

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。

QII53004-6.0.0

はじめに

スタティック・タイミング解析は、デザインのタイミング性能の解析、デバッグ、および妥当性検査を行うための手段です。クラシック・タイミング・アナライザは、デザインのすべてのパスの遅延、およびすべてのタイミング要件を解析して、正しい回路動作を支援します。スタティック・タイミング解析を機能シミュレーションと併用すると、デザインの全体的な動作を検証できます。



TimeQuest タイミング・アナライザへの切り替えについて詳しくは、「Quartus II ハンドブック Volume 3」の「TimeQuest タイミング・アナライザへの切り替え」の章を参照してください。

Quartus® II ソフトウェアは、コンパイル・フローの一環として自動的にスタティック・タイミング解析を実行します。そのため、特にタイミング解析ツールを起動する必要はありません。クラシック・タイミング・アナライザは、デザイン内のすべてのパスにおいてタイミング違反の有無をタイミング制約と照合してチェックし、結果をタイミング解析レポートに反映して、すぐにタイミング解析レポートにアクセスできるようにします。

この章では、読者が Tcl に関して多少の専門知識を持っていることを前提としています。この章全体で、タイミング解析用アサインメントを行うための代替方法の説明に Tcl コマンドを使用しています。GUI に相当するタイミング制約については、[8-41 ページの「Quartus II GUI を使用したタイミング解析」](#)を参照してください。

この章では、タイミング解析の以下の側面について説明します。

- スタティック・タイミング解析の概要
- クロック設定
- クロックの種類
- クロック不確実性
- クロック・レイテンシ
- タイミング例外
- I/O 解析
- 非同期パス
- スキュー管理
- report_timing コマンドでのタイミング解析レポートの生成
- その他のタイミング・アナライザ機能
- Quartus II GUI を使用したタイミング解析
- スクリプトのサポート

スタティク・ タイミン 解析の概要

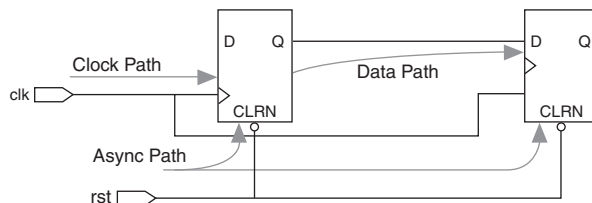
■ MAX+PLUS® II のタイミン解析手法

ここでは、この章全体、およびクラシック・タイミン・アナライザによって使用されるスタティク・タイミン解析の概念に関する情報を提供します。この項に示す概念を完全に理解すると、Quartus II ソフトウェアで提供されている強力なスタティク・タイミン解析機能を利用できるようになります。

どのデザインにも、あるレジスタの出力から別のレジスタの入力へのパスなど、デザイン・エレメントを互いに結合するさまざまなパスが存在します。タイミン・パスはスタティク・タイミン解析において重要な役割を果たします。タイミン・クロージャおよび最適化のために、タイミン・パスのタイプを理解することが重要です。ここでは通常解析されるパスのいくつかを説明します（図 8-1 を参照）。

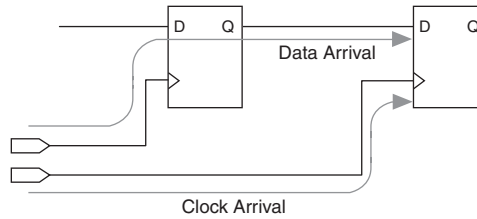
- クロック・パス — クロック・パスとは、デバイス・ピンまたは内部で生成されたクロック（クロック設定によってクロックとして指定されたノード）からレジスタなどのシーケンシャル・エレメントのクロック・ポートまでのパスのことです。
- データ・パス — データ・パスとは、あるシーケンシャル・エレメントのデータ出力ポートから別のシーケンシャル・エレメントのデータ入力ポートまでのパスのことです。
- 非同期パス — 非同期パスとは、ノードからシーケンシャル・エレメントの非同期セットまたはクリア・ポートまでのパスのことです。

図 8-1. パスの種類



クラシック・タイミン・アナライザは、パスのタイプが識別されると、すべての有効なレジスタ間パスのデータ到達時間とクロック到達時間を計算します。データ到達時間とは、ソース・クロックからデスティネーション・レジスタまでの遅延のことです。クラシック・タイミン・アナライザは、ソース・レジスタまでのクロック・パス遅延、ソース・レジスタの Clock-to-Out (μt_{CO})、およびソース・レジスタからデスティネーション・レジスタまでのデータ・パス遅延を加算することによってこの遅延を計算します。クロック到達時間とは、デスティネーション・クロック・ノードからデスティネーション・レジスタまでの遅延のことです。図 8-2 にデータ到達パスとクロック到達パスを示します。

図 8-2. データ到達およびクロック到達

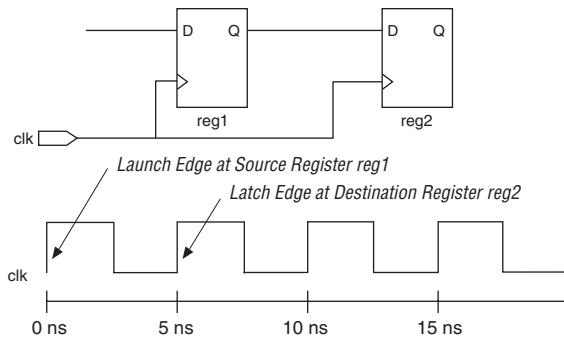


クラシック・タイミング・アナライザは、デザイン内のさまざまなパスを識別するほか、クロック特性を解析してレジスタ間パスのワースト・ケース値を計算します。この解析が実行される前に、タイミング制約を使用してデザイン内のすべてのクロック信号の特性を指定しなければなりません。

データ転送のソースとして機能する、シーケンシャル・エレメントのデータ出力を送出するアクティブ・クロック・エッジは送り出しエッジです。データ転送のデスティネーションとして機能する、シーケンシャル・エレメントのデータ・ポートでデータをキャプチャするアクティブ・クロック・エッジはラッチ・エッジです。

図 8-3 に、連続クロック・エッジを使用してデータを送信およびキャプチャするシングル・サイクル・システム、レジスタ間パス、および対応する送り出しエッジとラッチ・エッジのタイミング図を示します。この例では、0 ns に送り出しエッジでレジスタ reg1 からデータが送出され、5 ns にレジスタ reg2 のラッチ・エッジでデータがキャプチャされます。

図 8-3. 送り出しエッジおよびラッチ・エッジ



クラシック・タイミング・アナライザは、送り出しエッジおよびラッチ・エッジに対する特定パスを解析することによって、クロックのセットアップとホールド・チェックを実行し、タイミング制約と照合してそれらの妥当性検査を実行します。


クロック解析

包括的なスタティック・タイミング解析には、レジスタ間パス、I/O パス、および非同期リセット・パスの解析が含まれます。スタティック・タイミング解析ツールは、データ所要時間、データ到達時間、およびクロック到達時間を使用して、回路性能を検証し、起こり得るタイミング違反を検出します。クラシック・タイミング・アナライザは、デザインが正しく機能するために満足する必要があるタイミング関係を参照し、到達時間を所要時間と照合してタイミングを検証します。

クロック・セットアップ・チェック

クラシック・タイミング・アナライザは、デザインが性能を満たしているか否かを判断するために、クロック・タイミング、タイミング要求、およびタイミング例外を計算し、各デスティネーション・レジスタで、ソース・クロックおよびデスティネーション・クロックとタイミング制約、またはそれらのパスに適用可能な例外に基づいてクロックのセットアップ・チェックを実行します。クロック・セットアップ・チェックに問題がなければ、ソース・レジスタから送り出されたデータがデスティネーション・レジスタに正しくラッチされます。クラシック・タイミング・アナライザは、クロック・セットアップ・チェックを実行するために、データ到達時間には最長パス、クロック到達時間には最短パスを使用して、デスティネーション・レジスタにおけるクロック到達時間とデータ到達時間を決定します。クラシック・タイミング・アナライザは次に、[計算式 1](#) に示すように、差がデスティネーション・レジスタのセットアップ (t_{SU}) よりも大きいとそれと等しいことをチェックします。

- (1) $\text{Clock Arrival Time} - \text{Data Arrival Time} \geq t_{SU}$

 デフォルトでは、クラシック・タイミング・アナライザは送り出しエッジとラッチ・エッジが連続したアクティブ・クロック・エッジで発生すると想定しています。

クロック・セットアップ・チェックの結果は、スラックに換算してレポートされます。スラックとは、それによってタイミング要件が適合または不適合になるマージンのことです。正のスラックは要件に適合するマージンを示し、負のスラックは要件に適合しないマージンを示します。クラシック・タイミング・アナライザは、[計算式 2](#) ~ [計算式 5](#) を使用してクロック・セットアップのスラックを決定します。

- (2) $\text{Clock Setup Slack} = \text{Data Required Time} - \text{Data Arrival Time}$
- (3) $\text{Data Required} = \text{Clock Arrival Time} - t_{\text{SU}} - \text{Setup Uncertainty}$
- (4) $\text{Clock Arrival Time} = \text{Latch Edge} + \text{Shortest Clock Path to Destination Register}$
- (5) $\text{Data Arrival Time} = \text{Launch Edge} + \text{Longest Clock Path to Source Register} + \text{micro } t_{\text{CO}} + \text{Longest Data Delay}$

クラシック・タイミング・アナライザは、計算式 6～9 を使用してクロック・セットアップ・スラックをレポートします（これらは計算式 2～5 に相当します）。

- (6) $\text{Clock Setup Slack} = \text{Largest Register-to-Register Requirement} - \text{Longest Register-to-Register Delay}$
- (7) $\text{Largest Register-to-Register Requirement} = \text{Setup Relationship Between Source \& Destination} + \text{Largest Clock Skew} - t_{\text{CO}} \text{ of Source Register} - t_{\text{SU}} \text{ of Destination Register}$
- (8) $\text{Setup Relationship Between Source \& Destination Register} = \text{Latch Edge} - \text{Launch Edge} - \text{Setup Uncertainty}$
- (9) $\text{Largest Clock Skew} = \text{Shortest Clock Path to Destination Register} - \text{Longest Clock Path to Source Register}$

両方の等式セットを使用して、パスのスラック値を算出することができます。

クロック・ホールド・チェック

クラシック・タイミング・アナライザは、ホールド違反を防止するために、クロック・タイミング、タイミング要件、およびタイミング例外を計算して、各デステイネーション・レジスタでクロック・ホールド・チェックを実行します。クロック・ホールド・チェックによって、ソース・レジスタから送り出されるデータがセットアップのラッチ・エッジよりも前のアクティブ・クロック・エッジでキャプチャされず、またデステイネーション・レジスタが次のアクティブな送り出しエッジから送り出されるデータをキャプチャしなくなります。クラシック・タイミング・アナライザは、クロック・ホールド・チェックを実行するために、データ到達時間には最短パス、クロック到達時間には最長パスを使用して、デステイネーション・レジスタにおけるクロック到達時間とデータ

到達時間を決定します。クラシック・タイミング・アナライザは、**計算式 10** に示すように、差がデスティネーション・レジスタのホールド時間 (t_H) よりも大きいと等しいことをチェックします。

$$(10) \quad \text{Data Arrival Time} - \text{Clock Arrival Time} \geq t_H$$

クラシック・タイミング・アナライザは、**計算式 11 ~ 14** を使用してクロック・ホールド・スラックを決定します。

$$(11) \quad \text{Clock Hold Slack} = \text{Data Arrival Time} - \text{Data Required Time}$$

$$(12) \quad \text{Data Required Time} = \text{Clock Arrival Time} + t_H + \text{Hold Uncertainty}$$

$$(13) \quad \text{Clock Arrival Time} = \text{Latch Edge} + \text{Longest Clock Path to Destination Register}$$

$$(14) \quad \text{Data Arrival Time} = \text{Launch Edge} + \text{Shortest Clock Path to Source Register} + t_{CO} + \text{Shortest Data Delay}$$

クラシック・タイミング・アナライザは、**計算式 15 ~ 18** を使用してクロック・ホールド・スラックをレポートします。

$$(15) \quad \text{Clock Hold Slack} = \text{Shortest Register-to-Register Delay} - \text{Smallest Register-to-Register Requirement}$$

$$(16) \quad \text{Smallest Register-to-Register Requirement} = \text{Hold Relationship Between Source \& Destination} + \text{Smallest Clock Skew} - t_{CO} \text{ of Source Register} + t_H \text{ of Destination Register}$$

$$(17) \quad \text{Hold Relationship Between Source and Destination Register} = \text{Latch} - \text{Launch} + \text{Hold Uncertainty}$$

$$(18) \quad \text{Smallest Clock Skew} = \text{Longest Clock Path from Clock to Destination Register} - \text{Shortest Clock Path from Clock to Source Register}$$

これらの等式を使用してパスのスラック値を算出することができます。

マルチサイクル・パス

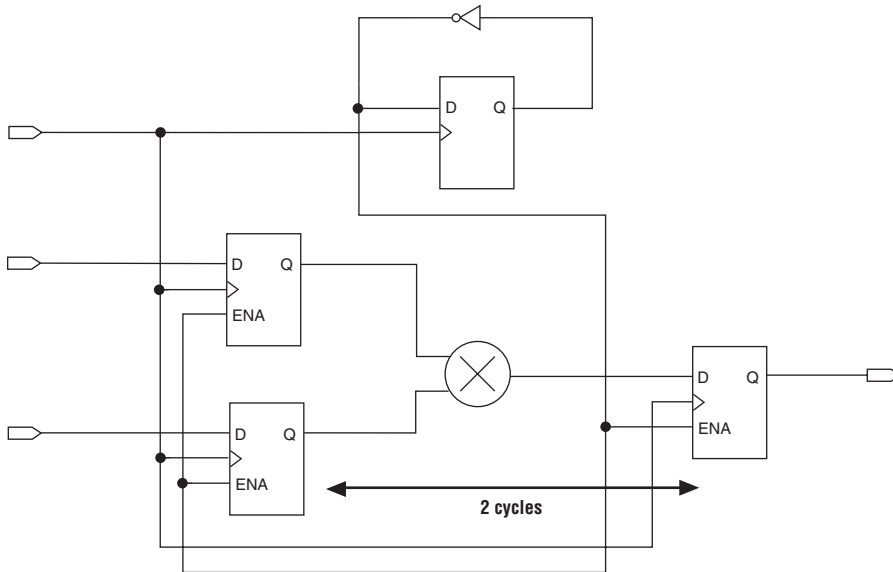
マルチサイクル・パスとは、デスティネーション・レジスタでデータをラッチするために 2 つ以上のクロック・サイクルを必要とするデータ・パスのことです。例えば、1 個のレジスタで、2 番目または 3 番目の立ち上がりエッジごとにデータをキャプチャすることが必要な場合があります。

す。図 8-4 に、乗算器の入力レジスタと、デスティネーションが1つおきのクロック・エッジでデータをラッチする出力レジスタとの間のマルチサイクル・パスの例を示します。



マルチサイクル例外について詳しくは、8-14 ページの「マルチサイクル」を参照してください。

図 8-4. マルチサイクル・パスの例を示す図



クロック設定

個別クロック設定およびデフォルト・クロック設定を使用して、デザイン内のクロックを定義することができます。これらのクロック設定は、デザイン内で定義済みの他のクロック設定に基づくことができます。



Quartus II フィッタが目的の性能要件を達成し、またクラシック・タイミング・アナライザが徹底的なスタティック・タイミング解析を実行するように、デザインをコンパイルする前にすべてのタイミング制約を指定しなければなりません。

個別クロック設定

個別クロック設定では、性能要件、オフセット、デューティ・サイクルなどのクロック・プロパティと、デザイン内の個々のクロック信号のその他のプロパティを指定することができます。

`create_base_clock` Tcl コマンドを使用して、個別クロック設定を定義することができます。以下の例では、`sys_clk` という名前の個別クロック設定が 100 MHz (10 ns) の要求値で定義され、クロック・ノード `clk` に割り当てられます。


```
create_base_clock -fmax 100MHz -target clk sys_clk
```

