



タイミング & インプリメンテーション デザイン・デバッグ ガイドライン

2018年12月

株式会社マクニカ アルティマカンパニー

Rev.1

アジェンダ

- はじめに
 - 本資料の目的
 - チェックリスト
 - チェックポイント
- チェックフロー
 - タイミングで確認すべき項目
 - 回路設計で気をつけるポイント
 - クロック信号の確認
 - 電源電圧の確認
- 不具合事例



はじめに

本資料の目的

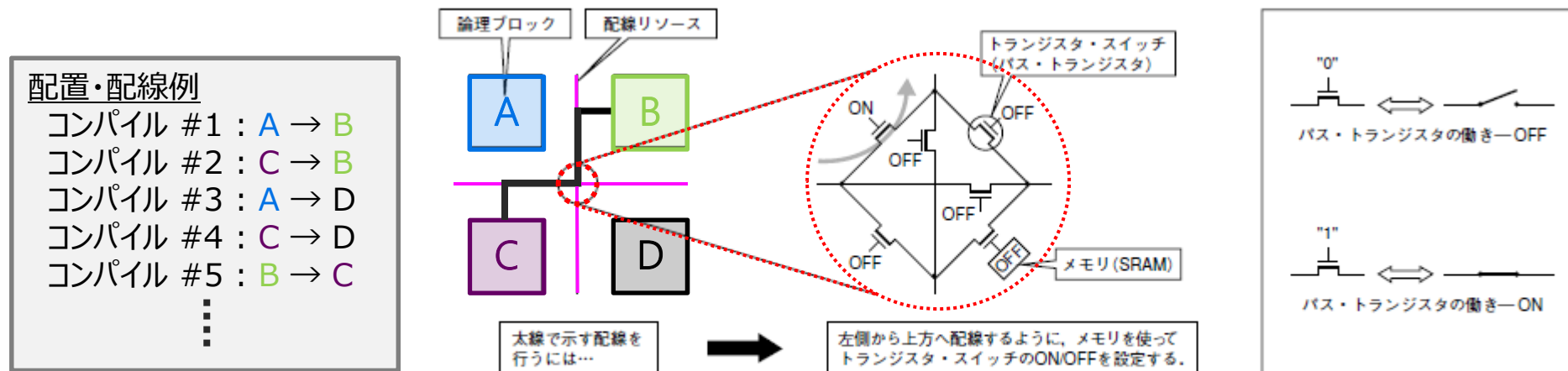
インテル® FPGA では、設計ソフトウェア（インテル® Quartus® Prime）上でのコンパイル（論理合成、配置、配線）工程を経て、FPGA 内の物理リソースを設定・接続するデータを作成し、そのコンパイルデータを FPGA 内にダウンロードすることで所要の機能をハードウェアとして実現します。

インテル® Quartus® Prime は、ユーザーの設定条件（動作周波数のようなタイミング制約等）を守るようにコンパイルを実行します。コンパイルでは、FPGA 内の固定された位置にある膨大な数のリソース（LUT、フリップフロップ(FF)、メモリー、DSP 等）を更に膨大な数の配線用の交差点トランジスタ・スイッチ（パス・トランジスタ）の On/Off によって接続します。

インテル® Quartus® Prime では、ユーザーデザイン（RTL や 回路図）が変更されない場合であっても、この配置や配線はコンパイル毎に変化します（場合によっては、論理合成結果も）。

また、半導体には、プロセス(P)、電源電圧(V)、温度(T) ばらつきがあり、この条件を加味したタイミングモデルを用いて、ユーザーが設定したタイミング制約に基づくタイミング解析が行われます。

正常な動作を保証するためには、適正な動作環境下で、適切に設定されたタイミング制約を守っている必要があります。



【図2】 トランジスタ・スイッチを使った配線経路の設定
FETのゲート電極に与える制御信号でオンオフを決める。制御信号はメモリ(SRAM)に蓄えてある。

引用 : Design Wave Magazine, 1999年12月号

本資料の目的 Cont.

一方で、実際に FPGA にコンパイルデータをダウンロードして動作させた場合、以下のような事象が発生する場合があります。

- ・ ユーザーデザインを変更していない場合であっても、**コンパイル毎に不具合動作・頻度・発生の有無が異なる**
- ・ 調査のために Signal Tap を挿入すると、不具合現象が再現しなくなる
- ・ FPGA の**個体**により不具合動作・頻度・発生の有無が異なる(P)
- ・ **電源電圧**を変化させると不具合動作・頻度・発生の有無が異なる(V)
- ・ **温度**が変化すると不具合動作・頻度・発生の有無が異なる（ある時間、動作させていると不具合動作が発生するようになる）(T)
- ・ ある時点から（ある**製品ロット**から）不具合動作が発生するようになる(P)
- ・ 対向デバイスを交換すると事象が変化する

このような現象は、経験的に約 80% の割合で、下記のような原因の複合で発生することが確認されています。

- ・ **適切にタイミング制約が設定されていない**
例：タイミング制約の漏れや上書き、False 設定による無効化
- ・ **信号間の到達時間差（どれかが早い、遅い、同時）に対して、その時間差を考慮した回路構成となっていない**
例：非同期リセット、非同期信号の分配
- ・ **タイミング制約およびタイミングモデルで規定した条件外での使用**
例：PVT 条件や入力ジッター条件を満たしていない、もしくはギリギリ

本資料では、タイミング制約の生成方法や不具合が発生しにくい回路構成を示すとともに、不具合発生時のデバッグ手順を示します。対象デバイスは インテル® Stratix® 10, Arria® 10, Cyclone® 10 をはじめとした、インテル® FPGA デバイスすべてを対象としています。

チェックリスト

◆ タイミング

- タイミング検証を行いタイミングエラーがないことを確認し解析結果 (*.rpt) をエビデンスとして残しているか
- 仕様通りの制約で SDC ファイルを作成したか
- False Path の制約を正しい箇所にものみ設定しているか
- 未制約のパス、無視されている制約はないか
- SDC ファイルが複数ある場合は、正しい順番で インテル® Quartus® Prime に登録を行ったか

◆ 回路

- グリッチのケアを行っているか
- 非同期クロック間の信号の受け渡しに組み合わせ回路を使用していないか
- 非同期となるレーシング回路の対策を行っているか
- インテル® Quartus® Prime でステートマシンの設定を “User Encoded” にしているか
- リセット信号のリセット解除タイミングを考慮しているか

◆ クロック

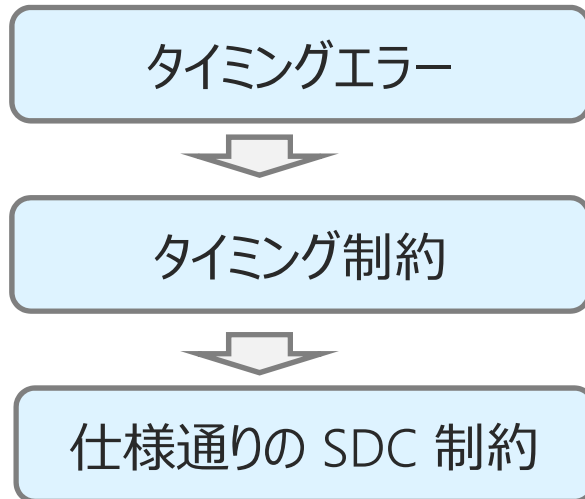
- 入力クロック信号の I/O 設定は正しいか
- FPGA コア・ファブリック側のクロック供給は外部から行っているか

◆ 電源

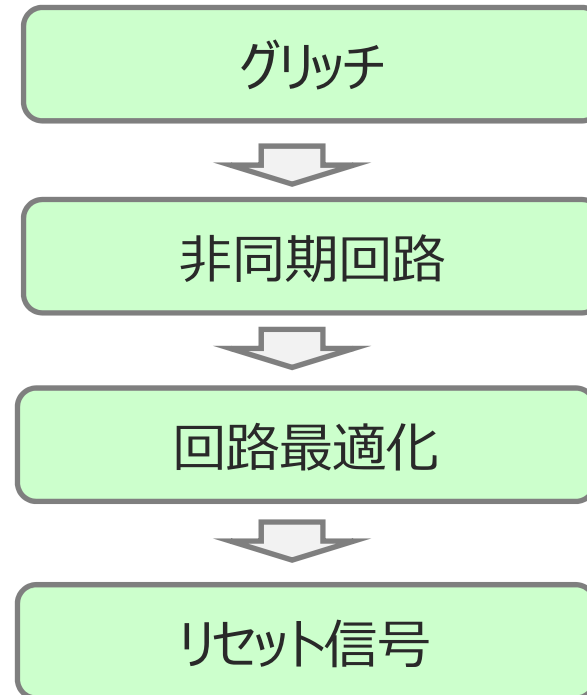
- 電源電圧はスペック内に収まっているか（過渡負荷時、同時スイッチング時）

チェックポイント

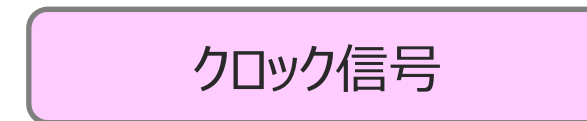
① タイミング関連の確認



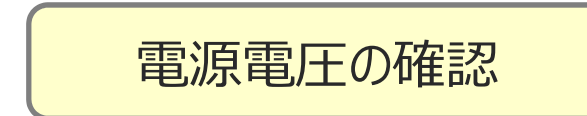
② デザインの確認



③ クロック信号の確認

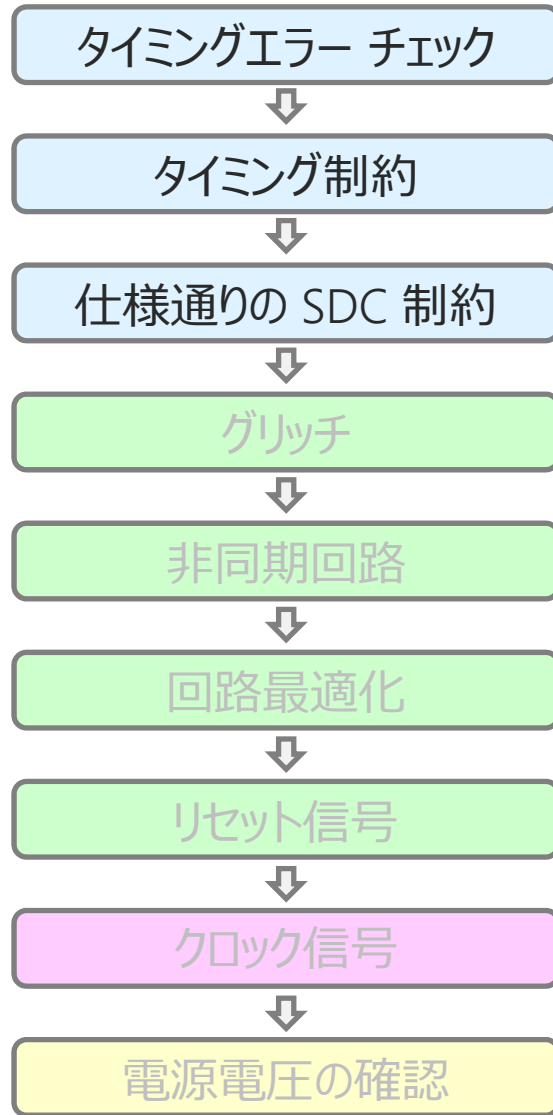


④ 電源電圧の確認



タイミングで確認すべき項目

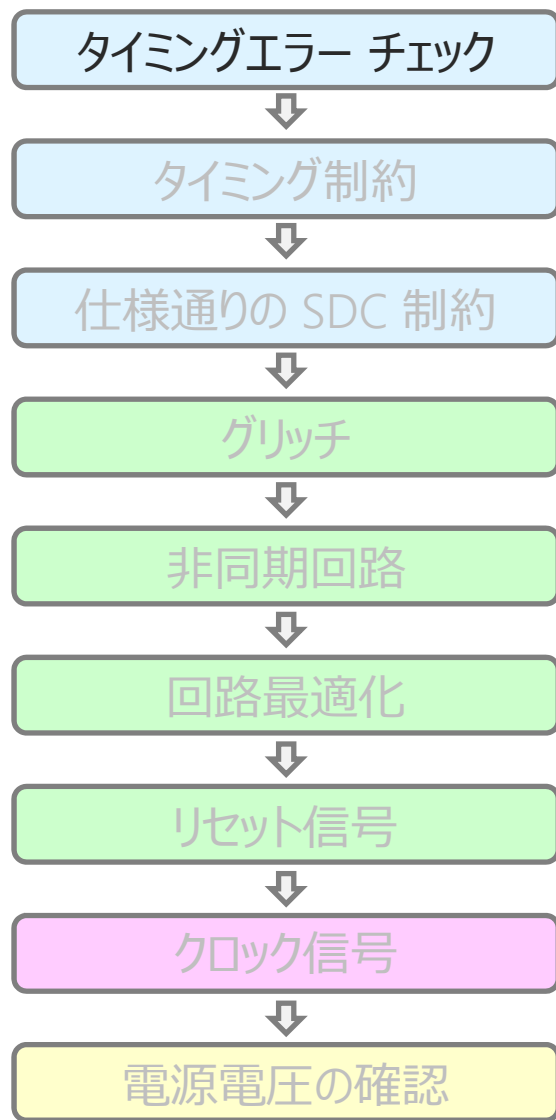
インプリメンテーション・ガイドライン – タイミング –



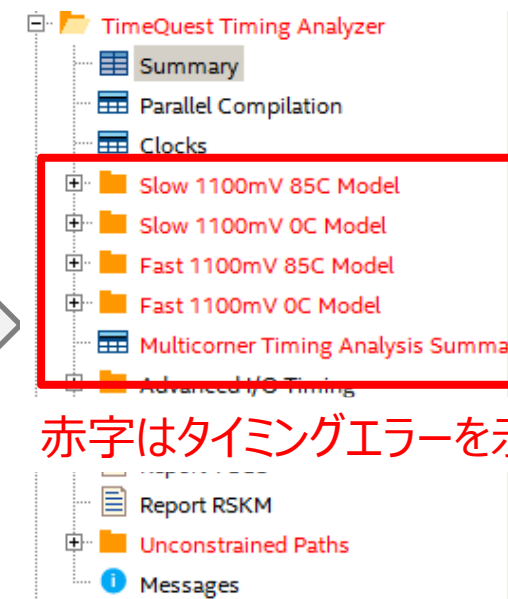
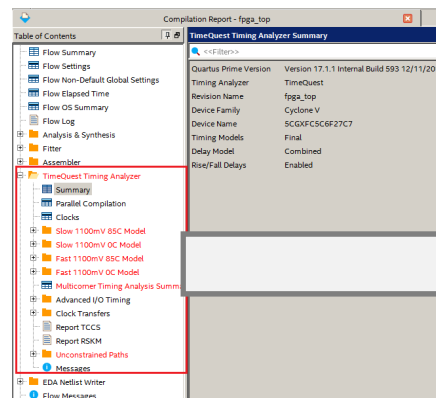
● タイミングでケアすること

- Timing Analyzer Summary のレポートで赤字がないか
 - インテル® Quartus® Prime のコンパイルレポートから確認
 - レポートを残しているか ([Appendix 1-1. 参照](#))
- タイミング・アナライザーでのレポート確認
 - 未制約のパス (Unconstrained Paths) はないか
 - 作成した SDC が反映されているか
 - 無視されている制約はないか
 - 必要な箇所にもみ False path の制約をかけているか
- クロック系統図は作成しているか
- 外部 AC スペックの確認は行ったか
- 仕様通りの SDC ファイルとなっているか
- SDC ファイルがインテル® Quartus® Prime のプロジェクトに登録されているか

タイミングエラー チェック



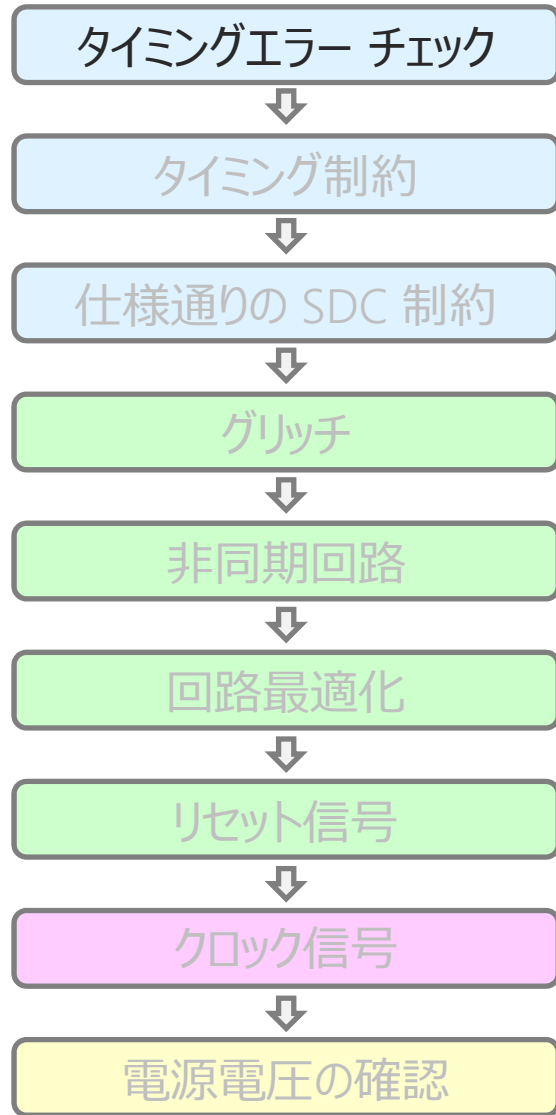
- インテル® Quartus® Prime でのコンパイル後、Timing Analyzer Summary レポートで赤字がないか確認



