MATLAB コマンド ウィンドウの上部に、「Installed System Generator dynamically」というメッセージが表示されます。これで、System Generator を実行できます。

次に、特定の状況で表示されるメッセージの一部を示します。この状況をデバッグするのに役立つ 4 つの関数も示します。

- このスクリプトの実行時に System Generator が既にインストールされている場合、次のエラー メッセージが表示されます。

```
System Generator currently found installed into matlab default path.
```

- 次の 4 つの関数を使用すると、XML ファイルを検索、検証、読み出し、またはテストできます。

xl_get_matlab_support_xmlfile

この関数では、System Generator に含まれる MATLAB サポートを判断するための共通 XML ファイルのディレクトリが読み出されます。

xl_verify_matlab_support_xmlfile

この関数では、XML ファイルが存在して読み出し可能であることが検証されます。XML ファイルが見つからない場合は、MATLAB のコンソールに次のエラー メッセージが表示されます。

```
Could not find ml_supported.xml to determine supported versions of  
MATLAB with System Generator.
```

XML ファイルを読み出すことができない場合は、MATLAB のコンソールに次のエラー メッセージが表示されます。

```
Could not read ml_supported.xml to determine supported versions of  
MATLAB with System Generator
```

xl_read_matlab_support_xmlfile

この関数では、MATLAB バージョン情報の取得のために XML ファイルを読み出して解析し、sysgen_startup.m スクリプト で使用されるエラー / 警告メッセージを提供します。

xl_test_matlab_support_xmlfile

この関数では、現在インストールされている MATLAB セッションをテストして、サポートされているバージョンと比較します。この比較結果に基づいてエラーまたは警告メッセージが表示されます。XML ファイルに情報が含まれていない場合は、MATLAB のコンソールに次のエラー メッセージが表示されます。

```
Matlab support table used by System Generator is empty!
```

XML ファイルの情報が予期するフォーマットに従っていない場合、MATLAB のコンソールに次のエラーが表示されます。

```
Input matlab support table is not well formed. It should have only  
2 columns!
```

古すぎてサポートされていないバージョンの MATLAB を使用している場合は、次のエラー メッセージが表示されます。

```
System Generator will not properly function under this version of  
MATLAB!
```

```
Error occurred while attempting to install System Generator into  
MATLAB path.
```

新しすぎるバージョンの MATLAB を使用している場合は、次のエラー メッセージが表示されます。

```
System Generator may not properly function under this version of  
MATLAB!
```

```
Installed System Generator dynamically.
```

インストール後のタスク

ハードウェア協調シミュレーション用のインストール

次に、ハードウェア協調シミュレーション用のハードウェアおよびソフトウェアのインストール手順へのリンクを示します。ハードウェア協調シミュレーションを使用しない場合は、これらの手順は必要はありません。

イーサネット ベースのハードウェア協調シミュレーション

[イーサネット ハードウェア協調シミュレーション用の ML402 プラットフォームのインストール](#)

[イーサネット ハードウェア協調シミュレーション用の ML506 プラットフォームのインストール](#)

[イーサネット ハードウェア協調シミュレーション用の ML605 プラットフォームのインストール](#)

[イーサネット ハードウェア協調シミュレーション用の Spartan-3A DSP 1800A スタータ プラットフォームのインストール](#)

[イーサネット ハードウェア協調シミュレーション用の Spartan-3A DSP 3400A 開発プラットフォームのインストール](#)

メモ：上記のプラットフォーム以外のインストール手順は、ご使用のプラットフォーム キットに付属のインストール ガイドを参照してください。

プロキシ実行ファイルのインストール (Linux ユーザー用)

Linux マシンでハードウェア協調シミュレーションを実行するには、まずプロキシ実行ファイルをインストールするシェルスクリプトを実行する必要があります。次の手順に従います。

1. Linux マシンでルート アカウントにログインします。
2. System Generator のインストール ディレクトリに含まれている bin ディレクトリに移動します。たとえば、次のように指定します。

```
cd $XILINX_DSP/sysgen/bin
```

3. シェルスクリプト install_pcap_proxy.sh を実行します。たとえば、シェルスクリプトに次を入力します。

```
./install_pcap_proxy.sh
```

JTAG ベースのハードウェア協調シミュレーション

[JTAG ハードウェア協調シミュレーション用の ML402 プラットフォームのインストール](#)

[JTAG ハードウェア協調シミュレーション用の ML605 プラットフォームのインストール](#)

[JTAG ハードウェア協調シミュレーション用の SP605 プラットフォームのインストール](#)

サードパーティ ハードウェア協調シミュレーション

ザイリンクスでは、XtremeDSP™ ソリューションの一環として、多数の代理店および OEM と協力して、さまざまな DSP プロトタイプの作成およびプラットフォームの開発を行っています。使用可能なプラットフォームの詳細は、ザイリンクス Web サイトの次のページを参照してください。

http://japan.xilinx.com/products/boards_kits/index.htm

ザイリンクス HDL ライブラリのコンパイル

System Generator デザインを ModelSim を使用してシミュレーションする場合は、IP (コア) ライブラリをコンパイルする必要があります。このセクションでは、この手順を説明します。

ModelSim SE

ModelSim SE で使用するライブラリをコンパイルするには、Compplib というザイリンクス ツールを使用します。たとえば、ModelSim SE で使用する VHDL および Verilog ライブラリ をすべてコンパイルするには、次のコマンドを使用します。

```
compplib -s mti_se -s all -l all
```

Compplib の実行方法は、『コマンド ライン ツール ユーザー ガイド』を参照してください。

System Generator キャッシュの設定

System Generator のシミュレータおよびデザイン ジェネレータでは、デザイン プロセスを繰り返し実行する場合に時間を短縮するため、ディスク キャッシュが使用されます。シミュレーションおよび生成に関連するファイルにタグを付けて保存し、シミュレーションまたは生成を次に実行するときに、これらのファイルを再生成するのではなく、キャッシュにあるファイル を呼び出すことにより、処理時間を短縮します。

サイズ

デフォルトでは、キャッシュでファイルを保存するのに 500MB までのディスク容量が使用されます。キャッシュで使用するディスク容量を指定するには、SYSGEN_CACHE_SIZE 環境変数を使用するキャッシュのサイズ (MB) に設定します。複数個の大型デザインを設計している場合は、デフォルトより大きな値を設定してください。

エントリ数

キャッシュ エントリ データベースには、決まった数のエントリが保存されます。デフォルトでは、20,000 個のエントリが保存されます。キャッシュ エントリ データベースのエントリ数を指定するには、SYSGEN_CACHE_ENTRIES 環境変数を設定します。エントリ数を小さくしすぎると、キャッシュのパフォーマンスが低下する可能性があります。複数個の大型デザインを設計している場合は、デフォルトより大きな値を設定してください。

xlCache 関数を使用すると、System Generator で使用されるさまざまなキャッシュのプロパティを管理できます。この関数の詳細は、『[System Generator ユーティリティ](#)』を参照してください。

System Generator のバージョンの表示と切り替え

複数のバージョンの System Generator をインストールできます。MATLAB コマンド **xlVersion** を使用するとインストールされているバージョンが表示され、バージョンを切り替えることができます。**xlVersion** は、モデルを最新版の System Generator 用にアップグレードする場合に便利です。

MATLAB コンソールに「**xlVersion**」と入力すると、インストールされている System Generator のバージョンが表示され、「**xlVersion <version>**」と入力すると指定したバージョンに切り替わります。たとえば、バージョン 9.2.01 と 11.1 がインストールされていて、現在選択されているバージョンが 11.1 である場合、「**xlVersion**」と入力すると次の内容が表示されます。

```
Available System Generator installations:  
Version 9.2.01 in C:/Xilinx/9.2.01/DSP_Tools/sysgen  
Version 11.1 in C:/Xilinx/11.1/DSP_Tools/sysgen  
Current version of System Generator is 11.1.
```

「xlVersion 9.2.01」と入力すると、System Generator のバージョンが 9.2.01 に切り替わります。バージョンを切り替えるのに、MATLAB を再起動する必要がある場合があります。この場合、「xlVersion 11.1」と入力すると、表示は次のように表示されます。

```
Please restart MATLAB and run xlVersion 11.1 again to switch.
```

切り替えが正常に実行されると、次のように表示されます。

```
Your System Generator has been switched. Please restart MATLAB.
```

System Generator 9.2.01 をインストールした後に 11.1 をインストールした場合、xlVersion を機能させるには 11.1 を再インストールする必要があります。

System Generator のバージョンを切り替えたら、ISE も対応するバージョンに切り替える必要があります。

リリース情報

リリース ノート 11.4

System Generator の改善点

新規デバイスのサポート

- Spartan®-6 低消費電力
- Spartan-6 XA

新規プラットフォームのサポート

System Generator では、Virtex®-6 ML605 でのイーサネット ハードウェア協調シミュレーションがサポートされるようになりました。

ザイリンクス ブロックセットの改善点

新規ブロック

DSP48 Macro 2.0

次の機能を備えた新しいブロックを使用できます。

- DSP48 Macro ブロックは、System Generator の基本ブロックになる代わりに下位の LogiCORE でサポートされるようになりました。
- 77 個の opcode 命令を追加して命令機能を拡張 (既存 72 個 + 77 個 = 149 個)
- Opmode、M Reg、P Reg 段を追加してパイプライン機能を拡張。このマクロでは 3 つのレイテンシ モード (Automatic、By Tier、および Expert) がサポートされています。Automatic および By Tier は均等なレイテンシ モデルで、Automatic では完全パイプライン モデルが提供されるのに対して、By Tier では精密な制御を実行できます。
- リセット ポートとイネーブル ポート : DSP Macro v2.0 ではグローバル sclr (System Generator の rst) および ce (System Generator の en) が提供され、XtremeDSP スライスに含まれる多種レジスタのイネーブル ポートおよびリセット ポートへのアクセスは提供されていません。
- Virtex-6、Spartan-6、および Spartan-3A DSP ファミリの前置加算器のサポート (前置加算器 D ポートを含む)
- 多種の丸め機能のサポート
- 最大 64 個の命令のサポート (以前の System Generator ブロックでは 8 個のサポート)

既存ブロックのアップデート

次のブロックは、次に示す機能でアップデートされています。

Complex Multiplier 3.1

- Spartan-6L および Spartan-6 XA FPGA のサポートの追加
- レイテンシを設定するオプションの追加

メモ：このブロックは、Complex Multiplier 3.0 ブロックに置き換わるものです。

Convolution Encoder 7.0

- Virtex-6 および Spartan-6 FPGA のサポートの追加
- パンクチャリング機能を含む、下位 LogiCORE に含まれるすべての機能に対するサポートを追加

メモ：このブロックは、Convolutional Encoder v6_1 ブロックに置き換わるものです。Convolution Encoder 7.0 ブロックでは、下位 LogiCORE で使用可能なカスタマイズおよびポート インターフェイスが提供されています。aclr 入力ポートは削除されました。

Interleave Deinterleaver 5.1

- Spartan-3A DSP のサポートの追加

メモ：このブロックは、Interleaver Deinterleaver v5_0 ブロックに置き換わるものです。Interleaver Deinterleaver 5.1 ブロックでは、下位 LogiCORE で使用可能なカスタマイズおよびポート インターフェイスが提供されています。

Reed-Solomon Decoder 7.0

- Virtex-6 および Spartan-6 FPGA のサポートの追加
- エリア最適化は、Virtex-5 ファミリでのみ選択可能になりました。
- チェック シンボルの最大数を 128 から 256 に増加

メモ：このブロックは、Reed-Solomon Decoder 6.1 ブロックに置き換わるものです。Reed-Solomon Decoder 7.0 ブロックでは、下位 LogiCORE で使用可能なカスタマイズおよびポート インターフェイスが提供されています。aclr 入力ポートは削除されました。非同期リセット入力 (reset) は削除されました。

Reed-Solomon Encoder 7.0

- Virtex-6 および Spartan-6 FPGA のサポートの追加

メモ：このブロックは、Reed-Solomon Decoder 6.1 ブロックに置き換わるものです。Reed-Solomon Decoder 7.0 ブロックでは、下位 LogiCORE で使用可能なカスタマイズおよびポート インターフェイスが提供されています。aclr 入力ポートは削除されました。非同期リセット入力 (reset) は削除されました。

Viterbi Decoder 7.0

- Virtex-6 および Spartan-6 FPGA のサポートの追加
- aclr ポートは下位 LogiCORE でサポートされていないために削除されました。
- スピード オプションが削除されました。

メモ：このブロックは、Viterbi Decoder v6_1 ブロックに置き換わるものです。Viterbi Decoder 7 ブロックでは、下位 LogiCORE で使用可能なカスタマイズおよびポート インターフェイスが提供されています。aclr 入力ポートおよびスピードオプションは削除されました。

システム要件および推奨事項

推奨ハードウェア

推奨	メモ
2.00GB の RAM	
600MB のハード ディスク容量	最低必要条件
ザイリンクス ハードウェア協調シミュレーション プラットフォーム	ハードウェア協調シミュレーション フローに必要

オペレーティング システム (OS) およびソフトウェア要件

表 3-1 :

Windows に関連する要件	メモ
Windows XP Professional 32 ビット/64 ビット オペレーティング システム SP2 (英語版および 日本語版)	System Generator for DSP を含む 32 ビット Windows 版の ISE® Design Suite 11 は、64 ビット オペレーティング システムでサポー トされています。
Windows Vista Business 32 ビット/64 ビット オペレーティング システム SP2 (英語版および 日本語版)	System Generator for DSP を含む 32 ビット Windows 版の ISE Design Suite 11 は、64 ビット オペレーティング システムでサポー トされています。
ザイリンクス ISE Design Suite 11.4	
MathWorks MATLAB® バージョン 2009a また は 2008b	MATLAB のインストール ディレクトリのパス 名には、C:\MATLAB\R2009a のように、ス ペースを含まない名前を使用する必要があります。 53 ビットより大きい信号には Fixed-Point Toolbox が必要です。
MathWorks Simulink (Fixed-Point Toolbox 含む) バージョン 2009a または 2009b	MATLAB のインストール ディレクトリのパス 名には、C:\MATLAB\R2009a のように、ス ペースを含まない名前を使用する必要があります。 53 ビットより大きい信号には Fixed-Point Toolbox が必要です。

表 3-2：

Linux に関連する要件	メモ
Red Hat Enterprise Linux WS v4.7、32 ビット/64 ビット オペレーティング システム (英語版のみ)	
Red Hat Linux 5.2、32 および 64 ビット オペレーティング システム (英語版のみ)	
SUSE Linux Enterprise v10.1、32 /64 ビット オペレーティング システム (英語版のみ)	
ザイリンクス ISE Design Suite 11.4	
MathWorks MATLAB バージョン 2009a または 2009b	<p>MATLAB のインストールディレクトリのパス名には、C:\MATLAB\R2009a のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>
MathWorks MATLAB、Simulink (Fixed-Point Toolbox 含む) バージョン 2009a または 2009b	<p>MATLAB のインストールディレクトリのパス名には、C:\MATLAB\R2009a のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>

既知の問題

System Generator の既知の問題は、次のザイリンクス Web サイトから参照できます。

<http://japan.xilinx.com/support/answers/29595.htm>

リリース ノート 11.3

System Generator の改善点

新規デバイスのサポート

- Virtex-6 低消費電力 (Virtex-6 -1L)
- Virtex-6 HXT
- Virtex-5Q

新規プラットフォームのサポート

System Generator では、Spartan-6 SP605 での JTAG ベースのハードウェア協調シミュレーションがサポートされるようになりました。

サポートされる OS

次の OS の完全サポートが追加されました。

- Windows Vista Business 32 ビット (英語版)
- Red Hat Enterprise Desktop 5.2、32 および 64 ビット
- SUSE Linux Enterprise 10、32 ビット/64 ビット

ザイリンクス ブロックセットの改善点

新規ブロック

CIC Compiler 1.3

次の機能を備えた新しいブロックを使用できます。

- Supports Virtex-6 および Spartan-6 FPGA デバイス
- 複数チャネル ソンディングのインプリメンテーションに入力および出力ストリーミング インターフェイスを追加
- ハードウェア オーバーサンプリング仕様をサンプル周期として指定するための機避を追加
- オーバー サンプリング係数を利用してリソースの使用率を最適化する機避を追加
- [Sample Period] がフォーマットに選択されたときのみ nd (new Data) 入力ポートを配置

メモ: このブロックは、C I CIC Compiler 1.2 ブロックに置き換わるものです。RATE_WE 信号はコアへのリセットとしては使用されなくなりました。コアは次の入力サンプル (シングル チャネル インプリメンテーション) または最初のチャネルへの次の入力 (複数チャネル インプリメンテーション) で新しいレートにアップデートされます。

詳細は、「[CIC Compiler 1.3](#)」を参照してください。

DDS Compiler 4.0

次の機能を備えた新しいブロックを使用できます。

- ブロックを Phase Generator または SIN/COS LUT のみとして使用するオプションを新しく追加。この機能により、Direct Digital Synthesizer を各アプリケーションの必要性に合わせてカスタマイズして構築できます。
- SFDR (Spurious Free Dynamic Range) を 120dB から 150 dB に増加

- システム レベルのパラメータ (SFDR、周波数解像度) またはハードウェア パラメータ (位相および出力幅) を使用した DDS をコンフィギュレーションするオプション
- 位相増分および位相オフセットを定数、プログラマブル、またはダイナミックとしてコンフィギュレーションするオプション

メモ：このブロックは、DDS Compiler 3.0 ブロックに置き換わるものです。DDS Compiler 4.0 は、前のバージョンに対してビット精度が高くありません。また、位相オフセットのレイテンシは、ストリーミング モードで使用しやすいように、位相増分のレイテンシに合わせて調整されます。この変更は、既存の [Programmable] モードおよび [Fixed] モードにも適用されます。

詳細は、「[DDS Compiler 4.0](#)」を参照してください。

既存ブロックのアップデート

- MULT、CMULT : LUT のインプリメンテーションにスピード最適化およびエリア最適化を利用する Multiplier LogiCORE v11.2 を使用
- Upsample ブロック : 高速クロック ドメインと低速クロック ドメインを分離することでタイミング クロージャを補助する新しいレイテンシ パターンを追加。 同じレイテンシ遅延が Upsample ブロックの入力 (つまりは低速クロック ドメイン) に追加されます。
- 次のブロックは Virtex-6 低消費電力 (ブロックの機能自体の変更なし) および Virtex-5Q デバイスをサポートするためにアップグレードされました。
 - ◆ ROM、シングル ポート RAM、デュアル ポート RAM、共有メモリで Block Memory Generator v3.3 LogiCORE が使用されるようになりました。
 - ◆ ROM、シングル ポート RAM、デュアル ポート RAM で Distributed Memory Generator v4.2 LogiCORE が使用されるようになりました
 - ◆ FIFO、From FIFO、ToFIFO で FIFO Generator v5.3 LogiCORE が使用されるようになりました。

