Quartus II TimeQuest タイミング・アナライザ・クックブック

101 Innovation Drive
San Jose, CA 95134
www.altera.com

MNL-01035-1.3

Software Version: 10.1
Document Version: 1.3
Document Date: January 2011

MNL-01035-1.3

目次

1. Quartus II TimeQuest タイミング・アナライザ・クックブック

このマニュアルでは、デザイン・シナリオ、制約のガイドライン、および推奨事項の集まりが含まれています。このガイドラインを適用するには、TimeQuest Timing Analyzer の経験、及び Synopsys Design Constraints (SDC) の基本的な知識が必要です。

TimeQuest アナライザおよび SDC については、www.altera.com/timequest を参照してください。

クロックと生成クロック

このセクションでは、さまざまなクロック構造および以下の構造を制約する方法を説明します。

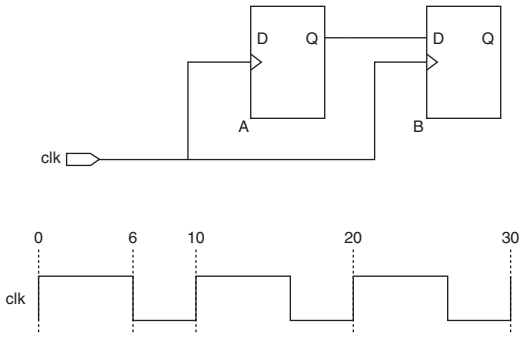
- 「Basic Non-50/50 のデューティ・サイクル・クロック」
- 1–2 のページの「オフセット・クロック」
- 1–2 のページの「-divide_by を使用した Basic Clock Divider」
- 1–4 のページの「トグル・レジスタ生成クロック」
- 1–7 のページの「マルチ周波数解析」

Basic Non-50/50 のデューティ・サイクル・クロック

このセクションで説明されている制約は create_clock です。

クロックのデューティ・サイクルは、デザインごとに異なる場合があります。デフォルトでは、TimeQuest アナライザで作成されたクロックのデューティ・サイクルは 50/50 に設定されています。ただし、-waveform のオプションを使用してクロックのデューティ・サイクルを変更することができます。図 1–1 に、60/40 のデューティ・サイクル・クロックによってクロックされる単純なレジスタ間パスを示します。

図 1–1. 簡単なレジスタ間パス



式 1–1 には、60/40 デューティ・サイクル・クロックの制約を示しています。

式 1–1. Non-50/50 デューティ・サイクル・クロックの制約

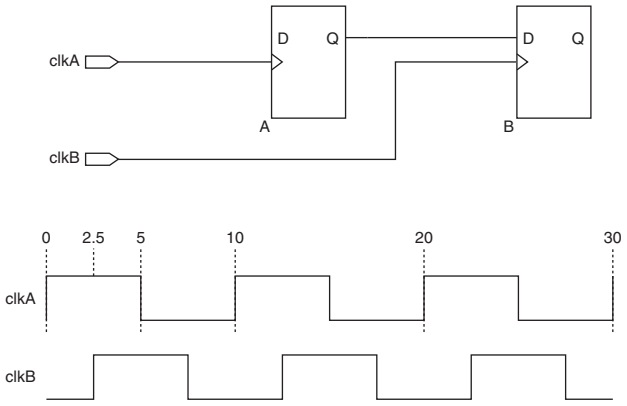
```
#60/40 duty cycle clock
create_clock \
    -period 10.000 \
    -waveform {0.000 6.000} \
    -name clk6040 [get_ports {clk}]
```

オフセット・クロック

このセクションで説明されている制約は create_clock です。

TimeQuest アナライザでクロックを制約する場合、クロックの最初の立ち上がりまたは立ち下がりエッジは、絶対 0 で発生します。-waveform のオプションを使用してクロックのオフセットを作成することができます。図 1–2 は 2.5 ns にシフトされ、clkB によってクロックされる簡単なレジスタ間パスを示します。

図 1–2. clkB でクロックされる簡単なレジスタ間パス



式 1–2 に、オフセット・クロックの制約を示します。

式 1–2. オフセット・クロックの制約

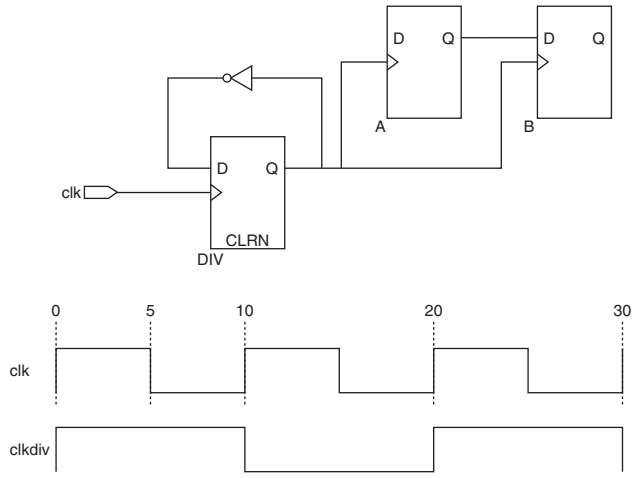
```
# -waveform defaults to 50/50 duty cycle
create_clock -period 10.000 \
    -name clkA \
    [get_ports {clkA}]

#create a clock with a 2.5 ns offset
create_clock -period 10.000 \
    -waveform {2.500 7.500} \
    -name clkB [get_ports {clkB}]
```

-divide_by を使用した Basic Clock Divider

このセクションで説明されている制約は create_clock および create_generated_clock です。

派生クロックはソース・クロックより遅いクロック・ソースからデザイン内のクロックを生成できます。クロック・ソースから派生した低速クロックを制約する場合、-divide オプションを使用します。図 1–3 には、2 分周した派生クロックを示しています。

図 1–3. 2 分周した派生クロック



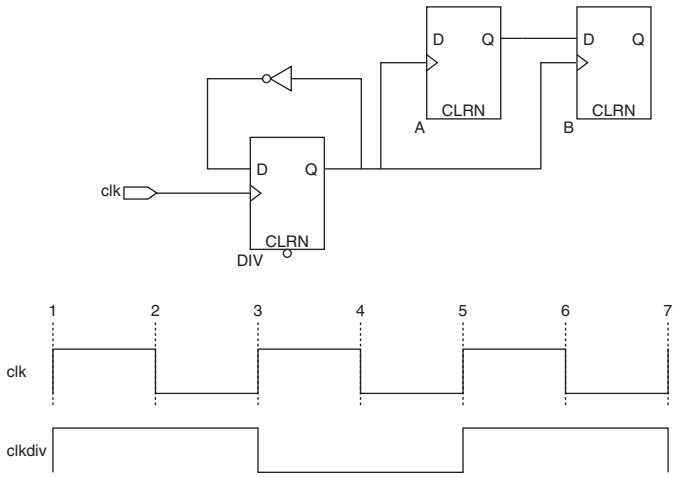
式 1–3 に、-waveform クロックでの分周の制約を示しています。

式 1–3. -waveform クロック制約の分周

```
create_clock -period 10.000 -name clk [get_ports {clk}]

# Using -divide_by option
create_generated_clock \
    -divide_by 2 \
    -source [get_ports {clk}] \
    -name clkdiv \
    [get_pins {DIV|q}]

# Alternatively use pins to constrain the divider without
# knowing about the master clock
create_generated_clock \
    -divide_by 2 \
    -source [get_pins {DIV|clk}] \
    -name clkdiv \
    [get_pins {DIV|q}]

# the second option works since the
# clock pin of the register DIV is
# connected to the same net fed by the
# clock port clk.
```

また、-edges のオプションを使用して分周クロックを作成することができます。図 1–4 に、-edges のオプションを使用して 2 分周クロックを示しています。

