



14. Chip Editor によるデザイン の解析および設計変更管理

この資料は英語版を翻訳したもので、内容に相違が生じる場合には原文を優先します。こちらの日本語版は参考用としてご利用ください。設計の際には、最新の英語版で内容をご確認ください。

QII53010-6.0.0

はじめに

FPGA の設計者が直面する最も困難な課題の 1 つに、デザイン・サイクルの後半での ECO (Engineering Change Order) の実装が挙げられます。デザイン・サイクルの後半で機能を変更する場合は、短期間で変更要求仕様を満たしながら機能やタイミングを維持することなどが重要になります。

Quartus® II ソフトウェアの Chip Editor では、アルテラ・デバイスの内部構造の表示、内部タイミングの調査、デバイス内のリソースに対する機能とパラメータ設定の編集を行うことができます。また、Chip Editor は、デザインで行ったすべての変更を文書化し管理するのにも役立ちます。

Chip Editor は配置配線後のデザイン・データベースを直接操作するため、フル・コンパイルを実行することなく、デザインの変更を数分で実装することが可能です。変更は特定のデバイス・リソースに限定されるため、デザインの残りの部分のタイミング性能は影響を受けません。デザインの不正な変更を防止するために、すべての変更に対してデザイン・ルール・チェックが実行されます。

この章では、Chip Editor の使用方法と以下のトピックについて説明します。

- Chip Editor
- Chip Editor によるデザインの解析
- Resource Property Editor
- Change Manager
- 一般的な用途
- Chip Editor 後のコマンド

Chip Editor

Chip Editor では、デザインに関連する以下のアーキテクチャ固有の情報を表示することができます。

- デザインで使用するデバイス配線リソース：ブロックの接続方法、およびブロックを接続する信号配線を視覚的に調べます。
- LE コンフィギュレーション：デザイン内でのロジック・エレメント (LE) のコンフィギュレーションの状況を表示します。例えば、どの LE 入力を使用されているか、LE がレジスタまたはロック・アップ・テーブル (LUT)、あるいはその両方を使用しているかどうか、そして LE での信号フローを表示することができます。

- ALM コンフィギュレーション：デザイン内でのアダプティブ・ロジック・モジュール（ALM）のコンフィギュレーションの状況を表示します。例えば、どの ALM 入力を使用されているか、ALM がレジスタ、上位 LUT、下位 LUT、またはそれらをすべて使用しているかどうかを表示できます。また、ALM での信号フローを表示することもできます。
- I/O コンフィギュレーション：デバイスの I/O リソースの使用状況を表示します。例えば、使用されている I/O リソースのコンポーネント、遅延チェーン設定が有効かどうか、設定されている標準 I/O 規格、および I/O での信号フローを表示できます。
- PLL コンフィギュレーション：デザイン内の PLL (Phase-Locked Loop) のコンフィギュレーションの状況を表示します。例えば、PLL のコントロール信号および PLL の設定を表示できます。
- タイミング：FPGA エレメントの入力と出力間の遅延を表示します。例えば、DATAB 入力から COMBOUT 出力のタイミングを解析できます。

Chip Editor で、アルテラ・デバイスの以下のプロパティを変更することができます。

- LE と ALM
- I/O セル
- PLL
- エレメント間の接続
- エレメントの配置

Chip Editor は、以下のアルテラ・デバイス・ファミリをサポートしています。

- Stratix® II
- Stratix II GX
- Stratix
- Stratix GX
- Cyclone™ II
- Cyclone
- MAX® II

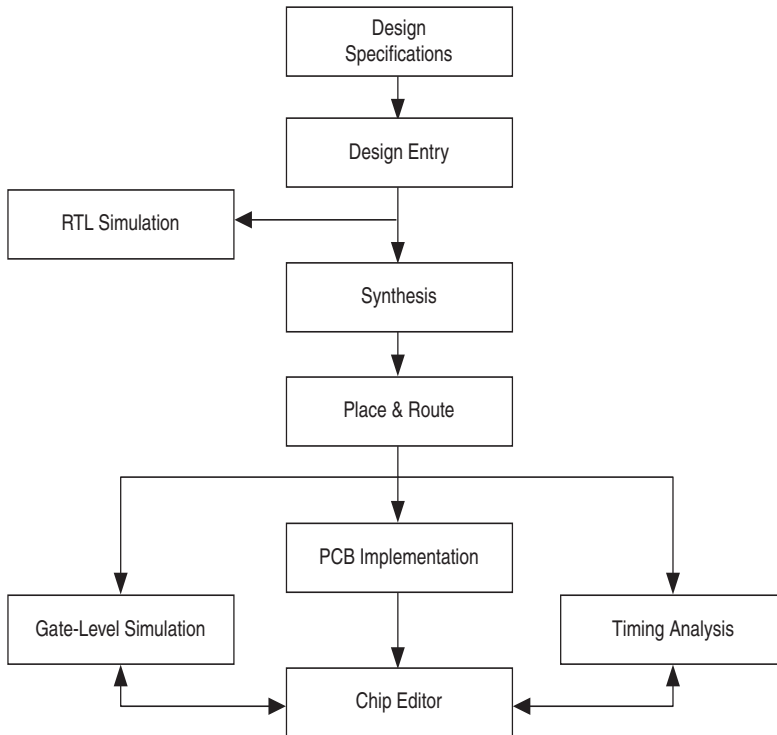
デザイン・フローにおける Chip Editor の使用

理想的なデザイン・フローは、デザイン仕様の開発に始まり、デザイン仕様を記述するレジスタ転送レベル（RTL）コードを作成し、記述した RTL コードが正しく実行されるかどうか検証し、さらにフィッティングしたデザインがデザインのタイミング制約に準拠しているかどうか検証します。ターゲットの FPGA のプログラミングに成功すれば、デザイン・フローは終了します。

残念ながら、多くの困難なプロセスと同様に、理想的なデザイン・フローが得られることはほとんどありません。RTL コードにバグが見つかったり、デザイン・サイクルの途中でデザインの仕様が変更されたりすることが頻繁に起こります。これらの種類のデザイン問題に、効率的に対処することが求められています。従来は、ソース・コードに戻って適切な変更を行い、合成、配置配線、検証など、デザイン・フロー全体をやり直していました。

アルテラの **Chip Editor** を使用すると、デザイン・サイクル時間を短縮することができます。デザインを変更したときに、**Quartus II** ソフトウェアでフル・コンパイルを実行する必要はありません。代わりに、配線配置後のネットリストに直接変更を加え、新しいプログラミング・ファイルを生成し、ソース・コードを修正することなくゲート・レベルのタイミング・シミュレーションおよびタイミング解析を実行することによって、改訂したデザインをテストします。問題が是正されるまで、**Chip Editor** を使用して繰り返し継続してデザインを変更することが可能です。図 14-1 に、デザイン・フローにおける **Chip Editor** の使用方法を示します。

図 14-1. Chip Editor デザイン・フロー



Chip Editor の機能

Chip Editor は迅速かつ効率的なデザインの変更を可能にする多くの最新機能を備えています。Chip Editor の統合ツール・セットが提供する機能は、以下のとおりです。

- **Chip Editor フロアプラン** : コンパイル後の配置、接続、および配線パスを表示し、新しいロジック・セルと I/O 素子を作成し、デザイン全体を表示した状態で既存のロジック・セルと I/O 素子を移動できます。
- **Resource Property Editor** : リソースのプロパティとパラメータを変更し、特定のタイプのリソース間の接続を変更できます。
- **Change Manager** : コンパイル後のデザインの変更を継続的に記録し、変更の衝突を回避します。また、行った変更の再現を可能にするツール・コマンド言語 (.tcl) ファイルを書き出すことも可能です。

Chip Editor フロアプランにより、迅速かつ簡単にコンパイル後の配置配線情報を表示することができます。Chip Editor フロアプランは、以下のいずれかの方法で起動します：

- Tools メニューの **Chip Editor** をクリックします。
- **Compilation Report** で右クリック・メニューを使用します。
- RTL ソース・コードで右クリック・メニューを使用します。
- タイミング・クロージャ・フロアプランで右クリック・メニューを使用します。
- **Node Finder** で右クリック・メニューを使用します。
- **Simulation Report** で右クリック・メニューを使用します。
- **RTL Viewer** で右クリック・メニューを使用します。

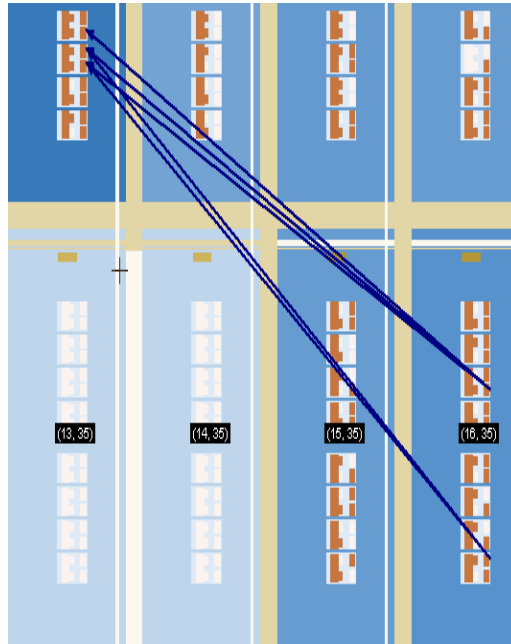
Chip Editor フロアプラン を使用した デザインの 解析

Chip Editor フロアプランは、Quartus II ソフトウェアでデザインのフル・コンパイルを実行した後、デザインを視覚的に解析するのに役立ちます。Chip Editor フロアプランを従来の Quartus II タイミング解析機能と併用すれば、デザイン解析を実行するための強力な手法が実現します。

クリティカル・パスの表示

クリティカル・パス表示機能は、[図 14-2](#) に示すように、Chip Editor の配線パスを表示します。パスのクリティカルリティはそのスラックによって決定され、タイミング解析レポートに示されます。View メニューの **Open Critical Path Settings** をクリックして、Chip Editor にクリティカル・パスを表示します。

