

概要

このクイック リファレンス ガイドでは、『FPGA および SoC 用 UltraFast 設計手法ガイド (Vivado Design Suite 用)』(UG949) の推奨事項に基づいて、タイミング クロージャをすばやく簡単に実行する手順を説明します。

- **初期デザイン チェック:** デザインのインプリメンテーション前に使用量、ロジックレベル、タイミング制約を確認。
- **タイミング ベースラインの作成:** 配線後にタイミング クロージャを達成しやすくするため、各インプリメンテーション段階後にタイミング違反を確認。
- **タイミング違反の解決:** セットアップまたはホールド違反の原因を見つけて、タイミング違反を解決。

QoR 評価レポート

QoR (結果の品質) 評価レポートからは、デザインをすばやく確認できます。ガイドライン制限に反する重要なデザインおよび制約マトリクスが比較されます。ガイドラインに従っていないものには、「REVIEW」と記述されます。レポートには、次のセクションが含まれます。

- デザイン特性
- 設計手法チェック
- ターゲット Fmax に基づいた保守的なロジックレベル評価

AMD Vivado® では、次を使用してレポートを生成できます。

report_qor_assessment

詳細は、『Vivado Design Suite ユーザー ガイド: デザイン解析およびクロージャ テクニク』(UG906) の「QoR評価レポートの概要」を参照してください。

QoR 推奨項目レポート

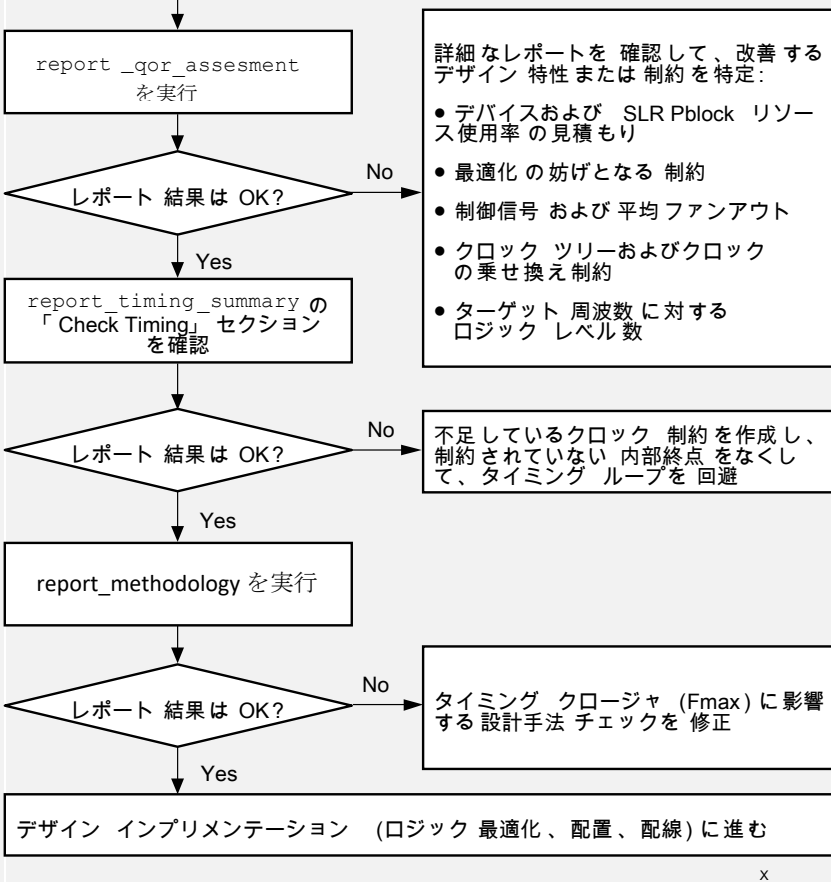
Vivado ツールでは、report_qor_suggestions はインプリメンテーション段階で呼び出されます。このレポートはデザインを解析し、推奨項目を示し、場合によってはその推奨項目を自動的に適用します。

Vitis 環境のレポート

AMD Vitis™ 環境では、v++ -R 1 または v++ -R 2 を使用すると、コンパイルフロー中に report_qor_assessment が呼び出されます。

初期デザイン チェック フロー

合成済みデザイン チェックポイント (DCP) または -opt_design 後の DCP (ある場合のみ) を開く



ヒント: インプリメンテーション中に最も困難なタイミング クロージャを自動的に解決するには、report_qor_suggestions、ML ベースのストラテジ予測、インクリメンタル コンパイルを利用した特殊なタイプのインプリメンテーションであるインテリジェント デザイン run (IDR) を使用できます。詳細は、UG949 の「インテリジェント デザイン run の使用」を参照してください。

AMD デバイスにデザインをインプリメントするタスクはかなり自動化されていますが、優れたパフォーマンスを達成し、タイミングや配線違反によるコンパイル問題を解決するのは複雑であり、時間がかかることもあります。問題の原因を単純なログ メッセージやツールで生成されるインプリメンテーション後のタイミング レポートから判断するのは困難なことがあります。このため、中間結果を確認してデザインが次のインプリメンテーション段階に進めるようにするなど、手順を追ったデザイン開発およびコンパイル手法を使用することが重要となります。

まず、最初のデザイン チェックで検出された問題をすべて解決します。これらのチェックは次のレベルで確認します。

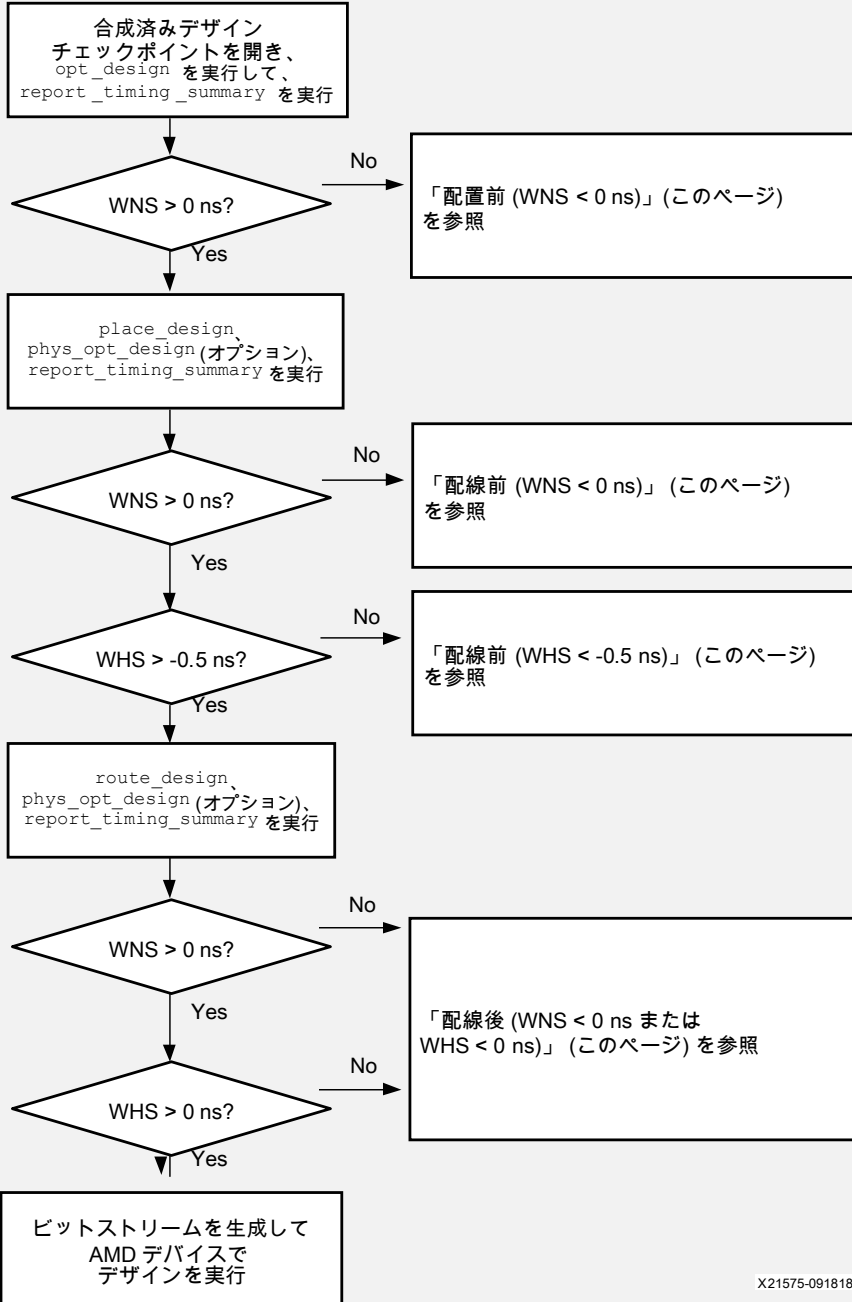
- カスタム RTL または Vivado HLS で生成されたカーネルごと。
注記: ターゲット クロック周波数制約が現実的なものであるかどうかを確認します。
- 複数のカーネル、IP ブロック、および接続ロジックを含む Vivado IP インテグレーターのブロック図など、サブシステムに該当する主な階層ごと。
- 主な関数および階層、I/O インターフェイス、全クロック供給回路、物理およびタイミング制約すべてを含む完全なデザイン。