図 1–4. -edges のオプションを使用して 2 分周クロック



式 1–4 に、-waveform クロックでの分周の制約を示しています。

式 1–4. -waveform クロック制約の分周

```
# Edge numbers are based on the master clock
create_generated_clock \
    -edges {1 3 5} \
    -source [get_pins {DIV|clk}] \
    -name clkdiv \
    [get_pins {DIV|q}]
```

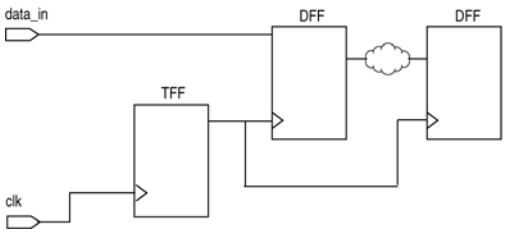
トグル・レジスタ生成クロック

2 分周クロックを作成するためにトグル・レジスタを使用してください。トグル・レジスタに供給するデータが論理 "1" の値に保持し、10 ns の周期のクロックによって供給されている場合、レジスタの出力は、20 ns の周期のクロックになります。

トグル・レジスタ・クロックに対する制約は、前の例と非常に似ています。

図 1–5 に、2 分周クロックを生成するトグル・レジスタを示します。

図 1–5. 2 分周クロックを生成するトグル・レジスタ



式 1–5 に、トグル・レジスタの制約を示しています。

式 1–5. トグル・レジスタの制約

```
# Create a base clock
create_clock \
    -period 10.000 \
    -name clk \
    [get_ports {clk}]

# Create the generated clock on the output
# of the toggle register.
create_generated_clock \
    -name tff_clk \
    -source [get_ports {clk}] \
    -divide_by 2 \
    [get_pins {tff|q}]
```

PLL クロック

このセクションで説明されている制約は derive_pll_clocks、create_clock、および create_generated_clock です。

PLL（Phase-Locked Loops）は、アルテラ FPGA にクロック合成を実行するために使用されます。すべての出力クロックは適切な分析のために制約される必要があります。PLL を制約する 3 つの方法があります。

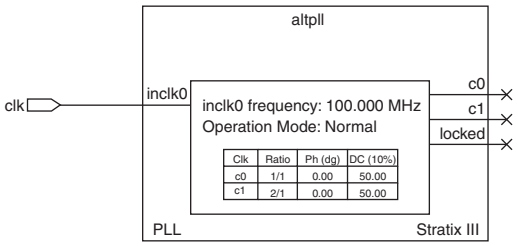
- **自動的にベースクロックと PLL 出力クロックを作成する**
- **手動でベースクロックを作成し、PLL 出力が自動的にクロックを供給**
- **手動でベースクロックを作成し、PLL 出力は、手動でクロックを供給**

このセクションでは、各方法の利点を示しています。

アルテラの FPGA の PLL 回路は、ALTPLL メガファンクションを使用してデザインに組み込まれています。

図 1–6 に、ALTPLL メガファンクションの例を示しています。

図 1–6. ALTPLL メガファンクション



方法 1 – Create Base Clocks and PLL Output Clocks Automatically

この方法は、自動的に PLL の入力および出力クロックの両方を制約することができます。ALTPLL メガファンクションで指定されたすべての PLL のパラメータは、PLL の入力および出力クロックを制約するために使用されています。メガファンクションへの変更は自動的に更新されます。PLL パラメータへの変更を追跡したり、PLL の入力および出力クロックを作成するときに正しい値を指定する必要はありません。

自動的にすべての入力と出力を制約するには、`-create_base_clocks` オプションを指定して `derive_pll_clocks` コマンドを使用してください。TimeQuest アナライザは、PLL の MegaWizard™ Plug-In Manager のインスタンス化に基づいて正しい設定を決定します。式 1–6 に、このコマンドを示しています。

式 1–6. Constraining PLL Base Clocks Automatically

```
derive_pll_clocks -create_base_clocks
```

方法 2 – Create Base Clocks Manually and PLL Output Clocks Automatically

この方法では、手動で PLL の入力クロックを制約し、TimeQuest アナライザは自動的に PLL の出力クロックを制約できるようにすることができます。また、ALTPLL メガファンクションで指定された入力クロック周波数の代わりに別の入力クロックの周波数を指定することができます。PLL 出力クロックは、自動的にメガファンクションで指定されたパラメータを使用して作成されます。同じ PLL 出力クロックのパラメータを保持しながら、さまざまな入力クロック周波数を試すことができます。

☞ 指定された入力クロックの周波数は、現在コンフィギュレーションされている PLL と互換性があることを確認してください。

`derive_pll_clocks`コマンドを使用してこのメソッドを使用して手動でPLLの入力クロックを作成することができます。式 1–7 は、このコマンドを示しています。

式 1–7. Constraining PLL Base Clocks Manually

```
create_clock -period 10.000 -name clk [get_ports {clk}]

derive_pll_clocks
```

方法 3 – Create Base Clocks and PLL Output Clocks Manually

この方法では、手動で PLL の入力クロックと出力クロックの両方を制約することができます。すべての PLL パラメータが指定されており、パラメータ値は、ALTPLL メガファンクションで指定されたものと異なる場合があります。さらに、様々な PLL 入力と出力の周波数とパラメータを試すことができます。

create_clockとcreate_generate_clockのコマンドの組み合わせでこのメソッドを使用することができます。式 1–8 は、これらのコマンドを示しています。

式 1–8. Constraining PLL Output and Base Clocks Manually

```
create_clock -period 10.000 -name clk [get_ports {clk}]

create_generated_clock \
    -name PLL_C0 \
    -source [get_pins {PLL|altpll_component|pll|inclk[0]}] \
    [get_pins {PLL|altpll_component|pll|clk[0]}]

create_generated_clock \
    -name PLL_C1 \
    -multiply_by 2 \
    -source [get_pins {PLL|altpll_component|pll|inclk[0]}] \
    [get_pins {PLL|altpll_component|pll|clk[1]}]
```

マルチ周波数解析

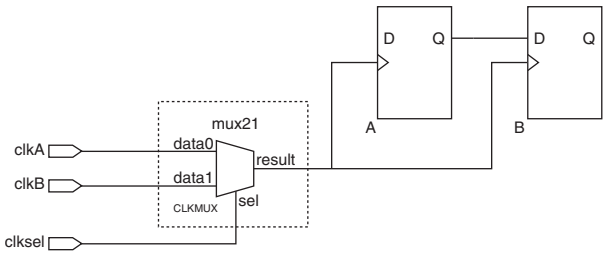
いくつかのデザインは、FPGA をドライブする複数のクロックを必要とします（1 クロックが他よりも速くなったり、遅くなる可能性がある）。

クロック・マルチプレキシング

このセクションで説明されている制約は create_clock と set_clock_groups です。

クロック・マルチプレキシングでは、2 つ以上のクロックから選択できます。図 1–7 に、標準的な 2:1 クロック・マルチプレクサのための制約を示します。

図 1–7. 標準的 2:1 クロック・マルチプレクサの制約



式 1–9 に、クロック・マルチプレクサのための制約を示します。

式 1–9. クロック・マルチプレクサの制約

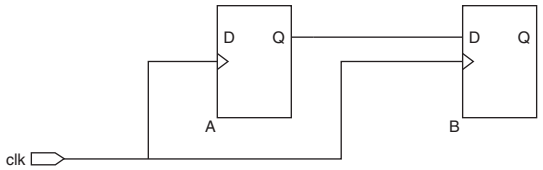
```
#Create the first input clock clkA to the mux
create_clock -period 10.000 -name clkA [get_ports {clkA}]

#Create the second input clock clkB to the mux
create_clock -period 20.000 -name clkB [get_ports {clkB}]

#Cut paths between clkA and clkB
set_clock_groups -exclusive -group {clkA} -group {clkB}
```

