
計算機科学実験及演習 3

ハードウェア

「e. タイミング制約の設定と検証」

「CADツールを用いた設計フロー」補足資料

目次

- TimeQuest Timing Analyzer
- タイミング制約
 - Clockの設定
 - Clock以外の設定
 - 例外パスの設定
- 検証
 - タイミング違反しない条件
 - 解析結果の見方
- タイミング違反の解消
- Q&A(主に本スライドの内容に関するQ&Aを掲載)

※本スライドの内容は随時改定していく予定です。適宜最新の内容を確認してください。

※載せてほしい情報など要望があれば検討しますので連絡ください。

※この資料だけで解決できないことはスタッフに質問してください。

TimeQuest Timing Analyzer

- 静的タイミング解析ツールである

- どんな入力が増えても要求速度で設計通り動くかだけ調べる

- ▶ 現実では論理ゲートや配線に遅延が発生し、遅延が大or小さすぎて回路が正しく動かないことがあるため、それを確かめるために

- ※ タイミング以前にそもそも設計が正しいかはRTL/Gate level Simで

The screenshot displays the Quartus Prime Lite Edition interface. The main window shows the TimeQuest Timing Analyzer tool, which is used for static timing analysis. A table of timing data is visible, showing slack, from node, to node, launch clock, latch clock, relationship, clock skew, and data delay. The table is as follows:

Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
-4.519	reg[0]	reg[15]_OTERM429	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM427	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM425	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM423	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM421	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM419	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM417	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM415	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM413	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM411	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM409	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM407	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM405	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM403	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM401	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM429	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM427	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM425	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM423	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496
-4.519	reg[0]	reg[15]_OTERM421	pll_0ataclk_gpr[1]clk[0]	pll_0ataclk_gpr[1]clk[0]	4.000	0.162	0.496

The screenshot also shows the Waveform window for RTL/Gate level simulation, displaying signals such as /lock/clk, /lock/rst_n, /lock/keys_in, /lock/lock_out, /lock/current_st..., /lock/next_state, /lock/reset_cou..., and /lock/stran/co... The waveform shows a clock signal and several control signals over time.

タイミング設計

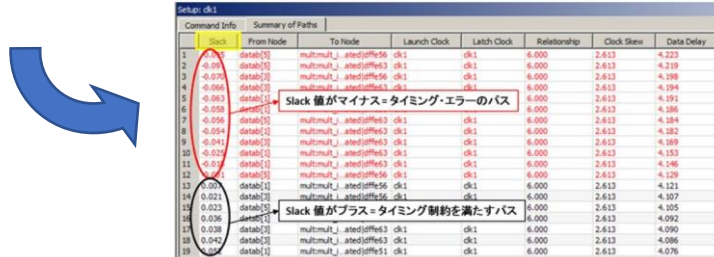
- 回路が実機上で動くために**タイミング設計**を行う必要がある
 1. **タイミング制約(要求速度など)を設定する**
 - 設計者の仕事
 2. **静的タイミング解析(STA)を行い、タイミング違反がないか検証**
 - STA = Static Timing Analysis
 - Timing Analyzerの仕事
 3. **違反をなくすように合成、配置配線で最適化/設計変更**
 - Quartus Compiler / 設計者の仕事
- タイミング設計をしないと、回路が動く保証はない