デザインで SLR (Super Logic Region) 割り当てや Pblock に割り当てられたロジックなどのフロアプラン制約を使用する場合は、物理制約ごとにリソース使用率の見積もりを確認し、使用率ガイドラインに従っていることを確認します。

report_qor_assessment を実行すると、SLR および Pblock 違反が自動的にチェックされます。違反がない場合は、デザインは許容範囲内にあるということです。

タイミング ベースライン フロー

タイミング ベースライン例



タイミング ベースラインを作成するのは、各インプリメンテーション段階後にタイミング問題を解析して解決し、デザインのタイミング要件が満たされるようにするためです。デザインおよび制約の問題をコンパイル フローの早期に修正すると、効果が大きく、パフォーマンスを向上できます。次の段階に進む前に、次の中間レポートを作成して、タイミング違反を確認して解決しておきます。

Vivado プロジェクト モードのレポート	Vivado 非プロジェクト モードのレポート	Vitis ソフトウェア プラットフォームのレポート
AMD UltraFast™ 設計手法またはタイミング クロージャレポート ストラテジを使用します。	各インプリメンテーション段階後に、次のレポート コマンドを追加します。 <ul style="list-style-type: none"> report_timing_summary report_methodology report_qor_assesment 	v++ -R 1 または v++ -R 2 オプションを使用して、中間タイミング レポート、DCP を <code><runDir>/_x/link/vivado/prj/prj.runs/impl_1</code> ディレクトリに生成します。

配置前 (WNS < 0 ns)

place_design 前のタイミング レポートには、ロジック パスごとにできるだけ最適なロジック配置を想定したデザインパフォーマンスが示されます。セットアップ違反は初期チェックの推奨事項に従って解決する必要があります。

配線前 (WNS < 0 ns)

route_design 前のタイミング レポートには、ホールド違反の修正の影響 (ネット配線の迂回) や密集を考慮せずに、ファンアウト ペナルティを付けて、ネットごとにできるだけ最適な配線遅延を想定したデザインパフォーマンスが示されます。セットアップ違反の主な原因は、(1) デバイスまたは SLR の使用率が高い、(2) ロジック接続が複雑なために配置が密集している、(3) ロジック レベル数の多いパスが多くある、(4) バランス調整されていないクロック間のクロック スキューが大きい、またはクロックのばらつきが大きいなどの配置の問題にあります。phys_opt_design を Explore または AggressiveExplore モードで実行して、place_design 後の QoR (結果の品質) を改善します。解決できない場合は、まず配置 QoR の改善に集中します。

配線前 (WHS < -0.5 ns)

パフォーマンス目標が配線後に満たされず、配線前のワースト ネガティブ スラック (WNS) が正の値の場合は、見積もりワースト ホールド スラック (WHS) 違反を減らすようにしてみてください。配線前のホールド違反が少なく小さいと、ホールド タイム違反を修正するのに時間を費やさずにすむので、route_design で Fmax に集中しやすくなります。

配線後 (WNS < 0 ns または WHS < 0 ns)

route_design が終了したら、まずログ ファイルを確認するか、配線後のデザイン チェックポイント (DCP) で report_route_status を実行して、デザインが完全に配線されたことを確認します。配線違反や大きなセットアップ (WNS) またはホールド (WHS) 違反がある場合は、配線が密集しています。「セットアップ違反の解析」(3 ページ目)、「ホールド違反の解決」(4 ページ目)、および「密集削減手法」(6 ページ目)を使用して、問題を特定して解決します。route_design の後に phys_opt_design を実行して、小さなセットアップ違反 (> -0.200 ns) を解決します。デザイン、制約、およびコンパイル ストラテジのイテレーション中は、段階ごとに密集情報も含めた QoR を記録します。QoR の表を使用して、run 特性を比較し、残りのタイミング違反を解決する際の優先順位を決めます。

Impl. Run	opt_design			place_design			phys_opt_design				route_design				
	Directive	WNS		Directive	WNS	Congestion	Directive	WNS	WHS	THS	Directive	WNS	TNS	WHS	Congestion
Run1	ExploreWithRemap	0.034		WLDriVenBlockPlacement	-0.07	5-4-5-5	Explore	0.001	-0.409	-851.052	NoTimingRelaxation	-0.02	-1.68	0.006	5-5-4-5
Run2	Explore	0.054		AltSpreadLogic_medium	-0.368	6-5-5-5	Explore	-0.068	-0.364	-852.889	Default	-1.50	-3680.32	0.003	5-6-5-6
Run3	Default	0.054		AltSpreadLogic_high	-0.393	5-4-5-5	Explore	0.035	-0.364	-906.036	Explore	-1.37	-1495.19	0.006	4-5-5-6
Run4	Default	0.054		ExtraTimingOpt	-0.41	5-5-5-5	Explore	0.075	-0.407	-902.348	Explore	-1.23	-2896.42	0.001	5-5-5-6

ヒント: place_design 後および route_design 後に report_qor_suggestions を使用すると、デザイン、制約、ツール オプションの変更箇所が自動的に検出され、新しいコンパイルの QoR を改善するのに役立ちます。

X21575-091818

セットアップ違反の解析フロー

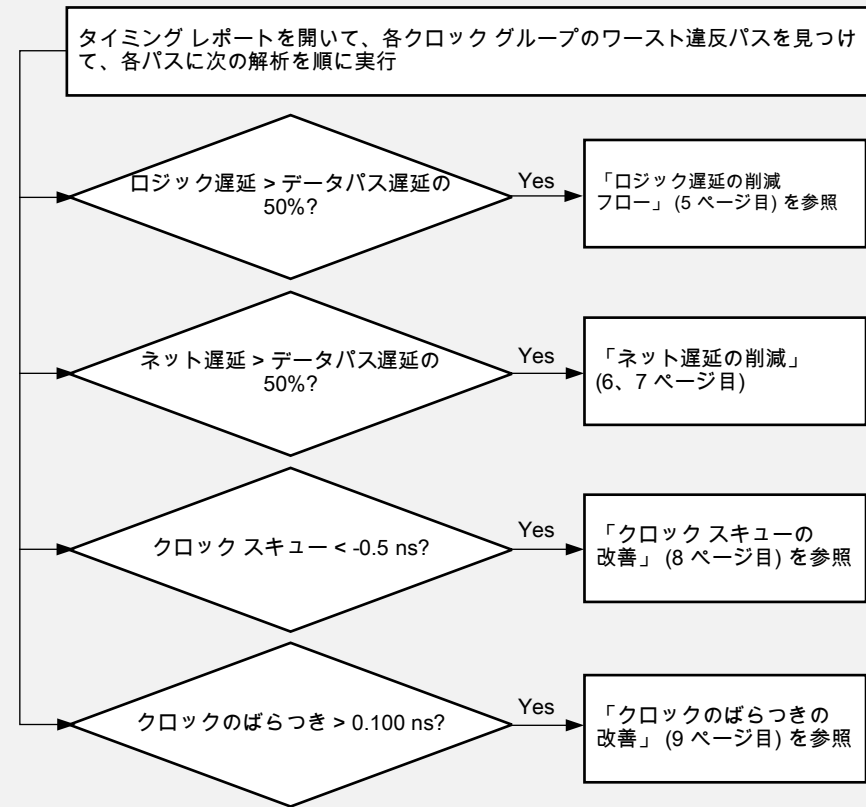
デザイン パフォーマンスは、次の要因によって決まります。

- クロック スキューおよびクロックのばらつき: クロックがどれくらい効率的にインプリメントされるか
- ロジック遅延: クロック サイクル中に移動するロジックの量
- ネットまたは配線遅延: Vivado インプリメンテーションによりデザインがどれくらい効率的に配置配線されるか