デフォルト・クロック設定

プロジェクト全体でのクロック要求値を割り当てて、デザイン内で検出された個別クロック設定を持たないすべてのクロックに制約をかけることができます。

`set_global_assignment -name FMAX_REQUIREMENT` Tcl コマンドは、グローバルなデフォルト要求値のアサインメントを指定します。以下の例では、100 MHz のデフォルト・クロック要求値が規定されます。

```
set_global_assignment -name FMAX_REQUIREMENT "100.0 MHz"
```

 最良の配置配線結果を得るために、デザイン内のすべてのクロックに個別クロック設定を適用します。デフォルトの F_{MAX} を採用するすべてのクロックは、デフォルトでは関係付けられていません。

クロックの種類

この項では、タイミング・アナライザが認識するクロックの種類について説明します。

- ベース・クロック
- 派生クロック
- 未定義クロック
- PLL クロック

ベース・クロック

ベース・クロックはデザイン内の他のクロックから独立しています。例えば、一般的なベース・クロックとして、デバイス・ピンから直接ドライブされるクロック信号があります。ベース・クロックは、個別クロック設定によって定義されるか、デフォルトのクロック設定を使用して自動的に検出されます。

`create_base_clock` Tcl コマンドを使用して、ベース・クロックの設定を定義し、それをクロック・ノードに割り当てることができます。以下の Tcl コマンドは、5 ns (200 MHz) の要求値を持つ `sys_clk` というクロック設定を作成し、それをクロック・ノード `main_clk` に適用するものです。

```
create_base_clock -fmax 5ns -target main_clk sys_clk
```

派生クロック

派生クロックは、先に定義されたベース・クロックに基づいています。派生クロックに対して、位相シフト、オフセット、逓倍および分周係数、およびデューティ・サイクルをベース・クロックに関連し指定できます。

`create_relative_clock Td` コマンドを使用して、派生クロックの設定を定義し、割り当てることができます。以下の例では、クロック・ノード `clkx2` に適用されるベース・クロック `system_clock` の 2 倍の速度を持つ `system_clockx2` という名前の派生クロック設定が作成されます。

```
create_relative_clock -base_clock system_clock -duty_cycle 50 -multiply 2 -target clkx2 \
system_clockx2
```

未定義クロック

クラシック・タイミング・アナライザは、デザイン内の未宣言クロックを検出し、以下のような警告を表示します。

```
Warning: Found pins functioning as undefined clocks and/or memory enables
Info: Assuming node "clk_src" is an undefined clock
Info: Assuming node "clk_dst" is an undefined clock
```

クラシック・タイミング・アナライザは、未定義クロックの実際のデータ遅延をレポートしますが、未定義クロックに対するクロック要求値は存在しないので、未定義クロックによってドライブされるレジスタ間パスのスラックはレポートしません。

PLL クロック

PLL (Phase-Locked Loop) は、アルテラ・デバイス内のクロック合成に使用されます。このデバイス機能は、Quartus II ソフトウェアに同梱されている `altpll` メガファンクションを使用して、コンフィギュレーションされデザインに使用されます。MegaWizard® Plug-In Manager を使用して、入力クロック周波数、逓倍係数、分周係数、および `altpll` メガファンクションのその他のパラメータをカスタマイズできます。




デザインでの PLL 機能の使用について詳しくは、「`altpll` Megafunction User Guide」またはターゲット・デバイス・ファミリのハンドブックを参照してください。

クラシック・タイミング・アナライザは、PLL に対して、PLL のパラメータ内容に基づき派生クロック設定を自動的に作成し、また入力クロック・ピンのベース・クロック設定も自動的に作成します。例えば、PLL への入力クロック周波数が 100 MHz で逓倍と分周の比が 5:2 の場

合、PLL クロックのクロック周期は 4.0 ns であり、これはクラシック・タイミング・アナライザで自動的に計算されます。

Stratix® および Cyclone™ デバイス・ファミリでは、PLL の入力クロック・ピンにクロック設定を適用することによって、PLL 入力クロック周波数を無効にすることができます。例えば、PLL の入力クロック周期が 5:2 の通倍対分周比で 10 ns (100 MHz) に設定されているが PLL の入力クロック・ピンに 20 ns (50 MHz) のクロック設定が適用された場合、セットアップ関係は 4.0 ns (250 MHz) ではなく 8.0 ns (125 MHz) になります。クラシック・タイミング・アナライザは、以下のようなメッセージを発行します。

```
Warning: ClockLock PLL "mypll_test:inst|altpll:altpll_component|_clk1" input frequency requirement of 200.0 MHz overrides default required fmax of 100.0 MHz -- Slack information will be reported
```

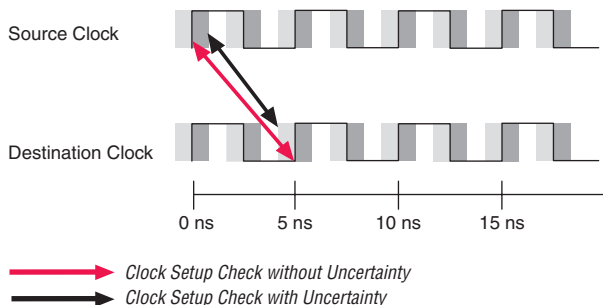
 クラシック・タイミング・アナライザでクロック設定を適用しても、PLL の出力クロック周波数を無効にすることはできません。

クロック不確実性

クロック・セットアップ不確実性およびクロック・ホールド不確実性アサインメントを使用して、ジッタまたはスキューをモデル化したり、あるいはクロック信号に関連付けられたガード・バンドを追加することができます。

クロック信号にクロック不確実性のアサインメントが存在する場合、タイミング・アナライザは最も堅実なセットアップ・チェックとホールド・チェックを実行します。クロック・セットアップ・チェックのために、データ所要時間からセットアップ不確実性が減算されます。図 8-5 に、クロック・セットアップ不確実性が適用されたクロック・ソースの例を示します。

図 8-5. クロック・セットアップ不確実性

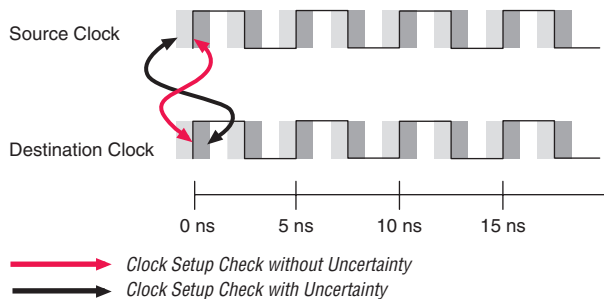


Tcl コマンド `set_clock_uncertainty` を使用して、クロック不確実性の制約を作成できます。スイッチ-`setup` と併せて `set_clock_uncertainty` 制約を使用し、クロック・セットアップ不確実性の制約を指定します。以下の例では、クロック信号 `clk` に適用される、2 ns の値を持つクロック・セットアップ不確実性の制約が作成されます。

```
set_clock_uncertainty -to clk -setup 2ns
```

クロック・ホールド・チェックのために、ホールド不確実性がデータ所要時間に加算されます。図 8-6 に、クロック・セットアップ不確実性とクロック・ホールド不確実性が適用されたクロック・セットアップ・チェックの例を示します。

図 8-6. クロック・ホールド不確実性



`set_clock_uncertainty` Tcl コマンドを `-hold` オプションと併せて使用して、クロック・ホールド不確実性の制約を指定できます。以下の例では、クロック信号 `clk` に対し、2 ns の値を持つクロック・ホールド不確実性の制約が作成されます。

```
set_clock_uncertainty -to clk -hold 2ns
```

2 つのクロック・ソース間にクロック不確実性の制約を適用することもできます。以下の例では、`clk1` がソース・クロック、`clk2` がデステーション・クロックの場合における、クロック・セットアップ・チェックに対するクロック・セットアップ不確実性の制約が作成されます。

```
set_clock_uncertainty -from clk1 -to clk2 -setup 2ns
```

クロック・レイテンシ

クロック・レイテンシの制約を使用して、クロック・ソースからの遅延をモデル化できます。例えば、クロック・レイテンシを使用して、オシレータなどの理想的なクロック・ソースからデバイスのクロック・ピンまたはクロック・ポートまでの外部遅延をモデル化することができます。

アーリ・クロック・レイテンシ制約によって、クロック・ソースの最も短いまたは最も早い遅延を指定できます。逆に、レイト・クロック・レイテンシ制約によって、クロック・ソースの最も長い、または最も遅い遅延を指定できます。

クラシック・タイミング・アナライザは、セットアップ解析時に、ソース・クロック・パスまたはデスティネーション・クロック・パスのクロック・スキューを決定する際に、ソース・クロック・パスの遅延にはレイト・クロック・レイテンシ制約の値を、デスティネーション・クロック・パスの遅延にはアーリ・クロック・レイテンシ制約の値を加算します。クラシック・タイミング・アナライザは、クロック・ホールド解析時に、ソース・クロック・パスまたはデスティネーション・クロック・パスのクロック・スキューを決定する際に、ソース・クロック・パスの遅延にはアーリ・クロック・レイテンシ制約の値を、デスティネーション・クロック・パスの遅延にはレイト・クロック・レイテンシ制約の値を加算します。

アーリ・クロック・レイテンシおよびレイト・クロック・レイテンシの制約によって、クロック設定で定義されたラッチ・エッジと送り出しエッジが変化しないため、ソース・クロックとデスティネーション・クロックの間のセットアップ関係またはホールド関係が変化することはありません。クロック・レイテンシのアサインメントでは、クロック・ネットワークにのみ遅延が加算されるため、クロック・スキューにしか影響ありません。

図 8-7 に、アーリ・クロック・レイテンシおよびレイト・クロック・レイテンシの制約使用時のセットアップ・チェックのクロック・スキューの計算に使用されるクロック・エッジを示します。

図 8-7. クロック・セットアップ・チェックのクロック・スキュー

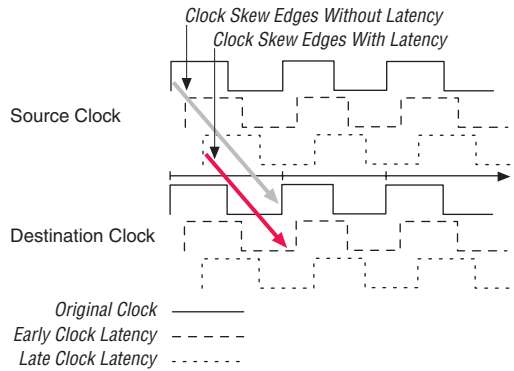
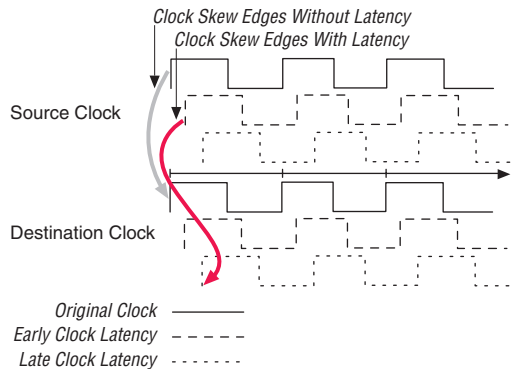


図 8-8 に、アーリ・クロック・レイテンシおよびレイト・クロック・レイテンシの制約使用時のホールド・チェックのクロック・スキューの計算に使用されるクロック・エッジを示します。


図 8-8. クロック・ホールド・チェックのクロック・スキュー



👉 クラシック・タイミング・アナライザは、ソース・レジスタと destinations レジスタでのクロック信号が同じ場合にはクロック・レイテンシを無視します。

set_clock_latency Tcl コマンドを-early スイッチまたは-late スイッチと共に使用して、アーリ・クロック・レイテンシ制約またはレイト・クロック・レイテンシ制約をそれぞれ指定できます。以下の例では、clk2 におけるクロック信号が早い場合は 1.8 ns、遅い場合は 2.0 ns で到達するように指定します。

```
set_clock_latency -early -to clk2 1.8ns
set_clock_latency -late -to clk2 2ns
```

 遅延が 1 つだけ指定されている場合、アーリ・クロック・レイテンシのデフォルト値はレイト・クロック・レイテンシ遅延と同じ、レイト・クロック・レイテンシのデフォルト値はアーリ・クロック・レイテンシ遅延と同じになります。

クラシック・タイミング・アナライザでクロック・レイテンシを解析するには、イネーブル・クロック・レイテンシ・オプションを ON に設定しなければなりません。このオプションが ON に設定されていると、クラシック・タイミング・アナライザは、実行する解析に応じて、クロック・レイテンシをソース・クロック・パスまたはデスティネーション・クロック・パスのクロック・スキュー計算の一部としてレポートします。イネーブル・クロック・レイテンシ・オプションを ON に設定するには、以下の Tcl コマンドを使用できます。


```
set_global_assignment -name ENABLE_CLOCK_LATENCY ON
```

イネーブル・クロック・レイテンシ・オプションがイネーブルされている場合、クラシック・タイミング・アナライザは、PLL 補償遅延などのオフセットを自動的に計算する代わりに、派生クロックのレイテンシを自動的に計算します。これらのクロック・パス遅延は、オフセットの場合のセットアップ関係またはホールド関係の一部ではなく、クロック・スキューとして計上されます。

タイミング例外

タイミング例外により、クラシック・タイミング・アナライザのデフォルト動作を変更できます。この項では、以下のタイミング例外について説明します。

- マルチサイクル
- セットアップ関係およびホールド関係
- 最大遅延および最小遅延
- フォルス・パス

 この章に示すタイミング例外がすべて HardCopy II デバイスに適用できるわけではありません。HardCopy II デバイス・ファミリを設計する場合は、「HardCopy II ハンドブック」の「Timing Constraint for HardCopy II」の章を参照してください。

マルチサイクル

デフォルトでは、クラシック・タイミング・アナライザはデザイン内の有効なレジスタ間パスのすべてに対して、シングル・サイクルの解析を実行します。マルチサイクル・アサインメントによって、ソース・レジ

スタがデータを送り出す前またはデスティネーション・レジスタがデータをラッチする前のクロック周期数が指定されます。マルチサイクル・アサインメントでラッチ・エッジまたは送り出しエッジが調整され、ソース・レジスタとデスティネーション・レジスタのペア間に要求されるクロック・セットアップ・チェックまたはクロック・ホールド・チェックが緩和されます。セットアップおよびホールドに対してマルチサイクルを個別に指定でき、マルチサイクルをソース・クロックまたはデスティネーション・クロックを基準にすることができます。マルチサイクル例外は、タイム・グループ、クロック・ノード、または共通のクロック・イネーブルに適用します。

デスティネーション・マルチサイクル・セットアップ例外

マルチサイクル例外と呼ばれるデスティネーション・マルチサイクル・セットアップで、レジスタで値がラッチされる前に必要な最小クロック・サイクル数が指定されます。マルチサイクル例外では、要求されるセットアップ関係を緩和することによってラッチ・エッジが変更されます。図 8-9 に、クロック・セットアップ・チェック用のラッチ・エッジ・ラベルが付けられた、デザイン内に存在するマルチサイクル・パスと関連のクロックのタイミング図を示します。


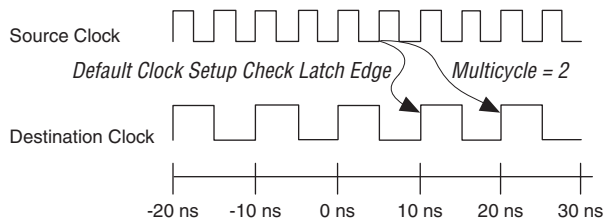
 デフォルトでは、マルチサイクル例外の値は 1 です。

図 8-9. マルチサイクル・セットアップ



マルチサイクル例外は、任意の 2 個のレジスタ間、または任意の 2 つのクロック・ドメイン間に適用できます。Tcl コマンド `set_multicycle_assignment` と、スイッチ `-setup` および `-end` を使用します。例えば、ソース・クロック `clk_src` で駆動されるすべてのレジスタの間と、デスティネーション・クロック `clk_dst` で駆動されるすべてのレジスタの間に 2 のマルチサイクル例外を適用する場合は、以下の Tcl コマンドを入力します。

```
set_multicycle_assignment -setup -end -from clk_src -to clk_dst 2
```

ソース・レジスタ reg1 とデスティネーション・レジスタ reg2 の間に 2 のマルチサイクル例外を適用するには、以下の Tcl コマンドを入力します。

```
set_multicycle_assignment -setup -end -from reg1 -to reg2 2
```