タイミング制約

• 開発ツールに指示する、タイミングに関する設計要求例

- 50ns周期(= 20MHz)のClockで動かしたい
- FPGA外部で5ns程度の遅延が発生してからFPGAに入力信号が入る

Timing Analyzerはそのタイミング制約下でタイミング違反が起こらないか調べる。違反があると知らせてくれる。



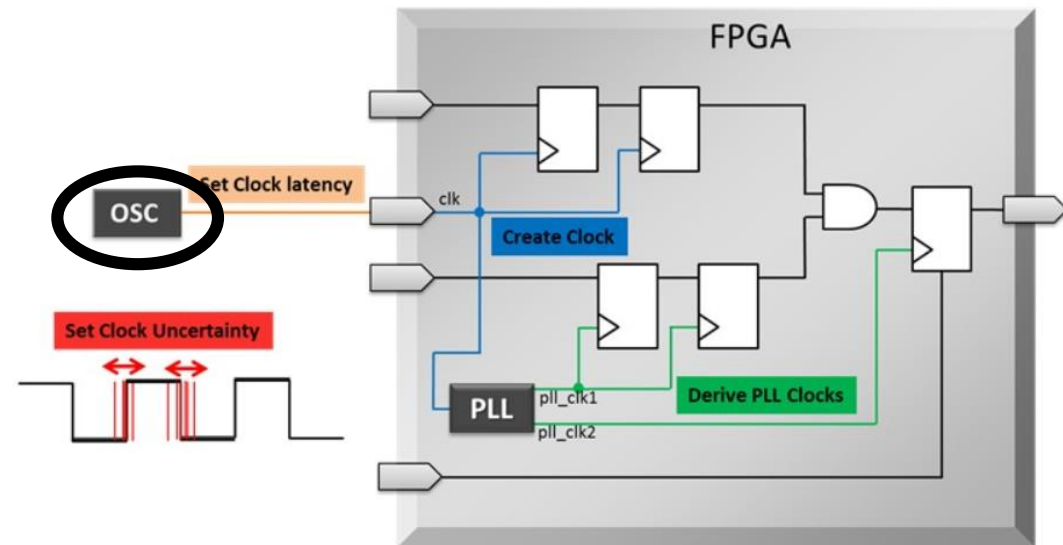
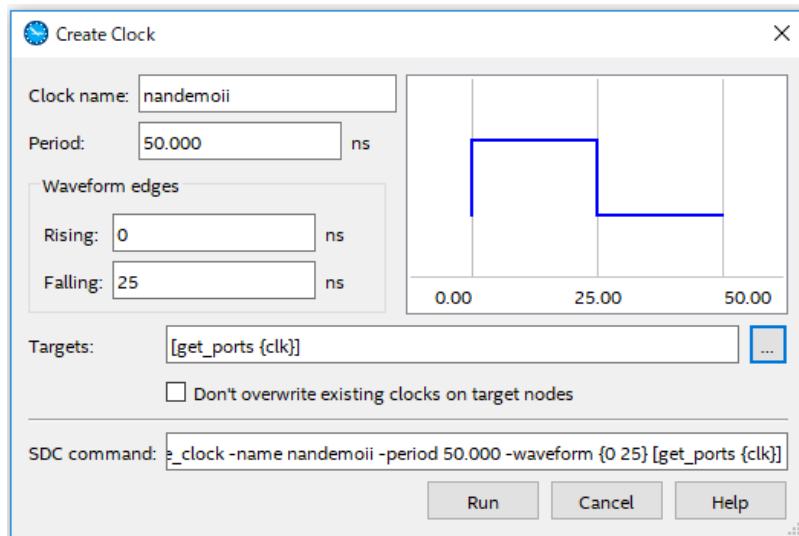
Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
0.000	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.223
-0.009	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.219
-0.070	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.198
-0.066	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.194
-0.063	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.191
-0.058	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.186
-0.056	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.184
-0.054	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.182
-0.041	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.169
-0.027	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.153
-0.011	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.146
0.000	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.129
0.021	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.121
0.022	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.107
0.036	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.105
0.038	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.092
0.042	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.086
0.048	data0[0]	multmult_1_atediff56 ck1	ck1	ck1	6.000	2.613	-4.076

また、Quartusはタイミング制約を基準に論理合成、配置配線で最適化してくれる

タイミング制約の設定(clock)

• Create clock

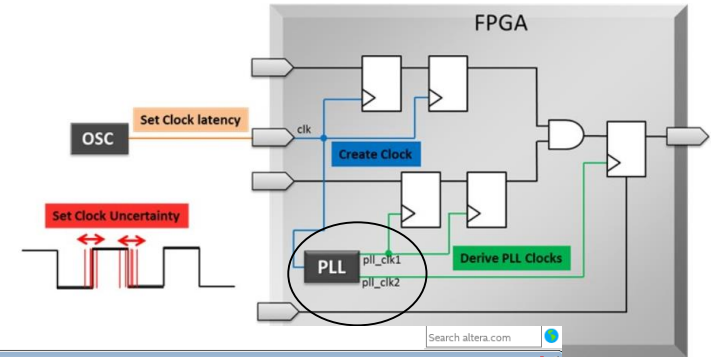
- FPGAのクロックポートに供給されるクロックの波形を定義
- これが目標動作周波数となる
- 20MHzの信号 → period = 50ns
- High/Lowの比(デューティー比) → 1:1でいいでしょう



タイミング制約の設定(clock)

• Create Generated Clock

- FPGA内部で生成される信号のどれがCLK信号の働きをするかをツールに指示
主にはPLLでclkを生成する時。PLLの出力信号もclk信号と設定
 - ✓ 周波数を1/2倍 → divide_by = 2
 - ✓ 周波数を5倍 → multiply_by = 5
 - ✓ 反転 → invertに✓
 - ✓ 位相シフトx度 → pahse = x度
- なおDerived pll clocksで一括定義できる



Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle	Divide by	Multiply by	Phase	Offset	Edge List	Edge 1
clk	Base	10.000	100.0 MHz	0.000	5.000							
pll_ojaltpll_component[auto_generated]pll1[clk[0]]	Generated	2.000	500.0 MHz	0.000	1.000	50.00	1	5				

pllクロックが自動的に設定された!

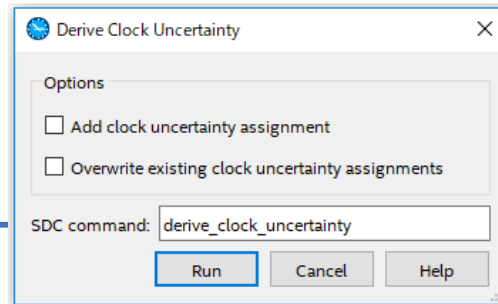
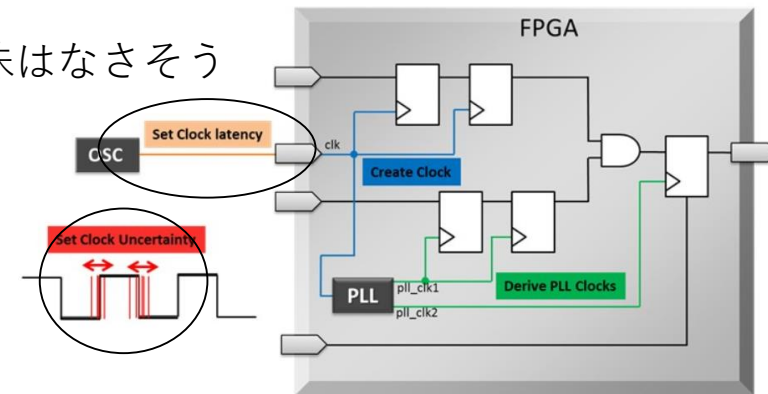
タイミング制約の設定(clock)