タイミング パスまたはデザイン解析レポートの情報を使用して次を実行します。

- これらのどれがタイミング違反の原因となっているかを特定
- QoR を改善する方法を決定

ヒント: 必要に応じて各段階後に DCP を開いて、その他のレポートを生成します。



X21576-091818

レポートからのセットアップ タイミング パス特性の検出

Vivado プロジェクト モードの場合、セットアップ タイミング パス特性を次のように見つけます。

- [Design Runs] ウィンドウで解析するインプリメンテーション run を選択します。
- [Implementation Run Properties] ウィンドウの [Reports] タブをクリックします。
- 選択したインプリメンテーション段階のタイミング サマリ レポートまたはデザイン解析レポートを開きます。
 - タイミング サマリ レポート: <runName>_<flowStep>_report_timing_summary (テキスト用は .rpt、Vivado IDE 用は .rpx)
 - デザイン解析レポート: <runName>_<flowStep>_report_design_analysis

Vivado の非プロジェクト モードまたは Vitis ソフトウェア プラットフォームの場合は、次のいずれかを実行します。

- インプリメンテーション run ディレクトリでレポートを開きます。
- Vivado IDE でインプリメンテーション DCP を開いて、RPX バージョンのレポートを開きます。

注記: Vivado IDE を使用すると、レポート、回路図、および [Device] ウィンドウ間をクロスプローブできます。

タイミング パスごとに、ロジック遅延、配線遅延、クロック スキュー、およびクロックのばらつき特性がパスのヘッダーに表示されます。

Summary	
Name	Path 4
Slack	-0.675ns
Source	inst_209033/inst_209021/inst_200956/inst_200920/inst_...
Destination	inst_209033/inst_209021/inst_200956/inst_200920/inst_...
Path Group	app_clk
Path Type	Setup (Max at Slow Process Corner)
Requirement	3.184ns (app_clk rise@3.184ns - app_clk rise@0.000ns)
Data Path Delay	3.505ns (logic 1.283ns (36.605%) route 2.222ns (63.395%))
Logic Levels	10 (CARRY8=2 LUT2=1 LUT4=1 LUT5=1 LUT6=5)
Clock Path Skew	-0.333ns
Clock Uncertainty	0.046ns

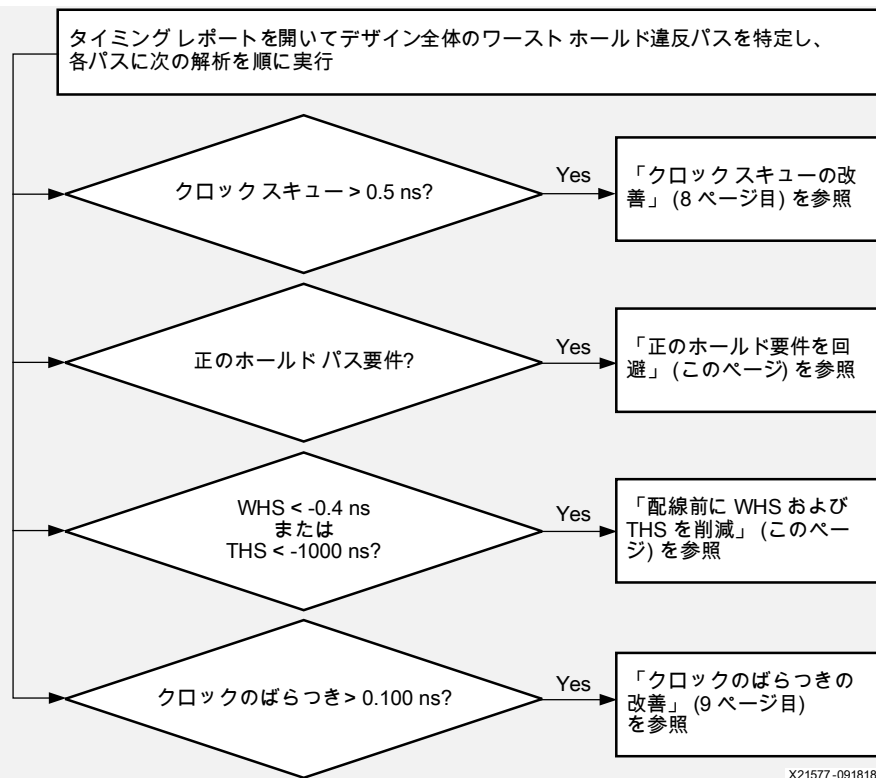
Slack (VIOLATED) :	-0.675ns (required time - arrival time)
Source:	inst_209033/inst_209021/inst_200956/inst_200920/inst_191361/...
Destination:	inst_209033/inst_209021/inst_200956/inst_200920/inst_188337/...
Path Group:	app_clk
Path Type:	Setup (Max at Slow Process Corner)
Requirement:	3.184ns (app_clk rise@3.184ns - app_clk rise@0.000ns)
Data Path Delay:	3.505ns (logic 1.283ns (36.605%) route 2.222ns (63.395%))
Logic Levels:	10 (CARRY8=2 LUT2=1 LUT4=1 LUT5=1 LUT6=5)
Clock Path Skew:	-0.333ns (DCD - SCD + CPR)
Destination Clock Delay (DCD):	2.884ns = (6.068 - 3.184)
Source Clock Delay (SCD):	3.380ns
Clock Pessimism Removal (CPR):	0.163ns
Clock Uncertainty:	0.046ns ((TSJA2 + DJA2)^1/2) / 2 + PE
Total System Jitter (TSJ):	0.071ns
Discrete Jitter (DJ):	0.060ns
Phase Error (PE):	0.000ns

同じタイミング パス特性はデザイン解析レポートの「Setup Path Characteristics」や、[Logic Levels] および [Routes] 列などにも表示されます。

Name	Slack	Requirement	Path Delay	Logic Delay	Net Delay	Clock Skew	Logic Levels	Routes	Logical Path
Path 1	-1.438	1.592	3.244	6%	94%	0.665	1	2	FDRE LUT4 FDRE
Path 2	-0.708	3.184	3.508	43%	57%	-0.362	5	5	RAMB18E2 LUT6 LUT6 LUT6 LUT6 LUT6 FDRE
Path 3	-0.683	3.184	3.483	42%	58%	-0.362	5	5	RAMB18E2 LUT6 LUT6 LUT6 LUT6 LUT6 FDRE
Path 4	-0.675	3.184	3.505	37%	63%	-0.333	10	8	FDRE LUT6 LUT6 LUT6 LUT5 LUT6 LUT2 CARRY8 CARRY8 LUT4 LUT6 FDRE

ヒント: テキスト モードでは、「Setup Path Characteristics」の列がすべて表示され、表が横に長くなります。Vivado IDE では、同じ表が見やすいように列数を減らして表示されます。表ヘッダーを右クリックすると、必要に応じて列を表示/非表示にできます。たとえば、DONT_TOUCH または MARK_DEBUG 列はデフォルトでは表示されません。これらは、飛ばされたロジック最適化解析に関する重要な情報で、このレポート以外では見つけるのが困難です。

ホールド違反の解決フロー



次は、クロック スキューが大きいホールド タイム パスの例です。

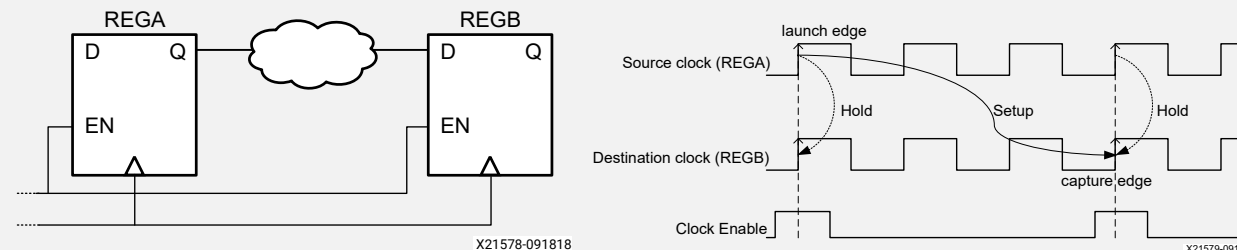
Summary	
Name	Path 259
Slack (Hold)	-1.129ns
Source	inst_209033/inst_381/inst_285879/inst_285870/inst_285584/i
Destination	inst_209033/inst_381/inst_285879/inst_285870/inst_285584/i
Path Group	app_clk
Path Type	Hold (Min at Slow Process Corner)
Requirement	0.000ns (app_clk rise@0.000ns - txoutclk_out[3]_3 rise@0.000ns)
Data Path Delay	0.180ns (logic 0.059ns (32.778%) route 0.121ns (67.222%))
Logic Levels	0
Clock Path Skew	1.247ns