デスティネーション・マルチサイクル・ホールド例外

マルチサイクル・ホールド例外と呼ばれるデスティネーション・マルチサイクル・ホールドは、デスティネーション・クロックに基づくレジスタ間パス用のクロック・ホールド・チェックに使用されるラッチ・エッジを変更します。マルチサイクル・ホールド例外は、要求されるホールド関係を緩和することによってラッチ・エッジを変更します。図 8-10 に、クロック・セットアップ・チェック用ラッチ・エッジのラベルを付けたタイミング図を示します。


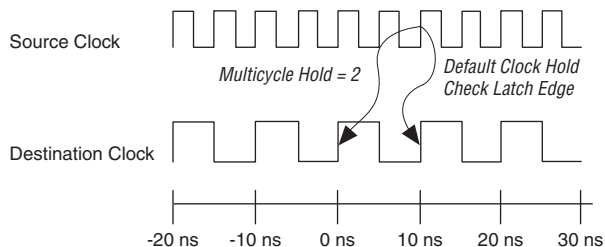
 マルチサイクル・ホールド値が指定されていない場合、マルチサイクル・ホールド値はデフォルトでマルチサイクル例外の値になります。

図 8-10. マルチサイクル・ホールド



Tcl コマンド `set_multicycle_assignment` と、スイッチ `-hold` および `-end` でマルチサイクル・ホールド例外を作成できます。以下の例では、レジスタ reg1 からレジスタ reg2 までのマルチサイクル・ホールド例外を 3 に指定します。

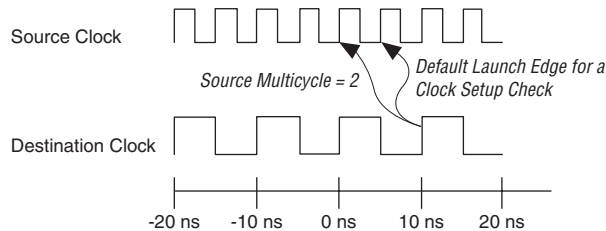
```
set_multicycle_assignment -hold -end -from reg1 -to reg2 3
```

ソース・マルチサイクル・セットアップ例外

ソース・マルチサイクル・セットアップ例外と呼ばれるソース・マルチサイクル・セットアップは、デスティネーション・クロックのラッチ・エッジではなく、ソース・クロックの送り出しエッジの調整（例えば、マルチサイクル・セットアップ）によって必要な遅延を延長するのに使

用されます。ソース・マルチサイクル例外は、ソース・レジスタとデスティネーション・レジスタに異なる周波数の関連クロックが供給される場合に役立ちます。図 8-11 に、送り出しエッジにクロック・セットアップ・チェックのためのラベルが付けられたソース・マルチサイクル例外の一例を示します。

図 8-11. ソース・マルチサイクル



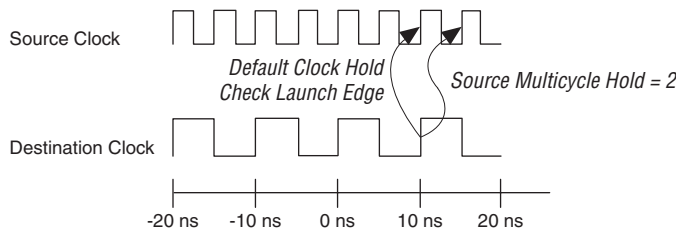
Tcl コマンド `set_multicycle_assignment` とスイッチ `-setup` および `-start` を用いてソース・マルチサイクル・セットアップ例外を作成できます。以下の例では、レジスタ `reg1` からレジスタ `reg2` までのソース・マルチサイクル例外を 3 に指定します。

```
set_multicycle_assignment -setup -start -from reg1 -to reg2 3
```

ソース・マルチサイクル・ホールド例外

ソース・マルチサイクル・ホールド例外は、ソース・クロックに基づくレジスタ間パスのクロック・ホールド・チェックに使用されるラッチ・エッジを変更します。ソース・マルチサイクル・ホールド例外は、ソース・クロック・サイクルを追加することによって要求されるホールド遅延を延長します。図 8-12 に、送り出しエッジにクロック・ホールド・チェック用のラベルを付けたソース・マルチサイクル・ホールドの例を示します。

図 8-12. ソース・マルチサイクル・ホールド



Tcl コマンド `set_multicycle_assignment` とスイッチ `-setup` および `-start` を用いてソース・マルチサイクル・ホールド例外を作成できます。以下の例では、レジスタ `reg1` からレジスタ `reg2` までのソース・マルチサイクル・ホールド例外を 3 に指定します。

```
set_multicycle_assignment -hold -start -from reg1 -to reg2 3
```

デフォルトのホールド・マルチサイクル

クラシック・タイミング・アナライザは、マルチサイクル例外が対応するホールド・マルチサイクルなしで入力されたときに、ホールド・マルチサイクル値をマルチサイクル値と等しくなるように設定します。この動作は、`DEFAULT_HOLD_MULTICYCLE` アサインメントで変更することができます。アサインメントの値は、"1" または " マルチサイクルと同じ値 " のいずれでもかまいません。


アサインメントの構文は以下のとおりです。

```
set_global_assignment -name DEFAULT_HOLD_MULTICYCLE "<value>"
```

クロック・イネーブル・マルチサイクル

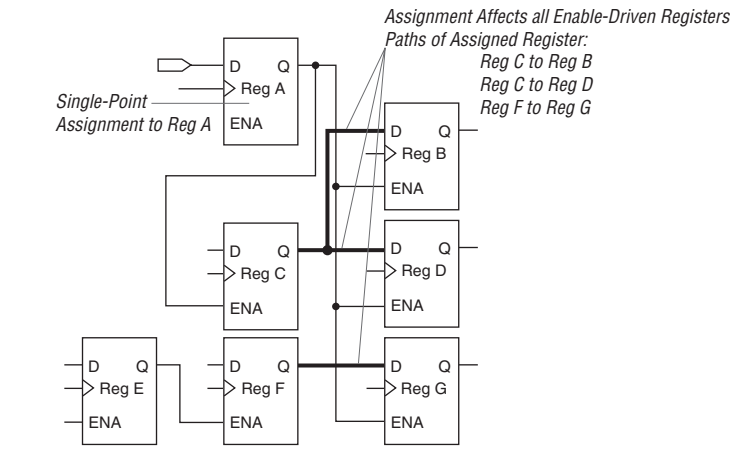
すべてのイネーブル・ドライブ・レジスタについて、クロック・イネーブル・マルチサイクル、クロック・イネーブル・マルチサイクル・ホールド、クロック・イネーブル・ソース・マルチサイクル、またはクロック・イネーブル・マルチサイクル・ソース・ホールドでセットアップ関係またはホールド関係を変更できます。

クロック・イネーブル・マルチサイクルは、指定されたクロック・イネーブルによってドライブされるすべてのレジスタに対してクロック・セットアップ・チェックが実行されるときにラッチ・エッジを変更し、クロック・イネーブル・マルチサイクル・ホールドは指定されたクロック・イネーブルによってドライブされるすべてのレジスタに対してクロック・ホールド・チェックが実行されるときにラッチ・エッジを変更します。クロック・イネーブル・ソース・マルチサイクルは、イネーブルでドライブされるすべてのレジスタに対してクロック・セットアップ・チェックが実行されるときに送り出しエッジを変更し、クロック・イネーブル・ソース・マルチサイクル・ホールドはイネーブルでドライブされるすべてのレジスタに対してクロック・ホールド・チェックが実行されるときに送り出しエッジを変更します。

 クロック・イネーブル・ベースのマルチサイクル例外は、専用のクロック・イネーブル回路を使用するレジスタにのみ適用されます。例えば、信号の優先順位設定のためにイネーブルがロジック・セルに合成されている場合、マルチサイクルは適用されません。

クロック・イネーブル・マルチサイクル、クロック・イネーブル・マルチサイクル・ホールド、クロック・イネーブル・ソース・マルチサイクル、およびクロック・イネーブル・マルチサイクル・ソース・ホールドは、シングル・ポイント・アサインメントまたはポイント・ツー・ポイント・アサインメントのいずれでもかまいません。図 8-13 にシングル・ポイント・アサインメントの例を示します。この例では、レジスタ Reg A にシングル・ポイント・アサインメントが適用されています。これは、イネーブル・ポートがレジスタ Reg A でドライブされるレジスタ間ラッチ・エッジの変更という影響を及ぼします。イネーブルがシングル・ポイント・アサインメントによってドライブされるすべてのレジスタ間パスは、たとえ異なるクロック・ソースでドライブされるものでも影響を受けます。

図 8-13. シングル・ポイント・クロック・イネーブル・マルチサイクル



ポイント・ツー・ポイント・アサインメントは、ソース・レジスタのイネーブル・ポートがソース・ノードによってドライブされ(ノードから)、デスティネーション・レジスタのイネーブル・ポートはデスティネーション・ノードによってドライブされる(ノードへ)すべてのパスに適用されます。図 8-14 に、異なるソース・レジスタとデスティネーション・レジスタに対して行われるポイント・ツー・ポイント・アサインメントの例を示します。この例では、レジスタ Reg A がソースとして、レジスタ Reg B がアサインメント用のデスティネーションとして指定されています。割り当てられたポイント・ツー・ポイント・レジスタによってイネーブルがドライブされるレジスタ間パスにおいてのみ、ラッチ・エッジが変更されます。

図 8-14. 異なるソースおよびデスティネーションのポイント・ツー・ポイント・アサインメントのクロック・イネーブル・マルチサイクル

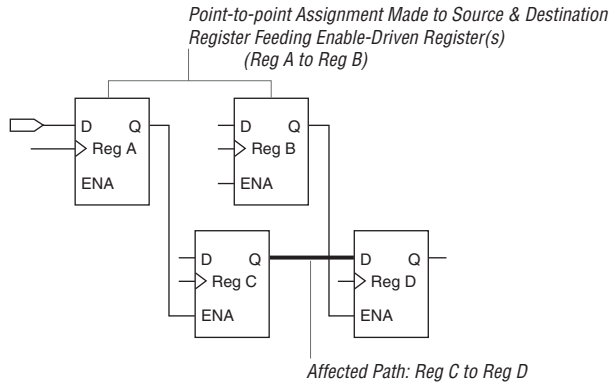
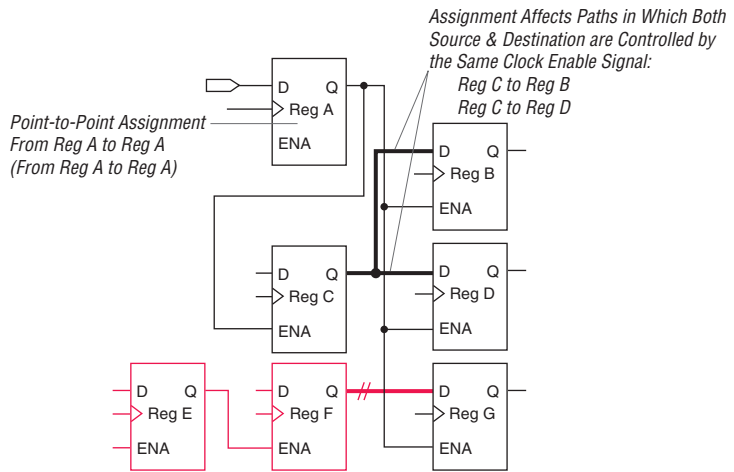


図 8-15 に、同じソース・レジスタとデスティネーション・レジスタに対して行われるポイント・ツー・ポイント・アサインメントの例を示します。この例では、レジスタ Reg A がソースおよびアサインメント用レジスタの両方に指定されています。ソース・イネーブル・ポートとデスティネーション・イネーブル・ポートの両方を持つレジスタ間パスにおいてのみ、ラッチ・エッジがポイント・ツー・ポイント・アサインメントによって変更されます。

図 8-15. 同じソースおよびデスティネーションのポイント・ツー・ポイント・アサインメントのクロック・イネーブル・マルチサイクル



```
set_instance_assignment -name CLOCK_ENABLE_MULTICYCLE
Tcl コマンドおよび set_instance_assignment -name
CLOCK_ENABLE_MULTICYCLE_HOLD Tcl コマンドを使用して、それぞ
れクロック・イネーブル・マルチサイクルまたはクロック・イネーブ
ル・マルチサイクル・ホールドのアサインメントを指定できます。以下
の例では、reg1 に 2 ns のシングル・ポイントのクロック・イネーブ
ル・マルチサイクル・アサインメントを指定します。
```

```
set_instance_assignment -name CLOCK_ENABLE_MULTICYCLE 2 -to reg1
```

以下の例では、レジスタ reg1 からレジスタ reg2 までのポイント・
ツー・ポイントのクロック・イネーブル・マルチサイクル・ホールド・
アサインメントを 2 に指定します。

```
set_instance_assignment -name CLOCK_ENABLE_MULTICYCLE_HOLD 2 -from reg1 -to reg2
```

```
set_instance_assignment -name
CLOCK_ENABLE_SOURCE_MULTICYCLE Tcl コマンドおよび
set_instance_assignment -name
CLOCK_ENABLE_MULTICYCLE_SOURCE_HOLD Tcl コマンドを使用し
て、それぞれクロック・イネーブル・マルチサイクルまたはクロック・
イネーブル・マルチサイクル・ホールドのアサインメントを指定できま
す。以下の例では、reg1 に 2 ns のシングル・ポイント・クロック・イ
ネーブル・マルチサイクル・アサインメントを指定します。
```

```
set_instance_assignment -name CLOCK_ENABLE_SOURCE_MULTICYCLE 2 -to reg1
```

以下の例では、レジスタ reg1 からレジスタ reg2 までのポイント・
ツー・ポイントのクロック・イネーブル・マルチサイクル・ホールド・
アサインメントを 2 に指定します。

```
set_instance_assignment -name CLOCK_ENABLE_SOURCE_MULTICYCLE_HOLD 2 -from reg1 -to reg2
```

セットアップ関係およびホールド関係

デフォルトでは、クラシック・タイミング・アナライザはすべてのセッ
トアップ関係とホールド関係をクロック設定に基づいて決定します。
セットアップ関係およびホールド関係例外により、デフォルトのセッ
トアップ関係またはホールド関係を無効にすることができます。例 8-1 に、
2 個のレジスタをドライブするクロック信号に 10 ns のクロック設定が
適用されたレジスタ間パスのパス詳細を示します。

例 8-1. 10 ns のクロック設定が適用されたデフォルトのセットアップ関係

```

Info: Slack time is 9.405 ns for clock "data_clk" between source register "reg9" and
destination register "reg10"
  Info: Fmax is restricted to 500.0 MHz due to tcl and tch limits
  Info: + Largest register to register requirement is 9.816 ns
    Info: + Setup relationship between source and destination is 10.000 ns
      Info: + Latch edge is 10.000 ns
        Info: - Launch edge is 0.000 ns
          Info: + Largest clock skew is 0.000 ns
            Info: - Micro clock to output delay of source is 0.094 ns
              Info: - Micro setup delay of destination is 0.090 ns
                Info: - Longest register to register delay is 0.411 ns

```

例 8-2 では、15 ns のセットアップ関係例外がレジスタ間パスに適用され、デフォルトの 10 ns のセットアップ関係が無効になっています。

例 8-2. 15 ns のセットアップ関係のアサインメント

```

Info: Slack time is 14.405 ns for clock "data_clk" between source register "reg9" and
destination register "reg10"
  Info: Fmax is restricted to 500.0 MHz due to tcl and tch limits
  Info: + Largest register to register requirement is 14.816 ns
    Info: + Setup relationship between source and destination is 15.000 ns
      Info: Setup Relationship assignment value is 15.000 ns between source "reg9" and
        destination "reg10"
        Info: + Largest clock skew is 0.000 ns
          Info: Total interconnect delay = 1.583 ns ( 51.31 % )
            Info: - Micro clock to output delay of source is 0.094 ns
              Info: - Micro setup delay of destination is 0.090 ns
                Info: - Longest register to register delay is 0.411 ns