- Set Clock Latency

- clk生成部からFPGAのclk入力ポートまでの遅延を定義
- Rise/Fall → Clkの立ち上がり/立下り
- Late/early → 最大遅延/最小遅延
- 他のチップとの同期とか無い場合は、特に設定する意味はなさそう

- Set Clock Uncertainty

- clk信号のばらつきを定義
- とりあえずDerive clock uncertaintyで自動定義しとく
 - ツールに埋め込まれたばらつきモデルが使われる
 - Alteraはこっちを推奨している



```
#####  
# Set Clock Uncertainty  
#####  
set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks {clk}] 0.020  
set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks {clk}] 0.020  
set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks {clk}] 0.020  
set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks {clk}] 0.020
```


タイミング制約の設定(clock)

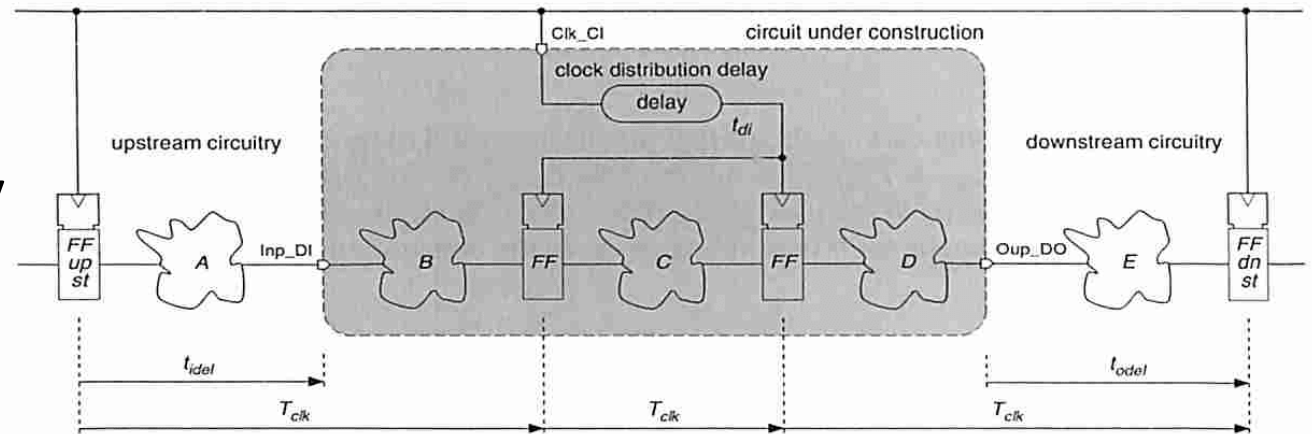
- Set clock groups
 - 回路上に同時に存在しないclk信号は、グループ分けして解析から外す
 - 複数クロックを用いたデジタル回路設計を多相クロック設計という
 - 検証が複雑になるので多用すべきでない
 - 動作高速化のためにPLLで高速なクロックを1, 2個作る程度に留めておく
 - 今後記載予定

タイミング制約の設定(clock)

- Virtual clock
 - 実際にFPGAに入力されるわけではないclk信号を定義
 - 同期する入出力遅延の定義などで使う
 - 今回の実験は、同期入出力ないので定義しない

タイミング制約の設定(clock以外)

- Set input delay
- Set output delay



- FPGA内のFFと外部デバイスのFFとの接続関係においてタイミング解析を実行するための設定(上図B,DにつながるFFのタイミング解析とB,Dの最適化)
 - clock → 外部FFに入力されるclk信号を設定
 - Input delay value → clkの立ち上(下)がりからFPGA入力ポートまでの遅延(上図A)
 - Output delay value → FPGA出力ポートから外部FFまでの遅延を定義(上図E)
 - なお、今回の実習は外部デバイスのFFとの接続はないので
 - ClkはFPGAに入力されるclkを設定
 - delay=0と適当に設定

タイミング制約の設定(clock以外)

- Set multicycle path
 - 複数サイクルでFF間を伝わるパスの設定
 - お勧めしない設計方法

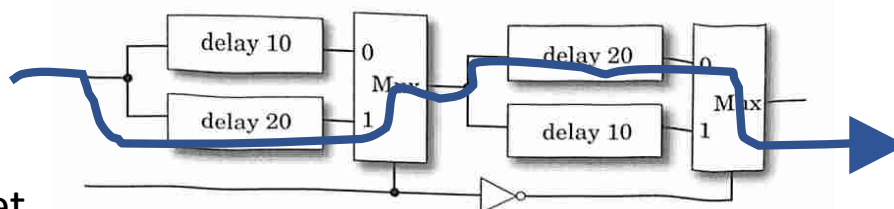
タイミング制約の設定(例外パス)

