

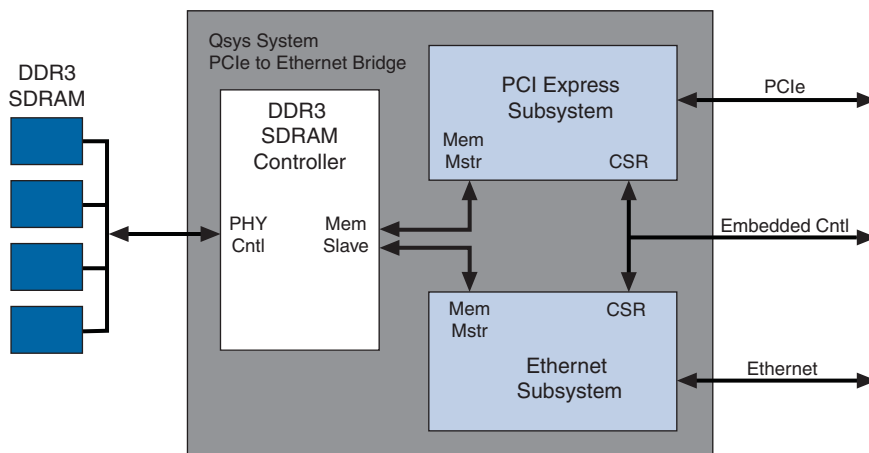
Qsys は、Quartus® II ソフトウェアの一部として含まれている強力なシステム統合ツールです。Qsys には抽象度の比較的に高いレベルでシステム・レベルのハードウェア・デザインをキャプチャします。また、IP コア、検証 IP、およびその他のデザイン・モジュールを含むことができるカスタマイズ HDL コンポーネントを統合する作業とタスクを定義することを自動化します。Qsys は、パッケージでデザインを再利用し、利用可能なカスタム・コンポーネントやシステムを使用できるようにします。Qsys は、Altera® のカスタム・コンポーネントおよびサード・パーティの開発者を統合しています。いくつかのケースでは、アルテラのコンポーネント・ライブラリからコンポーネントを使用してデザイン全体を実装することができます。システムの生成時に、Qsys は自動的に指定した接続のオプションから高性能のインタコネク・ロジックを作成し、システム・レベルの接続を指定するには、HDL の書き込みエラーと時間のかかりやすい作業を排除します。

Qsys はハードウェア・システム・デザインに対して以下の利点があります。

- コンポーネントのカスタマイズおよび統合のプロセスを自動化する。
- モジュラ・システム・デザインをサポートする。
- 大規模なシステムの可視化をサポートする。
- システム内のインタコネク・ファブリックとパイプラインの最適化をサポートする。
- Quartus II 開発ソフトウェアに完全な統合。

Qsys には、階層的なシステム・デザインをサポートします。別の Qsys システムのコンポーネントとしてインタフェースをエクスポートするすべての Qsys システムを含むことができます。図 5-1 は、イーサネットのブリッジに PCI Express を実装する Qsys のデザイン例のトップ・レベルを示しています。この例では、UniPHY IP コア付きのアルテラの DDR3 SDRAM コントローラと別の PCI Express とイーサネット・サブシステムを組合せします。デザイン・チームの別のメンバーは、完全なデザインの市場投入までの時間を減少させる同時に、様々なサブシステムを開発することもできます。デザイン例の詳細については、5-18 のページの「階層システムの例」を参照してください。

図 5-1. PCI Express のブリッジ・イーサネットのトップ・レベルのブロック図



Qsys の階層システム・デザインの利点は、以下の通りです。

- サブシステムに大規模なデザインを分割することによってチーム・ベースとモジュラー・デザインは有効にします。
- コンポーネントの任意の Qsys のシステムを使用できるようにすることによってデザインの再利用を有効にします。
- Qsys システムの複数のインスタンスをインスタンス化することができるようにすることでスケーラビリティを有効にします。

Qsys に、コンポーネント・パラメータを強化しサポートすると、最大効率とユーティリティのデザインをできるようになります。例えば、無制限のパラメータを可能にすると、実行時に RTL を生成するプログラムを供給するパラメータが指定されることがあります。

この章は、Qsys を紹介し、以下の項で構成されています。

- 5-2 のページの「Qsys GUI」
- 5-8 のページの「Qsys デザイン・フロー」
- 5-18 のページの「階層システムの例」

Qsys GUI


GUI の Quartus II ソフトウェアとコマンド・プロンプトを Qsys に起動できます。Quartus II ソフトウェアを Qsys 内に起動するには、File メニューの **New** をクリックします。New ダイアログ・ボックスで、**Qsys System File** をクリックします。

Qsys GUI について詳しくは、Quartus II ヘルプの [「About Qsys」](#) を参照してください。

Qsys Component Library

Qsys のコンポーネント・ライブラリを使用すると、Quartus II プロジェクトに含まれているかどうかにかかわらず、指定したコンポーネントの検索パス上に検出されたすべてのコンポーネントが含まれています。これらのコンポーネントは、アルテラが提供する IP コア、サード・パーティの IP コア、およびユーザーが提供するカスタムの IP コアが含まれています。Qsys のコンポーネントは、コンポーネント・ライブラリに記載され、インタフェースをエクスポートした場合は、デザインで使用することができます。コンポーネント・ライブラリは、インターコネクト Qsys に使用されているすべてのコンポーネントが含まれています。

アルテラは、ユーザーのコンポーネント・デザインの標準の Avalon インタフェースを使用することを推奨します。これらの標準インタフェースを使用することで、Qsys のコンポーネント・ライブラリのコンポーネントと相互運用性コンポーネントを作成することができます。デザインを検証するときに加えて、バス機能モデル (BFM)、モニタおよびその他の検証 IP を活用することができます。ただし、Qsys には、ユーザーのデザインが必要とする任意のインタフェースでデザインすることができます。一連の信号は、[「Avalon Interface Specifications」](#) に準拠できない場合は、コンジット・インタフェースとして信号をカプセル化することができます。Qsys 内のコンジット・インタフェースを接続すること、または即時モジュールの外部接続のためにそれらをエクスポートすることができます。

 すべてのインタフェース・タイプについて詳しくは、[「Avalon Interface Specifications」](#) を参照してください。BFMS について詳しくは、[「Avalon Verification IP Suite User Guide」](#) を参照してください。

アルテラとサード・パーティの開発者は、すぐに使用可能な Qsys コンポーネントを提供します。コンポーネント・ライブラリは、次のすべてを含むコンポーネントのさまざまな種類があります。

- Nios® II プロセッサなどのマイクロプロセッサ
- FIR Compiler II などの DSP IP コア
- PCI Express Compiler IP コアなどのインタフェース・プロトコル
- UniPHY 付きの RLDRAM II コントローラなどのメモリ・コントローラ
- Avalon-ST Multiplexer IP コアなどの Avalon-Streaming (Avalon-ST) コンポーネント
- アドレスをデコードする Qsys のマスタ・ルータなどの Qsys インタコネクト・コンポーネント

これらのコンポーネントは、Quartus II ソフトウェアで自動的にインストールされており、Qsys のコンポーネント・ライブラリで提供されています。


カスタム・コンポーネントの統合

Qsys のシステムにカスタム・コンポーネントを統合するには、次の手順を使用することができます。

1. カスタム・コンポーネントを対話するインタフェースを決定します。
2. Verilog HDL または VHDL を使用して、コンポーネントのロジックを作成します。

3. Hardware Component Description ファイルの (`_hw.tcl`) ファイルを定義するために、Qsys Component Editor を使用します。
4. システムでコンポーネントをインスタンス化します。