ホールド違反の解決手法

正のホールド要件を回避

マルチサイクルパス制限を使用してセットアップ チェックを緩和するには、次を実行します。

- 同じパスのホールド チェックも調整して、同じソースおよびデスティネーション クロック エッジがホールド タイム解析でも使用されるようにします。このようにしないと、正のホールド要件 (1 または複数のクロック周期) になり、タイミング クロージャを達成できなくなります。
- セルまたはクロックだけでなく、終点ピンを指定します。たとえば、次の図では終点セル REGB に C、EN、D の 3 つの入力ピンがありますが、マルチサイクルパス例外制限を設定する必要があるのは REGB/D ピンだけです。クロック イネーブル (EN) ピンは、各クロック サイクルで変更される可能性があるため、制限は設定しません。制限をピンではなくセルに適用すると、EN ピンを含むすべての有効な終点ピンがその制限に対して考慮されます。



常に次の構文を使用することをお勧めします。

```

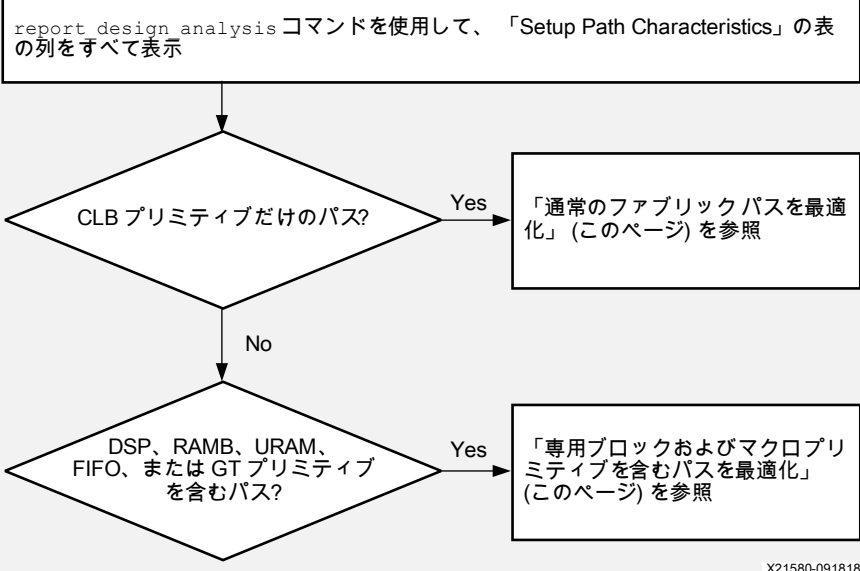
set_multicycle_path -from [get_pins REGA/C] -to [get_pins REGB/D] -setup 3
set_multicycle_path -from [get_pins REGA/C] -to [get_pins REGB/D] -hold 2
  
```

配線前に WHS および THS を削減

見積もられたホールド違反が大きいと、配線での問題が増え、route_design で解決できないこともあります。配置後の phys_opt_design コマンドには、ホールド違反を修正する複数のオプションがあります。

- 順次エレメント間に反対のエッジでトリガーされるレジスタを挿入すると、タイミングパスが 1/2 周期のパス 2 つに分割され、ホールド違反を大幅に削減できます。この最適化は、セットアップ タイミングが悪化しない場合にのみ実行されます。次のコマンドを使用します。
phys_opt_design -insert_negative_edge_ffs
- LUT1 バッファを挿入するとデータパスが遅延され、セットアップ違反を導入せずにホールド違反を削減できます。次のコマンドを使用します。
 - phys_opt_design -hold_fix: WHS 違反が最大のパスにのみ LUT1 を挿入します。
 - phys_opt_design -aggressive_hold_fix: より多くのパスに LUT1 を挿入するので、トータル ホールド スラック (THS) を大幅に削減できますが、LUT 使用率はかなり増え、コンパイル時間も長くなります。このオプションは、どの phys_opt_design 指示子とでも一緒に使用できます。
 - phys_opt_design -directive ExploreWithAggressiveHoldFix: LUT1 を挿入してホールド違反を修正するだけでなく、Fmax を改善するためのその他の物理最適化もすべて実行します。

ロジック遅延の削減フロー



X21580-091818

Vivado インプリメンテーションでは、まず最もクリティカルなパスに焦点が置かれますが、これにより配置後または配線後にそれほど困難でなかったパスがクリティカルになることがあります。合成後または opt_design 後に最長のパスを特定して向上することをお勧めします。これは QoR に最も大きく影響するので、通常タイミング クロージャを達成するまでの配置配線の実行回数を大幅に削減できます。report_design_analysis の「Logic Level Distribution」の表から、ロジックレベル分布と要件を比較して、最初に改善する必要のある部分を特定します。要件が低いほど、許容されるロジックレベル数は少なくなります。たとえば、次の配置前のロジックレベル分布レポートで次を確認します。

- txoutclk_out[0]_4: ロジックレベル数が 8 以上のパスすべて
- app_clk: ロジックレベル数が 11 以上のパスすべて

End Point Clock	Requirement	0	1	2	4	5	6	7	8	9	10	11	12	13	14	15	16
app_clk	3.184ns	0	0	0	1	0	0	135	16	37	30	16	16	16	16	15	7
txoutclk_out[0]_4	2.388ns	2	0	0	64	784	1677	0	9	3	0	3	4	0	0	0	0
txoutclk_out[3]_3	1.592ns	2100	5029	20	0	0	0	0	0	0	0	0	0	0	0	0	0

注記: カスケード接続された CARRY または MUXF セルはロジックレベル数を増加させる可能性があります。遅延への影響はあまりありません。

ヒント: Vivado IDE レポートでロジックレベル数をクリックしてパスを選択し、F4 を押すと、回路図が生成され、そのロジックを確認できます。

ロジック遅延の削減手法

通常ファブリックパスを最適化

通常ファブリックパスは、レジスタ (FD*) またはシフトレジスタ (SRL*) 間のパスで、LUT、MUXF、および CARRY の組み合わせを通過します。通常ファブリックパスに関する問題が発生した場合は、『Vivado Design Suite ユーザー ガイド: 合成』(UG901) および『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) を参照してください。

- ロジック段数が多い場合、パスは report_qor_assessment を使用して LUT/ネット バジェット チェックで識別できます。RTL の再コード化またはリタイミングを使用して、デザイン サイクルの早期段階でロジック段数の多さに対処してください。¹
推奨: 合成で -retiming をグローバルに使用します。ブロック合成ストラテジ BLOCK_SYNTH.RETIMING 1 を使用して、モジュールをターゲットにするか、RETIMING_FORWARD/BACKWARD を使用して特定のセルをターゲットにします。
- 小型のカスケード LUT (LUT1 ~ LUT4) は、デザイン階層、ファンアウトが 10 以上の中間ネット、あるいは KEEP、KEEP_HIERARCHY、DONT_TOUCH、または MARK_DEBUG プロパティの使用により妨害されている場合以外は、より少ない LUT に統合できます。¹
推奨: これらのプロパティを削除して、合成または opt_design -remap からやり直してみてください。
- 1 つの CARRY (カスケードなし) セルがあると、LUT 最適化が制限され、配置が最適なものにならないことがあります。
推奨: FewerCarryChains 合成指示子を使用するか、セルに CARRY_REMAP プロパティを設定して opt_design で削除されるようにします。
- シフトレジスタ SRL* 遅延がレジスタ FD* 遅延よりも大きい場合、SLR の配置が FD の配置よりも最適でない可能性があります。
推奨: RTL で SRL_STYLE 属性を使用するか、合成後のセルに SRL_STAGES_TO_INPUT または SLR_STAGES_TO_OUTPUT プロパティを使用して、SRL の入力または出力からレジスタを取り出します。ダイナミック SRL は RTL で変更する必要があります。
- ロジックパスがファブリックレジスタ (FD*) のクロック イネーブル (CE)、同期セット (S)、または同期リセット (R) ピンで駆動される LUT で終了すると、特にパスの最後のネットのファンアウトが 1 よりも大きい場合は、配線遅延がレジスタのデータピン (D) よりも大きくなります。
推奨: データピン (D) で終わるパスの方がスラックが大きく、ロジックレベル数も少なくなる場合は、RTL でその信号の EXTRACT_ENABLE または EXTRACT_RESET 属性を no に設定します。または、セルに CONTROL_SET_REMAP プロパティを設定して、opt_design 中に同じ最適化がトリガーされるようにします。