- Set max/min delay
 - 直接パスに絶対の最大/最小遅延を設定する
 - 組み合わせ回路の入出力遅延を制約したいとき
 - 非同期なclkが入るFF間のデータ接続に対する制約など
 - 今後記載予定

タイミング制約の設定(例外パス)

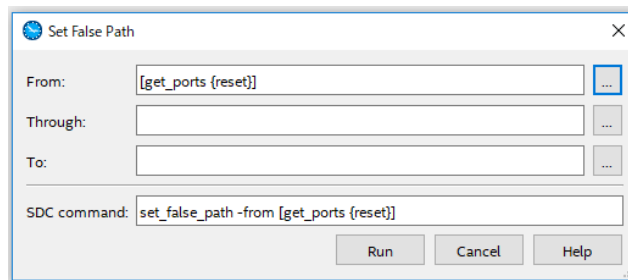
- Set false path

- タイミング解析しなくていいパスをツールに指示
 - 論理回路の構成上、活性化されることのないパス



- 非同期reset

- メタスタビリティの問題があるので本当はタイミングを考えないといけない
- 今後記載予定



- 最適化の必要ない一時的に作っただけで後で消す予定の入出力ポートとか

タイミング制約の設定

- ここまでで、全てのUnconstrained pathsが消えるはず

```
#####
# Time Information
#####
set_time_format -unit ns -decimal_places 3

#####
# Create Clock
#####
create_clock -name {clk} -period 10.000 -waveform { 0.000 5.000 } [get_ports { clk }]

#####
# Create Generated Clock
#####
create_generated_clock -name {pll_0|altpll_component|auto_generated|pll1|clk[0]} -source

#####
# Set Clock Uncertainty
#####
set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks {clk}] 0.020
set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks {clk}] 0.020
set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks {clk}] 0.020
set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks {clk}] 0.020

#####
# Set False Path
#####
set_false_path -from [get_ports {reset}]
set_false_path -from [get_ports {indata[0] indata[1] indata[2] indata[3] indata[4] indata[5] odata[0] odata[1] odata[2] odata[3] odata[4] odata[5] odata[6] odata[7] odata[8] odata[9]}]
```

Unconstrained Paths Summary

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	0	0
3	Unconstrained Input Ports	0	0
4	Unconstrained Input Port Paths	0	0
5	Unconstrained Output Ports	0	0
6	Unconstrained Output Port Paths	0	0

静的タイミング解析

- タイミング制約の設定後、Analyzerが静的タイミング解析を実行
 - Analyzerの中でやっていること
 1. 与えられたタイミング制約や設計のコンパイル結果から遅延を計算し、TimingNetlist(.sdo)として出力
 - 論理ゲートの内部遅延
 - 配線遅延
 - FPGA入出力遅延
 - ばらつき
 2. パスをいくつかの種類に分類する
 - Clock path
 - Data path
 - Asynchronous Clear Path
 3. それぞれのパスに対し解析を実行する
 - セットアップチェック
 - ホールドチェック
 - リカバリ及びリムーバルチェック
 4. その他特別な解析もしてくれる
今後記載予定