Virtex-6 低消費電力および Virtex-5Q デバイスをサポートするその他のブロックの詳細は、「[ザイリンクス LogiCORE バージョン](#)」を参照してください。

使用できなくなった System Generator の機能

FSL (高速シンプレックス リンク) のサポート

11.3 リリースより EDK プロセッサブロック上の FSL バスに対する System Generator サポートが中止されました。ISE Design Suite 11 では FSL を続行して使用はできますが、ISE Design Suite 12 からは含まれなくなる予定です。

システム要件および推奨事項

推奨ハードウェア

推奨	メモ
2.00GB の RAM	
600MB のハード ディスク容量	最低必要条件
ザイリンクス ハードウェア協調シミュレーション プラットフォーム	ハードウェア協調シミュレーション フローに必要

オペレーティング システム (OS) およびソフトウェア要件

表 3-3 :

Windows に関連する要件	メモ
Windows XP Professional 32 ビット/64 ビット オペレーティング システム SP2 (英語版および 日本語版)	System Generator for DSP を含む 32 ビット Windows 版の ISE Design Suite 11 は、64 ビット オペレーティング システムでサポー トされています。
Windows Vista Business 32 ビット/64 ビット オペレーティング システム SP2 (英語版および 日本語版)	System Generator for DSP を含む 32 ビット Windows 版の ISE Design Suite 11 は、64 ビット オペレーティング システムでサポー トされています。
ザイリンクス ISE Design Suite 11.4	
MathWorks MATLAB バージョン 2008b または 2009a	MATLAB のインストール ディレクトリの パス名には、C:\MATLAB\R2009a のように、 スペースを含まない名前を使用する必要が あります。 53 ビットより大きい信号には Fixed-Point Toolbox が必要です。
MathWorks Simulink (Fixed-Point Toolbox 含む) バージョン 2009b または 2009a	MATLAB のインストール ディレクトリの パス名には、C:\MATLAB\R2009a のように、 スペースを含まない名前を使用する必要が あります。 53 ビットより大きい信号には Fixed-Point Toolbox が必要です。

表 3-4：

Linux に関連する要件	メモ
Red Hat Enterprise Linux WS v4.7、32 ビット/64 ビット オペレーティング システム (英語版のみ)	
Red Hat Linux 5.2、32 および 64 ビット オペレーティング システム (英語版のみ)	
SUSE Linux Enterprise v10.1、32 /64 ビット オペレーティング システム (英語版のみ)	
ザイリンクス ISE Design Suite 11.4	
MathWorks MATLAB バージョン 2008b または 2009a	<p>MATLAB のインストール ディレクトリのパス名には、C:\MATLAB\R2009a のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>
MathWorks MATLAB、Simulink (Fixed-Point Toolbox 含む) バージョン 2009b または 2009a	<p>MATLAB のインストール ディレクトリのパス名には、C:\MATLAB\R2009a のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>

既知の問題

System Generator の既知の問題は、次のザイリンクス Web サイトから参照できます。

<http://japan.xilinx.com/support/answers/29595.htm>

リリース ノート 11.2

System Generator の改善点

Virtex-6 および Spartan-6 ファミリに対する包括的な DSP デザイン プラットフォームのサポート

- ザイリンクス ブロックセットのアップデート。次が主なアップデートです。
 - ◆ DSP48 マクロ、ChipScope Pro、PicoBlaze マイクロコントローラ
 - ◆ Shared Memory ブロックセット
 - ◆ 詳細は、「[ザイリンクス LogiCORE バージョン](#)」を参照してください。
 - ◆ Complex Multiplier 3.0、FIR Compiler 5.0、および Fast Fourier Transform 7.0 で XtremeDSP™ スライスの前置乗算器を使用
- ML605 Virtex-6 FPGA プラットフォームに JTAG に基づいたハードウェア協調シミュレーションのサポートの追加
- Synplify Pro バージョン C2009.06 のサポート

ハードウェア協調シミュレーションの向上

- Linux のポイント ツー ポイント イーサネット ハードウェア協調シミュレーションのサポート

次の防衛産業向けデバイス ファミリのサポート

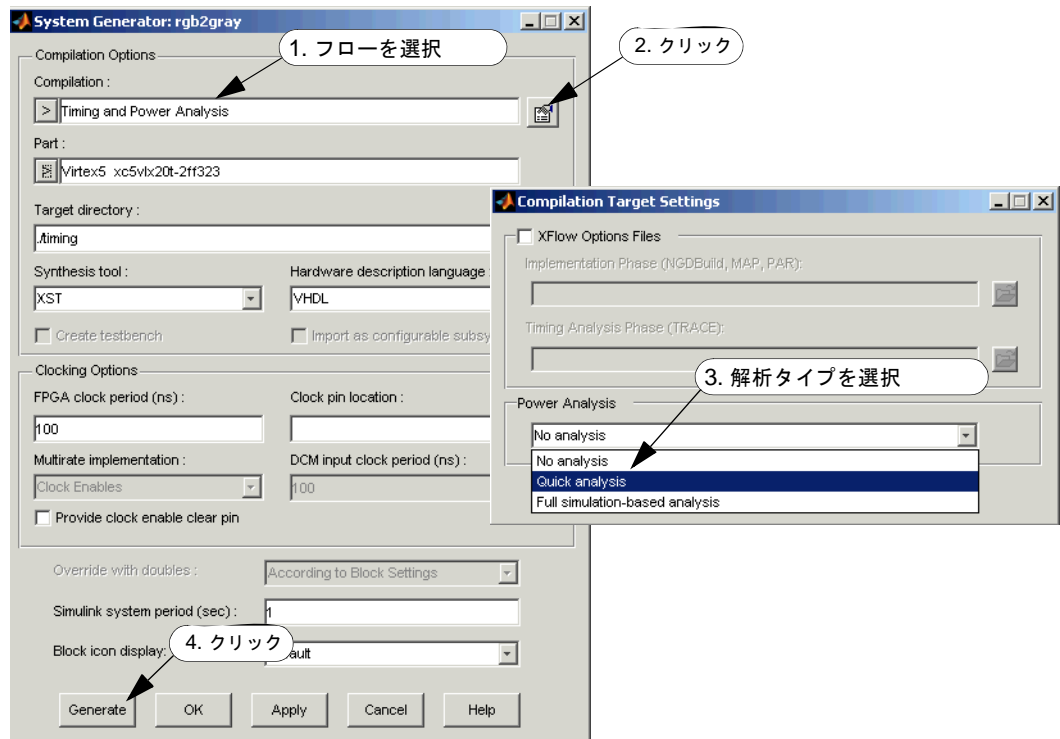
- QPro Virtex-4 Hi-Rel
- QPro Virtex-4 Rad Tolerant

MATLAB 2009a のサポート

MATLAB 2009a のサポート

System Generator と XPower の統合

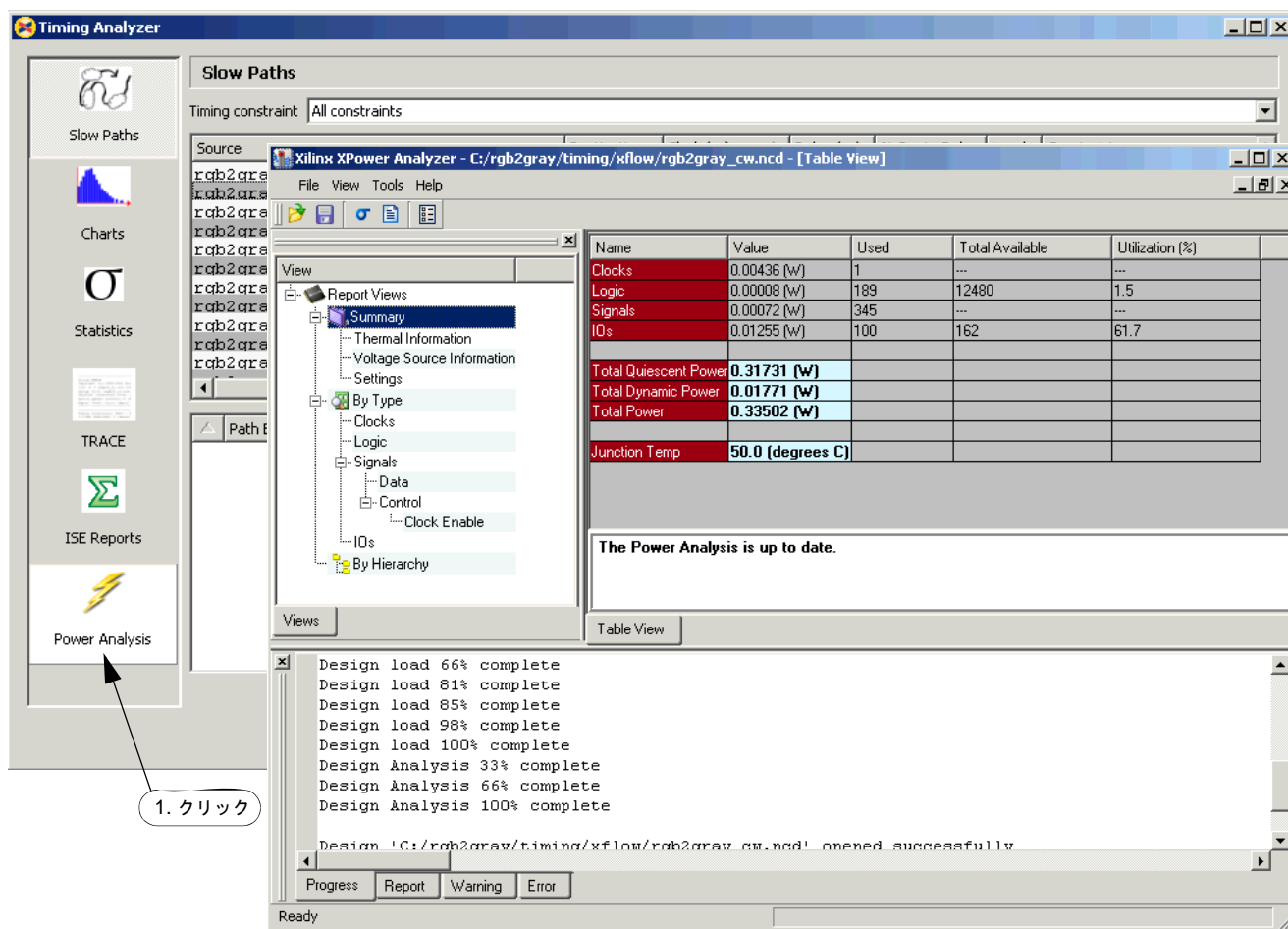
XPower が [Timing and Power Analysis] フローおよび [Bitstream] フローに統合されました。これにより、MATLAB および Simulink 環境を離れずに System Generator デザインの消費電力を解析できます。



上の図で示したように、フローを選択してからその右横にあるボタンをクリックし、[Power Analysis] オプションを選択します。

デフォルトは [No analysis] です。[Quick analysis] では高速解析を実行できますが、解析精度が低下します。[Full simulation-based analysis] では、ザイリンクス ISim シミュレータを使用してデザインに HDL シミュレーションが自動的に実行されます。時間は多少かかりますが、精度の高い解析結果を得ることができます。

解析が完了すると、[Timing Analyzer] ダイアログ ボックスが表示されます。次の図に示す [Power Analysis] ボタンをクリックすると、ザイリンクス Xpower Analyzer が起動して消費電力解析が表示されます。



ザイリンクス ブロックセットの改善点

DDS Compiler 3.0

次の機能を備えた新しいブロックを使用できます。

- Supports Virtex-6 および Spartan-6 FPGA デバイス

メモ：このブロックは、DDS Compiler 2.1 ブロックに置き換わるものです。

詳細は、「[DDS Compiler 4.0](#)」を参照してください。

Divider Generator 3.0

次の機能を備えた新しいブロックを使用できます。

- Virtex-6 および Spartan-6 FPGA デバイスのサポート

メモ：このブロックは、Divider Generator 2.0 ブロックに置き換わるものです。

詳細は、「[Divider Generator 3.0](#)」を参照してください。

Fast Fourier Transform 7.0

次の機能を備えた新しいブロックを使用できます。

- Virtex-6 および Spartan-6 FPGA デバイスのサポート
- ターゲット クロック周波数およびデータのスループットに基づいて最適なインプリメンテーションを自動的に選択するオプションの追加
- コンフィギュレーション可能な入力データのタイミング (オフセットなしまたは 3 サイクルの入力遅延)
- LUT、リソースが最適化されている 3 つの乗算器から構成される XtremeDSP スライス構造、またはパフォーマンスが最適化されている 4 つの乗算器から構成される XtremeDSP スライス構造をインプリメントするオプション
- Virtex-6 および Spartan-6 デバイスの XtremeDSP スライスに含まれる前置加算器を複雑な乗算器のインプリメンテーションに使用

メモ：このブロックは、Fast Fourier Transform 6.0 ブロックと置き換わるものです。

詳細は、「[Fast Fourier Transform 7.0](#)」を参照してください。

FIR Compiler 5.0

次の機能を備えた新しいブロックを使用できます。

- Virtex-6 および Spartan-6 FPGA デバイスのサポート
- Virtex-6 および Spartan-6 デバイスの XtremeDSP スライスに含まれる前置加算器を対称フィルタのインプリメンテーションに使用
- 固定分数レート デシメーション構造のクロック周波数およびサンプル周波数の範囲を拡張
- ハードウェア オーバーサンプリング レートを明示的に設定するオプションの追加

メモ：このブロックは、FIR Compiler 4.0 ブロックに置き換わるものです。

詳細は、「[FIR Compiler 5.0](#)」を参照してください。

システム要件および推奨事項

推奨ハードウェア

推奨	メモ
2.00GB の RAM	
600MB のハード ディスク容量	最低必要条件
ザイリンクス ハードウェア協調シミュレーション プラットフォーム	ハードウェア協調シミュレーション フローに必要

オペレーティング システム (OS) およびソフトウェア要件

表 3-5 :

Windows に関連する要件	メモ
Windows XP 32 ビット オペレーティング システム SP2 (英語版および日本語版)	
ザイリンクス ISE Design Suite 11.1.2 リリース	
MathWorks MATLAB バージョン 2008b または 2009a	
MathWorks Simulink (Fixed-Point Toolbox 含む) バージョン 2009b または 2009a	<p>MATLAB のインストール ディレクトリのパス名には、C:\MATLAB\R2009a のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>

表 3-6 :

Linux に関連する要件	メモ
Red Hat Linux 4.7、32 /64 ビット オペレーティング システム (英語版のみ)	
ザイリンクス ISE Design Suite 11.1.2 リリース	
MathWorks MATLAB バージョン 2008b または 2009a	
MathWorks MATLAB、Simulink (Fixed-Point Toolbox 含む) バージョン 2009b または 2009a	<p>MATLAB のインストール ディレクトリのパス名には、C:\MATLAB\R2009a のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>

既知の問題

System Generator の既知の問題は、次のザイリンクス Web サイトから参照できます。

<http://japan.xilinx.com/support/answers/29595.htm>

リリース ノート 11.1

System Generator の改善点

Linux のサポート

Red Hat Enterprise Linux 4 WS (32 および 64 ビット) OS がサポートされるようになりました。Linux 特定のインストール手順の詳細は、「[Linux OS への System Generator のインストール](#)」を参照してください。

MATLAB 2008b のサポート

このリリースでは MATLAB 2008b および MATLAB 2008a がサポートされています。

System Generator と Platform Studio SDK の使用

ザイリンクス Platform Studio ソフトウェア開発キット (SDK) は、ソフトウェア プラットフォーム デザインの作成用の統合開発環境 (IDE) です。SDK は Eclipse ベースの ISE で、ザイリンクス エンベデッド プロセッサの高性能 C/C++ コードを簡単に記述できます。System Generator では自動的に SDK ウォークスペースを生成し、Hello World プログラム テンプレートを提供することで SDK にアクセスできるようにします。このテンプレートには、短期間で生産性の高いコードを記述可能にするサンプル コードが含まれています。詳細は、「[Platform Studio SDK の使用](#)」を参照してください。

アップデートされたデザイン例

このリリースでは、Examples フォルダに含まれるデザインがアップデートされました。これらのデザインでは、最新のデバイスと最新の IP が使用されています。

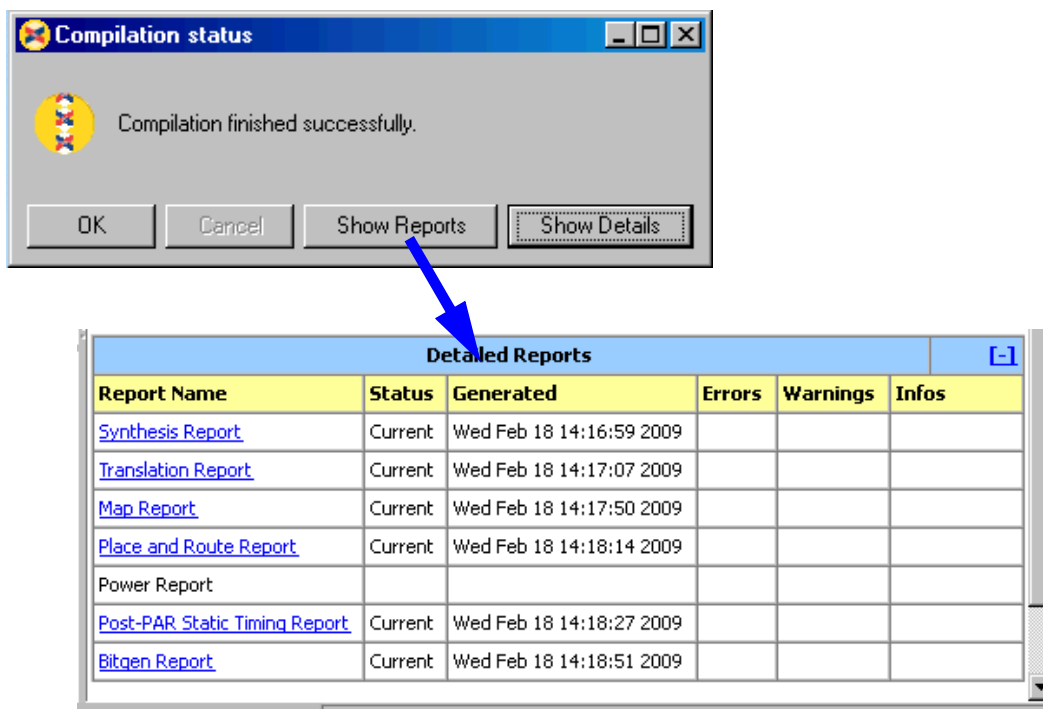
双方向ポートのサポート

System Generator のブラック ボックスで双方向の最上位ポート宣言を含む HDL がサポートされるようになりました。これらの双方向ポートは Simulink ダイアグラムでは表示されず、生成された System Generator HDL に含まれます。双方向ポートは、System Generator シミュレーション中にテキスト ファイルのデータを使用して駆動することも可能です。双方向ポートをイネーブルにする方法の詳細は、「[ブラック ボックスのコンフィギュレーション M 関数](#)」を参照してください。