一度に Qsys コンポーネントを作成すると、他の Qsys システムでのコンポーネントを使用し、他のデザイン・チームとコンポーネントを共有することができます。

 カスタム Qsys コンポーネントを開発する手順について、コンポーネントのファイル構造、または Component Editor の詳細について、QuartusII ハンドブック Volume 1 の「[Creating Qsys Components](#)」の章を参照してください。

サード・パーティ・コンポーネントの統合

また、サード・パーティの IP 開発者によって作成された Qsys のコンポーネントを使用することができます。アルテラは、完全に Qsys のサポートされている IP コアを Qsys Certified ラベルの認証を与えます。これらのコアは、Avalon インタフェースをサポートし、タイミング制約と配置制約、ソフトウェア・ドライバ、シミュレーション・モデル、およびリファレンス・デザインが含まれる場合があります。


 ユーザーが「[Intellectual Property & Reference Designs](#)」のウェブページで Qsys コンポーネントを見つけることができます。Search and IP Cores & and Reference Designs に Qsys Compliant  を入力してください。ユーザーはアドバンスド検索をするときに、「[Altera Product Selector](#)」のウェブページで、Launch the Altera Product Selector Guide をクリックし、IP Selector タブをクリックしてください。

System 内容の追加

System Contents タブでは、システムに追加したコンポーネントが表示されます。

コンポーネントの追加

ユーザーのシステムにコンポーネントを追加するときに、Component Library にコンポーネントをクリックして、Add ボタンをクリックしてください。Parameter Editor が表示すると、コンポーネントをカスタマイズすることができます。

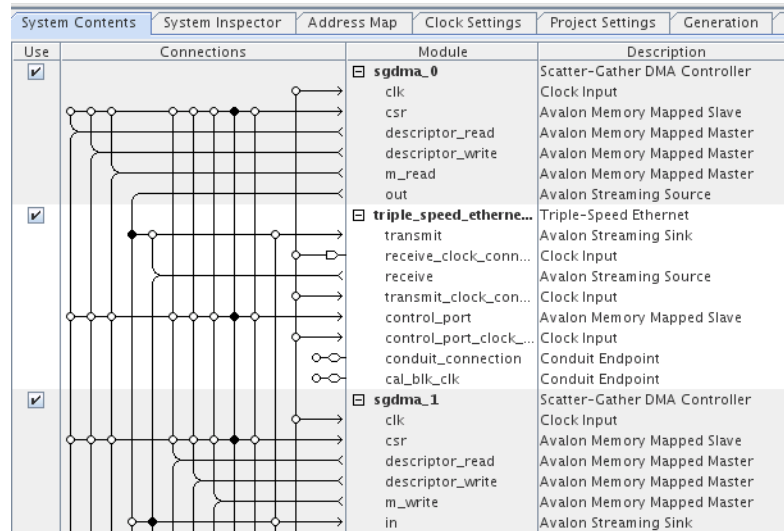
 文字列を含むすべてのコンポーネントを表示するには、Component Library の検索ボックスで一部またはすべてのコンポーネントの名前を入力することができます。

コンポーネントの接続

接続されたインタフェース内の個別の信号が自動的に接続することができるため、インタフェース・レベルでの接続を指定します。互換性のあるタイプのインタフェースと反対方向に接続します。例えば、ユーザーが Avalon-MM スレーブ・インターフェースへの Avalon Memory-Mapped (Avalon-MM) マスタ・インタフェース、または Avalon Interrupt レシーバ・インタフェースへの Avalon 割り込みの送信者のインターフェースを接続することができます。インタフェースの可能な接続を確認する

には、**System Contents** タブをクリックして、インタフェース名をクリックします。**Connections** カラムにカーソルを置きます。接続行列参照するには、白丸が可能な接続を表示し、黒丸は実行した接続を表示しています。接続を確立するには、2つのインタフェース名の交差点の白丸をクリックします。もう一度クリックすると、接続を削除します。図 5-2 に、接続行列を表示します。

図 5-2. 接続カラム



詳細について、Quartus II ヘルプの [「Connecting Qsys Components」](#) を参照してください。

コンポーネントのフィルタリング

System Contents タブで、システムの表示をフィルタリングするには、**Filters** ダイアログ・ボックスを使用することができます。インタフェース・タイプ、インスタンス名、またはカスタム・タグを使用して、システムの表示をフィルタすることができます。例えば、Avalon-MM インタフェースを含むインスタンスと特定の Nios II プロセッサに接続するインスタンスを表示し、一時的にクロックを削除し、または表示を簡略化するためのインタフェースをリセットするためにフィルタリングを使用することができます。


詳細について、Quartus II ヘルプの [「Filters Dialog Box」](#) を参照してください。

System Inspector の使用

System Inspector タブには、完全なシステムの基礎となるモデルが表示されます。**System Inspector** には、システムに関する包括的な詳細について以下の情報を指定します。

- すべての信号間の接続
- エクスポート・インタフェースを含むすべての信号の信号名

- コンポーネントとして含まれている Qsys のサブシステムの内部接続

 対照的に、System Contents タブでは、コンポーネントとして含まれている Qsys のサブシステムのみエクスポートされたインタフェースが表示されません。

- Project Settings タブで指定したグローバル・パラメータの設定。



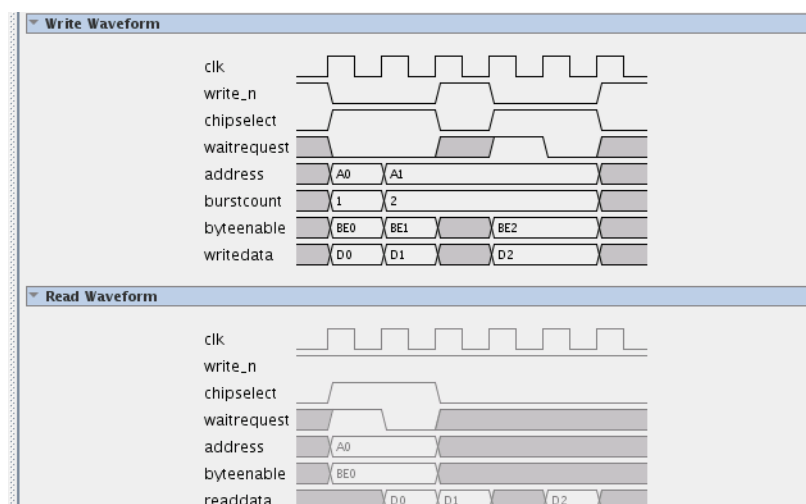

System Inspector タブを使用すると、コンポーネントのパラメータまたはインタフェースのタイミングを確認できます。例えば、5-19 ページの  5-8 に示す PCI Express-to-Ethernet システム用の Avalon-MM DMA ライト・マスタのタイミングを  5-3 に示します。

図 5-3. Project Settings タブで利用できる Avalon-MM ライト・マスタのタイミング波形




 インタフェースのタイミングを表示するには、そのインタフェースを表示するコンポーネントを展開し、インタフェース名をクリックしてください。

アドレス・マップの定義

Address Map タブはそれぞれの接続 Avalon-MM マスタがそのスレーブにアドレスするのに使用するデザインとアドレスの範囲にすべての Avalon-MM スレーブを含むテーブルを提供します。表は、各セルに表示される接続のアドレス・スパンで、上部でのマスタと左側でのスレーブを示しています。空白のセルは、マスタとスレーブ間の接続がないことを意味します。

マスタとスレーブのコンポーネント間の接続を変更または作成するために次の手順に従います。

1. Qsys に、**Address Map** タブをクリックします。
2. Avalon-MM マスタとスレーブのコンポーネント・ペア間の接続を表すテーブル・セルを配置します。
3. セル内にベース・アドレスを入力すること、またはセル内の現在のベース・アドレスを更新することのいずれかをします。

 2つの Avalon-MM マスタでは、異なるアドレスでの Avalon-MM スレーブにアクセスするシステムをデザインすることができます。この機能を使用する場合は、**System Contents** タブの **Base** と **End** アドレスのカラムは、むしろアドレスの範囲を提供するよりも、*mixed* ラベルが付いています。