図 14-2. クリティカル・パスがイネーブルされた Chip Editor

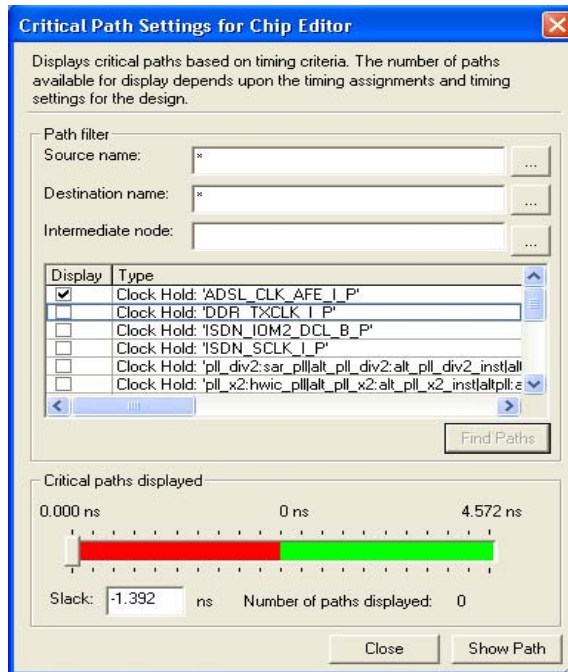


クリティカル・パスを表示する場合、表示するデザインのクロックを指定します。スラック・フィールドでスラック・スレッシュヨルドを指定して、表示するパスを決定します。例えば、スラック・フィールドで -1.392 ns のスラック・スレッシュヨルドを指定した場合、これ以上のスラックを持つ6つのパスが表示されます (図 14-3)。したがって、スラック・スレッシュヨルドを選択することにより、表示するクリティカル・パスを簡単に制御できます。



Chip Editor に表示されたクリティカル・パスのタイミング設定とタイミング解析を実行します。

図 14-3. Chip Editor のクリティカル・パスの設定



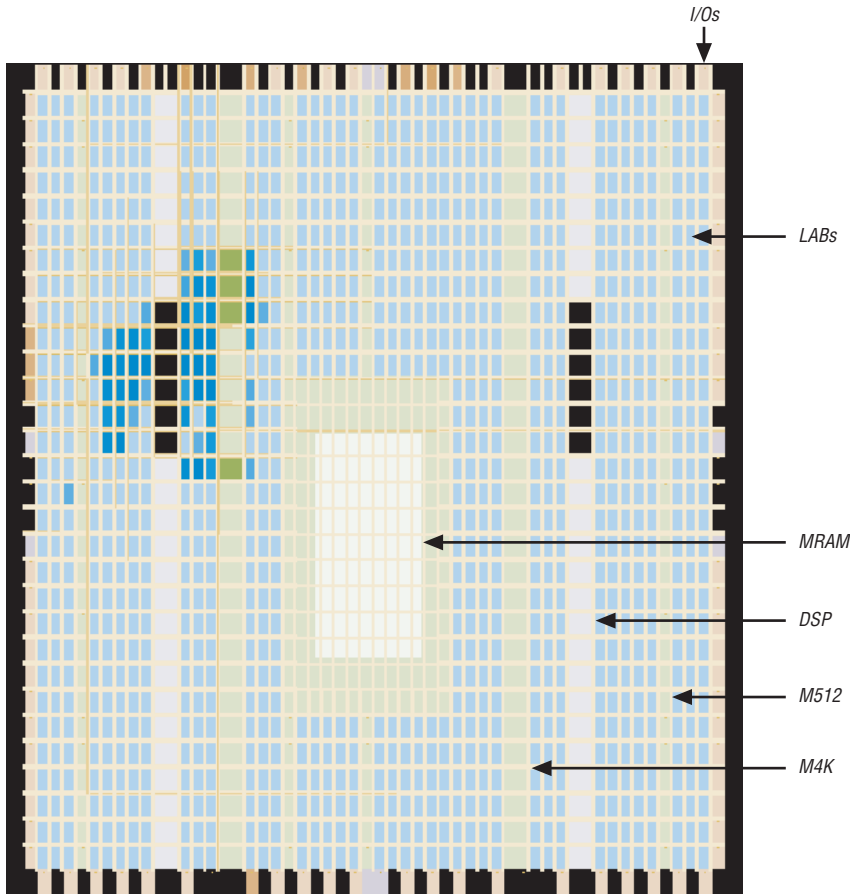
Chip Editor フロアプラン・ビュー

Chip Editor は、ターゲットのアルテラ・デバイスのさまざまな抽象化レベルを示す階層ズーム・ビューワを使用しています。ズーム・レベルを上げると抽象化レベルが下がり、デザインがより詳細に表示されます。Tools メニューの **Options** をクリックして、オート・ズーム機能の粒度を調整します。

第 1 (最高) レベル・ビュー

第 1 (最高) ズーム・レベルは、デバイス・フロアプラン全体のハイ・レベルな表示を提供します。Quartus II タイミング・クロージャ・フロアプランの Field View 並みの詳細な表示が可能です。デザインの任意のノードの配置を検索し、表示できます。図 14-4 に Chip Editor の第 1 レベル・ビューを示します。

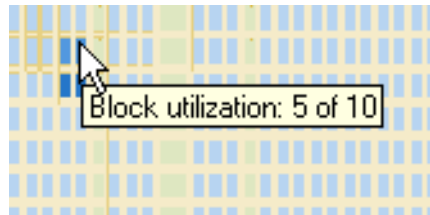
図 14-4. Chip Editor の第 1 (最高) レベル・ビュー



各リソースは別々の色で表示されるため、容易に区別することができます。Chip Editor フロアプランでは、階調カラー方式を採用しており、リソースの利用率が高くなるほど色が濃くなります。例えば、LAB で LE の使用が増えると、LAB の色が濃くなります。

マウス・ポインタをこのレベルでのリソースの上に置くと、高いレベルでのリソースの利用率を示すツールチップが表示されます (図 14-5)。

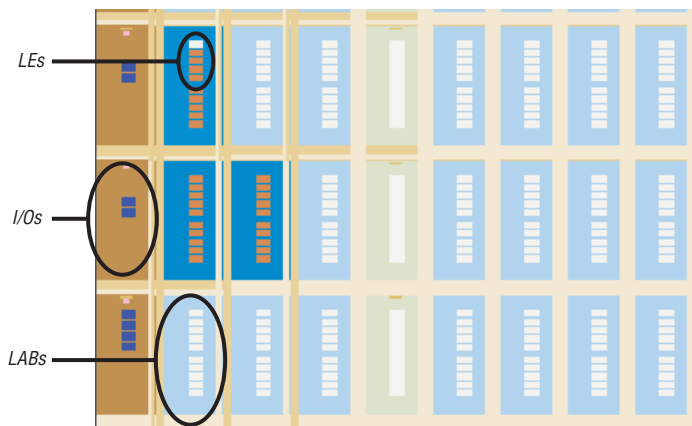
図 14-5. ツールチップ・メッセージ : 第 1 レベル・ビュー



第 2 レベル・ビュー

拡大表示すると、詳細レベルが増加します。図 14-6 に Chip Editor フロアプランの第 2 レベルのビューを示します。

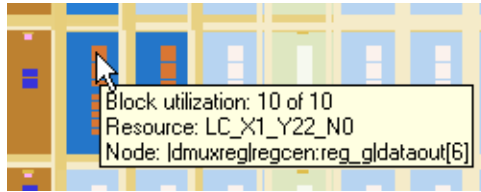
図 14-6. Chip Editor の第 2 レベル・ビュー



このレベルでは、LAB と I/O バンクの内容を表示できます。また、リソースの接続に使用される配線チャンネルも表示することができます。

このレベルでは、マウス・ポインタを LE の上に置くと、LE 名、LE の位置、およびその LAB で使用されるリソース数を示すツールチップが表示されます (図 14-7)。マウス・ポインタをインタコネクタ上に置くと、ツールチップはそのインタコネクタで使用される配線チャンネルを示します。

図 14-7. ツールチップ・メッセージ : 第 2 レベル・ビュー



第 3 レベル・ビュー

図 14-8 に最低レベルである第 3 レベルでの詳細レベルを示します。このレベルでは、FPGA の LAB で使用される各配線リソースを表示できます。

LE と I/O の物理的位置を移動することが可能です。リソースを移動するには、リソースを選択し、ドラッグして目的の位置でマウスを離します。このレベルでは、新しい LE と I/O を作成することもできます。リソースを作成するには、リソースを作成する位置で右クリックし、**Create Atom** をクリックします。リソースを削除するには、削除するリソース上で右クリックし、**Delete Atom** をクリックします。


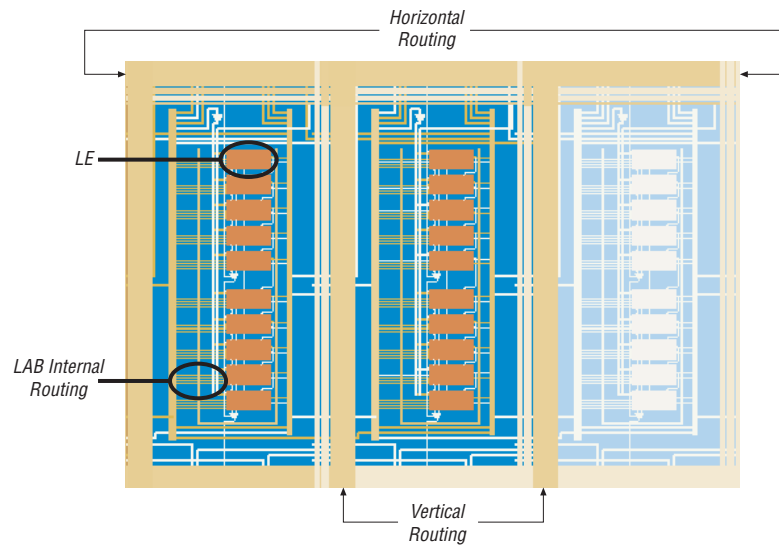
 リソースのすべてのファン・アウト接続を削除してからでないと、リソースを削除することはできません。