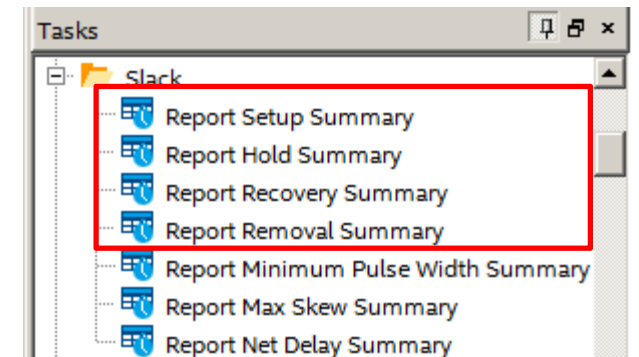
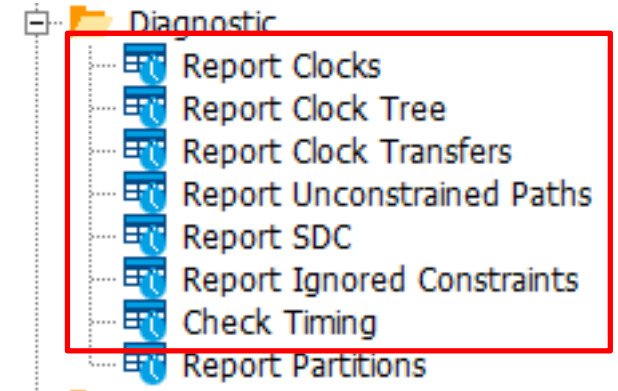
赤字はタイミングエラーを示す

- Timing Analyzer Summary レポートで赤字となっていた場合
 - タイミング収束を行い、赤字をなくす
 - Timing Optimization Advisor を活用 ([Appendix 7. 参照](#))
- タイミングを収束させても状況が改善しない場合
 - SDC の制約に誤りがないか、仕様通りの SDC を与えているか確認

タイミング・アナライザーでのレポート確認

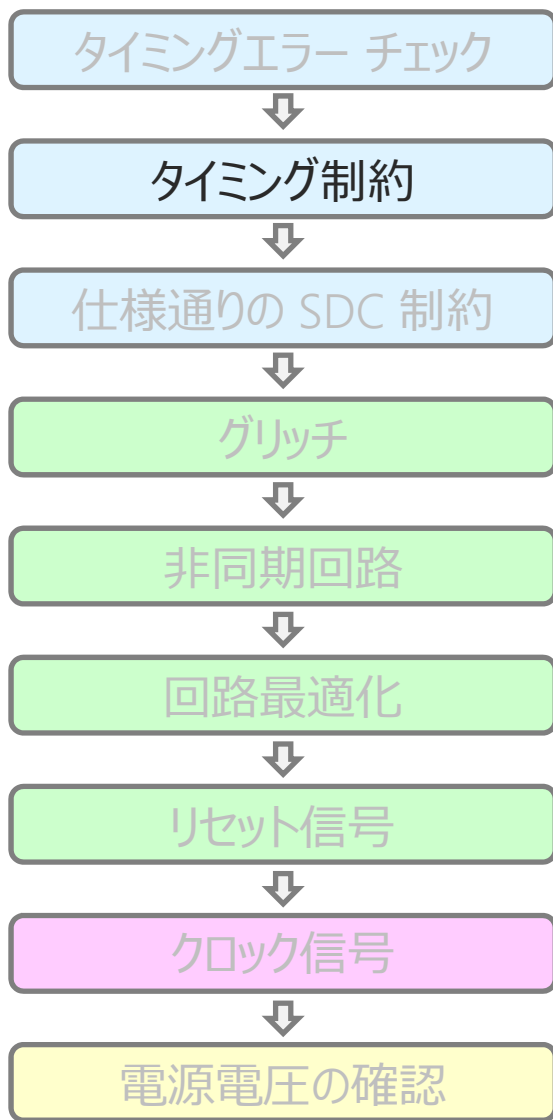


- 作成した SDC に問題がないかを確認
 - Report Clocks
 - 定義したクロック制約が正しく反映されているか
 - Report Clock Transfers
 - 解析対象外のクロック間パスがレポートされているか
 - Report Unconstrained Paths
 - 未制約のパスはないか
 - Report SDC
 - 作成した SDC は反映されているか
 - Report Ignored Constraints
 - 無視されている制約(コマンドはないか)
 - Check Timing
 - 制約の妥当性を確認
 - ・ 回路と制約の潜在的な問題を確認
- タイミング要求を満たしているか確認
 - Setup/Hold/Recovery/Removal
- 適切な箇所「のみ」に False Path 設定を行っているか ([Appendix 2-4. 参照](#))




※赤字で示したレポートは与えた SDC が妥当か必ず確認する項目

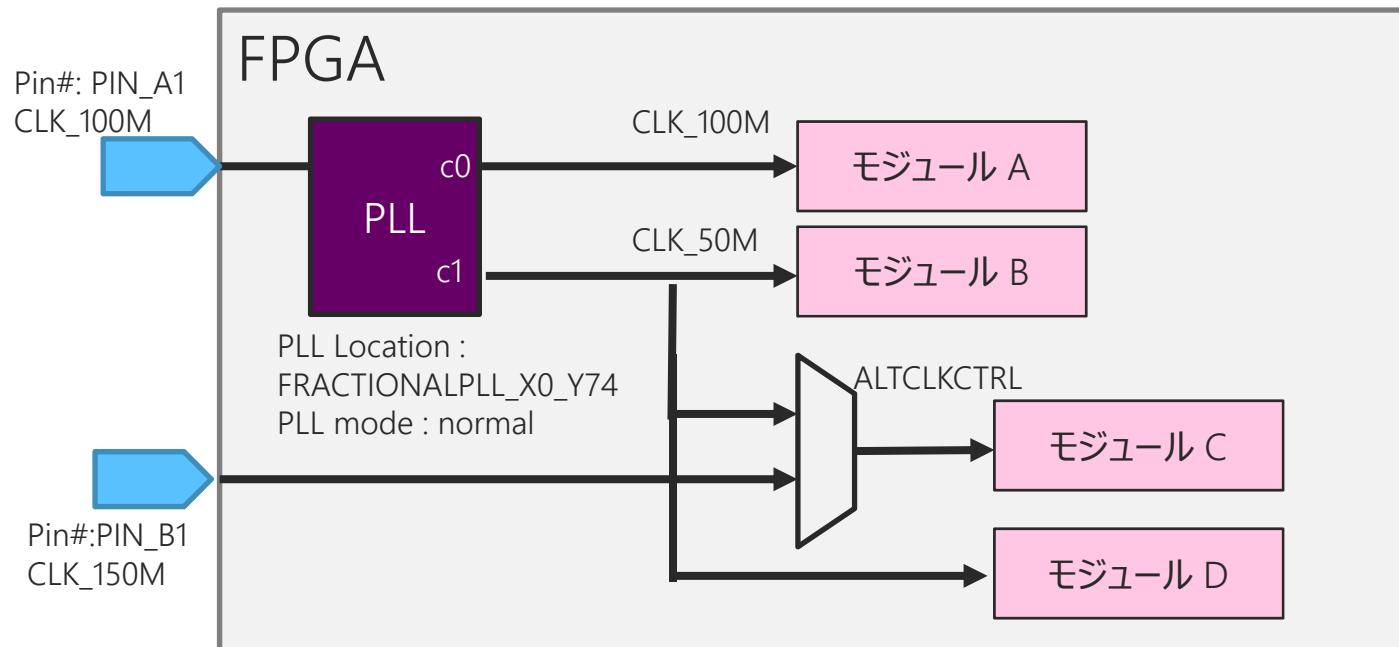
クロック信号系統図の確認：ブロック図での確認



- 設計者が FPGA 内部で使用しているクロックの種類を認識するためクロック信号系統図を作成

- クロック系統図作成例はこちらを参照 ⇒  クロック系統図例
- PLL Location, PLL mode 等は Fitter report で確認
 - [Appendix 5. 参照](#)

クロック系統図 例)



クロック信号系統図の確認：タイミング・アナライザー使用

タイミングエラー チェック



タイミング制約



仕様通りの SDC 制約



グリッチ



非同期回路



回路最適化



リセット信号

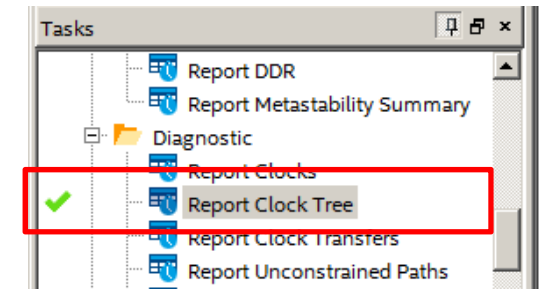


クロック信号



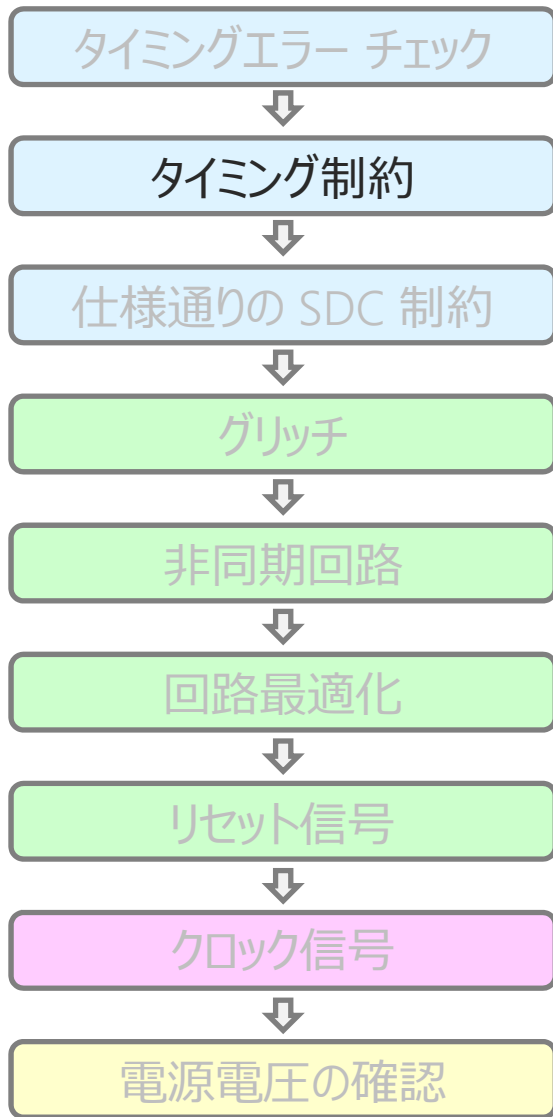
電源電圧の確認

- タイミング・アナライザーの Report Clock Tree を使用することで FPGA 内部のクロックツリーを確認
- タイミング・アナライザーのレポートと設計者が作成したクロックツリーに相違がない事を確認

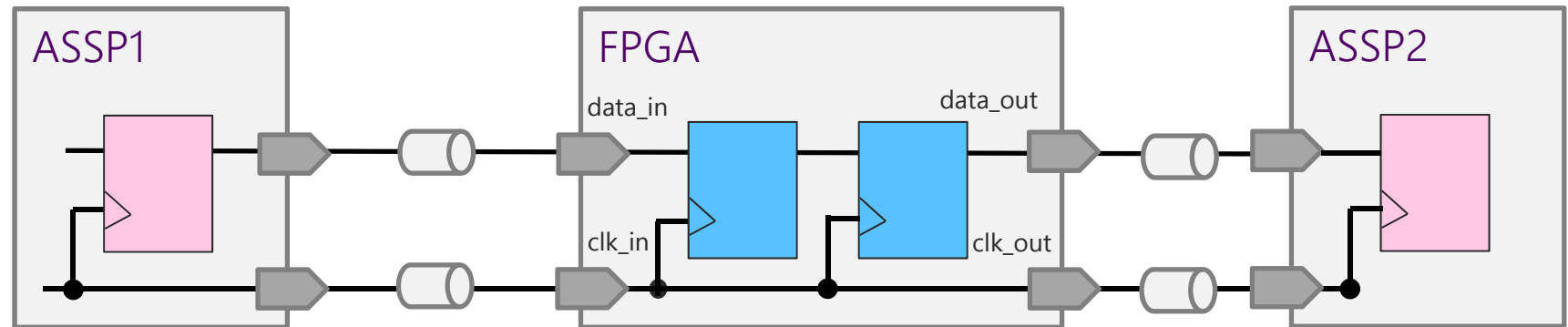


	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle	Divide by	Mul
1	CLK50M_PLL	Base	20.000	50.0 MHz	0.000	10.000			
1	PLL0 pll0_inst altera_pll_i general[0].gppll~FRACTIONAL_PLL vcoph[0]	Generated	2.000	500.0 MHz	0.000	1.000	50.00	2	20
1	PLL0 pll0_inst altera_pll_i general[0].gppll~PLL_OUTPUT_COUNTER divclk	Generated	10.000	100.0 MHz	0.000	5.000	50.00	5	1
1	c0_100m	Generated	10.000	100.0 MHz	0.000	5.000		1	1
1	selclk_c0	Generated	10.000	100.0 MHz	0.000	5.000		1	1
2	PLL0 pll0_inst altera_pll_i general[1].gppll~PLL_OUTPUT_COUNTER divclk	Generated	8.000	125.0 MHz	0.000	4.000	50.00	4	1
1	c1_125m	Generated	8.000	125.0 MHz	0.000	4.000		1	1
1	selclk_c1	Generated	8.000	125.0 MHz	0.000	4.000		1	1

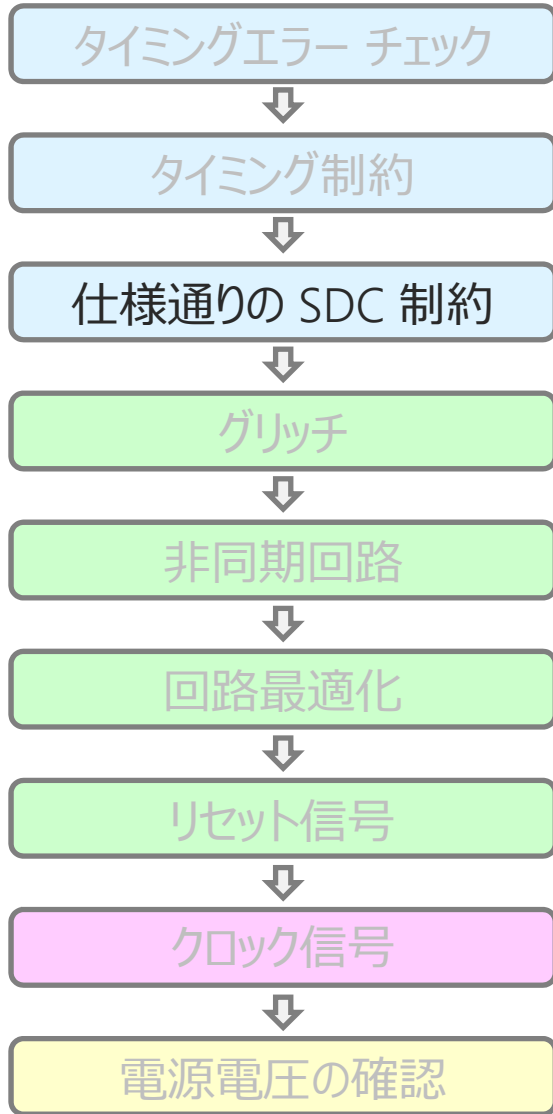
外部 AC スペックの確認



- FPGA 外部との AC スペックを定義するため、基板及び外部デバイスの情報が必要のため予め確認
 - 外部デバイスの Setup / Hold の値
 - 外部デバイスの Tco (min/max) の値
 - 外部デバイス ⇔ FPGA の PCB 配線遅延値
 - PCB クロックスキューの値
 - 入力クロックのジッター



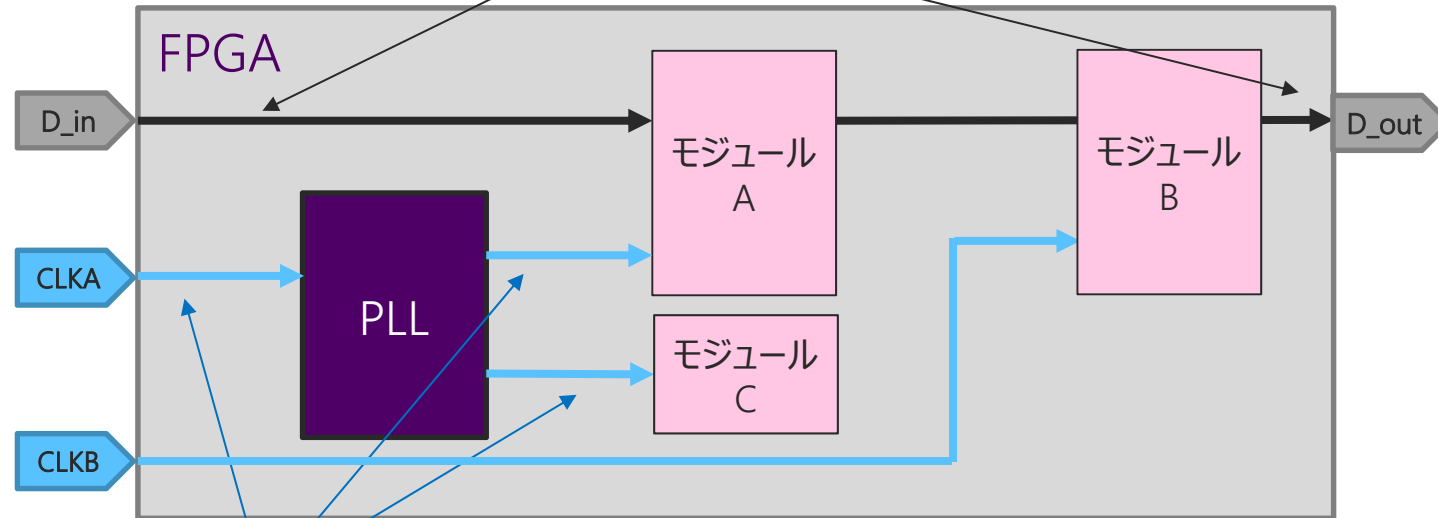
SDC 設定



- 各設定内容を確認

入力ピン、出力ピンの制約
([Appendix 2-5. 参照](#))

False path の制約 ([Appendix 2-3. 参照](#))
Multi-cycle path の制約 ([Appendix 2-6. 参照](#))



クロックの制約 ([Appendix 2-1. 参照](#))

