FPGA 設計入門

概要

Tutorial
TU0116 (v2.0) May 17, 2008

このチュートリアルでは、アルティウムの Innovation Station を使用した FPGA 設計の基本を説明します。 本書がカバーしているトピックは、Altium Designer を使った FPGA プロジェクトの生成、デスクトップ NanoBoard にプラグインされるドータボード上の物理的 FPGA デバイスに設計を書き込むこと、デザインプロセス - つまり、FPGA のプログラミングについて、です。 設計階層の使用、仮想計器についても、簡潔に説明します。

アルティウムの Innovation Station – Altium Designer とデスクトップ NanoBoard の強力なコンビは、FPGA 設計におけるデザインキャプチャ、インプリメント、テスト、デバッグにおいて必要とされるあらゆるツール、あらゆるテクノロジをリアルタイムで提供します。

アルティウムの Innovation Station は、最小限の労力でデバイスのインテリジェンスと機能にフォーカスすることができ、真の、持続可能な製品の差別化.を狙うことができます。 この革新的なデバイスにより、プロセッサや組込みソフトの世界に取りかかる前に、設計についての確実な基礎知識を習得することができます。つまり、基本的な設計をインプリメントし、デスクトップ NanoBoard にプラグインされた FPGA 上でそれを動作させる方法についてです。

このチュートリアルでは、カウンタベースの単純な（非プロセッサの）設計を扱います。デスクトップ NanoBoard 上のユーザ LED が左から右、または右から左へ順に点灯するように、ターゲットのドータボード FPGA にプログラミングします。 このチュートリアルのコースで、FPGA 設計の基本を身につけることができます。説明は以下のとおりです。

- Altium Designer における FPGA プロジェクトの生成と回路図ベースの設計をインプリメントする方法について。これには、部品の調達と、回路図上の配置、配線が含まれます。
- 設計をドータボードの FPGA にターゲッティングする方法。自動コンフィギュレーション機能を使用します。
- 設計のプロセス - コンパイル、回路の合成、ビルドなど、ターゲットデバイスのプログラムに使用するファイルを作成します。
- 設計階層を使った FPGA プロジェクト。簡単なカスタムロジック（HDL）を含みます。
- 仮想計器について。

このチュートリアルの設計例では、簡単なツイストリングカウンタを紹介しています（図 1）。 これは同期カウンタであり、反転させられた最終ステージの出力が開始ステージの入力に伝えられます。 個々のフリップフロップではなく、Altium Designer で提供されるシフトレジスタのコンポーネントを使用します。 基本となる回路図と追加のファイルについては、Altium Designer がインストールされているフォルダ内の `\Examples\Tutorials\Getting Started with FPGA Design` ディレクトリにあります。 サンプルは、いつでも参照し、さらに学習することができます。いくつかのステップは飛ばし読みしてもかまいません。

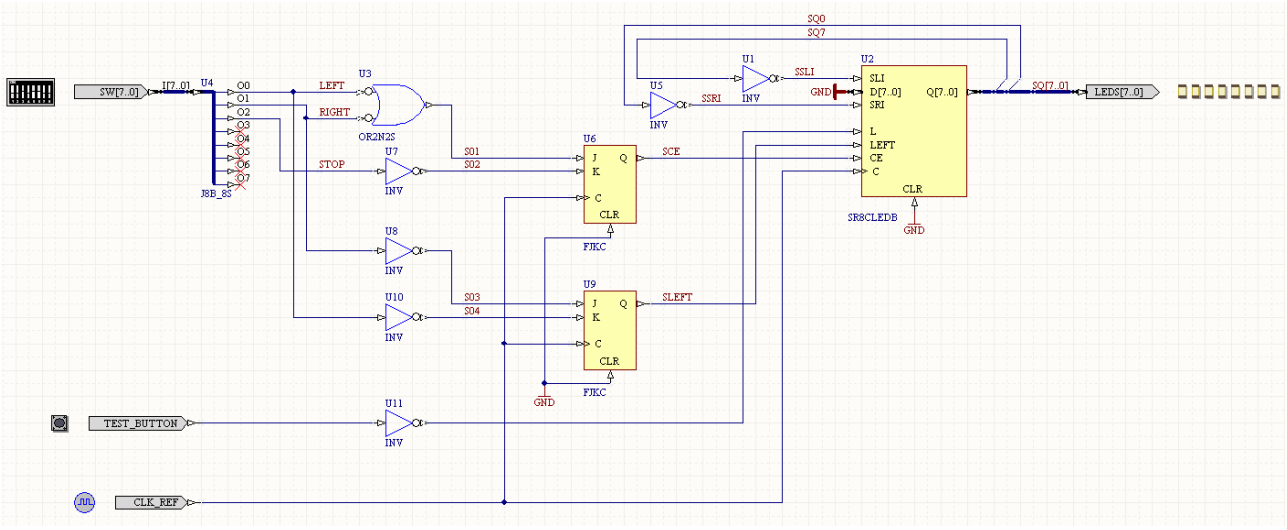


図1 シンプルで非プロセッサベースの設計 - ツイストリングカウンタ

回路内の論理コンポーネント用の同期クロック信号は、NanoBoard 上のリファレンスクロックから提供されます。 カウンタ出力は NanoBoard のユーザ LED に表示されます。

NanoBoard 上にあるリソースと一緒に使用できるデザイン内の追加ロジックは、以下のとおりです。

- **方向のコントロール** – NanoBoard 上で関連つけられたスイッチ（DIP スイッチの一部）の設定によって、カウントが左から右、または右から左へ進みます。
- **ストップコントロール** – NanoBoard 上で関連つけられたスイッチ（DIP スイッチの一部）の設定によって、カウントが止まったり、再開したりします。
- **クリアコントロール** – NanoBoard 上の `'DAUGHTER BD TEST/RESET'` ボタンの設定によって、カウンタ出力のクリア（全 LED がオフ）ができます。

Desktop NanoBoard, についての詳細は、*TR0143 Technical Reference Manual for Altium's Desktop NanoBoard NB2DSK01* を参照してください。

NanoBoardでサポートされているドータボードの範囲について、また、それぞれの仕様については、*www.altium.com/nanoboard/resources* を参照してください。

FPGAベンダツールについての重要事項

デスクトップ NanoBoard や、プラグインドータボードに常駐する物理的デバイスにデザインをダウンロードできるようになる前に、適切なベンダツールをコンピュータにインストールしておく必要があります。 これらのツールは、FPGA デザインをターゲットデバイスに配置配線するときに使用します。 FPGA ベンダのツールは、Altium Designer と一緒には提供されません。 各ベンダから独自に入手してください。

ドータボードについては、さまざまな種類をデスクトップ NanoBoard で利用することができます。 これらのドータボード上の FPGA デバイスは、商用ベースのツールだけでなく、各ベンダからウェブ経由でダウンロードできるツールによってもサポートされています。 選択したドータボードを使用するには、関連するツールをインストールする必要があります。

ベンダツールについての詳細は、各ベンダのウェブサイトを参照してください。

- Actel® Designer または、Libero® IDE については、*www.actel.com* をご覧ください。 ソフトウェアのダウンロードには、ライセンスが必要です。 ウェブのライセンスオプションをチェックしてください。
- Altera® Quartus® II については、*www.altera.com*をご覧ください。 アルテラの Quartus II ウェブエディション は、無料でダウンロードできます。ライセンスは必要ありません。
- Lattice® ispLever® については、*www.latticesemi.com* をご覧ください。 ispLever スタータソフトウェアのダウンロードには、ライセンスが必要です。 ウェブのライセンスオプションをチェックしてください。
- Xilinx® ISE™ については、*www.xilinx.com*をご覧ください。 Xilinx ISE WebPACK は、無料でダウンロードできます。ライセンスは必要ありません。

各ベンダのダウンロードツールへのリンクは、アルティウムのウェブサイトのベンダ情報のエリア でも、見つけることができます (*www.altium.com/Community/VendorResources*)。 このページは、Altium Designerから直接アクセスできます。メインの **Tools** メニューで、**Devices** ビューをアクティブ（**View » Devices View**）にして、**Vendor Tool Support** エントリを選択するだけです。

注記：アルティウムは、FPGA ベンダツールについての技術サポートは提供していません。 これらのツールのインストールについての詳細は、FPGA ベンダから提供される情報を参照してください。

デザインキャプチャ

まず最初にしなくてはならないのは、Altium Designer の環境に設計を取り込む（キャプチャする）ことです。 それはつまり、このチュートリアルで使用する回路デザインにとっては、必要なコンポーネントの回路図シートへの追加と、それらを順次配線していくことを意味しています。 回路図に取りかかる前に、その内容、つまり、プロジェクトを作成しましょう。 以下のセクションでご案内するのは、ツイストリングカウンタの設計をキャプチャするのに必要なステップです。

FPGAプロジェクトの作成

Altium Designer において、すべての設計作業のベースになるのが、プロジェクトファイルです。 FPGA デザインでは、FPGA プロジェクト（`*.PrjFpg`）を作成します。 プロジェクトドキュメントそれ自体は、ASCII ファイルです。プロジェクト情報などが収められており、プロジェクト、出力設定、コンパイル設定、エラーチェックの設定などのドキュメントなどで構成されています。

それでは FPGA プロジェクトを作成しましょう。

1. 新しい FPGA プロジェクトを作成します。**File » New » Project » FPGA Project** を使用してください。
2. **Projects** パネルの新しいプロジェクト（`FPGA_Project1.PrjFpg`）名の上で右クリックして、**Save Project** コマンドを選びます。 選んだ場所にプロジェクトを保存します。ファイル名 `Simple_Counter.PrjFpg` が `Basic FPGA Design Tutorial` という新しいフォルダに作成されます。

注 : スペース、あるいはハイフン（-）はプロジェクトファイル名、ドキュメントファイル名に使用しないでください。 違反すると、設計プロセスの途中で合成エラーに引っかかります。 アンダースコア（ _ ）で代用し、エラーを防ぐようにしましょう。

回路図ソースドキュメントの追加

FPGA プロジェクトは本来、階層構造になっています。 回路図、HDL（VHDL や Verilog）、OpenBus ドキュメントの数にかかわらず、階層下方のすべてのサブファイルを、シートシンボルで参照することができます。 全プロジェクトにわたる一般的な単位ですが、単一のトップレベルの回路図でなくてはなりません。 このシートは、デザイン用のポート - 設計がターゲットにしている FPGA デバイスの物理的なピンへのインターフェース - を含んでいるだけでなく、Altium Designer における FPGA と PCB の統合を容易にします。

このデザイン階層は、後ほど検討することにします。 今のところは、FPGA プロジェクトに単一の回路図シート（トップシート）を追加するだけにしておきましょう。

1. 新しい回路図ドキュメントを追加します。**Projects** パネルの FPGA プロジェクトのエントリ上で右クリックし、**Add New to Project » Schematic** コマンドを選びます。 ブランク（空白）の回路図シートが、アクティブなドキュメントとして、メインのデザインウィンドウに開かれます。
2. このドキュメントを保存（**File » Save**）します。`Simple_Counter.SchDoc` という名前のファイルが、親になるプロジェクトと同じフォルダに作成されます。
3. プロジェクトそれ自体は、**Projects** パネルに変更されて表示されます。 プロジェクトも保存します。プロジェクト名の上で右クリックし、**Save Project** を選んでください。

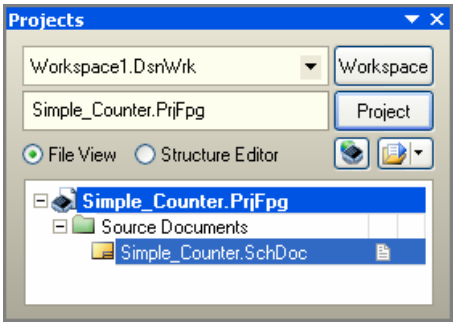


図2 新しい FPGA プロジェクト、ソース回路図を追加済み

コンポーネントの配置

これで、「真っ白なカンバス」が手に入りました。必要なコンポーネントを追加していきましょう。コンポーネントは、デザイン回路の機能を表すものであり、デスクトップ NanoBoard NB2DSK01 上にあるリソースとのインターフェースを提供します。

表 1 はツイストリングカウンタの回路を定義するのに必要なコンポーネントを定義しています。 これらのコンポーネントはすべて、FPGA Generic 統合ライブラリ（FPGA Generic.IntLib）の中で見つけることができます。ライブラリは、Altium Designer がインストールされているフォルダの \Library\Fpga ディレクトリにあります。

表1. ツイストリングカウンタの回路図に必要なデザインコンポーネント

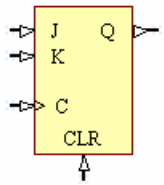
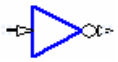
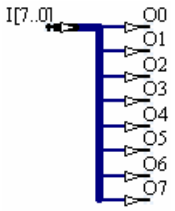
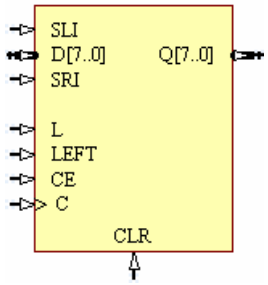
シンボル	コンポーネント名	内容	必要な数
	FJKC	J-K フリップフロップ 非同期クリア付	2
	INV	反転器	6
	J8B_8S	Bus Joiner – 8 ピン入力バス、8 個のシングルピンへの出力用	1
	OR2N2S	2 入力の OR ゲート、Low A、Low B 入力（シングルピンバージョン）	1
	SR8CLED	8 ビットの読取可能なシリアル/パラレル-入力、パラレル出力の双方向シフトレジスタ、クロック同期可能、非同期 クリア（バスバージョン）	1

表 2 は、デザインに必要なインターフェースコンポーネントを表しています。 これらのコンポーネントは、一般的に、ポートのコンポーネント（ポートプラグイン）として参照され、デスクトップ NanoBoard 上の関連するリソースと、ターゲットにしている FPGA ドータボードの物理的な I/O ピンの間の接続を自動的に確立します。 デザインを俯瞰する立場から見ると、それらはデザインの拡張と言えます。つまり、デザインからの信号が接続されるターゲットデバイスの物理的なピン、と言うことです。 これらのポートコンポーネントはすべて、FPGA NB2DSK01 Port-Plugin 統合ライブラリ（FPGA NB2DSK01 Port-Plugin.IntLib）の中で見つけることができます。ライブラリは、Altium Designer がインストールされているフォルダの \Library\Fpga ディレクトリにあります。