Clock Settings の指定

ユーザー・システムのクロックを定義するには、**Clock Settings** タブを使用することができます。**Clock Settings** タブは、各クロックの **Name**、**Source**、または周波数 (MHz) を定義します。新しいクロックを追加するために **Add** ボタンをクリックします。デフォルト値を変更するには、まず適切な列をクリックし、バック・スペースをクリックし、新しい値を入力します。

詳細については、Quartus II ヘルプの「[Adding Components to a Qsys System \(To define clock domains in a system\)](#)」を参照してください。

Project Settings の指定

Qsys システムのプロパティを表示および変更するために **Project Settings** タブを使用することができます。表 5-1 に、**Project Settings** タブで利用可能なシステム・レベルのパラメータについて説明します。

表 5-1. Project Settings のパラメータ (その 1)

パラメータ名	説明
Device Family	アルテラのデバイス・ファミリを指定します。最終的なデザインがの HardCopy [®] シリーズのデバイスをターゲットにしている場合、ここにそのデバイスを指定します。
Clock Crossing Adapter Type	自動的に挿入されたクロックのデフォルトの実装を指定します。以下のオプションを選択できます。 <ul style="list-style-type: none"> ■ Handshake— このアダプタは、クロックの境界を越えて転送制御信号および応答を伝達するための簡単なハンド・シェイク・プロトコルを使用します。次の転送を開始する前に、各転送が安全にターゲット・ドメインに伝播されるため、この方法論は、より少ないハードウェア・リソースを使用します。Handshake のアダプタは、低スループットを必要とするシステムに適しています。 ■ FIFO— このアダプタは、同期のためのデュアル・クロック FIFOs を使用しています。FIFO ベースのアダプタでは、より多くのハンド・シェイク・クロック・クロッシング・コンポーネントよりもクロック・サイクルのカップルです。しかし、FIFO ベースのアダプタは、任意の時点で飛行中の複数のトランザクションをサポートできるので、より高いスループットを維持することができます。FIFO ベースのクロック・クロッサは、より多くのリソースを必要とします。FIFO のアダプタは、クロック・ドメイン間で高いスループットを必要とするメモリ・マップの転送に適しています。 ■ Auto— Auto を選択する合で、Qsys は破裂したリンクのための FIFO アダプタを指定し、すべての他のリンクを Handshake アダプタを指定します。

表 5-1. Project Settings のパラメータ (その 1)

パラメータ名	説明
Limit interconnect pipeline stages to	各コマンドのパイプライン・ステージ、または Qsys が追加のレイテンシを犠牲にして f_{MAX} を高めるために挿入する可能性があることを応答のパスの最大数を指定します。ユーザーは、0-4 パイプライン・ステージ間で指定することができます (0 はインタコネクトの組み合わせデータ・パスを持つことを意味する)。この設定は、単位の Qsys システムまたはサブシステムであり、各サブシステムが異なる設定を持つことができることを意味します。単一の Quartus II プロジェクトにそれらを組み合わせる場合でも、この追加レイテンシの二つの Qsys インタシステム用のコマンドと応答の方向のためであることに注意してください。
Generation ID	ちょうど Qsys インタシステム生成の前にタイム・スタンプに設定されている一意の整数値。Generation ID はソフトウェアの互換性を確認するために使用されません。

システム生成

Generation タブで生成するファイルを指定してください。シミュレーション・モデルまたは回路図デザイン用の **Block Symbol File (.bsf)** または Quartus II シンセシスための HDL ファイルのテスト・ベンチを生成することができます。デフォルトでは、Qsys は、プロジェクト・ディレクトリのサブディレクトリにこれらの出力ファイルを配置します。デフォルトの動作を変更するには、**Output Path** のディレクトリをクリックして新しいディレクトリを指定します。

Quartus II プロジェクトには、Quartus II IP ファイル (**.qip**) ファイルを追加する必要があります。**.qip** ファイルは生成後の合成のディレクトリに格納されています。それは、Quartus II のコンパイルに必要なファイルが一覧表示されます。**.qip** ファイルは、次の情報への参照が含まれています。

- Qsys システムで使用する HDL ファイル
- TimeQuest Timing Analyzer Synopsys Design Constraint (**.sdc**) ファイル
- 目的をアーカイブするためのコンポーネントの定義ファイル

Quartus II のプロジェクトにファイルを追加することについては、Quartus II ヘルプの [「Managing Files in a Project」](#) を参照してください。

HDL Example の表示

HDL Example タブには、Verilog HDL または VHDL での Qsys システムのトップ・レベル HDL の定義を提供します。このタブには、VHDL コンポーネント宣言を表示します。システムは、Quartus II プロジェクトではトップ・レベルのモジュールでない場合は、この例をコピーして、Qsys システムをインスタンス化するトップ・レベル HDL ファイルに貼り付けることができます。

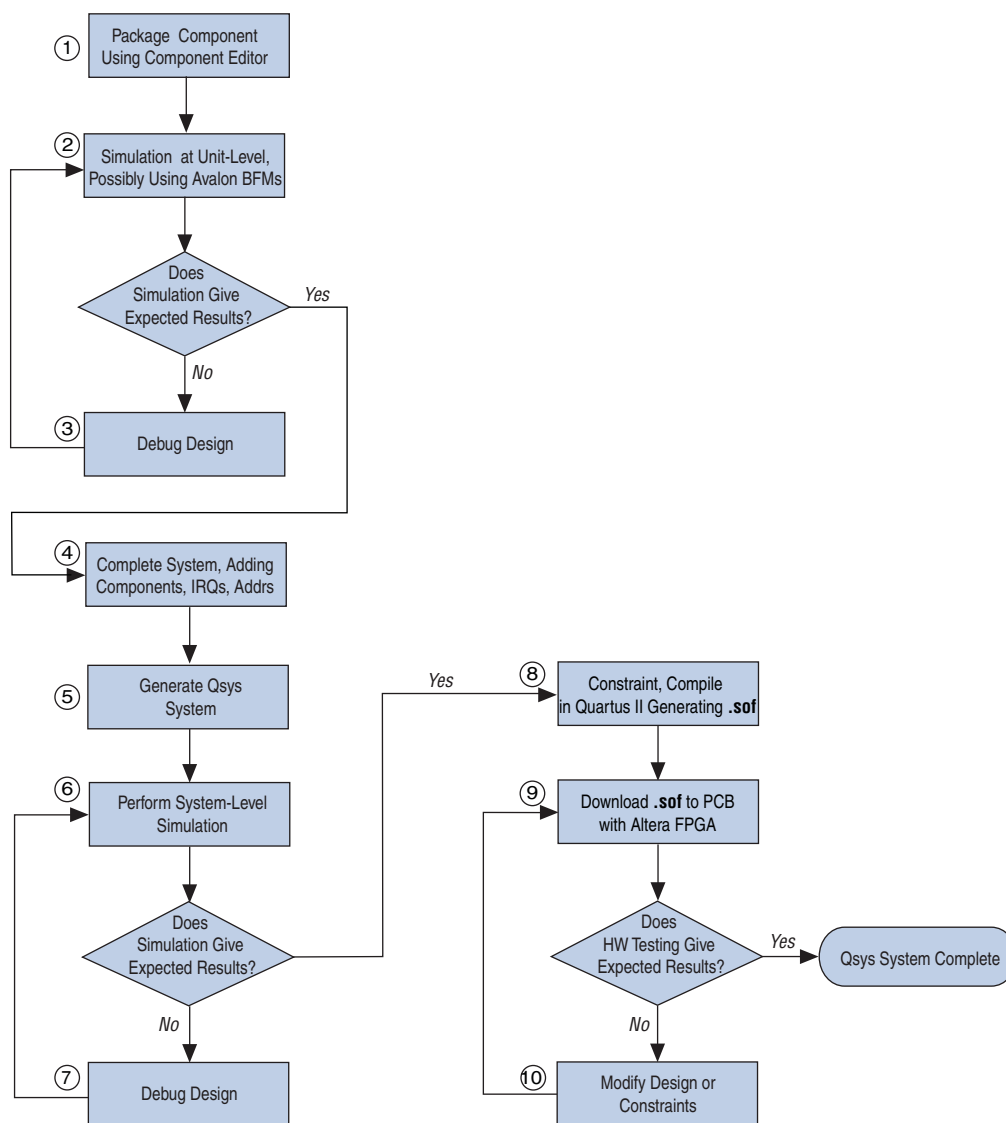
Qsys デザイン・フロー


図 5-4 は、コンポーネントのデザインで始まる Qsys のボトム・アップ・デザイン・フローの例を示しています。このフロー図に示すように、典型的なデザインフローは、次の高度な手順が含まれています。