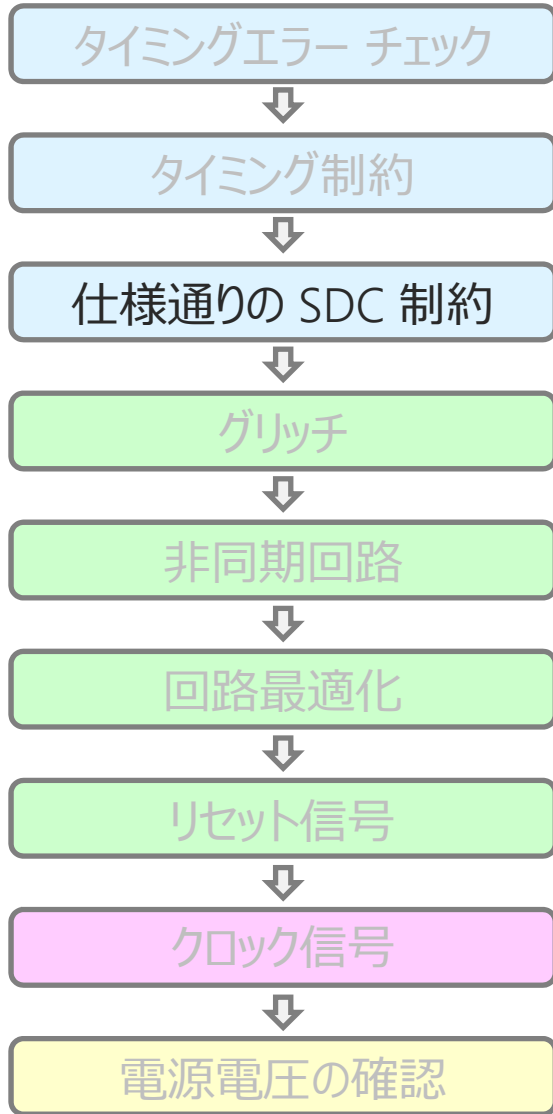
クロックグループの制約 ([Appendix 2-2. 参照](#))

FPGA 内部のクロックのバラツキの制約 ([Appendix 2-1. 参照](#))

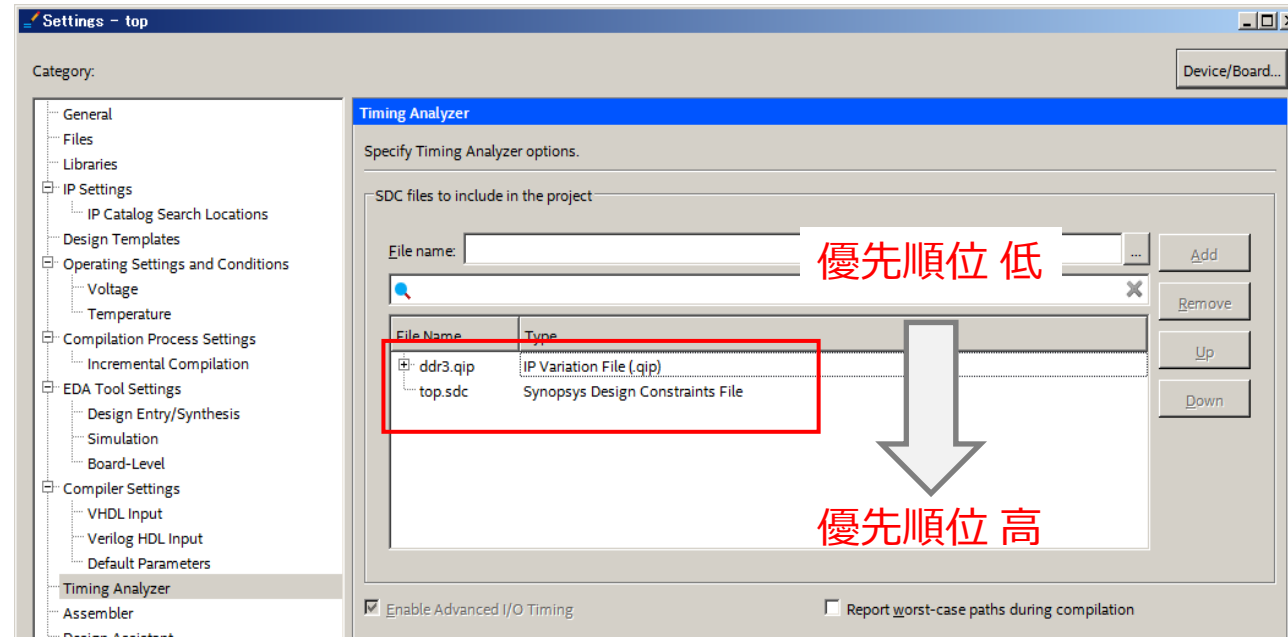
SDC を使ったタイミング制約

<http://monoist.atmarkit.co.jp/mn/articles/0811/28/news136.html>

SDC の登録



- インテル® Quartus® Prime のプロジェクトに SDC を登録
 - 作成した SDC が登録されているか確認
 - Assignments -> Settings -> Timing Analyzer
 - 複数の SDC を登録している場合、下に登録している SDC が優先順位が高い
 - ユーザーで作成した SDC は一番下に登録すること



回路で気をつけるポイント

回路で気をつけるポイント



- グリッチ
 - ゲートクロック
 - クロックセレクト
 - FPGA の入出力処理
- 非同期回路
 - 非同期の受け渡しに組み合わせ回路を使用
 - 非同期レーシングのケア
- 回路最適化
 - ステートマシン
- リセット信号
 - ユーザーモード時のリセットケア
 - Clock Crossing Bridge / FIFO のリセット
 - クロック (PLL 等) が安定するまで内部リセット

グリッチ

タイミングエラー チェック



タイミング制約



仕様通りの SDC 制約



グリッチ



非同期回路



回路最適化



リセット信号



クロック信号



電源電圧の確認

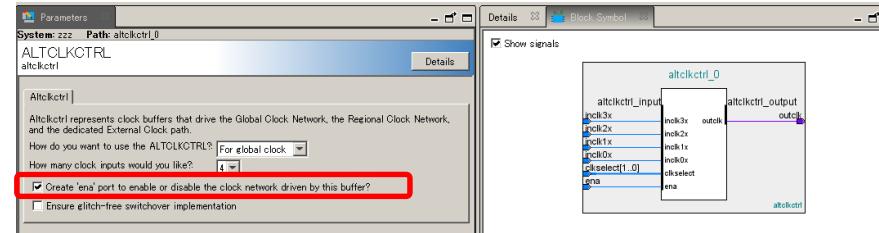
- ゲートクロック

- 不具合事象

- ゲートクロック の制御信号 (Enable) にグリッチが発生する

- 対策

- ALTCLKCTRL を使用 : ALTCLKCTRL の Create 'ena' port to enable or disable the clock network driven by this buffer を使用しグリッチを除去



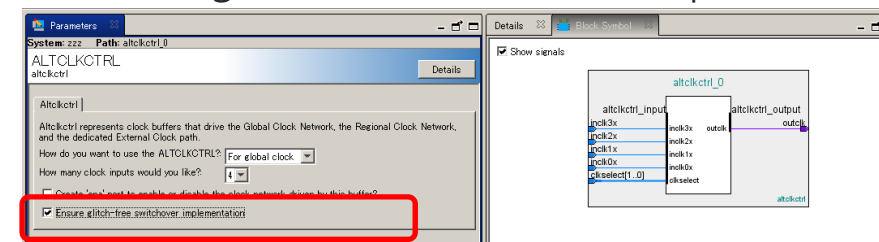
- クロックセレクト

- 不具合事象

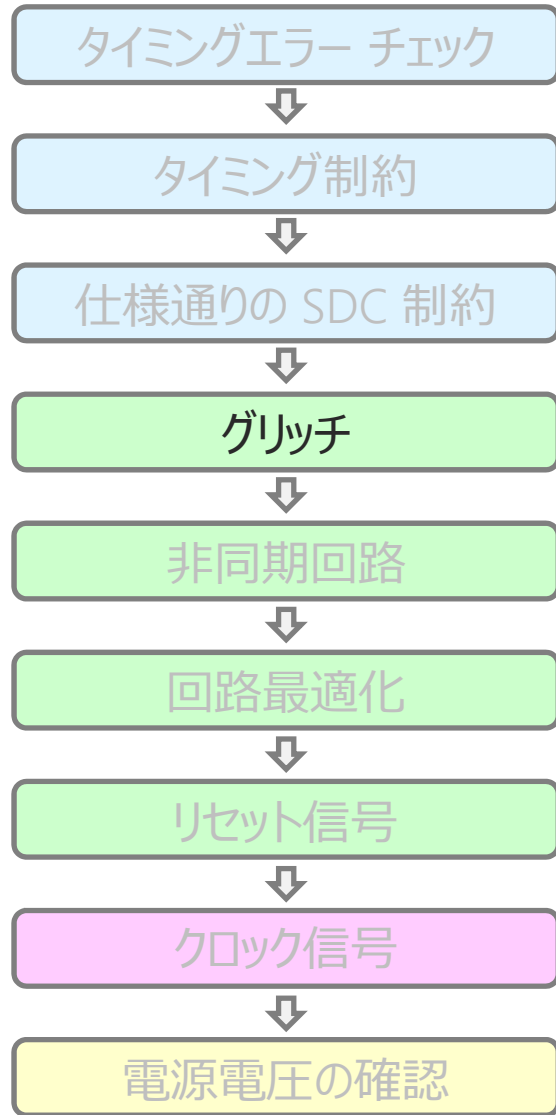
- クロックを排他的に使用する場合、クロックを切り替えた際にグリッチが発生する

- 対策

- ALTCLKCTRL で Ensure glitch-free switchover implementation を使用してグリッチを除去



グリッチ



● FPGA の入出力処理

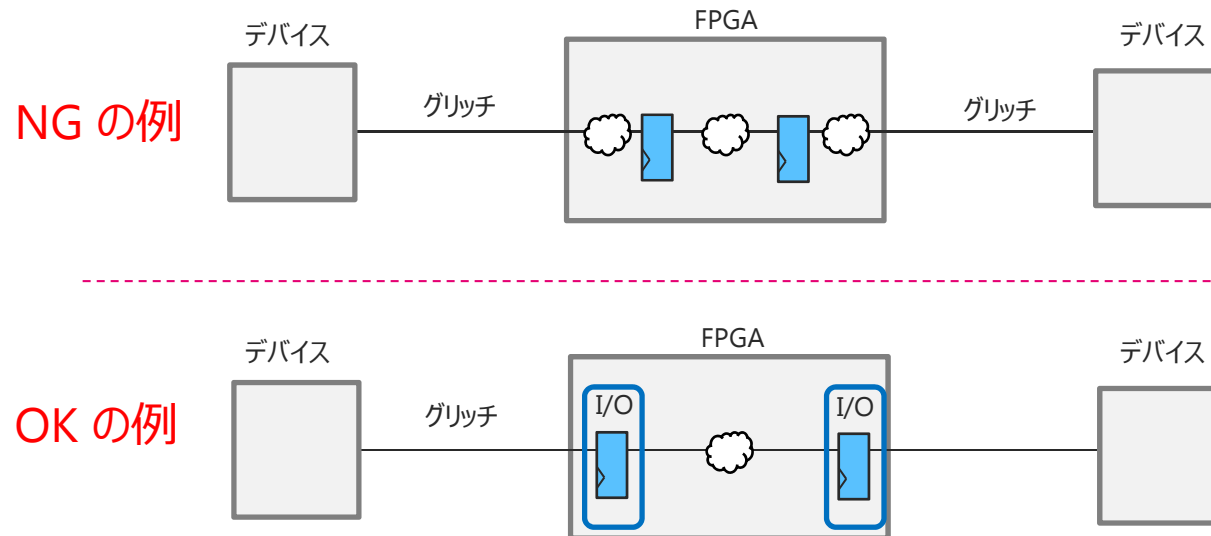
○ 不具合事象

- 外部デバイスからの信号を FPGA に取り込む・出力する際にグリッチが発生した信号を送受することがある

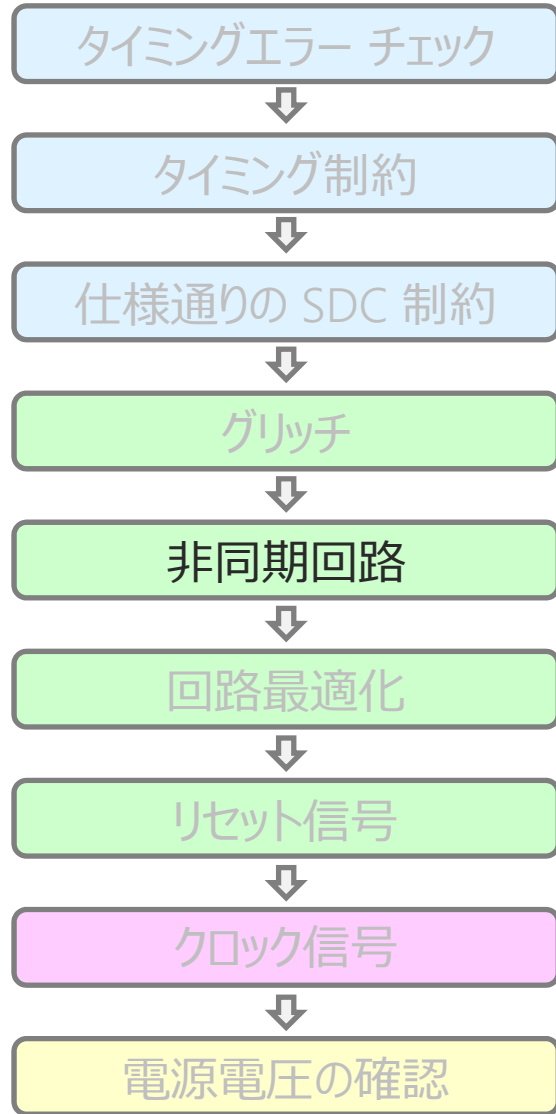
○ 対策

- FPGA の入力、出力は I/O 内の FF で叩いた信号を使用

- I/O 内の FF に割り当ててる方法は、Assignment Editor で Fast Input Register/Fast Output Register を割り当てて使用 ([Appendix 8. 参照](#))



非同期回路



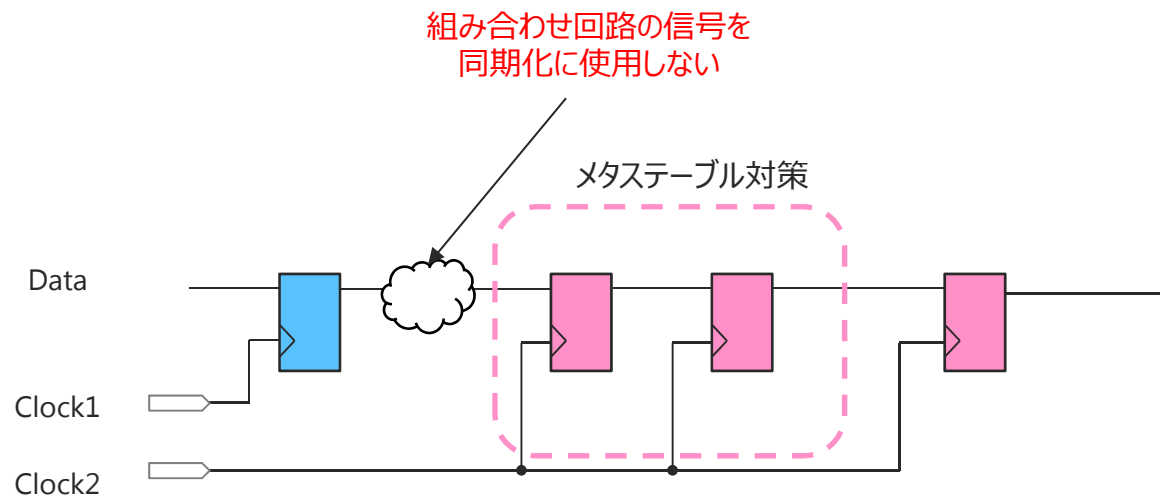
- 非同期の受け渡しに組み合わせ回路を使用

- 不具合事象

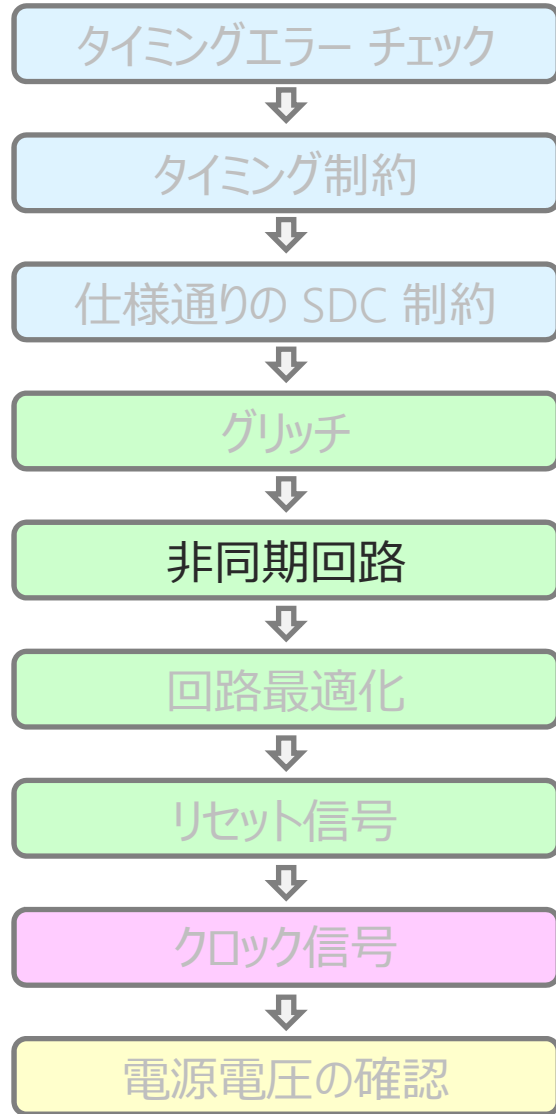
- 非同期の受け渡しに組み合わせ回路を使用するとグリッチが発生しデータ化けが発生する可能性あり