表2. ツイストリングカウンタの回路図に必要なポートコンポーネント

シンボル	コンポーネント名	内容
	CLOCK_REFERENCE	このコンポーネントは、デスクトップ NanoBoard 上の 20MHz 固定のシステムクロック信号のインターフェースです。 この信号を使って、同期クロック信号をフリップフロップとシフトレジスタのロジックに提供します。

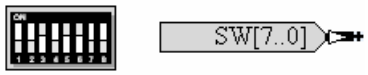
	DIPSWITCH	このコンポーネントは、Desktop NanoBoard 上の DIP スイッチのインターフェースです。 これらのスイッチのうち、3 つを使用してカウンタ（の点灯、左または右）の方向、または停止をコントロールします。
	LED	このコンポーネントは、Desktop NanoBoard 上のユーザ LED とインターフェースします。 LED は、カウンタの出力を視覚的に表示するために使用します。
	TEST_BUTTON	このコンポーネントは、Desktop NanoBoard 上の 'DAUGHTER BD TEST/RESET' ボタンとインターフェースします。 この（反転）信号は、シフトレジスタ上のロード可能なピンへの制御入力として使用します。、 このレジスタの D 入力を GND に接続するときに、このボタンを押すと、レジスタのデータ出力に'0'がロードされます。

図 3 に示すように、これらすべてのコンポーネントを回路図シート上に配置します。 上記二つの統合ライブラリ（FPGA Generic と FPGA NB2DSK01 Port-Plugin）がインストールされ、**Libraries** パネルからデフォルトで利用できれば OK です。 このチュートリアルではそれ以上のライブラリは使用しません。 あとは、関連するライブラリをパネル内でアクティブにして、コンポーネントエントリをリストに選択します。パネル右上の **Place** ボタンをクリックするか、コンポーネントを直にシート状へクリック&ドラッグしてください。 回路図配置のコントロールでは、反転や回転など、必要に応じた細かい調整ができます。

一度配置し、各コンポーネントの指定を終わらせます。**Tools » Annotate Schematics Quietly** コマンドを使ってください。

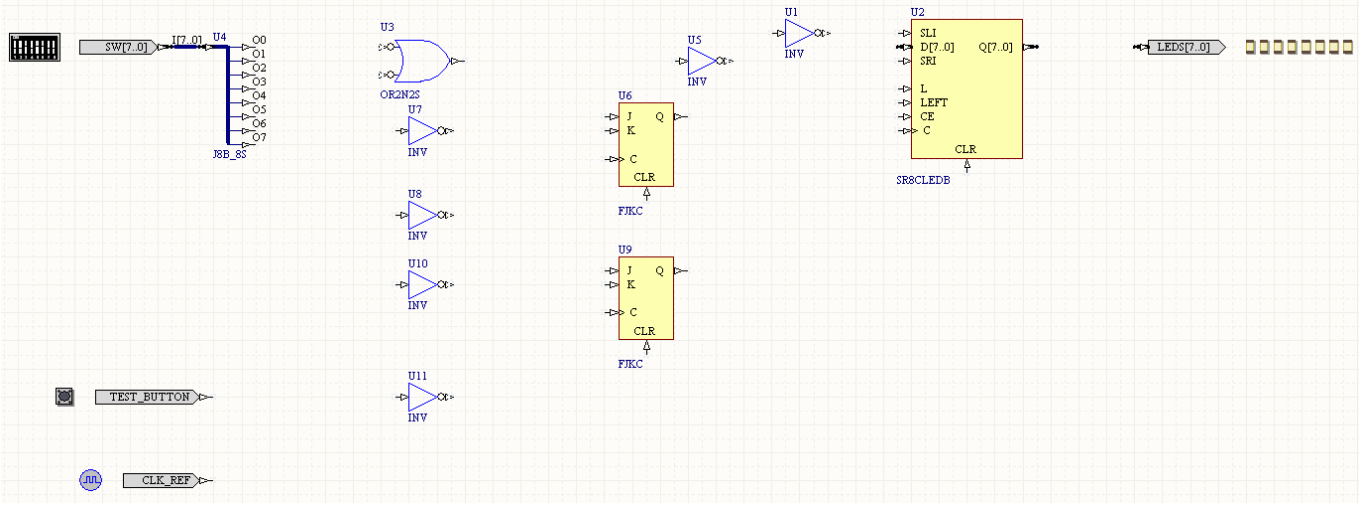


図 3 回路図シートへのコンポーネントの初期配置（Simple_Counter.SchDoc）

デザインの配線

ここまでで、すべてのコンポーネントが配置されました。今度は、それらをお互いに接続していきましょう。デザインに接続性を追加します。

1. それでは、配線しましょう。図 4 のように、**Place » Wire** コマンドと **Place » Bus** コマンドを使用します。コマンドは、**Wiring** ツールバーから、 と ボタンでそれぞれ、利用できます。

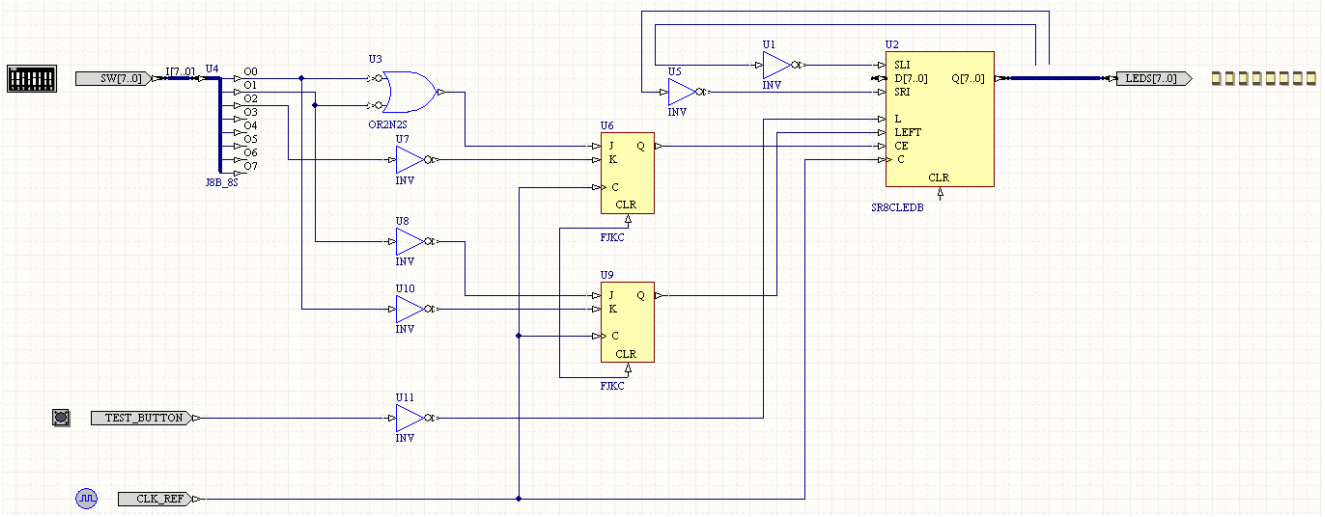


図 4 初期の配線状態 - ワイヤとバスの配置

2. **Wiring** ツールバーから、 ボタンをクリックして、GND 電源ポートを配置します。 ポートがカーソルの上に浮いている状態で、**TAB** キーを押します。 表示される *Power Port* ダイアログで、**Style** 属性を `Bar` に切り替えます。 2 つのフリップフロップの CLR ピンに接続しているワイヤの左下につながるように、ポートを配置します。

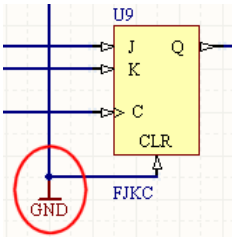


図5 フリップフロップ CLR 信号を GND に接続

3. 別の GND 電源ポートを配置します。再度 `Bar` を使って、シフトレジスタの CLR ピンにつながるようにしてください。
4. **Wiring** ツールバーから、 ボタンをクリックして、GND バス電源ポートを配置します。 ここでも、ポートの **Style** 属性を `Bar` に変更します。 ポートを配置して、シフトレジスタの D[7..0]ピンにつながるようにします。 必要に応じ、**Spacebar** を押して、ポートを回転させます。

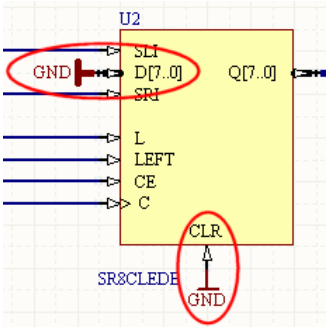


図6 シフトレジスタを GND に接続

5. ここで、シフトレジスタの出力バスからそれぞれの信号を取り出すため、バスエントリを追加する必要があります。 これらのエントリには、各自の変換器経由で、シフトレジスタの SLI と SRI 入力につながるそれぞれのワイヤを接続する必要があります。 **Wiring** ツールバーで ボタンをクリックし、2 つのバスエントリを配置します（図 7）。

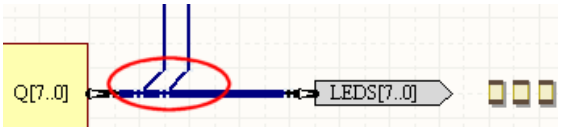


図7 バスエントリの追加

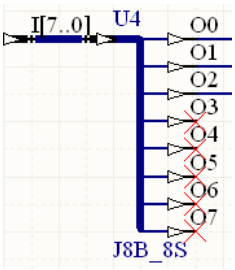


図8 ERC マーカのない終端の仕上げ

6. デザインをすっきりとさせ、コンパイル時の警告を最小限にとどめるには、No ERC マーカをそれぞれ、未使用のバス用コンポーネント（`U4`）の出力に配置しておくと良いでしょう。 **Wiring** ツールバーで、 ボタンをクリックし、No ERC マーカの配置モードに入ります。 未使用のピン O3 から O7 までのそれぞれに、マーカの位置を決め、配置します（図 8）。

回路内のキーネットにラベルを貼り付けて、デザインの配置を完了させましょう。 ラベルを貼ることで、設計の内容が理解しやすくなり、コンパイル時のデザインチェックには、問題の追跡に大いに役立ちます。

7. デザインにネットラベルを貼り付けましょう。図 9 のように、**Place » Net Label** コマンドを使用します（あるいは、**Wiring** ツールバーの Net ボタンをクリックします）。 図 9 のネットラベルは、サンプルです。 重複さえしなければ、別の名前をネットラベルとして使うこともできます。

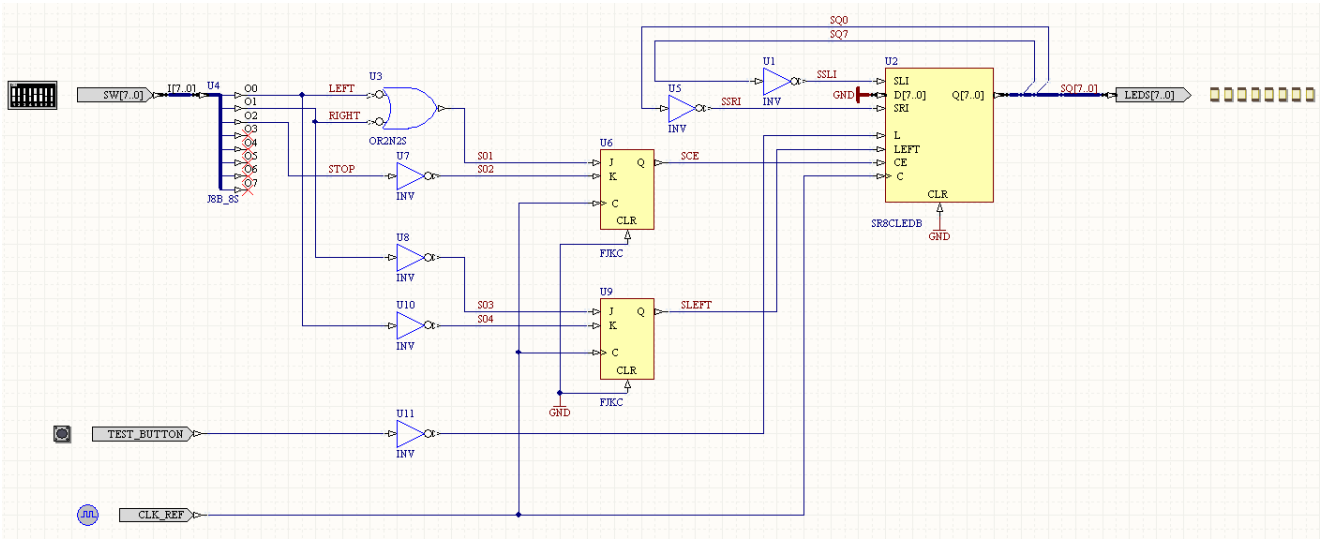


図 9 最終のデザイン - 配線終了、ネットラベル添付済み

8. 回路図とその親プロジェクトを保存します（**File » Save All**）。

お疲れ様でした。 これで、すべてのデザインを回路図上に取り込む（キャプチャする）ことができました。 このあとは、デザインを検証し、実際の FPGA デバイスをターゲットにしたダウンロードに移ります。

デザインの検証

作成したデザインを目的の FPGA デバイスに合わせてダウンロードする前に、インテグリティ（全体や信号の整合性）を検証しておくことをお勧めします。 検証行うには、Altium Designer の強力なデザインコンパイラを起動する必要があります。 コンパイルのプロセスは、プロジェクトの正しいネットリストを生成する上で、欠くことができません。 *Options for FPGA Project* ダイアログ（**Project » Project Options**）の **Error Reporting** や **Connection Matrix** タブで定義されるオプションに従って、コンパイラは、電子的エラーや図面上のエラーを幅広くチェックします。