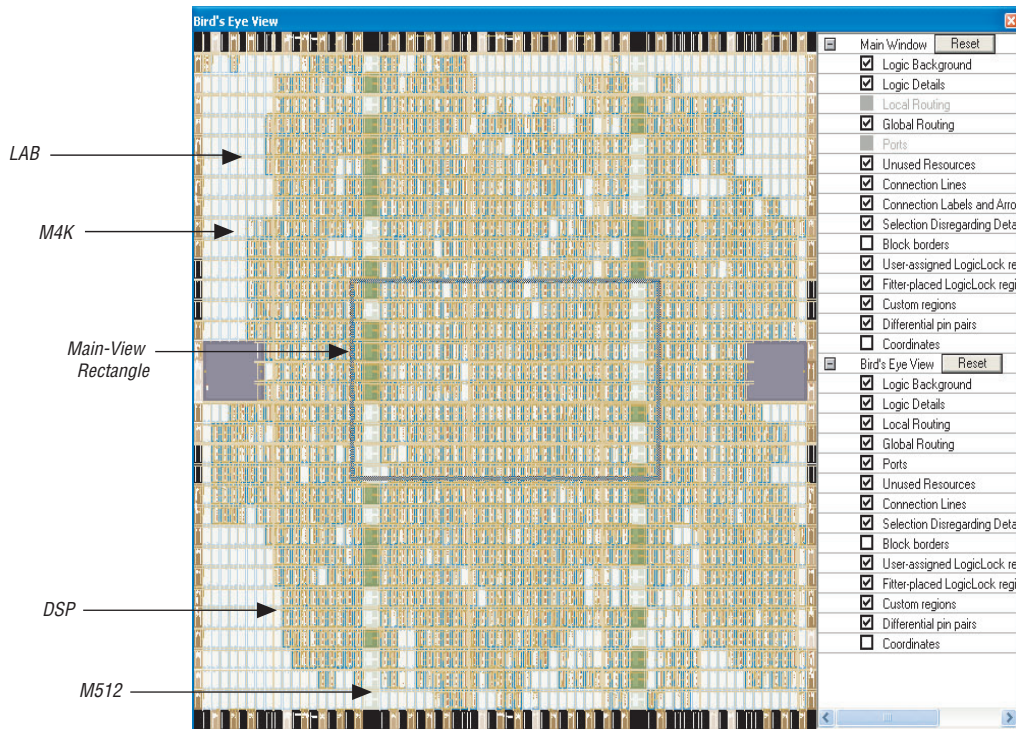
図 14-8. Chip Editor の第 3 レベル・ビュー



Bird's Eye View

Bird's Eye View (図 14-9) では、チップ全体のリソース使用を高レベル画像で表示し、高速かつ効率的に Chip Editor 内をナビゲートします。さらに、表示するグラフィック・エレメントを指定するよう制御できます。この指定は、Bird's Eye View または Chip Editor のメイン・ウィンドウのいずれかに適用できます。

図 14-9. Bird's Eye View



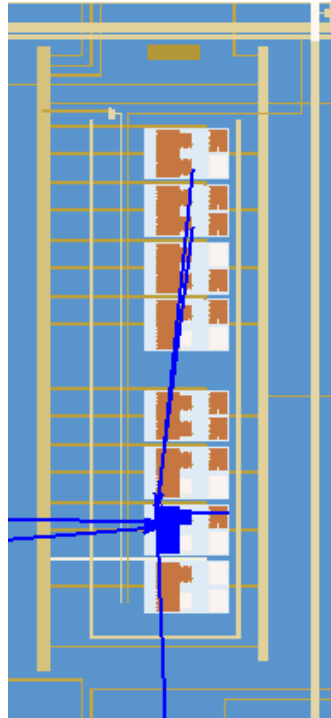
Bird's Eye View は Chip Editor フロアプランにリンクされている独立したウィンドウとして表示されます。Bird's Eye View 内のエリアを選択すると、Chip Editor フロアプランが自動的にリフレッシュされ、デバイスの領域を表示します。Bird's Eye View ウィンドウのメイン表示矩形のサイズを変更すると、Chip Editor Floorplan ウィンドウも拡大（または縮小）します。Bird's Eye View のメイン表示矩形を縮小すると、Chip Editor Floorplan ウィンドウでデザインがより詳細に表示されます。

Bird's Eye View は、表示したいデザインの部分がチップの反対側にあり、基準座標系を失わずにリソース・エレメント間を素早く移動したい場合に特に便利です。

ファン・イン・およびファン・アウト接続の生成

この機能は、選択した素子にファン・インまたは選択した素子からファン・アウトする素子を表示します。表示された接続を削除するには、**Chip Editor** ツールバーの **Clear Connections** アイコンを使用します。図 14-10 に選択したリソースのファン・イン接続を示します。

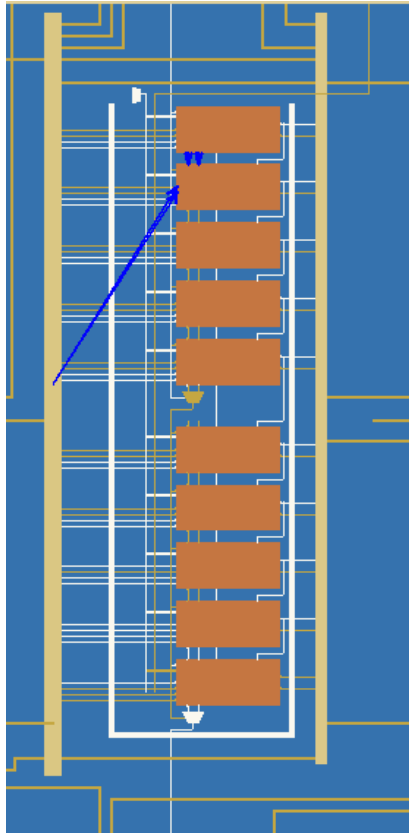
図 14-10.生成されたファン・イン



直接ファン・インおよびファン・アウト接続の生成

この機能により、選択した素子のファン・インおよびファン・アウト接続である直接リソースを表示できます。例えば、ロジック・リソースを選択して直接ファン・インを表示して、ロジック・リソースをドライブする配線リソースを確認することができます。すべてのロジック・リソースおよび配線リソースの直接ファン・インおよびファン・アウトを生成できます。表示された接続を削除するには、ツールバーの **Clear Connections** アイコンを使用します。図 14-11 に選択したリソースの直接ファン・アウト接続を示します。

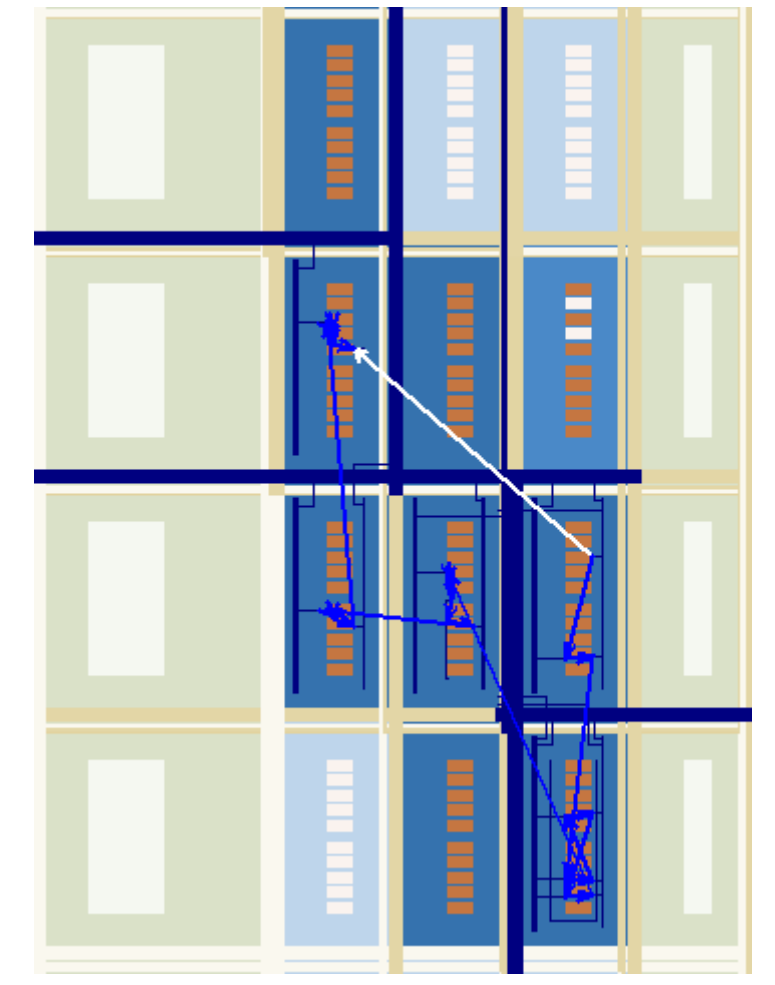
図 14-11.直接ファン・アウト接続



配線のハイライト

この機能により、選択したパスや接続に使用する配線リソースをハイライトすることができます。図 14-12 に 2 つのロジック・エレメント間に使用する配線リソースを示します。

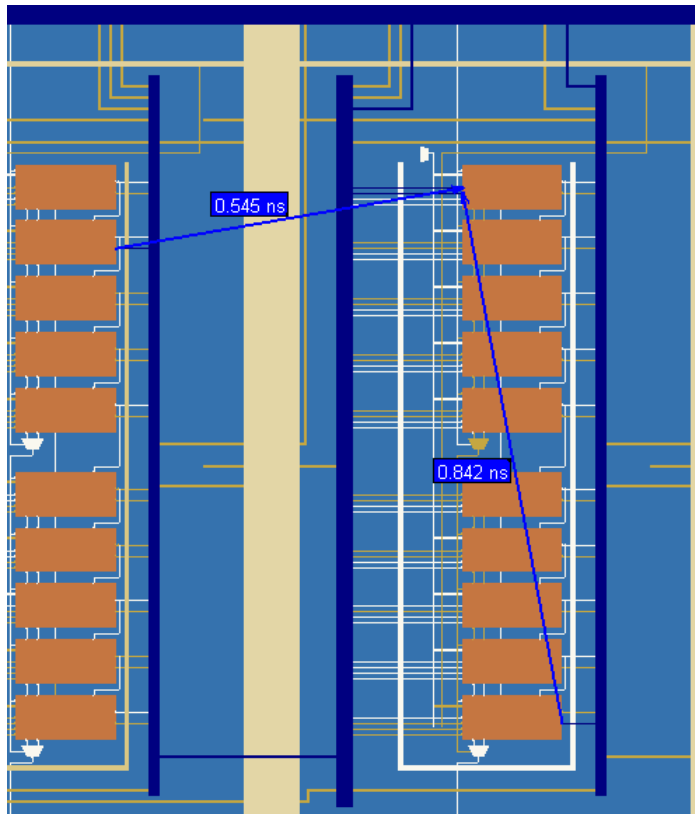
図 14-12.配線のハイライト



遅延の表示

エレメント間の接続を生成するときに、ハイライトされた接続のタイミング遅延を表示できます。例えば、2つのロジック・リソース間またはロジック・リソースと配線リソースの間の遅延を表示できます。図 14-13 に複数のロジック・エレメント間の遅延を示します。

図 14-13.遅延の表示



Chip Editor でのパスの探索

Chip Editor を使用してロジック・エレメント間のパスを探索できます。以下の例では、Chip Editor を使用してタイミング解析レポートからのパスを探します。

タイミング解析レポートから Chip Editor へのパスの検索

タイミング解析レポートから Chip Editor へのパスを検索するには、以下のステップを実行します。

1. 検索するパスを選択します。
2. タイミング解析レポートでパスを右クリックして、**Locate** を右クリックし、次に **Locate in Chip Editor** をクリックします (図 14-14)。