外部でスイッチされたクロック

このセクションで説明されている制約は create_clock および set_clock_groups です。

外部マルチプレクサまたはジャンパの設定により、デジタル・システムは、同じクロック・ポートに異なるクロック周波数を提供することができます。TimeQuest タイミング・アナライザは create_clock 制約および -add オプションでこの動作をモデル化することができます。図 1–8 に、100-MHz のクロックまたは 50-MHz のクロックで、クロックのポートのクロックをドライブすることができる簡単なレジスタ間パスを示します。

図 1–8. 簡単なレジスタ間デザイン



式 1–10 に、外部でスイッチされたクロックの制約を示します。

式 1–10. 外部でスイッチされたクロックの制約

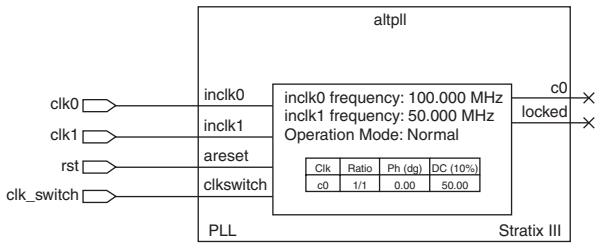
```
# The clk port can be driven at 100MHz (10ns) or
# 50MHz (20ns)

# clkA is 10ns
create_clock \
    -period 10.000 \
    -name clkA \
    [get_ports {clk}]

# clkB is 20ns assigned to the same port
# Requires -add option
create_clock \
    -period 20.000 \
    -name clkB \
    [get_ports {clk}] \
    -add

set_clock_groups \
    -exclusive \
    -group {clkA} \
    -group {clkB}
```

PLL クロック・スイッチオーバー

このセクションで説明されている制約は derive_pll_clocks です。

PLL は、アルテラの FPGA（図 1–9）における PLL クロック・スイッチオーバー機能を持つ 2 つの使用可能な入力クロックを選択できます。

図 1–9. PLL クロック・スイッチオーバー



式 1–11 に、PLL クロック・スイッチオーバーのための制約を示します。

式 1–11. PLL クロック・スイッチオーバーの制約

```
#create a 10ns clock for clock port clk0
create_clock \
    -period 10.000 \
    -name clk0 \
    [get_ports {clk0}]

#create a 20ns clock for clock port clk1
create_clock \
    -period 20.000 \
    -name clk1 \
    [get_ports {clk1}]

#automatically create clocks for the PLL output clocks
#derive_pll_clocks automatically makes the proper
#clock assignments for clock-switchover
derive_pll_clocks
```

I/O 制約

この項では、以下の章で構成されています。

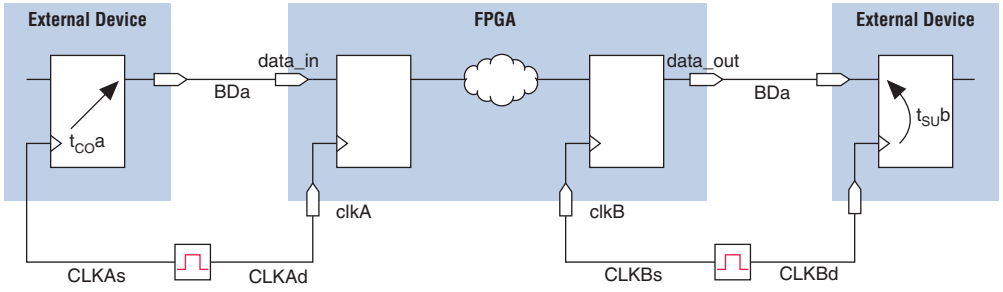
- 1–10 のページの「バーチャル・クロックでの入力および出力の遅延」
- 1–13 のページの「トライ・ステート出力」
- 1–14 のページの「システム同期入力」

バーチャル・クロックでの入力および出力の遅延

すべての入力および出力遅延、TimeQuest アナライザが導出できるように、仮想クロックを参照し、derive_clock_uncertainty のコマンドが使用されているときに正しいクロック不確実性の値を適用する必要があります。入力と出力は基準ベース・クロックまたは PLL クロックが仮想クロックより遅延する場合、derive_clock_uncertainty によって決定されるイントラ・クロックおよびインター・クロックの転送クロック不確実性は I/O ポートに正しく適用されていません。また、仮想クロックを使用して、追加の外部クロックの不確実性は derive_clock_uncertainty によって決定されるクロックの不確実性とは無関係に適用することができます。

仮想クロックの特性は、入力（入力遅延）や出力（出力遅延）ポートのいずれかのクロックに使用する元のクロックと同一である必要があります。図 1–10 に、仮想クロックが入力ポートと出力ポートに使用されるシンプルなチップ間のデザインを示しています。

図 1–10. チップ間のデザイン



リファレンス仮想クロックは図 1–10 で示されている入力と出力遅延を制約するには、式 1–12 に示すように、制約を使用してください。

式 1–12. 仮想クロックを参照する入力と出力遅延 (その 1)

```
#specify the maximum external clock delay from the external
#device
set CLKAs_max 0.200
#specify the minimum external clock delay from the external
#device
set CLKAs_min 0.100
#specify the maximum external clock delay to the FPGA
set CLKAd_max 0.200
#specify the minimum external clock delay to the FPGA
set CLKAd_min 0.100
#specify the maximum clock-to-out of the external device
set tCOa_max 0.525
#specify the minimum clock-to-out of the external device
set tCOa_min 0.415
#specify the maximum board delay
set BDa_max 0.180
#specify the minimum board delay
set BDa_min 0.120

#create the input maximum delay for the data input to the
#FPGA that
accounts for all delays specified
set_input_delay -clock clk \
-max [expr $CLKAs_max + $tCOa_max + $BDa_max - $CLKAd_min] \
[get_ports {data_in[*]}]

#create the input minimum delay for the data input to the #FPGA that
accounts for all delays specified
set_input_delay -clock clk \
-min [expr $CLKAs_min + $tCOa_min + $BDa_min - $CLKAd_max] \
[get_ports {data_in[*]}]

#create the input clock
create_clock -name clkA -period 10 [get_ports clkA]
#create the associated virtual input clock
create_clock -name clkA_virt -period 10
#specify any uncertainty from the external clock to the virtual clock
set_clock_uncertainty -from { clkA_virt } -setup 0.25

#create the output clock
create_clock -name clkB -period 5 [get_ports clkB]
#create the associated virtual input clock
create_clock -name clkB_virt -period 5
#specify any uncertainty from the external clock to the virtual clock
set_clock_uncertainty -from { clkB_virt } -setup 0.25

#determine internal clock uncertainties
derive_clock_uncertainty
#create the input delay referencing the virtual clock
#specify the maximum external clock delay from the external
#device
set CLKAs_max 0.200
#specify the minimum external clock delay from the external
#device
set CLKAs_min 0.100
#specify the maximum external clock delay to the FPGA
set CLKAd_max 0.200
#specify the minimum external clock delay to the FPGA
set CLKAd_min 0.100
```