注 : このチュートリアルでは、それらのタブの変更は特にありません。 デフォルト設定で十分です。

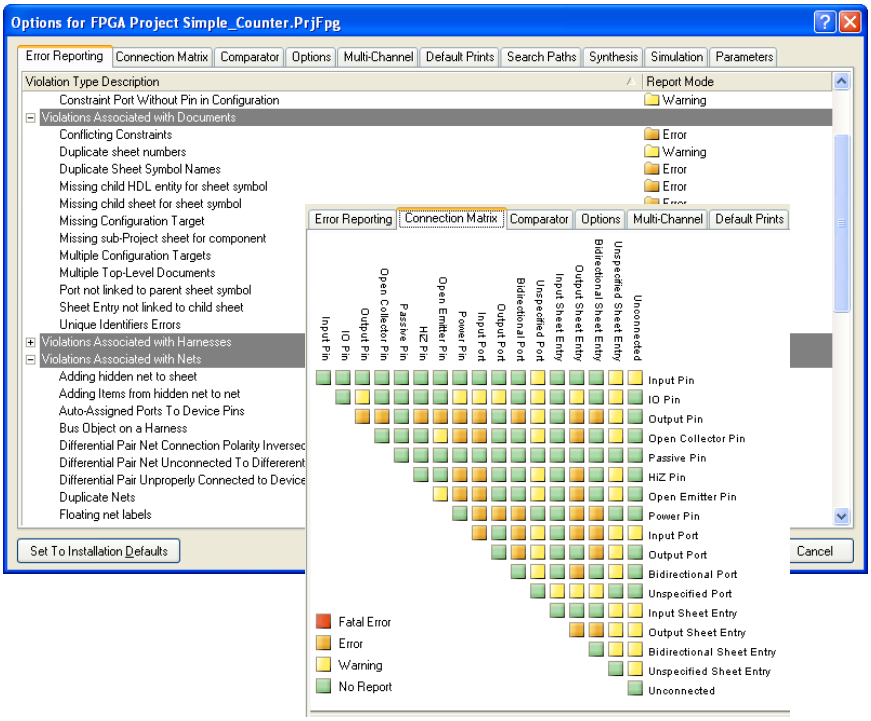


図10 コンパイラの設定は、Options for FPGA Project ダイアログの Error Reporting や Connection Matrix タブで定義されます

1. メインの回路図メニューから、**Project » Compile FPGA Project Simple_Counter.PrjFpg** を選びます。 プロジェクトのコンパイルへ進みます。
2. 警告やエラー、致命的な障害が発生すると、**Messages** パネル内のリストに記録されます。 エラーや致命的な障害がコンパイル中に発生したら、このパネルは自動的に表示されます。 発生したのが警告だけなら、このパネルは手動で表示する必要があります。メインデザインウィンドウ下部の **System** ボタンをクリックし、表示されるメニューから **Messages** を選びます。

   エントリされたメッセージをダブルクリックすると、**Compile Errors** パネルの中に、エラーについての情報ウィンドウがさらに表示されます。 重要な違反については、回路図上でズームされ、ハイライトされます。
3. 正しく配線された回路図（`Simple_Counter.SchDoc`）については、ロードされない信号（図 11）に関連する警告メッセージの数にだけ注目してください。 これらが発生するのは、SQ0 と SQ7 を、SQ1 から SQ6 ではなく、バス SQ[7..0] から取り出しているからです。 このような警告であれば、無視することができます。

Messages

Class	Document	Source	Message	Time	Date	No.
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[1] has no load	2:56:33 PM	18/04/2008	1
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[2] has no load	2:56:33 PM	18/04/2008	2
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[3] has no load	2:56:33 PM	18/04/2008	3
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[4] has no load	2:56:33 PM	18/04/2008	4
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[5] has no load	2:56:33 PM	18/04/2008	5
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[6] has no load	2:56:33 PM	18/04/2008	6
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[1] has no load	2:56:33 PM	18/04/2008	7
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[2] has no load	2:56:33 PM	18/04/2008	8
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[3] has no load	2:56:33 PM	18/04/2008	9
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[4] has no load	2:56:33 PM	18/04/2008	10
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[5] has no load	2:56:33 PM	18/04/2008	11
[Warning]	Simple_Counter.SchDoc	Compiler	Signal NamedSignal_SQ[6] has no load	2:56:33 PM	18/04/2008	12

図 11 正しく配線されたデザイン、コンパイル後に現れるメッセージ

これとは異なるメッセージが表示された場合は、問題を解決し、デザインプロジェクトをもう一度コンパイルし直してください。

コンパイル時のエラーについての詳細情報やどのように発生するのか、その解決方法などについては、*TR0142 Project Compiler Error Reference* を参照してください。

4. 回路図とその親プロジェクトを保存します。

物理 FPGA デバイスをターゲットとして指定する

ここまででデザインの作成段階は終了したため、使用する物理 FPGA デバイス (デザインのターゲットと、デザインを最終的にプログラムし実行するメディア) を指定する必要があります。 このチュートリアルでは、Desktop NanoBoard NB2DSK01 に挿入する、3 コネクタ付きドータボード上の FPGA デバイスをターゲットに指定します。

物理的インプリメンテーションのためのデザインのマッピングや制約条件のプロセスは、制約条件ファイルの作成を通じて実行されます。 合成に必要な最小限の情報はデバイスの仕様です。

制約条件ファイルのセットは、制約条件ファイルの名前の単なるリストであるコンフィギュレーションを生成することで、デザインのターゲッティングを行っています。

デスクトップ NanoBoard NB2DSK01 への配置における制約条件のシステムは、さまざまな制約条件ファイルをカバーしています。

- リソースと、NB2DSK01 マザーボード、サテライト周辺デバイス、ドーターボードへのピンマッピング
- サテライトボード（周辺ボードとドータボード）から NB2DSK01 マザーボードへの接続

デスクトップ NanoBoard を ターゲットにした FPGA デザインプロジェクト は手動でもコンフィギュレーションできます。コンフィギュレーションを追加し、必要なボード制約条件ファイルを割り当て、マッピング制約条件を手入力すれば、プロセスは、自動コンフィギュレーション機能を使うより、ずっとシンプルなものになります。

この機能を使うと、FPGA デザインプロジェクト用のターゲットコンフィギュレーションは自動的に生成されます。 必要なボードレベルの制約条件ファイルは、その後、自動的に定義され、システム内で見つかったハードウェア（マザーボード、ドータボード、ペリフェラルボード）をベースにして、コンフィギュレーションに追加されます。 追加マッピングの制約条件ファイルもまた、生成され、コンフィギュレーションに追加されます。これにより、システム内（ドータボードやペリフェラルボード）に検知されたサテライトボードとマザーボードとの接続が取り扱われることになります。

FPGA デザインプロジェクトを自動的にコンフィギュレーションできるようにするには、システムで使用中のハードウェアを特定できる Altium Designer の能力がカギになります。 ハードを特定する機能は、NB2DSK01 マザーボード、3 コネクタドータボード、ペリフェラルボードそれぞれの上にある 1 ワイヤベースのメモリデバイスの使用で可能になります。

注 : 古い 2 コネクタのアルティウムのドータボードも、デスクトップ NanoBoard NB2DSK01 と一緒に使用することができます。ただし、自動コンフィギュレーションに必要なメモリデバイスは保持しません。

コンフィギュレーションと制約の概念、およびデザインの移植性におけるそれらの役割の詳細は、*AR0124 Design Portability, Configurations and Constraints* を参照してください。

自動設定など、デスクトップ NanoBoard NB2DSK01 の制約システムの詳細は、*AP0154 Understanding the Desktop NanoBoard NB2DSK01 Constraint System* を参照してください。

それでは FPGA プロジェクトを設定します。

1. 自動コンフィギュレーション機能を使う前に、以下を確認してください。
   - デザインのターゲットとなる、FPGA デバイスが搭載された 3 コネクタ付きドータボードが、NB2DSK01 マザーボードに挿入されていること。
   - 我々の単純なカスタムデザインでは、プラグインペリフェラルボード上に常駐するリソースを使用しません。 これらのボードは、マザーボードに接続したままでも、取り除いてもかまいません。
   - NanoBoard が USB (またはパラレル) 接続を経由して PC に接続され、電源が投入されていること。
2. **Devices** ビューを開きます（**View » Devices View**）。 **Live** オプションを有効にし、**Connected** の表示が緑になっていることを確認します。
3. 自動コンフィギュレーション機能にアクセスできるようになります。実行には 2 つの方法があります。 このチュートリアルの目的にとっては、どちらの方法にも考慮する価値があります。 ここでは、以下の方法のうち、ひとつだけを取り上げて自動コンフィギュレーションプロセスを実行します。

   方法 1 デスクトップ NanoBoard のアイコン（ビュー内の NanoBoard チェーン）の上で右クリック、表示されるメニューから 、`Configure Fpga Project » Simple_Counter.PrjFpg` を選びます（図 12）。

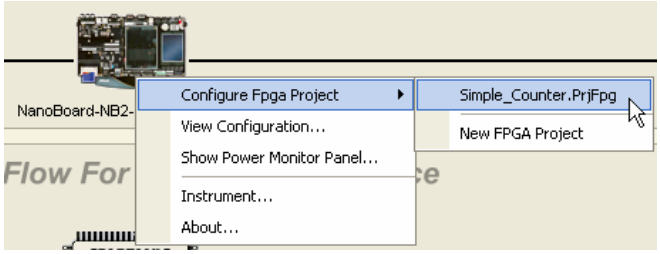


図 12 自動コンフィギュレーション、Devices ビューから指示

方法 2 デスクトップ NanoBoard のアイコンの上で右クリックし、NanoBoard Controllers パネルの Instrument Rack 関連の計器にアクセスします。 その後、Board View ボタンをクリックして、*NanoBoard Configuration* ダイアログにアクセスします。 Auto Configure FPGA Project のドロップダウンをダイアログの左下から選択して、`Simple_Counter.PrjFpg`

（図 13）を選びます。 ダイアログを使用することで、既存のデスクトップ NanoBoard NB2DSK01 システムの仮想的な（ダイナミック）概要を表示できます。 ダイアログ内のイメージは、マザーボード NB2DSK01 に物理的に接続されている特定のペリフェラルボードとドータボードを表しています。

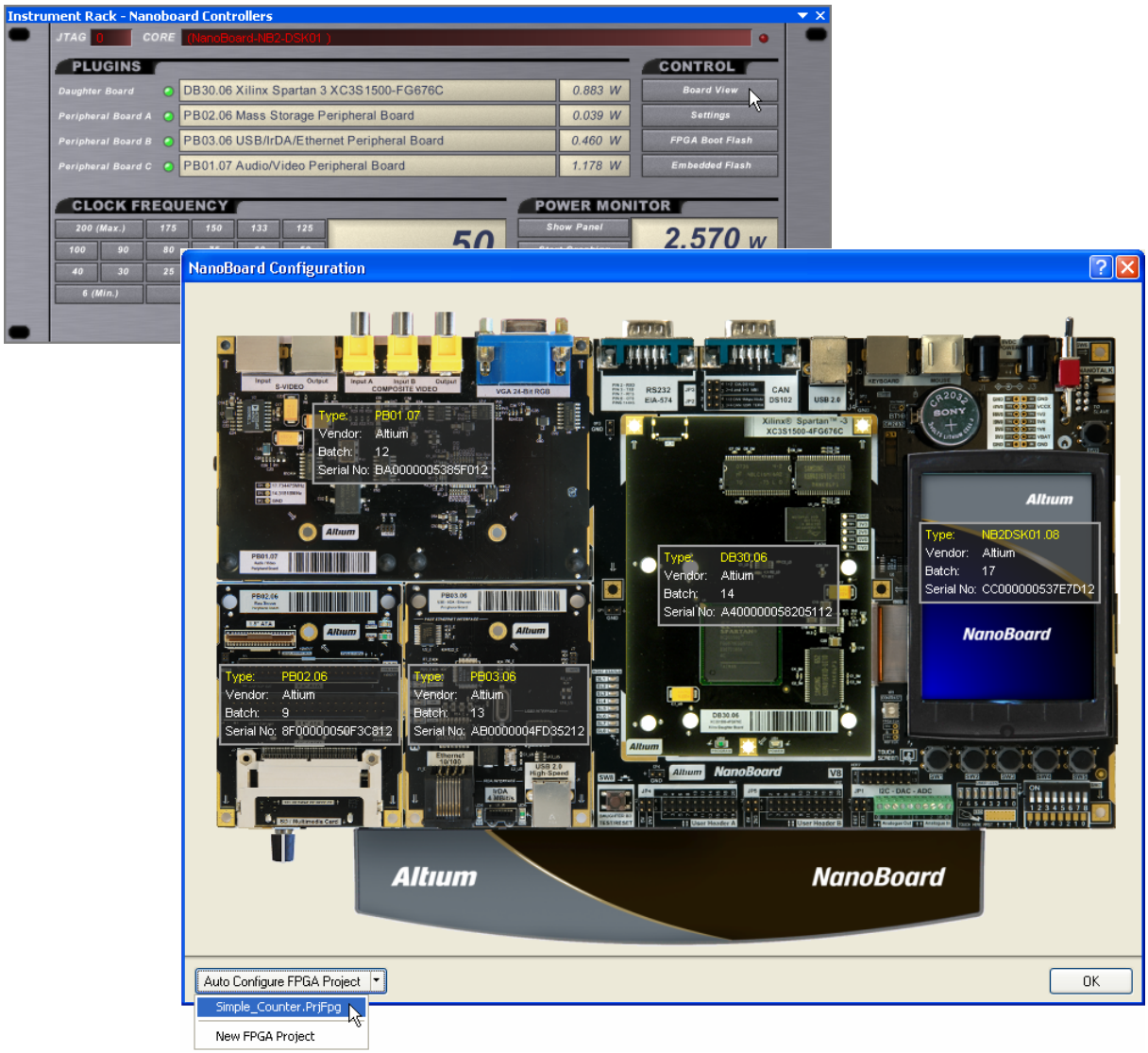


図13 仮想ベースの NanoBoard Configuration ダイアログからの自動コンフィギュレーション

自動コンフィギュレーションプロセスが実行されます。以下を参照してください。

- コンフィギュレーションが生成されると、Simple_Counter プロジェクトに追加されます。 コンフィギュレーションの名称は、デスクトップ NanoBoard と 使用されているドータボードのバージョンに依存します。 生成されるフォーマット名は次のとおりです。

  `motherboard code_revision_daughter board code_revision`

  例えば、デスクトップ NanoBoard NB2DSK01（レビジョン 8）で、ザイリンクス Spartan-3 のドータボード DB30 （レビジョン 6）を使用している場合、コンフィギュレーションは、 `NB2DSK01_08_DB30_06` という名前になります。
- 制約条件ファイルは、その後、システム内に検出されたそれぞれのボード（マザーボード、ドータボード、ペリフェラルボード）用のコンフィギュレーションに追加されます。 これらは、Altium Designer がインストールされているフォルダの `\Library\Fpga\NB2 Constraint Files` ディレクトリにソースがあります。 各ケースで使用されるファイルは、ボードのタイプとバージョンで決まります。 例えば、ザイリンクスの Spartan-3 ドータボード DB30 （レビジョン 6）を使用する場合、読み出されて、コンフィギュレーションに追加される制約条件ファイルは、`DB30.06.Constraint` になります。
- ドータボードとペリフェラルボードのマザーボードへのマッピングを定義する制約条件ファイルもまた、作成されて、オンザフライで（ただちに）、コンフィギュレーションに追加されます。 このファイルの名前は、末尾に '`_BoardMapping`' の付いた、コンフィギュレーションそのものの名前になります（例えば、`NB2DSK01_08_DB30_06_BoardMapping.Constraint` など）。 ファイルは、プロジェクトファイル（`Simple_Counter.PrjFpg`）と同じ場所に保存されます。