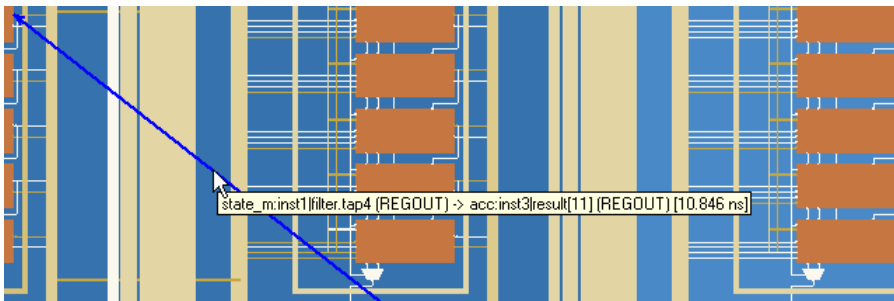
 表示される遅延は、タイミング解析レポートからのレジスタ間の遅延です。スキューまたはマイクロ・パラメータ計算は含まれません。

図 14-14. タイミング解析レポートからの検索

	Slack	Actual fmax (period)	From
1	16.913 ns	163.03 MHz (period = 6.134 ns)	dlsar_top:dlsar_toplutopia
2	17.705 ns	219.78 MHz (period = 4.550 ns)	dlsar_top:dlsar_toplutopia
3	17.711 ns	220.36 MHz (period = 4.538 ns)	dlsar_top:dlsar_toplutopia
4	17.721 ns	221.34 MHz (period = 4.518 ns)	dlsar_top:dlsar_toplutopia
5	17.721 ns	221.34 MHz (period = 4.518 ns)	dlsar_top:dlsar_toplutopia
6	17.721 ns	221.34 MHz (period = 4.518 ns)	dlsar_top:dlsar_toplutopia
7	17.721 ns	221.34 MHz (period = 4.518 ns)	dlsar_top:dlsar_toplutopia
8	17.721 ns	221.34 MHz (period = 4.518 ns)	dlsar_top:dlsar_toplutopia
9	17.721 ns	221.34 MHz (period = 4.518 ns)	dlsar_top:dlsar_toplutopia
10	19.721 ns	155.03 MHz (period = 3.64 ns)	dlsar_top:dlsar_toplutopia
11	19.721 ns	155.03 MHz (period = 3.64 ns)	dlsar_top:dlsar_toplutopia
12	28.232 ns	85.27 MHz (p	dlsar_top:dlsar_toplutopia
13	28.234 ns	85.28 MHz (p	dlsar_top:dlsar_toplutopia
14	28.244 ns	85.35 MHz (p	dlsar_top:dlsar_toplutopia
15	28.232 ns	85.27 MHz (p	dlsar_top:dlsar_toplutopia
16	28.234 ns	85.28 MHz (p	dlsar_top:dlsar_toplutopia
17	28.244 ns	85.35 MHz (p	dlsar_top:dlsar_toplutopia
18	28.272 ns	85.56 MHz (p	dlsar_top:dlsar_toplutopia

図 14-15 に Chip Editor に表示されるパスを示します。

図 14-15. 表示されたパス

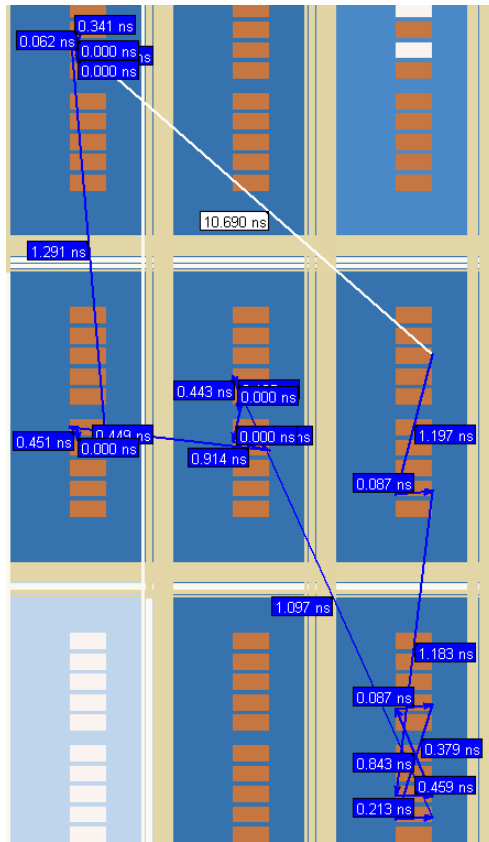


パスの接続の解析

Chip Editor のアイテム間の接続を決定するには、ツールバーの **Expand Connections/Paths** アイコンを使用します。各接続間のタイミング遅延を追加するには、ツールバーの **Show Delays** アイコンを使用します。

図 14-16 に Chip Editor に表示される選択したパスの接続を示します。

図 14-16.パスの解析



LogicLock 領域間の接続

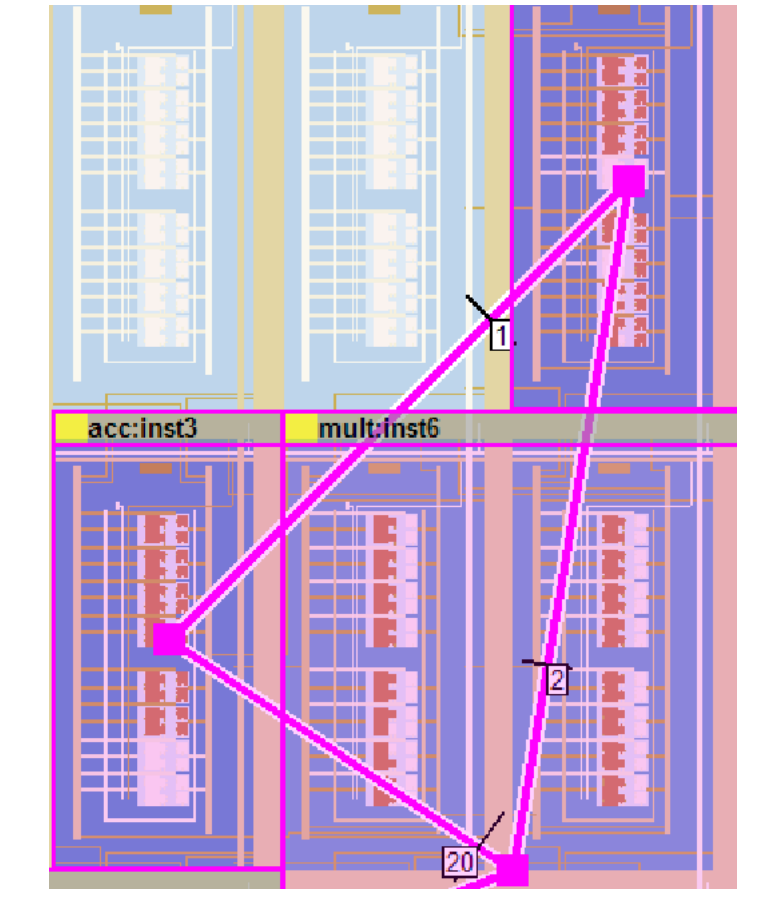
2つの LogicLock 領域間の複数の接続線を表示する代わりに、LogicLock 領域間の接続を1つにバンドルして表示するオプションを選択できます (図 14-17)。このオプションを使用するには、Chip Editor フロアプランを開き、View メニューで **Generate inter-region bundles** をクリックします。

Generate inter-region bundles ダイアログ・ボックスで、**Source node to region fanout less than** および **Bundle width greater than** の値を指定します。



Generate inter-region bundles ダイアログ・ボックスのパラメータについて詳しくは、Quartus II ヘルプを参照してください。

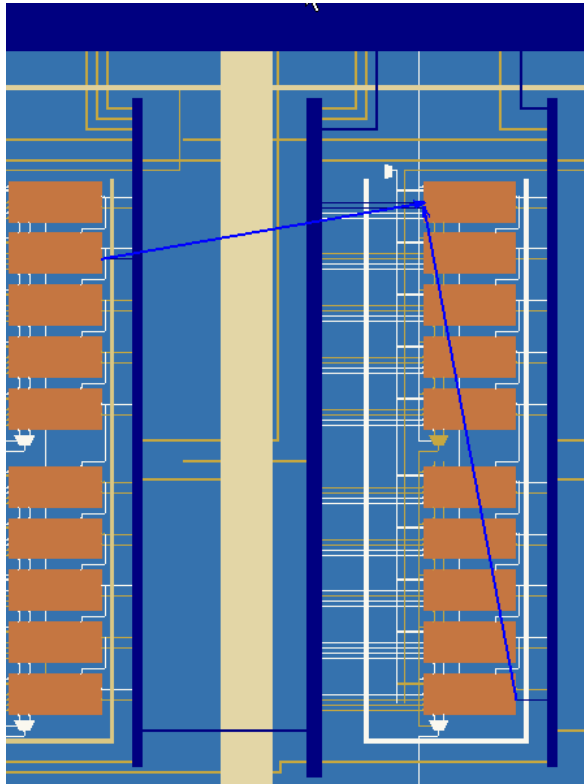
図 14-17. Generate Inter-Region Bundles ボックス



パスの配線チャンネル

接続間の配線チャンネルを決定するには、ツールバーの Highlight Routing アイコンをクリックします。図 14-18 に Chip Editor で選択したパスに使用する配線チャンネルを示します。

図 14-18.配線のハイライト



Resource Property Editor

Resource Property Editor では、以下のエレメントを表示および編集することができます。

- LE (Stratix、Stratix GX、Cyclone II、Cyclone、MAX II)
- ALM (Stratix II、Stratix II GX)
- I/O リソース
- PLL

ロジック・エレメント

アルテラの LE は、4 入力変数で構成されるファンクションを実装できる 4 入力 LUT を持っています。さらに、各 LE には LUT の出力または別の LE で生成される独立したファンクションによって供給されるレジスタがあります。

Resource Property Editor を使用して、FPGA 内の任意の LE を表示および編集することができます。以下の表示のいずれかの LE 上で **Locate in Resource Property Editor** (右クリック・メニュー) を選択して、Resource Property Editor を開きます。

- タイミング・クロージャ
- RTL Viewer
- Node Finder
- Chip Editor



特定のデバイス・ファミリの LE アーキテクチャについて詳しくは、デバイス・ファミリ・ハンドブックまたはデータ・シートを参照してください。

Resource Property Editor を使用して、以下の LE のプロパティを変更できます。

- 新しい LE 素子の作成
- 既存の LE 素子の移動
- LUT へのデータ入力
- LUT マスクまたは LUT 等式

ロジック・エレメント回路図ビュー

図 14-19 に Resource Property Editor に表示される LE を示します。図 14-19 は、DATAC および DATAD 入力と COMBOUT 出力を使用する LE を示しています。

図 14-19.Stratix LE アーキテクチャ 注 (1)、(2)