式 1–12. 仮想クロックを参照する入力と出力遅延 (その 2)

```
#specify the maximum clock-to-out of the external device
set tCOa_max 0.525
#specify the minimum clock-to-out of the external device
set tCOa_min 0.415
#specify the maximum board delay
set BDa_max 0.180
#specify the minimum board delay
set BDa_min 0.120

#create the input maximum delay for the data input to the
#FPGA that accounts for all delays specified
set_input_delay -clock clkA_virt \
-max [expr $CLKAs_max + $tCOa_max + $BDa_max - $CLKAd_min] \
[get_ports {data_in[*]}]

#create the input minimum delay for the data input to the
#FPGA that accounts for all delays specified
set_input_delay -clock clkA_virt \
-min [expr $CLKAs_min + $tCOa_min + $BDa_min - $CLKAd_max] \
[get_ports {data_in[*]}]

#creating the output delay referencing the virtual clock
#specify the maximum external clock delay from the external
#device
set CLKBs_max 0.100
#specify the minimum external clock delay from the external
#device
set CLKBs_min 0.050
#specify the maximum external clock delay to the FPGA
set CLKBd_max 0.100
#specify the minimum external clock delay to the FPGA
set CLKBd_min 0.050
#specify the maximum clock-to-out of the external device
set tSUb_max 0.500
#specify the hold time of the external device
set tHb 0.400
#specify the maximum board delay
set BDb_max 0.100
#specify the minimum board delay
set BDb_min 0.080

#create the output maximum delay for the data output from the
#FPGA that accounts for all delays specified
set_output_delay -clock clkB_virt \
-max [expr $CLKBs_max + $tSUb_max + $BDb_max - $CLKBd_min] \
[get_ports {data_out}]

#create the output minimum delay for the data output from the
#FPGA that accounts for all delays specified
set_output_delay -clock clkB_virt \
-min [expr $CLKBs_min - $tHb + $BDb_min - $CLKBd_max] \
[get_ports {data_out}]
```

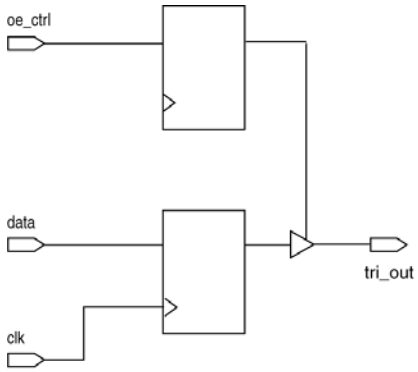
トライ・ステート出力

トライ・ステート出力で、有効なデータ信号またはハイ・インピーダンス信号が入力ポートからドライブすることができます。信号のタイミングは、デザインの全体的なシステム・タイミングで重要です。

トライ・ステート出力のタイミング制約は、通常の出力ポートと同じです。

図 1–11 に、トライ・ステート・バッファで供給される標準的な出力を示します。

図 1–11. トライ・ステート・バッファで供給される標準的な出力



式 1–13 に、トライ・ステートの出力ポートの制約を示します。

式 1–13. トライ・ステートの出力ポートの制約

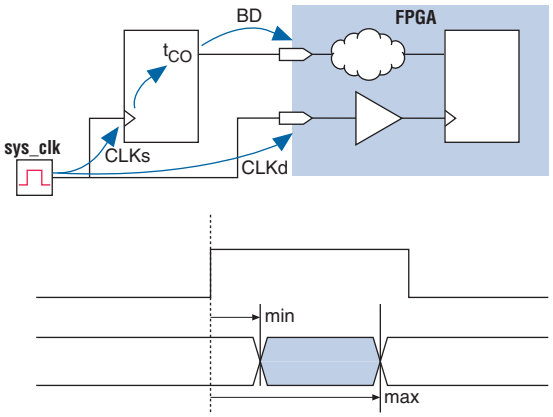
```
# Base clock
create_clock [get_ports {clk}] \
        -name {clk} \
        -period 10.0 \
        -waveform {0.0 5.0}

# Virtual clock for the output port
create_clock  \
        -name {clk_virt} \
        -period 10.0 \
        -waveform {0.0 5.0}

# Output constraints
set_output_delay 2.0 \
        -max \
        -clock [get_clocks {clk_virt}] \
        [get_ports {tri_out}]

set_output_delay 1.0 \
        -min \
        -clock [get_clocks {clk_virt}] \
        [get_ports {tri_out}]
```

システム同期入力

このセクションで説明されている制約は create_cloc と set_input_delay です。

図 1–12 に、インタフェースのための入力遅延を指定するために必要な典型的なチップ間の入力インタフェースおよび様々なパラメータを示しています。

図 1–12. シンプルチップ間入力インタフェース



式 1–14 には、システム同期入力の制約を示します。

式 1–14. システム同期入力の制約

```
#specify the maximum external clock delay from the external device
set CLKs_max 0.200

#specify the minimum external clock delay from the external device
set CLKs_min 0.100

#specify the maximum external clock delay to the FPGA
set CLKd_max 0.200

#specify the minimum external clock delay to the FPGA
set CLKd_min 0.100

#specify the maximum clock-to-out of the external device
set tCO_max 0.525

#specify the minimum clock-to-out of the external device
set tCO_min 0.415

#specify the maximum board delay
set BD_max 0.180

#specify the minimum board delay
set BD_min 0.120

#create a clock 10ns
create_clock -period 10 -name sys_clk [get_ports sys_clk]

#create the associated virtual input clock
create_clock -period 10 -name virt_sys_clk

#create the input maximum delay for the data input to the FPGA that
#accounts for all delays specified
set_input_delay -clock virt_sys_clk \
    -max [expr $CLKs_max + $tCO_max + $BD_max - $CLKd_min] \
    [get_ports {data_in[*]}]

#create the input minimum delay for the data input to the FPGA that
#accounts for all delays specified
set_input_delay -clock virt_sys_clk \
    -min [expr $CLKs_min + $tCO_min + $BD_min - $CLKd_max] \
    [get_ports {data_in[*]}]
```

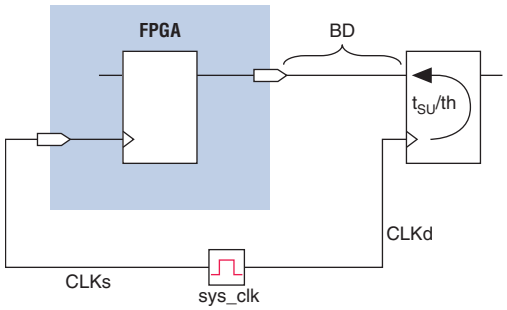
ソース同期入力と出力インタフェースの制約について詳しくは、*「AN 433: Constraining and Analyzing Source-Synchronous Interfaces」* を参照してください。

システム同期出力

このセクションで説明されている制約は create_clock と set_output_delay. です。

図 1–13 に、インタフェースのための出力遅延を指定するために必要な典型的なチップ間の出力インタフェースおよび様々なパラメータを示しています。

図 1–13. シンプルチップ間のインタフェース



式 1–15 には、システム同期出力の制約を示しています。

式 1–15. システム同期出力の制約

```
#specify the maximum external clock delay to the FPGA
set CLKs_max 0.200

#specify the minimum external clock delay to the FPGA
set CLKs_min 0.100

#specify the maximum external clock delay to the external device
set CLKd_max 0.200

#specify the minimum external clock delay to the external device
set CLKd_min 0.100

#specify the maximum setup time of the external device
set tSU 0.125

#specify the minimum setup time of the external device
set tH 0.100

#specify the maximum board delay
set BD_max 0.180

#specify the minimum board delay
set BD_min 0.120

#create a clock 10ns
create_clock -period 10 -name sys_clk [get_ports sys_clk]

#create the associated virtual input clock
create_clock -period 10 -name virt_sys_clk

#create the output maximum delay for the data output from the FPGA that
#accounts for all delays specified
set_output_delay -clock virt_sys_clk \
    -max [expr $CLKs_max + $BD_max + $tSU - $CLKd_min] \
    [get_ports {data_in[*]}]

#create the output minimum delay for the data output from the FPGA that
#accounts for all delays specified
set_output_delay -clock virt_sys_clk \
    -min [expr $CLKs_min + $BD_min - $tH - $CLKd_max] \
    [get_ports {data_in[*]}]
```