図 7-4. Quartus II TimeQuest タイミング・アナライザのタイミング・ネットリスト

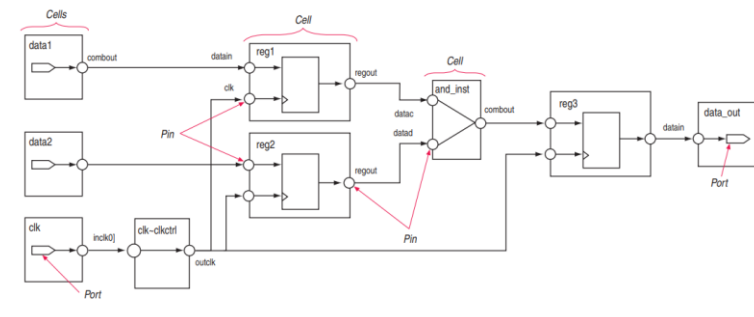
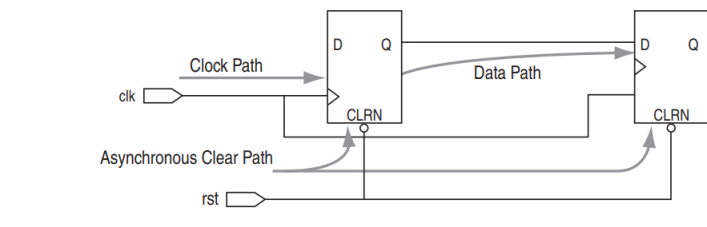
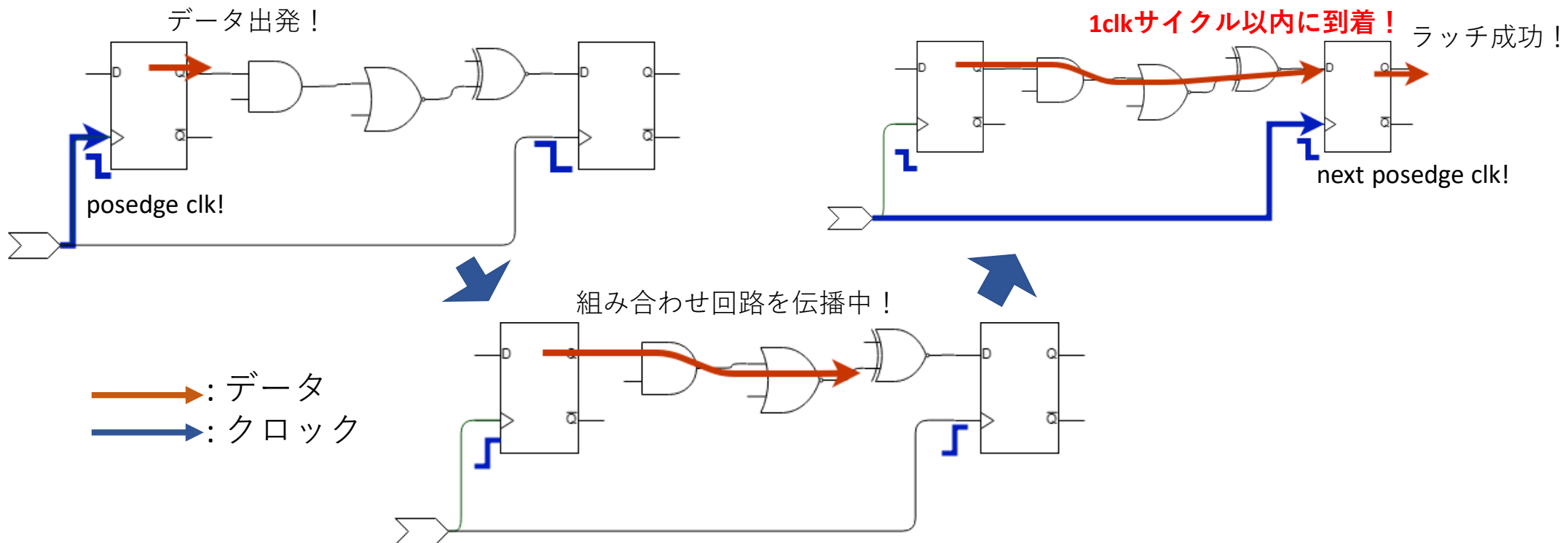


図 7-5. パスの種類



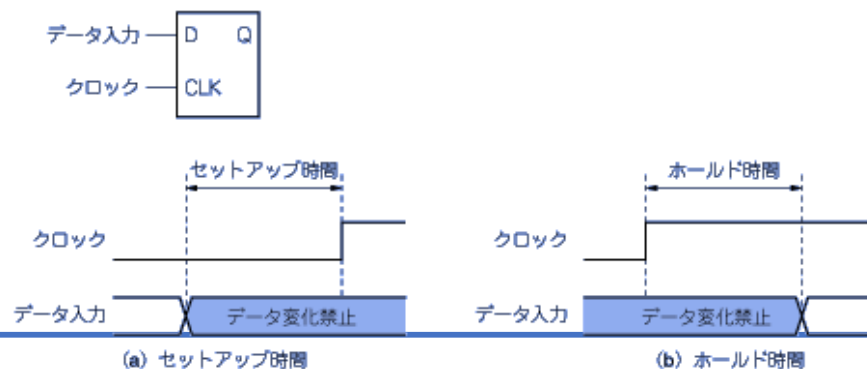
タイミング違反しない条件

- あるclkエッジでFFを出発したデータは、次のclkエッジで次段FFにラッチされればよい



タイミング違反しない条件

- さらに、FFが正常にラッチを行うために、clkエッジ前後のある程度の時間において入力信号は変化しないこと
 - この変化禁止の時間の長さはデバイス毎に決められている
 - Clkエッジより前の変化禁止時間：**Setup時間**
 - 入力データの到着がclkに対し**遅すぎるとSetup違反**につながる
 - Clkエッジより後の変化禁止時間：**Hold時間**
 - 入力データの到着がclkに対し**早すぎるとHold違反**につながる

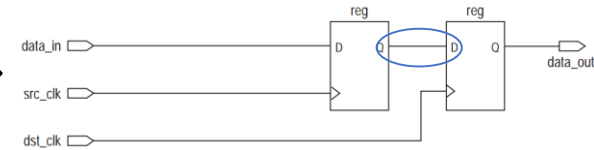


解析結果(Fmax, Setup)

- Fmax: 最大周波数

- 同じクロックで動くFF間のパスだけから判断される
- こういう異なるclkで動くFF間のパスは考慮から外される →

Slow 1200mV 100C Model				
	Fmax	Restricted Fmax	Clock Name	Note
1	117.38 MHz	117.38 MHz	pll_0[altpll_component]auto_generated[pll1]clk[0]	