ヒント: 合成で -retiming をグローバルに使用するか、モジュールにブロック合成ストラテジを使用します (例: BLOCK_SYNTH.RETIMING=1)。

専用ブロックおよびマクロプリミティブを含むパスを最適化

専用ブロックおよびマクロプリミティブ (DSP、RAMB、URAM、FIFO、または GT_CHANNEL など) で開始または終了するロジックパス、あるいはその間のロジックパスは、配置するのがさらに困難で、セルおよび配線遅延が大きくなるので、マクロプリミティブの周りにパイプラインを追加するか、マクロプリミティブパスのロジックレベル数を減らして、デザインパフォーマンス全体を改善します。

RTL を変更する前に、オプションの DSP、RAMB、URAM レジスタすべてをイネーブルにしてインプリメンテーションを実行し直し、パイプラインを追加することで QoR が改善するかどうかを検証します。この評価方法を使用する際は、ビットストリームを生成しないでください。次に例を示します。

```
set_property -dict {DOA_REG 1 DOB_REG 1} [get_cells xx/ramb18_inst]
```

次の RAMB18 パスの例では、追加のパイプラインレジスタまたはロジックレベル数の削減が必要です (route_design 後にレポート)。

Name	Slack	Requirement	Path Delay	Logic Delay ¹	Net Delay	Logic Levels	Routes	Logical Path	BRAM
↳ Path 5	-0.663	3.184	3.472	48%	52%	5	5	RAMB18E2 LUT6 LUT6 LUT6 LUT6 LUT6 FDRE	No DO_REG

¹report_qor_suggestions を使用して自動的に解決されます。

ネット遅延の削減フロー 1

グローバルな密集は、次のようにデザイン パフォーマンスに影響します。

- **レベル 4 (16x16):** route_design 中の QoR のばらつきは少ない。
- **レベル 5 (32x32):** 配置が最適ではなくなり、QoR が顕著にばらつく。
- **レベル 6 (64x64):** 配置配線が困難で、コンパイル時間が長くなる。タイミング QoR は、パフォーマンス目標が低くない限り、大きく低下します。
- **レベル 7 (128x128) 以上:** 配置配線が不可能。

route_design コマンドを実行すると、密集レベル 4 以上の場合、ログファイルに「Initial Estimated Congestion」の表が含まれます。

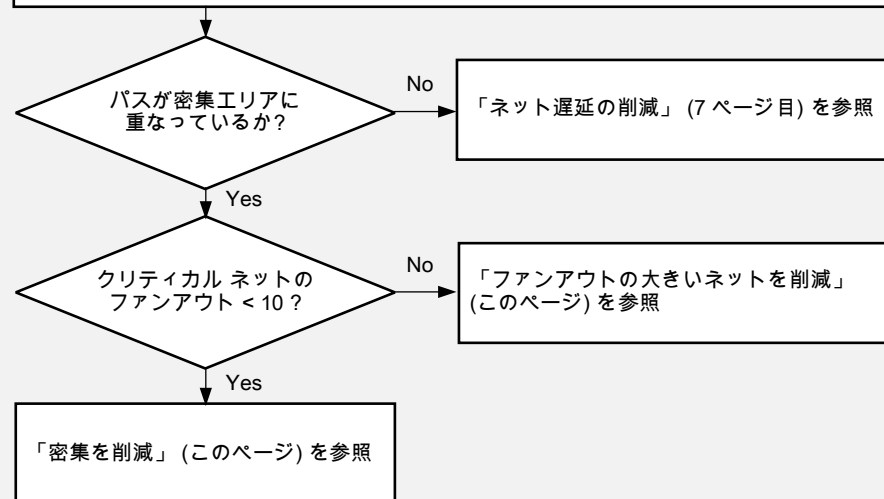
配置および配線密集情報の両方をレポートするには、

report_design_analysis -congestion を使用します。

ヒント: 配置後または配線後の DCP を開き、インタラクティブな

report_design_analysis ウィンドウを Vivado IDE で開いて、クロスプローブで密集したエリアをハイライトして個々のロジックパスの配置配線への影響を確認します。詳細は、UG949 の「密集によるネット遅延の削減」を参照してください。

密集レベルが 4 以上の場合、Vivado IDE でデザイン チェックポイントを開いて [Device] ウィンドウで密集マトリクスを表示し、タイミング パスをマークしてパスの配置配線を解析



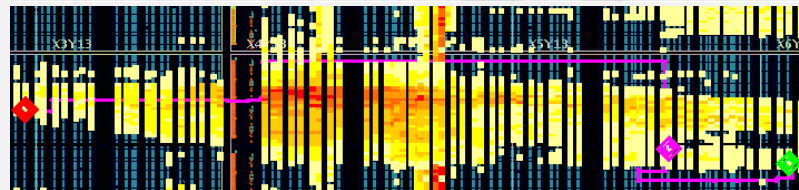
X21581-091818

密集削減手法

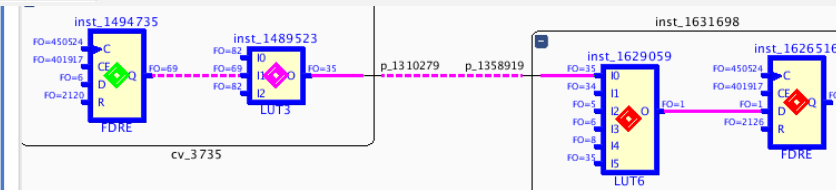
次は、ネット配線が密集エリアを迂回しているためにネット遅延が増加したクリティカル タイミング パスの例です。

- デザイン解析レポートからすべてのビューにアクセスできます。
- [Device] ウィンドウで [Interconnect Congestion Level] をオンにします。

Name	Slack	Requirement	Path Delay	Logic Delay	Net Delay	High Fanout	Logical Path	IO Crossings
Path 5	-0.230	4	3.839	6%	94%	69	FDRE LUT3 LUT6 FDRE	1



Delay Type	Incr (ns)	Path...	Location	Netlist Resource(s)
FDRE (Prop_EFF_SLICEM_C_Q)	(f) 0.076	4.883	Site: SLICE_X179Y744	inst_1784100/inst_174187
net (fo=69, routed)	1.175	6.058		inst_1784100/inst_174187
LUT3 (Prop_D6LUT_SLICEM_I1_O)	(r) 0.050	6.108	Site: SLICE_X163Y749	inst_1784100/inst_174187
net (fo=35, routed)	2.374	8.482		inst_1784100/inst_174187
LUT6 (Prop_C6L...SLICEM_I0_O)	(r) 0.098	8.580	Site: SLICE_X94Y761	inst_1784100/inst_174187
net (fo=1, routed)	0.066	8.646		inst_1784100/inst_174187