1. Component Editor を使用して Qsys のコンポーネントをパッケージします。

2. システムを確認するためにアバロンの BFM を組み込むユニット・レベルでシミュレートします。
3. Qsys デザインを完了するために、他のコンポーネント、割り込みの指定、クロック、リセットおよびアドレスを追加します。
4. Qsys システムを生成します。
5. システム・レベルのシミュレーションを実行します。
6. デザインを制約およびコンパイルします。
7. アルテラのデバイスにデザインをダウンロードします。
8. ハードウェアをテストします。



図 5-4. 完全な Qsys デザイン・フロー



 代替のトップ・ダウンの有効なデザイン・フローでは、Qsys システムをデザインすることで開始してから、カスタム Qsys を定義してインスタンス化します。このアプローチは、デザイン・プロセスの初期段階でシステム要件を明確にしています。

HardCopy デバイスをターゲットとしたデザインは、特定のデザイン上の制約が必要です。したがって、HardCopy シリーズ・デバイスをターゲットとしている場合は、HardCopy コンパニオン・デバイスのデザインを確認する必要があります。

両方のデバイスのために、デザインを検証するには、これらのガイドラインに従ってください。

1. Quartus II **Device** のダイアログ・ボックスで、FPGA と適切な HardCopy コンパニオン・デバイスの両方を選択します。
2.  [5-4](#) に示すように、デザイン・フローのステップ 8 で、FPGA および HardCopy デバイスの両方をコンパイルします。
3.  [5-4](#) に示すように、デザイン・フローのステップ 10 の後で、FPGA のすべての機能シミュレーションとハードウェアの検証テストに合格した場合は、HardCopy ハンド・オフ・アーカイブを生成し、バック・エンドのフローとテープ・アウトのための HardCopy デザイン・センターにこのアーカイブを送信します。

HardCopy デバイスをデザインすることについては、Quartus II ヘルプの [「About Designing HardCopy Devices」](#) を参照してください。

生成出力ファイル

Qsys システム生成は、コンポーネント間の相互接続を作成します。生成時には、シミュレーションのみのためのまたはシミュレーションと合成用のファイルを生成するように選択することができます。シミュレーションと合成用のファイルに加えて、Qsys は .bsf、システム・テスト・ベンチ、および HTML のデータ・シートを作成します。

図 5-5 は、出力ファイルのディレクトリ構造を示しています。

図 5-5. Qsys 生成されるファイルのディレクトリ構造

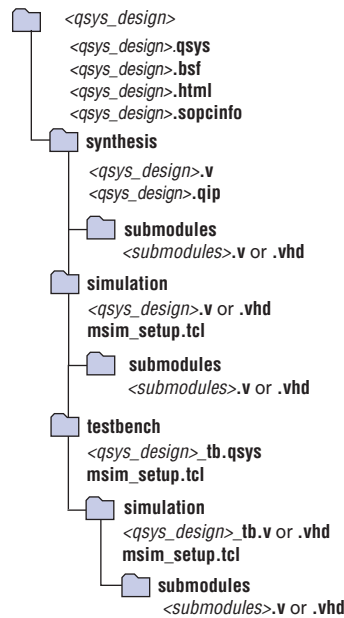


表 5-2 は、Qsys の生成されるファイルを説明します。システムを生成するたびに、Qsys は、これらのファイルが上書きされます。ボード・レベルのタイミング制約などの制約がある場合、アルテラでは、別のと Synopsys Design Constraints ファイル (.sdc) を作成し、Quartus II プロジェクトにそのファイルをインクルードすることを推奨します。

表 5-2. Qsys の生成するファイル

ファイル名またはディレクトリ名	説明
<code><qsys_design></code>	これがトップ・レベルのプロジェクト・ディレクトリです。
<code><qsys_design>.qsys</code>	Qsys のシステム・ファイル (.qsys)。 .qsys はシステム・コンポーネント、接続とそのパラメタリゼーションのリストが含まれています。
<code><qsys_design>.bsf</code>	ブロック・シンボル・ファイル (.bsf) は Quartus II のブロック図ファイル (.bdf) で使用するためのトップ・レベルの Qsys システムを表現します。
<code><qsys_design>.html</code>	これは、次の情報を含むシステムの概要を提供するシステムのデータ・シートです。 <ul style="list-style-type: none"> ■ システムのすべての外部接続 ■ それが接続されている各 Avalon-MM マスタに対する各 Avalon-MM スレーブのアドレスを示すメモリ・マップ ■ 各コンポーネントのすべてのパラメータ・アサインメント

表 5-2. Qsys の生成するファイル

ファイル名またはディレクトリ名	説明
<qsys_design>.sopcinfo	Qsys 情報ファイル (.sopcinfo) はシステムのコンポーネントとの接続のすべてを記述します。このファイルは、完全なシステムの説明であり、そのような Nios II プロセッサのツール・チェーンなどの下流のツールで使用されます。それは、システム内の各コンポーネントのパラメータについても説明し、結果的には、Qsys コンポーネント用のソフトウェア・ドライバを開発する際の要件を取得するには、その内容を解析することができます。 このファイルと Nios II ツール・チェーン用に生成される system.h ファイルは、スレーブにアクセスする各マスタに対する各スレーブのアドレス・マップの情報が含まれています。別のマスタは、特定のスレーブのコンポーネントにアクセスするために別のアドレス・マップを持つことができます。
/synthesis	このディレクトリには、Quartus II ソフトウェアはデザインを合成するために使用するファイルが含まれています。これらのファイルは、ユーザーのシステムを生成するたびに上書きされます。
<qsys_design>.v	トップ・レベルの Qsys システムおよびシステム内の各コンポーネントの HDL ファイルです。<qsys_design>/synthesis ディレクトリ下のファイルは、合成に使用されます。
<qsys_design>.qip	このファイルは、デザインをコンパイルするために必要な Quartus II ソフトウェアが表示されます。Quartus II プロジェクトに .qip ファイルを追加する必要があります。
/submodules	Verilog HDL または合成のための VHDL のサブモジュールのファイルが含まれています。
/simulation	このディレクトリは、デザインをシミュレーションするための VHDL ファイルまたは Verilog HDL ファイルが含まれています。
<qsys_design>.v or <qsys_design>.vhd	トップ・レベルの Qsys システムおよびシステム内の各サブモジュールの HDL ファイルです。
msim_setup.tcl	ModelSim のスクリプトはセットアップとシミュレーションを実行するためです。
/submodules	Verilog HDL またはシミュレーション用の VHDL のサブモジュールのファイルが含まれています。
/testbench	「Qsys システムのシミュレーション」のセクションに説明される Qsys のテスト・ベンチ・システムが含まれています。
<qsys_design>_tb.v or .vhd	トップ・レベルのテスト・ベンチ・ファイルは、<qsys_design>.qsys の最上位レベルのインタフェースに BFM を接続します。
msim_setup.tcl	ModelSim のスクリプトはセットアップとシミュレーションを実行するためです。
/submodules	Verilog HDL またはテスト・ベンチの VHDL のサブモジュールのファイルが含まれています。