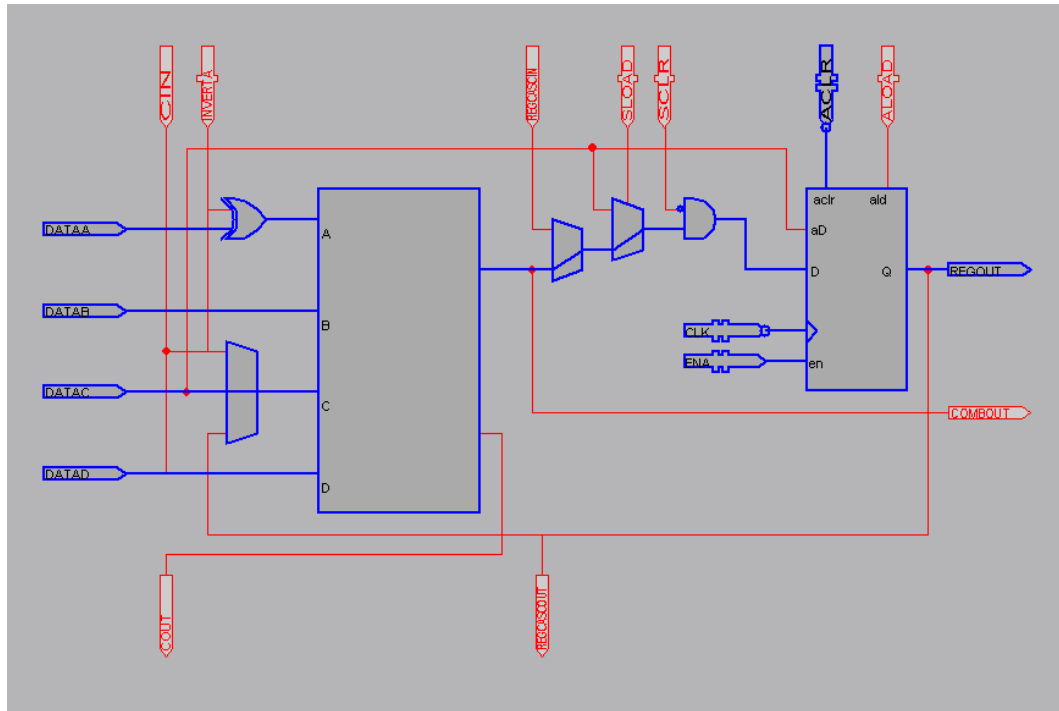


図 14-19 の注：

- (1) デフォルトでは、Quartus II ソフトウェアは、使用済みリソースを青、未使用のリソースをグレーで表示します。図 14-19 では、使用済みリソースを青、未使用リソースを赤で表示しています。
- (2) Stratix デバイスの LE アーキテクチャについて詳しくは、「Stratix デバイス・ハンドブック」を参照してください。

LE のプロパティ

図 14-20 に Resource Property Editor で表示される選択した LE のプロパティを示します。プロパティを表示するには、View メニューで **View Properties** をクリックします。

図 14-20.LE のプロパティ

Properties/Modes	Values	Sum Equation	D & (B # C) # ID & A & B	Node:	filterfaps_instxn_1[7]
LUT Mask	FC22	Carry Equation	N/A	COMBOUT(0)	REGOUT(0)
Sum LUT Mask	FC22			ACLRI(0)	N/A
Carry LUT Mask	N/A			CLK(0)	N/A
Operation Mode	normal			DATAA(0)	459 ps
Synchronous Mode	on			DATAB(0)	332 ps
Register Cascade Mode	off			DATAD(0)	87 ps
Latch Type	none				

動作モード

LE はノーマル・モードまたは演算モードで動作できます。



動作モードについて詳しくは、「Stratix デバイス・ハンドブック Volume 1」、
「Cyclone デバイス・ハンドブック Volume 1」または「MAX II デバイス・ハンドブック」を参照してください。

LE をノーマル・モードでコンフィギュレーションすると、LE の LUT は 4 入力のファンクションを実装できます。

LE を演算モードでコンフィギュレーションすると、LE の LUT は 2 つの 3 入力 LUT に分割されます。分割された LUT の内、最初の LUT は LUT の出力をドライブする信号を生成し、2 つ目の LUT はキャリー・アウト信号を生成します。キャリー・アウト信号は別の LE のキャリー・イン信号のみドライブできます。

SUM & CARRY 式

SUM & CARRY 式を変更して、LUT で実装されたロジック・ファンクションを変更することができます。LE をノーマル・モードでコンフィギュレーションした場合、変更できるのは SUM 式のみです。LE を演算モードでコンフィギュレーションすると、SUM 式と CARRY 式の両方を変更できます。

LUT マスクは LUT 等式を 16 進数で表したものです。LUT 等式を変更すると、Quartus II ソフトウェアで LUT マスクが自動的に変更されます。

LUT マスクを変更すると、Quartus II ソフトウェアで LUT 等式が自動的に計算されます。

同期モード

LEが同期モードのとき、同期ロード (sload) 信号と同期クリア (sclr) 信号が使用されます。sload 信号と sclr 信号を接続 (または切断) して、LEの同期モードを変更することができます。

LEに供給する sload 信号または sclr 信号を反転することができます。デザインの LEで sload 信号を使用する場合、信号と信号の反転状態は同じ LABの他のすべての LEと同じでなければなりません。例えば、LABの2個のLEが sload 信号で接続されている場合、両方のLEは同じ値の sload 信号を持っている必要があります。これは、sclr 信号にも当てはまります。

レジスタ・カスケード・モード

レジスタ・カスケード・モードがイネーブルされている場合、カスケード・イン・ポートがレジスタの入力に供給されます。デザインで一連のシフト・レジスタを実装する場合は、レジスタ・カスケード・モードが最もよく使用されます。カスケード・インを接続 (または切断) して、レジスタ・カスケード・モードを変更することができます。ただし、このポートを作成する場合、ソース・ポート LEがディスティネーション LEのすぐ上に位置していなければなりません。

セル遅延テーブル

セル遅延テーブルは、選択した LEのすべての入力と出力の間の伝播遅延を示します。

LE 接続

View メニューの **View Properties** をクリックして、LEに供給する接続およびLEから供給される接続を表示します。図 14-21は **View Properties** ウィンドウに表示された LE接続を示します。

図 14-21.View Properties

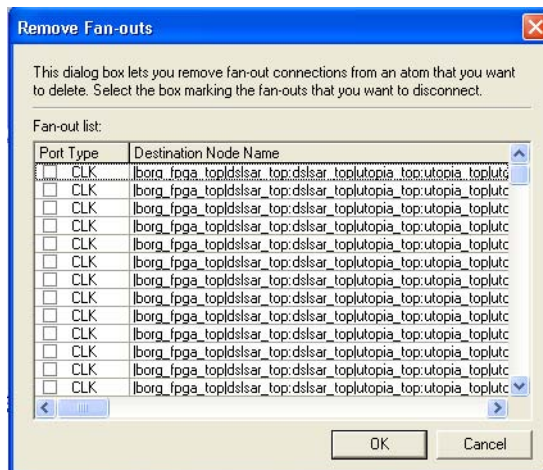
Input Port name	Signal name	Latch info	Output Port name	Signal name	Latch info
ACLK	filterreset	N/A	COMBOUT	filtertaps:inst:Selector0~14	N/A
ALOAD	<Disconnected>	N/A	COUT	<Disconnected>	N/A
CIN	<Disconnected>	N/A	REGCASOUT	<Disconnected>	N/A
CLK	filterclk	N/A	REGOUT	filtertaps:inst:tn_1[7]	N/A
DATAA	filtertaps:inst:tn [7]	N/A			
DATAB	filterstate_minst1sel[1]	N/A			
DATAD	filterstate_minst1sel[0]	N/A			
ENA	filterreset	N/A			

LE の削除

LE を削除するには、以下のステップを実行します。

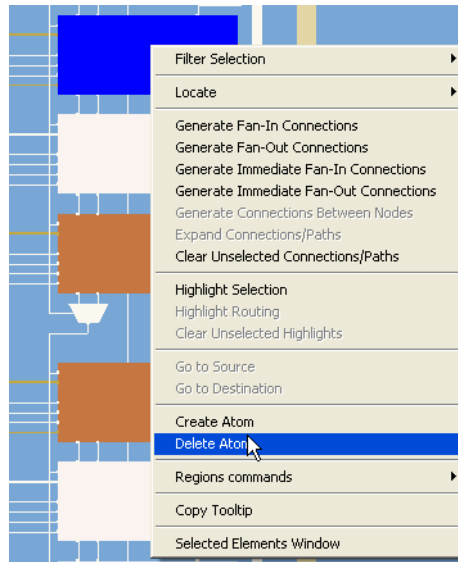
1. Resource Property Editor で LE を検索します。右クリック・メニューから **Locate** を選択し、LE で **Locate in Resource Property Editor** をクリックします。
2. ファン・アウト接続を削除します。Resource Property Editor で LE を検索したら、ファン・アウト接続を削除します。右クリック・メニューで **Remove** を選択し、すべての出力で **Fanouts** をクリックします。図 14-22 に表示されるダイアログ・ボックスを示します。

図 14-22. ファン・アウト接続の削除



3. すべてのファン・アウト接続を削除したら素子を削除し、Chip Editor で素子を再度参照します。右クリックして **Delete Atom** を選択します (図 14-23)。

図 14-23.素子の削除



新しい LE の作成

Chip Editor で LE を作成するには、以下のステップを実行します。

1. Chip Editor の何もない場所で、**Create Atom** を右クリックします。
2. **Create Logic Cell Atom** ダイアログ・ボックスに素子の名前を指定します。
3. 素子を作成したら、**Edit Connection** を選択し、入力で **Other** をクリックして作成した素子を他の素子に接続します。表示されるダイアログ・ボックスに接続する信号の名前を入力します。信号の名前が分からない場合は、**Node Finder** を使用して信号を見つけることができます。
4. **SUM** 式を設定して LE の機能を変更するにはロジック・セル・プロパティの **Sum Equation** フィールド内でダブルクリックして、式を変更します。図 14-24 に **Sum Equation** フィールドと **Carry Equation** フィールドを示します。

図 14-24.LUT 等式

Sum Equation	A \$ B \$ C
Carry Equation	A & IB & IC # IA & (IC # IB)