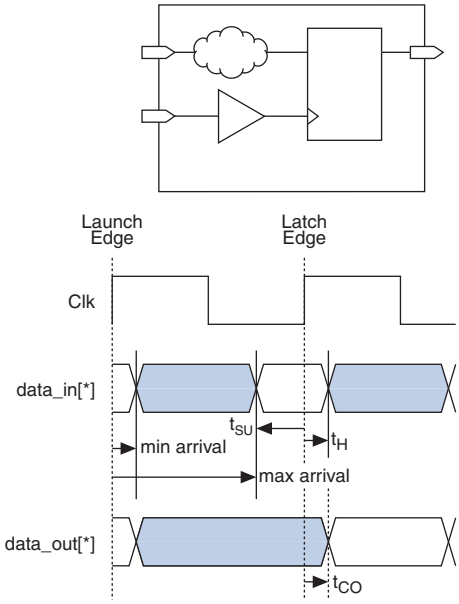
ソース同期入力と出力インタフェースの制約について詳しくは、*「AN 433: Constraining and Analyzing Source-Synchronous Interfaces」*を参照してください。

I/O タイミング要件 t_{SU}、t_H、および t_{CO}

このセクションで説明されている制約は、set_input_delay と set_output_delay です。

式 1–16 は t_{SU} と t_H の使用して set_input_delay を指定する方法を示しています、と set_output_delay を使用して t_{CO} を指定する方法。図 1–14 は、必要なタイミング仕様と FPGA とのタイミング図を示します。

図 1–14. I/O のタイミング仕様



式 1–16 には、t_{SU}、t_H、および t_{CO} の制約を示しています。

式 1–16. t_{SU}、t_H、および t_{CO} の制約

```
#Specify the clock period
set period 10.000

#Specify the required tSU
set tSU 1.250

#Specify the required tH
set tH 0.750

#Specify the required tCO
set tCO 0.4

#create a clock 10ns
create_clock -period $period -name clk [get_ports sys_clk]

#create the associated virtual input clock
create_clock -period $period -name virt_clk

set_input_delay -clock virt_clk \
    -max [expr $period - $tSU] \
    [get_ports {data_in[*]}]

set_input_delay -clock virt_clk \
    -min $tH \
    [get_ports {data_in[*]}]

set_output_delay -clock virt_clk \
    -max [expr $period - $tCO] \
    [get_ports {data_out[*]}]
```

例外

この項では、以下の章で構成されています。

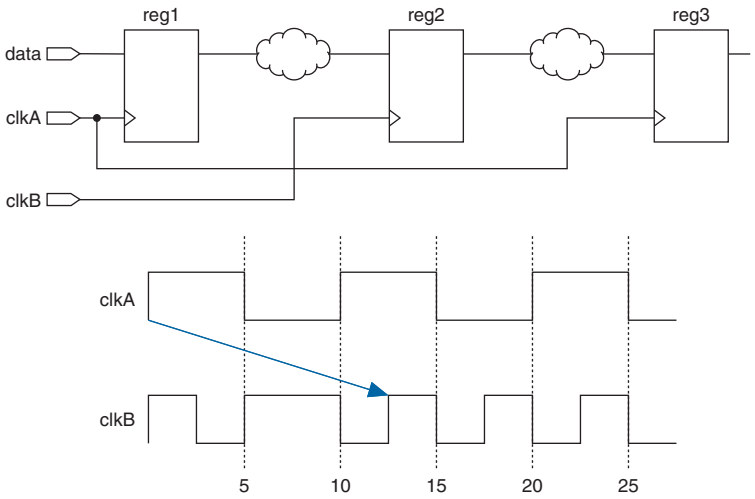
- 「マルチサイクル例外」
- 1–21 のページの「フォルス・パス」

マルチサイクル例外

この項で説明された制約は、create_clock および set_multicycle_path です。

デフォルトでは、TimeQuest タイミング・アナライザは、任意のレジスタ間パスのセットアップ関係とホールド関係の両方を決定するために、シングル・サイクルの分析を使用しています。これにより、最も制限されたのセットアップとホールドの必要条件に結果します。しかし、マルチサイクル例外は、任意のセットアップ関係とホールド関係をリラックスするために使用することができます。図 1–15 には、簡単なレジスタ間パスを示します。

図 1–15. レジスタ間パス



マルチサイクルは、クロックからのクロック転送したり、個々のレジスタに適用することができます。。クロック 間の転送にマルチサイクルを適用すると、ソース・クロックおよびデスティネーション・クロックが供給されるレジスタ間のパスのターゲット・クロックの指定されたすべてのセットアップとホールドの関係に影響を与えます。式 1–17 に、マルチサイクルの制約を示しています。

式 1–17. マルチサイクル のクロック間

```
create_clock -period 10 [get_ports clkA]
create_clock -period 5 [get_ports clkB]

set_multicycle_path -from [get_clocks {clkA}] -to [get_clocks {clkB}] -setup -end 2
```

式 1–17 では、セットアップ関係は、ソース・クロックは clkA である、またデスティネーション・クロックは clkB の場合に任意のレジスタ間パスの追加のデスティネーション・クロック周期によって緩和されます。これで、デフォルトの 5 ns の代わりに、12.5 ns のセットアップ関係を持つレジスタ reg1 と reg2 に結果します。レジスタ reg2 と reg3 の間のセットアップ関係は、マルチサイクルに影響を受けません。

個々のレジスタにマルチサイクルを適用すると、指定されたレジスタのセットアップ関係またはホールド関係にのみ影響を与えます。式 1–18 に、個々のレジスタにマルチサイクルを適用するための制約を示しています。

式 1–18. レジスタ間のマルチサイクル

```
create_clock -period 10 [get_ports clkA]
create_clock -period 5 [get_ports clkB]

set_multicycle_path -from [get_pins {reg1|q}] -to [get_pins {reg2|d}] -setup -end 2
```

式 1–18 では、セットアップ関係はレジスタ reg1 からレジスタ reg2 までレジスタ間のパスの追加のデスティネーション・クロック周期によって緩和されます。これで、デフォルトの 5 ns の代わりに、12.5 ns のセットアップ関係を持つレジスタ reg1 と reg2 に結果します。レジスタ reg2 と reg3 の間のセットアップ関係は、マルチサイクルに影響を受けません。

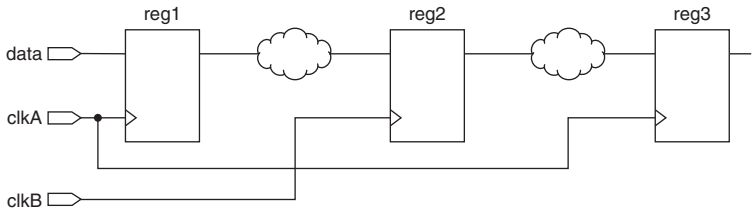
TimeQuest アナライザで使用可能なマルチサイクル例外のタイプの詳細については *「Quartus II ハンドブック Volume 3」* の *「Best Practices for the TimeQuest Timing Analyzer」* の章を参照してください。

フォルス・パス

このセクションで説明されている制約は create_clock と set_false_path です。

すべてのパスのタイミングを分析する必要はありません。非クリティカル・パスの同期化は、タイミング解析から削除またはカットすることができます。非クリティカル・パスを宣言するときに、Quartus II フィッタは、クリティカル・パスの最適化に集中し、全体のコンパイル時間を短縮することができます。図 1–16 に、レジスタ reg1 からレジスタ reg2 までパスをカットすることができる簡単なレジスタ間のデザインを示しています。

図 1–16. レジスタ間のパス



フォルス・パスは、クロック間に転送したり、個々のレジスタに適用できます。クロック間の転送にフォルス・パスを適用すると、ターゲット・クロック間のすべてのパスがカットされます。式 1–19 に、フォルス・パスを適用するための制約を示しています。

式 1–19. フォルス・パスのクロック間

```
create_clock -period 10 [get_ports clkA]
create_clock -period 5 [get_ports clkB]

set_false_path -from [get_clocks {clkA}] -to [get_clocks {clkB}]
```

式 1–19 では、ソース・クロックが clkA であり、デスティネーション・クロックが clkB の場合に、パスは任意のレジスタ間パスのために TimeQuest アナライザでカットし、分析されていません。これは、clkB によってクロックされるソース・レジスタおよび clkA によってクロックされるデスティネーション・レジスタのレジスタ間のパスには影響しません。