Qsys システムのシミュレーション

Qsys **Generation** タブで、システムをシミュレートするためのさまざまなオプションが用意されています。Qsys システムをシミュレートするため、次の 3 つのオプションがあります。

- ユーザーのシミュレーション環境で使用するシミュレーション・モデルを生成することにより、カスタム・テスト・ベンチを作成することができます。
- Avalon バス・ファンクション・モデル (BFM) をシステムの外部インターフェースを駆動する部品の標準でテスト・ベンチ・システムを生成することができます。
- そのシミュレーション・モデルを生成する前に、標準テスト・ベンチ・システムを生成し、Qsys システムを変更することができます。

表 5-3 は、Generation タブで別のオプションのまとめを示しています。

表 5-3. Qsys Generation Tab タブでシミュレーション設定の概要

シミュレーションの設定	値	説明
シミュレーション・モデルの作成	None Verilog VHDL	シミュレーション・モデルのファイルとシミュレーションのスクリプトを作成します。ユーザーのカスタム・ベンチのシミュレーション・モデルが含まれているために、このオプションを使用します。
テスト・ベンチ Qsys システムの作成	標準または、標準の Avalon インタフェース用の BFM	すべてのエクスポートされたインターフェースに接続されている Avalon BFM とテスト・ベンチ Qsys システムを作成します。システムの IP コアで指定された任意のシミュレーション・パートナーのモジュールが含まれています。このテスト・ベンチは、VHDL のシミュレーション・モデルではサポートされていません。シミュレーション・モデルを作成する前に、このオプションを使用して、テスト・ベンチを表示して、変更します。
	シンプル、またはクロックとリセット用の BFM	Avalon BFM を駆動するのみのクロックとインターフェース・リセットのテスト・ベンチ Qsys インタシステムを作成します。システムの IP コアで指定された任意のシミュレーション・パートナーのモジュールが含まれています。
テスト・ベンチ・シミュレーション・モデルの作成	None Verilog VHDL	上記の設定で指定されたテスト・ベンチ Qsys インタシステムのシミュレーション・スクリプトとシミュレーション・モデルのファイルを作成します。シミュレーションを実行する前に、Qsys で生成したテスト・ベンチを変更する必要はない場合は、このオプションを使用します。

テスト・ベンチのカスタムの生成

カスタム・テスト・ベンチのシミュレーション・モデルを生成するには、次の手順を実行してください。

1. **Generation** タブで、**Verilog** や **VHDL** に **Create simulation model** を設定します。
2. **Generate** をクリックします。

このオプションは、ModelSim® シミュレータのシミュレーション・スクリプトと一緒に、**Simulation** に指定された **Output Directory** シミュレーション・モデルのファイルを作成します。



Qsys は、Verilog HDL または VHDL で最上位レベルのデザインと Qsys のインターコネクタのシミュレーション・モデルを生成することができます。しかし、システム内の IP コンポーネントは、指定された言語でのシミュレーション・ファイルが用意されていない場合、完全なシミュレーション環境を得ることができない場合があります。

標準の Qsys テスト・ベンチの生成

システムへのスティミュラスを提供するテスト・ベンチ・システムを生成するためには、次の手順に従います。

1. **System Contents** タブで、アクセスされている Qsys システムのすべてのインタフェースをエクスポートします。
2. **Generation** タブで、すべてのエクスポートされたインタフェースに接続されている BFM とテスト・ベンチを作成するために、**Create testbench Qsys system to Standard, BFM for standard Avalon interfaces** を選択してください。また、BFMs だけのクロックの駆動とリセット・インタフェースのテスト・ベンチを作成するために、**Simple, BFM for clocks and resets** を選択します。
3. 同時にテスト・ベンチの Qsys システムのためのシミュレーション・モデルを生成するには、**Verilog** または **VHDL** に **Create testbench simulation model** を設定します。そのシミュレーション・モデルを生成する前に生成されたテスト・ベンチ・システムを表示し、変更する **None** のオプションを設定します。
4. **Generate** をクリックします。



Avalon の検証 BFM が含まれている Qsys システムの場合です。BFM コンポーネントで System Verilog に依存しているため、VHDL シミュレーション・モデルがサポートされていません。クロックとリセットを持つ唯一の簡単なテスト・ベンチは、VHDL のシミュレーションのためにサポートされています。

5. `<qsys_system>/testbench` のサブディレクトリの `<qsys_system>_tb.qsys` ファイルを開いて、テスト・ベンチ・システムを表示し、変更します。次の変更を加えることができます。
 - a. シミュレーション環境を一致させる BFM の名前を変更することができます。
 - b. 十分なテスト・カバレッジを確保するための BFM の設定を変更することができます。
 - c. カスタム・テスト・コンポーネントやモデルでデフォルトの BFM を置き換えることができます。

Qsys で生成された標準テスト・ベンチは、Qsys のデザインからエクスポートされたインタフェースで挿入する BFM にマッチします。BFM にトランザクションの刺激を提供するテスト・プログラムを書くために、アカウントに一致するインタフェースを取る必要があります。例えば、ワード・アラインされたアドレスを想定するエクスポートされた Avalon-MM スレーブ・インタフェースは、Avalon マスタ BFM に接続され、アバロンのマスタのデフォルトなバイトとシンボルのアドレスではなく、ワード・アラインされたアドレスをトランザクし、必要とします。

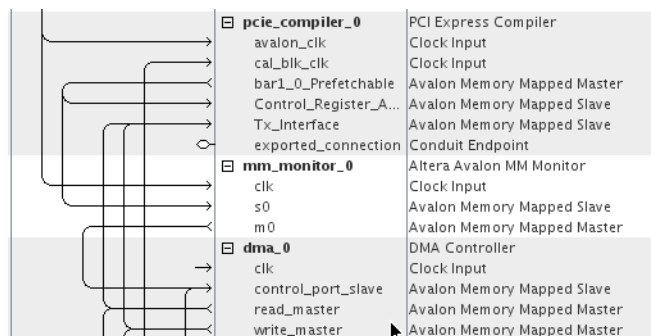


サンプルのシステムを示すチュートリアルなどのトランザクション・レベルのテスト・コードを書くことと BFM を使用することの詳細については「[Avalon Verification IP Suite User Guide](#)」を参照してください。

システム・モニタの追加

System Verilog アサーションをサポートするシミュレータと、プロトコルの正当性とテストのカバレッジを確認するための Avalon-MM とは、システムの Avalon-ST インタフェースにモニターを追加することができます。図 5-6 は、モニターの使用方法を示します。それは、以前に接続した pcie_compiler bar1_0_Prefetchable Avalon-MM マスタ・インタフェースと dma_0 control_port_slave Avalon-MM スレーブ・インタフェースの間に Avalon-MM モニタを配置します。

図 5-6. Avalon-MM マスタとスレーブ・インタフェースの間に Avalon-MM モニタの取り付け



同様の方法では、Avalon-ST ソースおよびシンクのインタフェースの間に Avalon-ST モニタを挿入することができます。

 サンプル・システムを実証するチュートリアルなどのシステム・モニターと BFM の使用方法については「[Avalon Verification IP Suite User Guide](#)」を参照してください。

ModelSim シミュレーション・スクリプト

Qsys は ModelSim シミュレーション環境を設定し、正しい順序で必要なデバイスのライブラリとシステムのデザインファイルをコンパイルし、シミュレーションのためのトップ・レベルのデザインをエラボレートまたはロードするためにエイリアス・コマンドを作成し、ModelSim シミュレーション・スクリプト、`msim_setup.tcl` を生成します。このスクリプトを実行するには、ModelSim Transcript のウィンドウで `source msim_setup.tcl` を入力してください。表 5-4 に ModelSim シミュレーションを実行するときに表示される追加のコマンドを表示されています。