4. コンフィギュレーションと制約条件ファイルの割り当てが、*Configuration Manager* ダイアログにリスト表示されます（図 14）。

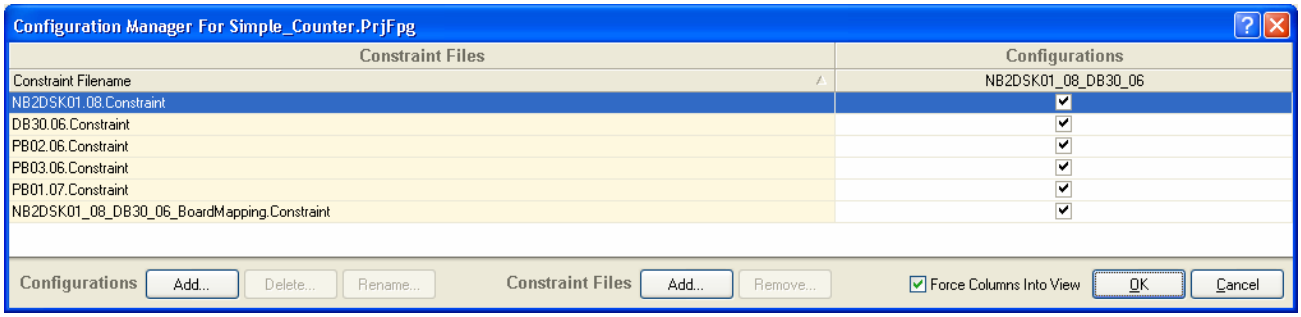


図14 コンフィギュレーションと制約条件ファイルの構成（Simple_Counter プロジェクト）

OK をクリックします。 `Settings` という名前のサブフォルダがプロジェクトに追加されて、**Projects** パネルにリスト表示されます。 制約条件ファイルそれ自体は、`Constraint Files` サブフォルダにリストされます。

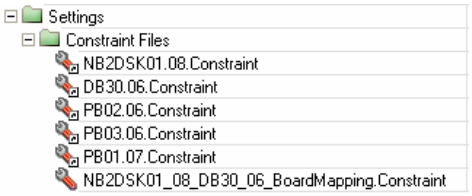


図15 コンフィギュレーションに追加された制約条件ファイル

5. プロジェクトファイルを保存します。

これで、デザインのコンフィギュレーションは完了です。 ドータボード FPGA デバイスをターゲットにして、プロセスを開始、最終的に、デバイスをプログラムします。

デザインの処理

FPGA ベースデザインのキャプチャが終了したら、次は、ソースファイルを処理する論理ステップに移ります。 これは、デバイスベンダの配置配線ツールに入力するソースネットリストのファイルを得て、デザインをコンパイル、合成するプロセスです。

選択された物理デバイスにデザインがフィットし、FPGA プログラミングファイルを生成できるように、プロセスは配置配線ツールの実行を継続します。 このプログラミングファイルは、プロセスチェーンの最終ステップ、つまり、物理的な FPGA デバイスへのデザインのプログラミング、へつながります。

全体的なプロセスの流れは、ソースファイルのキャプチャから物理的デバイスのプログラミングまで、**Devices** ビューから実行されます。

1. Altium Designer の **Devices** ビューがアクティブ（**View » Devices View**）になっていることを確認してください。
2. **Live** オプションが有効で、**Connected** 表示が緑であることを確認してください。

このビューは、検出された物理的 FPGA デバイスに関連するコントロールを行います。このチュートリアルで説明するハードデバイスチェーンでは、物理的デバイスの 'Process Flow' として参照されています（図 15）。

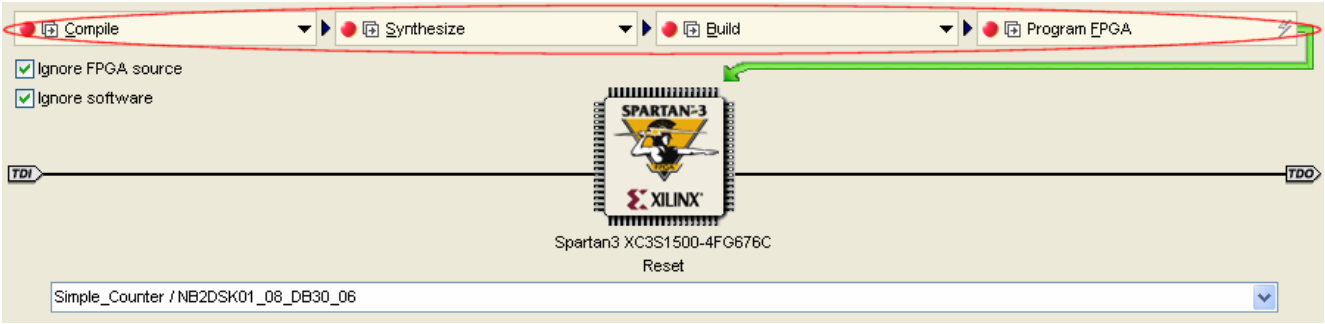


図16 物理的FPGA デバイスに関連する Process Flow

この Process Flow で は、以下の追加条件に合致するものについてだけ、説明しています。

- 使用するドータボートに合った FPGA デバイス用のベンダツールがインストールされていること。
- チェーン内のデバイス用アイコンの下のドロップダウンフィールドに、適切な `Project / Configuration` ペアが表示されていること プロジェクトに、検出された物理的デバイスへのデザインをターゲットにした、制約条件ファイルが含まれているコンフィギュレーションがある場合には、適正なペアが存在します。

自動コンフィギュレーション機能を実行していた場合は、ドータボードデバイスをターゲットにした制約条件ファイルを含む、このチュートリアル用の適切なコンフィギュレーションがすでに存在しています。 したがって、`Project / Configuration` のエントリは、`Simple_Counter / ConfigurationName` として表示されます（例えば、`Simple_Counter / NB2DSK01_08_DB30_06` など）。図 15 を参照。

Process Flow そのものは、4 つの別々のステージから成り立っていて、各ステージの出力が次のステージの入力になります。 Process Flow の全体は **Progran FPGA** ボタンを直接クリックして実行させることができます。ただし、フローの最後のステージでは、各ステージを順番に実行することも大切です。

ここでは、フローのすべてのステージを実行することができます。現在のステージについては、ステージボタン の左側の矢印アイコンをクリックして含めるようにしてください。

3. **Compile（コンパイル）**ボタンをクリックします。 このステージでは、関連するＦＰＧＡプロジェクト内のソースドキュメントをコンパイルします。 自分のプロジェクトにプロセッサコアが含まれている場合は、関連する組込みソフトウェアプロジェクトもこのステージで一緒にコンパイルします。チュートリアル用のシンプルな例は、プロセッサのないデザインになっているので、このステージの Design Compiler は単に電気的エラーと作図エラーを検証するためだけに使用します。 この節にたどり着くまでに、すでにプロジェクトのコンパイルを行っているので、このステージは楽勝でしょう。

   コンパイルのプロセスにおけるフィードバックは **Messages** パネルで見ることができます。 いずれかのステージがちゃんとコンパイルできれば、関連するインジケータが緑に変わります。

4. **Symthesize（論理合成）**ボタンをクリックします。 論理合成中に、ソースドキュメントは中間 VHDL ファイルに変換され、その後、ベンダの配置配線ツールに最適な形で、トップレベルの EDIF ネットリストが合成されます。 特定の FPGA デバイスをターゲットにして、デザイン内のコンポーネントに関連する合成モデルファイルが検索され、該当の合成出力フォルダにコピーされます。

   合成が成功すると、`Generated [ConfigurationName]` という名前のフォルダが作成されます。生成されたフォルダには、EDIF（`Simple_Counter.edf`）、VHDL（`Simple_Counter.vhd`）、合成ログ（`Simple_Counter_Synth.log`）があります。

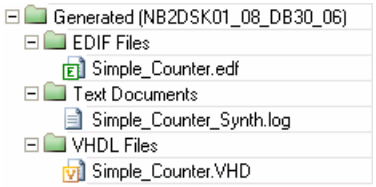


図17 フローの Ｓｙｎｔｈｅｓｉｓ（論理合成）ステージで生成されたファイル

   論理合成のプロセスにおけるフィードバックは **Messages** パネルで見ることができます。

5. **Build（ビルド）**ボタンをクリックします。 このステージでは、ターゲットの物理的ＦＰＧＡデバイス用のデザインを完成させるため、Altium Designer はベンダの配置配線ツールを呼び出して実行します。

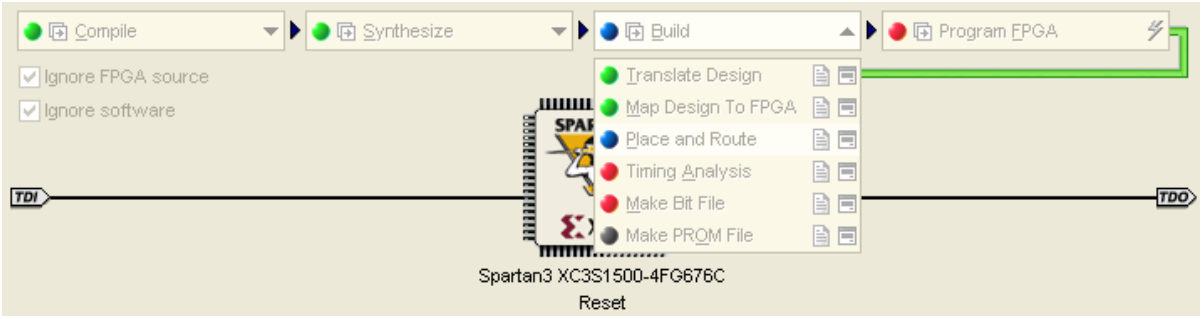


図 18 Build プロセスにおける配置配線ステージの開始

以下のサブステージ（5 つ）がそれぞれ順に実行されます。

- **設計の変換** Process Flow の論理合成のステージで生成されたトップレベルの EDIF ネットリストと、関連合成モデルファイルを使用して、NGD（Native Generic Database）フォーマットのファイルを生成します。
- **FPGA へのデザインマッピング** FPGA のプリミティブにデザインがマッピングされます
- **配置配線** 低レベルのデザイン記述を（マッピングのステージから）利用して、FPGA 内に目的の論理を配置する方法を導き出します。 配置後に目的の相互接続が配線されます。
- **タイミング解析** 定義された任意のタイミングにおける制約条件下で、設計のタイミングを解析します。 特定の制約条件がない場合（このチュートリアルには該当しません）、デフォルトのリストが使用されます。
- **BIT ファイルの作成** デザインを物理デバイスにダウンロードするのに必要なプログラミングファイルを生成します。

ビルドのプロセスにおけるフィードバックは **Messages** パネルで見ることができます。 ベンダに関する詳細なフィードバックについては、**Output** パネル（メインデザインウィンドウの下側にある **System** ボタンに関するメニューからアクセス）で見ることができます。

ステージがちゃんと終了したら、*Results Summary* ダイアログが表示されます。 このダイアログは、ターゲットデバイス内で使用されるそれぞれのリソースと関連付けた要約情報を、任意のタイミング情報とともに提供します。 このダイアログを閉じます

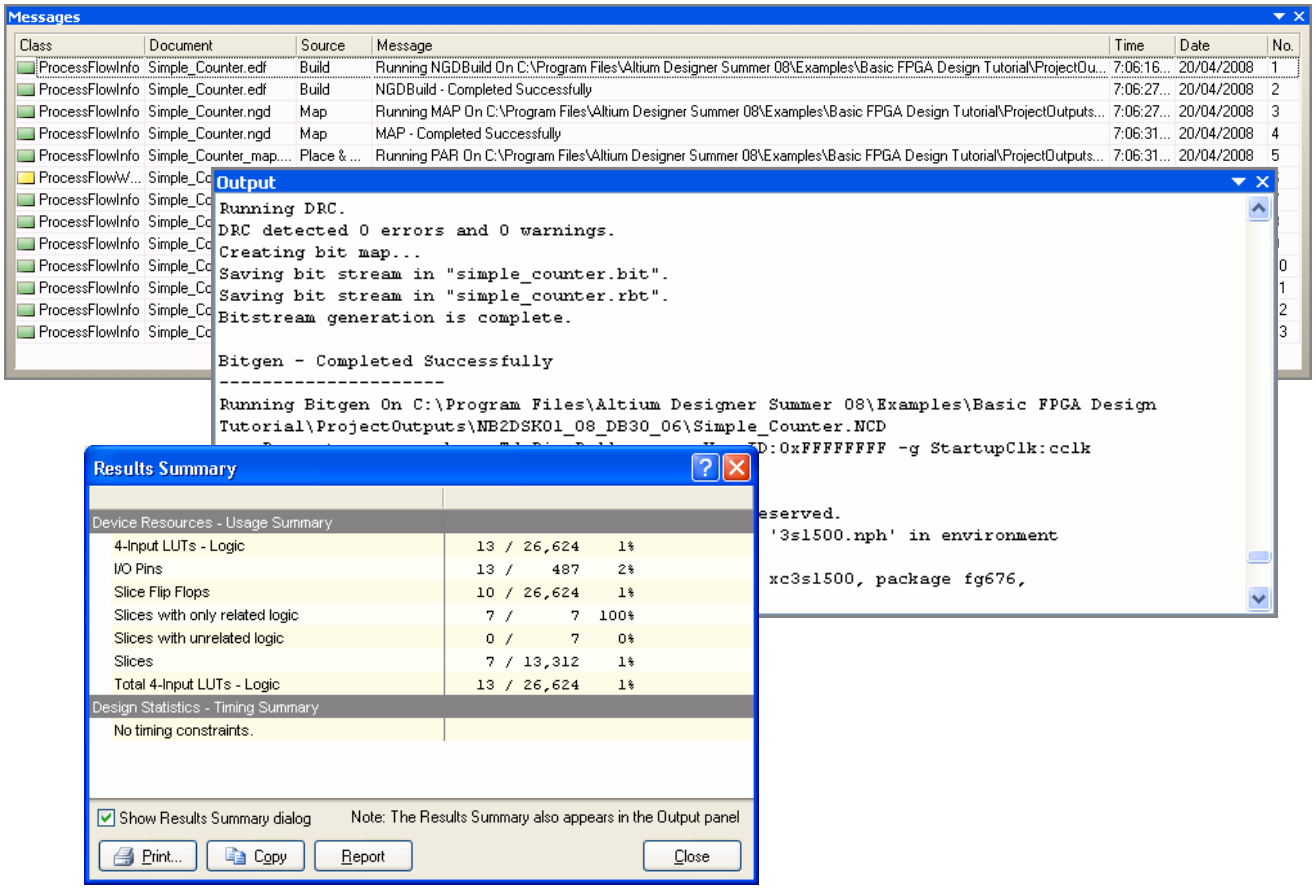


図19 ビルドプロセスの情報にアクセスし、ビルド結果の最終レポートを確認

6. **Program FPGA** ボタンをクリックします。 ビルドのステージで作成されたプログラミングファイルは、ドータボード上の物理的 FPGA デバイスにダウンロードされます。 ダウンロードは、PC からデスクトップ NanoBoard への JTAG リンクを通して実行されます。その過程は、Altium Designer のステータスバーに表示されます。