式 1–19 の場合、set_false_path コマンドは、クロック clkA から clkB へのパスをカットします。コマンドは clkB から clkA へのパスをカットしていません。clkB から clkA へのパスをカットするには、追加の set_false_path コマンドを適用する必要があります（例えば、set_false_path -from clkB -to clkA）。また、1 つのコマンドで clkA から clkB へのパスおよび clkB から clkA へのパスをカットするには set_clock_groups を使用することができます。

set_clock_groups コマンドについて詳しくは、「Quartus II ヘルプ」の *「Set Clock Groups Dialog Box (set_clock_groups)」* を参照してください。

個々のレジスタへのフォルス・パスを適用すると、指定されたパスのみをカットします。式 1–20 に、このための制約を示しています。

式 1–20. フォルス・パスのレジスタ間

```
create_clock -period 10 [get_ports clkA]
create_clock -period 5 [get_ports clkB]

set_false_path -from [get_pins {reg1|q}] -to [get_pins {reg2|d}]
```

式 1–20 に、レジスタ reg1 からレジスタ reg2 までのレジスタ間のパスがカットされます。他のすべてのパスは影響を受けません。

その他

この項では、以下の章で構成されています。

- 「JTAG 信号」
- 1–23 のページの「複数のクロックの入力および出力遅延」
- 1–28 のページの「クロック・イネーブル・マルチサイクル」

JTAG 信号

このセクションで説明されている制約は create_clock、set_input_delay、および set_output_delay です。

多くのイン・システム・デバッグ・ツールは、アルテラ FPGA の JTAG インタフェースを使用します。JTAG インタフェースによってデザインをデバッグするときに、JTAG 信号の TCK、TMS、TDI、および TDO がデザインの一部として実装されています。このため、TimeQuest アナライザは、制約のないパス・レポートが生成される場合、これらの信号は制約のない信号としてフラグされます。表 1–1 には、制約のない可能性のある JTAG 信号を示します。

表 1–1. JTAG 信号

信号名	説明
altera_reserved_tck	JTAG テスト・クロック入力ポート
altera_reserved_tms	JTAG テスト・モードの選択入力ポート
altera_reserved_tdi	JTAG テスト・データ入力のライン入力ポート
altera_reserved_tdo	JTAG テスト・データ出力のライン出力ポート

式 1–21 での SDC コマンドを適用することにより、JTAG 信号を制約することができます。

式 1–21. JTAG 信号の制約

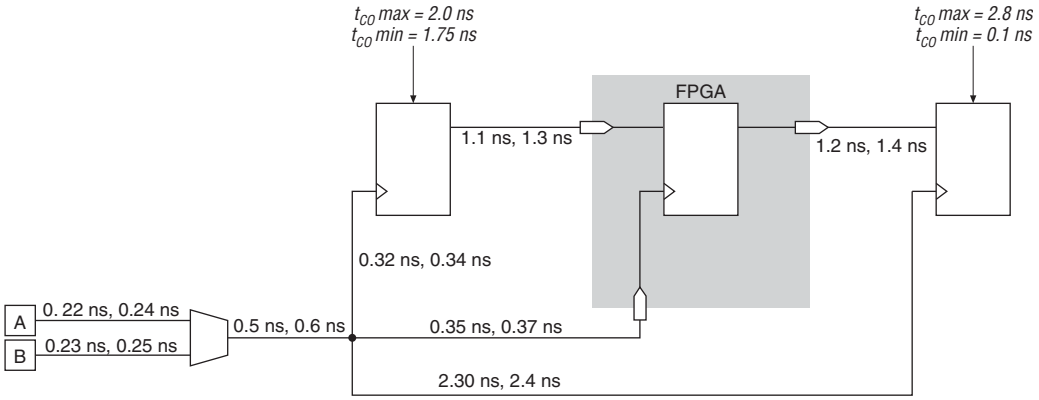
```
#JTAG Signal Constraints
#constrain the TCK port
create_clock \
-name tck \
-period "10MHz" \
[get_ports altera_reserved_tck]

#cut all paths to and from tck
set_clock_groups -exclusive -group [get_clocks tck]

#constrain the TDI port
set_input_delay \
-clock tck \
20 \
[get_ports altera_reserved_tdi]

#constrain the TMS port
set_input_delay \
-clock tck \
20 \
[get_ports altera_reserved_tms]

#constrain the TDO port
set_output_delay \
-clock tck \
20 \
[get_ports altera_reserved_tdo]
```

複数のクロックの入力および出力遅延

このセクションで説明されている制約は create_clock、create_generated_clock、set_clock_groups、set_clock_latency、set_input_delay、および set_output_delay です。

これらの制約は、プライマリおよびセカンダリ・クロックの両方を提供しています。プライマリ・クロックは、低スピードで冗長クロックとして機能するため、メイン・クロックおよびセカンダリ・クロックとして機能します。図 1–17 に、このセットアップの例を示します。

図 1–17. 簡単なレジスタ間デザイン



式 1–22 は、複数のクロックで入力遅延のコマンドを示します。

式 1–22. 複数のクロックでの入力遅延（その 1 ）

```
#########################
# Create all the clocks #
#########################
# Create variables for the clock periods.
set PERIOD_CLK_A 10.000
set PERIOD_CLK_B 7.000

# Create the clk_a clock which will represent the clock
# that routes to the FPGA.
create_clock \
    -name {clk_a} \
    -period \
    $PERIOD_CLK_A \
    [get_ports {clk}]

# Create the clk_b clock which will represent the clock
# that routes to the FPGA.
# Note the -add is needed because this is the second clock
# that has the same 'clk' port as a target.
create_clock \
    -name {clk_b} \
    -period $PERIOD_CLK_B \
    [get_ports {clk}] \
    -add

# Create a virtual clock which will represent the clock
# that routes to the external source device when clk_a is
# selected a the external mux.
create_clock \
    -name virtual_source_clk_a \
    -period $PERIOD_CLK_A

# Create a virtual clock which will represent the clock
# that routes to the external source device when clk_b is
# selected a the external mux.
create_clock \
    -name virtual_source_clk_b \
    -period $PERIOD_CLK_B

# Create a virtual clock which will represent the clock
# that routes to the external destination device when clk_a
# is selected a the external mux.
create_clock \
    -name virtual_dest_clk_a \
    -period $PERIOD_CLK_A

# Create a virtual clock which will represent the clock
# that routes to the external destination device when clk_b
# is selected a the external mux.
create_clock \
    -name virtual_dest_clk_b \
    -period $PERIOD_CLK_B
```

式 1–22. 複数のクロックでの入力遅延（その２）

```
##########################################
# Cut clock transfers that are not valid #
##########################################
# Cut this because virtual_source_clk_b can not be clocking
# the external source device at the same time that clk_a is
# clocking the FPGA.
set_clock_groups -exclusive \
    -group {clk_a} \
    -group {virtual_source_clk_b}
# Cut this because virtual_source_clk_a can not be clocking
# the external source device at the same time that clk_b is
# clocking the FPGA.
set_clock_groups -exclusive \
    -group {clk_b} \
    -group {virtual_source_clk_a}
# Cut this because virtual_dest_clk_b can not be clocking
# the external destination device at the same time that
# clk_a is clocking the FPGA.
set_clock_groups -exclusive \
    -group {clk_a} \
    -group {virtual_dest_clk_b}
# Cut this because virtual_dest_clk_a can not be clocking
# the external destination device at the same time that
# clk_b is clocking the FPGA
set_clock_groups -exclusive \
    -group {clk_b} \
    -group {virtual_dest_clk_a}

########################################
# Define the latency of all the clocks #
########################################
# Since TimeQuest does not know what part of the clock
# latency is common we must simply remove the common part
# from the latency calculation. For example when
# calculating the latency for virtual_source_clk_a we must
# ignore the 220ps,240ps route and the 500ps/600ps mux
# delay if we want to remove the common clock path
# pessimism.
#
# Define fastest and slowest virtual_source_clk_a path to
# the external source device.
set_clock_latency -source \
    -early .320 \
[get_clocks virtual_source_clk_a]
set_clock_latency -source \
    -late .340 \
    [get_clocks virtual_source_clk_a]
# Define fastest and slowest virtual_source_clk_b path to
# the external source device.
set_clock_latency -source \
    -early .320 \
    [get_clocks virtual_source_clk_b]
set_clock_latency -source \
    -late .340 \
    [get_clocks virtual_source_clk_b]
# Define fastest and slowest clk_a path to the FPGA.
set_clock_latency -source \
    -early .350 \
    [get_clocks clk_a]
set_clock_latency -source \
    -late .370 \
    [get_clocks clk_a]
```