- 対策

- 非同期の受け渡しは、組み合わせ回路は使用しない
- 組み合わせ回路の出力信号（LUT の出力）には「必ず」グリッチが発生するものとして取り扱うこと



非同期回路



● 非同期レーシングのケア

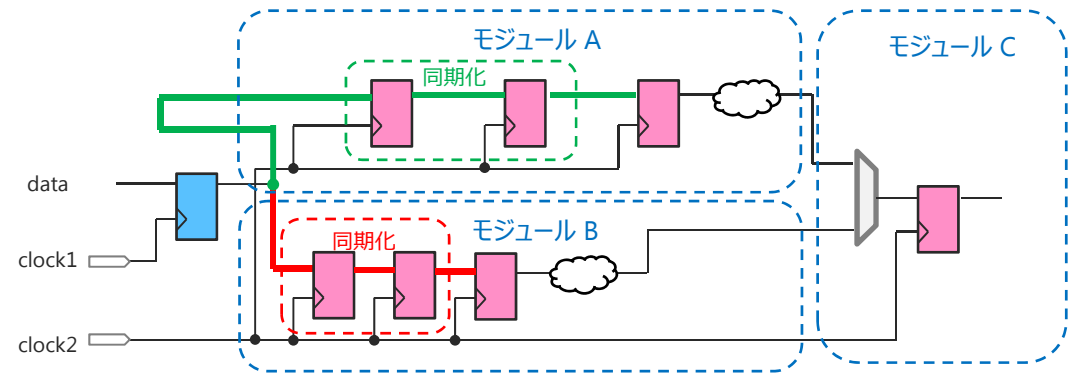
○ 不具合事象

- 1本の非同期信号を複数個所で同期化することにより、レーシング発生
- バスの伝送を FF で 2 段受けする場合は、1 クロック分データ化けが発生

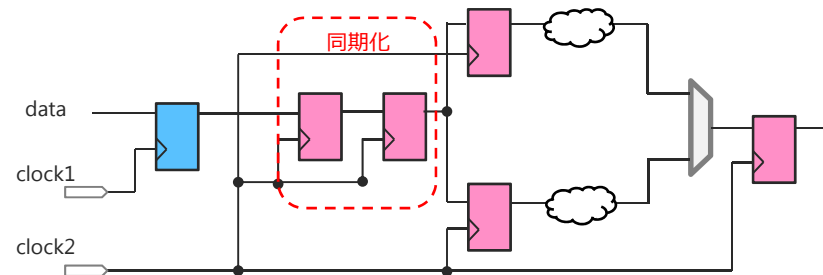
○ 対策

- レーシングを防ぐため、同期化してから分岐
- モジュールを跨って設計する際には注意が必要

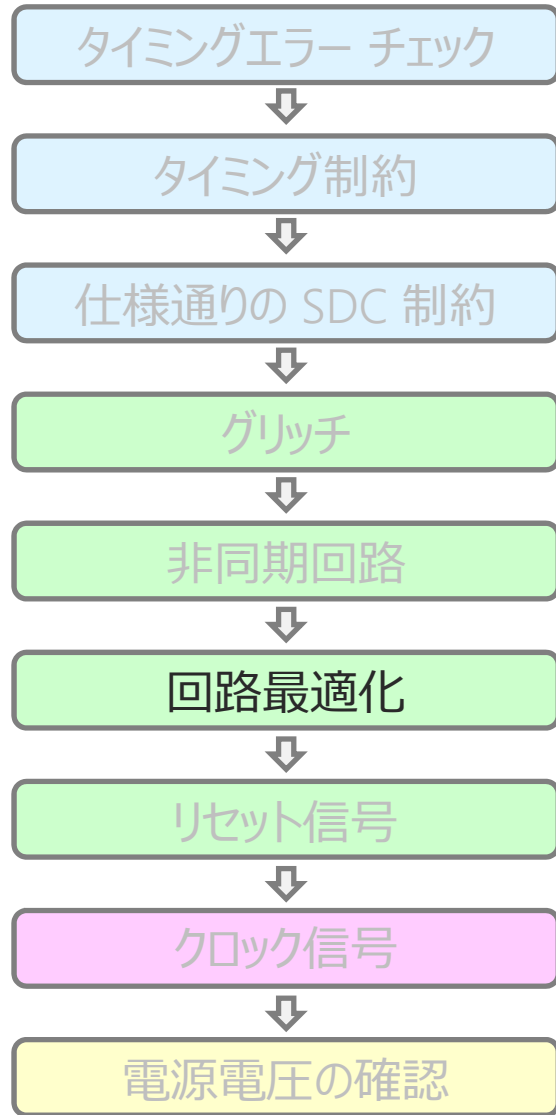
NG の例
分配後に同期化



OK の例
同期化してから分配



回路最適化



- ステートマシン

- 不具合事象

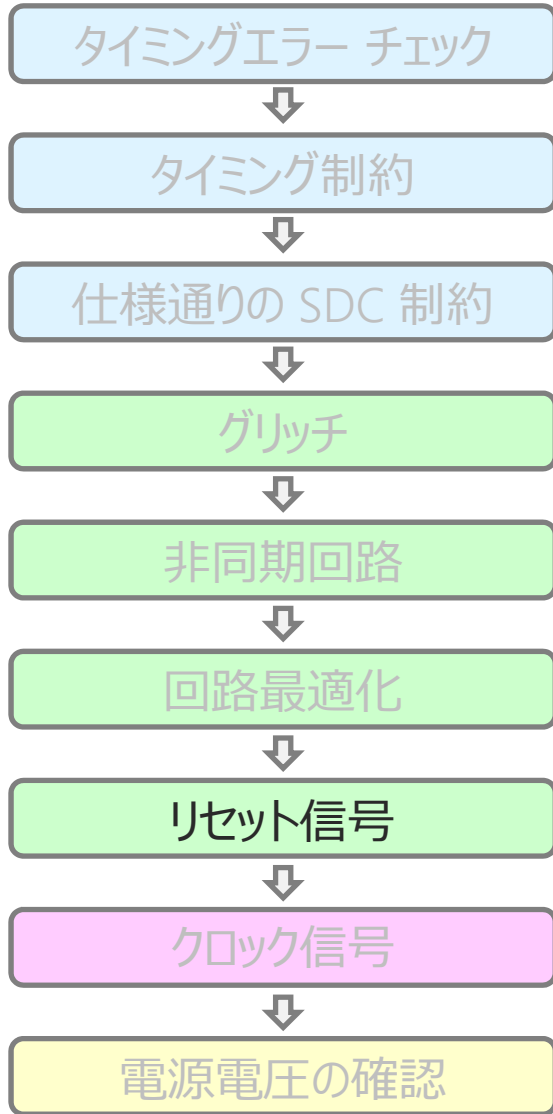
- コンパイルする際にイリーガルステートが最適化により実装されず、何らかの要因によりイリーガルステートに遷移した場合にデッドロックする可能性がある
- イリーガルステート記述
 - Verilog HDL : default
 - VHDL : others

- 対策

- インテル® Quartus® Prime の設定
⇒ "State Machine Processing" を "User Encoded" にして以下の設定を行う
 - イリーガルステートを記述している場合
= Safe State Machine = ON で設定 (イリーガルステートの最適化を防ぐため)
 - イリーガルステートを記述していない場合
= Safe State Machine = OFF で設定 (イリーガルステートが存在しないため)

設定方法については、[Appendix 9-5. ステートマシンの設定方法](#) 参照

リセット信号



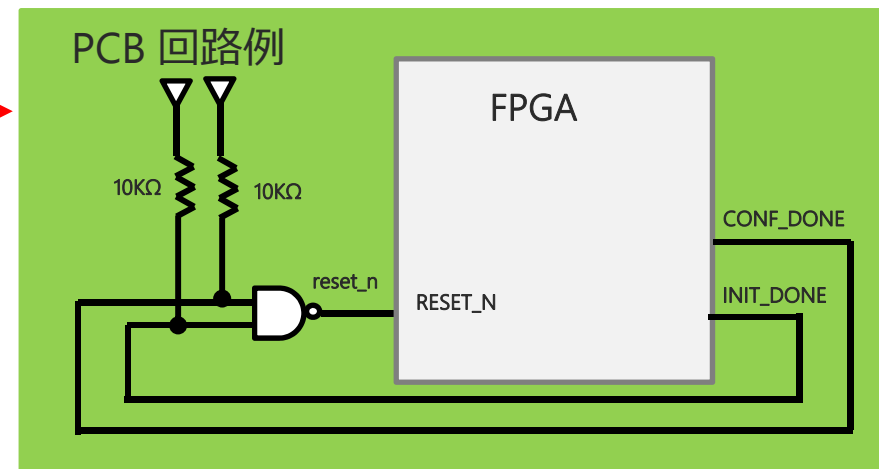
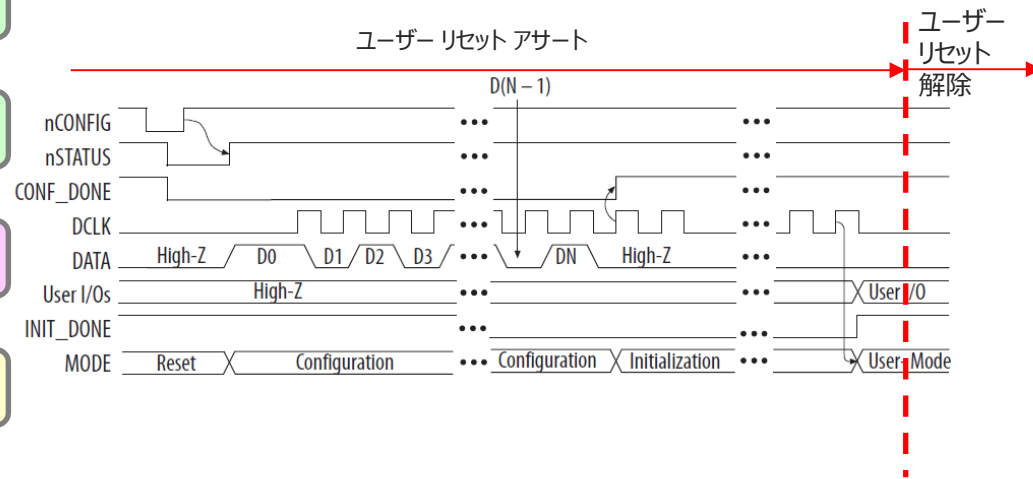
● ユーザーモード時のリセットケア

○ 不具合事象

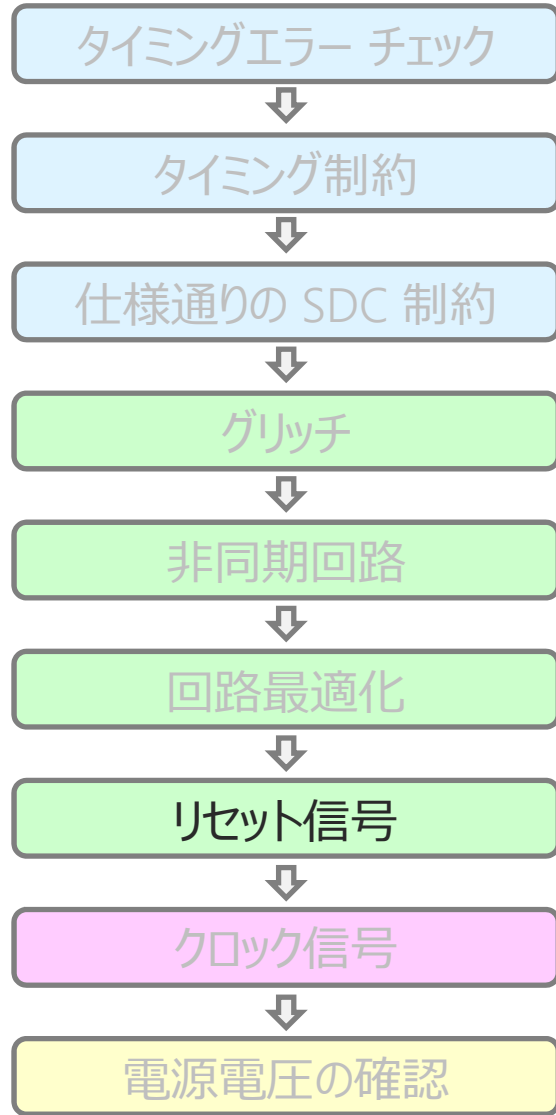
- コンフィグレーション開始からユーザーモードに至るまでリセットを入れておかないと FF の初期値が保証されない

○ 対策

- ユーザーモードに入るまでリセットを供給
 - https://www.intel.com/content/www/us/en/programmable/support/support-resources/knowledge-base/solutions/rd06112009_450.html
- CONF_DONE と INIT_DONE の AND したものをリセット信号として使用



リセット信号



- Clock Crossing Bridge / FIFO のリセット (Platform Designer 使用時)

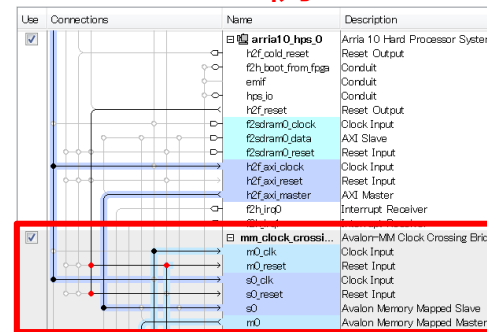
- 不具合事象

- マスターとスレーブのリセットが異なるために内部 FIFO のアドレスポインターのリセットタイミングが異なり誤動作

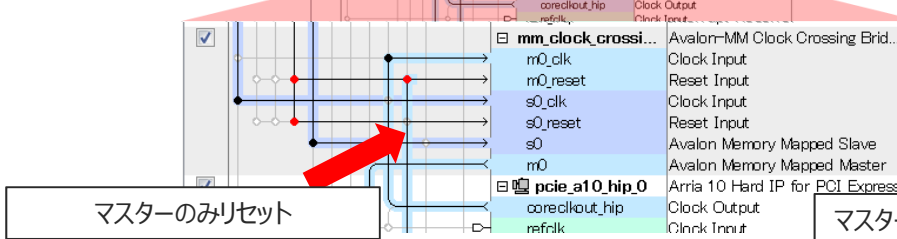
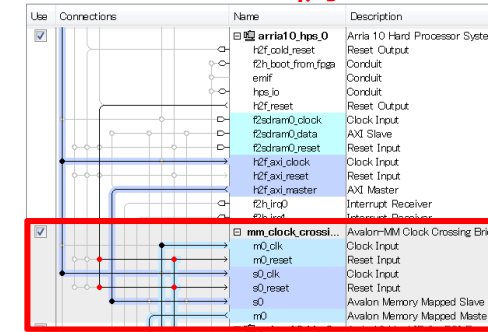
- 対策

- マスター側とスレーブ側のリセットは、同じ信号を使用してアサート
- https://www.intel.com/content/www/us/en/programmable/support/support-resources/knowledge-base/solutions/rd08042014_443.html