アダプティブ・ロジック・モジュール

Stratix II アーキテクチャのロジックの基本的なビルディング・ブロックは ALM です。ALM は効率的なロジック利用を可能にする最新機能を提供します。各 ALM は、2 個のアダプティブ LUT (ALUT) の間で分割可能なさまざまな LUT ベースのリソースを備えています。2 個の ALUT への最大 8 本の入力により、各 ALM で 2 つのファンクションのさまざまな組み合わせを実装できます。この適応性により、ALM は 4 入力 LUT アーキテクチャとの完全な下位互換性を提供します。1 個の ALM で、最大 6 本の入力を持つ任意のファンクションおよび特定の 7 入力ファンクションを実装することが可能です。アダプティブ LUT ベースのリソースに加えて、各 ALM には 2 個のプログラマブル・レジスタ、2 個の専用の全加算器、1 本のキャリー・チェーン、1 本の共有演算チェーン、および 1 本のレジスタ・チェーンも含まれています。これらの専用リソースにより、ALM は様々な演算ファンクションやシフト・レジスタを効率的に実装することができます。

1 個の ALM には、以下のタイプのファンクションを実装できます。

- 2 つの独立した 4 入力ファンクション
- 独立した 5 入力ファンクションおよび独立した 3 入力ファンクション
- 5 入力ファンクションおよび 4 入力ファンクション (1 つの入力を共有する場合)
- 2 つの 5 入力ファンクション (2 つの入力を共有する場合)
- 独立した 6 入力ファンクション
- 2 つの 6 入力ファンクション (4 つの入力と共有ファンクションを共有する場合)
- 特定の 7 入力ファンクション

以下の ALM プロパティを変更することができます。

- 新しい ALM 素子の作成
- 既存の ALM 素子の移動
- LUT マスクまたは LUT 等式

ALM 回路図

Resource Property Editor では、Stratix II デバイスの任意の ALM を表示および編集することができます (図 14-25)。特定の ALM を Resource Property Editor で表示するには、タイミング・クロージャ・フロアプラン、RTL Viewer、Node Finder、または Chip Editor の ALM で右クリックします。Locate in Resource Property Editor をクリックします。



ALM について詳しくは、「Stratix II デバイス・ハンドブック」を参照してください。

図 14-25.ALM 回路図 注 (1)

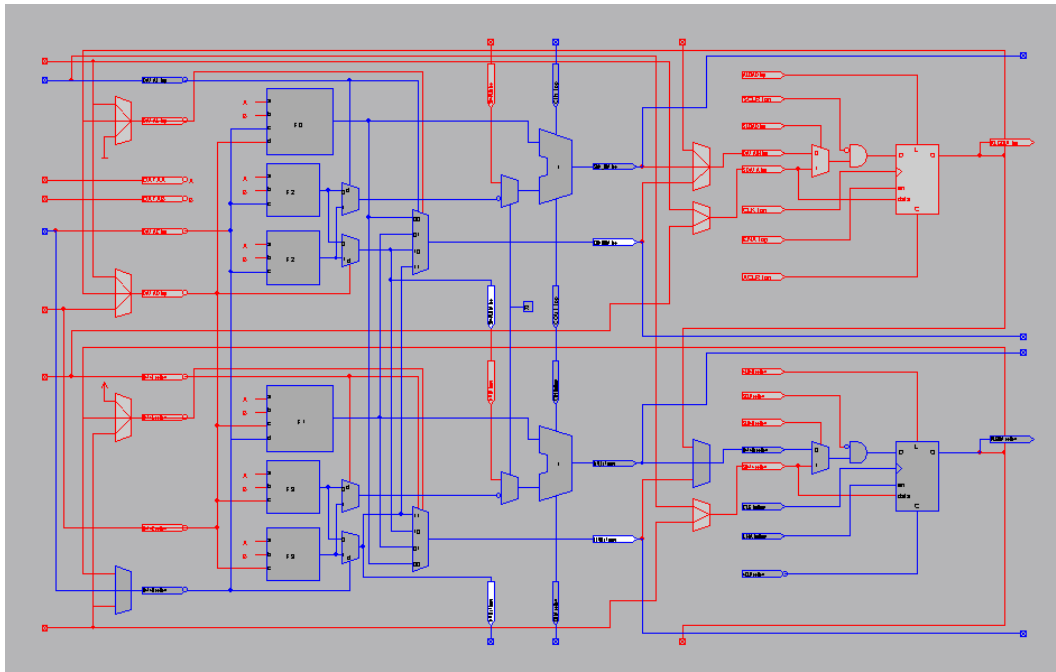


図 14-25 の注：

- (1) デフォルトでは、Quartus II ソフトウェアは、使用済みリソースを青、未使用のソースをグレーで表示します。図 14-25 では、使用済みリソースを青、未使用リソースを赤で表示しています。

ALM プロパティ

Stratix II の ALM に表示されるプロパティには、ALM の 2 つの組み合わせノードと 2 つのレジスタ・ノードの名前と位置を示す等式テーブル、各組み合わせノードの個々の LUT 等式、および combout、sumout、carryout、および shareout 式が含まれます。

図 14-26 に選択した ALM で使用するプロパティを示します。

図 14-26. ALM プロパティ

Properties/Modes	Values		
F0 LUT Mask	FFFF		
F1 LUT Mask	FFFF		
F2 LUT Mask	0000		
F3 LUT Mask	0000		

[-] Top Combinational Node	filterfaps.instbn_1[4]*feeder
Location String	LCCOMB_X31_Y12_N12
Latch Type	none
F0 LUT Equation	vcc
F1 LUT Equation	N/A
F2 LUT Equation	gnd
F3 LUT Equation	N/A
Combout Equation	F
Sumout Equation	N/A
Carryout Equation	N/A
Shareout Equation	N/A
[-] Top Register Node	filterfaps.instbn_1[4]
Location String	LCFF_X31_Y12_N13
[-] Bottom Combinational Node	filterfaps.instbn[4]*feeder
Location String	LCCOMB_X31_Y12_N14
Latch Type	none
F0 LUT Equation	N/A
F1 LUT Equation	vcc
F2 LUT Equation	N/A
F3 LUT Equation	gnd
Combout Equation	F
Sumout Equation	N/A
Carryout Equation	N/A
Shareout Equation	N/A
[-] Bottom Register Node	filterfaps.instbn[4]
Location String	LCFF_X31_Y12_N15

図 14-26 では、ALM は Bottom Combinational Node を使用しています。このノードの等式は combout 式で表されます。ALM で使用する入力を決めるには、等式を調べます。この例では、DATAE と DATAF でボトム・マルチプレクサをドライブする選択信号を制御し、DATAA、DATAB、および DATAC 信号で LUT をドライブしています。素子の等式では ALM の入力に対応する変数を使用します。例えば、DATAA 入力は等式の “A” 変数に対応します。

ALM タイミング・テーブル

図 14-27 に選択した ALM のすべての入力と出力の間の伝播遅延を示します。

図 14-27.ALM タイミング

Node: snap xmit_packet:xmit_packet dest_id[7]~620	
	COMBOUT[0]
DATAA[0]	382 ps
DATAD[0]	275 ps
DATAE[0]	155 ps
DATAF[0]	53 ps

ALM の接続

View メニューの **View Properties** をクリックして、LE に供給する接続および LE から供給される接続を表示します (図 14-28)。

図 14-28.ALM の接続

Input Port name	Signal name	Latch info	Output Port name	Signal name	Latch info
ACLRL bottom	fitrefreset~clkctrl	N/A	COMBOUT bottom	fitrefitaps:instn[4]~feeder	N/A
ACLRL top	fitrefreset~clkctrl	N/A	COMBOUT top	fitrefitaps:instn_1[4]~feeder	N/A
ALOAD bottom	<Disconnected>	N/A	COOUT bottom	<Disconnected>	N/A
ALOAD top	<Disconnected>	N/A	COOUT top	<Disconnected>	N/A
CIN bottom	<Disconnected>	N/A	REGOUT bottom	fitrefitaps:instn[4]	N/A
CIN top	<Disconnected>	N/A	REGOUT top	fitrefitaps:instn_1[4]	N/A
CLK bottom	fitrefclk~clkctrl	N/A	SHAREOUT bottom	<Disconnected>	N/A
CLK top	fitrefclk~clkctrl	N/A	SHAREOUT top	<Disconnected>	N/A
DATAA	<Disconnected>	N/A	SUM_OUT bottom	<Disconnected>	N/A
DATAB	<Disconnected>	N/A	SUM_OUT top	<Disconnected>	N/A

FPGA の I/O エLEMENT

最大 6 個のレジスタでバックされた高性能 I/O エLEMENT を備えたアルテラ FPGA は、多数の標準 I/O エLEMENT 規格をサポートし、デザインを高速で実行できます。



Stratix II デバイスの I/O エLEMENT について詳しくは、「Stratix II デバイス・ハンドブック Volume 1」の「Stratix II アーキテクチャ」の章を参照してください。

Stratix デバイスの I/O エLEMENT について詳しくは、「Stratix デバイス・ハンドブック Volume 1」の「Stratix アーキテクチャ」の章を参照してください。

Cyclone II デバイスの I/O エlement について詳しくは、「Cyclone II デバイス・ハンドブック」の「Cyclone II アーキテクチャ」の章を参照してください。

Cyclone デバイスの I/O エlement について詳しくは、「Cyclone デバイス・ハンドブック Volume 1」の「Cyclone アーキテクチャ」の章を参照してください。

MAX II デバイスの I/O エlement について詳しくは、「MAX II デバイス・ハンドブック」の「MAX II アーキテクチャ」の章を参照してください。

以下の I/O プロパティを変更することができます。

- 新しい I/O 素子の作成
- 既存の I/O 素子の移動
- 遅延チェーン
- バス・ホールド
- ウィーク・プル・アップ
- 低速スルー・レート
- 標準 I/O 規格
- 電流強度
- 拡張 OE ディセーブル
- PCI I/O
- レジスタ・リセット・モード
- レジスタ同期リセット・モード
- レジスタ・パワー・アップ
- レジスタ・モード

Stratix II、Stratix、Stratix II GX および Stratix GX I/O エlement

Stratix シリーズ・デバイス・ファミリの I/O は、1 個の双方向の I/O バッファ、6 個のレジスタ、および完全に双方向のシングル・データ・レートまたは DDR 転送に対応するラッチで構成されています。図 14-29 に Stratix と Stratix GX の I/O エlement 構造を示します。I/O エlement 構造は、2 個の入力レジスタ（および 1 個のラッチ）、2 個の出力レジスタ、および 2 個の出力イネーブル・レジスタで構成されています。図 14-30 に Stratix II および Stratix II GX の I/O エlement 構造を示します。

図 14-29.Stratix および Stratix GX デバイスの I/O エLEMENT または構造 注 (1)、(2)