System Generator の XReport の表示

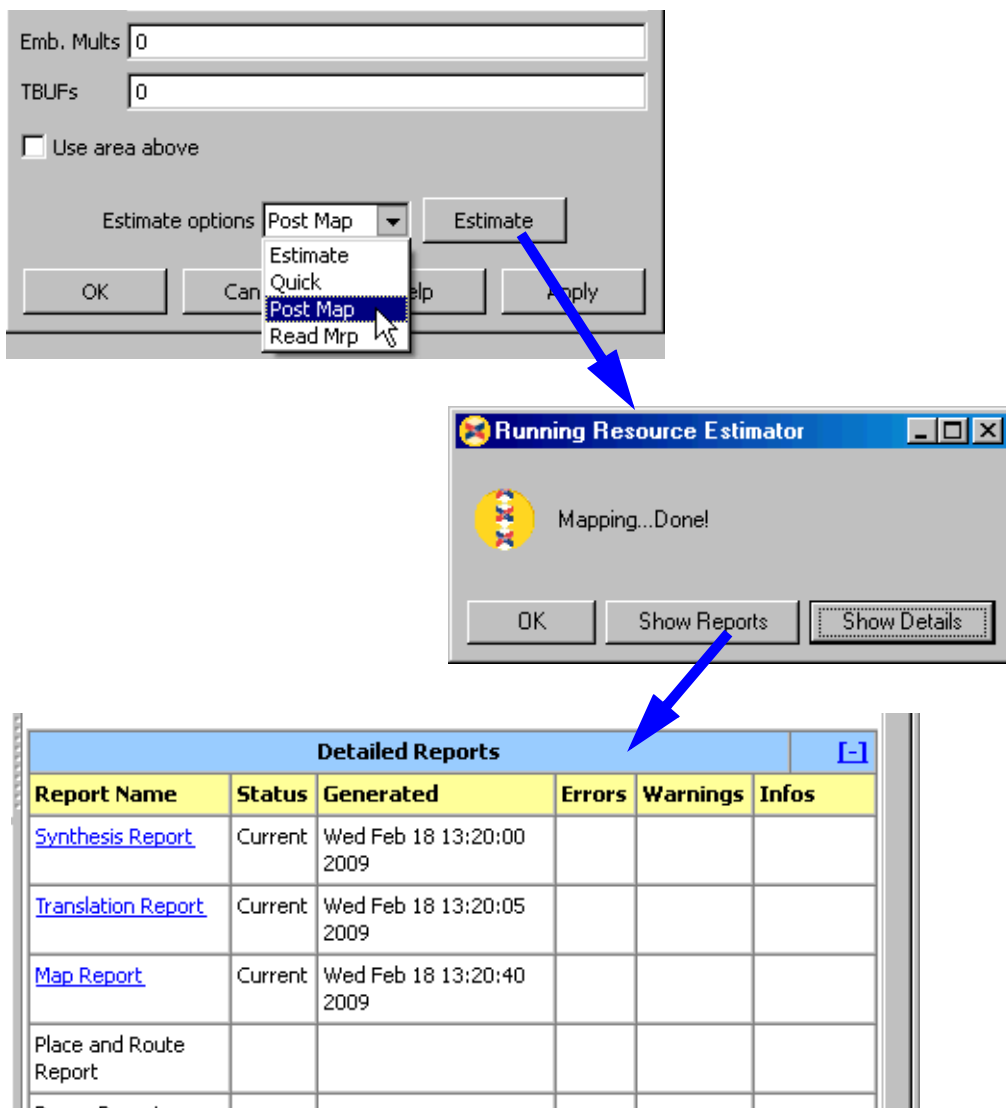
コンパイル後の XReport の表示

次に示すように、[Bitstream] または [Timing Analysis] をターゲットにしたデザインをコンパイルすると、[Compilation status] ダイアログ ボックスに新しい [Show Reports] ボタンが表示されます。このボタンをクリックすると、関連する XReport を表示できます。



Estimator ブロックからの XReport の表示

デザインに Estimator ブロックが含まれているときに [Estimate options] で [Post Map] を選択すると、[Running Resource Estimator] ダイアログ ボックスに新しい [Show Reports] ボタンが表示されます。このボタンをクリックすると、関連する XReport を表示できます。



ザイリンクス ブロックセットの改善点

Complex Multiplier 3.0

次の機能を備えた新しいブロックを使用できます。

- 2 の複素数を乗算
- すべてのオペランドおよびその結果は、符号付きの 2 の補数形式で示されます。
- オペランドの幅とその結果の幅は、パラメータ指定可能です。

詳細は、「[Complex Multiplier 3.0](#)」を参照してください。

CORDIC 4.0

次の機能を備えた新しいブロックを使用できます。

- 次の論理式タイプを持つ CORDIC (Coordinate Rotational Digital Computer) アルゴリズムをインプリメントします。
 - ◆ 直交座標 <-> 極座標の変換
 - ◆ 三角関数
 - ◆ 双曲線
 - ◆ 平方根
- 次の 2 つのアーキテクチャ コンフィギュレーションを提供
 - ◆ シングル サイクルのデータ スループットを使用した完全なパラレル コンフィギュレーション (シリコン エリアは増大)
 - ◆ マルチサイクル スループットを使用したワード シリアル インプリメンテーション (わずかなシリコン エリアを使用)
- CORDIC アルゴリズムに結果の振幅にスケール係数を使用
- CORDIC スケール係数を自動的に補正するオプション

詳細は、「[CORDIC 4.0](#)」を参照してください。

EDK Processor ブロックの改善点

- [Initial Program] : 初期プログラム ファイル (.elf) を設定可能。[Bitstream] または [Hardware Co-simulation] をコンパイル ターゲットに使用して EDK プロセッサを含むビットストリームを作成する場合、このフィールドで指定された初期プログラム ファイルがビットストリームの作成後にプロセッサのプログラム メモリに読み込まれます。
 - ◆ レジスタのリードバック : メモリ マップのインターフェイスは、通常一方向で、レジスタにはプロセッサからの読み出しまたはプロセッサへの書き込みのいずれかを実行できます。[Register Read-Back] をイネーブルにすると、書き込みと読み出しを実行できます。このオプションをオンにすると、メモリ マップへの入力が増加し、スピードの低下とエリア使用率の増加につながります。

ザイリンクス基本ブロックの改善点

次の基本構築ブロックがアップデートされました。

- Adder Subtractor 11.0
- Accumulator 11.0

- Binary Counter 11.0
- Multiplier 11.0
- RAM-Based Shift Register 11.0
- Block Memory Generator 3.1、Distributed Memory Generator 4.1、および FIFO Generator 5.1
を使用するように次のメモリ ブロックをアップデート
 - ◆ Single Port RAM
 - ◆ Dual Port RAM
 - ◆ ROM
 - ◆ FIFO
 - ◆ Shared Memory
 - ◆ To FIFO
 - ◆ From FIFO

置き換えられたザイリンクス ブロック

次のザイリンクス DSP ブロックは、置き換えられました。次の表にこれらのブロックの代わりに使用するブロックを示します。

表 3-7：置き換えられたザイリンクス DSP ブロック

置き換えられたブロック	代わりに使用するブロック
CIC Compiler 1.0	CIC Compiler 1.2
CIC Compiler 1.1	CIC Compiler 1.2
Convolutional Encoder v3_0	Convolutional Encoder v6_1
Convolutional Encoder v6_0	Convolutional Encoder v6_1
DDS Compiler v1_1	DDS Compiler 2.1
DDS v4_0	DDS Compiler 2.1
DDS v5_0	DDS Compiler 2.1
FFT v1_0	Fast Fourier Transform 6.0
FFT v3_1	Fast Fourier Transform 6.0
FFT v3_2	Fast Fourier Transform 6.0
FFT v4_1	Fast Fourier Transform 6.0
FFT v5_0	Fast Fourier Transform 6.0
FIR Compiler v1_0	FIR Compiler 4.0
FIR Compiler v2_0	FIR Compiler 4.0
FIR Compiler v3_0	FIR Compiler 4.0
FIR Compiler v3_1	FIR Compiler 4.0
FIR Compiler v3_2	FIR Compiler 4.0

表 3-7: 置き換えられたザイリンクス DSP ブロック

置き換えられたブロック	代わりに使用するブロック
Interleaver Deinterleaver v4_0	Interleaver Deinterleaver v5_0
RS Decoder v5_1	Read-Solomom Decoder 6.1
RS Decoder v6_0	Read-Solomom Decoder 6.1
RS Eecoder v5_0	Read-Solomom Encoder 6.1
RS Eecoder v6_0	Read-Solomom Encoder 6.1
Viterbi Decoder v5_0	Viterbi Decoder v6_1
Viterbi Decoder v6_0	Viterbi Decoder v6_1

システム要件および推奨事項

推奨ハードウェア

推奨	メモ
2.00GB の RAM	
600MB のハード ディスク容量	最低必要条件
ザイリンクス ハードウェア協調シミュレーション プラットフォーム	ハードウェア協調シミュレーション フローに必要

オペレーティング システム (OS) およびソフトウェア要件

表 3-8:

Windows に関連する要件	メモ
Windows XP 32 ビット オペレーティング システム SP2 (英語版および日本語版)	
ザイリンクス ISE Design Suite 11.1 リリース	
MathWorks MATLAB バージョン 2008a または 2008b	
MathWorks Simulink (Fixed-Point Toolbox 含む) バージョン 2008a または 2008b	<p>MATLAB のインストールディレクトリのパス名には、C:\MATLAB\R2008b のように、スペースを含まない名前を使用する必要があります。</p> <p>53 ビットより大きい信号には Fixed-Point Toolbox が必要です。</p>

表 3-9：

Linux に関連する要件	メモ
Red Hat Linux 4.7、32 および 64 ビット オペレーティング システム (英語版のみ)	
ザイリンクス ISE Design Suite 11.1 リリース	
MathWorks MATLAB バージョン 2008a または 2008b	
MathWorks MATLAB、Simulink (Fixed-Point Toolbox 含む) バージョン 2008b または 2008b	MATLAB のインストール ディレクトリのパス名には、C:\MATLAB\R2008b のように、スペースを含まない名前を使用する必要があります。 53 ビットより大きい信号には Fixed-Point Toolbox が必要です。

既知の問題

System Generator の既知の問題は、次のザイリンクス Web サイトから参照できます。

<http://japan.xilinx.com/support/answers/29595.htm>

リリース ノート 10.1.3

ザイリンクス DSP ブロックセットの改善点

演算子向けの DSP48 抽象化

DSP48 または従来の LUT ベースのインプリメンテーションを使用して Accumulator、AddSub、および Counter ブロックをインプリメントできるようになりました。これにより、サポートされるザイリンクス デバイス間でのデザインの移植が可能になります。

Fast Fourier Transform 6.0

次の機能を備えた新しいブロックを使用できます。

- データ幅および位相係数幅を 34 ビットに拡張
- Pipelined Streaming I/O アーキテクチャでのブロック浮動小数点のサポート

WaveScope

- WaveScope 波形を直接プリンタに直接送信し、印刷プレビュー機能を使用して表示し、印刷前に波形のフォーマットをカスタマイズできます。この機能は、[File] メニューまたはツールバーのショートカット キーからアクセスできます。

ツール フローとの互換性

System Generator 10.1.3 は、次のツールと互換性があります。

ツール	バージョン
The Mathworks 社 MATLAB および Simulink	2007b および 2008a
Mentor Graphics 社 ModelSim® SE	6.3c
Synplicity 社 Synplify Pro	8.8.0.4 (ハードウェア協調シミュレーション用にフローティング ライセンスが必要)
ザイリンクス AccelDSP	10.1.03
ザイリンクス ChipScope Pro	10.1.03
ザイリンクス EDK	10.1.03
ザイリンクス ISE	10.1.03
ザイリンクス ISE IP アップデート	10.1 IP アップデート 3
ザイリンクス ISE Simulator	10.1.03

既知の問題

System Generator の既知の問題は、次のザイリンクス Web サイトから参照できます。

<http://japan.xilinx.com/support/answers/29595.htm>

リリース ノート 10.1.2

System Generator の改善点

ハイブリッド DCM - CE サポート

10.1 リリースでは、デザインに DCM (デジタル クロック マネージャ) を自動的に含めるクロッキング オプションが追加されました。このオプションで利用できるクロック レートは 3 個までに制限されていました。

10.1.2 リリースでは、このオプションが向上されて、3 個以上のクロック レートを含むデザインをサポートできるようになりました。4 個目以降のクロック レートは、CE (クロック イネーブル) 手法を使用して自動的にサポートされます。たとえば、デザインに 6 個のクロック レートが含まれる場合、レートが高い方から 3 つのクロック レートは DCM でサポートされ、残りの 3 つは CE 手法でサポートされます。

MATLAB 2008a

MATLAB 2008a が System Generator でサポートされるようになりました。

ザイリンクス DSP ブロックセットの改善点

FIR Compiler 4.0

次の機能を備えた新しいブロックを使用できます。

- データおよび係数幅を最大 49 ビットまで拡張
- チャネライザ アプリケーションおよび転置型 MAC (Multiply and Accumulate) アーキテクチャに対する多相フィルタ バンクのサポート
- 最大 16 個の平行データ パス間で制御および係数メモリ リソースを共有可能
- 分散演算アーキテクチャでの Virtex-5 および Spartan-3A DSP のサポートを追加

FIR Compiler LogiCORE v4.0 でサポートされるすべての機能をサポート

Divider Generator 2.0

整数を除算するための除算アルゴリズムを生成する新ブロック

- 最大 54 ビット幅までのオプションのオペランド幅、同期制御、および選択可能なレイテンシ
- Radix-2 の整数除算および高基数除算アルゴリズムでの Virtex-4、Virtex-5、および Spartan-3A DSP のサポート

ツール フローとの互換性

System Generator 10.1.2 は、次のツールと互換性があります。

ツール	バージョン
The Mathworks 社 MATLAB および Simulink	2007b および 2008a
Mentor Graphics 社 ModelSim SE	6.3c
Synplicity 社 Synplify Pro®	8.8.0.4 (ハードウェア協調シミュレーション用にフローティング ライセンスが必要)
ザイリンクス AccelDSP	10.1.02
ザイリンクス ChipScope Pro	10.1.02
ザイリンクス EDK	10.1.02
ザイリンクス ISE	10.1.02
ザイリンクス ISE IP アップデート	10.1 IP アップデート 2
ザイリンクス ISE Simulator	10.1.02

既知の問題

System Generator の既知の問題は、次のザイリンクス Web サイトから参照できます。

<http://japan.xilinx.com/support/answers/29595.htm>

リリース ノート 10.1.1

System Generator の改善点

- EDK インポート フローでの UCF サポートの向上

EDK インポート フローでのユーザー制約ファイル (UCF) の処理が向上され、サイズの大きい UCF ファイルがサポートされるようになりました。インポートされた XPS プロジェクトの UCF ファイルが解析されて、EDK プロセッサ ブロックの設定に基づいた新規 UCF ファイルが生成されます。元の UCF ファイルを参照、変更することも可能で、XPS プロジェクトに再インポートできます。

- PLB デュアル クロック サポートの向上

PLB バス、MicroBlaze プロセッサ、およびその他のハードウェア ペリフェラルを異なるクロックで駆動するためにクロック ジェネレータを使用していたザイリンクス Platform Studio のプロジェクトを自動的に System Generator にインポートして、HDL ネットリストの生成およびハードウェア協調シミュレーションを実行できます。

ザイリンクス DSP ブロックセットの改善点

CIC Compiler 1.2

既存ブロックのアップデート

- CIC Compiler 1.1 と比べてシミュレーション速度が最大 4 倍向上

DDS Compiler 2.1

既存ブロックのアップデート

- 以前のバージョンと比べてコアの生成時間が最大 10 倍短縮
- 負の周波数の特定可能
- リセットがディアサートされた後に RDY 出力が 1 クロック サイクル早く High になる問題を修正

ツール フローとの互換性

System Generator 10.1.1 は、次のツールと互換性があります。

ツール	バージョン
The Mathworks 社 MATLAB および Simulink	2007a および 2007b
Mentor Graphics 社 ModelSim SE	6.3c
Synplicity 社 Synplify Pro	8.8.0.4 (ハードウェア協調シミュレーション用にフローティング ライセンスが必要)
ザイリンクス AccelDSP	10.1.01
ザイリンクス ChipScope Pro	10.1.01
ザイリンクス EDK	10.1.01
ザイリンクス ISE	10.1.01
ザイリンクス ISE IP アップデート	10.1 IP アップデート 1
ザイリンクス ISE Simulator	10.1.01

既知の問題

System Generator の既知の問題は、次のザイリンクス Web サイトから参照できます。

<http://japan.xilinx.com/support/answers/29595.htm>

リリース ノート 10.1

System Generator の改善点

System Generator と Project Navigator の統合

System Generator デザインは、Project Navigator の [New Source] ダイアログ ボックスを使用して Project Navigator 内のデザインに簡単に組み込むことができます。または、Project Navigator から System Generator デザインを起動できます。

DCM サポート

デザインに自動的に DCM を含めるオプションが追加されました。オプションの DCM は設計者により取り除かれますが、生成されたデザインではシリコン上で使用可能な DCM が利用されます。

DCM に手動で接続するために最上位のクロック ポートを表示するオプションも追加されました。

PLB46 用のデュアル非同期クロック サポート

デザインの DSP およびエンベデッド プロセス部分を異なるクロック レートで実行可能になり、柔軟性が向上されます。

ランタイム速度の向上

- シミュレーションの最初の初期化に要する速度が最大 2 倍に向上
- >Simulink Library Browser に含まれるザイリンクス ブロックセットを読み込む際の初期化の要する時間が 10 倍以上短縮

M ベースのハードウェア協調シミュレーション

ハードウェア協調シミュレーション用にコンパイルされた System Generator モデルを組み込んで、コンフィギュレーションし、MATLAB コード スクリプトで使用できるようになったことにより、MATLAB からハードウェアへの呼び出しが実行できるようになりました。

ザイリンクス DSP ブロックセットの改善点

FFT 5.0

既存のブロックに CP (Cyclic Prefix) の挿入を加えてアップデート

FIR Compiler 3.2

Virtex-II および Spartan-3A のサポートを追加

Reset Generator

サンプリング レートを下げた同期リセット信号を生成する新規ブロック

CIC Compiler 1.1

新規ブロック

ツール フローとの互換性

System Generator 10.1 は、次のツールと互換性があります。

ツール	バージョン
The Mathworks 社 MATLAB および Simulink	2007a および 2007b
Mentor Graphics 社 ModelSim SE	6.3c
Synplicity 社 Synplify Pro	8.8.0.4 (ハードウェア協調シミュレーション用にフローティング ライセンスが必要)
ザイリンクス AccelDSP	10.1
ザイリンクス ChipScope Pro	10.1
ザイリンクス EDK	10.1
ザイリンクス ISE	10.1
ザイリンクス ISE IP アップデート	10.1 IP アップデート 1
ザイリンクス ISE Simulator	10.1