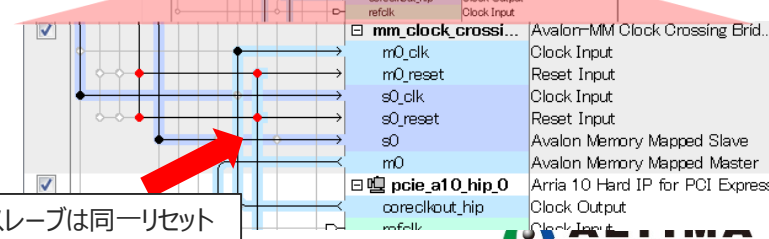
NG の例



OK の例

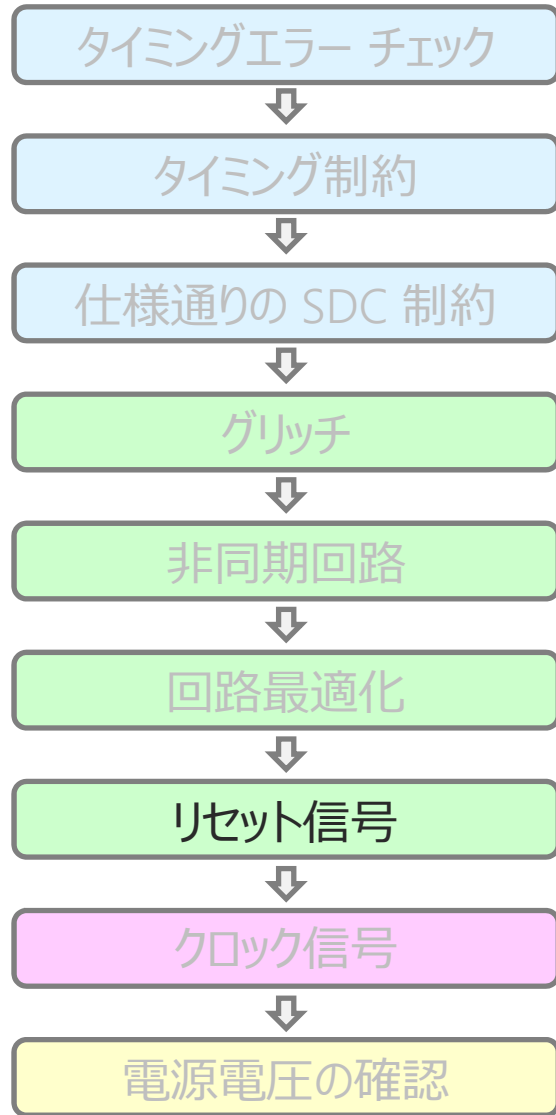


マスターのみリセット



マスターとスレーブは同一リセット

リセット信号



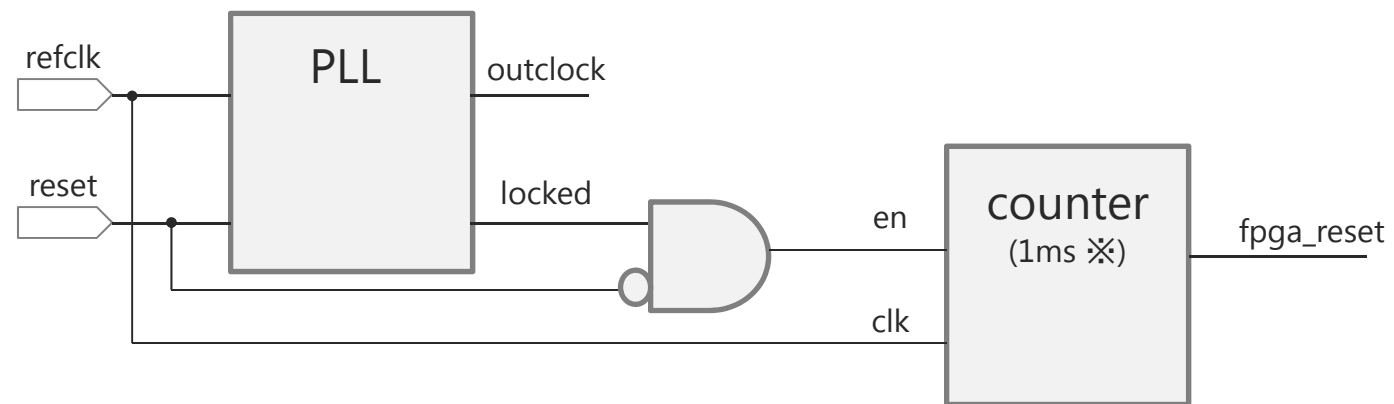
- クロック (PLL 等) が安定するまで内部リセット

- 不具合事象

- クロックが不安定な状態で論理を動作させてしまうと、意図しない挙動で回路が動作

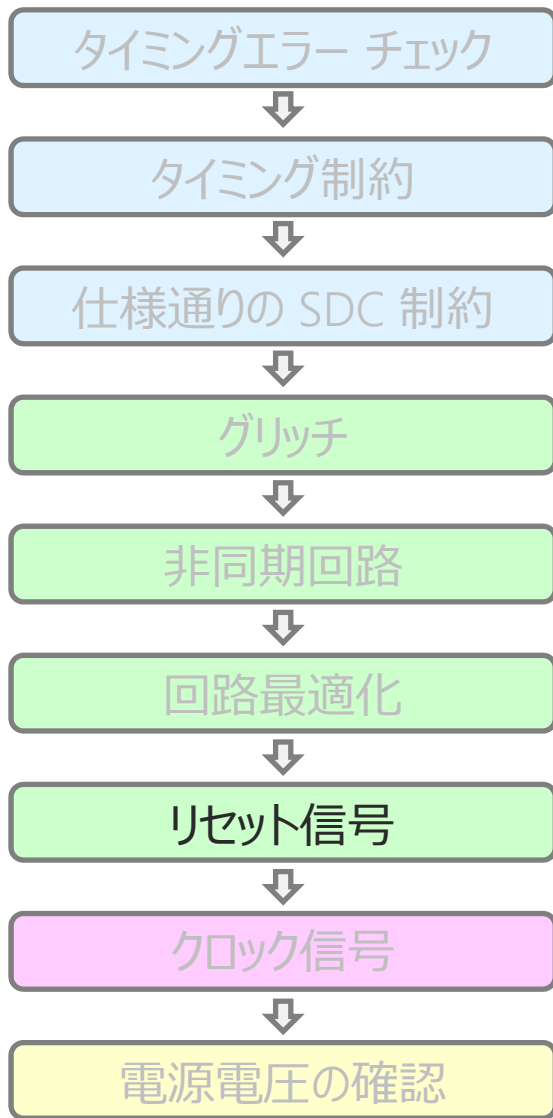
- 対策

- フリーランニングのリファレンス・クロックが要求されている場合は、フリーランニングが必須
 - トランシーバー・クロックなど
- PLL 使用時、Lock 信号を見て、ユーザー回路のリセットを解除
- コンフィグレーション終了時にクロックが未供給の場合は、安定したクロックが供給されたあとにリセットを解除する



※ counter 値は、Datasheet の tLOCKを参照

リセット信号：推奨リセット構成



- 同期化された非同期リセット

- 非同期でアサートを行い、同期でディアサートする

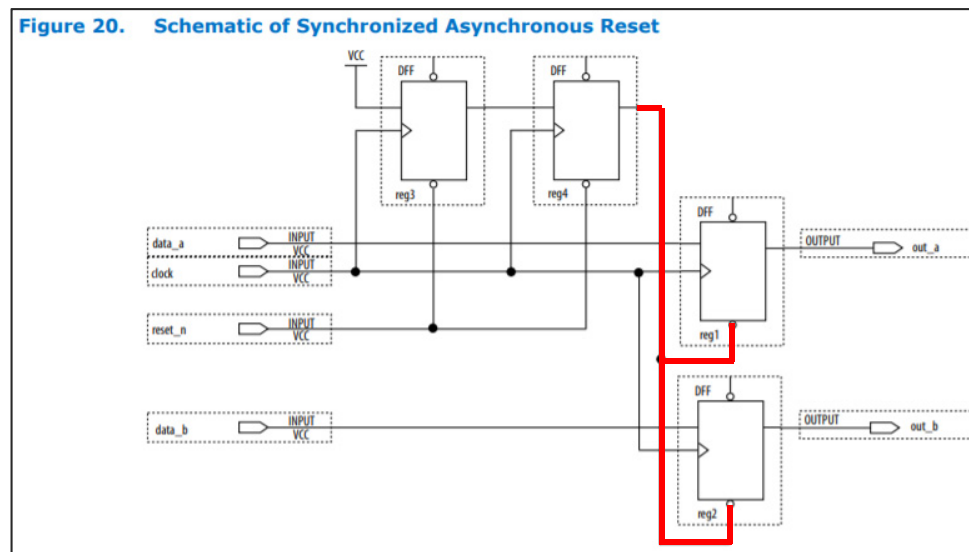
- メリット

- 非同期アサート：リセットを掛けると、非同期で全ての回路にリセットが入る

- 同期ディアサート：解除時はクロックに同期しているためメタステーブルの問題が無くなる

- 回路例：下図

- 赤線が Reg3/Reg4 により同期化された Reg1/Reg2 に対する非同期リセット



リセット信号：推奨リセット構成 Cont.

タイミングエラー チェック



タイミング制約



仕様通りの SDC 制約



グリッチ



非同期回路



回路最適化



リセット信号

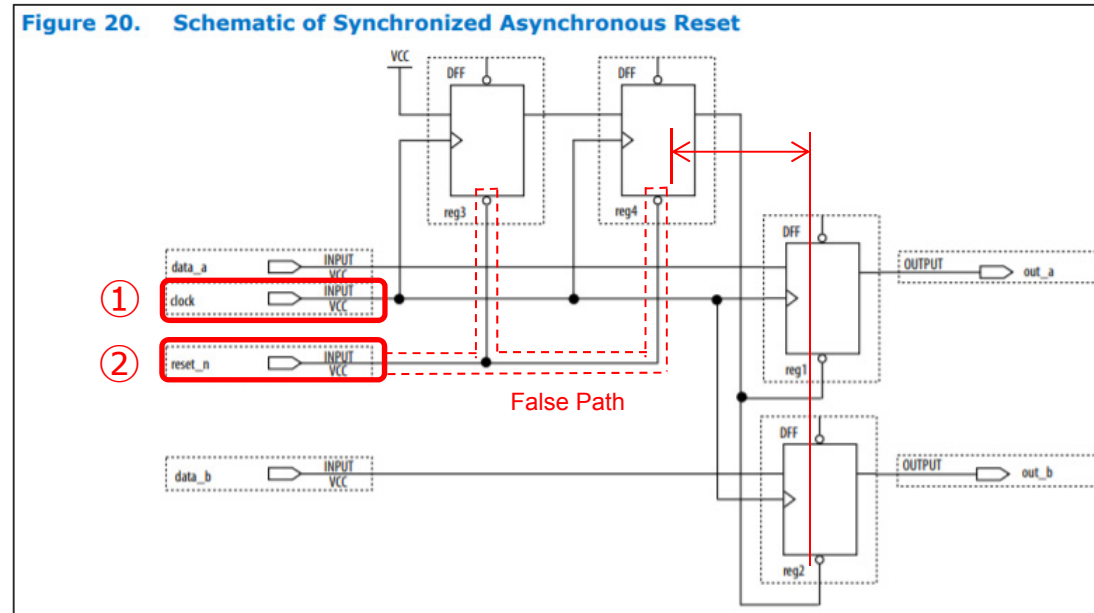


クロック信号



電源電圧の確認

- 同期化された非同期リセット：非同期パスに対する SDC 制約のかけ方
 - 以下の 2 つの制約が必要
 1. クロック（図の①）に対して create_clock の制約をする
 2. リセット（図の②）に対して set_false_path の制約をする
 - set_false_path -from [get_ports {reset_n}] -to <all_registers>



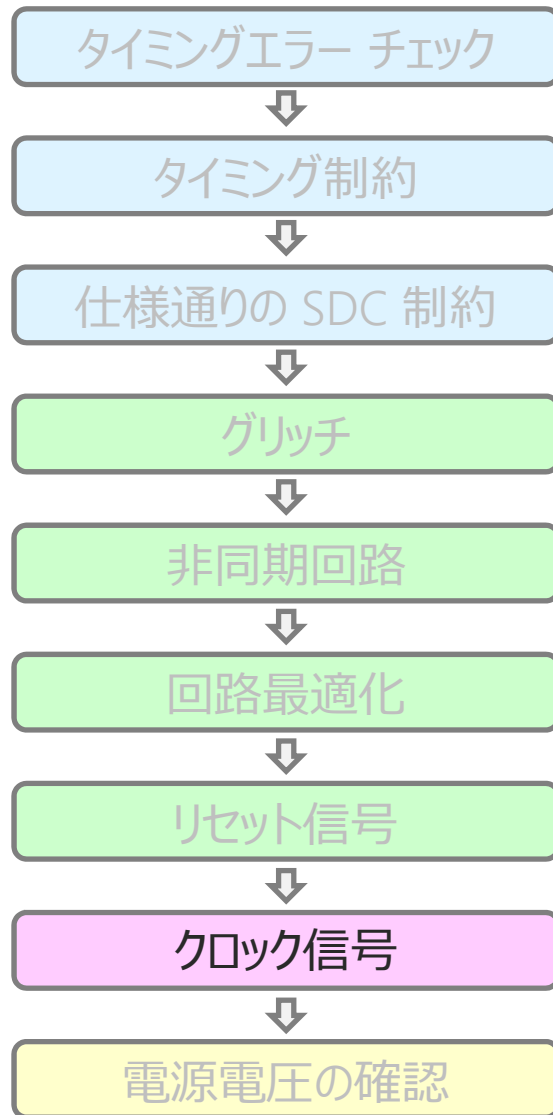
Reg1、Reg2 に対して create_clock で制約したクロックが非同期リセットの Recovery/Removal を満たしているか、ツールが自動で解析



タイミング・アナライザーにて、該当クロックに対して Recovery/Removal の Slack 値がマイナスになっていないか確認

クロック信号の確認

クロック信号の確認



- 入力クロック信号の I/O 設定は正しいか
 - インテル® Quartus® Prime のレポートファイルで確認 ([Appendix 1-3. 参照](#))
 - Differential / Single-end
 - AC-Coupling / DC-Coupling
 - On-Chip Termination / External Termination
- シングル・エンド・クロックを使用する場合はクロック入力ピンの差動ペアの p チャネルにクロック信号をアサインをしているか
- 入力クロックのジッターは FPGA のスペックに収まっているか
- 入力クロックは SDC で設定したスペックに収まっているか
- コンパイル時にワーニングは発生していないか

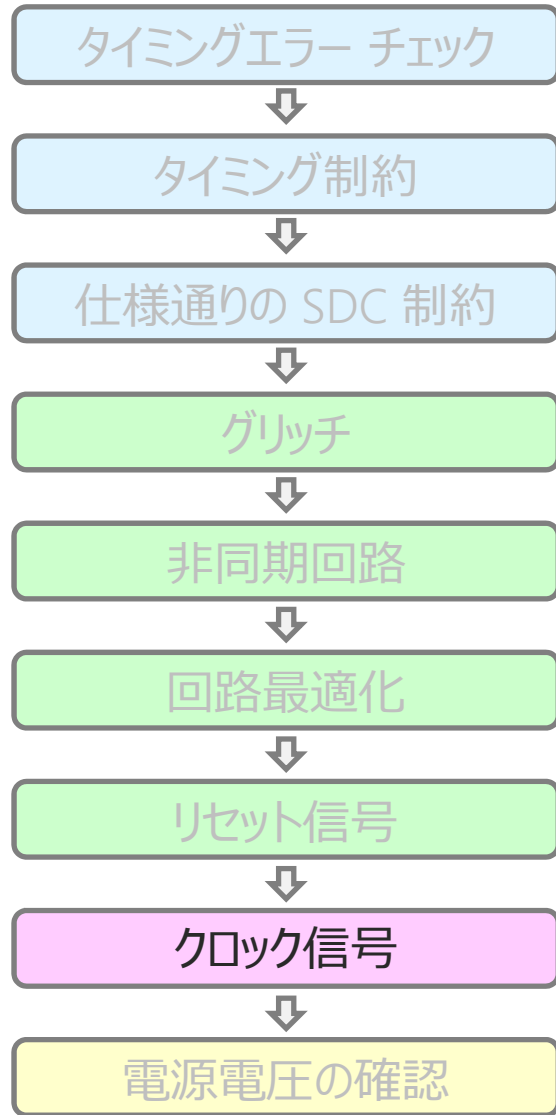
Arria 10 Datasheet

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-10/a10_datasheet.pdf#page=37

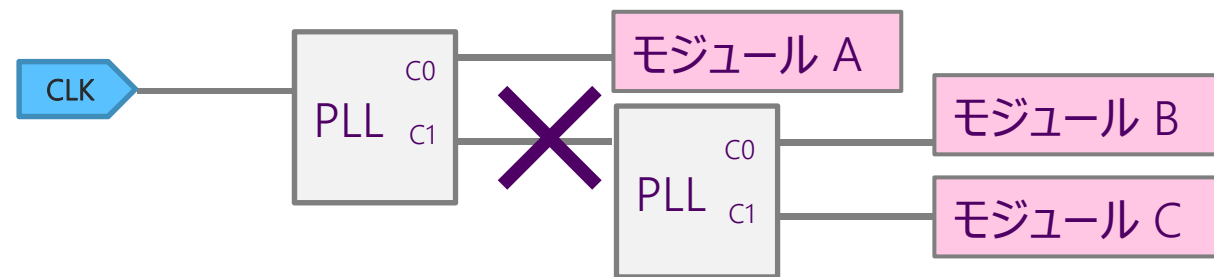
Stratix 10 Datasheet

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-10/s10_datasheet.pdf#page=43

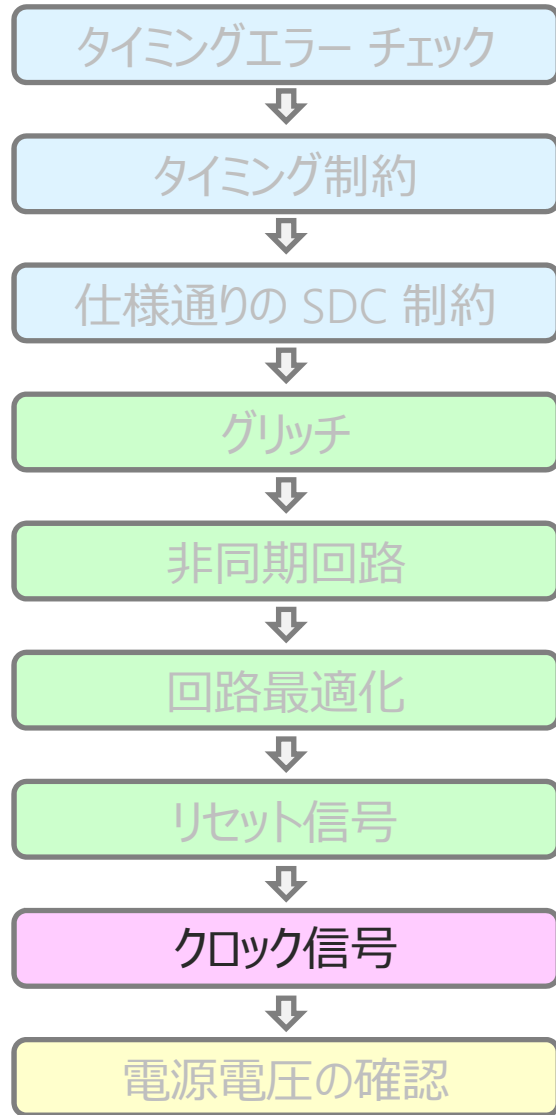
クロック信号の確認



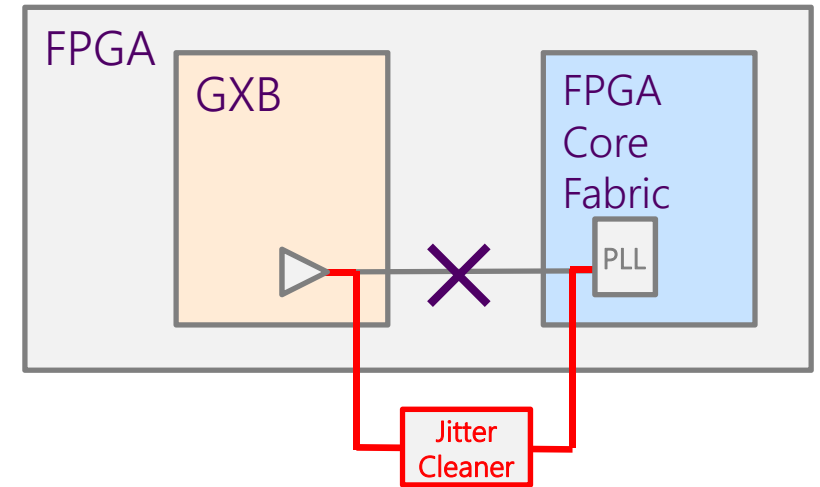
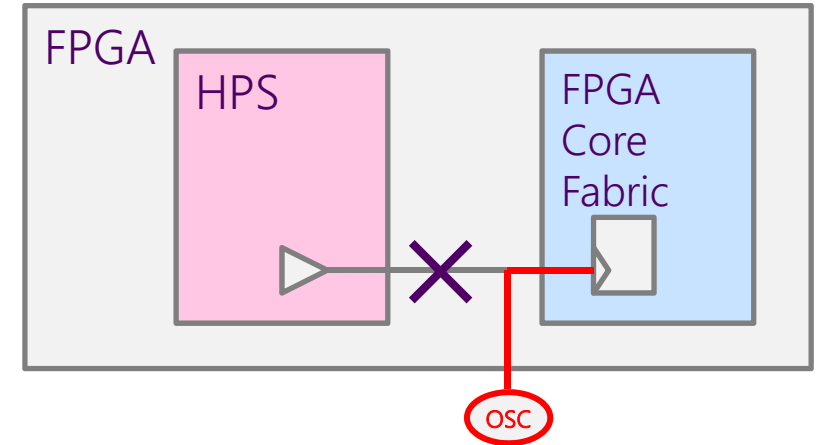
- PLL をカスケード接続していないか
 - PLL カスケードは避ける
 - PLL カスケードを使用する必要がある場合は、適切な設定で使用する ([Appendix 10-1. 参照](#))
 - カスケード専用配線を使用する
 - PLL 設定 (Bandwidth) は下記の通りに設定
 - ソース (upstream) PLL の Bandwidth 設定は Low に設定
 - ディステーション (downstream) PLL の Bandwidth 設定は High に設定
 - スペックがない場合は、実測した計測値を SDC に入力する