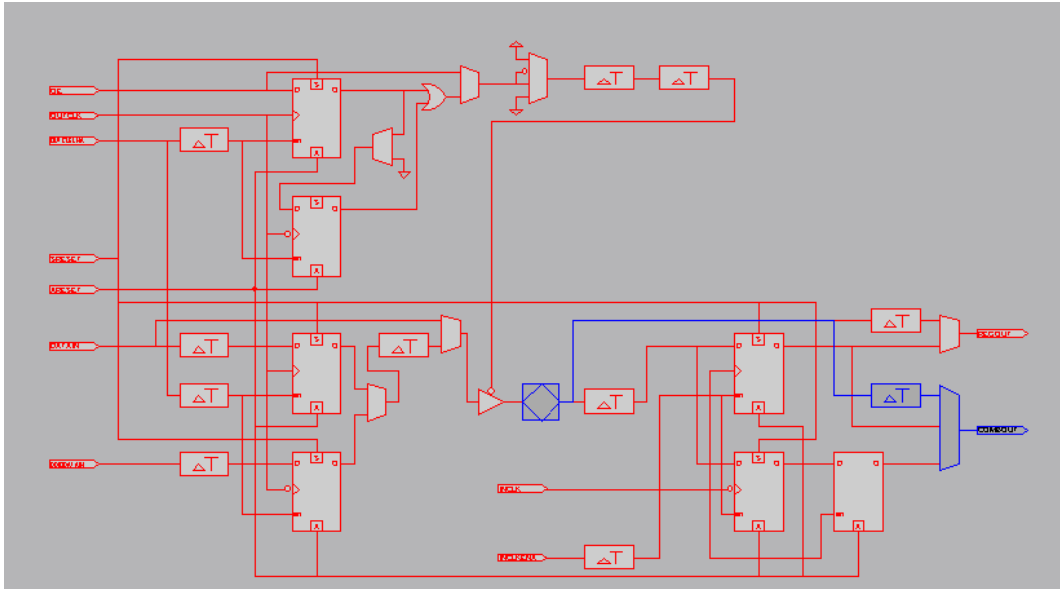


図 14-29 の注：

- (1) デフォルトでは、Quartus II ソフトウェアは、使用済みリソースを青、未使用のリソースをグレーで表示します。図 14-29 では、使用済みリソースを青、未使用リソースを赤で表示しています。
- (2) Stratix および Stratix GX デバイスの I/O エLEMENT について詳しくは、「Stratix デバイス・ハンドブック」および「Stratix GX デバイス・ハンドブック」を参照してください。

MAX II の I/O エLEMENT

MAX II デバイスの I/O エLEMENT は、1 個の双方向の I/O バッファで構成されています。図 14-32 に MAX II の I/O エLEMENT 構造を示します。隣接する LAB のレジスタは I/O エLEMENT の双方向の I/O バッファをドライブすることができます。あるいは、またはそれらの I/O バッファからレジスタをドライブすることができます。

図 14-32. MAX II デバイスの I/O エLEMENT または構造 注 (1)、(2)

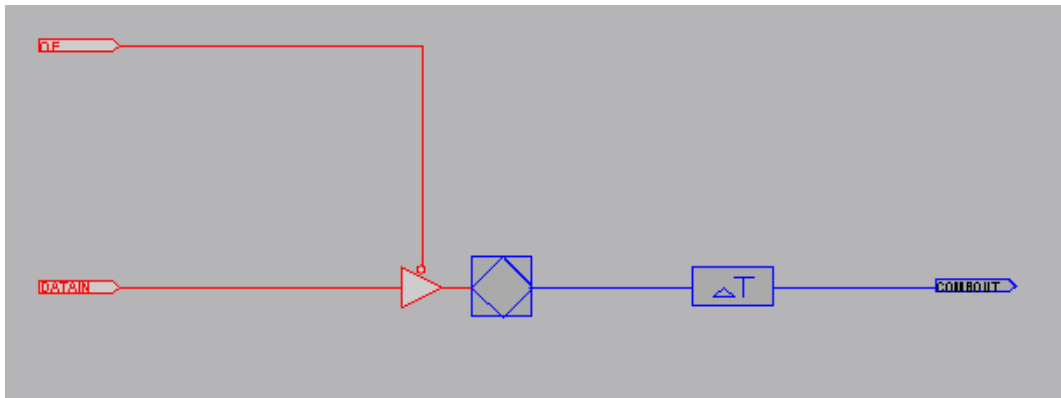


図 14-32 の注：

- (1) デフォルトでは、Quartus II ソフトウェアは、使用済みリソースを青、未使用のソースをグレーで表示します。図 14-32 では、使用済みリソースを青、未使用リソースを赤で表示しています。
- (2) MAX II デバイスの I/O エLEMENT について詳しくは、「MAX II デバイス・ハンドブック」を参照してください。

Resource Property Editor での I/O エLEMENT の特長

Resource Property Editor を使用して、接続の表示、接続の編集、および I/O エLEMENT のプロパティの編集を行います。Chip Editor を使用して、I/O エLEMENT の配置の変更、削除、および新しい I/O エLEMENT の作成を行います。これらの操作はすべて、Stratix II、Stratix II GX、Stratix、Stratix GX、Cyclone II、Cyclone、および MAX II の各デバイスでも行うことができます。

Chip Editor を使用した PLL の変更

PLL はクロック信号を変更および生成して、デザイン要件を満たすために使用されます。また、クロック信号をデザイン内のさまざまなデバイスへの分配、デバイス間のクロック・スキューの低減、I/O タイミングの改善、および内部クロック信号の生成の目的でも使用されます。

PLL プロパティ

Resource Property Editor では、位相シフト、出力クロック周波数、デューティ・サイクルなどの PLL オプションを変更できます。さらに、以下を含む多くの PLL プロパティを変更することもできます。

- 入力周波数
- MVCO タップ
- M イニシャル
- M 値
- N 値
- M カウンタ遅延
- N カウンタ遅延
- M2 値
- N2 値
- SS カウンタ
- チャージ・ポンプ電流
- ループ・フィルタ抵抗
- ループ・フィルタ・キャパシタンス
- カウンタ遅延
- カウンタ High
- カウンタ Low
- カウンタ・モード
- イニシャル
- VCO タップ

デューティ・サイクルの調整

以下の等式を使用して、個々の出力クロックのデューティ・サイクルを調整します。

- (1) **High % = カウンタ High / (カウンタ High + カウンタ Low)**
Low % = カウンタ Low / (カウンタ High + カウンタ Low)

位相シフトの調整

以下の等式を使用して、PLL の出力クロックの位相シフトを調整します。

- (2) **位相シフト = (周期 VCO × 0.125 × タップ VCO) + (初期 VCO × 周期 VCO)**



設定の詳細については、Quartus II ヘルプを参照してください。Stratix デバイス PLL について詳しくは、「Stratix デバイス・ハンドブック Volume 1」の「Stratix アーキテクチャ」の章を参照してください。

ノーマル・モードでは、以下の設定で位相シフトを計算します。

$$\begin{aligned} \text{タップ VCO} &= \text{カウンタ遅延} - M \text{ タップ VCO} \\ \text{イニシャル VCO} &= \text{カウンタ} \cdot \text{イニシャル} - M \text{ イニシャル} \\ \text{周期 VCO} &= \text{イン} \cdot \text{クロック周期} \times N / M \end{aligned}$$

外部フィードバック・モードでは、以下の設定で位相シフトを計算します。

$$\begin{aligned} \text{タップ VCO} &= \text{カウンタ遅延} - M \text{ タップ VCO} \\ \text{イニシャル VCO} &= \text{カウンタ} \cdot \text{イニシャル} - M \text{ イニシャル} \\ \text{周期 VCO} &= \text{イン} \cdot \text{クロック周期} \times N / (M + \text{カウンタ High} + \text{カウンタ Low}) \end{aligned}$$

出力クロック周波数の調整

以下の等式をノーマル・モードで使用して、PLL 出力を調整します。

$$(3) \quad \text{Output Clock Frequency} = \text{Input Frequency} \cdot \frac{M \text{ initial}}{N \text{ initial} + \text{Counter High} + \text{Counter Low}}$$

以下の等式を外部フィードバック・モードで使用して、PLL 出力クロックを調整します。

$$(4) \quad \text{OUTCLK} = \text{INCLK} \cdot \frac{M \text{ initial} + \text{External Feedback Counter High} + \text{External Feedback Counter Low}}{N \text{ initial} + \text{Counter High} + \text{Counter Low}}$$

スペクトラム拡散の調整

以下の等式を使用して PLL のスペクトラム拡散を調整します。

$$(5) \quad \% \text{spread} = 1 - \frac{M_2 N_1}{M_1 N_2}$$

Change Manager

Change Manager では、Resource Property Editor で実行した変更の記録を維持します。Change Manager の各ロウは、Resource Property Editor で行った 1 つの変更を表します。変更には順番に番号が付けられ、番号が大きいほど最近行った変更です。

より複雑な変更には、Change Manager に “+” マークが付けられます。Change Manager で複雑なエントリを展開して、すべての変更内容を明示できます。複雑な変更の一例として、素子の作成または削除があります。

表 14-1 に Change Manager に表示される情報を要約します。

表 14-1. Change Manager 情報	
カラム名	説明
Index	Chip Editor または Resource Property Editor で行った変更に対応する変更記録を、連番で識別します。 複雑な変更記録の場合、Index カラムは主な変更だけでなく部分的な変更も識別します。
Node Name	変更を行ったリソースを個別に識別します。
Change Type	リソースに行った変更の種類を識別します。
Old Value	変更を行う直前のリソースの値をリストします。
Target Value	Resource Property Editor、Chip Editor、または SignalProbe を使用して目的のターゲット値 (新しい値) をリストします。
Current Value	現在メモリでアクティブなネットリストのリソースの値をリストします (ディスクに保存したネットリストの値とは異なり、変更を行ったが、まだ Check and Save All Netlist Changes コマンドを使用していない場合は値が異なる可能性があります)。 Current Value フィールドには、以下の 4 つの可能な値のいずれかを含むことができます。 <ul style="list-style-type: none"> ● current value = old value。変更が反映されていないことを示します。 ● current value = target value。変更が反映されていることを示します。 ● current value = neither old nor target value。無効な変更を試みたことを示します。 ● current value = Data Not Available 不完全なデータ・セットを使用している可能性を示します。これは以下の例で発生する可能性があります。 <ul style="list-style-type: none"> - 使用中のデザインが現在のコンピュータでコンパイルされていないため、ネットリストが存在しません。 - ソース・コードが変更されたため、変更しようとしているノードがネットリストに存在しません。 - ノード名は最初にコンパイラにより割り当てられましたが、異なる最適化パラメータを使用してフィッタ処理を実行した結果、ノード名が変更されたため、変更しようとしているノードが存在しません。
Disk Value	ディスク上のリソースの現在の値をリストします。
Comment	Change Manager の変更記録にコメントを追加します。 変更記録にコメントを追加するには、アノテートする記録の Comment フィールドをダブルクリックして、コメントを入力します。