   デザインがダウンロードされると、Devices ビューの物理デバイスアイコンの下にあるテキストが `Reset` から `Programmed` に変わります。ハードウェア側では、ドータボードの `'Program'` LED が緑に点灯し、デザインが物理デバイスにロードされたことを示します。

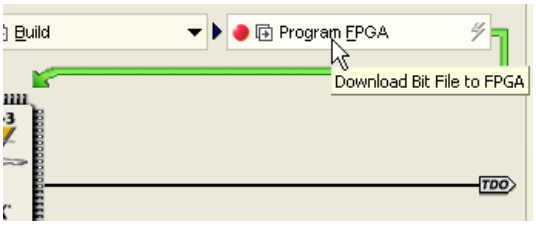


図20 プログラミングプロセスの開始

プロセスフローと **Devices** ビューの詳細は、 *AP0103 Processing the Captured FPGA Design* を参照してください。

7. NanoBoard 上の DIP スイッチを以下のように設定してください。
   - *Switch 8*: ON にしてください。止まっていたカウンタがスタートします。あるいは、LED が左から移動します。
   - *Switch 7*: ON にしてください。止まっていたカウンタがスタートします。あるいは、LED が右から移動します。
   - *Switch 6*: On にしてください。カウンタがストップします。 カウントが停止します。スイッチ 7、8 は OFF になります。
8. NanoBoard 上の `'DAUGHTER BD TEST/RESET'` ボタンを押して、LED をクリアします。
9. プロジェクトを保存します。

デスクトップ NanoBoard 上のユーザ LED がすべて同時にオンになります。これは、ツイストリングカウンタを構成するという当初の目的とは違っています。 これは、NanoBoard 上のリファレンスクロックが 20MHｚであるためです。 ここは、Mhz のファクタでクロックをスローダウンさせ、LED が順に表示されるようにする必要があります。 チュートリアルの次のセクションでは、いくつかの分周器を追加して、この課題をクリアしていきます。その上で、FPGA デザインにおける階層の使用についても検討します。

デザイン階層の検討

FPGA プロジェクトファイル（*PrjFpg）はさまざまなソースドキュメントとリンクすることで単一のプロジェクトとしてまとめられています。したがって、ドキュメント間やネット接続などの関連はドキュメント自体の情報で定義されています。

階層のあるデザインでは、デザインは論理ブロックで仕切られており、それぞれのブロックがトップの回路図シートにシートシンボルとして表現されています。 各シートシンボルの **Filename** の属性は、それが表している元のデザインファイルを参照しています。 元のデザインファイルには、以下のものがあります。

- 回路図シート
- OpenBus システムのドキュメント
- VHDL ファイル
- Verilog ファイル

回路図のサブシートにも、さらに下位のデザインファイルを参照しているシートシンボルが含められていることがあります。このアプローチで、任意の深さや組合せを持つデザイン階層 を作成することができます。

階層になったネットやバスとドキュメントとの接続性は、標準的な階層プロジェクトの接続の動作に従っています。そこでは、サブドキュメント上のポートが、シートシンボル上でそのドキュメントを表わす同じ名前のシートエントリに接続しています。回路図やVHDLのサブドキュメントについては、図 21 をご覧ください。

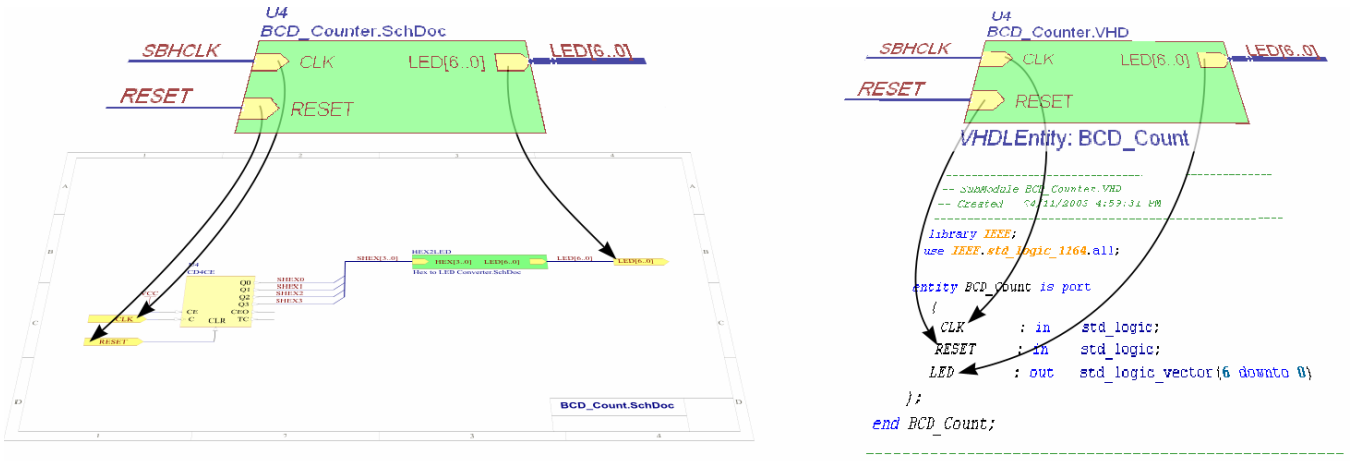


図21 階層のあるネットの接続性。下記ドキュメント上のシートエントリから相当するポートまで。

このチュートリアルでベースとなるシンプルなカウンタのデザイン例にとっては、デスクトップ NanoBoard から供給される同期クロック信号が早すぎます。 カウンタをスローダウンさせるため、分周器用の回路を追加します。 Simple_Counter の回路図にこの回路を追加するのではなく、サブファイルとして（最初は回路図に、その後に VHDL ファイルに）キャプチャします。それによって、FPGA をプログラムする際、階層デザインがどのように、使用されるか、デモしてみましょう

回路図サブシートを使用する分周器

さらに進んで、分周器の回路を回路図サブシートにキャプチャしてみましょう。

1. 回路図ドキュメント Simple_Counter.SchDoc を開きます。
2. シートシンボルを配置します（**Place » Sheet Symbol**）。最初は、シート上のフリースペースに配置してください。 このシートシンボルはサブシートを表しています。サブシートの上に分周器の回路を定義します。
3. シートシンボルをダブルクリックし、表示される *Sheet Symbol* ダイアログで以下のように設定します。
   - **Designator（部品番号）**: U_Clock_Divider
   - **ファイル名**: Clock_Divider.SchDoc.
4. シングルシートエントリをシートシンボルの左右に追加します（**Place » Add Sheet Entry**）。 以下のプロパティを設定します。
   - 左側のシートエントリ： **Name**: CLK_REF, **I/O Type**: Input
   - 右側のシートエントリ： **Name**: CLK_OUT, **I/O Type**: Output
5. シートシンボルをメインの回路内、CLK_REF ポートコンポーネントの隣に接続します（図 23）。

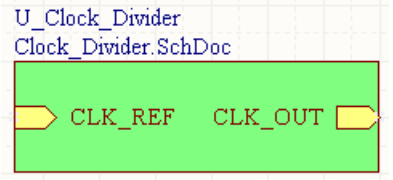


図22 元の回路図サブシートを参照しているシートシンボル

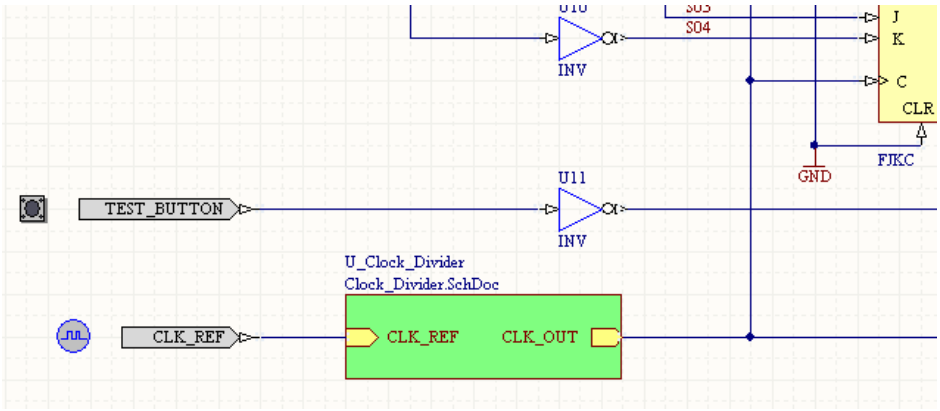


図23 シートシンボルが付された Simple_Counter の回路図。Clock_Divider.SchDoc サブシートが配置され、配線されています。

これで、親になるシートシンボルができました。参照先となり、目的の回路が載ったサブシートを実際に作成しなくてはなりません。

注記：以下のステップ 6-9 は、回路図サブシートとその内容を線画で定義するプロセスです。 ステップのこの部分を省略するには、Project パネル内の `Simple_Counter.PrjFpg` のエントリを右クリックして、Add Existing to Project を選択してください。. *Choose Documents to Add to Project* ダイアログが表示され、`Clock_Divider.SchDoc` ファイルを開くようにナビゲートされます。このファイルは、Altium Designer をインストールしたディレクトリの `\Examples\Tutorials\Getting Started with FPGA Design` フォルダにあります。
その後、プロジェクトと回路図のトップシートを保存して、ステップ 10 に進んでください。

6. シートシンボルを右クリックして、**Sheet Symbol Actions » Create Sheet From Symbol** を選択します。 新しい回路図ドキュメント `Clock_Divider.SchDoc` が作成され、アクティブドキュメントとしてメインデザインウィンドウ内に開かれます。 最初、このドキュメントには 2 つのポート、CLK_REF と CLK_OUT があります。これらは親のシートシンボルにおけるそれぞれのシートエントリに関連（接続）するものです。
7. **Libraries** パネルにアクセスして、6 つの分周器コンポーネント（CDIV10DC50 - 50%の負荷サイクル出力で 10 分の 1 分周）を FPGA の一般ライブラリ（`FPGA Generic.IntLib`）からこの新しいシートの上に配置します。 コンポーネントを順に接続し、2 つのポートの間をつなぎます（図 24）。

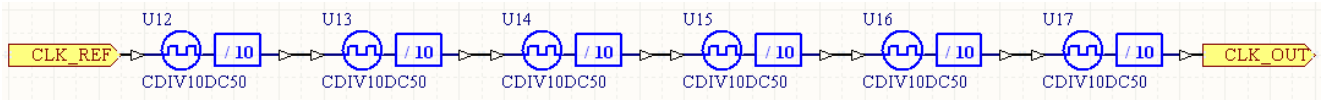


図24 回路図サブシート Clock_Divider.SchDoc 上のクロック分周回路

8. **Tools » Annotate Schematics Quietly** を使用してコンポーネントにアノテートを施します。
9. **File » Save All** を使用して回路図と親プロジェクトの両方を保存します。 `Clock_Divider.SchDoc` の保存には、デフォルトの場所（トップレベル回路図と同名のフォルダなど）を用意してください。
10. プロジェクトをコンパイルして、エラーをチェックします。 エラーがあれば、修正、保存し、もう一度コンパイルしてください。
11. コンパイル後、プロジェクトのシート階層を **Projects** パネルでチェックできます。 これでプロジェクトがサブシート（`Clock_Divider.SchDoc`）を Simple_Counter の回路図の子として認識するようになります。

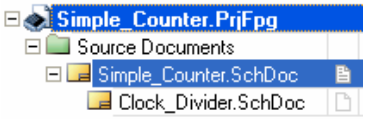


図25 回路図サブシートの階層

クロック分周器の回路の追加はこれで終了です。テストを行ってみましょう。

12. **Devices** ビューを開き、**Live** オプションが有効で、**Connection** の表示が緑であることを確認します。
13. 物理的 FPGA デバイスがプログラムされ、Process Flow は全ステージがちゃんと完了したことを表示（緑）しています。デザインのソースドキュメントを変更した場合、どうしてこのような結果になるのでしょうか。 その答えは、**Ignore FPGA source** オプション（デフォルトで有効）にあります。 このオプションを無効にしなくてはならない場合、Process Flow のステータス評価の際に、変更されたソースドキュメントが考慮されるようになります。 試しに、**Ignore FPGA source** オプションを無効化してみてください。Process Flow の各ステージが黄色に変わり、失効、再度実行が必要、という状態になります。

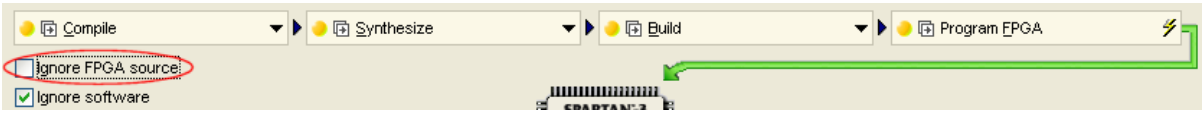


図26 Ignore FPGA source オプションを無効にした場合、Process Flow の評価時に、変更されたソースが検討される

14. ドータボードの FPGA デバイスを再度プログラムします。 最も簡単な方法は、Process Flow の **Program FPGA** ステージを直接クリックすることです。 それに先立つすべてのステージが自動的に、順次実行されます。なぜなら、それらは、いまや失効とされているからです。 論理合成ステージの後で、この新しい回路図サブシートに関連する中間 VHDL ファイルが、プロジェクト用に生成された VHDL ファイル内に現れます。