既知の問題

System Generator の既知の問題は、次のザイリンクス Web サイトから参照できます。

<http://japan.xilinx.com/support/answers/29595.htm>

ザイリンクス System Generator モデルのアップデート

V2.x 以前のモデルのアップデート

v3.1 より前のバージョンのモデルをアップデートする場合は、**System Generator v7.x** を入手してモデルを v7.x にアップデートしてから **v9.1.01** にアップデートする必要があります。

v3.x、v6.x、および v7.x モデルのアップデート

このセクションでは、**System Generator v3.x**、**v6.x**、および **v7.x** モデルを **v9.1.01** で機能するようにアップデートする手順を示します。

メモ：このセクションの **v7.x** に関する手順は、**v3.x** または **v6.x** にも適用できます。

v7.x モデルを **v9.1.01** にアップデートする基本的な手順は、次のとおりです。1) **v7.1** モデルおよびモデルで使用されるユーザー定義ライブラリのバックアップ コピーを作成します。2) **xlUpdateModel** をまずライブラリに対して実行し、その後モデルに対して実行します。3) **xlUpdateModel** のレポートを参照し、指示に従います。4) モデルが **v9.1.01** で動作するかどうかを確認します。

これらの手順を、次に詳しく説明します。

1. **v7.1** モデルおよびモデルで使用されるユーザー定義ライブラリのバックアップ コピーを作成します。
2. **xlUpdateModel** を実行します。

MATLAB コンソールで **cd** コマンドを使用し、モデルを含むディレクトリに移動します。モデル名が **designName.mdl** の場合は、「**xlUpdateModel('designName')**」と入力します。

xlUpdateModel は、次のタスクを実行します。

- ◆ **v7.x** デザインの各ブロックを、同じ設定を使用した対応する **v9.1.01** ブロックにアップデートします。
- ◆ 加えた変更を説明するレポートを作成します。このレポートに、ユーザーが手動で加える必要のある変更も記述されます。

ほとんどの場合、**xlUpdateModel** で等価の **v9.1.01** モデルが生成されますが、変更が必要な構文が含まれている可能性があります。レポートを参照し、このセクションの残りの手順に従うことが重要です。

3. **xlUpdateModel** レポートを参照し、その指示に従います。

レポートに次の問題が記述されている場合は、手動の変更が必要です。

- a. **System Generator v7.x** モデルに削除されたブロックが含まれている。
次のブロックは、**System Generator** から削除されています。**CIC**、**Clear Quantization Error**、**Digital Up Converter**、**J.83 Modulator**、**Quantization Error**、**Sync**
- b. **System Generator v7.x** モデルに廃止予定のブロックが含まれている。
DDSV4.0 ブロックはまだ **System Generator** に含まれていますが、廃止予定です。
- c. **System Generator v7.x** モデルでサンプリング周期を明示的に指定するフィールドが使用されている。
サンプリング周期を明示的に指定するフィールドは、**System Generator v9.1.01** のソース以外のほとんどのブロックで削除されています。**Counter** ブロックなどのソース ブロック

では、サンプリング周期の明示的な指定が可能です。フィードバック ループを含むモデルをアップデートする場合は、**System Generator** でパスの適切なレートとタイプを判断できるようにするため、通常 `xlUpdateModel` を実行した後に **Assert** ブロックを追加する必要があります。次のメッセージは、**Assert** ブロックが必要であることを示しています。

“The data rates could not be established for the feedback paths through this block. You may need to add Assert blocks to instruct the system”

この場合、各フィードバック ループに **Assert** ブロックを追加し、このブロックでレートとタイプを指定します。

変換スクリプトでは、**v7.1** モデルの周期が明示的に指定された部分すべてに対して、モデルが変換されたことが示されます。変換後のモデルでは、ほとんどの場合 **Assert** ブロックを追加する必要はありません。**Assert** ブロックが必要かどうかを判断するには、ダイアグラムをアップデートします ([編集] → [モデルの更新] をクリック)。レートが決定しない場合は、1 つ以上の **Assert** ブロックを挿入する必要があります。

明示的なサンプリング周期の設定を使用するブロックの後に自動的に **Assert** ブロックを追加するように、変換スクリプトを設定できます。このオプションを使用するには、次のコマンドを実行します。

```
xlUpdateModel(designName, 'assert')
```

4. アップデートされたモデルを保存し、閉じます。

アップデート前のモデルのバックアップ コピーを作成していない場合は、アップデート後のモデルを別の名前で保存します。

5. モデルが **System Generator v9.1.01** で動作することを確認します。

上記の手順に従っていれば、モデルは **System Generator v9.1.01** で動作するはずです。モデルを **System Generator v9.1.01** で開き、実行してください。

例

例 1

```
>> xlUpdateModel('my_model_name');
```

現在の **MATLAB** 作業ディレクトリにある `my_model_name.mdl` ファイルをアップデートします。

例 2

```
>> xlUpdateModel('my_model_name', 'lib');
```

現在の **MATLAB** 作業ディレクトリにある `my_model_name.mdl` ファイルと関連するライブラリをアップデートします。

例 3

```
>> xlUpdateModel('my_model_name', 'assert');
```

現在の **MATLAB** 作業ディレクトリにある `my_model_name.mdl` ファイルをアップデートします。必要に応じて **Assert** ブロックを追加します。

入門

概要

この章には、System Generator の主要な機能を紹介する 6 つのレッスンが含まれています。各レッスンは、10 分程度で読み終えることのできる説明と、演習で構成されています。演習フォルダは、System Generator のインストール ディレクトリにあり、データ ファイルと手順を含みます。

コンピュータに System Generator がインストールされている場合は、これらの演習を時間のあるときに独自のペースで進めることができます。System Generator がインストールされていない場合は、ザイリンクス Web サイトの次のページから録音版 e-ラーニングにアクセスできます。

<http://japan.xilinx.com/support/training/rel/system-generator.htm>

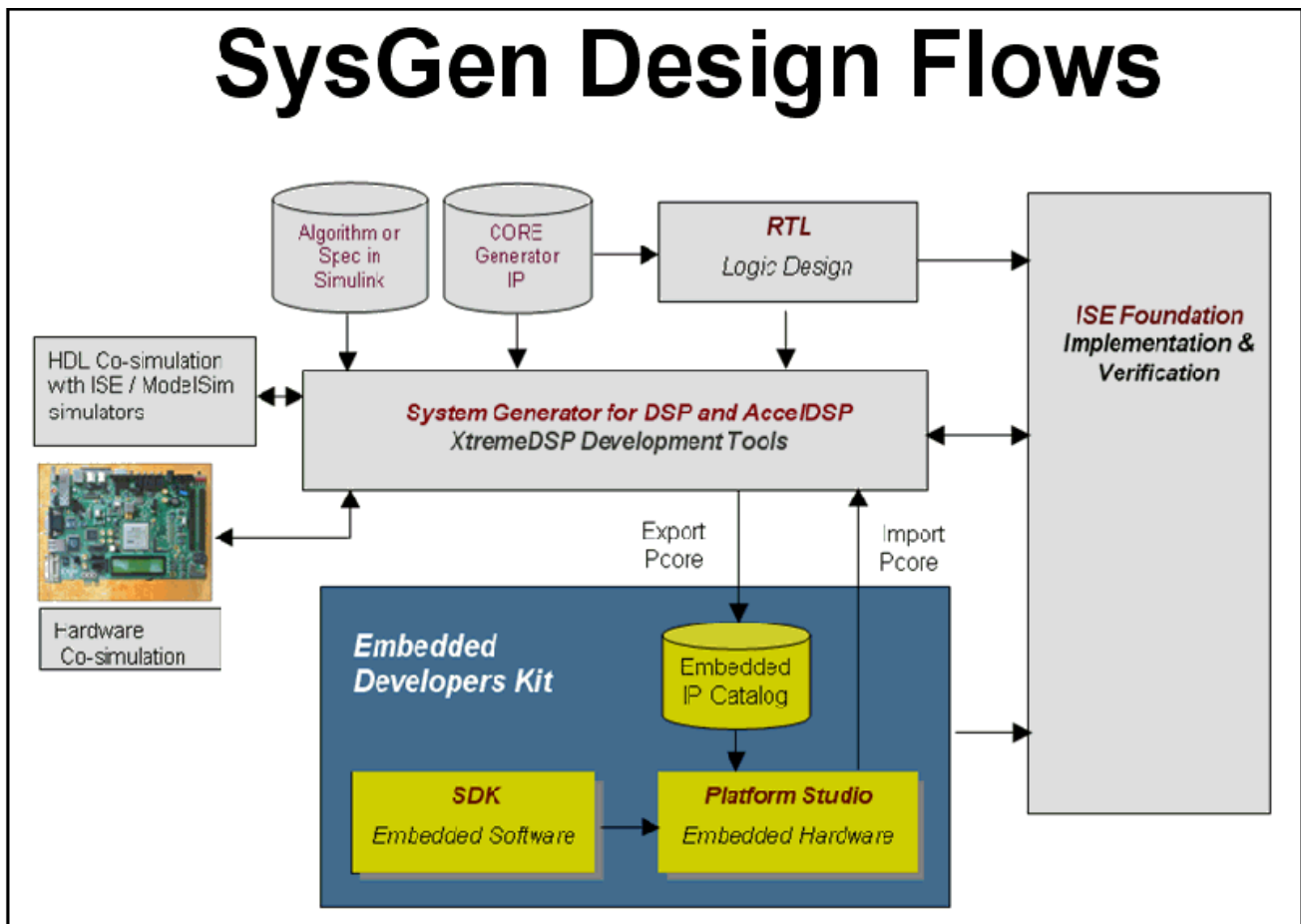
この章に含まれるレッスンは、次のとおりです。

- **レッスン 1: デザイン作成の基礎** - System Generator を使用した DSP デザインの作成およびインプリメンテーションの基礎を説明します。
- **レッスン 2: 固定小数点およびビット操作** - 固定小数点信号の個々のビットを抽出および操作する System Generator 配線ブロックの使用について説明します。
- **レッスン 3: システム制御** - 有限ステート マシン、論理制御条件、FFT に典型的なバーストデータの処理およびフィルタ処理を System Generator を使用して作成するのに適した方法を示します。
- **レッスン 4: マルチレート システム** - データのサンプリング レートを増減させることによりマルチレート システム作成する方法を示します。
- **レッスン 5: メモリの使用** - ザイリンクス ブロック RAM リソースおよび DSP ブロックの適切な使用方法を示します。
- **レッスン 6: フィルタの設計** - ザイリンクス デバイスで効率的な FIR フィルタを作成する方法、フィルタのインプリメンテーションに FIR Compiler ブロックを使用する方法、フィルタ デザインに FDATool を使用する方法を示します。

レッスン 1：デザイン作成の基礎

System Generator デザイン フロー

System Generator は、Simulink モデル ベースのデザイン手法に基づいています。ほとんどの実行仕様は、Simulink 標準ブロックセットを使用して作成します。この仕様は、浮動小数点数値精度を使用して、ハードウェアの詳細なしで作成できます。機能および基本的なデータフローを定義したら、System Generator を使用してザイリンクス デバイス用のハードウェア インプリメンテーションの詳細を指定できます。System Generator は Simulink 用のザイリンクス DSP ブロックセットを使用し、自動的に CORE Generator™ を起動して DSP 機能ブロックの高度に最適化されたネットリストを作成します。System Generator から、すべてのインプリメンテーション ツールを実行して、FPGA をプログラムするビットストリームを生成できます。また、Simulink 環境から抽出したテスト ベクタを使用して、ModelSim または ISE® Simulator で使用するテストベンチを作成することも可能です。

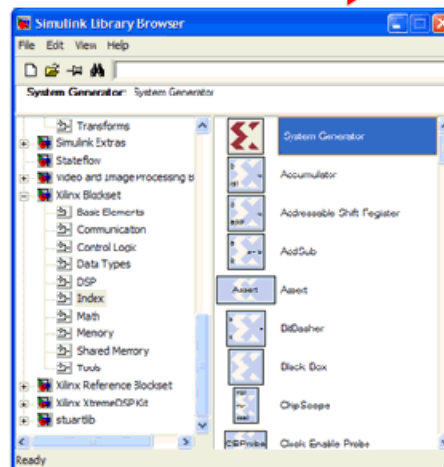


ザイリンクス DSP ブロックセット

ザイリンクス DSP ブロックセットには、Simulink Library Browser からアクセスできます。Simulink Library Browser は、標準 MATLAB ツールバーから開くことができます。ブロックセットは、検索しやすいようにサブカテゴリに分類されています。Index というサブカテゴリにはすべてのブロックが含まれているので、ブロックを使用し慣れている場合はこのサブカテゴリからすばやくアクセスできます。DSP システムの作成用に 90 個以上の DSP 機能ブロックが用意されています。

The Xilinx DSP Blockset

- Over 90 DSP building blocks available
- Blocks are accessed through the Simulink Library Browser
 - This can be launched from the MATLAB toolbar

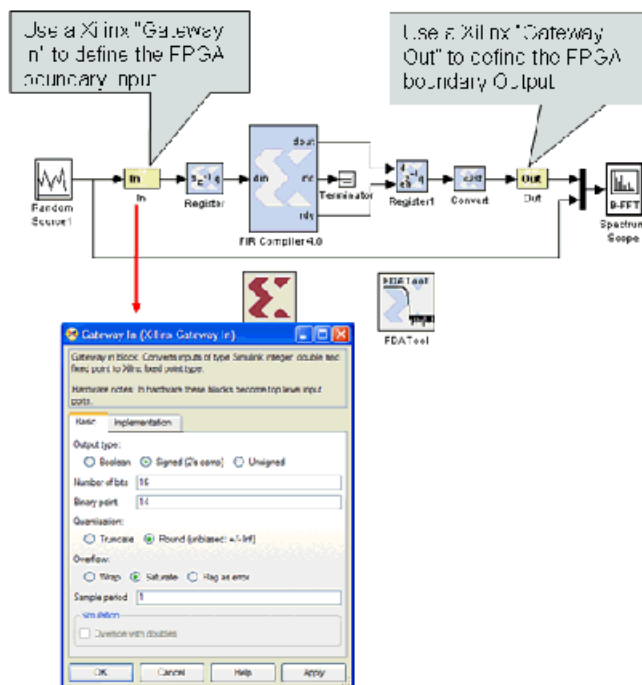


FPGA の境界の定義

System Generator では、標準 Simulink モデルを使用できます。Gateway In および Gateway Out という 2 つのブロックは、FPGA と Simulink シミュレーション モデルとの境界を定義します。Gateway In ブロックは、浮動小数点入力を固定小数点値に変換します。ブロックをダブルクリックすると、固定小数点値を指定するパラメータ ダイアログ ボックスが表示されます。

Defining the FPGA Boundary

- The FPGA boundary is defined by the Xilinx "Gateway In" and "Gateway Out" blocks
- The "Gateway In" block converts the floating-point input to fixed-point
 - Saturation and rounding modes are defined
- The "Gateway Out" block converts the FPGA outputs back to double precision

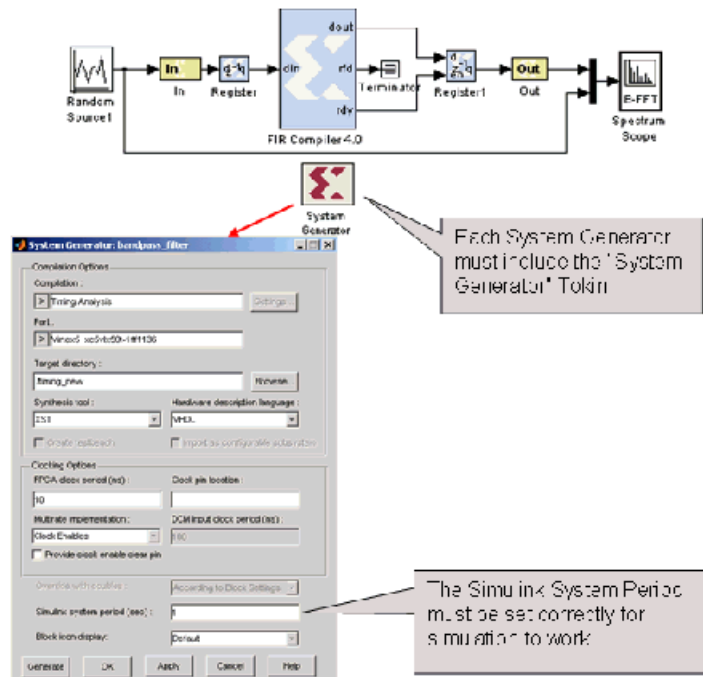


System Generator トークンの追加

System Generator ダイアグラムには、System Generator トークンを少なくとも 1 つ配置する必要があります。このトークンは接続されませんが、FPGA インプリメンテーション プロセスを駆動します。このトークンのパラメータ ダイアログ ボックスでは、ターゲット ネットリスト、デバイス、パフォーマンス ターゲット、およびシステム周期を指定できます。このトークンがない場合、System Generator でエラー メッセージが表示されます。

Adding The System Generator Token

- Every design must include a System Generator Token
- Sets the global netlisting options required for FPGA implementation
 - Target device
 - VHDL / Verilog RTL
 - Clock performance requirements
 - Downstream toolflow

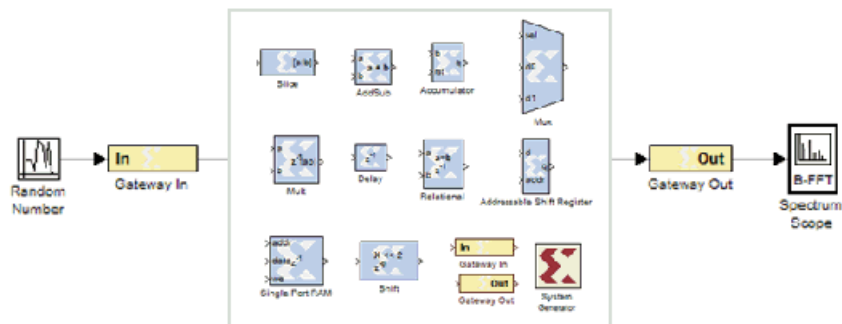


DSP デザインの作成

Gateway ブロックを使用して FPGA の境界を定義したら、ザイリンクス DSP ブロックセットのブロックを使用して DSP デザインを作成します。標準 Simulink ブロックは、Gateway In と Gateway Out ブロックの境界内では使用できません。フィルタ、FFT、FEC コア、メモリ、演算、論理、ビット単位ブロックなど、DSP デザインを構築するためのさまざまなブロックがあります。これらのブロックは、それぞれサイクル精度およびビット精度です。