クロック信号の確認

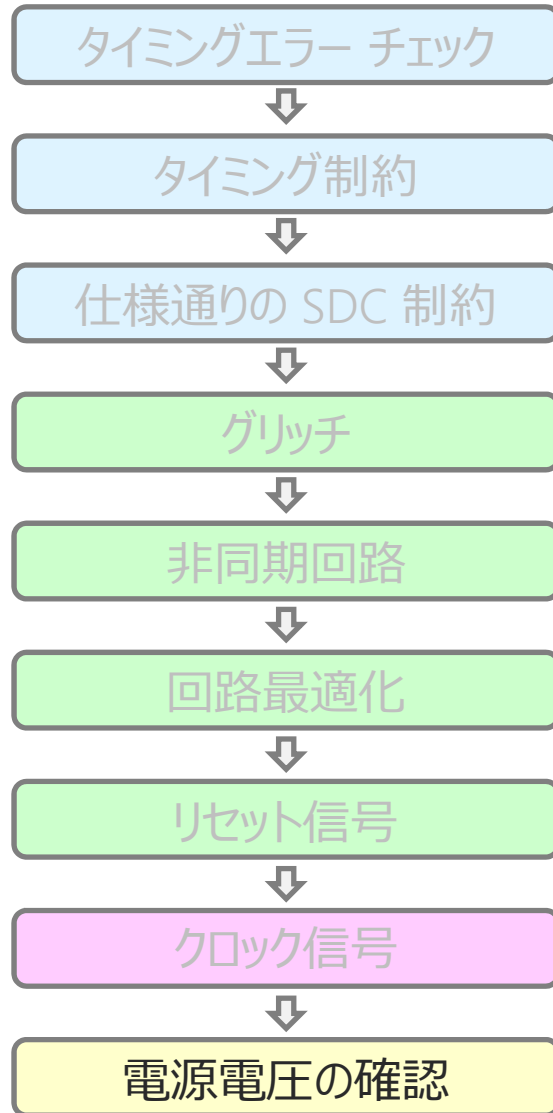


- HPS クロックを FPGA で使用しない
 - HPS クロックのジッター定義ができない
 - 対策：外部からクロックを入力
- トランシーバーからのリカバリークロックを PLL に接続して使用しない
 - トランシーバー・クロックのジッター定義ができない
 - 対策：ジッター・クリーナー・デバイスを使用して外部からクロックを入力



電源電圧の確認

電源電圧の確認



- 電源電圧、温度環境によって現象は変化するか
 - 電源電圧を高く・低くして不具合の発生頻度の変化を調べる
 - デバイス温度を高く・低くして不具合の発生頻度の変化を調べる
- 過渡負荷時、同時スイッチング時に現象が発生していないか
 - 電源が安定的に供給されているか確認
- “Power & Thermal 関連 デザイン & デバッグガイドライン” を参考にして、電源を見直す
 - http://www.altima.jp/members/p1-literature/2-device/1-altera/086_power_thermal_dd_guide.cfm

それでも解決しなかったら.....

それでも解決しなかったら..... (残りの20%)

● ボード起因

- リファレンス抵抗値の間違い
- Vcc または Ground への直接接続箇所への抵抗挿入
- ボードの製造不良
 - はんだの接触不良 (押すと改善する) やコンデンサー/抵抗の値の間違いや値抜け (試作や手付けの場合に多い)
- 対向デバイスの故障
 - 外部メモリーに多い
- 実装デバイスが想定と異なる (ヒートシンクの下なので気づかず)
 - ES デバイス (ES への Production デバイスのデータの書込み)
 - スピードグレードが異なるデバイス (遅いデバイスへの速いデバイスのデータを書込み)
- 信号品質
 - アイマスクのマージン不足・割れ

● ツール起因

- 古いバージョンのインテル® Quartus® Prime / IP を使用
 - インテル® Quartus® Prime バージョンと IP バージョンが違う場合は特に注意が必要 (基本的に禁止)
- スピードグレードの選択ミス

不具合事例

不具合事例

- **事例①**：再コンパイルを行うと、DDR4 のデータが稀に化ける時がある。温度を上げるとデータ化けが起きなくなる確率が高くなる
- **原因**：DDR4 とユーザー回路間のパスでタイミングエラーが発生していたが、タイミング解析を行う必要がないパスだと誤認識し、ユーザーが False Path に設定してしまっていた

- **事例②**：Ethernet 通信が 1 週間ほど経過するとスタックしてしまう
- **原因**：非同期信号のレーシングが発生しており、ユーザー回路を同期化することで安定的に動作した

- **事例③**：2 年間動作実績がある装置が、新たに納入されたデバイスから数%の割合で誤動作する
- **原因**：タイミング解析を行っていなかったため、多くのパスでタイミングエラーが発生していた。制約を見直してタイミングエラーを解決することで安定動作した

- **事例④**：Nios[®] II が動作中にフリーズしてしまう
- **原因**：Platform Designer でクロック・クロッシング・ブリッジを挿入しておらず非同期パスに対して False Path 設定を行っていた。Platform Designer の構成を見直すことで正常に動作した

不具合事例 Cont.

- **事例⑤**：ロケーションによって PLL が Lock しない場合がある
- **原因**：Cyclone[®] V で、PLL 用のリファレンス電圧 RREF ピンを 2k Ω でプルダウンしておらず、正しく処理することで定常的に PLL が Lock するようになった

- **事例⑥**：PCI Express (PCIe) が電源の ON/OFF でリンクしない場合がある
- **原因**：トランシーバーのリコンフィグ・コントローラーのクロックが外部クロックではなく、内部の PLL からのクロックが使用されていた

- **事例⑦**：PCIe が電源の ON/OFF でリンクしない場合がある
- **原因**：トランシーバーの電源が FPGA 端でドロップしていた。qsf の設定値と実際の電圧もあっているか確認

- **事例⑧**：DDR3 にてキャリブレーションは成功したにもかかわらず稀にデータ化けが起きる
- **原因**：タイミング制約の見直し及び電源の精度を確保したことにより改善

さいごに

FPGA の正常な動作を保証するためには、**適正な動作環境**下で、**適切に設定されたタイミング制約**を守っている必要があります。一方で、実際に FPGA にコンパイルデータをダウンロードして動作させた場合、下記のような不具合事象が発生する場合があります。

- ・ ユーザーデザインを変更していない場合であっても、**コンパイル毎に不具合動作・頻度・発生の有無が異なる**
- ・ 調査のために **Signal Tap を挿入**すると、不具合現象が再現しなくなる
- ・ FPGA の**個体**により不具合動作・頻度・発生の有無が異なる(P)
- ・ **電源電圧**を変化させると不具合動作・頻度・発生の有無が異なる(V)
- ・ **温度**が変化すると不具合動作・頻度・発生の有無が異なる（ある時間、動作させていると不具合動作が発生するようになる）(T)
- ・ ある時点から（ある**製品ロット**から）不具合動作が発生するようになる(P)
- ・ 対向デバイスを交換すると事象が変化する

これらの事象は、多くの場合、以下のような FPGA の特徴や半導体の特性に起因して発生することが経験的に確認されています。

- ・ **適切にタイミング制約が設定されていない**
- ・ **信号間の到達時間差（どれかが早い、遅い、同時）に対して、その時間差を考慮した回路構成となっていない**
- ・ **タイミング制約およびタイミングモデルで規定した条件外での使用**

本資料では、タイミング制約の生成方法や不具合が発生しにくい回路構成を示すとともに、不具合発生時のデバッグ手順を示しました。

本資料が不具合発生を防ぎ、不具合発生時にはデバックの一助となることを期待します。



Thank you!



Appendix

アジェンダ

1. レポートファイルの生成方法
 1. サマリーレポート
 2. 特定パスのレポート
 3. 入力クロックピンの I/O 設定
2. タイミング制約
 1. クロックの設定
 2. クロックグループの設定
 3. False Path の設定
 4. コンパイル後の False path の指定
 5. AC スペック の設定
 6. マルチ・サイクル・パスの設定
3. コンパイル毎/Signal Tap 追加時に動作しない場合の確認事項
4. Signal Tap 追加時の SDC 制約における注意事項
5. PLL Location, PLL Mode 等の確認方法
6. タイミングスラックの改善を模索
7. Timing Optimization Advisor
8. I/O エLEMENTレジスター設定
9. その他回路記述での注意点
10. PLL カスケード関連情報
 1. 各種デバイスハンドブック
 2. ユーザーガイド

1-1. レポートファイルの生成方法 <サマリーレポート>

- Timing Analyzer Summary Report の生成/確認
 - Timing Analyzer Summary Report はフルコンパイル時に自動生成
 - デフォルトでは下記に保存されている
 - パス : < Project Folder >¥output_files
 - 名前 : < Project Name >.sta.rpt
 - タイミング解析詳細レポート
 - 名前 : < Project Name >.sta.summary
 - タイミング解析サマリー
- 特定パスのレポートの生成方法は次項で説明

1-2. レポートファイルの生成方法 <特定パスのレポート>



- 特定のパスはタイミング・アナライザーで生成する必要がある
- 特定のパスのレポート生成方法
 - Quartus -> Tools -> Timing Analyzer
 - 所望のレポートをタイミング・アナライザー上で生成
 - レポートを生成する特定のパスをハイライトし、右クリック -> Export
 - テキスト方式もしくは HTML 形式を選択して保存

該当パスを右クリック ->
Export

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-3.112	a_c1	OOOUT_C1~reg0	c1_125m	selclk_c0	0.500	-0.722	2.710
	-0.332			_100m	selclk_c0	2.500	-0.133	2.519

1-3. 入力クロックピンの I/O 設定



- Fitter 実行後に生成されるレポートを確認
 - GUI の場合は
 - Compile Report -> Fitter -> Resource Section -> Input Pins

	Name	Pin #	I/O Bank	I/O Standard	Termination
1	clk1	AB27	5B	2.5 V	Off
2	clk2	AB30	5B	LVDS	Differential
3	clk2(n)	AA30	5B	LVDS	Differential
4	clk3	W8	B0L	HCSL	AC Coupling 100 Ohm OCT
5	clk3(n)	W7	B0L	HCSL	AC Coupling 100 Ohm OCT
6	d	AA28	5B	2.5 V	Off

- テキストファイルの場合は
 - パス : < Project Folder >¥output_files
 - 名前 : < Project Name >.fit.rpt

2-1. タイミング制約 <クロックの設定>



- タイミング解析の対象となるクロックを定義
 - FPGA に入力されるクロックの定義
 - コマンド : `create_clock`
 - 例 : `create_clock -name clkin_a -period 10 [get_ports CLKA]`
 - PLL の出力クロックを定義 (Stratix 10 を除く)
 - コマンド : `derive_pll_clocks`
 - FPGA 内部で分周/逡倍されたクロックの定義
 - コマンド : `create_generated_clock`
 - 例 : `create_generated_clock -name divclk -source [get_ports CLKA] -divide_by 2 [get_pins DIVFF|regout]`
- FPGA 内部のクロックのバラツキ (ジッターやスキュー)を定義
 - コマンド : `derive_clock_uncertainty`
 - 全てのデザインに使用することを推奨
- 外部クロックのジッターやクロックに余裕を持たせる場合などに使用
 - コマンド : `set_clock_uncertainty`
 - 例 : `set_clock_uncertainty -from { CLKA } -to * -setup 0.2`
- 同一ポートに二重でクロックを定義した場合は下の行の制約が有効
 - 上書きしないためには `-add` コマンドを使用
- 確認方法
 - コンパイル後、Timing Analyzer -> Report Clocks
 - 与えた制約が意図した通りに認識されているか確認
 - クロック信号統図との相関を確認

Timing Analyzer - C:/work/Timing/clock/clock_v180_restored/top - top

File View Netlist Constraints Reports Script Tools Window Help

Set Operating Conditions

Slow 1100mV 85C Model
Slow 1100mV 0C Model
Fast 1100mV 85C Model
Fast 1100mV 0C Model

Report

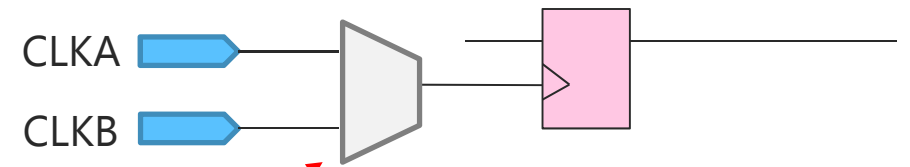
Timing Analyzer Summary
Advanced I/O Timing
SDC File List
Clocks Summary

	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle
1	CLK50M_PLL	Base	20.000	50.0 MHz	0.000	10.000	
2	c0_100m	Ge...ed	10.000	100.0 MHz	0.000	5.000	
3	c1_125m	Ge...ed	8.000	125.0 MHz	0.000	4.000	
4	PLL0 pl_coph[0]	Ge...ed	2.000	500.0 MHz	0.000	1.000	50.00
5	PLL0 pll_R divclk	Ge...ed	10.000	100.0 MHz	0.000	5.000	50.00
6	PLL0 pll_R divclk	Ge...ed	8.000	125.0 MHz	0.000	4.000	50.00
7	selclk_c0	Ge...ed	10.000	100.0 MHz	0.000	5.000	
8	selclk_c1	Ge...ed	8.000	125.0 MHz	0.000	4.000	

2-2. タイミング制約 <クロックグループの設定>



- 同期クロックをグループで纏め、グループ間で非同期のクロックとして指定し、タイミング解析時に指定したクロック間（不要なパス）の解析を行わないようにする
- コマンド：`set_clock_groups`
 - `-group`：クロックのグループを指定
 - `-asynchronous`：クロック間の関係が非同期の時
 - `-exclusive`：クロック間の関係が排他的の時



- 例：`set_clock_groups -exclusive -group {CLKA} -group {CLKB}`

確認方法

- Timing Analyzer -> Report Clock Transfers
- 解析不要なクロック間のパスは "false path" としてレポート

	From Clock	To Clock	RR Paths	FR Paths	RF Paths	FF Paths
1	c0_100m	c0_100m	1	0	0	0
2	c1_125m	c1_125m	1	0	0	0
3	c0_100m	selclk_c0	1	0	0	0
4	c1_125m	selclk_c0	false path	0	0	0
5	c0_100m	selclk_c1	false path	0	0	0
6	c1_125m	selclk_c1	1	0	0	0

2-3. タイミング制約 <False Path の設定>



- 特定のノードに対してタイミング制約を行わない場合に使用

- コマンド : `set_false_path`

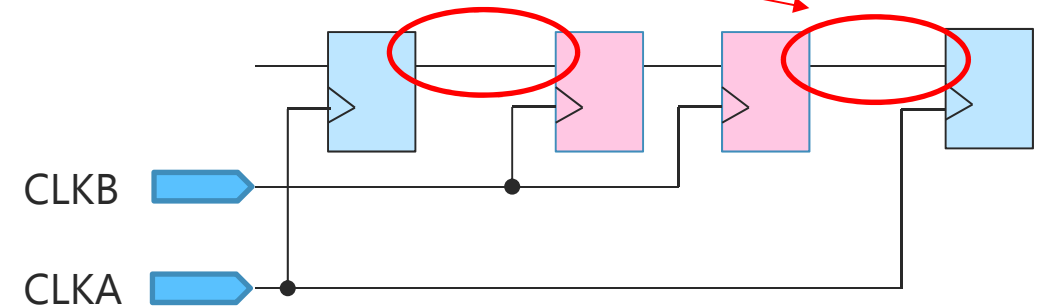
- 例 :

- `set_false_path -from [get_clocks CLKA] -to [get_clocks CLKB]`

- `set_false_path -from [get_clocks CLKB] -to [get_clocks CLKA]`

- False path を指定する場合の回路例

- 非同期であることを考慮して設計してる場合



- 注意事項

- ワイルドカード「*」を使用して制約を与えた場合、本来解析しなければならないパスを解析の対象から外してしまう可能性があるため注意が必要

- 一度 `set_false_path` を記述してしまうと、そのあとの行でタイミング制約を与えても無効になる

2-4. コンパイル後の false_path 指定



- コンパイル後、タイミング・アナライザーで非同期パスを false_path 指定する方法
 - 該当のパスをタイミング・アナライザーで表示

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-1.612	q_c1	QOUT_C1~reg0	c1_125m	selclk_c0	2.000	-0.722	2.710
2	7.168	q_c0	QOUT_C0~reg0	c0_100m	selclk_c0	10.000	-0.133	2.519

- 該当パスを右クリック -> set_false_path を選択

TO NODE	LAUNCH CLOCK	LATCH CLOCK	RELATIONSHIP
QOUT_C1~reg0	c1_125m	selclk_c0	2.000
QOUT_C0~reg0	c0_100m	selclk_c0	10.000

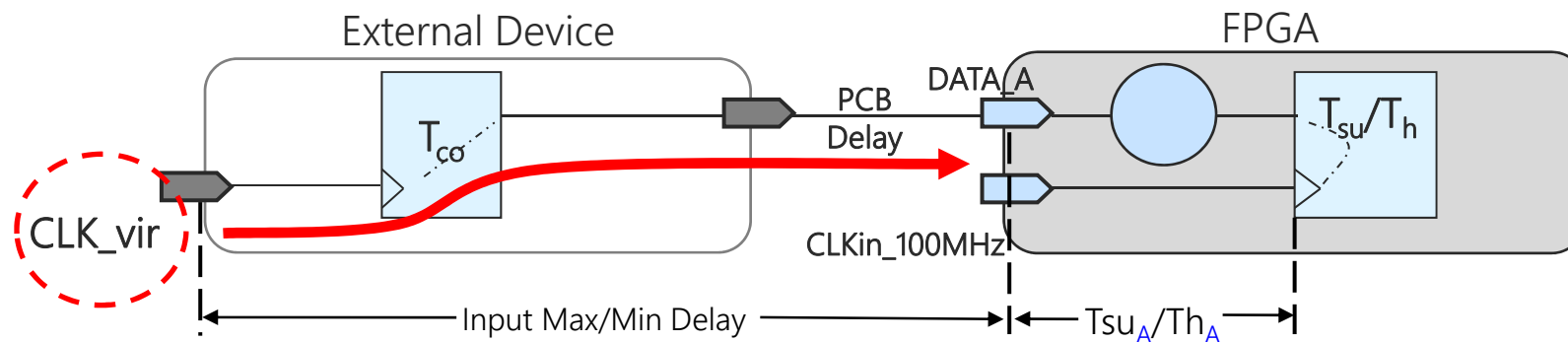
- Copy
- Select All
- Undo Sort
- Collapse All
- Expand All
- Report Worst-Case Path
- Report Timing...
- Report Timing Closure Recommendations...
- Set False Path (between nodes)**
- Set False Path (between clocks)**
- Set Multicycle (between nodes)...
- Set Multicycle (between clocks)...