表 5-4. ModelSim のコマンド

コマンド	内容
h	コマンド・ライン・エイリアスのリストを示します。
ld	すべてのデザイン・ファイルをコンパイルし、トップ・レベルのデザインを詳しく説明します。
run -all	デザインをロードした後に、このコマンドは、シミュレーションを開始します。

msim_setup.tcl スクリプトは、変数が用意されると、独自のトップ・レベルのシミュレーション・ファイル内の Qsys テスト・ベンチ・システムをインスタンス化、および Qsys で生成したシミュレーション・モデルの場所へのパスを指定できるようになります。Qsys のテスト・ベンチは、シミュレーション環境でトップ・レベルのインスタンスでない場合は、最上位の階層の名前に `TOP_LEVEL_NAME` 変数を設定します。Qsys で生成されるシミュレーション・ファイルは、シミュレーション作業ディレクトリにない場合は、Qsys のシミュレーション・ファイルのディレクトリの場所を指定するために `QSYS_SIMDIR` 変数を使用してください。

例 5-1 は、`pattern_generator` と呼ばれる Qsys システム用に生成されたテスト・ベンチのシステム `pattern_generator_tb` のためのシンプルなトップ・レベルのシミュレーション・ファイルを示しています。`top.sv` ファイルは `pattern_generator_tb` シミュレーション・モデルだけでなく、`test_program` と呼ばれる BFM のトランザクション付きのカスタム System Verilog のテスト・プログラムをインスタンス化する `top` モジュール・ファイルを作成します。

例 5-1. トップ・レベル・シミュレーション・ファイル

```
module top();  
  pattern_generator_tb tb();  
  test_program pgm();  
endmodule
```

例 5-2 は、カスタムのトップ・レベルのシミュレーション・スクリプトを示しており、例 5-1 のデザインに、階層の変数 `TOP_LEVEL_NAME` を `top` に設定し、Qsys で生成された **msim_setup.tcl** ファイルの場所に変数 `QSYS_SIMDIR` を設定します。この例では、トップ・レベルのシミュレーション・ファイルは、オリジナルの被試験の Qsys システムと同じディレクトリに格納されています。`QSYS_SIMDIR` 変数は、Qsys のテ

スト・ベンチのシミュレーション・モデルのために生成する相対的な階層のパスを提供します。スクリプトは、Qsys で生成された ModelSim のスクリプトを呼び出し、Qsys ファイルをコンパイルし、または詳しく説明するために `msim_setup.tcl` ファイルは、カスタム・テスト・プログラムとトップ・レベルのシミュレーション・ファイルと一緒にシミュレーションに必要があります。

例 5-2. カスタム・テスト・プログラムに含むトップ・レベルのシミュレーション・スクリプト

```
# Set hierarchy variables used in the Qsys-generated files

set TOP_LEVEL_NAME "top"
set QSYS_SIMDIR "./pattern_generator/testbench"

# Source Qsys-generated script and set up alias commands used below
source $QSYS_SIMDIR/msim_setup.tcl

# Compile device library files
dev_com

# Compile design files in correct order
com

# Compile the additional test files
vlog -sv ./test_program.sv
vlog -sv ./top.sv

# Elaborate the top-level design
elab
```



このシミュレーションの例について詳しくは、「[Qsys Tutorial Design Example](#)」の「[Simulating Custom Components](#)」章を参照してください。

トップ・レベルのシミュレーション・スクリプトやテスト・プログラムを使用すると、Qsys システムを変更して再生成する際に必要なすべてのシミュレーションファイルをコンパイル確保するために Qsys で生成された ModelSim のスクリプトを利用しています。また、独立に、上書きされている Qsys で生成したシミュレーション・ファイルのトップ・レベルのシミュレーション環境を変更することができます。

Nios II プロセッサ上で動作をシミュレートするソフトウェア

Nios II エンベデッド・プロセッサによって駆動されるシステムでのソフトウェアのコードをシミュレートするには、次の手順を実行して、単純 Qsys テスト・ベンチ・システムのためのシミュレーション・モデルを生成します。

1. **Generation** タブで、**Create testbench Qsys system** は **Simple, BFM for clocks and resets** に設定します。
2. **Create testbench simulation model** は **Verilog** または **VHDL** に設定します。
3. **Generate** をクリックします。

シミュレーション用の Eclipse を使用するために、次のステップに従います。

1. Eclipse 用の NiosII SBT を開きます。
2. `<qsys_system>.sopcinfo` ファイルにアプリケーションのプロジェクトとボード・サポート・パッケージ (BSP) を設定します。

3. シミュレーション用の BSP と無効なハードウェアのプログラミングを最適化するには、BSP のプロジェクトを右クリックし、**Properties** を選択します。そして **Nios II BSP Properties** を選択し、**ModelSim only, no hardware support** をオンにします。
4. シミュレートするには Eclipse のアプリケーション・プロジェクトを右クリックし、**Run as** をポイントし、**4Nios II ModelSim** を選択します。**Run As Nios II ModelSim** コマンドは ModelSim シミュレーション環境を設定しコンパイルし、または Nios II ソフトウェアのシミュレーションをロードします。
5. ModelSim でシミュレーションを実行するには、ModelSim トランススクリプト・ウィンドウに `run -all` を入力してください。
6. プロンプトが表示された場合は、ModelSim の構成設定を設定し、適切な Qsys Testbench Simulation Package Descriptor (**.spd**) file, `<qsys_system>_tb.spd` を選択してください。SPD ファイルは、Nios II デザインのためのテスト・ベンチのシミュレーション・モデルで生成され、Nios II ソフトウェアのシミュレーションに必要なすべてのファイルを指定しています。

 Eclipse 用に NiosII ソフトウェア・ビルド・ツールについては、「NiosII ソフトウェア開発ハンドブック」の「*Getting Started with the Graphical User Interface*」を参照してください。NiosII ソフトウェア・ビルド・ツールのコマンド・ラインについては、「NiosII ソフトウェア開発ハンドブック」の「*Getting Started from the Command Line*」を参照してください。

階層システムの例

図 5-7 は 5-2 ページの図 5-1 にある非常に高いレベルで図示されている PCI Express の例のサブシステムの詳細を示します。この例のシステムでは、複雑なプロセッサは DMA コントローラ動作するソフトウェア・アプリケーションをプログラムします。DMA コントローラの Avalon-MM のリードとライトのマスタ・インタフェースは、DDR3 メモリへ、そして DDR3 メモリから送信を開始します。また、PCI Express Avalon-MM TX データ・ポートへの転送も開始します。システムは、Avalon-MM パイプライン・ブリッジを介して DMA のマスタ・インタフェースをエクスポートします。図に示すように、すべての 3 つのマスタは、単一のスレーブ・インタフェースに接続します。システム生成時には、このスレーブ・インタフェースへのアクセスを制御するために、Qsys は自動的にアービトレーション・ロジックを挿入します。デフォルトでは、アービタは、すべての要求元のマスタへの平等なアクセスを提供しますが、要求するマスタのためにアービトレーション・シェアの数を変更することにより、アービトレーションを加重することができます。第二の Avalon-MM パイプライン・ブリッジは、制御とステータスのインタフェースが内部でエクスポートに接続することができます。