- Setup: セットアップ制約

- Slack: どれくらい違反しているか
- From: パスの始点
- To: パスの終点
- Launch clk: 始点のFFに入るclk
- Latch clk: 終点のFFに入るclk
- Relationship: clk周期
- Data delay: 実際の遅延

FromからToまでの遅延が大きすぎるということ

Slow 1200mV 100C Model Setup: 'pll_0[altpll_component]auto_generated[pll1]clk[0]'								
	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-4.519	reg0[0]	reg1[15]_OTERM29	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
2	-4.519	reg0[0]	reg1[15]_OTERM27	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
3	-4.519	reg0[0]	reg1[15]_OTERM25	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
4	-4.519	reg0[0]	reg1[15]_OTERM23	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
5	-4.519	reg0[0]	reg1[15]_OTERM21	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
6	-4.519	reg0[0]	reg1[15]_OTERM19	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
7	-4.519	reg0[0]	reg1[15]_OTERM17	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
8	-4.519	reg0[0]	reg1[15]_OTERM15	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
9	-4.519	reg0[0]	reg1[15]_OTERM13	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
10	-4.519	reg0[0]	reg1[15]_OTERM11	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
11	-4.519	reg0[0]	reg1[15]_OTERM9	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
12	-4.519	reg0[0]	reg1[15]_OTERM7	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
13	-4.519	reg0[0]	reg1[15]_OTERM5	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
14	-4.519	reg0[0]	reg1[15]_OTERM3	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
15	-4.519	reg0[0]	reg1[15]_OTERM1	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
16	-4.519	reg0[1]	reg1[15]_OTERM29	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
17	-4.519	reg0[2]	reg1[15]_OTERM29	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496
18	-4.519	reg0[3]	reg1[15]_OTERM29	pll_0[altpll...][pll1]clk[0]	pll_0[alt...][l1]clk[0]	4.000	0.182	8.496

解析結果(Hold)

- Hold: ホールド制約

- Setupと大体同じ見方
- Holdなので、RelationshipよりDelayが大きければよい
 - Holdの説明は前頁参照

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	1.592	reg0[0]	reg1[0]~_Duplicate_1	pll_0[altpl...pll1]clk[0]	pll_0[altpl...ll1]clk[0]	0.000	0.160	1.938
2	1.656	reg0[11]	reg1[15]_OTERM51	pll_0[altpl...pll1]clk[0]	pll_0[altpl...ll1]clk[0]	0.000	-0.038	1.416
3	1.669	reg0[13]	reg1[15]_OTERM55	pll_0[altpl...pll1]clk[0]	pll_0[altpl...ll1]clk[0]	0.000	-0.038	1.429
4	1.676	reg1[0]	odata[4]~reg0	pll_0[altpl...pll1]clk[0]	pll_0[altpl...ll1]clk[0]	0.000	0.219	2.081
5	1.676	reg1[15]_OTERM31	odata[4]~reg0	pll_0[altpl...pll1]clk[0]	pll_0[altpl...ll1]clk[0]	0.000	0.219	2.081
6	1.676	reg1[15]_OTERM33	odata[4]~reg0	pll_0[altpl...pll1]clk[0]	pll_0[altpl...ll1]clk[0]	0.000	0.219	2.081
7	1.676	reg1[15]_OTERM35	odata[4]~reg0	pll_0[altpl...pll1]clk[0]	pll_0[altpl...ll1]clk[0]	0.000	0.219	2.081

今後記載予定

- Removal:
- Recovery:
- Minimum Pulse Width:

解析結果(Worst-Case Path)

- タイミング違反が発生したとき、より詳細な情報を見ることもできる
 - Timing Analyzer → 見たいパスの数字のところを右クリック → Report Worst-Case Path
 - Data Arrival Path: データパス(FF間の組み合わせ回路を通るパス)の遅延時間が詳しく載っている

Timing Analyzer - C:/intel/FPGA_lite/17.1/1A/2019/test - test

Slack	From Node	To Node	Launch Clock
-92.278	indata[0]	reg0[0]	clk
-6.020	reg0[0]	reg1[15]_OTERM29	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM27	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM25	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM23	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM21	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM19	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM17	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM15	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM13	pll_0[altpll_component]auto_generated
-6.020	reg0[0]	reg1[15]_OTERM11	pll_0[altpll_component]auto_generated

Report Timing (Worst-Case Path)

Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
-6.020	reg0[0]	reg1[15]_OTERM11	pll_0[altpll_component]clk[0]	pll_0[altpll_component]clk[0]	4.000	-0.104	9.711

Path #1: Setup slack is -6.020 (VIOLATED)

Path Summary	Statistics	Data Path	Waveform	Extra Fitter Information
Data Arrival Path				
Total	Incr	RF	Type	Fanout
0.000	0.000			
10.286	10.286			
10.286	10.286	R		
19.997	9.711			

Path Summary	Statistics	Data Path	Waveform	Extra Fitter Information
Data Required Path				
Total	Incr	RF	Type	Fanout
4.000	4.000			
14.182	10.182			
8.791	4.791	R		
14.182	5.391			