式 1–22. 複数のクロックでの入力遅延（その３）

```
# Define fastest and slowest clk_b path to the FPGA.
set_clock_latency -source \
    -early .350 \
    [get_clocks clk_b]
set_clock_latency -source \
    -late .370 \
    [get_clocks clk_b]
# Define fastest and slowest virtual_dest_clk_a path to
# the external destination device.
set_clock_latency -source \
    -early 2.3 \
    [get_clocks virtual_dest_clk_a]
set_clock_latency -source \
    -late 2.4 \
    [get_clocks virtual_dest_clk_a]
# Define fastest and slowest virtual_dest_clk_b path to
# the external destination device.
set_clock_latency -source \
    -early 2.3 \
    [get_clocks virtual_dest_clk_b]
set_clock_latency -source \
    -late 2.4 \
    [get_clocks virtual_dest_clk_b]

#####################################
# Constrain the input port 'datain' #
#####################################
# This Tco is the min/max value of the Tco for the
# external module.
set Tco_max 2.0
set Tco_min 1.75
# Td is the min/max trace delay of datain from the
# external device
set Td_min 1.1
set Td_max 1.3

# Calculate the input delay numbers
set input_max [expr $Td_max + $Tco_max]
set input_min [expr $Td_min + $Tco_min]
# Create the input delay constraints when clk_a is selected
set_input_delay \
    -clock virtual_source_clk_a \
    -max $input_max \
    [get_ports datain]
set_input_delay \
    -clock virtual_source_clk_a \
    -min $input_min \
    [get_ports datain]
# Create the input delay constraints when clk_b is selected
set_input_delay \
    -clock virtual_source_clk_b \
    -max $input_max \
    [get_ports datain] \
    -add_delay
set_input_delay \
    -clock virtual_source_clk_b \
    -min $input_min \
    [get_ports datain] \
    -add_delay
```

式 1–22. 複数のクロックでの入力遅延（その４）

```
########################################
# Constrain the output port 'dataout' #
########################################
# This Tsu/Th is the value of the Tsu/Th for the external
# device.
set Tsu 2.8
set Th 0.1
# This is the min/max trace delay of dataout to the
# external device.
set Td_min 1.2
set Td_max 1.4

# Calculate the output delay numbers
set output_max [expr $Td_max + $Tsu]
set output_min [expr $Td_min - $Th]

# Create the output delay constraints when clk_a is
# selected.
set_output_delay \
    -clock virtual_dest_clk_a \
    -max $output_max \
    [get_ports dataout]
set_output_delay \
    -clock virtual_dest_clk_a \
    -min $output_min \
    [get_ports dataout]
# Create the output delay constraints when clk_b is
# selected.
set_output_delay \
    -clock virtual_dest_clk_b \
    -max $output_max \
    [get_ports dataout] \
    -add_delay
set_output_delay \
    -clock virtual_dest_clk_b \
    -min $output_min \
    [get_ports dataout] \
    -add_delay
```

クロック・イネーブル・マルチサイクル

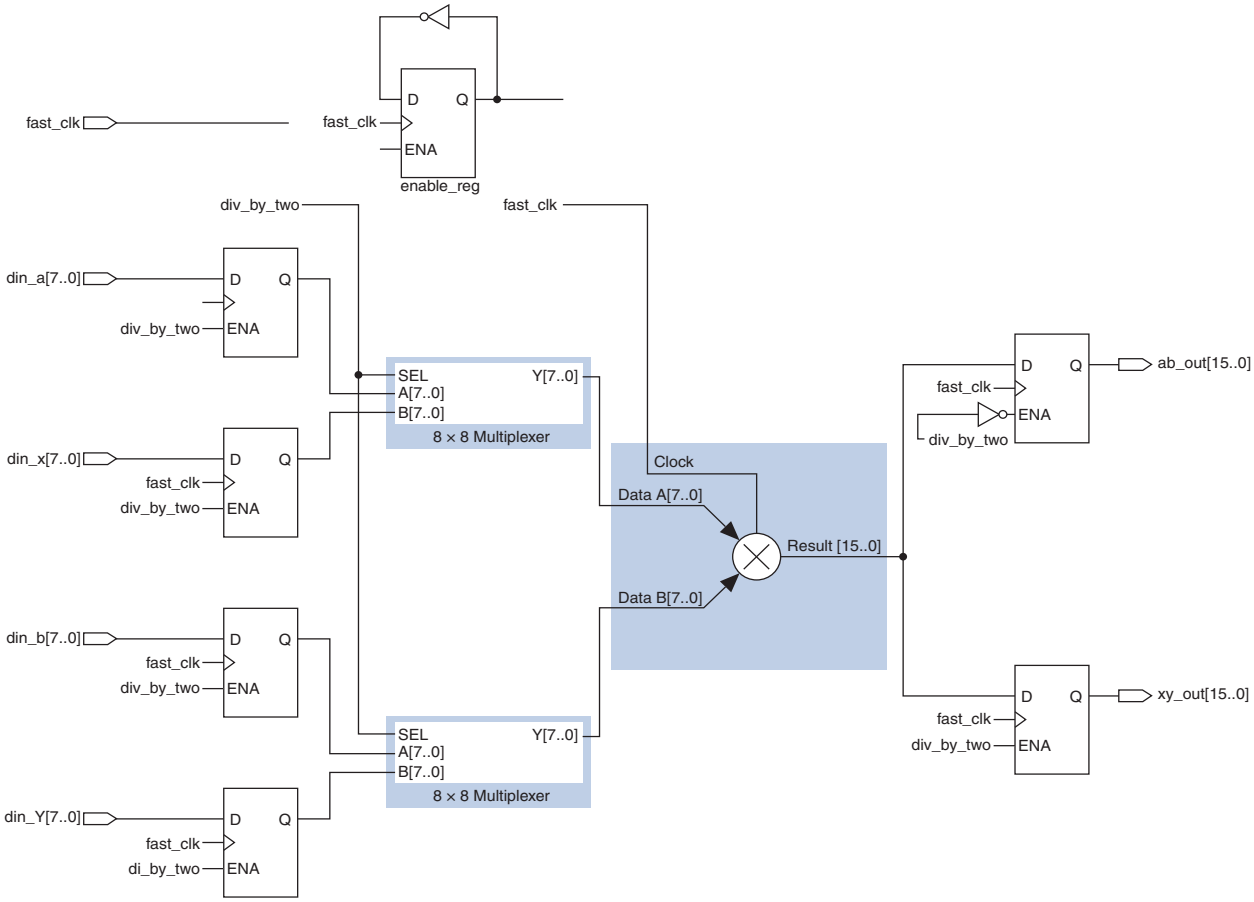
このセクションで説明されている制約は create_clock、set_multicycle_path、および get_fanouts です。