詳細は、「QuartusII ハンドブック Volume 1」の「Qsys Interconnect」の章の「Arbitration」を参照してください。

図 5-7. PCI Express のサブシステムのブロック図

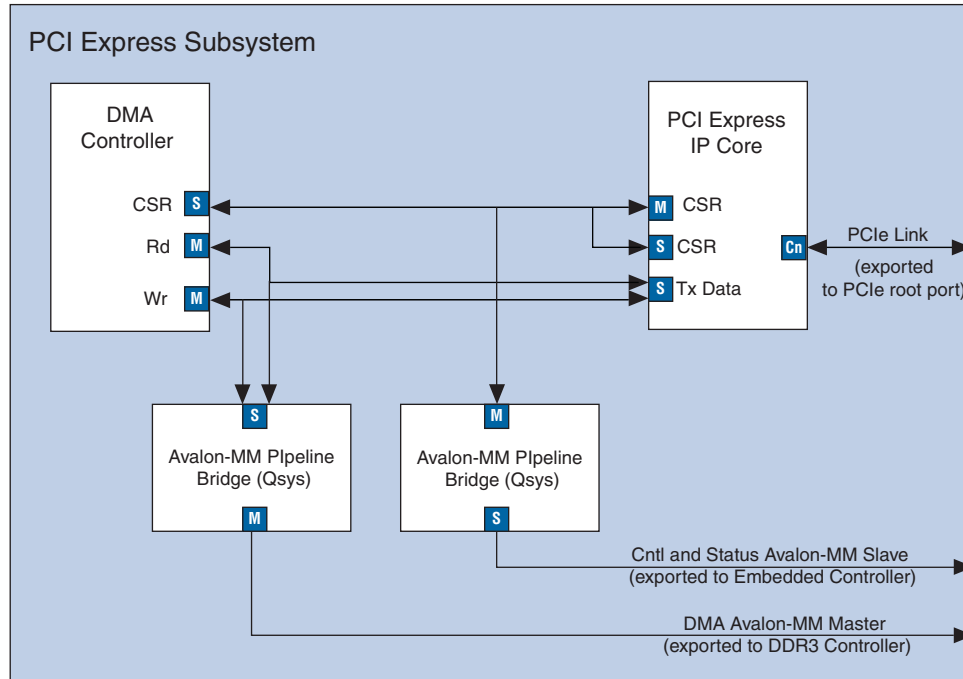


図 5-8 に、PCI Express のサブシステムの Qsys 表現を示しています。

図 5-8. PCI Express サブシステムの Qsys 表現

Use	Connections	Module	Description	Export As
<input checked="" type="checkbox"/>		pcie_compiler_0	PCI Express Compiler	
		bar1_0_Prefetchable	Avalon Memory Mapped Master	Click to export
		Control_Register_A...	Avalon Memory Mapped Slave	Click to export
		Cralrq	Interrupt Sender	Click to export
		RxmIrq	Interrupt Receiver	Click to export
		Tx_Interface	Avalon Memory Mapped Slave	Click to export
		exported_connection	Conduit Endpoint	pcie_link
<input checked="" type="checkbox"/>		dma_0	DMA Controller	
		control_port_slave	Avalon Memory Mapped Slave	Click to export
		irq	Interrupt Sender	Click to export
		read_master	Avalon Memory Mapped Master	Click to export
		write_master	Avalon Memory Mapped Master	Click to export
<input checked="" type="checkbox"/>		mm_bridge_0	Avalon-MM Pipeline Bridge (Qsys)	
		s0	Avalon Memory Mapped Slave	Click to export
		m0	Avalon Memory Mapped Master	ddr3_sdr3_master

図 5-9 は図 5-1 からイーサネットの例のサブシステムの詳細が表示されます。このサブシステムでは、送信 (TX) DMA は、DDR3 メモリからデータを受信し、Avalon-ST ソース・インタフェースを使用してアルテラのトリプル・スピード・イーサネット IP コアに書き込みます。レシーバ (RX) DMA が Avalon-ST シンク・インタフェース上でトリプル・スピード・イーサネット IP コアからデータを受け取り、DDR3 メモリに書き込みます。

読み取りと書き込みマスタの両方の Scatter-Gather DMA コントローラとトリプル・スピード・イーサネット IP コアは、Avalon-MM パイプライン・ブリッジ経由で DDR3 メモリに接続します。このイーサネットの例ではサブシステムは、Qsys システムの外部にコントローラに接続された Avalon-MM パイプライン・ブリッジを介してすべての 3 つの制御とステータスのインタフェースをエクスポートします。

図 5-9. Scatter-Gather DMA-to-Ethernet の例のサブシステム

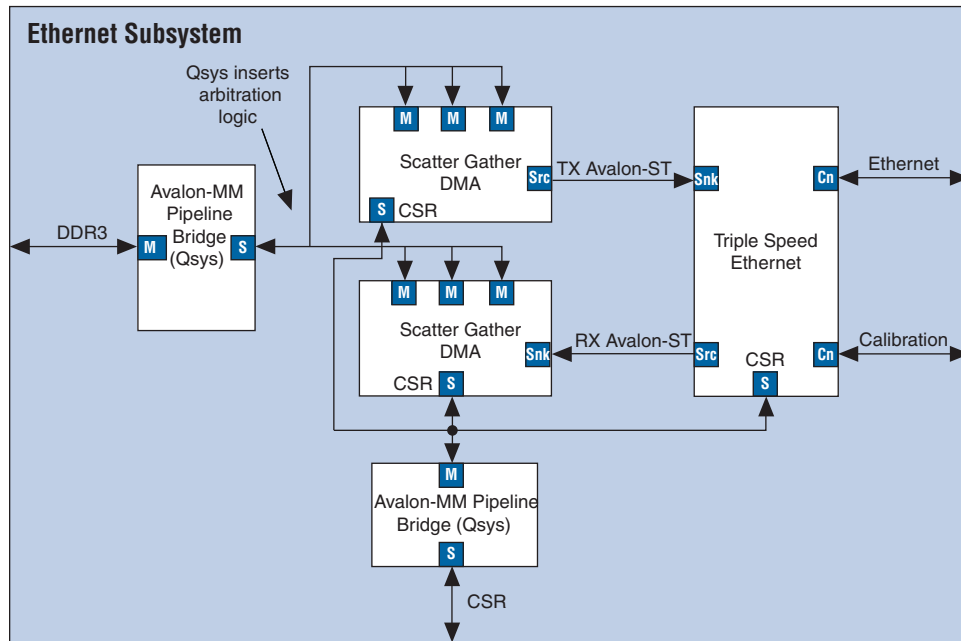


図 5-10 は、イーサネットのサブシステムの Qsys 表現を示しています。

図 5-10. イーサネットサブシステムの Qsys 表現

Use	Connections	Module	Description
<input checked="" type="checkbox"/>		sgdma_0	Scatter-Gather DMA Contro
		csr	Avalon Memory Mapped Sla
		descriptor_read	Avalon Memory Mapped Ma
		descriptor_write	Avalon Memory Mapped Ma
		csr_irq	Interrupt Sender
		m_read	Avalon Memory Mapped Ma
		out	Avalon Streaming Source
<input checked="" type="checkbox"/>		triple_speed_ethernet_0	Triple-Speed Ethernet
		transmit	Avalon Streaming Sink
		receive	Avalon Streaming Source
		control_port	Avalon Memory Mapped Sla
		conduit_connection	Conduit Endpoint
		cal_blk_clk	Conduit Endpoint
<input checked="" type="checkbox"/>		sgdma_1	Scatter-Gather DMA Contro
		csr	Avalon Memory Mapped Sla
		descriptor_read	Avalon Memory Mapped Ma
		descriptor_write	Avalon Memory Mapped Ma
		csr_irq	Interrupt Sender
		m_write	Avalon Memory Mapped Ma
		in	Avalon Streaming Sink
<input checked="" type="checkbox"/>		anmm_bridge_0	Avalon-MM Bridge
		avalon_slave	Avalon Memory Mapped Sla
		avalon_master0	Avalon Memory Mapped Ma
		avalon_master1	Avalon Memory Mapped Ma
		avalon_master2	Avalon Memory Mapped Ma