```

Tcl コマンド `set_instance_assignment -name SETUP_RELATIONSHIP` でセットアップ関係例外を作成できます。以下の例では、レジスタ `reg1` からレジスタ `reg2` までのセットアップ関係例外を 5 に指定します。

```
set_instance_assignment -name SETUP_RELATIONSHIP 5ns -from reg1 -to reg2
```


ホールド関係例外を使用して、レジスタ間パスのデフォルトのホールド関係を無効にすることができます。

`set_instance_assignment -name HOLD_RELATIONSHIP` Tcl コマンドを使用して、ホールド関係のアサインメントを指定できます。以下の例では、レジスタ `reg1` からレジスタ `reg2` までのアップホールド関係例外を 1 に指定します。

```
set_instance_assignment -name HOLD_RELATIONSHIP 1ns -from reg1 -to reg2
```

最大遅延および最小遅延

最大遅延および最小遅延アサインメントを使用して、ピンからレジスタへのパス、レジスタ間パス、レジスタからピンへのパスに対する遅延要件を指定できます。最大遅延アサインメントによって、パスに対するすべてのセットアップ関係が無効になります。最小遅延アサインメントによって、パスに対するすべてのホールド関係が無効になります。

 クラシック・タイミング・アナライザは、デザインを最大遅延および最小遅延アサインメントと照合してチェックする際にクロック・スキューの影響を無視します。

`set_instance_assignment -name MAX_DELAY Tcd` コマンドおよび `set_instance_assignment -name -MIN_DELAY Tcd` コマンドを使用して、それぞれ最大遅延アサインメントまたは最小遅延アサインメントを指定できます。以下の例では、ソース・レジスタ `reg1` とデステイネーション・レジスタ `reg2` の間の最大遅延を `2 ns` に指定します。

```
set_instance_assignment -name MAX_DELAY 2ns -from reg1 -to reg2
```

以下の例では、入力ピン `data_in` からデステイネーション・レジスタ `dst_reg` までの最小遅延を `1 ns` に指定します。

```
set_instance_assignment -name MIN_DELAY 1ns -from data_in -to dst_reg
```

フォルス・パス

フォルス・パスとは、テスト・ロジックなどの回路の動作に関係しないパスのことです。関連のないクロック・ドメインや双方向ピンを通過するパスなど、異なるクラスのパスを切断するためのグローバル・アサインメントは複数ありますが、特定のフォルス・パスへの個々のタイミング・パスも切断できます。

タイミング・アナライザは、デザインからのフォルス・パスの削除を可能にする、以下の3つのグローバル・オプションを提供します。

- I/O ピンからのフィードバックのカット・オフ
- 書き込み中のリード信号パスのカット・オフ
- 関連のないクロック・ドメイン間のパスの切断

双方向 I/O ピンがラッチの入力と出力の両方に直接または間接的に接続されている場合、`set_global_assignment -name CUT_OFF_IO_PIN_FEEDBACK ON Tcd` コマンドを使用してフィードバック・パスを切断することができます。

```
set_global_assignment -name
```

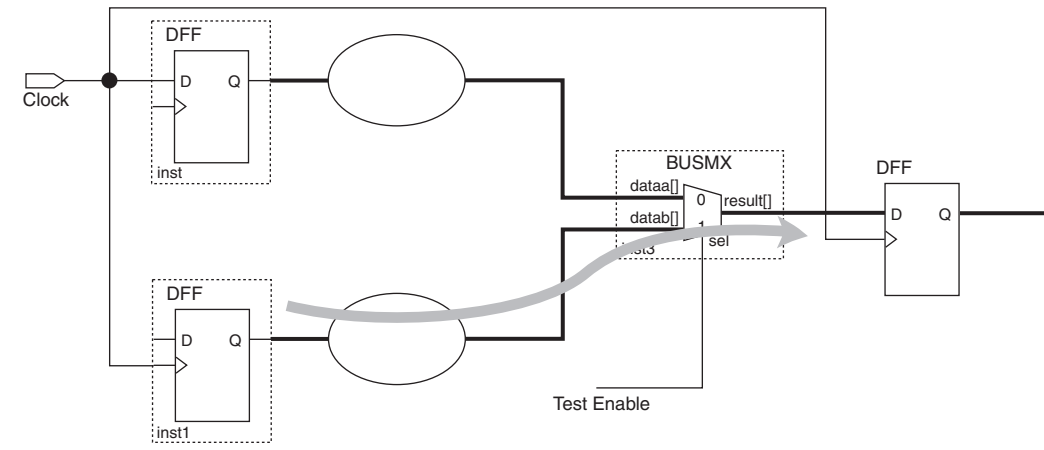
CUT_OFF_READ_DURING_WRITE_PATHS ON Tcl コマンドを使用して、ライト・イネーブル・レジスタからメモリ・エレメントを介してデスティネーション・レジスタに至るパスを切断することができます。

```
set_global_assignment -name
```

CUT_OFF_PATHS_BETWEEN_CLOCK_DOMAINS ON Tcl コマンドを使用して、ソース・クロックとデスティネーション・クロックが異なるレジスタ間パスを切断することができます。

set_timing_cut_assignment Tcl コマンドを使用して、特定のタイミング・パスを切断することができます。図 8-16 では、inst1 から乗算器を介して inst2 に至るパスはデザイン・テスト専用です。このフォルス・パスは通常動作では不要であり、スタティック・タイミング解析時に解析する必要はありません。図 8-16 に、フォルス・パスの例を示します。

図 8-16. フォルス・パス信号



ソース・レジスタ inst1 からデスティネーション・レジスタ inst2 までのタイミング・パスを切断するには、以下の Tcl コマンドを入力します。

```
set_timing_cut_assignment -from inst1 -to inst2
```

set_timing_cut_assignment Tcl コマンドは、シングル・ポイント・アサインメントとして使用することもできます。シングル・ポイント・アサインメントが使用される場合、ノードのすべてのファンアウトが切断されます。例えば、以下の Tcl コマンドは、ノード src_reg に向かうすべてのタイミング・パスを切断します。


```
set_timing_cut_assignment -to src_reg
```

I/O 解析

クラシック・タイミング・アナライザが実行する I/O 解析によって、アルテラ FPGA デザインは確実に外部デバイスにインタフェースするためのタイミング仕様をすべて満たします。この項では、I/O 解析に関連するアサインメント、およびクラシック・タイミング・アナライザで利用できる他の I/O 解析機能およびオプションについて説明します。

外部入力遅延および出力遅延アサインメント

外部入力遅延および出力遅延は、外部デバイスまたはボード・トレースから、あるいは外部デバイスまたはボード・トレースへの遅延を表します。入力遅延および出力遅延アサインメントを行って、クラシック・タイミング・アナライザに完全なシステム解析を確実に実行させることができます。入力遅延と出力遅延を提供することにより、クラシック・タイミング・アナライザは、これらのパスに対するクロック・セットアップ・チェックとクロック・ホールド・チェックを実行できます。これにより、入力パスと出力パスにマルチサイクルまたはクロック不確実性など、その他のタイミング解析も適用できます。

 個別またはグローバルの t_{SU} 、 t_{PD} 、 t_{CO} 、最小 CO 、または最小 PD アサインメントを入力遅延または出力遅延アサインメントと併用してはなりません。


入力遅延アサインメント

外部入力遅延は入力最大遅延または入力最小遅延アサインメントを用いて指定されます。入力最大遅延アサインメントを行って、指定されたクロック・ソースに対して相対的な外部レジスタから FPGA の指定された入力ピンまたは双方向ピンへの信号の最大遅延を指定します。入力最小遅延アサインメントを行って、指定されたクロック・ソースに対して相対的な外部レジスタから FPGA の指定された入力ピンまたは双方向ピンへの信号の最小遅延を指定します。

クラシック・タイミング・アナライザは、クロック・セットアップ・チェックを実行するときに、入力最大遅延アサインメント値をデータ到達時間に加算します（あるいはポイント・ツー・ポイント要求値からアサインメント値を減算します）。

クラシック・タイミング・アナライザは、クロック・ホールド・チェックを実行するときに、入力最小遅延アサインメント値をデータ到達時間に加算します（あるいはポイント・ツー・ポイント要求値からアサインメント値を減算します）。

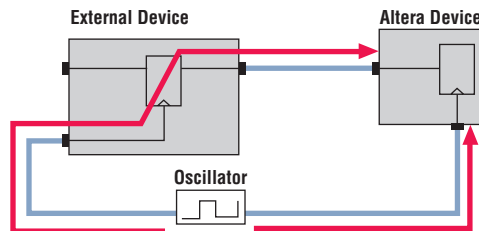
入力遅延アサインメントの値は通常、外部デバイスの t_{CO} 、アルテラ・デバイスの入力ピンまでの実際のボード遅延、およびボード上でのクロック・スキューの合計になります。

 入力最小遅延と入力最大遅延のいずれか1つしか指定されていない場合、デフォルトで入力最小遅延は入力最大遅延になり、入力最大遅延は入力最小遅延になります。

例えば、入力最大遅延と入力最小遅延は、アルテラ FPGA にドライブされる外部デバイスに関連する遅延をモデル化するのに使用できます。図 8-17 に、入力遅延パスの例を示します。図 8-17 の場合、計算式 19 に示すとおりに入力最大遅延を計算することができます。

$$(19) \quad \text{Input Maximum Delay} = \text{External Device Board Clock Path} + \text{External Device } t_{CO} + \text{External Device to Altera Device Board Delay} - \text{External Clock Path to Altera Device}$$

図 8-17. 入力遅延




Tcl コマンド `set_input_delay` を使用して、入力遅延を指定します。以下の例では、クロック・ノード `clk` から入力ピン `data_in` までの入力最大遅延アサインメントを `1.5 ns` に指定します。

```
set_input_delay -clk_ref clk -to "data_in" -max 1.5ns
```

以下の例では、クロック・ノード `clk` から入力ピン `data_in` までの入力最小遅延アサインメントを `1 ns` に指定します。

```
set_input_delay -clk_ref clk -to "data_in" -min 1ns
```

入力遅延アサインメントを使用するときは、特定のクロック・リファレンスを指定します。これによって、クラシック・タイミング・アナライザは入力パスに対する適切な解析を実行することができます。

 入力ピンに対してレポートされた t_{SU} 、 t_H 、 t_{PD} 、および最小 t_{PD} タイミング・パスには、アルテラ FPGA 内部の入力遅延アサインメントが適用されており、ピンからのデータ遅延は含まれていますが、クロック・セットアップ関係、クロック・ホールド関係、またはスラックは考慮されていません。


出力遅延アサインメント

外部出力遅延は、出力最大遅延または出力最小遅延アサインメントを用いて指定できます。出力最大遅延アサインメントを行って、指定された FPGA 出力ピンから外部レジスタへの信号の指定されたクロック・ソースを基準にした最大遅延を指定します。出力最小遅延アサインメントを行って、指定された FPGA 出力ピンから外部レジスタへの信号の指定されたクロック・ソースを基準にした最小遅延を指定します。

クラシック・タイミング・アナライザは、クロック・セットアップ・チェックを実行するときに、出力最大遅延アサインメント値をデータ所要時間から減算します（あるいはポイント・ツー・ポイント要求値からアサインメント値を減算します）。

クラシック・タイミング・アナライザは、クロック・ホールド・チェックを実行するときに、出力最小遅延アサインメント値をデータ所要時間から減算します（あるいはポイント・ツー・ポイント要求値からアサインメント値を減算します）。

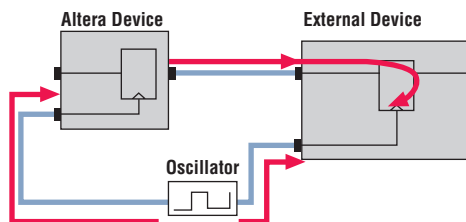
このアサインメント値は通常、外部デバイスの t_{SU} 、アルテラ・デバイスの出力ピンからの実際のボード遅延、およびボード上でのクロック・スキューの合計になります。

 遅延が 1 つだけ指定されている場合、出力最小遅延のデフォルト値は出力最大遅延と等しくなり、出力最大遅延のデフォルト値は出力最小遅延と等しくなります。

例えば、出力最大遅延と出力最小遅延を、外部デバイスにドライブされるアルテラ FPGA の出力に関連する遅延をモデル化するのに使用します。図 8-18 に、出力遅延パスの例を示します。図 8-18 の場合、計算式 20 に示すとおり出力最大遅延を計算することができます。

$$(20) \quad \text{Output Maximum Delay} = \text{Altera Device to External Device Board Delay} + \text{External Device } t_{SU} + \text{External Clock Path to Altera Device} - \text{External Device Board Clock Path}$$

図 8-18. 出力遅延




Tcl コマンド `set_output_delay` は、出力遅延アサインメントを指定します。以下の例では、クロック `clk` から出力ピン `data_out` までの出力最大遅延アサインメントを `2 ns` に指定します。

```
set_output_delay -clk_ref clk -to data_out -max 2ns
```

以下の例では、クロック `clk` から出力ピン `data_out` までの出力最小遅延アサインメントを `1 ns` に指定します。

```
set_output_delay -clk_ref clk -to data_out -min 1ns
```


出力遅延アサインメントを使用するときは、特定のクロック・リファレンスを指定します。これによって、クラシック・タイミング・アナライザは出力パス上で適切なスタティック・タイミング解析を実行することができます。

 出力ピンに対してレポートされた t_{CO} 、最小 t_{CO} 、 t_{PD} 、および最小 t_{PD} タイミング・パスには、アルテラ FPGA 内部の出力遅延アサインメントが適用されており、これらのピンへのデータ遅延のみ含まれていますが、クロック・セットアップ関係、クロック・ホールド関係、またはスラックは考慮されていません。

仮想クロック

仮想クロックを使用してアルテラ FPGA 外部のクロック信号(すなわち、アルテラ FPGA 内のいかなるものも直接ドライブしないクロック)をモデル化することができます。例えば、仮想クロックを使用して、アルテラ FPGA に供給する外部出力レジスタに供給されるクロック信号をモデル化することができます。

`create_base_clock` Tcl コマンドの `-virtual` オプションを使用して、仮想クロックのアサインメントを指定します。

 仮想クロックを使用して入力または出力遅延アサインメントを行う前に、仮想クロックの仮想クロック・リファレンス・アサインメントを仮想クロック設定に対してイネーブルしなければなりません。

例 8-3 のコードは、`200 MHz` の要求値を持つ `virt_clk` という名前の仮想クロックを作成し、その仮想クロック設定を入力遅延アサインメント用のクロック・リファレンスとして使用します。

例 8-3. virt_clk という名前の仮想クロックの作成

```
# 仮想クロック設定を作成する。
create_base_clock -fmax 200MHz -virtual virt_clk

# 仮想クロック設定に対する仮想クロック・リファレンスをイネーブルする。
set_instance_assignment -name VIRTUAL_CLOCK_REFERENCE On -to virt_clk

# 仮想クロック設定を入力遅延アサインメント用のクロック・リファレンスとして使用する。
set_input_delay -clk_ref virt_clk -to data_in -max 2ns
```

非同期パス

クラシック・タイミング・アナライザは、レジスタのクリア、プリセット、またはロード・ポートに接続する非同期信号を解析することができます。この項では、クラシック・タイミング・アナライザが非同期パスを解析する方法について説明します。


リカバリおよびリムーバル

リカバリ時間は、クリアやプリセットなどの非同期コントロール信号がアクティブ・クロック・エッジ前に安定していなければならない最小時間長です。リムーバル時間は、非同期コントロール信号がアクティブ・クロック・エッジ後に安定していなければならない最小時間長です。イネーブル・リカバリ / リムーバル解析オプションでは、レジスタの非同期クリア、プリセット、またはロード信号で終了するパスのリカバリ・チェックとリムーバル・チェックの結果がレポートされます。

以下の Tcd コマンドでリカバリ解析およびリムーバル解析をイネーブルします。

```
set_global_assignment -name ENABLE_RECOVERY_REMOVAL_ANALYSIS ON
```

このオプションがイネーブルされていると、クラシック・タイミング・アナライザはリカバリ解析とリムーバル解析の結果をレポートします。

 デフォルトでは、リカバリ解析とリムーバル解析はデイスーブルされています。このオプションは、非同期コントロール信号を含むすべてのデザインに対してイネーブルしなければなりません。