- Constraints -> Write SDC で SDC を生成
- 該当のパスが false_path 指定されていることを確認し、ユーザーで作成した SDC に追記

2-5. タイミング制約 <AC スペックの設定 >



- FPGA への入出力に対する制約には**仮想クロックを定義**して制約を与える
 - 仮想クロックを使用して `set_input_delay` / `set_output_delay` を定義
 - 同一クロックソースであっても仮想クロックの使用を推奨
 - コマンド : `create_clock`
 - 例 : `create_clock -name {CLK_vir} -period 10`
 - FPGA に対応する port / pin は存在しないため周期のみの定義となる
- 外部デバイスと FPGA 間の入出力に対する制約
 - コマンド : `set_input_delay` / `set_output_delay`
 - 例 :
 - `set_input_delay -clock {CLK_vir} -max 0.5 [get_ports {DATA_A}]`
 - `set_input_delay -clock {CLK_vir} -min 0.2 [get_ports {DATA_A}]`



2-6. タイミング制約 <マルチ・サイクル・パスの設定>



- データの伝搬に 1 クロックサイクル以上必要なパスに与える制約

- コマンド : `set_multicycle_path`

- `-setup` : セットアップ解析時のサイクル数を増加させる (デフォルト値は 1)
- `-hold` : ホールド解析時のサイクル数を増加させる (デフォルト値は 0)
- `-start` : 送信クロックサイクルでサイクル数を指定をする場合に使用
- `-end` : 受信クロックサイクルでサイクル数を指定をする場合に使用

- 例 : 右図の赤丸の場合

- `set_multicycle_path -end -setup -from CLKA -to CLKB 2`
- `set_multicycle_path -end -hold -from CLKA -to CLKB 1`

- 設計者はマルチサイクルで与えた制約で、正しく回路が動作するように設計する必要がある

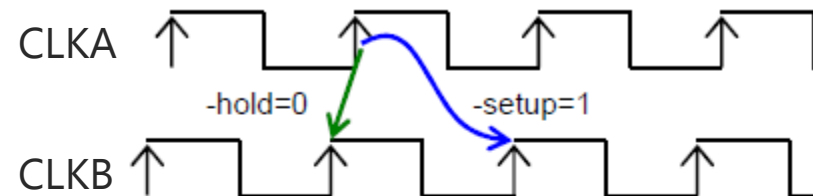
- マルチ・サイクル・パスの使用例

- デザインが 1 クロックでデータ伝搬を要求していない場合

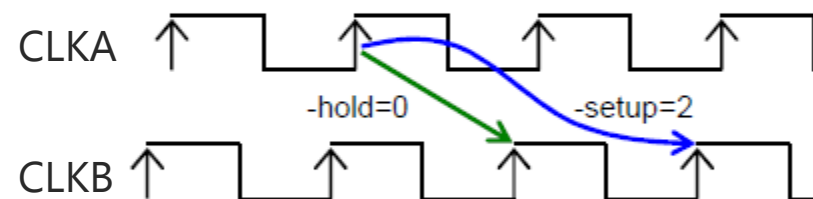
- 必要以上に厳しい制約条件を与えることを避ける

- レジスタが各クロックエッジごとにデータをサンプリングする必要がない場合

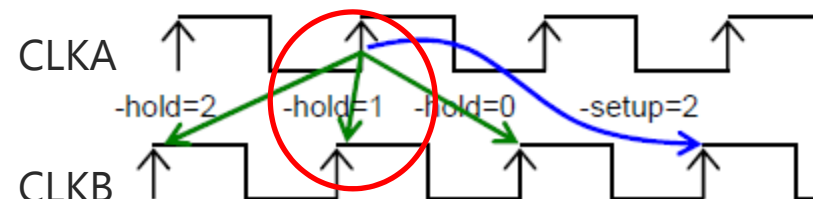
(A) デフォルト設定値



(B) Hold エッジは Setup に依存して変わる



(C) -hold 値の定義



3. コンパイル毎/Signal Tap 追加時に動作しない場合の確認事項

- SDC は正しく与えているか
 - AC スペックとクロック信号統図を基に SDC 制約内容を確認
 - 前項の各種レポートの内容を確認
- Signal Tap 追加時は JTAG 制約と false_path 制約を与える
 - 注意事項は次項に記載
- 要求タイミングを満たしているか
 - SDC が正しいが、タイミングを満たしていない場合の対処方法例を後述

4. Signal Tap 追加時の SDC 制約における注意事項

- JTAG 制約
 - インテル® Quartus® Prime Handbook に記載の [Cook book](#) を活用
 - インテル® FPGA ダウンロード・ケーブル (USB-Blaster II) 使用時、TCK の周波数を指定可能
 - 実機で使用している TCK 周波数と SDC で与えている TCK 周波数を同一にする
 - 10 ピンヘッダから FPGA までのトレース長を考慮した SDC 制約を与えることが可能
- Signal Tap に対して False path 制約を与える

5. PLL Location / PLL Mode の確認方法



- インテル® Quartus® Prime の Fitter report で確認可能
 - Compile Report -> Fitter -> Resource Section -> PLL Usage Summary

The screenshot shows the 'PLL Usage Summary' report in Quartus Prime. The left sidebar shows the 'Table of Contents' with 'PLL Usage Summary' selected under the 'Resource Section'. The main window displays a table of PLL parameters for 'PLL0:PLL0'. Several rows are highlighted with red boxes and callouts:

- Row 2: '-- PLL Location' is highlighted with a callout 'PLL Location の確認'.
- Row 5: '-- Reference Clock Frequency' is highlighted with a callout 'リファレンス・クロックの周波数はロックレンジに収まっているか'.
- Row 8: '-- PLL Operation Mode' is highlighted with a callout 'PLL mode は意図通りか'.

Other rows in the table include: PLL Type (Integer PLL), PLL Feedback clock type (none), PLL Bandwidth (Auto), PLL Bandwidth Range (1200000 to 600000 Hz), Reference Clock Sourced by (Dedicated Pin), PLL VCO Frequency (500.0 MHz), PLL Freq Min Lock (30.000000 MHz), PLL Freq Max Lock (65.000000 MHz), and PLL Enable (On).

Additional callouts on the right side of the screenshot include:

- 'リファレンス・クロック周波数が実機とあっているか' (Is the reference clock frequency the same as the actual hardware?)
- 'リファレンス・クロックは専用端子から入力しているか' (Is the reference clock input from a dedicated pin?)

6. タイミングスラックの改善を模索



- インテル® Quartus® Prime の設定/制約を変更
 - 配置配線結果が変わることで、タイミング解析結果が変わることを期待
 - Compiler Settings
 - Timing Optimization Advisor
 - 他の設定や制約
- デバイス変更
 - スピードグレードを上げる
- デザインを変更
 - レジスター間の組み合わせ回路の見直し
 - レジスターの追加（パイプライン化）
など...

7. Timing Optimization Advisor

✔ : 設定済み
⚠ : 未設定 (設定するかは設計者判断)



- パフォーマンスの最適化に関するアドバイザー機能
 - Tools メニュー -> Advisor -> Timing Optimization Advisor

Timing Optimization Advisor

Timing Summary

- How to use the Timing Optimization Advisor
- General Recommendations
 - Get more information
 - Create a revision
 - ⚠ Use smart compilation
 - Review timing constraints - TQ
 - Select a Faster Speed Grade Device
- Maximum Frequency (fmax)
 - ⚠ Use High-Effort fmax Optimization Settings
 - ⚠ **Optimize for speed**
 - ⚠ Use physical synthesis optimizations
 - ✔ Increase physical synthesis optimizations effort
 - ⚠ Enable Logic Cell Insertion - Logic Duplication
 - Set maximum
 - ⚠ Avoid unrelate
 - ✔ Enable Beneficial Skew Optimization
 - Other Recommendation
 - I/O Timing (tsu, tco, tp)
 - ✔ Use timing-driven
 - ✔ Turn on Auto Global Clock

For quarter-time integrated synthesis, choose Speed under Optimization Technique in the Advanced Analysis & Synthesis Settings page under Compiler Settings in the Settings dialog box (Assignments menu). It is also recommended to set the optimization technique to Balanced if it is currently set to Area. Balanced technique gives better fmax than Area, worse than Speed. But resource usage is an with Speed (worse than with Area). You specify the Optimization Technique logic option for individual partitions in your design using the Assignment Editor (Assignments menu), while leaving the project Optimization Technique setting at Balanced (for the best trade off between area and speed for certain device families) or Area (if area is an important concern).

説明

ワンクリックで設定

Settings や Assignment Editor へ簡単アクセス

Correct the Settings

Undo

Current Global Settings:
Optimization Technique = BALANCED (Recommended: SPEED)

[Open Settings dialog box - Advanced Analysis & Synthesis Settings dialog box](#)

[Open Assignment Editor](#)

8. I/O エlementレジスター設定



- I/O Element内のレジスターへマッピングさせるオプション
 - タイミングを高速にするオプション
 - セットアップ時間、クロック to アウトプット時間

● 設定手順

1. Pin Planner を起動 (Assignments メニューより)
 2. Pin Planner 内の All Pins リストにて確認
- 下図は、Fast Output Register を On に設定する例

● 設定後の確認

Compilation Report -> Fitter -> Resource Section
目的のピンを確認 (下図の例: Output Pins)
Output Register に "yes" と表示されていることを確認

Node Name	Direction	Location	I/O Bank	I/O Standard	Fast Input Register	Fast Output Register	Fast Output Enable Register
button	Input			2.5 V (default)			
clock_50	Input			2.5 V (default)			
clr	Input			2.5 V (default)			
led[7]	Output	PIN_H9	8A	2.5 V		On	
led[6]	Output	PIN_H8	8A	2.5 V		Off	
led[5]	Output	PIN_B6	8A	2.5 V		On	
led[4]	Output	PIN_A5	8A	2.5 V		On	
led[3]	Output	PIN_E9	8A	2.5 V		On	
led[2]	Output	PIN_C8	8A	2.5 V		On	
led[1]	Output	PIN_D6	8A	2.5 V		On	
led[0]	Output	PIN_L7	8A	2.5 V		On	

Name	Pin #	I/O Bank	Output Register	Output Enable Register
led[0]	L7	8A	yes	no
led[1]	D6	8A	yes	no
led[2]	C8	8A	yes	no
led[3]	E9	8A	yes	no
led[4]	A5	8A	yes	no
led[5]	B6	8A	yes	no
led[6]	H8	8A	yes	no
led[7]	H9	8A	yes	no

9. その他回路記述での注意点

1. グリッチ
 - パルス（リセット信号）生成回路
2. 非同期回路
 - 異なるクロック間のケア
3. 非同期回路の確認方法
 - タイミング・アナライザーを使用して MTBF を計測
4. リセット信号
 - Stratix[®] 10 (Hyper-Register) を使用した場合のリセットのケア
5. 回路最適化
 - ステートマシンの 設定方法
6. デザインチェック
 - Design Assistant
7. HDL コーディング
 - Block RAM, DSP への割り当て

9-1. グリッチ

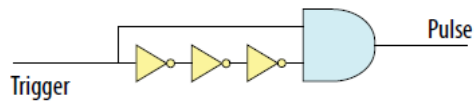
- パルス（リセット信号）生成回路

- 不具合事象

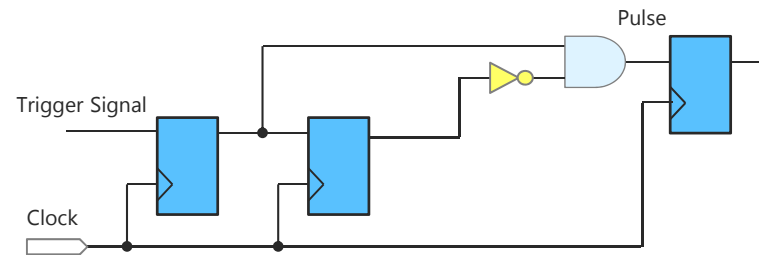
- パルス回路を組み合わせ回路で生成するとグリッチが発生する

- 対策

- パルス信号を同期化



NG の例



OK の例

9-2. 非同期回路

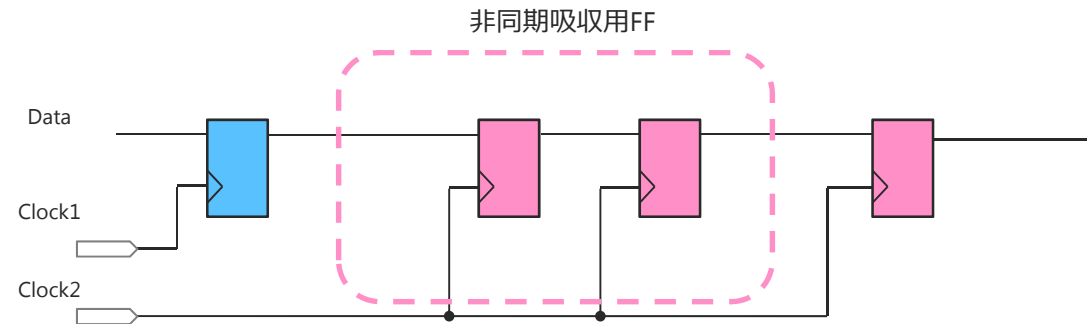
- 異なるクロック間のケア

- 不具合事象

- 異なるクロックドメインで非同期渡しする際にメタステーブルが発生し、正しいデータが伝搬できない

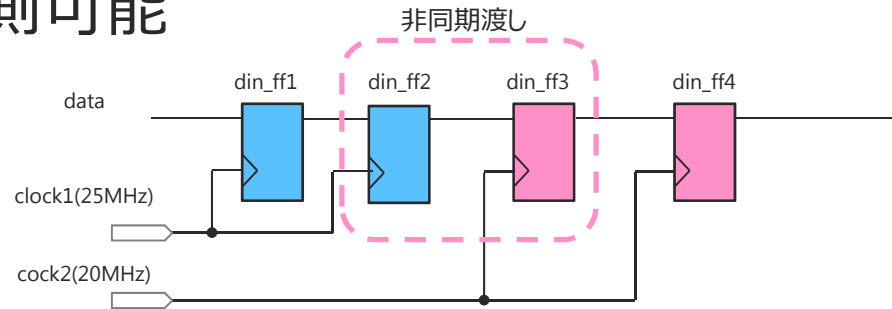
- 対策

- 非同期吸収用の FF を 2 段以上挿入
 - FIFO よる非同期吸収 (Platform Designer の場合、Clock Crossing Bridge を使用)



9-3. 非同期回路の確認方法

- タイミング・アナライザーを使用して MTBF を計測
 - 非同期に非同期吸収用の FF を入れた場合、挿入した FF の段数でどの程度信頼できるかが計測可能



The screenshot shows the 'MTBF Summary' tab in a timing analysis tool. The table below is a representation of the data shown in the image:

	Source Node	Synchronization Node	Worst-Case MTBF (Years)	Typical MTBF (Years)	Included in Design MTBF
1	din_ff2	din_ff3	Greater than 1 Billion	Greater than 1 Billion	Yes

Below the table, the 'Synchronizer Chain #1: Worst-Case MTBF is Greater than 1 Billion Years' section is visible, with a 'Report Metastability...' option highlighted in the left-hand 'Tasks' pane.