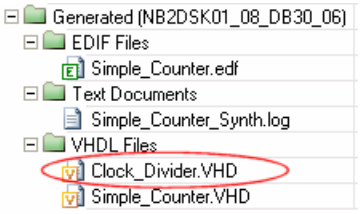


図27 論理合成で生成されたサブシート用の追加VHDL ファイル

一度プログラムされ、カウンタ（DIP スイッチのスイッチ 7、スイッチ 8 がオン）を開始、NanoBoard のユーザ LED の出力がスローダウンしているのを確認します。以上でシーケンスはずっとクリアになりました。 DIP スイッチを使用して、方向を切り替えてみてください。今度はうまくいくはずです。

HDLサブファイルを使用する分周器

回路図シートと HDL コードの組合せでデザインをキャプチャする場合、デザイン階層のコンセプトは、容易に拡張できます。 VHDL や Verilog のサブドキュメントは、回路図サブシートと同じ方法で参照されます。つまり、シートシンボル中で、それが表しているサブドキュメントのファイル名を特定することで、参照されます。

VHDL サブドキュメントを参照する際、考慮される接続性は、シートシンボルから VHDL ファイルにおけるエンティティの宣言までになります。 エンティティを VHDL のファイル名とは異なる名前で参照するには、`VHDLEntity` パラメータをシートシンボルに含めます。この値は、VHDL ファイルにおいて宣言された Entity の名前です。

Verilog サブドキュメントを参照する際も、プロセスは同様です。考慮される接続性は、シートシンボルから Verilog ファイルにおけるモジュールの宣言までになります。 モジュールを Verilog のファイル名とは異なる名前で参照するには、`VerilogModule` パラメータをシートシンボルに含めます。この値は、Verilog ファイルにおいて宣言された Module の名前です。

Mhz の単純な遅延によるクロック分周を装備した VHDL ファイルとして、追加したばかりの回路図サブシートの代用を考えてみましょう。

最初に VHDL ソースファイルを生成します。

注記：以下のステップ 1-3 は、VHDL サブファイルとその内容を線画で定義するプロセスです。 ステップのこの部分を省略するには、Project パネル内の `Simple_Counter.PrjFpg` のエントリを右クリックして、Add Existing to Project を選択してください。. *Choose Documents to Add to Project* ダイアログが表示され、`Clock_Divider.SchDoc` ファイルを開くようにナビゲートされます。このファイルは、Altium Designer をインストールしたディレクトリの `\Examples\Tutorials\Getting Started with FPGA Design` フォルダにあります。
その後、プロジェクトを保存して、ステップ 4 へ進みます。

1. **Projects** パネルの `Simple_Counter.PrjFpg` エントリを右クリックして、**Add New to Project » VHDL Document** を選択します。 新しい VHDL ドキュメント `VHDL1.vhd` が作成され、アクティブドキュメントとしてメインデザインウィンドウ内に開かれます。 ドキュメントを `Clock_Divider.vhd` の名前で、プロジェクトファイルと同じ場所（フォルダ）に保存します。
2. 以下の VHDL コードをドキュメントに入力してください。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity Clock_Divider is
     port (
          CLK_REF : in std_logic;
          CLK_OUT : out std_logic
     );
```

```
end entity;