Creating the DSP Design

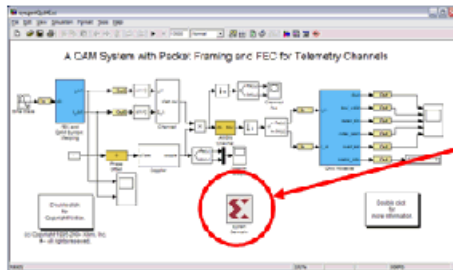
- All the blocks in the Xilinx DSP blockset are available for creating DSP designs targeting FPGAs
 - Over 90 blocks are available
 - Basic building blocks such as arithmetic and logical operators
 - System Generator IP blocks such as FIR compiler, FFT, CIC compiler and etc...
- These blocks leverage Xilinx IP generators to produce optimal results for Xilinx devices



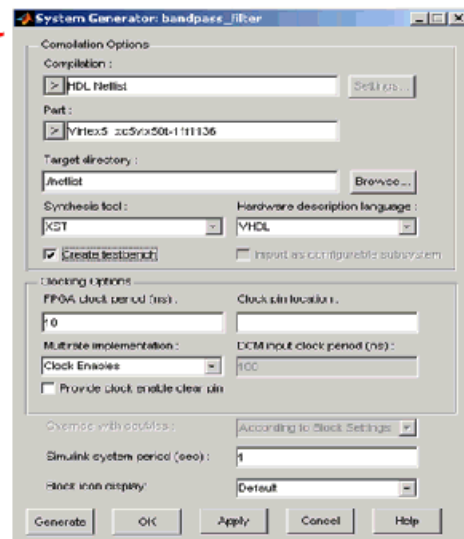
HDL コードの生成

デザインが完了したら、System Generator トークンのパラメータ ダイアログ ボックスにある [Generate] ボタンをクリックしてハードウェア インプリメンテーション ファイルを生成します。[Compilation] で [HDL Netlist] を選択すると、RTL 合成および配置配線の FPGA インプリメンテーション プロセスを、ツールのユーザー インターフェイスを使用して実行できます。[Bitstream] を選択すると、System Generator ですべてのインプリメンテーション プロセスが自動的に実行されます。

Generating the HDL Code



Once complete, double-click the System Generator token



- Select HDL Netlist as the compilation mode
- Select the target part
- Set HDL language
- Set the FPGA Clock Period
- Check Create Testbench
- Generate the HDL

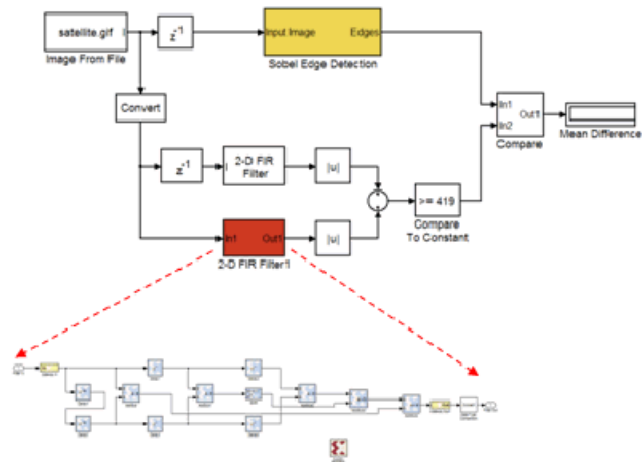
[Create testbench] をオンにすると、Simulink シミュレーションからテスト ベクタ ファイルが抽出されて保存され、ModelSim 用の HDL テストベンチ ファイルとスクリプト ファイルが生成されます。これは、生成されたハードウェアが Simulink シミュレーションと機能的に等価であるかどうかを検証するための手順です。スクリプト ファイルは、ModelSim と対話形式で使用する必要があります。

System Generator を使用したモデル ベースのデザイン

モデル ベースのデザインとは、標準 Simulink ブロックセットまたは MATLAB を使用してまず高レベルの実行仕様を作成し、ハードウェアの詳細を最小限にして機能を定義する手法のことを指します。この実行仕様は、ザイリンクス DSP ブロックセットを使用してハードウェア表現を指定する際にリファレンス モデルとして使用されます。

Model-Based Design using System Generator

- System Generator extends the model based design environment of Simulink for FPGA Design
 - First develop a high-level executable spec using standard Simulink blocks
 - Create an FPGA specific implementation using System Generator
 - Use Simulink to compare for functional and fixed-point differences



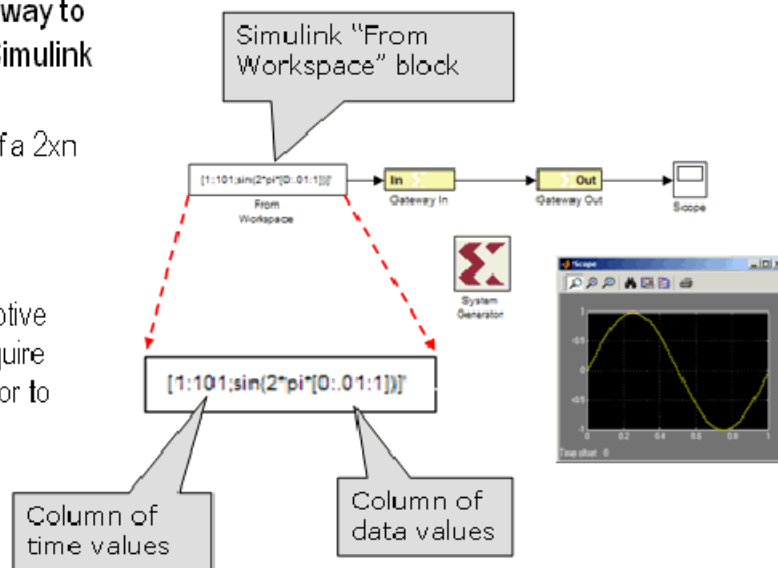
MATLAB を使用した入力ベクタの作成

Simulink は MATLAB をベースとして構築されており、入力信号の生成および出力の解析に MATLAB 言語を使用できます。Simulink の Sources ライブラリにある From Workspace ブロックおよび Sinks ライブラリにある To Workspace ブロックを使用できます。入力値は n 行 2 列の行列で指定する必要があります。ここで、 1 列目はシミュレーション時間、 2 列目は入力値です。これは、System Generator デザインの入力ベクタを生成するのによく使用される方法です。

Creating Input Vectors using MATLAB

- The Simulink “From Workspace” block provides a convenient way to generate input stimulus for Simulink designs

- Data must be in the form of a $2 \times n$ matrix
 - Column 1 = time values
 - Column 2 = data values
- Often this is a more descriptive approach and does not require sourcing a MATLAB file prior to simulation



レッスン 1 のまとめ

- Gateway In と Gateway Out ブロックを使用して、FPGA デザインと Simulink システムを分離します。
- 各シートに必ず System Generator トークンを含めます。
- Gateway ブロックの境界内では、ザイリンクス DSP ブロックセットのブロックのみが使用可能です。
- From Workspace および To Workspace ブロックを使用すると、MATLAB を使用して入力生成および出力の解析が可能です。

演習： Simulink の使用

この演習では、Simulink の基礎を学びます。Simulink ブロックセットを使用して単純なデザインを生成し、シミュレーションまで実行します。その後、サンプリング設定を変更して、出力への影響を確認します。サブシステムの作成方法も学びます。

この演習の手順は、次の場所にあります。

...<path_to_sysgen>\examples\getting_started_training\lab1\lab1.pdf

演習： System Generator 入門

この演習では、Simulink によるモデル ベースのデザイン フローで System Generator を使用してデザインを作成するための基本的な概念を示します。デザインは、単純な乗算/加算回路です。

この演習の手順とデザインは、次の場所にあります。

...<path_to_sysgen>\examples\getting_started_training\lab2\

演習手順：lab2.pdf

演習デザイン：lab3.mdl

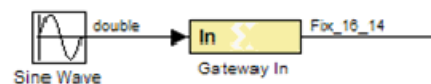
レッスン 2：固定小数点およびビット操作

固定小数点数値精度

System Generator では、正の値のみの DSP 操作に対して符号なし、負の値を含む DSP 操作に対して符号付き 2 の補数、1 ビット制御信号に対してブール値の 3 つのデータ型がサポートされています。各ブロックには、通常量子化パラメータがあります。この初期量子化は、Gateway In ブロックで定義します。

Fixed-Point Numeric Precision

- The Xilinx “Gateway In” block will convert the Simulink “double” datatype to fixed-point numeric precision
 - Fixed-point arithmetic trades off numerical precision for hardware efficiency
- System Generator supports unsigned (ufixed) and two’s complement (fixed)
 - Use “fixed” for negative numbers
 - Reduced dynamic range



UNSIGNED

Decimal	Bit Pattern
15	1111
14	1110
13	1101
12	1100
11	1011
10	1010
9	1001
8	1000
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000

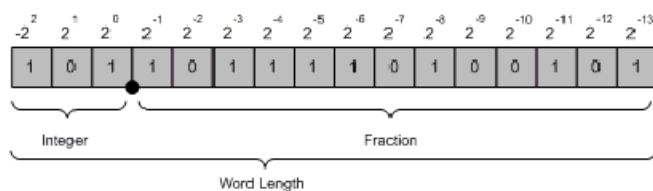
TWO'S COMPLEMENT

Decimal	Bit Pattern
7	0111
6	0110
5	0101
4	0100
3	0011
2	0010
1	0001
0	0000
-1	1111
-2	1110
-3	1101
-4	1100
-5	1011
-6	1010
-7	1001
-8	1000

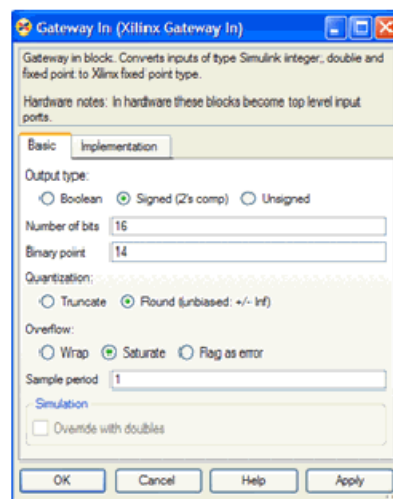
System Generator 固定小数点量子化

ザイリンクスの固定小数点データ型は、合計ビット数と 2 進小数点の位置により定義します。2 進小数点の左側のビット数は、符号なしの場合は整数ビット、符号付きの場合は整数ビットと符号ビットを足したものです。ザイリンクス FPGA では、DSP プロセッサのように、固定小数点値が定義済みの 8 ビット境界に収まる必要はありません。ロジックをビットごとに拡張し、必要な固定小数点精度を達成できます。

System Generator Fixed-Point Quantization



- System Generator supports the following fixed-point data types
 - Signed (2's Complement)
 - Required for negative numbers
 - Unsigned
 - Provides a greater range with same hardware when numbers are all positive
- To optimize the dynamic range of a number
 - Use a minimal # of integer bits to accommodate the range of possible values
 - Use a minimal # of fraction bits to accommodate acceptable precision



Fractional Bits	Fractional Values Available
1	0, 0.5
2	0, 0.25, 0.5, 0.75
3	0.125, 0.25, 0.375, 0.5, 0.625, 0.75, 0.875

オーバーフロー モードと量子化モード

System Generator では、オーバーフロー モードとして [Wrap] (最上位ビットより上のビットを切り捨て)、[Saturate] (正または負の最大値を使用)、および [Flag as error] (オーバーフローをエラーとしてレポート) がサポートされています。デフォルトでは、[Wrap] に設定されています。この設定にすると、ハードウェアのコストが最も低くなります。[Saturate] に設定すると、この操作を実行するためにロジックを追加する必要があるため、アプリケーションで必要な場合のみ使用するようにしてください。

量子化プロセスでは、LSB の [Truncate] (最下位ビットより右のビットを切り捨て) および [Round] (最も近い値に丸める) がサポートされています。[Truncate] に設定すると、ハードウェアのコストが最小限に抑えられるので、これがデフォルトです。[Round] に設定すると、追加のロジックが必要となるため、アプリケーションで必要な場合のみ使用するようにしてください。

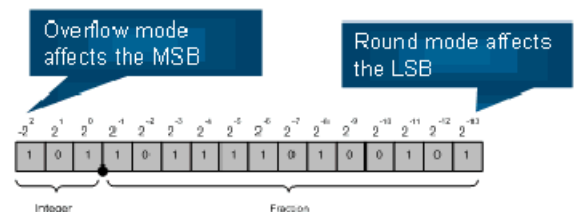
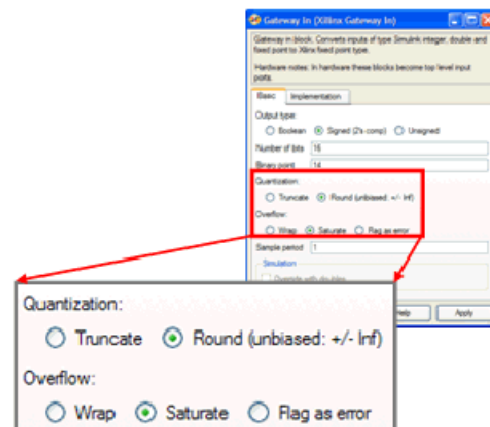
Overflow and Round Modes

Overflow

- Defines how System Generator handles the MSB of a number
 - When a number is too large to be represented by the integer bits of a number
- Wrap – the MSB's are dropped
 - Simple to implement in hardware
- Saturate – The result is set to the maximum value
 - Requires additional logic

Round Mode

- Defines how System Generator handles the LSB of a number
 - When a floating-point number is converted to fixed-point, "unnecessary" precision is lost
- Users must decide to "cut the precision off" (truncate) or to round to the nearest precision value (round)



ビット レベルの操作

実際の DSP ハードウェア システムでは、すべての操作が数学的に表現できるわけではありません。ほとんどの場合、信号は個々のビットでアクセスする必要があります。System Generator ではビット レベルの操作がサポートされており、信号の個々のビットを再解釈、結合、変換、および抽出できます。これにより、信号ビットのパディング、切り取りなどを厳密に制御して実行できます。これらのブロックでは、ハードウェア リソースは使用されません。

Bit-level Operations

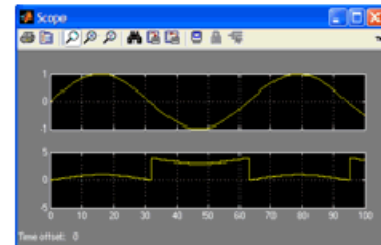
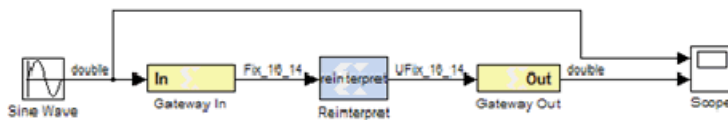
- ・ Implementing a DSP design in hardware will typically require some operations to be performed at the bit level
- ・ System Generator support blocks to perform the following bit-level operations:
 - **Reinterpret** unsigned data as signed or the converse
 - **Combine** two data buses together to form a new bus
 - **Convert** a fixed-point data type to a new fixed-point data type
 - **Extract** certain bits of data, especially when there is bit growth

Reinterpret ブロック

Reinterpret ブロックは、信号のビットを、その数値または小数点の位置にかかわらず、新しいタイプの値にします。信号のビット数は変更されず、データ型のみが再解釈されます。たとえば、符号なしの [4 1] では 1000 は 4 ですが、符号なしの [4 0] で解釈すると 1000 は 8 になります。

Reinterpret Block

- Forces the output to a new type without regard for the numerical value represented by the input
- The total number of bits in = total number of bits out
- Allows unsigned data to be reinterpreted as signed data and the converse
- Allows scaling of the data through repositioning of the binary point
 - '1000' quantized to unsigned [4 1] = 4
 - '1000' reinterpreted to unsigned [4 0] = 8

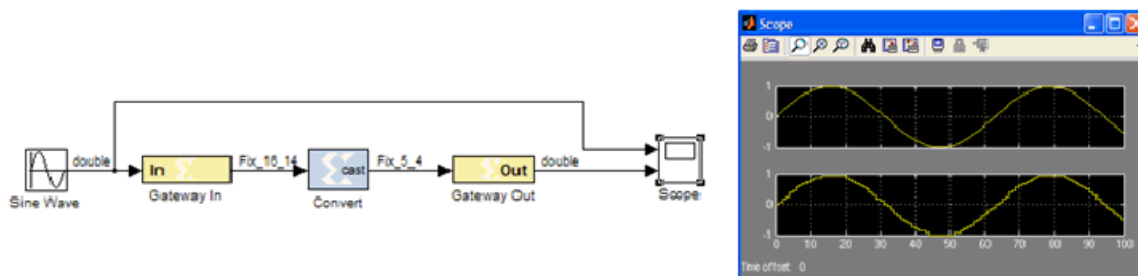


Convert ブロック

Convert ブロックは、値ではなく量子化を変更します。このブロックでは、値を表すビット数が増える場合があります。符号付きデータ型を符号なしデータ型に変換したり、符号なしのデータ型を符号付きのデータ型に変更したりできます。Convert ブロックは、乗算の後に出力の小数点以下のビットを切り捨てるためによく使用されます。

Convert Block

- converts each input sample into a number of a desired arithmetic type
 - Converts a number to a signed (twos complement), unsigned value, or Boolean
 - The total number of bits and the binary point are user specified
 - Overflow and quantization options apply to the output value

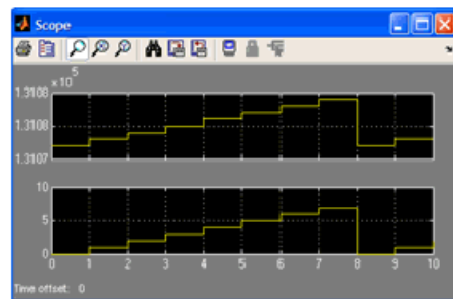
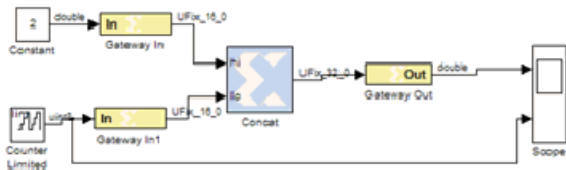


Concat ブロック

Concat ブロックは、2 つの入力をビット レベルで 1 つの出力に連結します。このブロックには、hi および lo という 2 つの入力ポートがあります。hi 入力 は出力信号の上位ビット、lo 入力は下位ビットになります。このブロックは、信号の上位または下位にゼロをパディングする場合に有益です。

Concat Block

- concatenates two inputs up to 16 bits
- All inputs must be unsigned integers
 - That is, unsigned numbers with binary points at position zero
- The Reinterpret block provides signed-to-unsigned conversion capabilities that can extend the functionality of the concat block