リカバリ・レポート

ENABLE_RECOVERY_REMOVAL_ANALYSIS を ON に設定すると、クラシック・タイミング・アナライザはリカバリ時間を、非アクティブになりつつある非同期コントロール信号と次のアクティブ・クロック・エッジとの間に必要な最小時間と判断し、これをデザインと比較して、結果をスラックとしてレポートします。リカバリ・レポートは、非同期入力为非アクティブになった後アクティブ・クロック・エッジが発生するまでの時間が短すぎたために、レジスタのデータが不確実になった状態を知らせます。

リカバリ・スラック時間計算はクロックのセットアップ・スラックの計算に似ており、データ到達時間とデータ所要時間に基づきます（ただし、非同期コントロール信号を除きます）。非同期コントロールが登録されている場合、クラシック・タイミング・アナライザは、[計算式 21 ~ 23](#) を使用してリカバリ・スラック時間を計算します。

- (21) $\text{Recovery Slack Time} = \text{Data Required Time} - \text{Data Arrival Time}$
- (22) $\text{Data Arrival Time} = \text{Launch Edge} + \text{Longest Clock Path to Source Register} + \text{micro } t_{CO} \text{ of Source Register} + \text{Longest Register-to-Register Delay}$
- (23) $\text{Data Required Time} = \text{Latch Edge} + \text{Shortest Clock Path to Destination Register} - \text{micro } t_{SU} \text{ of Destination Register}$

[例 8-4](#) に、タイミング・アナライザがレポートするリカバリ時間を示します。

例 8-4. 登録された非同期リセット信号のリカバリ時間のレポート

```
Info: Slack time is 8.947 ns for clock "a_clk" between source register "async_reg1" and destination register "reg_1"
Info: Requirement is of type recovery
Info: - Data arrival time is 4.028 ns
Info: + Launch edge is 0.000 ns
Info: + Longest clock path from clock "a_clk" to source register is 3.067 ns
Info: + Micro clock to output delay of source is 0.094 ns
Info: + Longest register to register delay is 0.867 ns
Info: + Data required time is 12.975 ns
Info: + Latch edge is 10.000 ns
Info: + Shortest clock path from clock "a_clk" to destination register is 3.065 ns
Info: - Micro setup delay of destination is 0.090 ns
```

非同期コントロールが登録されていない場合、クラシック・タイミング・アナライザは、[計算式 24 ~ 計算式 26](#) を使用してリカバリ・スラック時間を計算します。

- (24) $\text{Recovery Slack Time} = \text{Data Required Time} - \text{Data Arrival Time}$
- (25) $\text{Data Arrival Time} = \text{Launch Edge} + \text{Maximum Input Delay} + \text{Maximum Pin to Register Delay}$
- (26) $\text{Data Required Time} = \text{Latch Edge} + \text{Shortest Clock Path to Destination Register Delay} - \text{micro } t_{SU} \text{ of Destination Register}$

例 8-5 に、タイミング・アナライザがレポートするリカバリ時間を示します。

例 8-5. 登録されていない非同期リセット信号のリカバリ時間のレポート

```
Info: Slack time is 8.744 ns for clock "a_clk15" between source pin "a_arst2" and destination register "inst5"
```

```
    Info: Requirement is of type recovery
    Info: - Data arrival time is 4.787 ns
          Info: + Launch edge is 0.000 ns
          Info: + Max Input delay of pin is 1.500 ns
          Info: + Max pin to register delay is 3.287 ns
    Info: + Data required time is 13.531 ns
Info: + Latch edge is 10.000 ns
Info: + Shortest clock path from clock "a_clk15" to destination register
is 3.542 ns
    Info: - Micro setup delay of destination is 0.011 ns
```



非同期リセット信号がデバイス・ピンから来る場合は、クラシック・タイミング・アナライザがそのパスでリカバリ解析を実行するには、非同期リセット・ピンに対して入力最大遅延アサインメントを行わなければなりません。

リムーバル・レポート

ENABLE_RECOVERY_REMOVAL_ANALYSIS を ON に設定すると、クラシック・タイミング・アナライザはリムーバル時間を、非同期入力が入力アクティブのときに発生するアクティブ・クロック・エッジと非同期コントロール信号のデアサーションとの間の最小所要時間と判断します。クラシック・タイミング・アナライザは、次にこれをデザインと比較し、結果をスラックとしてレポートします。リムーバル・レポートは、非同期入力信号がクロック・エッジ後に非アクティブになるまでの時間が短すぎたために、レジスタのデータが不確実になった状態を知らせます。

リムーバル時間のスラック計算は、クロック・ホールド・スラックの計算に似ており、データ到達時間とデータ所要時間に基づきます（ただし、非同期コントロール信号を除きます）。非同期コントロールが登録されている場合、クラシック・タイミング・アナライザは、[計算式 27 ~ 29](#) を使用してリムーバル・スラック時間を計算します。

$$(27) \quad \text{Removal Slack Time} = \text{Data Arrival Time} - \text{Data Required Time}$$

$$(28) \quad \text{Data Arrival Time} = \text{Launch Edge} + \text{Shortest Clock Path From Source Register Delay} + \text{Micro } t_{\text{CO}} \text{ of Source Register} + \text{Shortest Register-to-Register Delay}$$

$$(29) \quad \text{Data Required Time} = \text{Latch Edge} + \text{Longest Clock Path to Destination Register Delay} + \text{micro } t_{\text{H}} \text{ of Destination Register}$$

例 8-6 に、クラシック・タイミング・アナライザがレポートするリムーバル時間を示します。

例 8-6. 登録された非同期リセット信号のリムーバル時間のレポート

```
Info: Minimum slack time is 814 ps for clock "a_clk" between source register "async_reg1"
and destination register "reg_1"
Info: Requirement is of type removal
Info: + Data arrival time is 4.028 ns
Info: + Launch edge is 0.000 ns
Info: + Shortest clock path from clock "a_clk" to source register is 3.067 ns
Info: + Micro clock to output delay of source is 0.094 ns
Info: + Shortest register to register delay is 0.867 ns
Info: - Data required time is 3.214 ns
Info: + Latch edge is 0.000 ns
Info: + Longest clock path from clock "a_clk" to destination register is 3.065 ns
Info: + Micro hold delay of destination is 0.149 ns
```

非同期コントロールが登録されていない場合、クラシック・タイミング・アナライザは、[計算式 30](#) ~ [32](#) を使用してリムーバル・スラック時間を計算します。

$$(30) \quad \text{Removal Slack Time} = \text{Data Arrival Time} - \text{Data Required Time}$$


$$(31) \quad \text{Data Arrival Time} = \text{Launch Edge} + \text{Input Minimum Delay of Pin} + \text{Minimum Pin to Register Delay}$$

$$(32) \quad \text{Data Required Time} = \text{Latch Edge} + \text{Longest Clock Path to Destination Register Delay} + \text{micro } t_{\text{H}} \text{ of Destination Register}$$

例 8-7 に、クラシック・タイミング・アナライザがレポートするリムーバル時間を示します。

例 8-7. 登録されていない非同期リセット信号のリムーバル時間のレポート

```
Info: Minimum slack time is 1.131 ns for clock "a_clk15" between source pin "a_arst2" and
destination register "inst5"
Info: Requirement is of type removal
Info: + Data arrival time is 4.787 ns
Info: + Launch edge is 0.000 ns
Info: + Min Input delay of pin is 1.500 ns
Info: + Min pin to register delay is 3.287 ns
Info: - Data required time is 3.656 ns
Info: + Latch edge is 0.000 ns
Info: + Longest clock path from clock "a_clk15" to destination register
is 3.542 ns
Info: + Micro hold delay of destination is 0.114 ns
```

 非同期リセット信号がデバイス・ピンから来る場合は、クラシック・タイミング・アナライザがこのパスでリムーバル解析を実行できるように、非同期リセット・ピンに対して入力最小遅延アサインメントを行わなければなりません。

スキュー管理

クロック・スキューは、2 個の異なるレジスタにおけるクロック信号の到達時間の差であり、2つのクロック・パスのパス長の差、あるいはゲーテッド・クロックまたはリップル・クロックの使用が原因で発生する可能性があります。クロック周期が短くなるほど、データ到達時間とクロック到達時間の間のスキューが大きくなります。クラシック・タイミング・アナライザは、データおよびクロック信号のスキューを解析し、それを制約するための2つのアサインメントを提供します。

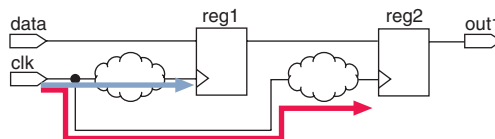
最大クロック到達スキュー


最大クロック到達スキュー・アサインメントを行って、クロック信号と各種デスティネーション・レジスタとの間の最大許容クロック到達スキューを指定します。クラシック・タイミング・アナライザは、レジスタのクロック・ポートへの最長クロック・パスとレジスタのクロック・ポートへの最短クロック・パスを比較して、最大クロック到達スキューが達成されているか否かを判断します。最大クロック到達スキューは、[計算式 33](#) を使用して計算されます。

$$(33) \quad \text{Maximum Clock Arrival Skew} = \text{Longest Clock Path} - \text{Shortest Clock Path}$$

例えば、[図 8-19](#) に示すように、クロック・ピン clk からレジスタ reg1 のクロック・ポートまでの遅延が 1.0 ns で、クロック・ピン clk からレジスタ reg2 のクロック・ポートまでの遅延が 3.0 ns の場合、クラシック・タイミング・アナライザは 2.0 ns のクロック・スキュー・スラック時間を提供します。

図 8-19. クロック到達パス



 最大クロック到達スキュー・アサインメントをクロック・ノードとレジスタ・グループに適用する必要があります。最大クロック到達スキュー・アサインメントを行うと、フィタはスキュー要件を満たすよう試みます。

`set_instance_assignment -name max_clock_arrival_skew`
 Tcl コマンドを使用して、最大クロック到達スキュー・アサインメントを指定できます。以下の例では、クロック信号 `clk` から `reg*` にマッチングするレジスタのバンクまでの最大クロック到達スキューを `1 ns` に指定します。

```
set_instance_assignment -name max_clock_arrival_skew 1ns -from clk -to reg*
```

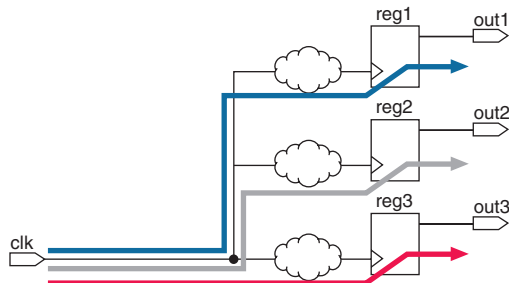
最大データ到達スキュー


最大データ到達スキュー・アサインメントを行って、各種デスティネーション・レジスタまたはピンへの最大許容データ到達スキューを指定します。クラシック・タイミング・アナライザは、最長データ到達パスと最短データ到達パスを比較して、最大データ到達スキューが達成されているか否かを判断します。最大データ到達スキューは、[計算式 34](#) を使用して計算されます。

$$(34) \quad \text{Maximum Data Arrival Skew} = \text{Longest Data Arrival Path} - \text{Shortest Data Arrival Path}$$

例えば、[図 8-20](#) に示すように、出力ピン `out1` へのデータ到達時間が `2.0 ns`、出力ピン `out2` へのデータ到達時間が `1.5 ns`、出力ピン `out3` へのデータ到達時間が `1.0 ns` の場合、クラシック・タイミング・アナライザは `1.0 ns` の最大データ到達スキュー・スラック時間を提供します。

図 8-20. データ到達パス




 最大データ到達スキュー・アサインメントを行うと、フィッタはスキュー要件を満たすよう試みます。

set_instance_assignment -name max_data_arrival_skew Tcl コマンドを使用して、最大データ到達スキュー値を指定できます。以下の例では、クロック信号 clk から出力ピン dout のバンクまでの最大データ到達スキューを 1 ns に指定します。

```
set_instance_assignment -name max_data_arrival_skew 1ns -from clk -to dout[*]
```

report_timing でのタイミング 解析レポートの 生成

クラシック・タイミング・アナライザには、テキスト・ベースのタイミング解析レポートを生成するための report_timing Tcl コマンドが用意されています。デザインの任意のパスに対する詳細および概要タイミング・レポートの生成を可能にする複数のスイッチを使用して、report_timing の出力をカスタマイズすることができます。

 report_timing Tcl コマンドは、quartus_tan 実行コマンドで使用できます。

report_timing Tcl コマンドを使用する前に、Quartus II プロジェクトを開いてタイミング・ネットリストを作成しなければなりません。例えば、以下の 2 つの Tcl コマンドでこれを行います。

```
project_open my_project
create_timing_netlist
```

report_timing Tcl コマンドは、特定のソースおよびデスティネーション・ノードをフィルタするための -from および -to スイッチを提供します。例えば、以下の report_timing Tcl コマンドは、スイッチ -clock_setup を用いて、レジスタ src_reg* と dst_reg* の間にあるすべてのクロック・セットアップ・パスをレポートします。-npaths 20 スイッチはレポートされるパス数を 20 に制限します。

```
report_timing -clock_setup -from src_reg* -to dst_reg* -npaths 20
```

スイッチ -clock_filter および -src_clock_filter も、特定のクロック・ソースに基づくフィルタリングに使用できます。例えば、以下の report_timing Tcl コマンドは、デスティネーション・レジスタが clk で駆動されるすべてのクロック・セットアップ・パスをレポートします。

```
report_timing -clock_setup -clock_filter clk
```

以下の例では、デスティネーション・レジスタが clk で駆動され、ソース・レジスタが src_clock で駆動されるクロック・セットアップ・パスをレポートします。

```
report_timing -clock_setup -clock_filter clk -src_clock_filter src_clk
```

以下、quartus_tan 実行コマンドで提供可能なスクリプト例を示します。

```
# プロジェクトを開く。
project_open my_project
# 常にネットリストを最初に作成する。
create_timing_netlist
# クロック clk のクロック・セットアップ・パスをリストする。
# レジスタ abc* からレジスタ xyz* まで
report_timing -clock_setup -clock_filter clk -from abc* -to xyz*
# トップ5つのピン間組み合わせパスをリストする。
report_timing -tpd -npaths 5
# トップ5つのピン間組み合わせパスをリストし、
# 出力を out.tao ファイルに書き込む。
report_timing -tpd -npaths 5 -file out.tao
# 最小tpdを計算し、結果を既存の out.tao ファイルに追加する。
report_timing -min_tpd -npaths 5 -file out.tao -append
# a* と b* の間の最長パス（レジスタ間のデータ・パス）を表示する。
report_timing -longest_paths -npaths 1
delete_timing_netlist
project close
```

その他の タイミング・ アナライザ機能

クラシック・タイミング・アナライザは、以下のようなスタティック・タイミング解析の効率をカスタマイズし向上させるための多くの機能を提供します。

- ワイルドカード・アサインメント
- アサインメント・グループ
- 高速コーナ解析
- 早期タイミング見積もり
- タイミング制約チェック
- ラッチ解析

ワイルドカード・アサインメント

多数のノード・アサインメントを作成する作業を簡略化するために、Quartus II ソフトウェアは * および ? のワイルドカード文字を受け入れます。これらのワイルドカード文字を使用して、デザインに対して作成する必要がある個別アサインメント数を減らします。

ワイルドカード文字 "*" は任意の文字列にマッチングします。例えば、クラシック・タイミング・アナライザは、reg* として指定されたノードにアサインメントが行われると、プリフィックスの reg とそれに続く 0、1、または複数の文字から成る文字列 (reg1、reg[2]、regbank、reg12bank など) にマッチングするすべてのデザイン・ノードを検索し、それらのノードにアサインメントを適用します。

ワイルドカード文字 "?" は任意の 1 文字にマッチングします。例えば、クラシック・タイミング・アナライザは、reg? として指定されたノード

にアサインメントが行われると、プリフィックスの `reg` とそれに続く任意の 1 文字から成る文字列 (`reg1`、`rega`、`reg4` など) にマッチングするすべてのデザイン・ノードを検索し、それらのノードにアサインメントを適用します。

アサインメント・グループ

タイム・グループとしても知られるアサインメント・グループによって、タイミング・アサインメントを割り当てることができるノードのカスタム・グループを定義できます。特定のノード、ワイルドカード、およびあるタイム・グループからのタイム・グループを排除することもできます。