この例のシステムでは、2つのクロック・ドメインが含まれています。PCI Express とイーサネットのサブシステムは 125MHz で動作します。DDR3 SDRAM のコントローラは 200 MHz で動作します。Qsys は自動的に、PCI Express とイーサネット・サブシステムとの DDR3 SDRAM コントローラを同期するために、クロック・クロッシングを挿入します。図 5-11 は、例のシステムの最上位レベルを示しています。

図 5-11. PCI Express-to-Ethernet ブリッジの例のシステム

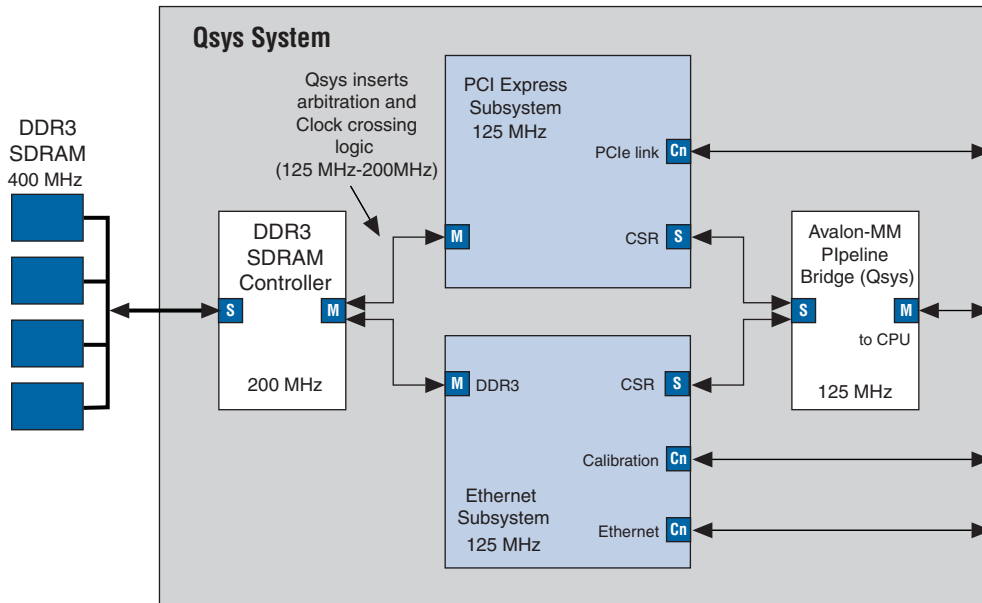


図 5-12 に、完全なデザインの Qsys 表現を示しています。


図 5-12. イーサネット・ブリッジへの完全な PCI Express の Qsys の表現

Use	Connections	Module	Description	Export As
<input checked="" type="checkbox"/>		uniphy_ddr3_0	DDR3 SDRAM Controller with UniPHY (New)	
		avalon_slave_0	Avalon Memory Mapped Slave	Click to export
		memory_phy	Conduit	ddr3_sdram
		Other	Conduit	Click to export
		PLL_sharing	Conduit	Click to export
<input checked="" type="checkbox"/>		ethernet_dma_subsystem	ethernet_dma_subsystem	
		cal_blk_if	Conduit Endpoint	dma_calibration
		csr_if	Avalon Memory Mapped Slave	Click to export
		dma_if	Avalon Memory Mapped Master	Click to export
		ethernet_if	Conduit Endpoint	ethernet
<input checked="" type="checkbox"/>		pcie_subsystem	pcie_subsystem	
		pcie_link	Conduit Endpoint	pcie_link
		ddr3_sdram_master	Avalon Memory Mapped Master	Click to export
		dma_control_slave	Avalon Memory Mapped Slave	Click to export
<input checked="" type="checkbox"/>		mm_bridge_0	Avalon-MM Pipeline Bridge (Qsys)	
		s0	Avalon Memory Mapped Slave	embedded_control
		m0	Avalon Memory Mapped Master	Click to export

パイプライン・ブリッジの使用

PCI Express to Ethernet ブリッジの例のシステムは、複数のパイプライン・ブリッジを使用します。これらのブリッジはシステム階層の次の高いレベルのコンポーネントや発信サブシステムのコンポーネントなどの連結成分のすべてのアドレス範囲を対応するように、構成される必要があります。名前が示すように、パイプライン・ブリッジは接続されたコンポーネントの間にパイプライン・ステージを挿入します。アルテラは、次の理由により、サブシステムのインタフェース・レベルで信号を登録することを推奨します。

- インタフェース信号を登録すると、1 サイクルで完了する必要が組み合わせロジックの量を減少させるになり、タイミングをクローズすることが簡単になります。
- インタフェース信号を登録すると、システムのスループットに影響を与える可能性のあるサイクルのレイテンシが追加の犠牲にしてデザインの、潜在的な周波数、または f_{MAX} を発生させます。
- サブシステムの境界が登録されている場合で、Quartus II インクリメンタル・コンパイル機能は、より良い f_{MAX} の結果を得ることができます。

 インクリメンタル・コンパイルについて詳しくは、「Quartus II ハンドブック Volume 1」の「*Quartus II Incremental Compilation for Hierarchical and Team-Based Design*」を参照してください。

階層的なコンポーネントの作成

インタフェースをエクスポートする任意の Qsys システムは、他の Qsys システムで使用できるようになります。図 5-13 に、PCI Express to Ethernet ブリッジの例のシステムのコンポーネント・ライブラリで、PCI Express やイーサネット・サブシステムなどをコンポーネントとしてコンポーネント・ライブラリを示しています。他のデザインでこの Qsys のコンポーネントを含めるには、コンポーネント・ライブラリに追加するか、Qsys のためのコンポーネントの検索パスで、このコンポーネント用のディレクトリを含めることができます。


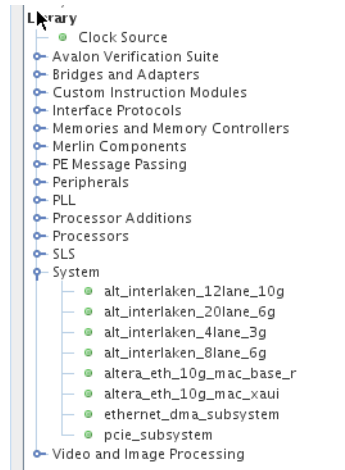


 Qsys システムは、コンポーネント・ライブラリのコンポーネントになっているため、使ったシステムに名前をつけないように注意してください。

図 5-13. Qsys Component Library





-  IP 検索パスと Qsys システムへのパス情報を提供するために、IP インデックス・ファイル (.ipx) を使用することについて詳しくは「Quartus II ハンドブック Volume 1」の「[Creating Qsys Components](#)」の章の「Component Search Path」のセクションを参照してください。ライブラリにコンポーネントを追加することについて詳しくは「Quartus II ハンドブック Volume 1」の「[Creating Qsys Components](#)」の章の「Installing Additional Components」のセクションを参照してください。
-  Qsys を使用することについて詳しくは、アルテラ・ウェブサイトの「[Altera Training](#)」ページを参照してください。

改訂履歴

表 5-5 に、このドキュメントの改訂履歴を示します。

表 5-5. 改訂履歴

日付	バージョン	変更内容
2011 年 5 月	11.0.0	<ul style="list-style-type: none"> ■ ベータのステータスを削除。 ■ Verilog HDL または VHDL にシミュレーション・サポートを追加。 ■ テスト・ベンチ生成のサポートを追加。 ■ 生成ファイルとシミュレーションを变新。
2010 年 12 月	10.1.0	初版。

-  Quartus II ハンドブックの以前のバージョンについて、「[Quartus II Handbook Archive](#)」を参照してください。
-  このハンドブックの章について、「[online survey](#)」でフィードバックを提供してください。