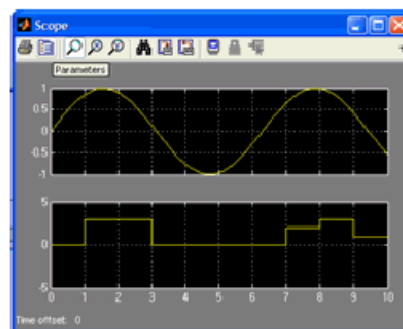
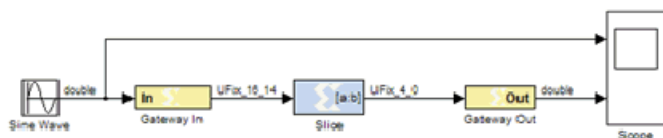


Slice ブロック

Slice ブロックは、量子化された値の個々のビットにアクセスするために使用します。このブロックでは、ビットのシーケンスを指定するのにいくつかの手法が提供されています。パラメータ化の際に入力のタイプがわかっている場合は、これらの手法を使用することで機能的な利点はありませんが、入力データの幅および 2 進小数点の位置が変化するようなデザインでは、これらの手法が有益になります。たとえば、入力の最上位ビットのみ、整数ビットのみ、または小数点以下上位 3 ビットのみを抽出するよう設定できます。

Slice Block

- Slices off a sequence of bits from the input data to create a new data value
- The output data type is unsigned, with its binary point at zero
 - One bit slices can be set to type "boolean"

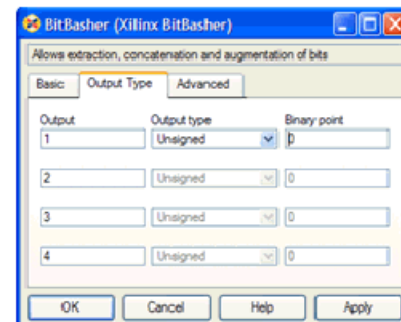
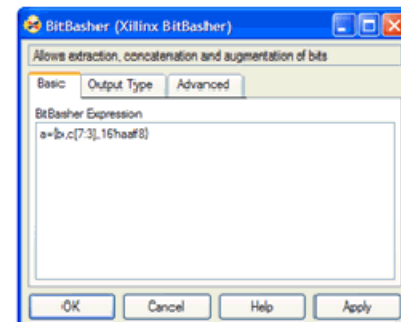
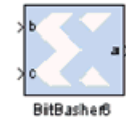


BitBasher ブロック

BitBasher ブロックを使用すると、ビット レベルでの信号処理を Verilog 構文に基づくテキストで指定できます。入力信号の連結および切り取り、定数の追加がサポートされています。このブロックでは、4 つまでの出力がサポートされており、論理式により推論されます。

BitBasher Block

- Bit manipulation and augmentation through textual specification
 - Based on Verilog syntax
 - Supports Concatenation, Slicing and Repeat operators
 - Allows augmentation with constants specified as binary, decimal, octal or hex
 - At least one of the inputs must be from input port
 - Supports up to four outputs
 - Number of Output type and Binary point fields available in Output Type tab depends on number of output equations in Basic tab



レッスン 2 のまとめ

- ブロックの出力がユーザー定義の場合、量子化とオーバーフローのオプションを設定できます。
- 量子化は、小数点以下のビット数が値の小数点以下の部分を表現するのに不十分な場合に行われます。
- オーバーフローは、データが表現可能な値の範囲外である場合に行われます。
- ビット レベル操作ブロックでは、複数のバスを 1 つのバスに連結、ビット数を変更せずにデータ型を変換、ビットを抽出、値を別のフォーマットに変換できます。
- BitBasher ブロックでは、Verilog に基づくテキスト仕様によりビットの操作および追加を実行できます。

演習：信号配線

この演習では、System Generator 信号配線ブロックを使用して、パディングおよびパディング解除ロジックの設計と検証を実行します。

この演習の手順は、次の場所にあります。

...<path_to_sysgen>\examples\getting_started_training\lab3\lab3.pdf

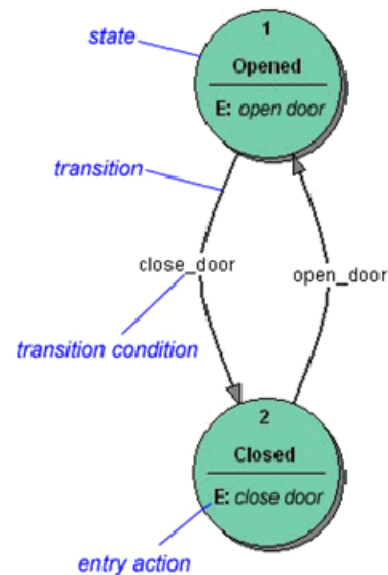
レッスン 3：システム制御

DSP システムの制御

ハードウェアで DSP システムを開発する場合、あるレベルの制御が通常必要となります。ステートに応じたビヘイビア、フィルタ係数のアップデートなどの単純な操作の実行や、ノンストリーミング FFT などのバースト データの制御用にシステム レベルの制御が必要となる場合があります。

Controlling a DSP System

- Real hardware will require some level of operational System control
- System Generator supports the following control mechanisms
 - Finite State Machines
 - Bursty data flow control
 - Reset and Clock Enable pins
 - Logical expressions

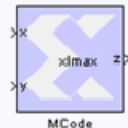


MCode ブロック

MCode ブロックは、ステートに応じた操作や分岐条件制御をインプリメントするのに MATLAB を使用できるようにします。このブロックは、FIR フィルタやマトリックスの反転など、アルゴリズム操作の記述には適しません。これらの操作には、ザイリンクス AcceIDSP™ 合成ツールを使用できます。MCode ブロックは、ステート マシンや複雑なマルチプレクサ条件をインプリメントするのに効率的で便利な方法です。System Generator で有限ステート マシンをインプリメントする場合は、MCode ブロックを使用することをお勧めします。

The MCode Block

- System Generator includes an “MCode Block” that supports using MATLAB for modeling low-level hardware control structures
 - MATLAB is translated in VHDL during hardware generation
 - The MCode block does not support algorithmic MATLAB - use AcceIDSP
- Recommended for implementing state machines in System Generator
 - A state variable is declared with the MATLAB keyword *persistent* and must be initially assigned with an *xl_state* function call
- Restrictions
 - All block inputs and outputs must be Xilinx fixed-point type
 - The block must have at least one input port and one output port



```
function q = accum(din, rst)
init = 0;
persistent s, s=xl_state(init, xLSigned,4,0);
q=s;
if rst
    s=init;
else
    s=s+din;
end
```

ザイリンクス xl_state データ型

MCode ブロックを使用してステート マシンをインプリメントする場合は、ザイリンクスが提供する xl_state という MATLAB 関数を使用して、持続型変数を初期化する必要があります。この関数では、初期条件と変数の量子化の 2 つの引数を指定します。たとえば、ステート マシンに 6 つのステートがある場合、符号なし 4 ビットの量子化が必要です。

The Xilinx “xl_state” Datatype

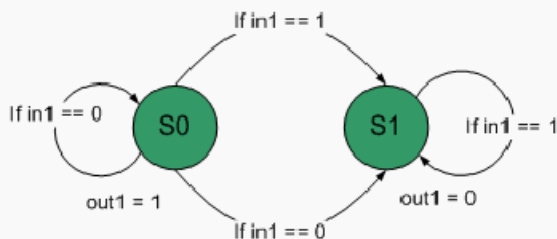
- “xl_state” is a Xilinx provided MATLAB datatype that can be used with the MCode Block to specify FSM state variables
 - Vectors specified with “xl_state” will be efficiently implemented hardware
 - `v = xl_state(init, precision)`
 - Init = initial value of state register after reset
 - Precision = a Xilinx fixed-point datatype defined for the MCode block:
 - `xlUnsigned(<word length>, <binary point>)`
 - `xlSigned(<word length>, <binary point>)`

ステート マシンの例

次の図は、単純な 2 ステート FSM を示します。これを、3 ステート以上のステート マシンに簡単に拡張できます。state という変数は `persistent` と宣言されており、`xl_state` 関数を使用して符号なしの 2 ビット値に初期化されています。入力のデコード、次のステートへの分岐、出力の割り当てには、`switch-case` 文が使用されています。

State Machine Example

- The following simple FSM example shown how the MCode block can be used to implement a finite state machine
- This example can be easily extended to include more complex behavior



```

function [out1] = fsm(in1)

persistent state,
state=xl_state(0, {xlUnsigned,2,0});

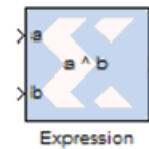
switch double(state)
case 0
    if in1==1;
        out1=0;
        state=1
    else
        out1=0;
        state=0;
    end
case 1
    if in1==0
        out1=1;
        state=0;
    else
        out1=0;
        state=1;
    end
otherwise
    state=0;
    out1=0;
end
end
  
```

Expression ブロック

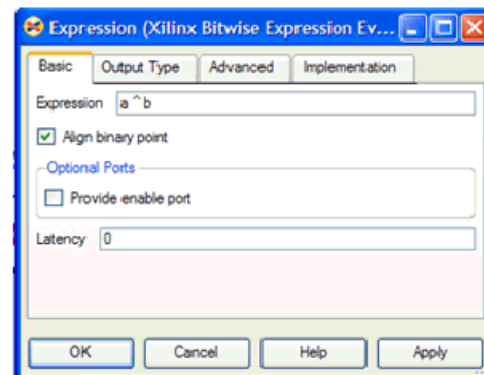
Expression ブロックは、2 つの入力信号に対してビットごとの NOT、AND、OR、および XOR を実行します。入力ワード長は、2 以上にできます。2 つの入力のワード長が異なる場合は、2 進小数点の位置が揃えられてから、エレメントごとにブール演算が実行されます。DSP システムに論理制御をインプリメントする場合は、このブロックを使用すると便利です。

The Expression Block

- The expression block provides an easy way to implement logical control using expressions
- Number of input ports is inferred from the expression
 - “a & b | c” = 3 inputs



Operator	Symbol
Precedence	{ }
NOT	~
AND	&
OR	
XOR	^

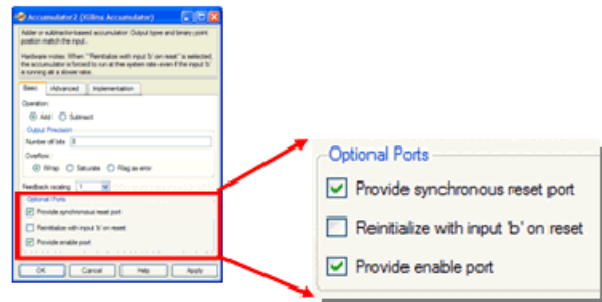
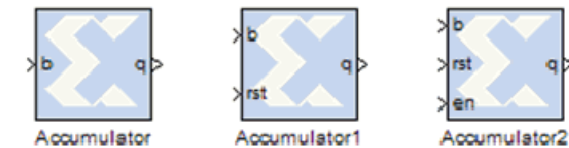


リセット ポートとイネーブル ポート

メモリまたはストレージを含む System Generator ブロックのほとんどには、リセット ポートとクロック イネーブル ポートを使用するオプションがあります。選択しない場合は、これらのポートは自動的にハードウェアのグローバル リセットとクロック イネーブルまたは DCM スキームに接続されます。System Generator ブロックでこれらのポートを選択すると、グローバル信号またはローカル信号が TRUE にアサートされたときにブロックをリセットまたはイネーブルにする条件が作成されます。DSP システムでこれらの機能を厳密に制御する必要がある場合には、これらのポートを使用してください。

Reset and Enable Ports

- System Generator blocks that include storage generally offer the option to add a reset and clock enable pin
 - The signals driving these ports must be of type "boolean"



バースト データ

ザイリンクス DSP ブロックセットに含まれるより複雑な DSP ブロックでは、バースト データが生成されます。たとえば、ノンストリーミング FFT では、入力データを処理して有効な出力データを生成するのに数クロック サイクルかかります。これらのブロックには、DSP システムで使用する必要のあるデータ フロー制御ポートが含まれます。これらのポートは、基本的なプッシュ モードのデータフロー制御を可能にします。入力に有効なデータがあることを示す **vin** ポートと、出力に有効なデータがあることを示す **vout** ポートがあります。

Bursty Data

- Often the dataflow through the system is not continuous but rather comes in “bursts”
 - Non-streaming FFT
 - Resource shared FIR Filter
- In these cases the user will need to implement dataflow control into the System Generator diagram
- System Generator blocks that require extra data processing time will include two flow control ports called “**vin**” & “**vout**”



Blocks that have valid bit modeling:

- FIR (optional)
- FFT
- Reed-Solomon Encoder/Decoder
- Viterbi Decoder
- Convolutional Encoder
- Interleaver/DeInterleaver
- CIC

レッスン 3 のまとめ

- ステート マシンや分岐条件ロジックには、**MCode** ブロックを使用します。
- ビット レベルでの論理制御をインプリメントするには、**Expression** ブロックを使用します。
- 記憶エレメントには、リセット ピンとクロック イネーブル ピンを含めることができます。これらのピンは、**System Generator** で接続できます。
- バースト データを処理するブロックには、**vin** および **vout** というデータフロー制御ピンが含まれます。

演習： システム制御

この演習では、**MCode** ブロックを使用して、1011 という 2 進数値のシーケンスを検出する単純なステート マシンを作成します。この **FSM** では、10111011 のようにシーケンスが連続する場合も検出する必要があります。

この演習のデータと手順は、次の場所にあります。

...<path_to_sysgen>\examples\getting_started_training\lab4\

演習手順：lab4.pdf

演習データ：lab4.mdl

レッスン 4：マルチレート システム

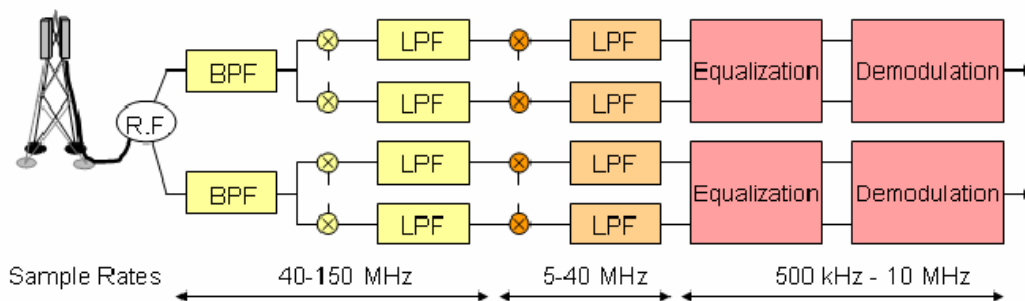
マルチレート システムの作成

次の図は、一般的な基地局レシーバを示します。電波塔には複数のアンテナがあり、エリアをセクタごとに網羅しています。ダイアグラムには、2 つのレシーバ チャネルが示されています。これらの各チャネルでは、複雑なミキシングが行われ、実チャネルと虚チャネルに分けられます。

このような DSP システムでは、イコライズおよび復調中に実行されるデジタル フィルタの前に、入力信号のサンプリング レートが下げられます。サンプリング レートを下げることで、フィルタのデザインおよびハードウェアを大幅に簡略化できます。これらのシステムは、マルチレート システムと呼ばれます。

Creating Multi-Rate Systems

- Down-sampling and up-sampling data through a DSP system is a common approach to improving hardware efficiency
 - A common example, shown below, is a wireless base station
- System Generator supports the design of multi-rate systems through rate changing blocks

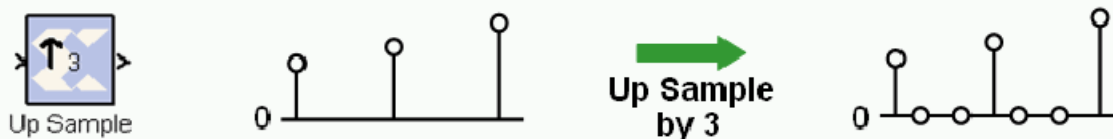


Up Sample および Down Sample ブロック

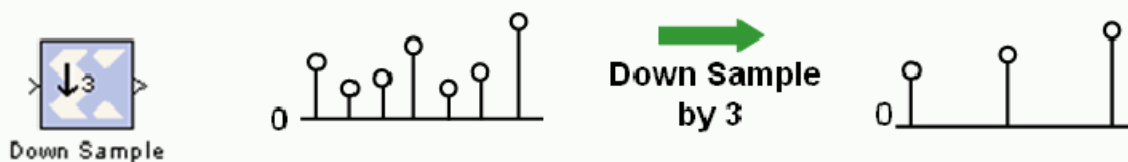
System Generator には、システムのサンプリング レートを変更する Up Sample および Down Sample ブロックが含まれています。Up Sample ブロックは、サンプリング レートを上げます。追加されたサンプルの値は、ブロック オプションに応じて、0 かその前のサンプルの値になります。Down Sample ブロックは、サンプルを破棄して必要なサンプリング レートを達成します。たとえば、3 でサンプリング レートを下げる場合、3 サンプルごとに 2 つのサンプルが破棄されます。

Up and Down Sampling Blocks

- Use the “Up Sample” and “Down Sample” blocks to change the rate of a signal in System Generator
 - The **up sample** can either replicate the same number M-1 times or insert M-1 zeros to achieve the higher sampling rate



- The **down sample** “throws away” M-1 samples to achieve the lower sampling rate







レート変更ファンクション ブロック

System Generator には、Up Sample および Down Sample ブロックに加え、特定のファンクションも実行するレート変更ファンクションブロックが含まれています。Parallel to Serial ブロックはサンプリング レートを上げ、Serial to Parallel ブロックはサンプリング レートを下げます。FIR Compiler は、リソースが共有される乗算器を使用する場合はサンプリング レートを下げ、TDM ブロックはサンプリング レートを上げます。

Rate Changing Functional Blocks

- The following functional blocks will also change the rate of a DSP system

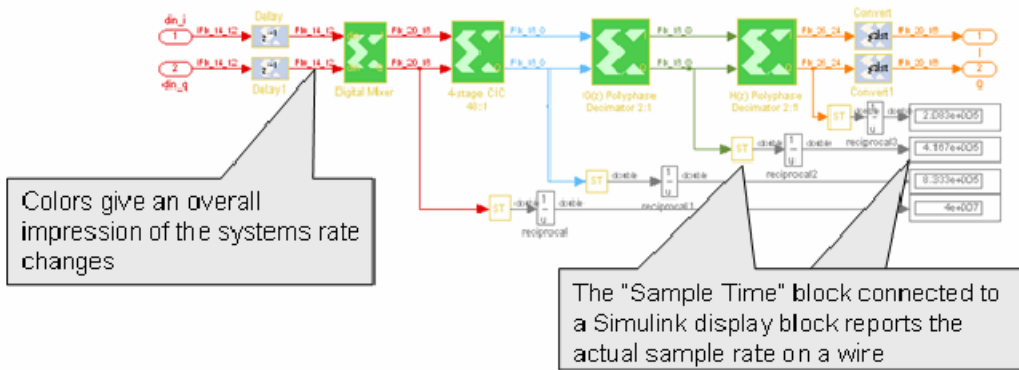
Parallel to Serial: The output rate will be M-times faster, where M is the width of the input parallel data	 Parallel to Serial
Serial to Parallel: The output rate will be M-times slower, where M is the width of the output parallel data	 Serial to Parallel
FIR and FIR Compiler: Can be used as a polyphase interpolation or decimation FIR	 FIR Compiler 4.0
The Time Division Multiplexer block multiplexes values presented at input ports into a single faster rate output. The up sample rate is determined by the number of input	 Time Division Multiplexer