密集を削減

密集を削減するには、次の手法を順に実行します。

- 全体的なリソース使用率が 70 ~ 80% を超える場合、デザイン機能を一部削除するか、モジュールまたはカーネルの一部を別の SLR に移動して、デバイスまたは SLR の使用率を下げます。LUT と DSP/RAMB/URAM の使用率が同時に 80% を超えないようにします。マクロプリミティブの使用率 (%) を高くする必要がある場合、LUT 使用率を 60% 未満に抑えると、複雑なフロアプラン制約を使用しなくても、密集エリアで配置を分散させることができます。report_qor_assesment を使用して配置後の各 SLR の使用率を確認します。
- 密集領域のクリティカルでないファンアウトの大きなネットをグローバル クロック配線に次のようにプロモートします。¹
set_property CLOCK_BUFFER_TYPE BUFG [get_nets <highFanoutNetName>]
- 合成で複製されたネットを統合して、密集エリアの等価ネットの重複を削減します。RTL および合成 XDC から MAX_FANOUT プロパティを削除するか、ターゲットセルで set_property EQUIVALENT_DRIVER_OPT merge を使用します。¹
- 複数の配置指示子 (AltSpreadLogic* や SSI_Spread* など)、Congestion_* インプリメンテーション実行ストラテジまたは ML ストラテジなど試してみます。
- 密集エリアでの MUXF* および LUT の組み合わせを減らします。RDA 密集レポートの該当する列を参照してください。密集した最下位セルで MUXF_REMAP を 1 に SOFT_HLUTNM を "" に設定します。report_qor_suggestions を使用すると役立ちます。¹
- report_design_analysis -complexity -congestion を使用して、接続が複雑 ([Rent Exponent] > 0.65 または [Average Fanout] > 4) で、15,000 セルを超える大きな密集モジュールを見つけます。密集を解決するための合成設定を使用します (XDC ファイルに追加)。
- 密集度の低かった前のインプリメンテーション run からの DSP、RAMB、および URAM 配置制約を再利用します。次に例を示します。
read_checkpoint -incremental routed.dcp -reuse_objects [all_rams] -fix_objects [all_rams]

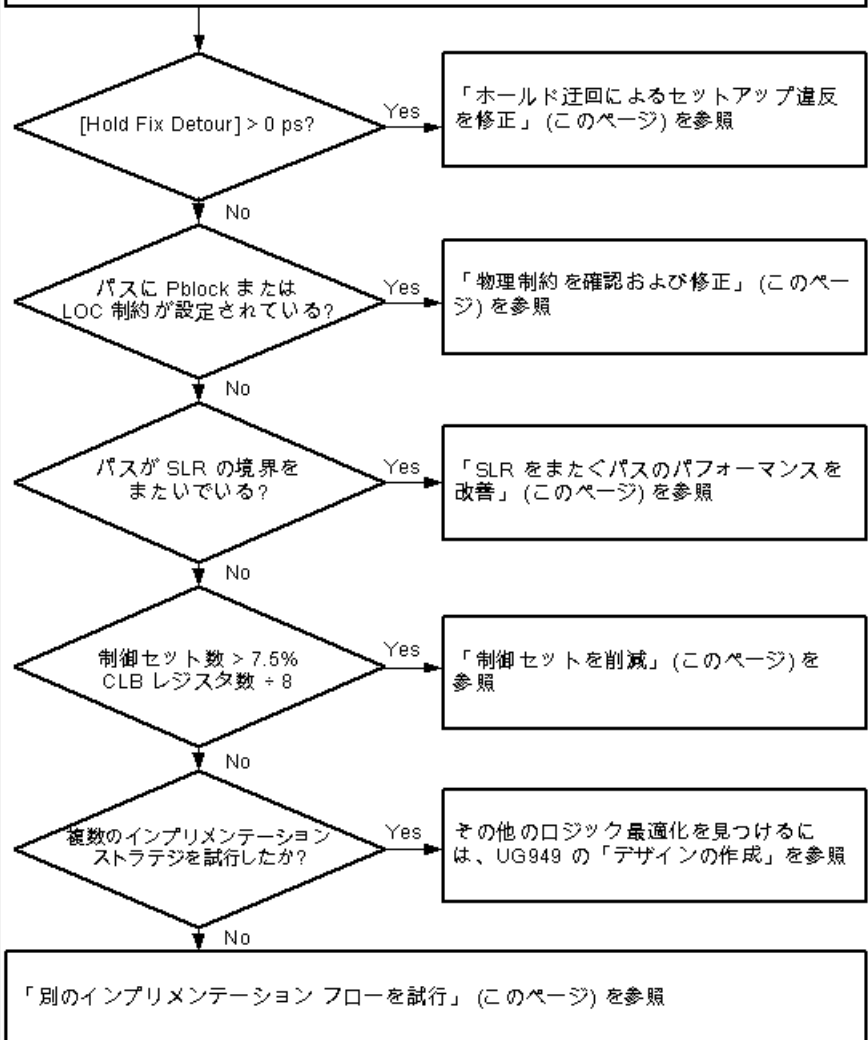
ファンアウトの大きいネットを最適化

- RTL で階層ベースのレジスタ複製を明示的に指定するか、次のロジック最適化を使用します。
opt_design -merge_equivalent_drivers -hier_fanout_limit 512
- 配線前の物理的最適化のステップを増やしてクリティカルなファンアウトの大きいネットで複製を実行します。
set_property FORCE_MAX_FANOUT¹

1.report_qor_suggestions を使用して自動的に解決されます。

ネット遅延の削減フロー 2

report_design_analysis コマンドを使用し、「Setup Path Characteristics」の表の列すべてを表示し、report_utilization または report_qor_assesment を使用して、配置後の制御信号数を取得



X21582-091818

ネット遅延の削減手法

ホールド迂回によるセットアップ違反を修正

デザインがハードウェアで動作するようにするには、ホールド違反をセットアップ違反(または Fmax)よりも優先して修正する必要があります。次の例は、セットアップ要件の厳しいスキューの大きい 2 つの同期クロック間のパスを示しています。

Name	Slack	Requirement	Path Delay	Clock Skew	Hold Fix Detour	Logical Path	Start Point Clock	End Point Clock	SLR Crossings
↳ Path 1	-1.438	1.592	3.244	0.665	1181	FDRE LUT4 FDRE	txoutclk_out[3]_3	app_clk	1

注記: [Hold Fix Detour] の単位はピコ秒です。ホールド迂回の Fmax への影響に対処するには、「ホールド違反の解決手法」(4 ページ目)を参照してください。

物理制約を確認および修正

すべてのデザインに物理制約が含まれます。I/O ロケーションは通常変更できませんが、デザインを変更する際には、Pblock およびロケーション制約は注意して検証する必要があります。変更により、ロジックが離れたり、ネット遅延が大きくなってしまふことがあります。Pblock が複数含まれるパス ([PBlocks] 列) およびロケーション制約が含まれるパス ([Fixed Loc] 列)を確認します。

SLR をまたぐパスのパフォーマンスを改善

スタックドシリコン インターコネクト (SSI) テクノロジ デバイスをターゲットにする場合は、早期に次の点を考慮すると、パフォーマンスを改善できます。

- 主なデザイン階層またはカーネルの境界にパイプラインレジスタを追加すると、長距離および SLR をまたぐパスの配線をしやすくなります。
- 各 SLR 使用率がガイドラインの範囲内かどうかを確認します (report_qor_assesment を使用)。
- USER_SLR_ASSIGNMENT 制約を使用してインプリメンテーション ツールをガイドします。詳細は、UG949 の「ソフト SLR フロアプラン制約の使用」を参照してください。
- ソフト制約で改善しない場合は、SLR Pblock 配置制約を使用します。
- 配置後または配線後に phys_opt_design -slr_crossing_opt を使用します。

制御セットを削減

制御セット数がガイドライン (7.5%) を超える場合は、デバイス全体または各 SLR で次を実行します。

- RTL でクロック イネーブル、セット、リセット信号の MAX_FANOUT 属性を削除します。¹
- 最小の合成制御信号のファンアウトを増加します (例: synth_design -control_set_opt_threshold 16)。¹
- opt_design -control_set_merge または -merge_equivalent_drivers を使用して複製した制御信号を統合します。
- CLB レジスタセルに CONTROL_SET_REMAP プロパティを設定して、ファンアウトの小さな制御信号を LUT にリマップします。¹

別のインプリメンテーション フローを試行

デフォルトのコンパイルフローでは、すばやくベースライン制約を取得し、タイミングが満たされているかどうかを解析し始めることができます。最初のインプリメンテーションでタイミングが満たされない場合は、その他の推奨フローを試してください。

- 複数の place_design 指示子 (最大 10) および複数の phys_opt_design 反復 (Aggressive*、Alternate* 指示子)を試します。
- set_clock_uncertainty を使用して place_design/phys_opt_design 中に最もクリティカルなクロックの制約を (最大 0.500 ns まで) 厳しくします。
- group_path -weight を使用してタイミングを満たす必要のあるタイミングクロックのタイミング QoR の優先度を上げます。
- デザインを少し変更する場合は、インクリメンタル コンパイル フローを使用して QoR を保持したままランタイムを削減します。
- report_qor_suggestions で生成されたデザインの上位 3 つのインプリメンテーション ML ストラテジを実行します。

¹report_qor_suggestions を使用して自動的に解決されます。

クロックのスキューの改善フロー

report_design_analysis コマンドを使用し、「Setup Path Characteristics」の表の列すべてを表示し、report_clock_utilization を使用して (オプション)、クロック ネットの既存制約を確認

[Clock Relationship] が [Safely Timed] になっているか?