クロック・イネーブル・マルチサイクルとのレジスタの対応ポートに基づいてマルチサイクルを指定することができます。例えば、図 1–18 に、レジスタ enable_reg が din_a_reg[7..0]、din_b_reg[7..0]、din_x_reg[7..0]、din_y_reg[7..0]、a_times_b、および x_times_y レジスタ用にレジスタ・イネーブル信号を作成するために使用される単純な回路を示します。

イネーブルレジスタ enable_reg は、レジスタの 2 倍のクロック周期イネーブル・パルスを生成するので、マルチサイクル例外は、正確な解析に適用する必要があります。レジスタ enable_reg で供給されるイネーブル・ドライブ・レジスタに 2 のマルチサイクル・セットアップと 1 のマルチサイクル・ホールドを適用する必要があります。マルチサイクル例外はデスティネーション・レジスタが enable_reg によって

制御されるレジスタ間のパスににのみ適用されます。これを実現するには、すべてのイネーブル・ドライブ・レジスタに set_multicycle_path 例外を適用することができます。すべてのイネーブル・ドライブ・レジスタを指定する必要があるので、これは、面倒な場合があります。また、式 1–23 に示すように、set_multicycle_path と get_fanouts の組み合わせを使用できます。

図 1–18. クロック・イネーブル・マルチサイクル・デザイン



式 1–23. クロック・イネーブル・マルチサイクル制約

```
#Setup multicycle of 2 to enabled driven destination registers
set_multicycle_path 2 -to [get_fanouts [get_pins enable_reg|q*] \
-through [get_pins -hierarchical *|*ena*]] -end -setup

#Hold multicycle of 1 to enabled driven destination registers
set_multicycle_path 1 -to [get_fanouts [get_pins enable_reg|q*] \
-through [get_pins -hierarchical *|*ena*]] -end -hold
```

set_multicycle_path の例外のターゲットは、フィード・レジスタのイネーブル・ポートを供給するレジスタ enable_reg のすべてのファンアウトに制限されています。以下のオプションを使用してください。

```
[get_fanouts [get_pins enable_reg|q*] -through [get_pins -hierarchical *|*ena*]]
```

表 1–2 に、マルチサイクル例外が適用された後、デザイン内のすべてのイネーブル・ドライブ・レジスタ間のパスの新しいセットアップ関係とホールド関係を示しています。

表 1–2. イネーブル・ドライブ・レジスタのセットアップ関係とホールド関係

ソース・レジスタ	デスティネーション・レジスタ	セットアップ関係	ホールド関係
din_a[*]	din_a_reg[*]	2× （ラッチ・エッジ時間）	1× （ラッチ・エッジ時間）
din_x[*]	din_x_reg[*]	2× （ラッチ・エッジ時間）	1× （ラッチ・エッジ時間）
din_b[*]	din_b_reg[*]	2×（ラッチ・エッジ時間）	1× （ラッチ・エッジ時間）
din_y[*]	din_y_reg[*]	2× （ラッチ・エッジ時間）	1× （ラッチ・エッジ時間）
fast_mult:mult|*	a_times_b[*]	2×（ラッチ・エッジ時間）	1× （ラッチ・エッジ時間）
fast_mult:mult|*	x_times_y[*]	2× （ラッチ・エッジ時間）	1× （ラッチ・エッジ時間）
enable_reg	din_a_reg[*], din_b_reg[*], a_times_b[*], x_times_y[*]	2× （ラッチ・エッジ時間）	1× （ラッチ・エッジ時間）

表 1–2 に、セットアップ関係とホールド関係が enable_reg レジスタで開始され、それぞれ 2 と 1 のいずれかのイネーブル・ドライブ・レジスタで終了されることを示します。これらのパスがセットアップ関係とホールド関係に変更が必要ではない場合、元の関係を適用するには、次のマルチサイクル例外を使用することができます。

```
set_multicycle_path 1 -from [get_pins enable_reg|q*] -end -setup
set_multicycle_path 0 -from [get_pins enable_reg|q*] -end -hold
```

マルチサイクル例外について詳しくは、*「Quartus II ハンドブック Vol 3」*の *「Best Practices for the TimeQuest Timing Analyzer」*の章を参照してください。

改訂履歴

表 1 にこのユーザー・ガイドの章の改訂履歴を示します。

表 1. 改訂履歴

日付	バージョン	変更内容
2011 年 1 月	1.3	■ 新しい項「トグル・レジスタ生成クロック」および「トライ・ステート出力」を追加。 ■ テキストのマイナーな編集。
2010 年 3 月	1.2	スクリプト例のエラーを修正。
2010 年 3 月	1.1	スクリプト例のエラーを修正。
2008 年 8 月	1.0	初版。

アルテラへのお問い合わせ

Altera 製品に関する最新情報については、表 2 を参照してください。

表 2. アルテラのお問い合わせ先

お問い合わせ先 (1)	お問い合わせ方法	アドレス
テクニカル・サポート	ウェブサイト	www.altera.com/mysupport/
テクニカル・トレーニング	ウェブサイト	www.altera.com/training
製品資料	ウェブサイト	www.altera.com/literature
（ソフトウェア・ライセンス）	メール	authorization@altera.com

表 2 の注：

(1) 詳しくは、日本アルテラまたは販売代理店にお問い合わせください。

表記規則

本資料では、表 3 に示す表記規則を使用しています。

表 3. 表記規則（その 1）

書体	意味
太字かつ文頭が大文字	コマンド名、ダイアログ・ボックス・タイトル、チェックボックス・オプション、およびダイアログ・ボックス・オプションは、太字かつ文頭が大文字で表記されています。例：**Save As** ダイアログ・ボックス。
太字	外部タイミング・パラメータ、ディレクトリ名、プロジェクト名、ディスク・ドライブ名、ファイル名、ファイルの拡張子、およびソフトウェア・ユーティリティ名は、太字で表記されています。例：$\mathbf{f_{MAX}}$, **\qdesigns** ディレクトリ、**d:** ドライブ、**chiptrip.gdf** ファイル。
斜体かつ文頭が大文字	資料のタイトルは、斜体かつ文頭が大文字で表記されています。例：*AN 75: High-Speed Board Design*。

表 3. 表記規則（その 2）

書体	意味
斜体	内部タイミング・パラメータおよび変数は、斜体で表記されています。例：t_{PIA}, $n + 1$。 変数は、山括弧 （< >）で囲み、斜体で表記されています。例：< ファイル名 >、< プロジェクト名 >.**pof** ファイル。
文頭が大文字	キーボード・キーおよびメニュー名は、文頭が大文字で表記されています。例：Delete キー、Options メニュー。
「小見出しタイトル」	資料内の小見出しおよびオンライン・ヘルプ・トピックのタイトルは、鉤括弧で囲んでいます。例：「表記規則」
Courier フォント	信号およびポート名は、Courier フォントで表記されています。例：data1、tdi、input. アクティブ Low 信号は、サフィックス nn で表示されています。（例：resetn） 表示されているとおりに入力する必要があるものは、Courier フォントで表記されています。例：c:\qdesigns\tutorial\chiptrip.gdf. また、Report ファイルのような実際のファイル、ファイルの構成要素 (例：AHDL キーワードの SUBDESIGN)、ロジック・ファンクション名 (例：TRI) も Courier フォントで表記されています。
1.、2.、3. および a.、b.、c. など	手順など項目の順序が重要なものは、番号が付けられリスト形式で表記されています。
■ ■	箇条書きの黒点などは、項目の順序が重要ではないものに付いています。
✓	チェックマークは、1 ステップしかない手順を表します。
☞	指差しマークは、要注意箇所を表しています。
CAUTION	注意は、製品または作業中のデータに損傷を与えたり、破壊したりするおそれのある条件や状況に対して注意を促します。
WARNING	警告は、ユーザーに危害を与えるおそれのある条件や状況に対して注意を促します。
↵	矢印は、Enter キーを押すことを示しています。
👣	足跡マークは、詳細情報の参照先を示しています。