Simulink でのレート変化の表示

Simulink では、異なるサンプリング時間を異なる色で表示できますが、この機能は **System Generator** ブロックで完全にサポートされています。サンプリング時間の色をイネーブルにするには、[書式] → [ポート/信号の表示] → [サンプル時間の色分け表示] をクリックします。Simulink では、この操作を行ってもモデルの色は自動的にアップデートされないため、[編集] → [モデルの更新] をクリックして表示をアップデートする必要があります。元の色に戻すには、[書式] → [ポート/信号の表示] → [サンプル時間の色分け表示] をもう一度クリックしてサンプリング時間の色をディスエーブルにします。

Viewing Rate Changes in Simulink

- Sample rates can be displayed in different colors using Simulink
 - Use the pull down menu command (**Format** → **Sample Time Colors**)
- The actual sample rate of a particular wire can be displayed using the "Sample Time" (ST) block in the Xilinx Blockset
 - Use the Simulink display block to view the output of the sample block
 - Does not add hardware to the design



ツールのデバッグ

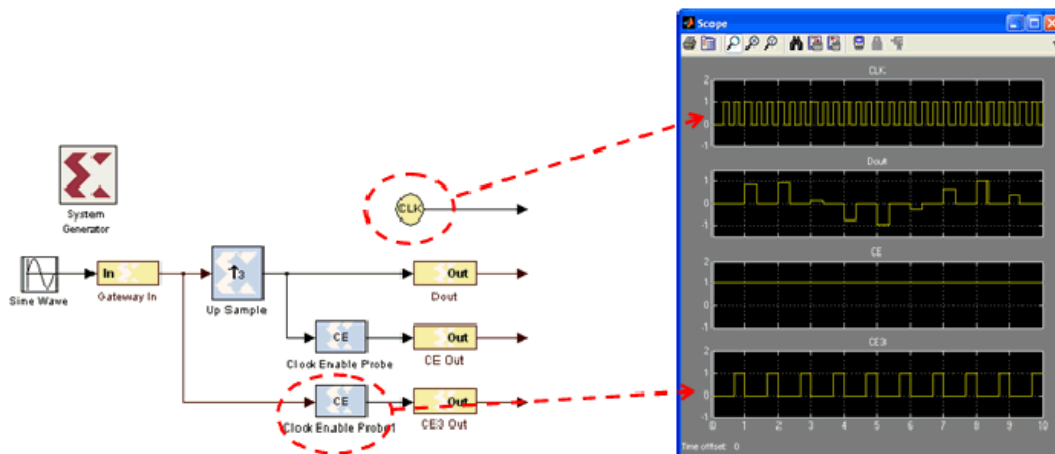
System Generator には、複雑なマルチレート システムをデバッグするため、3 つのデバッグ ユーティリティが含まれています。

Sample Time (ST) プローブを System Generator 信号に接続し、Sinks ライブラリにある Simulink の display ブロックに接続すると、接続されたネットのサンプリング時間がディスプレイに表示されます。

Clock Probe (clk) は入力には接続されず、スコープ出力のみに接続され、マスタ クロックが表示されます。これは、Clock Enable Probe と共に使用して、サンプリング レートを下げる際のさまざまな時点におけるクロック イネーブル信号の動作を表示させることができます。

Debugging Tools

- The “clk” probe and the “clock enable probe” can be used to view the behavior of the multi-rate system
 - These blocks add no logic
 - Their outputs can be connected directly to a Simulink sink block



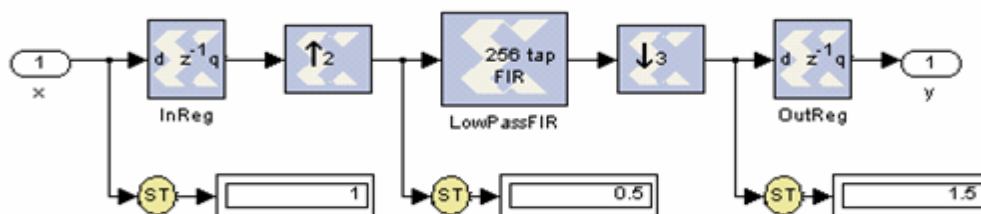
サンプリング周期に関する規則

次の図は、マルチレート システムにおいて Simulink のシステム周期を算出し、System Generator ブロックのパラメータ ダイアログ ボックスに入力する方法を示します。

不正な周期を入力した場合、サンプリング周期解析ツールにより適切なサンプリング周期が算出され、値をアップデートするようメッセージが表示されます。

Sample Period “Rules”

- The system period is the global sample period from which all other sample periods can be derived
 - The System Period is set in the System Generator token
- Every sample period in a design must be a multiple of the system period



Block Output	X	Up Sample	Down Sample
Sample Period	1	.5	1.5
Sample Period (GCD)	2/2	1/2	3/2

演習：マルチレート システム

この演習では、System Generator に含まれるレート変更ブロックの動作を調べます。レート変更ブロックには、Up Sample、Down Sample、Serial to Parallel、および Parallel to Serial などがあります。

この演習の手順とデザインは、次の場所にあります。

...<path_to_sysgen>\examples\getting_started_training\lab5\

演習手順 : lab5.pdf

演習デザイン : lab5.mdl

レッスン 5：メモリの使用

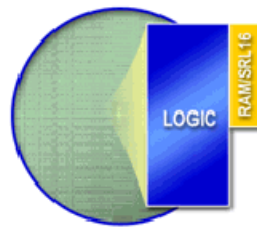
ブロック RAM と分散 RAM

ザイリックス FPGA のメモリには、ブロック RAM と分散 RAM があります。ブロック RAM は、オンチップの専用ハードウェア リソースを使用しており、エリアを最も効率的に使用した RAM のインプリメンテーションが可能です。ブロック RAM は高パフォーマンスですが、チップの決まった場所にあるため、配線遅延が多少大きくなる可能性があります。分散 RAM は、FPGA スライスのルックアップ テーブルを使用してインプリメントされるので、論理演算に使用できるルックアップ テーブルの数が減少しますが、チップの任意の位置に配置できるため、配線遅延が最小限に抑えられ、比較的高いパフォーマンスを達成できる可能性があります。分散 RAM は、小型の FIFO に適しています。

Block vs. Distributed RAM

- Xilinx devices offer two implementation options for RAMs, FIFOs and ROMs
 - Block RAM – uses dedicated on-chip RAM resources
 - More area efficient
 - Distributed RAM – uses the FPGA lookup tables
 - Higher performance
 - Subtracts from available area for logic
- System Generator RAM, FIFO and ROM blocks support either implementation

Distributed RAM/SRL16



- Very efficient, localized memory
- Minimal impact on logic routing
- Great for small FIFOs

On-chip BRAM/FIFO



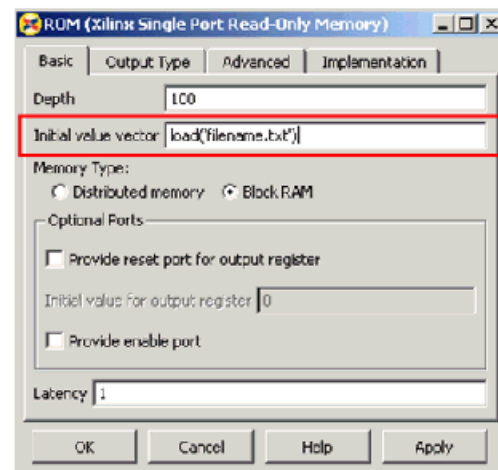
- Efficient, on-chip blocks
- Flexible + optional FIFO logic
- Ideal for mid-sized FIFOs/buffers

RAM および ROM の初期化

RAM および ROM ブロックは、RAM のワード数に一致する $1 \times n$ ベクタに初期化できます。初期値ベクタを設定するには、MATLAB を使用します。imread、auread、wavread、load などのファイル読み出しコマンドを含め、 $1 \times n$ ベクタを得られる MATLAB 文を使用できます。

Initializing RAMs and ROMs

- MATLAB statements are used to initialize the RAMs and ROMs.
 - Statement must create a $1 \times n$ vector
- Loading a text file
 - `load('filename.txt')`
- Using a MATLAB statement
 - `sin(pi*(0:15)/16)`
- Reading other file formats
 - `imread`
 - `auread`
 - `wavread`
 - `load`

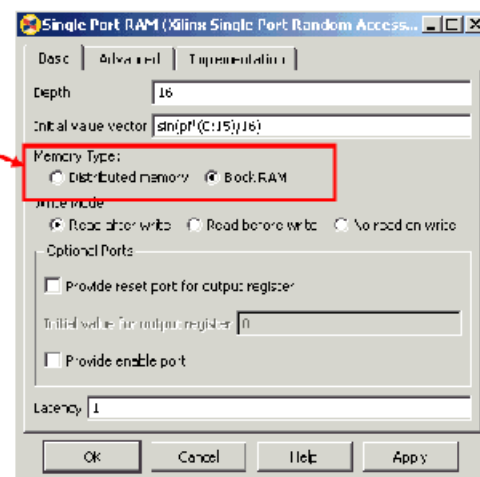


System Generator の RAM ブロック

System Generator では、シングルポートおよびデュアルポートの RAM ブロックが提供されています。64K までのワード数がサポートされています。分散 RAM またはブロック RAM を使用してインプリメントできます。System Generator によりザイリックス メモリ コンパイラが呼び出され、指定のパラメータ、ビット幅、ワード数のメモリ構造がハードウェアに作成されます。特定の Virtex® ブロックまたは分散 RAM 構造の詳細について考慮する必要はありません。シングルポートおよびデュアルポート RAM ブロックの両方で、初期化がサポートされています。RAM のアドレスポートに接続する信号は、小数点以下のビットのない符号なしの信号にする必要があります。

System Generator RAM Blocks

- System Generator offers both single and dual port RAM blocks
 - Options include selection of "Block" vs. "Distributed" implementation options
- These blocks call the Xilinx IP memory generator to create efficient RAM implementations for any depth and bit widths

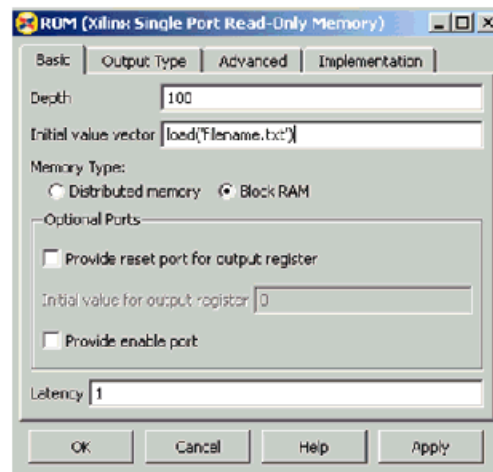


System Generator のROM ブロック

ROM ブロックは、ブロック RAM または分散 RAM を使用してインプリメントでき、MATLAB コマンドを使用して初期化します。アドレス ポートに接続する信号は、小数点以下のビットのない符号なしの信号にする必要があります。

System Generator ROM Blocks

- Offers both Block RAM vs. Distributed RAM implementation Options
- Address port must be unsigned with no fractional bits
- Depth and data widths are user configurable

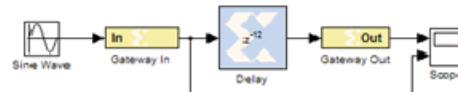


Delay ブロック

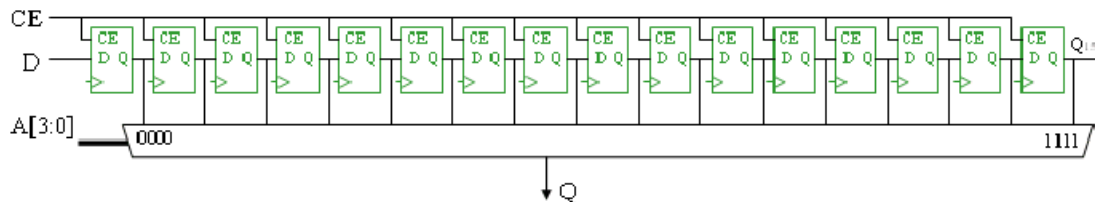
Delay ブロックは、FPGA を通過するデータフローを同期化するのに使用します。このブロックは、SRL16 というスライスルックアップテーブルで構築される効率的なシフトレジスタ構造にマップされ、レジスタを使用するよりもエリアを 85% 小さくできます。

The Delay Block

- Use the Delay block to synchronize the dataflow of signals through the design
- The implementation will be constructed from “SRL16E” primitives
 - Highly efficient use of the Virtex distributed RAM for implementing delay elements and shift registers



SRL16E Structure

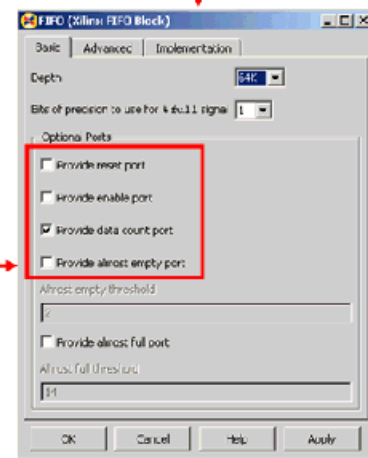
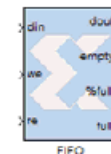


FIFO ブロック

FIFO ブロックは、ブロック RAM および分散 RAM を使用してインプリメントできます。64K までのワード数がサポートされています。empty、full、および %full の 3 つの出力フラグがあります。%full フラグは、ビット幅によって異なります。1 ビットの場合は、FIFO が 50% 未満フルか 50% 以上フルかを示します。2 ビットの場合は FIFO が 25% フルになるまでゼロで、その後 25%、50%、75% フルであることを示します。

FIFO Block

- Can be implemented in either Block or Distributed RAM
- Supports FIFO depths up to 64K
- Supports a "full" and "percentage full" output signals
 - %full Specified as number of bits
 - Unsigned, fractional
 - 1 bit shows <50% or >50% full
 - 2 bits show 25% full increments
- Includes optional control signals

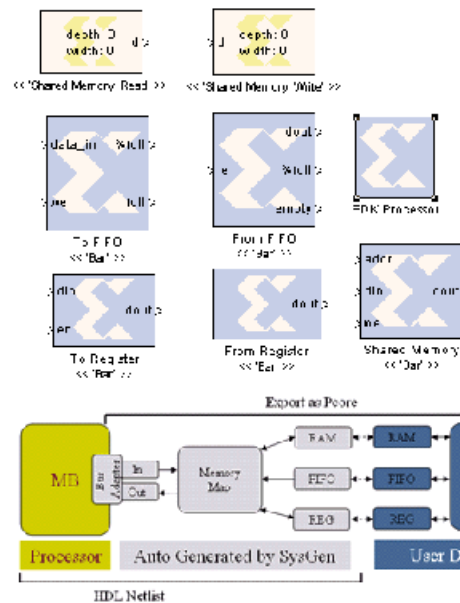


Shared Memory ブロック

System Generator では、プロセッサにカスタム ロジックを追加するための単純な抽象表現が提供されます。これは、カスタム ロジックに含まれるメモリをプロセッサのメモリ アドレス空間に簡単にマップできるようにするためです。System Generator では、System Generator ブロックセットに含まれている Shared Memory ブロックを使用してこれを実現できます。

Shared Memory Block

- Shared memories (RAMs, FIFOs, registers) allows data to be accessed from the DSP or embedded portion of the design
- System Generator provides the necessary hardware interfaces and software drivers
- Can operate across different clock domains
- Can be compiled and co-simulated in FPGA hardware
- Generates memory-map interfaces between processor and user logic
- Supports both PLB and FSL bus types
- Provides API documentation



演習：メモリの使用

この演習では、ブロック RAM または分散 RAM を使用して Arcsin などの LUT ベースの演算をインプリメントするのに、ザイリンクス ROM ブロックを使用する方法を示します。この方法では、10 ビット以下に量子化可能な入力を持つ三角法および演算ファンクションを効率的にインプリメントできます。

この演習の手順とデザインは、次の場所にあります。

...<path_to_sysgen>\examples\getting_started_training\lab6\

演習手順：lab6.pdf

演習デザイン：lab6.mdl

レッスン 6：フィルタの設計

概要

デジタル フィルタは一般的な DSP 操作であり、FPGA へのインプリメントに適しています。各クロック サイクルで結果を返すパラレル フィルタは、高パフォーマンス アプリケーションで有益です。Virtex®-5 デバイスには、640 個までのパラレル乗算器が含まれています。FIR Compiler は、一般的な FIR フィルタの作成にこれらの乗算器が最も効率的に使用されるよう設計されています。別のインプリメンテーション方法は「分散演算」と呼ばれ、乗算器を使用せずに、シフト加算手法を使用して FIR フィルタを作成します。この方法は、小型のデバイスで乗算器がほかのファンクションに割り当てられている場合に使用されます。

Introduction

- ・ Digital filters are the most common functions found in DSP systems
- ・ The following blocks are supported by System Generator for digital filtering
 - FIR Compiler block (DSP Blockset)
 - DAFIR block (DSP Blockset)
 - CIC block (Reference Designs Blockset)
- ・ The digital filtering technique will depend on several factors
 - Sample rate
 - Sample width
 - Profile of the coefficients
 - Clock rate
 - Technological resources

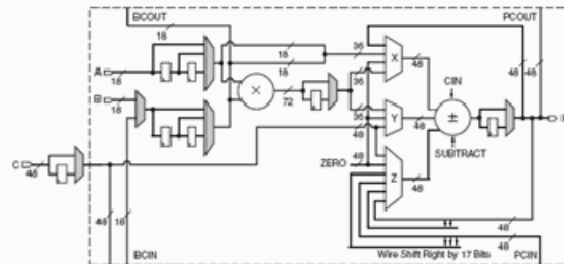
Virtex DSP48 スライス

Virtex® ファミリーには、乗算器と共に、高パフォーマンスの演算ユニットである低消費電力の DSP48 スライスが含まれています。次の図は、DSP48 の構造を詳細に示しています。DSP48 スライスは、(1) I/O レジスタ、(2) 符号付き乗算器、(3) 3 入力加減算器、(4) OPMODE マルチプレクサの 4 つのセクションから構成されています。