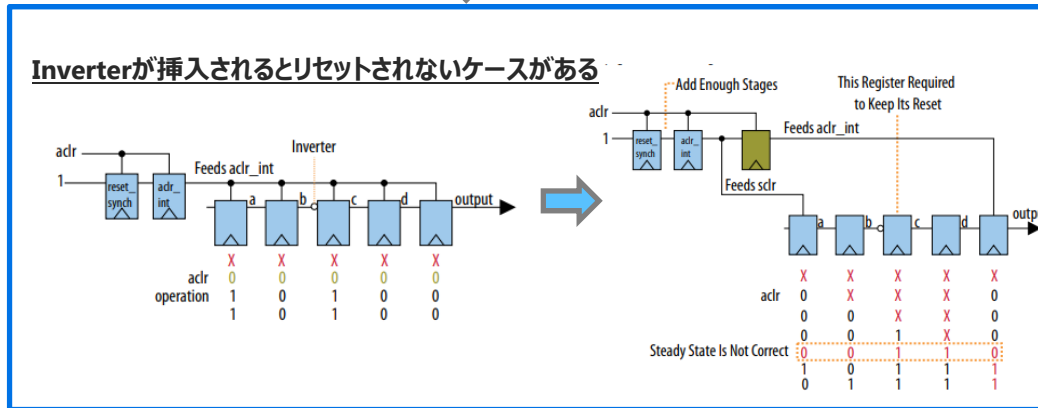
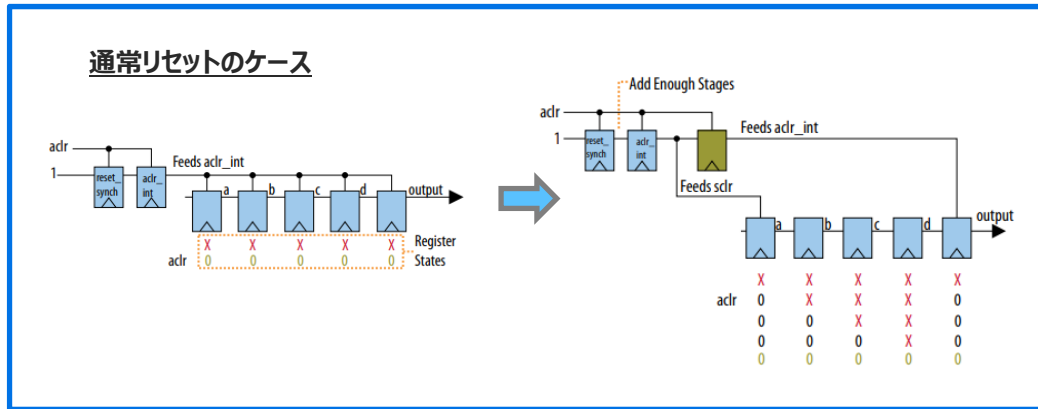
Typical も Worst も MTBF は Greater than 1 Billion years となり 10億年以上に 1回メタステーブルが発生という意味になる。この値は 最長値となり、この場合は最善の処置がとられている。これより少ない 値が表示されている場合は、同期化回路を挿入するなどし、故障間 隔を改善させることが可能。

9-4. リセット信号

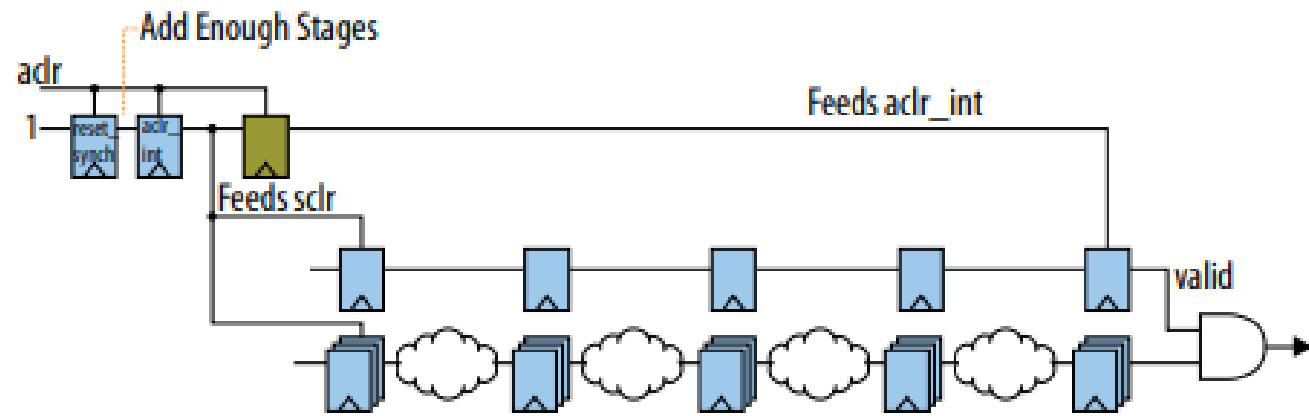
- Stratix® 10 (Hyper-Register) を使用した場合のリセットのケア

- 不具合事象

- Hyper-Register は、リセットなしの FF になるため、誤ったリセット構成になり誤動作の原因となる



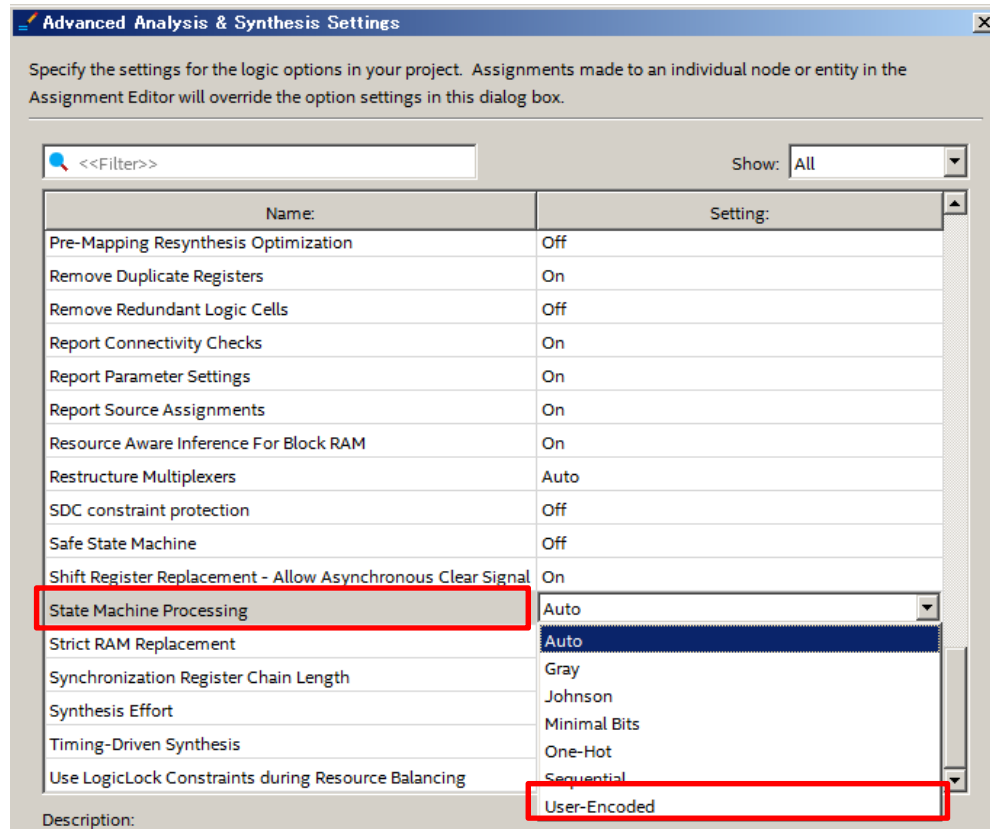
推奨リセット構成



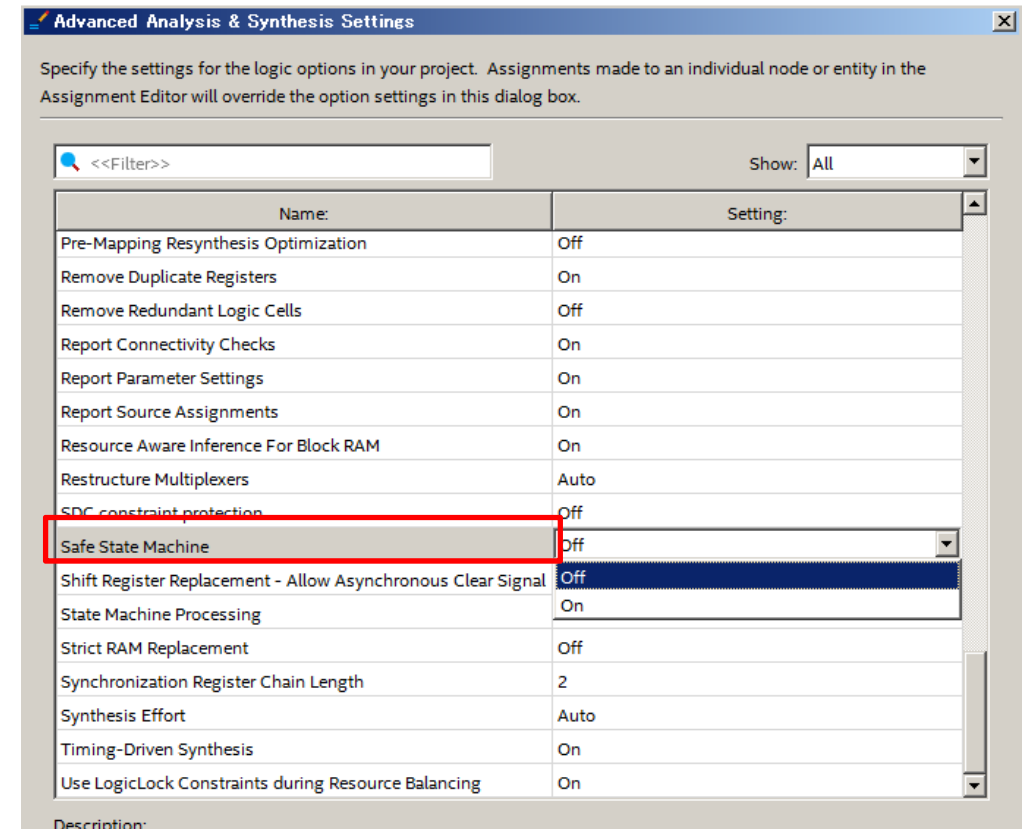
9-5. ステートマシンの設定方法



- State Machine Processing / Safe State Machine の設定方法
 - Assignments -> Settings -> Compiler Settings -> Advanced Settings (Synthesis...) より設定



State Machine Processing

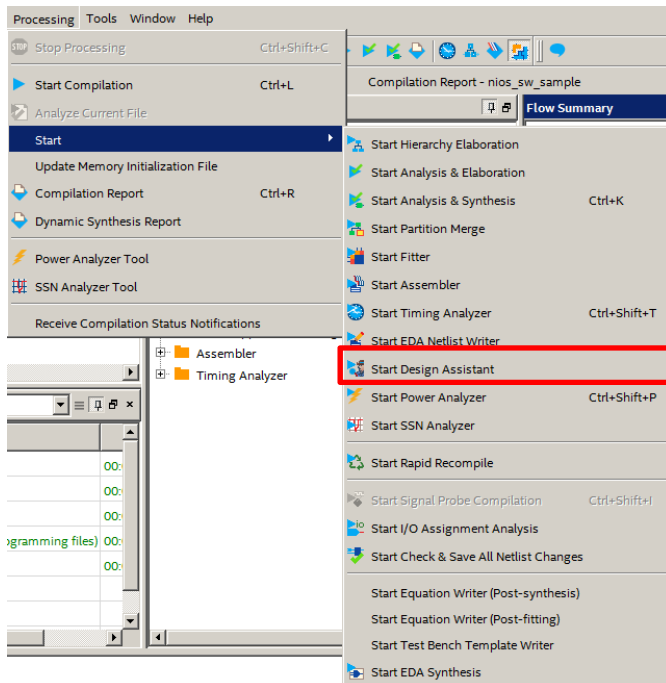


Safe State Machine

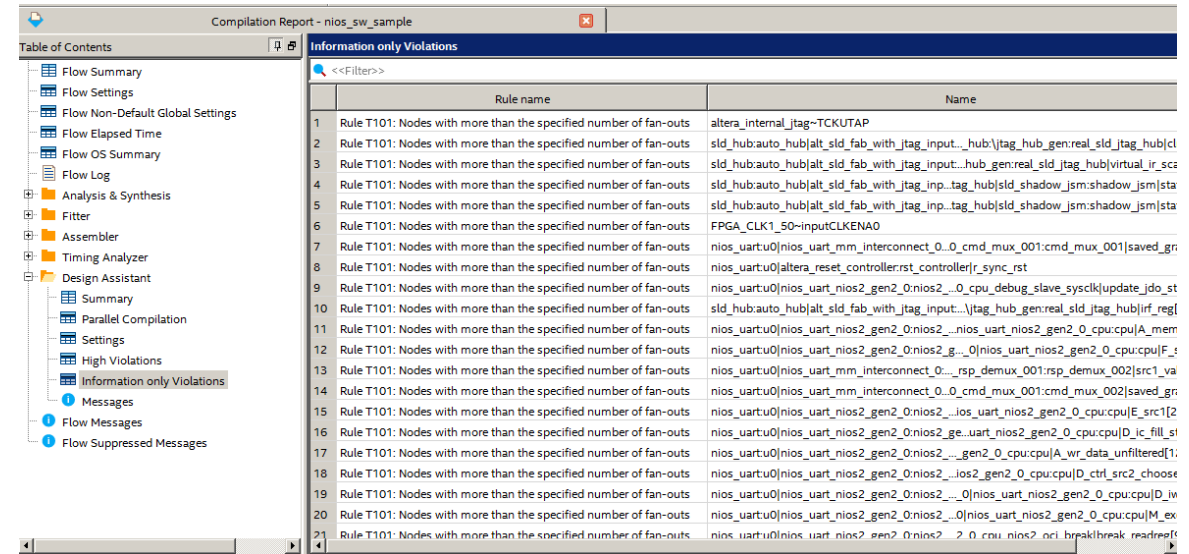
9-6. デザインチェック

- Design Assistant

- インテル® Quartus® Prime には、コーディングをチェックするツール(HDL の lint チェック機能)
- Stratix® 10, Arria® 10, Cyclone® 10 GX, MAX® 10 はサポート外
- 実行方法
 - フルコンパイル完了後、Processing -> Start -> Start Design Assistant より実行



実行結果



9-7. HDL コーディング

- HDL のコーディング

- 不具合事象

- RTL の記述によっては、Block RAM や DSP に割当たらないケースあり

- 対策

- インテル® Quartus® Prime の RTL のテンプレートを使用して記述

- Verilog-HDL / VHDL のテキスト編集画面で 下記アイコンから Insert Template 用アイコンをクリックし、RAM や DSP のテンプレートを選択

The image illustrates the process of inserting a template into a Verilog-HDL code editor. It shows the following steps:

- Insert Template 用アイコン**: The user clicks the 'Insert Template' icon in the toolbar.
- テンプレートを選択**: The 'Insert Template' dialog box is shown, where the user selects a template from the 'Language templates' list.
- テンプレートが決まったら Insert**: The user clicks the 'Insert' button in the dialog box to insert the selected template into the code editor.

```
1 // Quartus Prime Verilog Template
2 // Single port RAM with single read/write address
3
4 module single_port_ram
5   #(parameter DATA_WIDTH=8, parameter ADDR_WIDTH=6)
6   (
7     input [(DATA_WIDTH-1):0] data,
8     input [(ADDR_WIDTH-1):0] addr,
9     input we, clk,
10    output [(DATA_WIDTH-1):0] q
11  );
12
13 // Declare the RAM variable
```

```
// Quartus Prime Verilog Template
// Single port RAM with single read/write address
#(parameter DATA_WIDTH=8, parameter ADDR_WIDTH=6)
module single_port_ram
(
  input [(DATA_WIDTH-1):0] data,
  input [(ADDR_WIDTH-1):0] addr,
  input we, clk,
  output [(DATA_WIDTH-1):0] q
);
// Declare the RAM variable
reg [DATA_WIDTH-1:0] ram[2**ADDR_WIDTH-1:0];
// variable to hold the registered read address
reg [ADDR_WIDTH-1:0] addr_reg;
always @ (posedge clk)
begin
  // write
  if (we) ram[addr] <= data;
end
addr_reg <= addr;
// Continuous assignment implies read returns NEW data.
// This is the natural behavior of the TriMatrix memory
// blocks in Single Port mode.
assign q = ram[addr_reg];
endmodule
```

10-1. PLL Cascading 関連情報：各種デバイスハンドブック



- Intel® Stratix® 10 Clocking and PLL User Guide
 - <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-10/ug-s10-clkpll.pdf#page=23>
- Intel® Arria® 10 Core Fabric and General Purpose I/Os Handbook
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-10/a10_handbook.pdf#page=89
- Intel® Cyclone® 10 GX Core Fabric and General Purpose I/Os Handbook
 - <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-10/c10gx-51003.pdf#page=80>
- Intel® MAX® 10 Clocking and PLL User Guide
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/ug_m10_clkpll.pdf#page=39
- Stratix V Device Handbook Volume 1: Device Interfaces and Integration
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-v/stx5_core.pdf#page=101
- Arria V Device Handbook Volume 1: Device Interfaces and Integration
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-v/av_5v2.pdf#page=112
- Cyclone V Device Handbook Volume 1: Device Interfaces and Integration
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_5v2.pdf#page=87

10-1. PLL Cascading 関連情報：各種デバイスハンドブック Cont.



- Stratix IV Device Handbook, Volume 1
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-iv/stx4_5v1.pdf#page=137
- Arria II Device Handbook
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/arria-ii-gx/aiigx_5v1.pdf#page=129
- Cyclone IV Device Handbook, Volume 1
 - <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-5v1.pdf#page=95>

10-2. PLL Cascading 関連情報：ユーザーガイド



- IOPLL Intel FPGA IP Core User Guide
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_altera_iopll.pdf
 - Arria 10, Cyclone 10 GX
- ALTPLL (Phase-Locked Loop) IP Core User Guide
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_altpll.pdf
 - Cyclone 10 LP, Cyclone IV, Arria II, Stratix IV, or older
- Altera Phase-Locked Loop (Altera PLL) IP Core User Guide
 - https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/altera_pll.pdf
 - Stratix V, Arria V, Cyclone V



Thank you

改版履歴

Revision	年月	概要
1	2018年12月	初版作成

弊社より資料を入手されたお客様におかれましては、下記の使用上の注意を一読いただいた上でご使用ください。

1. 本資料は非売品です。許可なく転売することや無断複製することを禁じます。
2. 本資料は予告なく変更することがあります。
3. 本資料の作成には万全を期していますが、万一ご不明な点や誤り、記載漏れなどお気づきの点がありましたら、弊社までご一報いただければ幸いです。
4. 本資料で取り扱っている回路、技術、プログラムに関して運用した結果の影響については、責任を負いかねますのであらかじめご了承ください。
5. 本資料は製品を利用する際の補助的な資料です。製品をご使用になる場合は、英語版の資料もあわせてご利用ください。