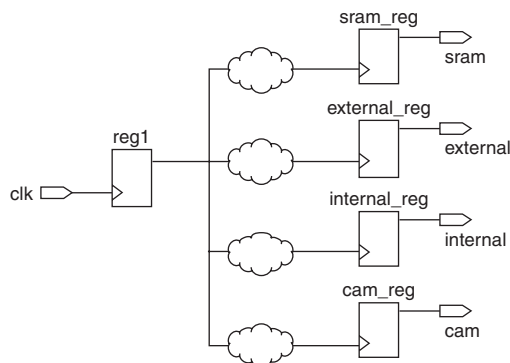
`timegroup Tcl` コマンドを使用して、アサインメント・グループを作成します。以下の例では、アサインメント・グループ `srcgrp` が作成され、`src1*` にマッチングする名前を持つノードをグループに追加します。

```
timegroup srcgrp -add_member src1*
```

例えば、[図 8-21](#) では、ソース・レジスタ `reg1` とデスティネーション・レジスタのバンク `sram_reg`、`external_reg`、`internal_reg`、および `cam_reg` との間に、切断すべきフォルス・パスがあります。アサインメント・グループを使用しない場合、必要なアサインメントは以下のとおりです。

```
set_timing_cut_assignment -from reg1 to sram_reg
set_timing_cut_assignment -from reg1 to external_reg
set_timing_cut_assignment -from reg1 to internal_reg
set_timing_cut_assignment -from reg1 to cam_reg
```


図 8-21. フォルス・パス



dst_reg_bank と呼ばれるアサインメント・グループでは、以下のアサインメントが必要です。


```
# dst_reg という名前のタイム・グループを作成する。
timegroup dst_reg_bank -add_member sram_reg
timegroup dst_reg_bank -add_member external_reg
timegroup dst_reg_bank -add_member internal_reg
timegroup dst_reg_bank -add_member cam_reg
# タイミング・パスを切断する。
set_timing_cut_assignment -from reg1 to dst_reg_bank
```

一度アサインメント・グループが定義されると、そのアサインメント・グループを再定義することなく、タイム・グループに対して適用可能なタイミング・アサインメントを行うことができます。

 個々のノードをタイム・グループに割り当て、これらのグループにタイミング・アサインメントを適用すると、クラシック・タイミング・アナライザの性能を向上させることができます。

高速コーナ解析

高速コーナ解析では、最高速度・グレード・デバイスに対して、ベスト・ケースの条件（電圧、プロセス、および温度）で生成されたタイミング・モデルを使用します。

 高速コーナおよび低速コーナ・スタティック・タイミング解析レポートのいずれも、<プロジェクト名>.tan.rpt ファイルに保存され、以前のタイミング解析レポートを上書きすることがあります。レポートのコピーを保持するには、次回のコンパイルまたはスタティック・タイミング解析の前にファイルを新しい名前で保存するか、**Combined Fast/Slow Analysis**（高速 / 低速組み合わせ解析）レポート機能を使用します。


Quartus II ソフトウェアはさらに、低速コーナ（デフォルト）解析後の最小遅延チェックもレポートします。これらの結果は、ワースト・ケースのタイミング・モデルを使用した最小遅延チェックのレポートによって生成されます。

ベスト・ケースのタイミング・モデルを用いた高速コーナ・スタティック・タイミング解析を実行するには、switch --fast_model=on を quartus_tan 実行コマンドと共に使用することができます。以下の Td コマンドによって高速タイミング・モデルがイネーブルされます。

```
quartus_tan <project_name> --fast_model=on
```

早期タイミング見積もり

Quartus II ソフトウェアのコンパイル時間の大半は、最適なデザイン結果を得るために使用される配置配線プロセスによって消費されます。Quartus II ソフトウェアは、大規模なデザインのデザイン・プロセスを高速化するために早期のタイミング見積もりを提供します。この機能は、デザイン上で完全な最適化を行わずに暫定的な配置配線を実行することによって、フル・コンパイルに必要な数分の1の時間で高速スタティック・タイミング解析を提供し、それにより、完全にフィッティングされたデザインと比較して合計コンパイル時間が最大1/5短縮されます。

 早期タイミング見積もりフィッティングは、完全に最適化されず、または正式に配線されません。タイミング遅延レポートは見積もりにすぎません。一般に、見積もり遅延は、現実的な設定を使用したときに完全フィットで得られる遅延値の10%以内に収まります。

早期タイミング見積もりには、タイミング見積もり値を生成するための設定として、「Realistic」、「Optimistic」、「Pessimistic」の3通りがあります。表 8-1 に、これらの設定の説明を示します。

設定	説明
Realistic (デフォルト設定: 標準的なフィッティングを使用して最終的なタイミングを見積もります)	フル・コンパイル結果に最も近いと考えられるタイミング見積もり値を生成します。
Optimistic (ベスト・ケースの最終的なタイミングを見積もります)	フル・コンパイルの結果と同等以上の可能性が高いタイミング見積もり値を生成します。
Pessimistic (ワースト・ケースの最終的なタイミングを見積もります)	フル・コンパイルの結果と同等以下の可能性が高いタイミング見積もり値を生成します。

早期タイミング見積もり機能を使用するには、フィッティングを実行するとき以下の Tcl コマンドを入力します。

```
quartus_fit --early_timing_estimate[=<realistic|optimistic|pessimistic>]
```

早期タイミング見積もりの完了後に、早期配置配線の遅延に基づいて完全なタイミング・レポートが生成されます。さらに、タイミング・クロージャ・フロアプランに暫定的なロジック配置を表示することも可能です。早期タイミング配置によって、初期配置を実行し、さまざまな配置トポロジーのタイミング相互作用を表示することができます。

タイミング制約チェッカ

アルテラでは、フル・コンパイルを実行する前にすべてのタイミング制約を Quartus II ソフトウェアに入力することを推奨しています。これによって、フッタが正しいタイミング要件をターゲットにし、クラシック・タイミング・アナライザがデザイン内のすべてのタイミング・パスに対して違反を正しくレポートできるようになります。すべての制約がデザイン・ノードに確実に適用されるように、タイミング制約チェック機能によってデザイン内で制約されていないすべてのパスがレポートされます。例 8-8 に、フル・コンパイル後に生成されるタイミング制約チェックの要約を示します。

例 8-8. タイミング制約チェックの要約

```

+-----+
; Timing Constraint Check Summary ;
+-----+
; Timing Constraint Check Status ; Analyzed - Tue Feb 28 11:42:31 2006 ;
; Quartus II Version ; 6.0 Internal Build 143 02/20/2006 SJ Full Version ;
; Revision Name ; test ;
; Top-level Entity Name ; Block1 ;
; Unconstrained Clocks ; 0 ;
; Unconstrained Paths (Setup) ; 22 ;
; Unconstrained Reg-to-Reg Paths (Setup) ; 0 ;
; Unconstrained I/O Paths (Setup) ; 22 ;
; Unconstrained Paths (Hold) ; 12 ;
; Unconstrained Reg-to-Reg Paths (Hold) ; 0 ;
; Unconstrained I/O Paths (Hold) ; 12 ;
+-----+

```

タイミング制約チェックを実行するには、switch `--check_constraints` を `quartus_tan` 実行コマンドと共に使用します。以下の Tcl コマンドは、デザイン・システムのセットアップおよびホールドの両方でタイミング制約チェックを実行します。

```
quartus_tan block1 --check_constraints=both
```

ラッチ解析

ラッチは、自身にフィードバックするルック・アップ・テーブル (LUT) として Quartus II ソフトウェアに実装されています。クラシック・タイミング・アナライザは、これらのラッチを組み合わせたエレメントではなく同期エレメントとして解析することができます。クロック・イネーブルは反転クロックとして解析されます。クラシック・タイミング・アナライザは、これらのラッチに対するセットアップおよびホールド解析の結果をレポートします。

以下の Tcl コマンドで、**Analyze Latches As Synchronous Elements** オプションをオンにすることができます。

```
set_global_assignment -name ANALYZE_LATCHES_AS_SYNCHRONOUS_ELEMENTS ON
```

Quartus II GUI を使用した タイミング解析

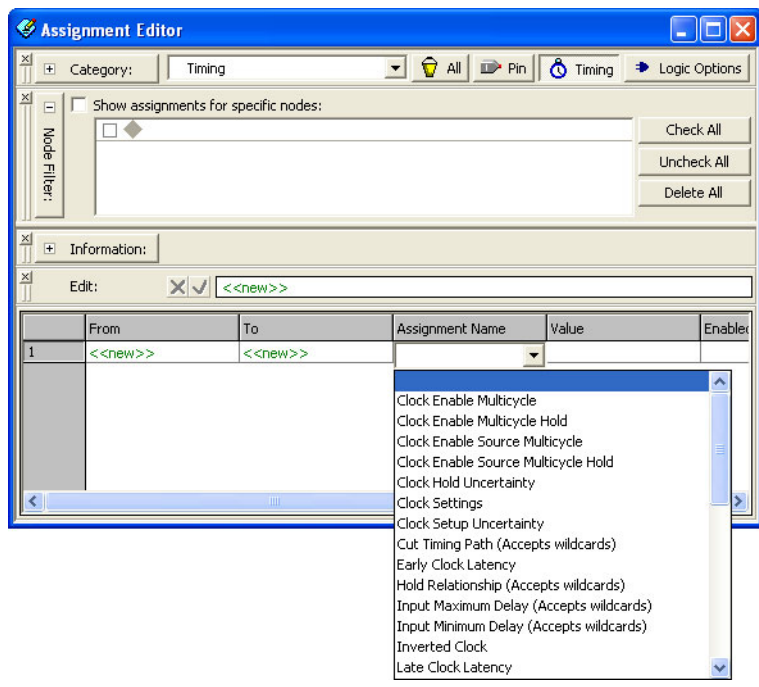
Quartus II ソフトウェアは、クラシック・タイミング・アナライザで使用できる広範なスクリプティング・サポートに加え、Assignment Editor およびその他のユーザ・インタフェース・ツールを提供しており、クラシック・タイミング・アナライザ機能やアサインメントを利用することができます。

Assignment Editor

Assignment Editor は、アサインメントの追加、変更、および削除に使用するスプレッドシート式のインタフェースです。

Assignment Editor でタイミング・アサインメントを行うには、カテゴリ・リストから **Timing** を選択して Assignment Name カラムにタイミング・アサインメントのみを表示します。**Assignment Name** フィールドの <<new>> をダブル・クリックすると、**Assignment Name** リストが表示されます。図 8-22 に、Assignment Name リストにタイミング・アサインメントのタイプが表示された Assignment Editor を示します。

図 8-22. Assignment Editor

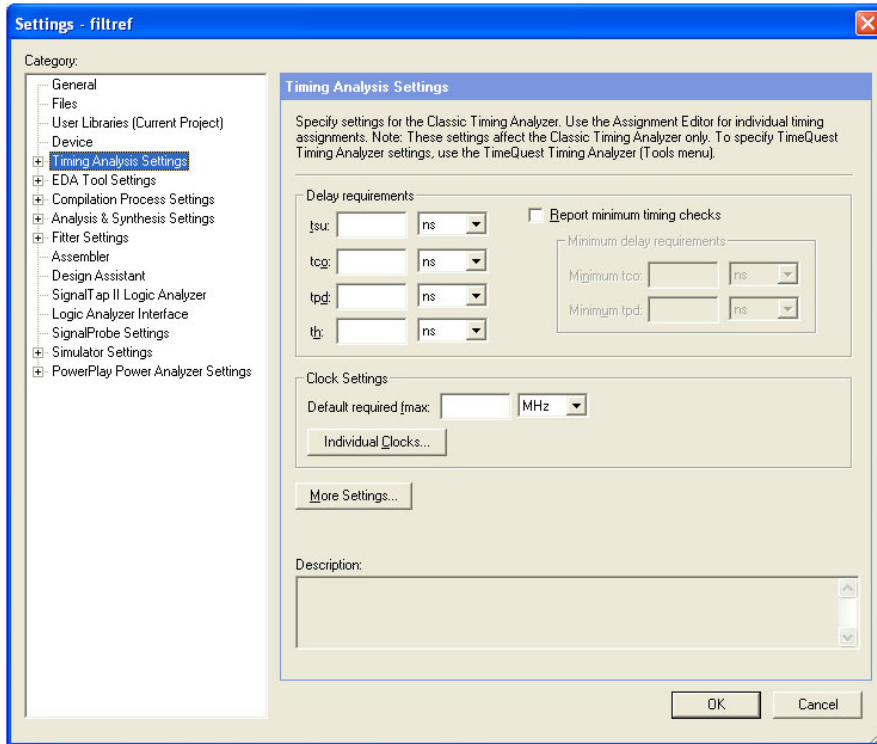


Assignment Editor について詳しくは、「Quartus II ハンドブック Volume 2」の「Assignment Editor」の章を参照してください。

タイミング設定

Settings ダイアログ・ボックスの **Timing Analysis Settings** ページで、遅延要件とクロック設定を指定することができます。このページにアクセスするには、Assignments メニューの **Timing Analysis Settings** をクリックします。**Timing Analysis Settings** ページが表示されます (図 8-23)。

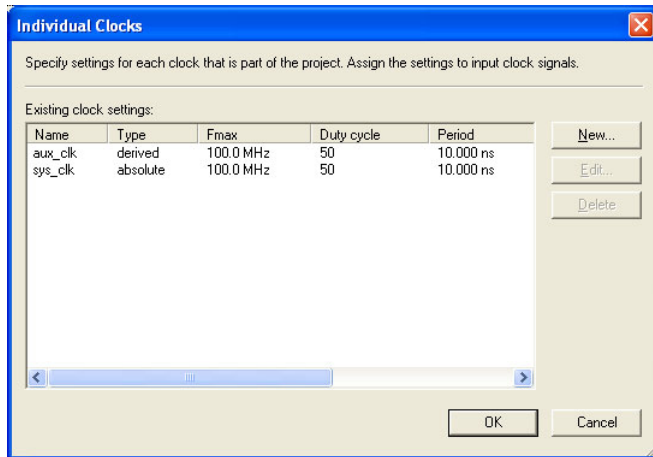
図 8-23. Timing Analysis Settings ダイアログ・ボックス



Clock Settings ダイアログ・ボックス

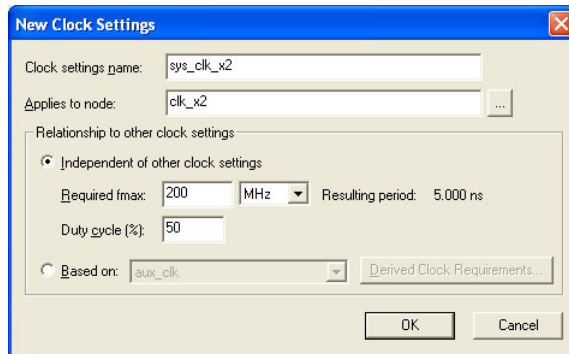
Clock Settings ダイアログ・ボックスを使用して、ベース・クロック設定または派生クロック設定を作成または変更できます。Assignments メニューの **Timing Analysis Settings** をクリックします。**Timing Analysis Settings** ページが表示されます。**Clock Settings** の下の **Individual Clocks** をクリックします。**Individual Clock** ダイアログ・ボックスが表示されます (図 8-24)。

図 8-24. Individual Clocks ダイアログ・ボックス



Individual Clocks ダイアログ・ボックスの **New** ボタンをクリックして、**New Clock Settings** ダイアログ・ボックスにアクセスし、ベース・クロック設定または派生クロック設定を作成します (図 8-25)。