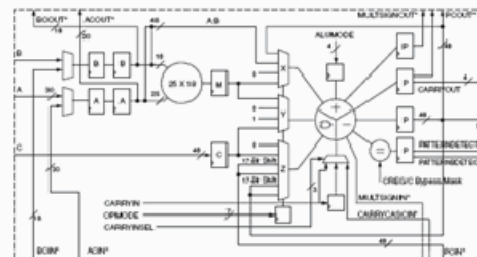
The Virtex DSP48 Math Slice

- The Virtex4 & 5 devices include up to 550 DSP48 slices
 - Performs 48 unique math operations common to DSP operations
 - Configuration set through an "opcode" input
- Efficient use of DSP48 slice is required to get high performance and efficient filters

Virtex4 DSP48 Math Slice



Virtex5 DSP48 Math Slice

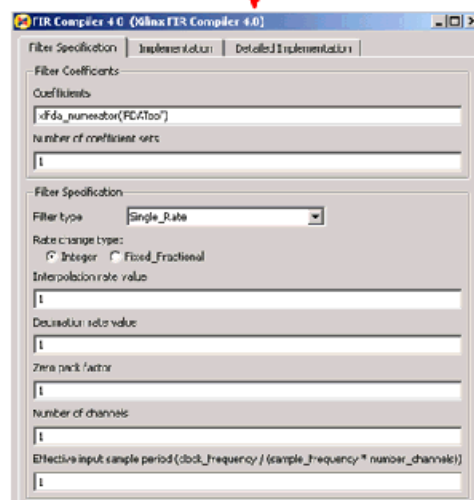


FIR Compiler ブロック

ザイリンクス FIR Compiler ブロックは、高速 MAC ベースの FIR フィルタをインプリメントします。このブロックは、入力データのストリームを受信し、フィルタのコンフィギュレーションに応じてフィルタ処理した結果を、固定の遅延で出力します。FIR Compiler では、リソースの共有またはパラレル FIR 構造、多相デシメーションおよび補間構造を生成できます。また、オーバーサンプリングもサポートされています。係数は、MATLAB コマンドを使用して指定します。

FIR Compiler Block

- The Xilinx FIR Compiler block implements a high speed MAC based using DSP48/DSP48E/DSP48A primitives or Distributed Arithmetic FIR filters
- Supports polyphase decimation, polyphase interpolation and over sampling implementations

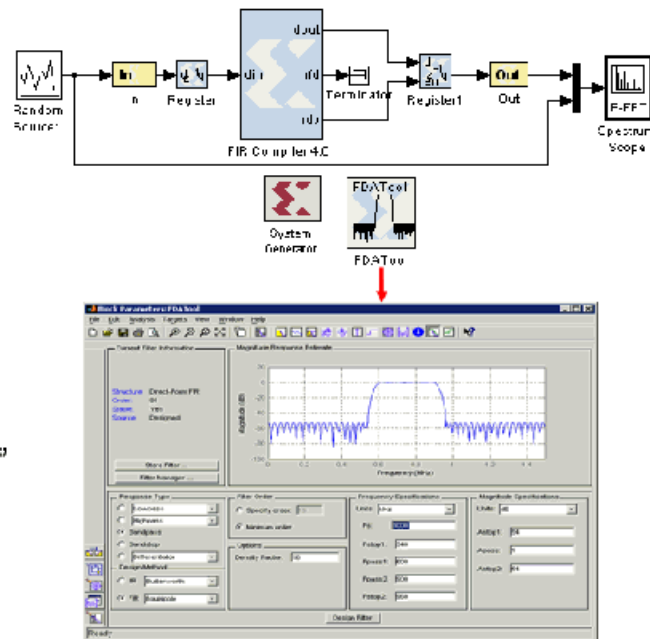


FDATool を使用した係数の作成

MathWorks 社の FDATool は、FIR Compiler ブロックの係数を生成するために使用できるグラフィカルなフィルタ設計プログラムです。ザイリンクスの FDATool ブロックは、MATLAB の Signal Processing Toolbox の一部である FDATool ソフトウェアへのインターフェイスになります。

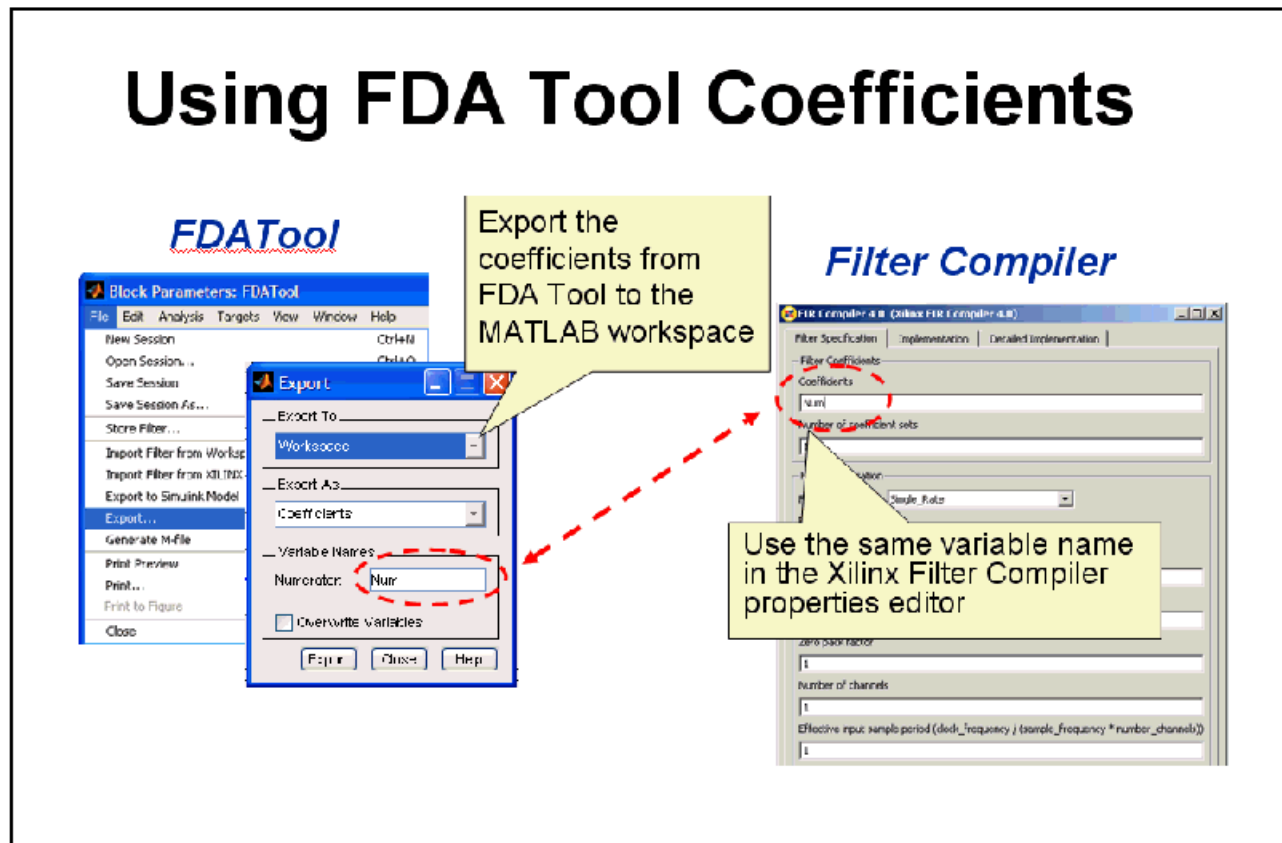
Creating Coefficients with FDATool

- Provides an interface to the FDATool software
- Enables the use of The Mathworks FDATool for creating filter coefficients graphically
- Provides a powerful means for defining digital filters with GUI
- Coefficients can be “exported” to either the MATLAB workspace or a text file



FDATool の係数の使用

適切なフィルタ応答を設計したら、[ファイル] → [エクスポート] をクリックして係数をワークスペースにエクスポートします。このワークスペース変数は、FIR Compiler のパラメータ ダイアログ ボックスで使用できます。



演習：フィルタの設計

この演習では、Filter Compiler ブロックを使用して、Virtex®-4 アーキテクチャ用に最適化されたフィルタを生成します。

この演習の手順とデザインは、次の場所にあります。

...<path_to_sysgen>\examples\getting_started_training\lab7\

演習手順：lab7.pdf

演習デザイン：lab7.mdl

その他の例とチュートリアル

System Generator のマニュアルには、System Generator の機能を示すさまざまな例が含まれています。これらの例は、<sysgen_tree>\examples ディレクトリにあり、次の表にリストされています。これらの例に加え、System Generator にも デモ ページから 実行可能なデモ モデルが含まれています。

メモ：MATLAB ヘルプ ブラウザを使用している場合は、このページから例を直接開き、実行できます。例を実行するには、リンクをクリックしてください。MATLAB のディレクトリが例のディレクトリに変更され、例のモデルが開きます。

ブラック ボックスの例

例	説明
VHDL モジュールのインポート	ブラック ボックスを使用して VHDL を System Generator デザインにインポートし、ModelSim を使用して VHDL モジュールに対して協調シミュレーションを実行する方法を示します。
複数のブラック ボックスの同時シミュレーション	1 つの ModelSim ライセンスを使用して、複数のブラック ボックスに対して同時に協調シミュレーションを実行する方法を示します。
ダイナミック ブラック ボックス	ブラック ボックスのパラメータ指定方法を示します。
Verilog モジュールのインポート	ブラック ボックスを使用して Verilog を System Generator デザインにインポートし、ModelSim を使用して Verilog モジュールに対して協調シミュレーションを実行する方法を示します。
CORE Generator モジュールのインポート	CORE Generator モジュールをブラック ボックスとしてインポートする方法を示します。

ChipScope の例

例	説明
ChipScope Pro Analyzer を使用したリアルタイム ハードウェア デバッグ	ザイリンクスのデバッグ ツールである ChipScope Pro™ を System Generator 内で接続し、使用方法を示します。System Generator フローに ChipScope Pro を統合すると、システム スピードでリアル タイムのデバッグを実行できます。

DSP の例

例	説明
DSP48 ブロック	DSP48 ブロックを、DSP48 命令を供給する Constant ブロックと共に使用方法を示します。
DSP48 Macro ブロック	DSP48 Macro ブロックを使用して複素乗算器をインプリメントする方法を示します。

例	説明
DSP48 ブロック (DSP48 と Constant ブロックを使用した 35 ビット乗算器)	DSP48 と Constant ブロックを使用して、35 X 35 ビットの乗算器を異なるサンプリング レートでインプリメントする方法を示します。サンプルごとに 1、2、4 クロック サイクルの 3 つの乗算器インプリメンテーションを示します。
DSP48 Macro ブロック (DSP48 Macro ブロックを MAC ファンクションとして使用した FIR フィルタ)	DSP48 Macro ブロックを使用して 35 X 35 乗算器をインプリメントする方法を示します。
DSP48 ブロック DSP48 ブロックを使用した FIR フィルタの例	DSP48 と Constant ブロックを使用して FIR フィルタをインプリメントする方法を示します。このデザインには、タイプ 1 およびタイプ 2 のアーキテクチャを使用したパラレル、セミパラレル、シーケンシャル FIR フィルタが含まれています。各フィルタは、DSP48 ベースの 16 タップ FIR フィルタをインプリメントします。
DSP48 設計手法 (DSP48 ベースのダイナミックシフト)	2 つの DSP48 ブロックを使用して、35 ビットの符号付きライトシフトをインプリメントする方法を示します。
DSP48 設計手法 (Virtex®-4 用の合成可能な FIR フィルタ)	System Generator を使用して、Virtex®-4 アーキテクチャに効率的にマップできる合成可能な FIR フィルタをインプリメントする方法を示します。
DSP48 Macro ブロック (DSP48 Macro ブロックを MAC ファンクションとして使用した FIR フィルタ)	DSP48 Macro ブロックを使用してシーケンシャル FIR フィルタをインプリメントする方法を示します。
MAC FIR フィルタ	MAC エンジンとデータと係数の保存にデュアルポート RAM を使用した 43 タップ FIR フィルタをインプリメントします。
複素 FIR フィルタ	System Generator と Simulink ライブラリのブロックから複素 FIR フィルタを作成する方法を示します。

MCode の例

例	説明
単純なセレクタ	入力 of 最大値を返すファンクションをインプリメントする方法を示します。
単純な数値演算	単純な数値演算をインプリメントする方法を示します。
レイテンシのある複素乗算器	レイテンシのある複素乗算器を構築する方法を示します。
シフト操作	シフト操作のインプリメント方法を示します。
MCode ブロックへパラメータを渡す	MCode ブロックにパラメータを渡す方法を示します。

例	説明
オプションの入力ポート	MCode ブロックにオプションの入力ポートをインプリメントする方法を示します。
有限ステートマシン	有限ステート マシンのインプリメント方法を示します。
パラメータ指定 アキュムレータ	パラメータ指定アキュムレータの構築方法を示します。
FIR ブロックと検証	FIR ブロックのモデリング方法とシステムの検証方法を示します。
RPN カリキュレータ	RPN カリキュレータ (スタック マシン) のモデリング方法を示します。
disp 関数の例	disp 関数を使用する方法を示します。

プロセッサの例

例	説明
MicroBlaze プロセッサ ペリフェラルの設計と エクスポート	MicroBlaze™ プロセッサ用のペリフェラル (pcore) の設計方法と、 System Generator から Xilinx Platform Studio (EDK) にエクスポートする方法を示します。 RGB からグレースケールへの色空間変換を作成し、 EDK へのエクスポートを使用して pcore へ生成します。
チュートリアル： MicroBlaze プロセッサ システムの設計とシ ミュレーション	XPS を使用して作成した MicroBlaze プロセッサを System Generator にインポートする方法を示します。 MicroBlaze プロセッサへのコプロセッサとして、 DSP48 ブロックが使用されています。
PicoBlaze マイクロコ ントローラ アプリケー ションの設計	PicoBlaze™ プログラムを System Generator にインプリメントする方法を示します。この例では、割り込み中の DDS (Direct Digital Synthesizer) の出力周波数を変更するよう PicoBlaze をプログラムします。

共有メモリの例

例	説明
さまざまなモデル間のシミュレーション	Simulink モデル間で通信する共有メモリを示します。
ホスト PC 共有メモリのアクセス	共有メモリと通信するための Developer Studio プロジェクト
ハードウェア協調シミュレーションを使用した高速ビデオ処理	高速協調シミュレーション バッファ インターフェイスについて説明し、このインターフェイスを 5X5 フィルタ カーネルを使用したビデオ ストリームのリアルタイム処理に使用する例を示します。
高速 I/O バッファ処理	ハードウェア協調シミュレーション用の高速共有メモリ I/O バッファ インターフェイスを示します。

例	説明
SharedMemory (mex 関数インターフェイス)	共有メモリへのインターフェイスとして mex 関数を使用する方法を示します。
複数クロックのサイクル単位アイランドの生成	2 つの非同期クロックを使用した例を示します。
共有メモリ、FIFO への送信、レジスタへの送信、レジスタからの送信	共有メモリ、FIFO、およびレジスタを使用して情報を転送する方法を示します。
ハードウェア協調シミュレーションを使用したフレームベースのシミュレーション	FPGA ハードウェア協調シミュレーションを使用したシミュレーションを高速化するためにフレーム ベースまたはベクタベースの転送を使用する方法を示します。

タイミング解析の例

例	説明
チュートリアル：タイミング解析の使用	System Generator のタイミング解析ツールを使用して、System Generator デザインのタイミング要件を満たす方法を示します。また、デザインがタイミングを満たさない場合の手法も説明します。

その他の例

例	説明
System Generator デザインの大型システムへのインポート	System Generator から VHDL ネットリストを取り出して合成し、大型デザインに組み込む方法を示します。また、System Generator で作成した VHDL をシステム全体のシミュレーション モデルに組み込む方法も示します。
コンフィギャブル サブシステムと System Generator	シミュレーションおよび生成用にコンフィギャブル サブシステムを使用する方法を示します。
積分器	積分器を使用してエラー解析機能を示します。
ブロック RAM ベースのステート マシン	リファレンス ライブラリの Mealy State Machine ブロックの使用を示します。

System Generator デモ

System Generator for DSP を使用すると、Simulink を使用して FPGA に高パフォーマンスの DSP システムをインプリメントできます。ザイリンクス ブロックセットには、演算ファンクションや論理ファンクション、デジタル フィルタ処理用の DSP 機能、スペクトル解析、デジタル通信のビット精度およびサイクル精度のモデルが含まれています。System Generator は、ザイリンクス ブロックの Simulink モデルを、FPGA で効率的に動作するように設計された合成可能 VHDL および IP ブロックを使用したハードウェア インプリメンテーションに変換します。

System Generator には、ツールの主要な機能を活用した多数のデモ デザインが含まれており、現実のデザイン アプリケーションを使用して適切なデザイン手法を示します。これらのデザインには、System Generator デモ ページからアクセスできます。

索引

I

ISE

ザイリンクス 20, 35, 36, 39, 40, 45, 51, 52

ISE Design Suite インストーラ 21

M

MATLAB 20, 35, 36, 39, 40, 45, 51, 52, 54, 56, 58, 60

ModelSim 54, 56, 58, 60

ModelSim SE 21

S

Simulink 35, 36, 39, 40, 45, 51, 52

System Generator

ISE Design Suite インストーラ 21

キャッシュ 30

ソフトウェアのダウンロード 19

バージョンの切り替え 30

バージョンの表示 30

System Generator ユーティリティ

xlUpdateModel 61

X

xlUpdateModel 61

い

インストール

ソフトウェア要件 21

ハードウェア協調シミュレーション 29

こ

互換性

MATLAB 20, 35, 36, 39, 40, 45, 51, 52, 54, 56, 58, 60

ModelSim 54, 56, 58, 60

ModelSim SE 21

Simulink 35, 36, 39, 40, 45, 51, 52

Synplify Pro 21, 54, 56, 58, 60

ザイリンクス ISE 35, 36, 39, 40, 45, 51, 52, 20

コンパイル

ザイリンクス HDL ライブラリ 30

さ

ザイリンクス HDL ライブラリ

コンパイル 30

サポートされる OS

Red Hat Linux 4.7 (32 ビットおよび 64 ビット) 20, 36, 40, 45, 52

Red Hat Linux 5.2 (32 ビットおよび 64 ビット) 36, 40

SUSE Linux Enterprise 10 (32 ビットおよび 64 ビット) 40

SUSE Linux Enterprise 10.1 (32 ビットおよび 64 ビット) 36

Windows Vista 32 ビットおよび 64 ビット 35, 39

Windows XP 32 ビット 20, 45, 51

Windows XP 32 ビットおよび 64 ビット 35, 39

せ

設定

System Generator キャッシュ 30

た

ダウンロード

System Generator 19

は

ハードウェア協調シミュレーション

インストール 29