Report Timing: Found 200 setup paths (200 violated). worst case slack is -6.020
 No fmax paths to report
 Report timing -setup -from {reg0[0]} -to {reg1[15]_OTERM11} -from_clock {pll_0[altpll_component]auto_generated}pll_0[altpll_component]clk[0]
 reg0[0] is interpreted as {get_keepers {reg0[0]}}.
 reg1[15]_OTERM11 is interpreted as {get_keepers {reg1[15]_OTERM11}}.
 Report Timing: Found 1 setup paths (1 violated). worst case slack is -6.020

解析結果(Locate Node)

- タイミング違反が発生したとき、そのパスがどのように合成され配置配線されたのか見ることもできる
 - Report Worst-Case Path ウィンドウ → パスの数字右クリック → Locate Path → どれか選ぶ
 - 視覚的に理解できる

Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay	
						4.000	-0.104	9.711

Location	Total	Incr	
launch edge	1	0.000	0.000
clock path	2	10.286	10.286
clock networ	1	10.286	10.286
data path	2	19.997	9.711

Location	Total	Incr		
	1	4.000	4.000	
	2	14.182	10.182	
	1	8.791	4.791	R
	2	14.182	5.391	



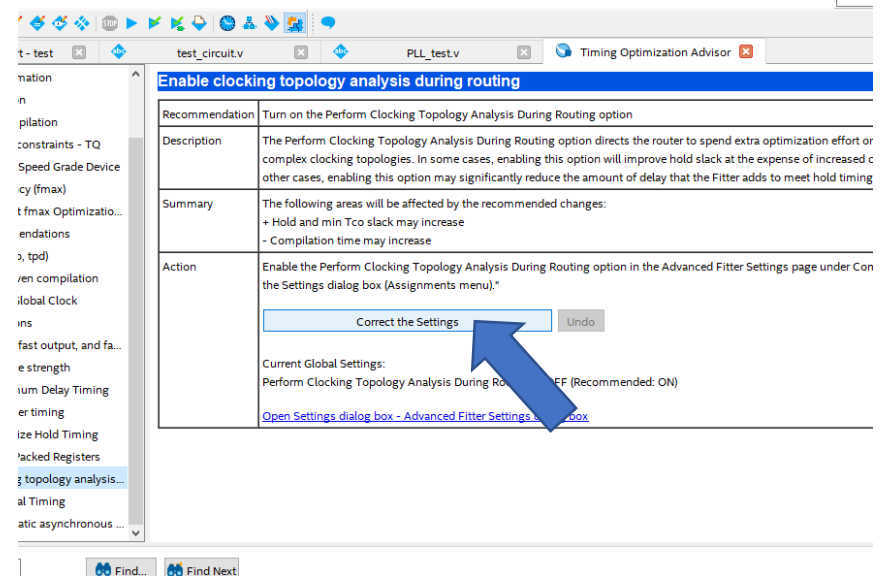
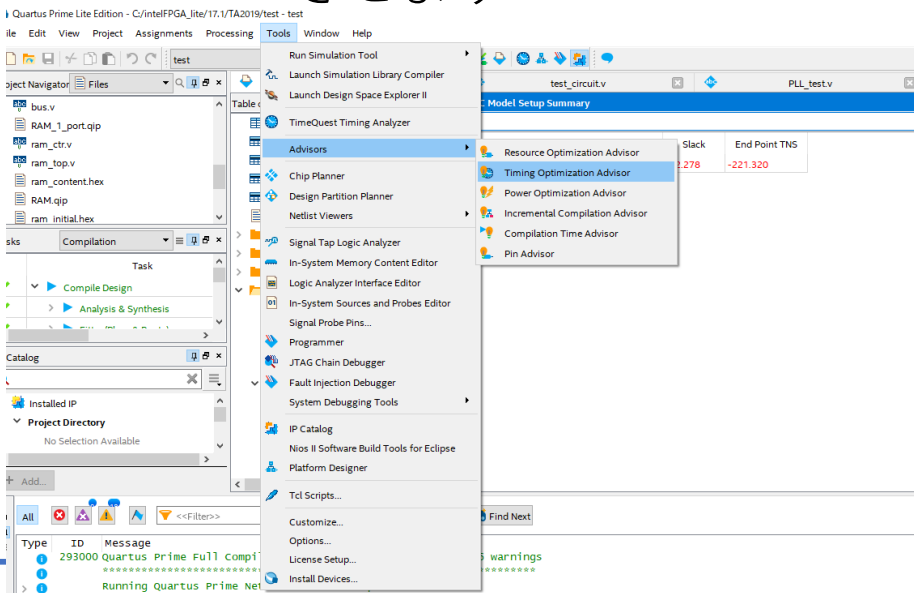
```
graph LR; reg0[reg0] --> mux[mult_sel_mux0_generated]; mux --> dsp[dsp_mult]; dsp --> reg1[reg1];
```