No 「非同期クロック間にタイミング例外を追加」(このページ)を参照

パスはバランス調整されたクロック間にあるか?

No 「クロック ツリーで使用されるロジックをクリーンアップ」(このページ)を参照

スキュー < 0.5 ns?

No 「クロック配線のマッチング」(このページ)を参照

クロックは I/O およびファブリックセルに接続されているか?

Yes 「クロック ロードの配置を関連する I/O バンクの隣に制約」(このページ)を参照

データパスは SLR の境界または I/O 列をまたいでいるか?

Yes 「クロック ロードの配置をより小さなエリアに制約」(このページ)を参照

No 「物理ソースを移動してクロック ネット遅延を削減」(このページ)を参照


X2 1663-0918 18

クロックのスキューの改善手法

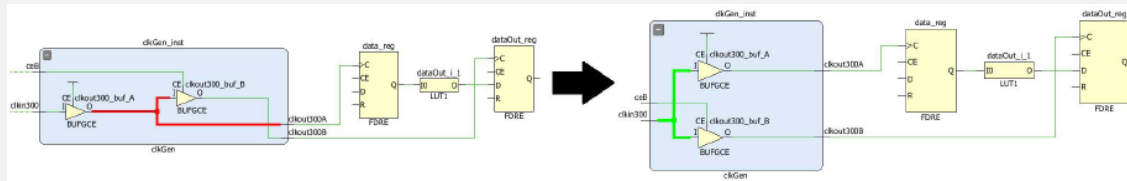
非同期クロック間にタイミング例外を追加

ソースクロックとデスティネーションクロックが異なるプライマリクロックから供給されているタイミングパスまたは共通ノードのないタイミングパスは、非同期クロックとして扱う必要があります。この場合、スキューが極端に大きくなり、タイミングクロージャを達成するのが不可能になります。set_clock_groups、set_false_path および set_max_delay -datapath_only 制約を必要に応じて追加します。詳細は、UG949 の「非同期クロック間にタイミング例外を追加」を参照してください。

クロック ツリーで使用されるロジックをクリーンアップ

opt_design コマンドは、クロックロジックに DONT_TOUCH 制約が使用されていない場合は、クロックツリーを自動的にクリーンアップします。タイミングパスを選択し、[Clock Path Visualization] ツールバー ボタン  をクリックし、回路図を開いて (F4) クロックロジックを確認します。

- 不要なバッファを削除するか、バッファを並列に接続することにより、カスケード接続されたクロックバッファ間にタイミングパスがないようにします。次に例を示します。



- クロックが同等である場合は、並列のクロックバッファを 1 つのクロックバッファにまとめます。
- クロックパスに LUT または組み合わせロジックがあると、クロック遅延とクロックスキューが予測できなくなるので、削除します。

クロック配線のマッチング

CLOCK_DELAY_GROUP を使用すると、2 つのクロック ネットに既に同じ CLOCK_ROOT が使用されている場合でも、クリティカルな同期クロック間のクロック配線遅延マッチングを改善できます。次の例は、CLOCK_DELAY_GROUP のない 2 つの同期クロックを示しています。

Global Clock Resources										
Global Id	Source Id	Driver Type/Pin	Constraint	Site	Clock Region	Root	Load Clock Region	Clock Loads	Clock Period	Clock
g0	src0	BUFG_GT/O	None	BUFG_GT_X1Y212	X5Y8	CLOCK_REGION_X5Y6	30	110934	3.184	app_clk
g1	src0	BUFG_GT/O	None	BUFG_GT_X1Y215	X5Y8	CLOCK_REGION_X5Y6	2	5202	1.592	txoutclk_out[3]_3

クロック ロードの配置を関連する I/O バンクの隣に制約

I/O ロジックとファブリックセル間のクロックセルのロードが 2000 未満の場合、クロック ネットに CLOCK_LOW_FANOUT プロパティを設定すると、クロックバッファ (BUFG*) と同じクロック領域内のロードがすべて自動的に配置され、挿入遅延とスキューを抑えることができます。¹

クロック ロードの配置をより小さなエリアに制約

Pblock を使用すると、クロック ネットロードの配置をより小さなエリア (1 つの SLR など) に制約して、挿入遅延とスキューを減らし、スキューペナルティの原因となる I/O 列などの特殊な列をまたがないようにできます。

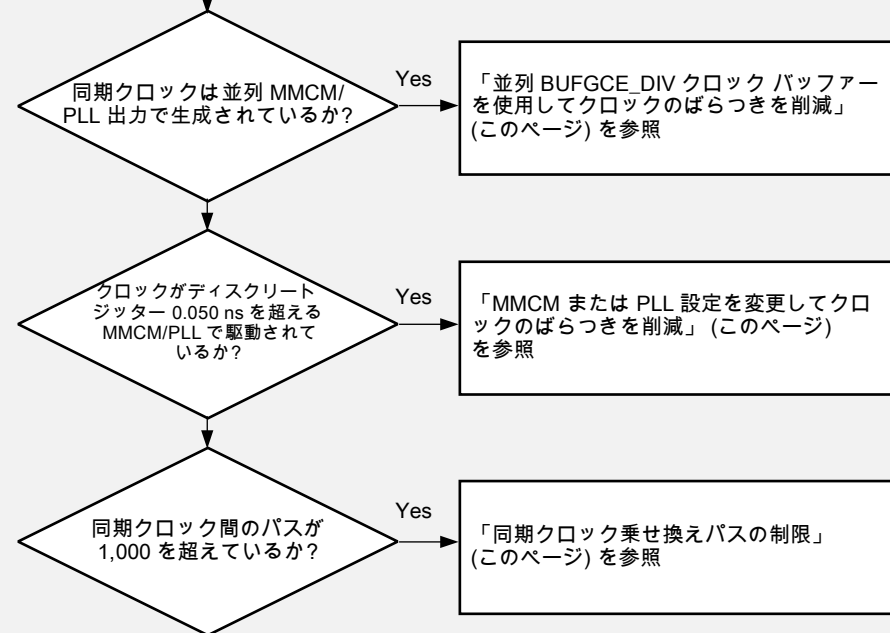
物理ソースを移動してクロック ネット遅延を削減

ロケーション制約を使用してソースの MMCM (混合モードクロックマネージャー) または PLL (位相ロックループ) をクロックロードの中央に移動すると、最大クロック挿入遅延を削減して、クロックの不必要に悪い見積り部分およびスキューを低減できます。詳細は、UG949 の「UltraScale および UltraScale+ デバイスでのスキューの向上」を参照してください。

¹ report_qor_suggestions を使用して自動的に解決されます。

クロックのばらつきとは、ハードウェア動作条件を正確にモデリングするために、理想的なクロック エッジに追加する入力ジッター、システム ジッター、ディスクリート ジッター、位相エラー、またはユーザーが追加したばらつきの量の事です。クロックのばらつきは、セットアップおよびホールド タイミング パスの両方に影響し、クロック ツリーで使用されるリソースによって異なります。

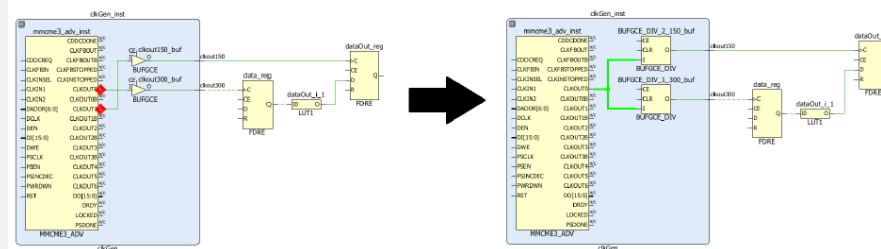
タイミング レポートを開き、各クロック グループのワースト違反パスを見つけ、各パスに対して次の手順を実行