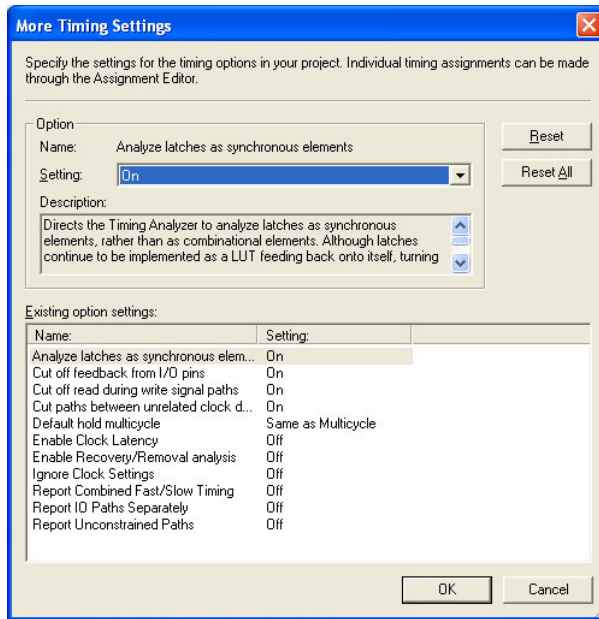
図 8-25. New Settings ダイアログ・ボックス



More Timing Settings ダイアログ・ボックス

Settings ダイアログ・ボックスの **Timing Analysis Settings** ページで、**More Settings** をクリックして **More Timing Settings** ダイアログ・ボックスを表示します (図 8-26)。**More Timing Settings** ダイアログ・ボックスでは、多数のグローバル・タイミング解析オプションにアクセスできます。

図 8-26. More Timing Settings ダイアログ・ボックス



タイミング・レポート

クラシック・タイミング・アナライザ・レポートは、スタティック・タイミング解析の結果が記載された、コンパイル・レポートのセクションです。クラシック・タイミング・アナライザ・レポートには、すべてのクロック・ソースのクロック・セットアップとクロック・ホールドの測定値が記載されています。このレポートには、デザイン内のすべての出力ピンの t_{CO} 、デザイン内のすべての入力ピンの t_{SU} と t_H 、およびデザイン内の任意のピン間組み合わせパスの t_{PD} も表示されます。各種の解析およびデバイス機能については、他のレポートが作成されます。

デザイン用のタイミング・アサインメントがない場合、クラシック・タイミング・アナライザは検出したクロック・ノードに対してスラック・レポートを生成しません。クラシック・タイミング・アナライザは、個別またはグローバルの t_{SU} 、 t_H 、または t_{CO} のアサインメントが行われたピンのスラック測定値のみレポートします。正のスラックはパスがクロックのタイミング要件を上回るマージンを示します。負のスラックは、パスがクロックのタイミング要件を達成しないマージンを示します。



このタイミング解析レポートは、テキスト・フォーマットでも提供され、デザイン・ディレクトリに <リビジョン名>.tan.rpt というファイル名で置かれます。

コンパイル・レポートの Timing Analyzer フォルダの下の解析タイプを選択して、Clock Setup または Clock Hold などの解析レポートを表示します。図 8-27 に、クロック信号 clk のクロック・セットアップ・レポートの例を示します。

図 8-27. タイミング解析レポート

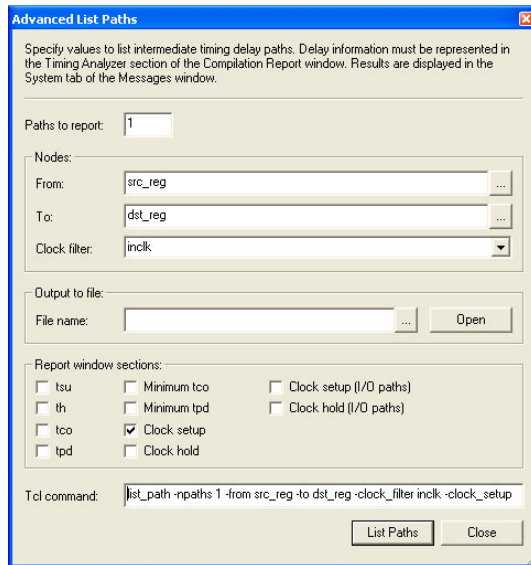
	Slack	Actual fmax (period)	From	To	From Clock	To Clock	Required Setup Relationship	Required Longest P2P Time
1	9.049 ns	91.32 MHz (period = 10.951 ns)	state_m_inst1filter_tap4	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
2	9.664 ns	96.75 MHz (period = 10.336 ns)	state_m_inst1filter_tap3	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
3	9.796 ns	98.00 MHz (period = 10.204 ns)	state_m_inst1filter_tap2	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
4	10.239 ns	102.45 MHz (period = 9.761 ns)	taps_inetbn[2]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
5	10.283 ns	102.91 MHz (period = 9.717 ns)	taps_inetbn[5]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
6	10.319 ns	103.30 MHz (period = 9.681 ns)	state_m_inst1filter_tap4	acc_inst3result[10]	clk	clk	20,000 ns	19,798 ns
7	10.391 ns	104.07 MHz (period = 9.609 ns)	taps_inetbn[0]	acc_inst3result[11]	clk	clk	20,000 ns	19,773 ns
8	10.480 ns	105.04 MHz (period = 9.520 ns)	taps_inetbn_2[0]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
9	10.523 ns	105.52 MHz (period = 9.477 ns)	taps_inetbn[1]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
10	10.757 ns	108.19 MHz (period = 9.243 ns)	taps_inetbn_1[2]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
11	10.768 ns	108.32 MHz (period = 9.232 ns)	taps_inetbn[4]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
12	10.951 ns	109.30 MHz (period = 9.149 ns)	taps_inetbn_2[2]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
13	10.934 ns	110.30 MHz (period = 9.066 ns)	state_m_inst1filter_tap3	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
14	10.996 ns	111.06 MHz (period = 9.004 ns)	taps_inetbn_1[4]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns
15	11.066 ns	111.93 MHz (period = 8.934 ns)	state_m_inst1filter_tap2	acc_inst3result[10]	clk	clk	20,000 ns	19,798 ns
16	11.110 ns	112.49 MHz (period = 8.890 ns)	taps_inetbn[3]	acc_inst3result[11]	clk	clk	20,000 ns	19,798 ns

アドバンスト・リスト・パス

Advanced List Paths ダイアログ・ボックスでは、任意の 2 つの有効なレジスタ間パスの間のインタコネクトやセル遅延など、特定のパスに関する詳細情報が提供されます (図 8-28)。

Advanced List Paths ダイアログ・ボックスでは、リストするパスの種類を選択できます。例えば、特定のクロックのクロック・セットアップとクロック・ホールドに関する詳細情報を得ることができます。さらに、ウィンドウ内の Tcl command フィールドは、カスタム Tcl スクリプトまたは Tcl コンソールで使用できる同等の Tcl コマンドにマッチングします。

図 8-28. Advanced List Paths ダイアログ・ボックス



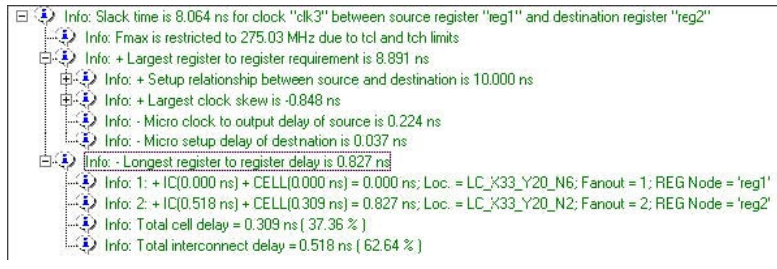
list path コマンドはタイミング解析レポートから直接実行できます。これを行うには、パスを右クリックして **List Path** をクリックします(図 8-29)。


Advanced List Paths ダイアログ・ボックスには、タイミング解析レポートに表示されるパスのみ表示されます。クラシック・タイミング・アナライザでレポートされるパス数を増やすには、Assignments メニューの **Timing Analysis Settings** をクリックします。Category リストで、**Timing Analysis Settings** を展開し、**Timing Analyzer Reporting** を選択します。**Timing Analyzer Reporting** ページで、クラシック・タイミング・アナライザがレポートする情報の範囲を指定します。



Advanced List Paths コマンドおよび **List Path** コマンドはどちらも、System message ウィンドウにパス情報を表示します。

図 8-29. メッセージ・ウィンドウ内のリスト・パス



 **Combined Fast/Slow Timing** オプションがイネーブルされている場合、List Path Tcl コマンドは Slow Model セクションにレポートされるパス遅延のみ表示します。

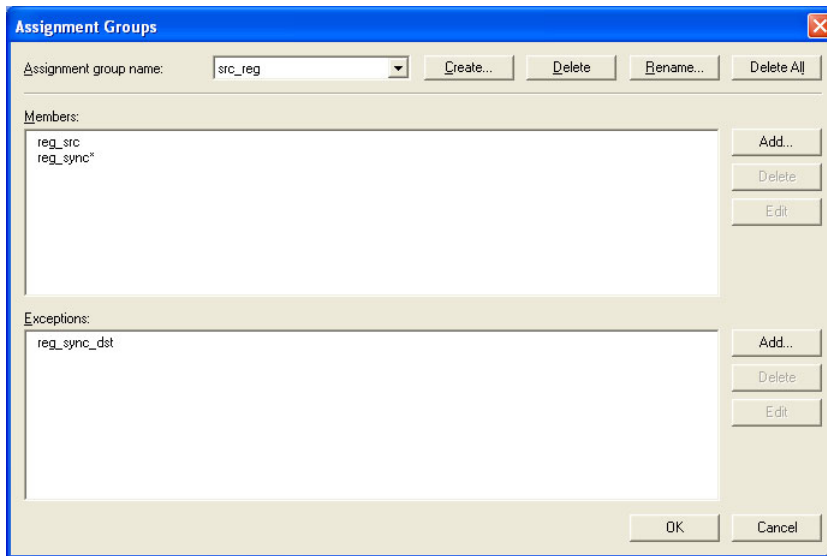
早期タイミング見積もり

早期タイミング見積もりを起動するには、Processing メニューの Start をポイントし、**Start Early Timing Estimate** をクリックします。**Early Timing Estimate** モードを指定するには、Assignments メニューの **Settings** をクリックします。**Category** リストで、**Compilation Processes Settings** を選択し、**Early Timing Estimate** を選択して、目的のタイミング見積もりモードをクリックします。早期タイミング見積もり機能について詳しくは、8-39 ページの「早期タイミング見積もり」を参照してください。

アサインメント・グループ

タイム・グループとしても知られるアサインメント・グループの定義、変更、および削除を 1 つのダイアログ・ボックスから行うには、Assignments メニューの **Assignment (Time) Groups** をクリックします。**Assignment Groups** ダイアログ・ボックスが表示されます (図 8-30)。

図 8-30. Assignment Groups ダイアログ・ボックス



スクリプトのサポート

Tcl スクリプトで、手順を実行してこの章で説明する設定を行うことができます。一部の手順はコマンド・プロンプトでも実行できます。スクリプティング・コマンド・オプションについては、**Quartus II Command-Line** および **Tcl API Help** ブラウザを参照してください。この Help ブラウザを使用するには、コマンド・プロンプトで次のコマンドを入力します。

```
quartus_sh --qhelp
```



この情報を PDF 形式で表示するには、**Scripting Reference Manual** を参照してください。



Tcl スクリプトについては、「**Quartus II ハンドブック Volume 2**」の「**Tcl スクリプト**」の章を参照してください。**Quartus II** ソフトウェアにおける設定および制約については、「**Quartus II Settings File Reference Manual**」を参照してください。コマンド・ライン・スクリプトについては、「**Quartus II ハンドブック Volume 2**」の「**Command-Line Scripting**」の章を参照してください。

クロックの作成

デザイン内でクロックの定義が可能な Tcl コマンドは、`create_base_clock` と `create_relative_clock` の 2 つです。

ベース・クロック

`create_base_clock` Tcl コマンドを使用して、ベース・クロックを定義します。

```
create_base_clock [-h | -help] [-long_help] -fmax <fmax> [-duty_cycle <integer>]
[-virtual] [-target <name>] [-no_target] [-entity <entity>] [-disable]
[-comment <comment>] <clock_name>
```

`sys_clk` という名前のベース・クロック設定をノード `clk_src` に 100 MHz (10 ns) の条件を定義して適用するには、以下の Tcl コマンドを入力します。

```
create_base_clock -fmax 100MHz -target clk_src sys_clk
```

派生クロック

`create_relative_clock` Tcl コマンドを使用して、相対クロックを定義します。

```
create_relative_clock [-h | -help] [-long_help] -base_clock <Base clock>
[-duty_cycle <integer>] [-multiply <integer>] [-divide <integer>] [-offset <offset>]
[-phase_shift <integer>] [-invert] [-virtual] [-target <name>] [-no_target]
[-entity <entity>] [-disable] [-comment <comment>] <clock_name>
```

`aux_clk` という名前の相対クロックをノード `rel_clk` に適用される 2 の通倍係数で定義するには、以下の Tcl コマンドを入力します。

```
create_relative_clock -base_clock sys_clk -multiply 2 -target rel_clk aux_clk
```

クロック・レイテンシ

`set_clock_latency` Tcl コマンドを使用して、アーリまたはレイト・クロック・レイテンシのアサインメントを作成できます。

```
set_clock_latency [-h | -help] [-long_help] [-early] [-late] -to <to> [<value>]
```

1ns のアーリ・クロック・レイテンシと 2ns のレイト・クロック・レイテンシをクロック・ノード `clk` に適用するには、以下の Tcl コマンドを入力します。

```
set_clock_latency -early -to clk 2ns
```

クロック不確実性

set_clock_uncertainty Tcl コマンドを使用して、以下の例に示すようなクロック不確実性のアサインメントを作成できます。

```
set_clock_uncertainty [-h] [-help] [-long_help [-from <source clock name> ] -to
<destination clock name> [-setup] [-hold] [-remove] [-disable] [-comment <comment>] <value>
```

ソース・クロック・ノード clk_src とデステイネーション・クロック・ノード clk_dst の間に 50 ps のクロック・セットアップ不確実性を適用するには、以下の Tcl コマンドを入力します。

```
set_clock_uncertainty -from clk_src -to clk_dst -setup 50ps
```

クロック・ノード clk_sys に 25 ps のクロック・ホールド不確実性を適用するには、以下の Tcl コマンドを入力します。

```
set_clock_uncertainty -to clk_sys -setup 25ps
```

タイミング・パスの切断

set_timing_cut_assignment Tcl コマンドを使用して、カット・タイミング・アサインメントを作成することができます。

```
set_timing_cut_assignment [-h | -help] [-long_help] [-from <from_node_list>]
[-to <to_node_list>] [-remove] [-disable] [-comment <comment>]
```

ソース・レジスタ reg1 からデステイネーション・レジスタ reg2 までのタイミング・パスを切断するには、以下の Tcl コマンドを入力します。

```
set_timing_cut_assignment -from reg1 -to reg2
```

入力遅延アサインメント

set_input_delay Tcl コマンドを使用して、入力遅延アサインメントを作成することができます。

```
set_input_delay [-h | -help] [-long_help] [-clk_ref <clock>] -to <input_pin> [-min] [-max]
[-clock_fall] [-remove] [-disable] [-comment <comment>] [<value>]
```

クロック・ソース clk で駆動されるレジスタに供給する、data_in の名前の入力ピンに、2 ns の入力最大遅延を適用するには、以下の Tcl コマンドを入力します。

```
set_input_delay -clk_ref clk -to data_in -max 2ns
```

最大および最小遅延

以下の Tcl コマンドは、それぞれ最大遅延アサインメントと最小関係アサインメントを作成します。

```
set_instance_assignment -name MAX_delay <value> -from <node> -to <node>
set_instance_assignment -name MIN_delay <value> -from <node> -to <node>
```

ソース・レジスタ `reg1` とデステイネーション・レジスタ `reg2` の間に、`8 ns` の最大遅延と `5 ns` の最小遅延を適用するには、以下の Tcl コマンドを入力します。

```
set_instance_assignment -name MAX_DELAY 8ns -from reg1 -to reg2
set_instance_assignment -name MIN_DELAY 5ns -from reg1 -to reg2
```

ソース・クロック `clk_src` からデステイネーション・クロック `clk_dst` までのすべてのパスに `10 ns` の最大遅延を適用するには、以下の Tcl コマンドを入力します。

```
set_instance_assignment -name MAX_DELAY 10ns -from clk_src -to clk_dst
```

最大クロック到達スキュー

以下の Tcl コマンドは、最大クロック到達スキュー・アサインメントを定義します。

```
set_instance_assignment -name max_clock_arrival_skew <value> -from <clock> -to <node>
```