architecture RTL of Clock_Divider is
begin
    process(CLK_REF)
        variable i : integer range 0 to 999999;
    begin
        if rising_edge(CLK_REF) then
            if i = 0 then
                CLK_OUT <= '1';
                i := 999999;
            else
                CLK_OUT <= '0';
                i := i - 1;
            end if;
        end if;
    end process;
end architecture;
```

3. ドキュメントを保存します。

VHDL ソースファイルができました。これで、シートシンボルを直にそこから作成することができます。 でもその前に、既存のシートシンボルを、トップレベルの回路図と参照先の回路図サブシートから削除しましょう。

4. 回路図サブシート Clock_Divider.SchDoc を FPGA プロジェクトから削除します。**Projects** パネル内のサブシート名を右クリックして、**Remove from Project** をメニューから選択します。
5. Simple_Counter の回路図ドキュメント Simple_Counter.SchDoc を開きます。 既存のシートシンボルをクリックして、**Delete** キーを押します。
6. メインメニューから、**Design » Create Sheet Symbol From Sheet Or HDL** を選択します。 *Choose Document to Place* ダイアログが表示されますので、Clock_Divider.vhd エントリを選び、**OK** をクリックします。
7. 新しいシートシンボルを回路に配置、配線します（図 28）。 注意していただきたいのは、**Designator** と **Filename** が自動的に U_clock_divider と Clock_Divider.vhd に、それぞれ設定されることです。 VHDLENTITY パラメータもまた、value = clock_divider（VHDL サブファイルにおけるエンティティの名前）として追加されます。

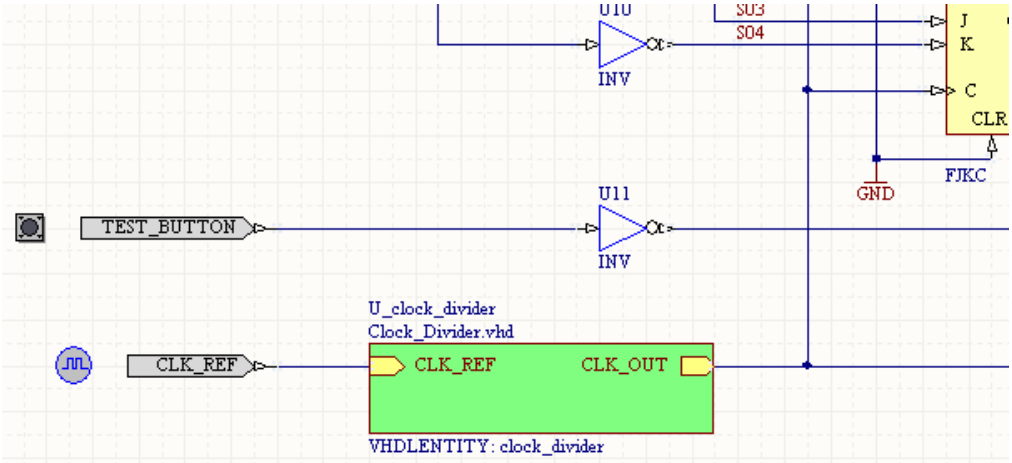


図28 シートシンボルが付された Simple_Counter の回路図。Clock_Divider.vhd サブファイルが配置され、配線されています。

8. 回路図とプロジェクトを保存します。
9. プロジェクトをコンパイルして、エラーをチェックします。 エラーがあれば、修正、保存し、もう一度コンパイルしてください。
10. コンパイル後、プロジェクトの階層を **Projects** パネルでチェックできます。 これでプロジェクトが VHDL サブファイル（`Clock_Divider.vhd`）を Simple_Counter の回路図の子として認識するようになります。

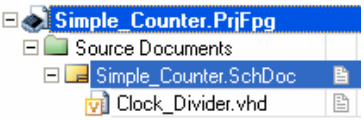


図 29 VHDL サブファイルの階層

クロック分周器の VHDL バージョンの追加はこれで終了です。テストを行ってみましょう。

11. **Devices** ビューを開き、ドータボードの FPGA デバイスを再プログラムします。Process Flow の **Program FPGA** ステージをクリックしてください。
12. プログラムを行い、カウンタ（DIP スイッチのスイッチ 7、または、スイッチ 8 のどちらかをオンに設定）をスタートさせます。 NanoBoard のユーザ LED で、スローダウンしたカウンタ出力と、DIP スイッチによる出力のコントロールとをはっきりと確認できます。

デバイスピンの状態をモニタします - ライブモニタリング！

デザインが、いったん FPGA にダウンロードされると、Hard Devices のチェーンが FPGA ピンの状態のモニタに使えるようになります。 実際に実行するには、デバイスに関連する **JTAG Viewer** パネル（計器パネルからアクセスできます） を **Live Update** モードで実行するように設定します。

高密度なコンポーネントパッケージが、デバイスピンの物理的なプロービングを不可能にしているところでは、**JTAG Viewer** パネルが物理的デザインのデバッグを「仮想的なスタイル」で促進します。 そこでは、JTAG 通信を標準で使用して、FPGA だけでなく、デザイン内の任意の JTAG 準拠デバイスのピン状態を問い合わせています。 各ピンの状態を表し、回路図シンボルとフットプリントの両方のイメージを含めることで、デザインの分析とデバッグを補助します。

すでにクロック分周器を導入して、カウンタをスローダウンさせたので、デバイスピンの動作をよりクリアに観察することができます。さらに先に進んで、中をのぞいてみましょう。

1. **Devices** ビューから、FPGA 上でプログラムを実行しながら、Hard Devices チェーン内の物理 FPGA デバイス用アイコンをダブルクリックします。 **Instrument Rack – Hard Devices** パネルが表示されます（図 30）。

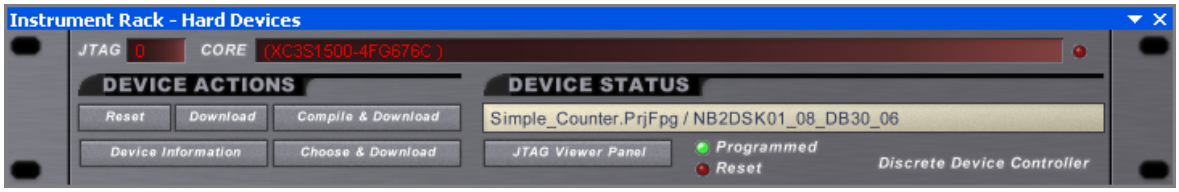


図 30 物理的 FPGA デバイス用の Instrument パネル

2. **JTAG Viewer Panel** ボタンをクリックして、**JTAG Device Viewer** パネルにアクセスします。 **Live Update** と **Hide Unassigned I/O Pin** の両オプションが有効になっていることを確認してください（図 31）。

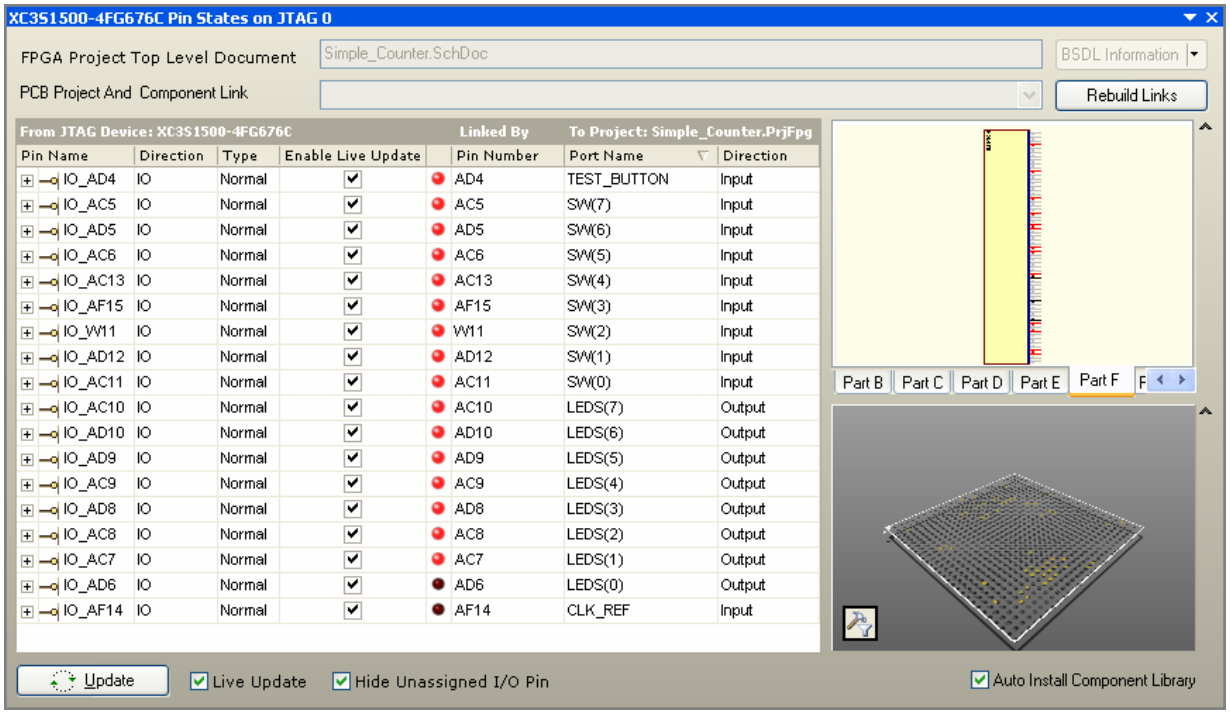


図 31 デバイスピンのライブモニタリング

3. これで、回路の動作中でも、物理 FPGA デバイスのピンを観察できるようになりました。 注意していただきたいのは、回路の実行に伴って、LED アイコンが次の LED ポートへ、どのように点灯していくか、また、ツイストリングカウンタの典型的な出力を示しているかかどうか、です。 コンポーネントシンボル上の関連ピンや点灯するフットプリントについて、各ピンがアクティブになる様子を確認することもできます。
4. カウントシーケンスの向きを、NanoBoard の DIP スイッチを使用して変更し、パネルの変更が反映されていることを確認します。

仮想計器の追加

デザイン内部のノードの状態をテストするため、仮想計測器を ▯ 接続する'ことができます。 計測器の▯ ハードウェア'の部分は、他のコンポーネントと同じように回路図上に配置、配線され、FPGA に論理合成されます。 各計測器へのインターフェースには、**Devices** ビューからアクセスします。

仮想計測器は、`\Library\Fpga\FPGA Instruments.IntLib` ライブラリから配置します。

一組の計測器をチュートリアル用のデザインに追加してみましょう。

- 周波数カウンタ（FRQCNT2） - クロック分周器回路からの周波数出力の表示用
- デジタル I/O モジュール（DIGITAL_IO） - カウンタ出力と NanoBoard の DIP スイッチに関連する 3 つのスイッチについての現在状態の表示用

これらの計測器についての、詳細は、*CR0101 FRQCNT2 Frequency Counter* または *CR0179 DIGITAL_IO Configurable Digital IO Module* のドキュメントをそれぞれ参照してください。

この時点で、論理合成後のプロジェクトには、VHDL サブファイルで供給されたクロック分周器があります。 チュートリアルのこのパートでは、それをそのままにしておくことにします。回路図での分周器の編集作業に戻る必要は特にないからです。もちろん、試してみることはいつでも可能です。

周波数カウンタの追加

では、はじめましょう。周波数カウンタをチュートリアル用のシンプルなカウンタデザインに配置し、接続します。

1. Simple_Counter の回路図ドキュメント `Simple_Counter.SchDoc` を開きます。
2. **Libraries** パネルにアクセスして、FRQCNT2 コンポーネントを FPGA Instruments という統合ライブラリ（`FPGA Instruments.IntLib`）から配置します。ＶＨＤＬサブファイルの参照に使ったシートシンボルの右下に配置します。
3. **Tools » Annotate Schematics Quietly** コマンドでアノテートを施します。

モニタしたい信号は、クロック分周器（CLK_OUT）の出力です。 この信号をＦＲＥＱＡ入力に接続します。 TIMEBASE 信号は、NanoBoard（CLK_REF）からのオリジナルの周波数である必要があります。

4. 計器を接続します（図 32 参照）。 ＧＮＤポートを配置します。計器のＦＲＥＱＡピンに直に接続されていますが、このチャネルは使用していません。

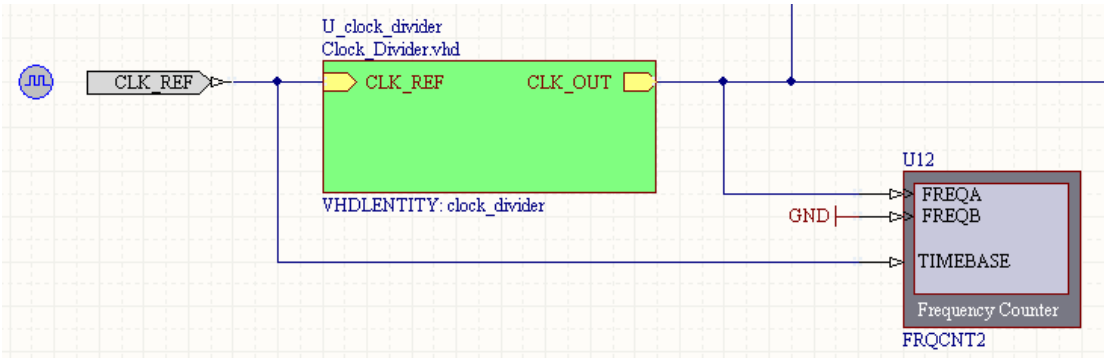


図32 シンプルカウンタのデザインに配置され、接続された周波数カウンタ

デジタルＩ/Ｏモジュールの追加

デジタルＩ/Ｏの計器を追加しましょう。

1. **Libraries** パネルにアクセスして、デザイン中のメイン回路の中央上方に、DIGITAL_IO コンポーネントを配置します。
2. **Tools » Annotate Schematics Quietly** コマンドでアノテートを施します。

接続を検討する前に、モニタしたい信号について、計器を構成しましょう

3. 計器を右クリックして、**Configure U13 (DIGITAL_IO)**をメニューから選択。*Digital I/O Configuration* ダイアログにアクセスします（図 33）。 注意していただきたいのは、デフォルトで、計器の構成がシングル 8-bit 入力バス（`AIN[7..0]`）、シングル 8-bit 出力バス（`AOUT[7..0]`）になっていることです。

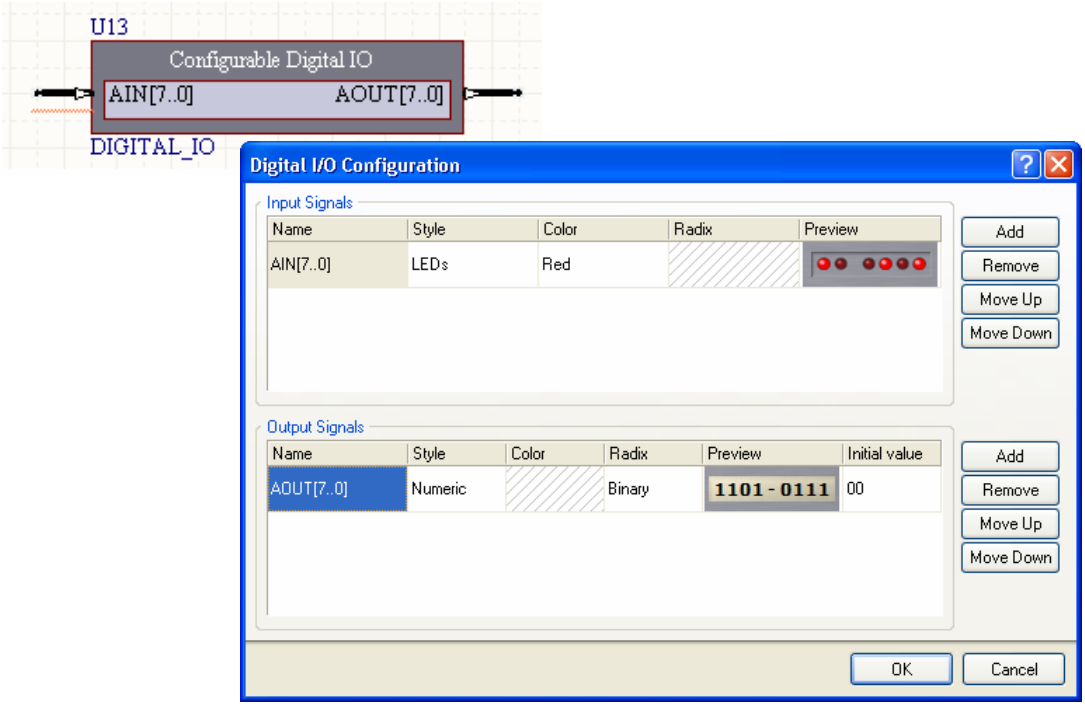


図33 デジタル I/O計器構成のコントロールにアクセス

4. 任意の出力を生成することで、デフォルトのエントリは簡単に削除することができます。 `AOUT[7..0]`信号をクリックして選び、**Remove** ボタンをクリックします。
5. デフォルトの入力信号は、チュートリアルの目的にかなう周波数幅なので、この信号は保持しましょう。ただし、もっと意味のある名前にします。 `Count_Output[7..0]`という名前にします。 **Style** 設定を `LEDs` として保持します。ただし、**Color** 設定は、`Green`（NanoBoard 上の LED の色を模倣）に変更します。
6. 関連する DIP スイッチ信号をモニタするため、さらに 3 つの入力信号を構成に追加しましょう。 3 つの信号は以下のように定義します。
   - Signal 1 – **Name**: `Shift_Left`, **Style**: `LEDs`, **Color**: `Green`.
   - Signal 2 – **Name**: `Shift_Right`, **Style**: `LEDs`, **Color**: `Green`.
   - Signal 3 – **Name**: `STOP`, **Style**: `LEDs`, **Color**: `Red`.

   ダイアログの Input Signal 領域は、図 34 のようになるはずです。

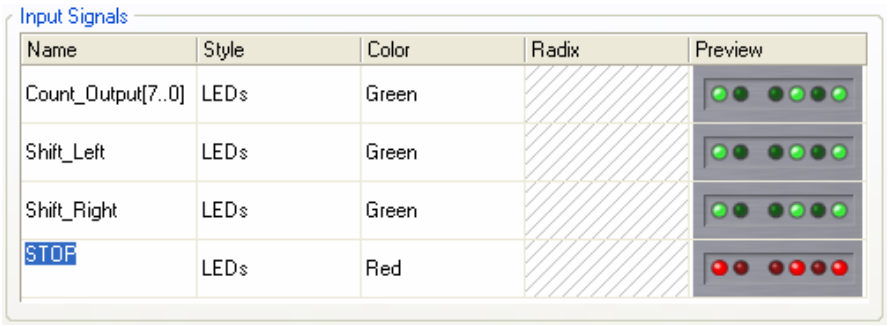


図34 構成後のデジタル I/O 計器