タイミング違反の解消

- 最適化オプション設定
 - 論理合成、配置配線での最適化を指示する
- その他解消方法は順次掲載予定

最適化オプション設定

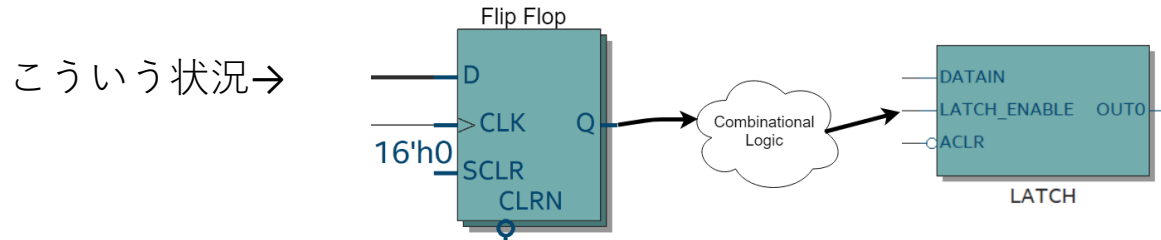
- ツールの指示に従えば、最適な設定ができる
 - QuartusのAdvisor機能を使う
 - Top → Tools → Advisor → Timing Optimization Advisor
 - 後は指示に従って設定を変えていくだけ。簡単。ただ最適化の限界はある
 - 実験3HWではツール設定に制限はないので、ツールにどんどん最適化させよう



Q&A

Q. クロックのタイミング制約を定義したのに、クロック制約のレポートで**Unconstraint**が消えない

A) FFの出力が、回路内のどこかのFFのCLK入力、もしくはLATCHのENABLE入りに接続されている可能性があります。



このような状況に陥っていないかRTL viewerで探してみましょう。(RTL viewerの検索欄で信号名を入れて検索)

LATCHを含む設計はお勧めしません。LATCHを使わない設計になるよう工夫しましょう。

Q. でも、RTLをコンパイルしたら勝手にLATCHが生成されてしまう。

A) `always @(*)` や `always @(posedge clk`以外の何かのwire変数) などが怪しい場合が多いです。

`always @(ココ)`には、

FPGAに外部から入力されるclk信号、PLLで生成したclk、reset、だけ書いて設計することを推奨

Q&A

Q. PLLでクロックを生成する、とは？

A. あるクロック信号をもとに、周波数を倍にしたり位相を変えたり色々アレンジして新しいクロックを生成してくれるのがPLL。自分でverilogを書いて所望のクロックを生成するより安全。

ex.) ボード上で標準で用意されている40MHzだと遅すぎるから、40MHzから200MHzクロックを新たに生成してそれを使いたい。

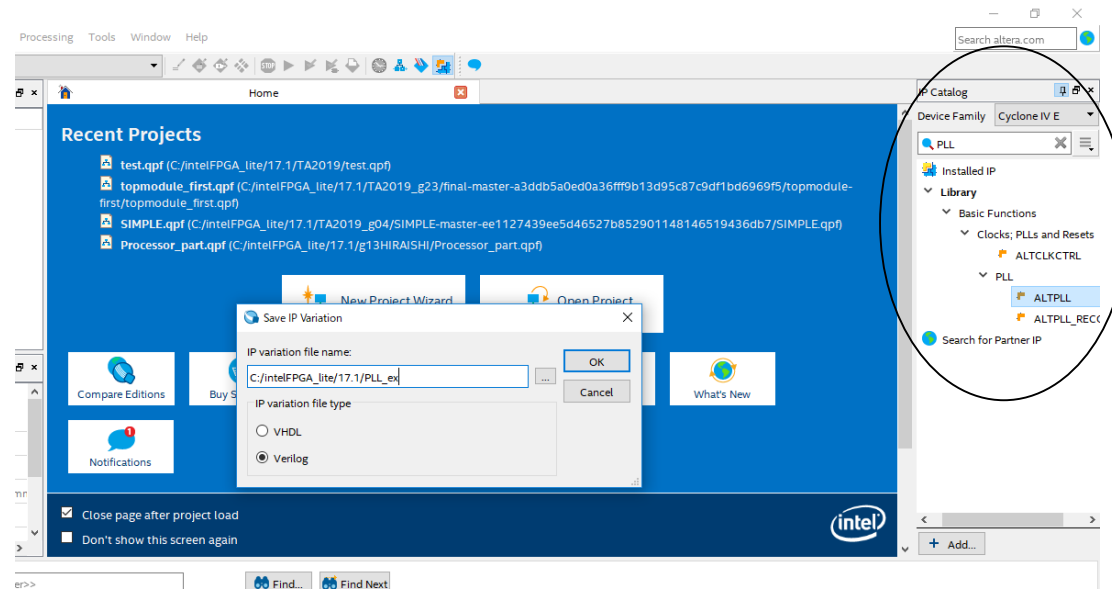
使い方

DeviceFamilyでPLLと検索。

ALTPLLを選ぶ

後は指示に従い設定していく

- Inclk0がもともになるクロック
- C0~c4が新たに生成されるクロック



参考文献

- Quartusのマニュアル
 - https://www.intel.co.jp/content/dam/altera-www/global/ja_JP/pdfs/literature/hb/qts/qts_qii53018_j.pdf
- 教科書
 - Hubert Kaeslin, “Top-Down Digital VLSI Design from Architectures to Gate-level Circuits and FPGAs”
 - 藤田昌宏, “システムLSI設計工学”

※わかりやすい文献、Webページなどあればスタッフに教えてください。

この資料に関する相談等の連絡先

スタッフ宛: le3ahw@lab3.kuis.kyoto-u.ac.jp