`reg_group` と呼ぶ事前に定義されたタイム・グループに、クロック・ソース `clk` に対して `1 ns` の最大クロック到達スキューを適用するには、以下の Tcl コマンドを入力します。

```
set_instance_assignment -name max_clock_arrival_skew 1ns -from clk -to reg_group
```

最大データ到達スキュー

`set_instance_assignment -name max_data_arrival` Tcl コマンドを使用して、最大データ到達スキュー・アサインメントを作成します。

```
set_instance_assignment -name max_data_arrival_skew <value> -from <clock> -to <node>
```

`pin_group` と呼ぶ事前に定義されたピン・グループに、クロック・ソース `clk` に対して `1 ns` の最大データ到達スキューを適用するには、以下の Tcl コマンドを入力します。

```
set_instance_assignment -name max_data_arrival_skew 1ns -from clk -to pin_group
```

マルチサイクル

`set_multicycle_assignment` Tcl コマンドを使用して、マルチサイクル・アサインメントを作成します。

```
set_multicycle_assignment [-h | -help] [-long_help] [-setup] [-hold] [-start] [-end]
[-from <from_list>] [-to <to_list>] [-remove] [-disable] [-comment <comment>]
<path_multiplier>
```

ソース・レジスタ `reg1` とデステイネーション・レジスタ `reg2` の間に、2 のマルチサイクルセットアップと 1 のホールド・マルチサイクルを適用するには、以下の Tcl コマンドを入力します。

```
set_multicycle_assignment -setup -end -from reg1 -to reg2 2
set_multicycle_assignment -hold -end -from reg1 -to reg2 1
```

ソース・レジスタ `reg1` とデステイネーション・レジスタ `reg2` の間に、2 のソース・マルチサイクル・セットアップを適用するには、以下の Tcl コマンドを入力します。

```
set_multicycle_assignment -setup -start -from reg1 -to reg2 1
```

ソース・クロック `clk_src` からデステイネーション・クロック `clk_dst` までのすべてのパスに、2 のマルチサイクル・セットアップを適用するには、以下の Tcl コマンドを入力します。

```
set_multicycle_assignment -setup -end -from clk_src -to clk_dst 2
```

出力遅延アサインメント

Tcl コマンド `set_output_delay` を使用して、出力遅延アサインメントを作成します。

```
set_output_delay [-h | -help] [-long_help] [-clk_ref <clock>] -to <output_pin> [-min]
[-max] [-clock_fall] [-remove] [-disable] [-comment <comment>] [<value>]
```

クロック・ソース `clk` で駆動されるレジスタから供給される、`data_out` という出力ピンに 3 ns の出力最大遅延を適用するには、以下の Tcl コマンドを入力します。

```
set_output_delay -clk_ref clk -to data_out -max 3ns
```

タイミングのレポート

タイミング・レポートを生成するには、`report_timing Tcl` コマンドを使用します。

```
report_timing [-h | -help] [-long_help] [-npaths <number>] [-tsu] [-th] [-tco] [-tpd]
[-min_tco] [-min_tpd] [-clock_setup] [-clock_hold] [-clock_setup_io] [-clock_hold_io]
[-clock_setup_core] [-clock_hold_core] [-recovery] [-removal] [-dqs_read_capture]
[-stdout] [-file <name>] [-append] [-table <name>] [-from <names>] [-to <names>]
[-clock_filter <names>] [-src_clock_filter <names>] [-longest_paths] [-shortest_paths]
[-all_failures]
```

以下の例では、レジスタ `src_reg*` からレジスタ `dst_reg*` までのクロック・ソース `clk` に対するすべてのクロック・セットアップ・パスのリストを生成します。

```
report_timing -clock_setup -clock_filter clk -from src_reg* -to dst_reg*
```

セットアップおよびホールド関係

以下の `Tcl` コマンドは、それぞれセットアップ関係およびホールド関係のアサインメントを作成します。

```
set_instance_assignment -name SETUP_RELATIONSHIP <value> -from <node> -to <node>
set_instance_assignment -name HOLD_RELATIONSHIP <value> -from <node> -to <node>
```

ソース・レジスタ `reg1` とデステイネーション・レジスタ `reg2` の間に、**12 ns** のセットアップ関係と **2 ns** のホールド関係を適用する場合は、以下の `Tcl` コマンドを入力します。

```
set_instance_assignment -name SETUP_RELATIONSHIP 12ns -from reg1 -to reg2
set_instance_assignment -name HOLD_RELATIONSHIP 2ns -from reg1 -to reg2
```

ソース・クロック `clk_src` からデステイネーション・クロック `clk_dst` までのすべてのパスに、**10 ns** のセットアップ関係を適用するには、以下の `Tcl` コマンドを入力します。

```
set_instance_assignment -name SETUP_RELATIONSHIP 10ns -from clk_src -to clk_dst
```

アサインメント・グループ

アサインメント・グループは、以下のように `timegroup Tcl` コマンドを使用して作成します。

```
timegroup [-h | -help] [-long_help] [-add_member <name>] [-add_exception <name>]
[-remove_member <name>] [-remove_exception <name>] [-get_members] [-get_exceptions]
[-overwrite] [-remove] [-disable] [-comment <comment>] <group_name>
```

以下の例では、メンバ `dst_reg*` を持つ `reg_bank` と呼ばれるアサインメント・グループが作成され、レジスタ `dst_reg5` が排除されます。

```
timegroup reg_bank -add_member dst_reg* -add_exception dst_reg5
```

仮想クロック

`create_relative_clock` Tcl コマンドを `-virtual` スイッチと共に使用して、仮想クロックアサインメントを作成します。

```
create_relative_clock [-h | -help] [-long_help] -base_clock <Base clock>
[-duty_cycle <integer>] [-multiply <integer>] [-divide <integer>] [-offset <offset>]
[-phase_shift <integer>] [-invert] [-virtual] [-target <name>] [-no_target]
[-entity <entity>] [-disable] [-comment <comment>] <clock_name>
```

`brd_sys` という名前のベース・クロック設定 `clk_aux` から派生する仮想クロックを定義するには、以下の Tcl コマンドを入力します。


```
create_relative_clock -base_clock clk_aux -virtual brd_sys
```

MAX+PLUS II のタイミング 解析手法

この項では、MAX+PLUS II デザイン・ソフトウェアを起源とし、Quartus II ソフトウェアで使用可能な基本的なスタティック・タイミング解析とアサインメントについて説明します。

f_{MAX} 関係

最大クロック周波数とは、デザインのクロックが内部セットアップ・タイムおよびホールド・タイム要件に違反することなく動作可能な最高速度のことです。Quartus II ソフトウェアは、1つおよび複数のクロック・デザインにおいてスタティック・タイミング解析を実行します。

 性能要件が確実に満たされるように、デザイン内のすべてのクロック・ノードにクロック設定を適用します。詳しくは、8-7 ページの「クロック設定」を参照してください。

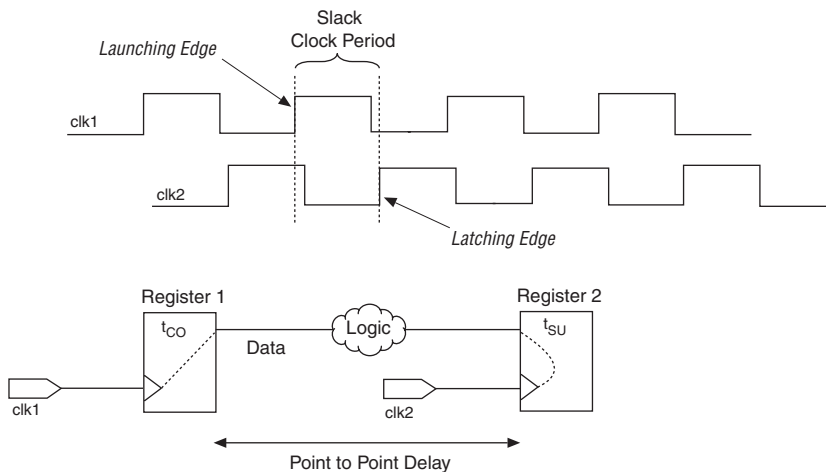
スラック

スラックとは、それによって f_{MAX} などのタイミング要件が適合または不適合になるマージンのことです。正のスラックは、要件が満たされるマージンを示します。負のスラックは、要件が満たされないマージンを示します。Quartus II ソフトウェアは、[計算式 35 ~ 38](#) を使用してスラックを求めます。

- (35) Clock Setup Slack = Longest Register-to-Register Requirement – Longest Register-to-Register Delay
- (36) Register-to-Register Requirement = Setup Relationship + Largest Clock Skew – micro t_{CO} of Source Register – micro t_{SU} of Destination Register
- (37) Clock Hold Slack = Shortest Register-to-Register Delay – Smallest Register-to-Register Requirement
- (38) Shortest Register-to-Register Requirement = Hold Relationship + Smallest Clock Skew – micro t_{CO} of Source Register – micro t_H of Destination Register

[図 8-31](#) にスラック計算図を示します。

図 8-31. スラック計算図



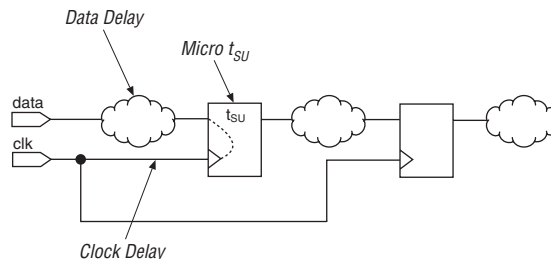
I/O タイミング

この項では、Quartus II ソフトウェアで I/O タイミング用に行われる基本的な測定について説明します。

t_{SU}

t_{SU} は、関連するクロック I/O ピンでのクロック遷移前にデータが外部入力ピンに到達し安定している必要がある時間を規定します。 t_{SU} 要件は、FPGA の I/O ピンを基準とした入力レジスタに対するこの関係を示しています。図 8-32 にクロック・セットアップ・タイムの図を示します。

図 8-32. クロック・セットアップ・タイム (t_{SU})

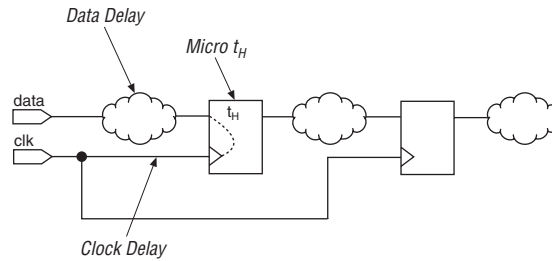


マイクロ t_{SU} は、レジスタの内部セットアップ・タイムです。これはレジスタの特性であり、レジスタに供給される信号の影響を受けません。計算式 39 は、図 8-32 に示す回路の clk に対するデータの t_{SU} を算出します。

$$(39) \quad t_{SU} = \text{Longest Data Delay} - \text{Shortest Clock Delay} + \text{micro } t_{SU} \text{ of Input Register}$$

t_H

t_H は、関連するクロック I/O ピンでのクロック遷移後に、外部ピンでデータを安定状態に保持する必要がある時間を規定します。 t_H 要件は、FPGA の I/O ピンを基準とした入力レジスタに対するこの関係を示しています。図 8-33 にクロック・ホールド・タイムの図を示します。

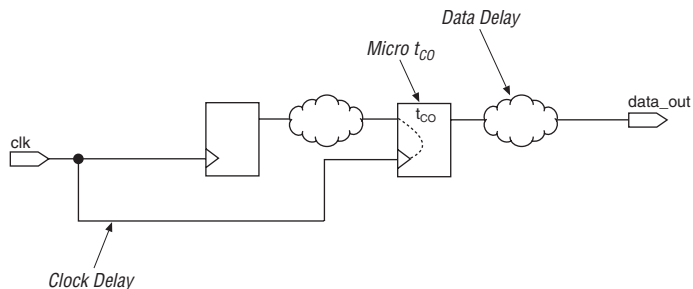
図 8-33. クロック・ホールド・タイム (t_H)

マイクロ t_H はレジスタの内部ホールド・タイムです。計算式 40 は、図 8-33 に示す回路の clk に対するデータの t_H を算出します。

$$(40) \quad t_H = \text{Longest Clock Delay} - \text{Shortest Data Delay} + \text{micro } t_H \text{ of Input Register}$$

t_{CO}

「Clock-to-Output」遅延は、レジスタを駆動する入力ピン上でのクロック遷移後に、レジスタから供給される出力ピンで有効出力を取得するのに必要な最大時間のことです。マイクロ t_{CO} は、レジスタの内部「Clock-to-Output」遅延です。図 8-34 に、「Clock-to-Output」遅延の図を示します。

図 8-34. 「Clock-to-Output」遅延 (t_{CO})

計算式 41 は、図 8-34 に示す回路のクロック・ノード clk に対する出力ピン $data_out$ の t_{CO} を算出します。

$$(41) \quad t_{CO} = \text{Longest Clock Delay} + \text{Longest Data Delay} + \text{micro } t_{CO} \text{ of Output Register}$$

最小 t_{CO}

最小「Clock-to-Output」遅延は、レジスタを駆動する入力ピン上でのクロック遷移後に、レジスタから供給される出力ピンで有効出力を取得するのに必要な最小時間のことです。マイクロ t_{CO} は、アルテラ FPGA 内のレジスタの内部「Clock-to-Output」遅延です。 t_{CO} アサインメントと異なり、最小 t_{CO} アサインメントは最短遅延パスを調べます。


(計算式 42)

$$(42) \quad \min t_{CO} = \text{Shortest Clock Delay} + \text{Shortest Data Delay} + \text{micro } t_{CO} \text{ of Output Register}$$

t_{PD}

ピン間遅延 (t_{PD}) は、入力ピンからの信号が組み合わせロジックを通過して伝播し、外部出力ピンに現れるのに必要な時間です (計算式 43)。

$$(43) \quad t_{PD} = \text{Longest Pin-to-Pin Delay}$$

 Quartus II ソフトウェアでは、入力ピンと出力ピンの間で t_{PD} アサインメントを行うことができます。

最小 t_{PD}

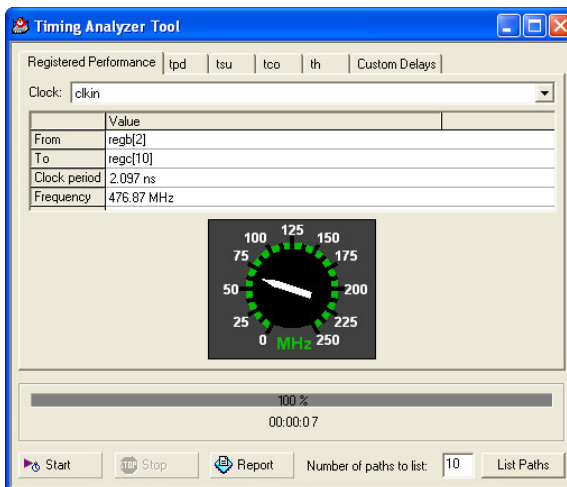
最小ピン間遅延 (t_{PD}) は、入力ピンからの信号が組み合わせロジックを通過して伝播し、外部出力ピンに現れるのに必要な時間です。 t_{PD} アサインメントとは異なり、最小 t_{PD} アサインメントは最短ピン間遅延に適用されます (計算式 44)。

$$(44) \quad \min t_{PD} = \text{Shortest Pin-to-Pin Delay}$$

タイミング解析ツール

クラシック・スタティック・タイミング解析フローおよび制約を容易にするために、Quartus II ソフトウェアには Tools メニューで使用できる、MAX+PLUS II スタイルのタイミング解析ツールがあります。タイミング解析ツールは、レジスタ間での性能、I/O タイミング、およびカスタム遅延値をレポートする、MAX+PLUS II のタイミング解析ツールに類似したシンプルなインタフェースを提供します (図 8-35 参照)。

図 8-35. タイミング解析ツール



まとめ

進化するデザインおよび革新的なプロセス・テクノロジーにより、FPGA デザインの大規模化と高性能化が求められています。デザインが複雑になると、デザインのタイミング要件の検証を支援するスタティック・タイミング解析ツールの機能強化が求められます。高度なスタティック・タイミング解析ツールがなければ、複雑なデザインでの回路故障のリスクが生じます。クラシック・タイミング・アナライザは、SOPC (system-on-a-programmable-chip) デザインを実現する際に、極めて重要となる強力なスタティック・タイミング解析機能セットを備えています。