7. 図 35 のように、計器を接続します。 注意していただきたいのは、乱雑な接続を避けることです。そのため、ネットラベルを使用してモニタしたい信号（`SO4`、`SO3`、`SO2`）を取り出します。 このようなネットラベルは、データ取得ポイントのためにすでに作ってあるので、コピー&ペーストすれば事足ります。 注意していただきたいのは、モニタしている信号は、すでにそれぞれの変換器を通過した後のものだということです。 このことは、DIP スイッチに関連するスイッチがアクティブローであるために起こります。したがって、Off よりも、ON ポジションに設定されたときに、ライトの点灯を確認するほうがよいでしょう。

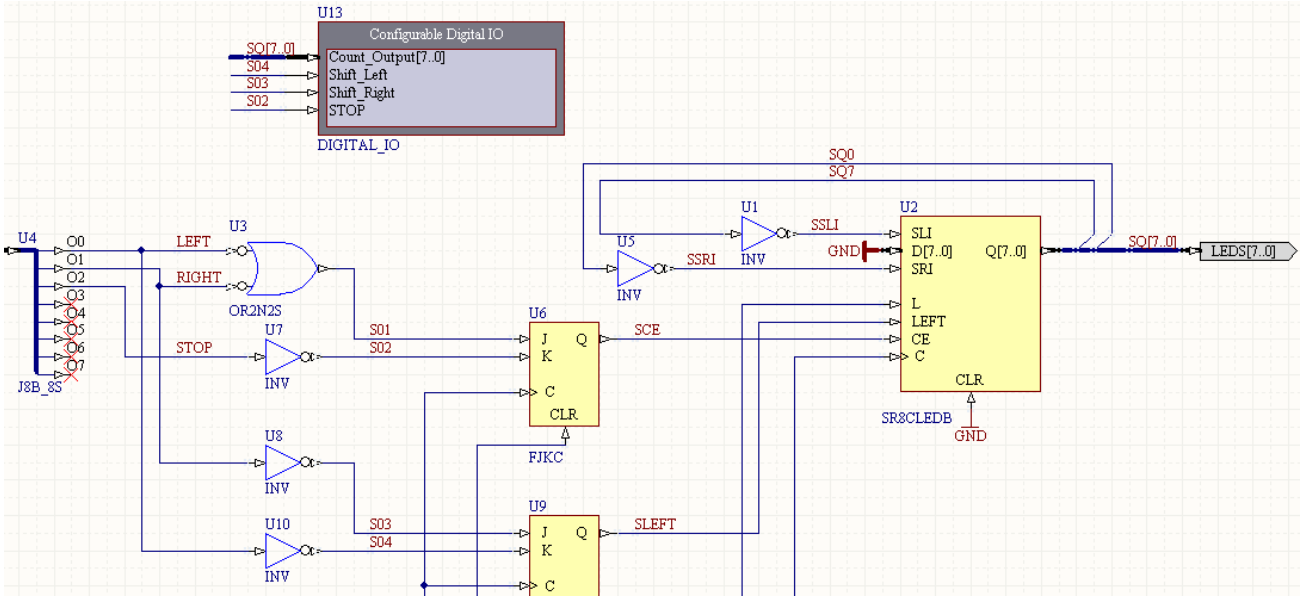


図35 シンプルカウンタのデザインに配置され、接続されたデジタルI/O

ソフトデバイスJTAGのチェーンを有効にする

Altium Designer 環境から、組込みプロセッサや FPGA デザイン内の仮想計測器への通信は、JTAG 通信リンクを通じて行われます。 この通信は、デスクトップ NanoBoard 上でソフト JTAG（または Nexus）チェーンとして参照されます。

ソフト JTAG チェーンの信号（NEXUS_TMS、NEXUS_TCK、NEXUS_TDI、NEXUS_TDO）は、デスクトップ NanoBoard の NanoTalk コントローラ（ザイリンクス Spartan-3）に由来するものです。 通信チェーンの一環として、これらの信号は FPGA ドータボードの 4 つのピンに接続されています。 これらのピンとの通信には、NEXUS_JTAG_CONNECTOR というデザインインターフェースのコンポーネントが必要です（図 36）。 このコンポーネントは、FPGA NB2DSK01 Port-Plugin という統合ライブラリ（\Library\Fpga\FPGA NB2DSK01 Port-Plugin.IntLib）にあります。

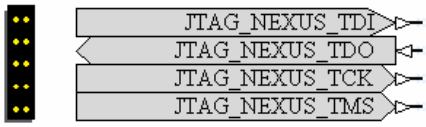


図36 Nexus JTAG コネクタ

このコンポーネントで、ソフト JTAG が使用できるようになります。 関係するすべての Nexus 対応可能デバイス（このチュートリアルでは、2 つの仮想計測器）をこのチェーンに接続するには、NEXUS_JTAG_PORT コンポーネント（図 37）の配置し、これを直接 NEXUS_JTAG_CONNECTOR に接続することが必要です。 このコンポーネントは、FPGA Generic 統合ライブラリ（\Library\Fpga\FPGA Generic.IntLib）にあります。

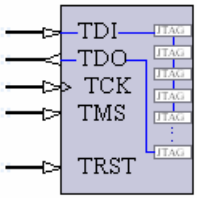


図37 Nexus JTAG ポート

この NEXUS_JTAG_PORT コンポーネントが存在すると、NEXUS_JTAG_DEVICE=True というパラメータを持つすべてのコンポーネントがソフト JTAG チェーンに接続されます。

JTAG通信についての詳細は、*AR0130 PC to NanoBoard Communications*のドキュメントを参照してください。

1. NEXUS_JATAG_CONNECTOR と NEXUS_JTAG_PORT コンポーネントを Simple_Counter の回路図上に配置し、お互いを接続してください。
2. VCC 電源ポートを NEXUS_JTAG_PORT コンポーネントの TRST 入力に配置、配線します。

これで接続は完了です。2 つの仮想計測器が設定され、図 39 のようになるはずです。

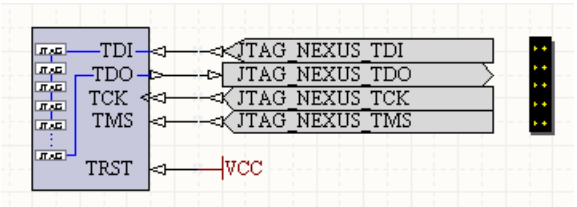


図38 JTAG デバイスをソフト JTAG チェーンに接続

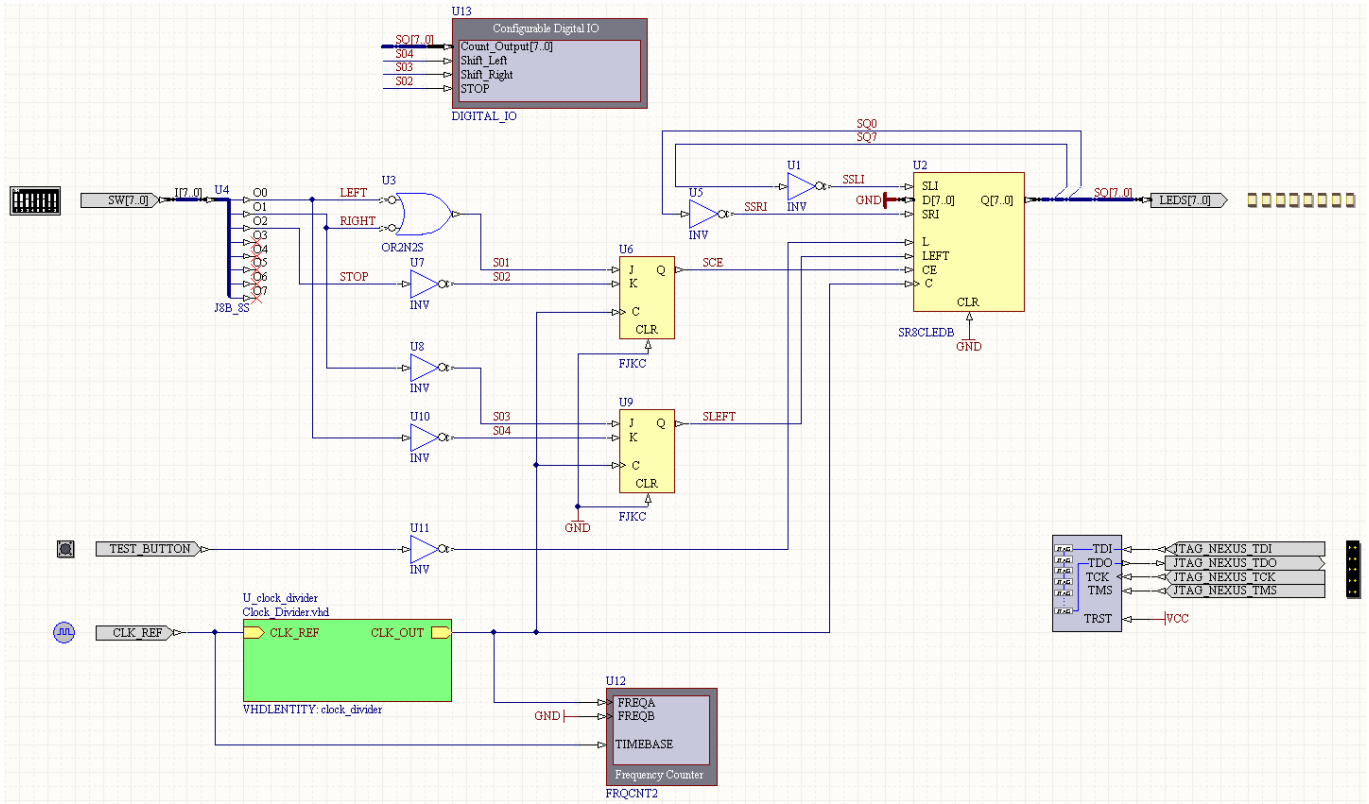


図 39 デザインの最終段階、2 つの仮想計測器の完成

3. 回路図とその親プロジェクトを保存します。
4. プロジェクトをもう一度コンパイルします。 今回は、**Messages** パネルには、何も表示されません。 これは、すべての SQ 出力が Digital I/O に使われているためです。これで、デザインのロードに進むことができます。

計測器コントロールへのアクセス

前のセクションで、ソフト JTAG チェーンに追加された仮想計測器の構造を確認しました。 これから FPGA を再プログラムし、実際に計測器にアクセスしてみるその前に、Altium Designer の環境でこれらの計測器にアクセス可能かどうか、検討してみることは有意義なことです。

ホストコンピュータは、ターゲットの計測器に IEEE 1149.1（JTAG）標準インターフェースで接続されています。 これは、物理的なインターフェースです。FPGA デバイスの物理的なピンへの接続を提供することで、計測器の組込みを可能にします。

Nexus 5001 標準は、プロトコルとしてホスト～全デバイス間の通信に使用され、これを通じた各デバイスのデバッグが可能です。 デバッグ可能なデバイスには、デジタル I/O、周波数カウンタが含まれます。その他の Nexus 準拠のデバイスであるデバッグ可能なプロセッサや周波数発生器、ロジックアナライザ、クロスポイントスイッチなども同様です。

これらすべてのデバイスがチェーン - Soft Device チェーン - の内部に接続されます。チェーンが定義されるのは、デザインがターゲットの FPGA デバイス内に実装された時です。チェーンは **Devices** ビュー内に表示されます。 これは、物理的なチェーンではないので、外部の配線を見たりすることはできません。Nexus 対応が可能なデバイス間で必要な接続は、FPGA 自身の内部で行われます。

1. **Devices** ビューを開いて、ドータボードの FPGA デバイスを再プログラムします。
2. 一度プログラムしておけば、Soft Devices チェーンは最新のチェーンとしてビュー内に表示されます。チュートリアル用の 2 つの仮想計測器（図 40）のアイコンもそこに含まれています

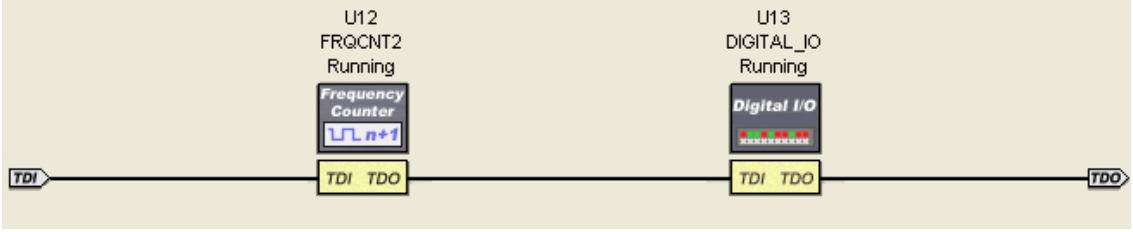


図 40 プログラム後の Soft Devices チェーン内にある仮想計測器

3. カウンタをスタートさせてください。NanoBoard の DIP スイッチ 7 または 8 を使います。
4. 計測用のアイコンをダブルクリックして、関連する 2 つのパネルを **Instrument Rack – Soft Devices panel** に表示させます（図 41）。 これらのパネルは必要なコントロールを提供し、計測器との対話をリアルタイムで表示します。

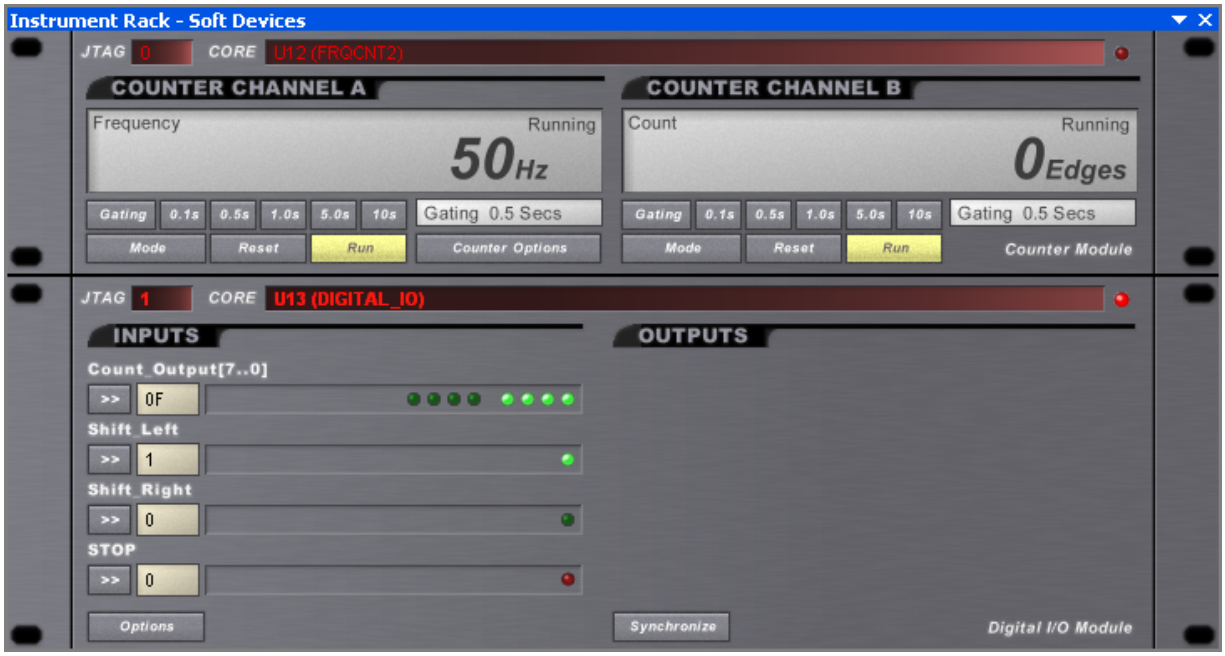


図 41 周波数カウンタとデジタル I/O モジュールの各計測器用コントロールパネル

パネルを見れば、異常時の確認ができます。 まず、周波数カウンタが周波数 50Hz を表示します。 覚えておいででしょうか。NanoBoard からのリファレンス周波数は 20MHz でした。作成したクロック分周器が MHz を分周しています。 期待値の 20Hz になっていません。 次は、カウンタ出力用の LED 表示です。デジタル I/O パネルは、ややチラチラしすぎているようです。LED は確かに、NanoBoard 上で見られるようなスムーズなシフトを示していません。

この 2 つのシチュエーションを打開するには、それぞれの計測器に関するオプションをさらに構成していく必要があります。作業はそれぞれの計測器パネルから行います。

5. 周波数カウンタ用の計測器パネルで **Counter Options** ボタンをクリックします。 *Counter Module – Options* ダイアログが表示されます。**Counter Time Base** エントリをデフォルトから変更。50.000MHz を 20.000MHz にします（これで、計測器の TIMEBASE 入力に接続した信号と同じ周波数になるはずです）。 ダイアログを閉じます。ご覧ください。表示される周波数は、いまや 20Hz になっています。
6. デジタル I/O モジュール用の計測器パネルで **Options** ボタンをクリックします。 *Digital I/O Module – Options* ダイアログが表示されます。**Update Display From Core Every** エントリをデフォルトから変更。250ms を最小値 100ms にします。 ダイアログを閉じます。 より早く更新させることで、LED はよりスムーズに表示されるようになります。

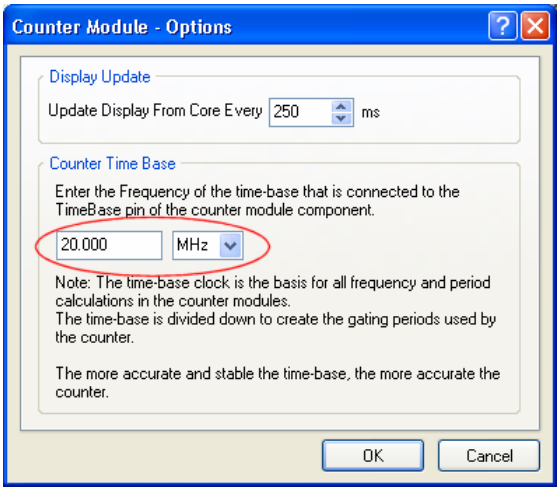


図 42 カウンタのタイムベースを計測器用に変更して、計測器の TIMEBASE 入力に物理的に接続されている周波数と同じになるようにする

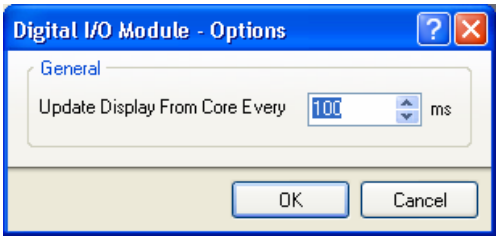


図 43 表示の更新速度をアップ

7. 作成したデザインを走らせてみましょう。カウント方向を切り替えたり、停止させたりすることで、デジタル I/O モジュールはライブで、あるいは操作に伴って、アップデートされる様を観察します。

次のステップ

このチュートリアルでは、FPGA 設計の基礎を学んでいただくため、アルティウムの Innovation Station を使用することを前提に記述されています。 チュートリアル用のシンプルなカウンタの実装を通じてご紹介してきた基本的なコンセプト、そのしっかりとした基礎を身に着けたことで、皆さんはもう、ご自身の FPGA 設計に乗り出すことができます。

Altium Designer とデスクトップ NanoBoard が提供する開発環境は、このチュートリアルでご紹介した以上の、はるかに拡張的なツールと機能のセットをお届けすることができます。 32 ビット処理、ハードウェアアクセラレーション、カスタムロジックなどは、人をひきつけるためのキャッチフレーズでしかありません。しかし、フルカスタム可能な仮想計測器があれば、自分専用のパネルインタフェースをレイアウトすることさえできるのです。

FPGA設計とアルティウムのInnovation Stationについての高度なイントロダクションついては、*GU0123 An Introduction to Embedded Intelligence*のドキュメントを参照してください。

さらに、より進んだチュートリアル、より多くのツールや機能の提供については、以下のドキュメントを参照してください。

TU0122 組込みソフトウェア入門

TU0123 Creating a Core Component

TU0126 Checking Signal Integrity on an FPGA Design

TU0128 Implementing a 32-bit Processor-based Design in an FPGA

TU0129 Converting an Existing FPGA Design to the OpenBus System

TU0130 Getting Started with the C-to-Hardware Compiler

TU0131 Capturing Video the Easy Way

TU0133 Designing Custom FPGA Logic using C

TU0135 FPGA デザインへのカスタム計器の追加

更新履歴

日付	バージョン番号	変更内容
16-Jan-2004	1.0	初リリース
23-Sep-2004	1.1	Clock divider components updated
18-Jan-2005	1.2	Components in Johnson_Counter.SchDoc updated
13-Apr-2005	1.3	Updated for Altium Designer
22-Aug-2005	1.4	Verilog references included
28-Nov-2005	1.5	Updated for Altium Designer 6
12-Apr-2007	1.6	Updated for Altium Designer 6.7
17-May-2008	2.0	Updated for Altium Designer Summer 08. Project and schematic documents renamed to Simple_Counter.PrjFpg and Simple_Counter.SchDoc respectively. Tutorial content enhanced throughout. Section on virtual instrumentation added.