X21581-091818

並列 BUFGE_DIV クロック バッファーを使用してクロックのばらつきを削減

同じ MMCM または PLL で生成され、複数のクロック出力で駆動される周期率 2、4、または 8 の同期クロックの場合は、MMCM または PLL 出力を 1 つだけ使用して、並列の BUFGE_DIV クロック バッファーに接続します (AMD UltraScale™ および AMD UltraScale+™ デバイスのみ)。このクロック トポロジを使用すると、クロックのばらつき (ほとんどの場合約 0.120 ns) の原因となっていた MMCM または PLL 位相エラーが発生しなくなります。



次は、150 MHz クロックおよび 300 MHz クロック間のクロック乗せ換え (CDC) パスのクロックのばらつきを削減する例です。

- 前: 0.188 ns (セットアップ)、0.188 ns (ホールド)
- 後: 0.068 ns (セットアップ)、0.000 ns (ホールド)

Clocking Wizard を使用して並列 BUFGE_DIV バッファーを含むクロック トポロジを生成し、クロックに CLOCK_DELAY_GROUP プロパティを設定します。

MMCM または PLL 設定を変更してクロックのばらつきを削減

MMCM や PLL などのクロック調整ブロックは、ディスクリートジッター、位相エラーなどを発生させ、クロックのばらつきの原因となります。¹

- Clocking Wizard または set_property コマンドで M (乗数) および D (除数) 値を変更して電圧制御オシレーター (VCO) 周波数を増加します。たとえば、MMCM (VCO = 1 GHz) の場合は 167 ps ジッター + 384 ps 位相エラーが発生しますが、MMCM (VCO = 1.43 GHz) の場合は 128 ps ジッター + 123 ps 位相エラーが発生します。
- PLL の方がクロックのばらつきは少なくなるので、可能であれば MMCM ではなく PLL を使用してください。

同期クロック乗せ換えパスの制限

別のクロック バッファで駆動される同期クロック間のタイミング パスでは、共通のクロック ツリーのノードがクロック バッファの前にあり、スキューが大きくなるので、タイミング解析で不必要に悪い見積り部分が大きくなってしまいます。このため、特にクロック周波数が高い (500 MHz 以上) 場合など、これらのパスでセットアップ要件とホールド要件の両方を同時に満たすのが困難になります。2 つのクロック間のパス数は、report_timing_summary ([Inter-Clock Paths] セクション) または report_clock_interactions でわかります。次の例は、2 つの高速クロック (要件 = 1.592 ns) 間に多数のパスが含まれるデザインです。これらのパスの 30% がタイミングを満たせず、非常にインプリメントしにくくなっていることがわかります。

Source Clock	Destination Clock	WNS (ns)	TNS (ns)	Failing Endpoints (TNS)	Total Endpoints (TNS)	Path Req (WNS)	Inter-Clock Constraints
app_clk	txoutclk_out[3]_3	-0.348	-162.119	1668	5623	1.592	Timed
rxoutclk_out[0]_4	rxoutclk_out[0]_4	0.262	0.000	0	2998	2.388	Partial False Path
pcie_refclk	pcie_refclk	5.054	0.000	0	1508	7.960	Timed
txoutclk_out[3]_3	app_clk	-0.153	-0.196	2	1198	1.592	Partial False Path

クロック乗せ換えに関連するロジックを確認し、不必要なロジック パスを削除するか、次のように変更します。

- 新規データはサイクルごとに転送されないため、クロック イネーブルで制御されるパスにマルチサイクル パス制約を追加します。
- クロック乗せ換えロジックを非同期クロック乗せ換え回路と適切なタイミング例外に置換します (レイテンシは長くなります)。たとえば、非同期 FIFO または XPM_CDC パラメーター指定マクロを使用します。詳細は、『UltraScale アーキテクチャライブラリ ガイド』(UG974) を参照してください。

1.report_gor_suggestions を使用して自動的に解決されます。

QoR 評価レポートの概要

QoR 評価レポートには、次のセクションが含まれます。

1. **[Overall Assessment Summary]:** QoR 評価スコアと QoR 改善のための推奨項目が表示されます。
2. **[QoR Assessment Details]:** 各項目ごとに QoR 情報が表示されます。スコアが 5 より低い項目があると、[Status] 列に REVIEW と表示されます。
ヒント: REVIEW ステータス項目のリスクを判断するには、[Thresh] 列 (しきい値) と [Actual] 列のデータを比較します。しきい値は、デザインおよびターゲット デバイスに合わせて自動的に調整されます。チェックに合格した項目を表示するには、-full_assessment_details オプションを使用します。
3. **[Methodology Check Details]:** 設計手法に関連し、QoR に影響する、エラーになった項目が表示されます。
4. **[ML Strategy Availability]:** ML ストラテジを生成するトレーニング run に必要な指示子をリストします (詳細は表示されません)。

1. Overall Assessment Summary

QoR Assessment Score	3 - Design runs have a small chance of success
Flow Guidance	Run report_methodology and fix or waive critical warnings

2. QoR Assessment Details

Name	Thresh.	Actual	Used	Available	Status
Utilization					OK
Clocking					OK
Congestion					OK
Timing					
WNS	-0.100	-0.330	-	-	REVIEW
TNS	-0.100	-29.398	-	-	REVIEW
Paths above Net/LUT Budgeting	0	132	-	-	REVIEW

3. Methodology Check Details

ID	Description	Criticality	No. Vio.
TIMING-1	Invalid clock waveform on Clock Modifying Block	Critical Warning	6
TIMING-9	Unknown CDC Logic	Warning	1

4. ML Strategy Availability

Conditions for ML Strategy Availability	Value	Status
opt_design directive	Explore	OK
place_design directive		-
phys_opt_design directive		-
route_design directive		-

QoR 評価レポートの詳細

QoR 評価スコア

QoR 評価スコアでは、デザインがタイミング目標を達成する可能性が次のように見積もられます。

- **1:** デザインはインプリメンテーションを完了しません。
- **2:** デザインはインプリメンテーションを完了しますが、タイミングは満たされません。
- **3:** デザインはタイミングを満たさない可能性があります。
- **4:** デザインはタイミングを満たす可能性が高くあります。
- **5:** デザインはタイミングを満たします。

さまざまな設計段階での QoR 評価スコアの精度

フロー前半に QoR 評価スコアを使用すると、コンパイル時間を最も節約できます。ただし、フローの段階によって、解析の精度レベルは異なります。report_qor_assessment コマンドは、フロー後半の最適化処理も含めて、デザインの現在の状態に応じて解析が自動的に調整されます。デザインの各状態で実行される解析は次のとおりです。

- **[Unplaced]:** セル使用率、クロック スキューしきい値の増加、および LUT/ネット バジエット チェックの解析。密集解析はなし。
- **[Placed]:** 密集解析およびより厳密なクロック スキューしきい値の解析。LUT/ネット バジエット チェックはなし。
- **[Routed]:** 完全に正確な解析としきい値。

注記: スコアは、約 ±1 の精度内の最適なメトリックに基づいて算出されます。

QoR 評価スコアの改善

report_qor_assessment の後に report_qor_suggestions を実行すると、コンパイルエラーのリスクを修正または低減する方法が推奨されます。推奨項目が存在する場合は、評価スコアが低い項目 (QoR 評価レポートで REVIEW ステータスになっているもの) が自動的に QoR 推奨項目レポートの上位に表示されます。

SLR および Pblock 解析

SLR および Pblock 解析は自動的に実行されます。スコアが 5 より低い項目のみがレポートされます。

追加解析の実行

-csv_output_dir オプションを使用して、さらに詳細な解析に使用可能な次の CSV ファイルを出力します。

- qor_timing_<design_stage>.csv: ネットおよび LUT のバジエット チェックでエラーになったタイミングパスを詳細に示します。
- qor_dont_touch_<design_stage>.csv: DONT_TOUCH プロパティを持つネットだけでなく、リーフ/階層セルもリストされます。

ヒント: プロジェクトモードの場合は、次の Tcl コマンドを使用して評価レポートを追加します。

```
set_property STEPS.OPT_DESIGN.TCL.POST <path>/postopt.tcl [get_runs impl_*]
```

次に postopt.tcl の例を示します。

```
report_qor_assessment -file postopt_rqa.rpt -csv_output_dir ./rqa
```