デザインの変更が完了したら、Change Manager 内で右クリックし、**Check & Save All Netlist Changes** をクリックして、ネットリストの完全性をチェックします。適用された変更が正常にネットリスト・チェックに合格した場合は、ディスクに書き込まれます。変更がネットリスト・チェックに合格しない場合は、前回成功したネットリスト・チェック以降に行った変更はすべて元に戻されます。図 14-33 に Change Manager を示します。

Current Value および **Disk Value** カラムの色付きインジケータは、これらのカラムのデータの現在の状態を示します。**Current Value** カラムの緑で表示されている部分は、変更が行われた値がメモリの値と同じになったことを示します。**Disk Value** カラムの青で表示される部分は、変更がネットリスト・チェックに合格したことを示します。**Check & Save All Netlist Changes** (右クリック・メニュー) を選択すると、そのカラムの値がディスクの値と同じになります。

図 14-33.Change Manager の結果

Index	Node Name	Change Type	Old Value	Target Value	Current Value	Disk Value
1	test1	SignalProbe	Disconnected	filterinst4	Disconnected	Disconnected
2	mult_inst6lpm_mult_lpm_mult_componentImu...	Modify Source	filtertaps.in...	Disconnected	filtertaps instbx[4]	filtertaps instbx [4]
3	mult_inst6lpm_mult_lpm_mult_componentImu...	Modify Source	filtertaps.in...	Disconnected	filtertaps instbx[4]	filtertaps instbx [4]
4	taps instbx_1[4] feeder:DATAF:0	Modify Source	filtertaps.in...	Disconnected	filtertaps instbx[4]	filtertaps instbx [4]
5	mult_inst6lpm_mult_lpm_mult_componentImu...	Modify Source	filtertaps.in...	Disconnected	Disconnected	Disconnected
6	mult_inst6lpm_mult_lpm_mult_componentImu...	Modify Source	filtertaps.in...	Disconnected	Disconnected	Disconnected
7	taps instbx_1[0] feeder:DATAF:0	Modify Source	filtertaps.in...	Disconnected	Disconnected	Disconnected

Change Manager: Netlist Check Required -- 3 Pending Changes

Change Manager の各行は、変更された記録を表します。簡単な変更は 1 行で表示されますが、変更を達成するのに何回かの処理を必要とするより複雑な変更は、“+” が付いた 1 行の形で表示されます。“+” をクリックすると、変更の一環として実行されるすべてのコンポーネントの処理が表示されます。

Change Manager での複雑な変更

Resource Property Editor または **Chip Editor** で行う特定の種類の変更 (素子の作成または削除、接続の変更など) は、自己完結型のように見えますが、実際には複数の処理を必要とします。これらの種類の変更は、**Index** カラムに “+” サインを付けて記録されます。図 14-34 に Change Manager を示します。

図 14-34.Change Manager

Index	Node Name	Change Type	Old Value	Target Value	Current Value	Disk Value
1	test1	SignalProbe	Disconnected	fitrefinet4	Disconnected	Disconnected

図 14-34 に示す例では、新しい素子が作成されます。**Change Manager** の変更記録では、実際の変更処理を 1 行で表します。“+” アイコンをクリックすると、変更記録が展開され、変更を行うコンポーネントの処理が表示されます (図 14-35)。“+” をクリックすると、リストが展開され、+ が“-” に変わります。


図 14-35.Change Manager の結果

Index	Node Name	Change Type	Old Value	Target Value	Current Value	Disk Value
+ 1	test	New Lcell Comb	None	Exists	Exists	None
2	Iniosii_mandelbrot_toplmandelbrot_sys:instr...	Location Index	LCCOMB_X...	LCCOMB_X...	LCCOMB_X...	LCCOMB_...
3	Iniosii_mandelbrot_toplmandelbrot_sys:instr...	Location Index	LCCOMB_X...	LCCOMB_X...	LCCOMB_X...	LCCOMB_...
4	Iniosii_mandelbrot_toplmandelbrot_sys:instr...	Location Index	LCCOMB_X...	LCCOMB_X...	LCCOMB_X...	LCCOMB_...
+ 5	Iniosii_mandelbrot_toplmandelbrot_sys:instr...	Modify Source	Iniosii_mand...	Disconnected	Disconnected	Disconnec...

“+” をクリックすると、以下の 3 つの処理で構成される素子を作成することが分かります。

- 新しいロジック・セルの作成。
- 新たに作成したロジック・セルの出力ポートの作成。
- 新たに作成したロジック・セルへのロケーション・インデックスの割り当て。

複雑な変更記録の中から個々のコンポーネントを選択することはできません。複雑な変更記録の一部を選択すると、複雑な記録全体が選択されます。

 **Change Manager** での変更の管理例については、Quartus II ヘルプの「Example of Managing Changes With the Change Manager」を参照してください。

SignalProbe 信号の管理

Tools メニューの **SignalProbe Pins** ダイアログ・ボックスで作成する SignalProbe アサインメントは、Change Manager に記録されます。SignalProbe アサインメントを作成したら、Change Manager の右クリックメニューで **Revert to Last Saved Netlist** を選択して、SignalProbe アサインメントを素早く無効にすることができます。

図 14-36.SignalProbe ピン作成後の Change Manager の結果

Index	Node Name	Change Type	Old Value	Target Value	Current Value	Disk Value
1	test1	SignalProbe	Disconnected	filtrefinst4	filtrefinst4	Disconnected
1.1	test1	New Output Pin	None	Exists	Exists	None
1.2	test1:DATAIN:0	Modify Source	Disconnected	filtrefinst4	filtrefinst4	Disconnected

変更内容のエクスポート

変更内容はすべて、ツールコマンド言語 (Tcl) スクリプト、カンマ区切り値 (.csv) ファイル、またはテキスト (.txt) ファイルにエクスポートできます。Tcl ファイルでは、コンパイルで削除された変更を再適用するスクリプトを記述できます。また、作成した他の Quartus II ソフトウェア・プロジェクトに適用するスクリプトを記述することもできます。カンマ区切り値ファイルまたはテキスト・ファイルでは、変更内容のリストが表形式で表示されます。変更内容をエクスポートするには、以下のステップを実行します。

1. 右クリック・メニューで、**Export Changes** を選択します。
2. Tcl ファイル名を指定します。
3. **OK** をクリックします。

エクスポートが完了した Tcl ファイルで、類似する変更を別の Quartus II のデザインに実装することも可能です。

一般的な用途

Chip Editor の機能を使用して、システムを短時間で構築することができます。

- 内部信号を出力ピンに配線する
- I/O タイミングに適合させて PLL の位相シフトを調整する
- デザインでの機能的な欠陥を修正する

内部信号の出力ピンへの配線

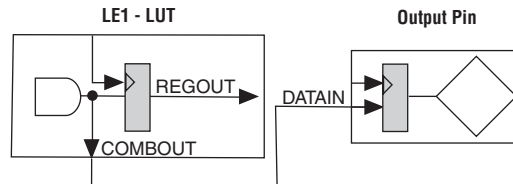
Chip Editor を使用して、内部信号を未使用の出力ピンに配線することができます。この機能により、外部ロジック・アナライザで FPGA 内部の信号をキャプチャできます。

これらの信号の配線プロセスは明解であり、短時間で実行できるため、セットアップ時間が短くてすみ、より多くの時間をデバッグに費やすことができます。

以下のステップでは、内部信号を出力ピンに配線するのに必要なプロセスを示します (図 14-37)。

1. 出力ピンを作成します。

図 14-37. 内部信号の出力ピンへの配線



2. ソース LE の REGOUT または COMBOUT が存在しない場合は、それを作成します。
3. 出力ピンの DATAIN をソース LE の REGOUT または COMBOUT に接続します。
4. オプション — クロックを出力ピンの CLK ポートに接続します。クロックを接続すると、信号が外部に出力される前にラッチすることができます。

I/O タイミングに適合した PLL の位相シフトの調整

デザインで PLL を使用すると I/O タイミングへの適合に役立ちます。ただし、I/O タイミング要件に適合していない場合は、PLL の位相シフトを調整してデザインの I/O タイミング要件を満足するように試みることができます。クロックを後方にシフトすると、 t_{CO} は改善しますが t_{SU} が増加し、前方にシフトすると t_{SU} は改善しますが t_{CO} と t_H が増加します。

14-35 ページから 14-36 ページの「Chip Editor を使用した PLL の変更」に示すとおり、等式を使用して新しい位相シフト値を設定して、I/O タイミングを最適化します。

Chip Editor 操作後の コマンド

ここでは、Chip Editor で変更を行った後に実行可能な操作について説明します。

Quartus II タイミング・アナライザの実行

Chip Editor で変更を行った後に、Quartus II タイミング・アナライザでデザインのタイミング解析を実行し、変更がデザインのタイミング性能に悪影響を与えていないことを確認します。

例えば、特定のピンの遅延チェーン設定の1つをオンにすると、I/O タイミングが変化します。したがって、すべてのタイミング要求に適合していることを確認するには、タイミング解析を実行する必要があります。

Chip Editor を使用してデザインを変更する場合は、その都度 Quartus II シミュレータまたは別の EDA ベンダのシミュレーション・ツールでデザインのタイミング・シミュレーションを実行することを推奨します。

他の EDA ツール用ネットリストの生成

Chip Editor を使用するときは、アルテラがサポートするシミュレーション・ツールを使用した機能の検証かタイミング解析ツールを使用したタイミングの検証、あるいはその両方の実行が必要な場合があります。Netlist Writer を実行してゲート・レベルのネットリストを生成し、EDA シミュレーション・ツールまたはタイミング解析ツールでシミュレーションまたはタイミング解析を実行できます。Processing メニューで **Start** をポイントし、**Start EDA Netlist Writer** をクリックして Netlist Writer を開きます。

プログラミング・ファイルの生成

シミュレーションおよびタイミング解析を実行し、変更がデザインの要件を満たしていると判断したら、Quartus II アセンブラでプログラミング・ファイルを生成します。プログラミング・ファイルを使用して、デザインをアルテラ・デバイスに実装します。

まとめ

「Time-to-Market」短縮の要求が高まる中、短期間で完全に機能するデザインを作成することがますます重要になっています。この課題に対応するために、アルテラは **Quartus II Chip Editor** を開発しました。**Chip Editor** により、デザインの配置配線後のプロパティを変更することができます。特に LE、I/O エlement、および PLL リソースの主要なプロパティの変更が可能です。最も重要な点は、**Chip Editor** で行った変更はフルに再コンパイルする必要がなく、手間の掛かる RTL 修正、再合成、新たな配置配線サイクルが要らないことです。

要約すると、**Chip Editor** は検証サイクルを短縮し、短期間でデザインのタイミング・クロージャに到達します。

