Vivado Design Suite
ユーザー ガイド

階層デザイン

UG905 (v2017.2) 2017 年 7 月 26 日

XILINX
ALL PROGRAMMABLE™

改訂履歴

次の表に、この文書の改訂履歴を示します。

日付	バージョン	改訂内容
2017 年 7 月 27 日	2017.2	ザイリンクス トレーニング コースへのリンクを更新。
2017 年 4 月 5 日	2017.1	「概要」に、この資料は 7 シリーズ デバイス ファミリのみに該当することを示す重要注記を追加。 「既知の問題」から UltraScale™ デバイスに関する記述を削除。

目次

Vivado 階層デザイン

概要

階層デザイン (HD) フローを使用すると、デザインを管理しやすい小型のブロックに分割して個別に処理できます。Vivado® Design Suite では、この分割されたパーティション モジュールを残りのデザインから独立させて (アウト オブ コンテキスト OOC) インプリメンテーションできます。Vivado Design Suite での設計手法は、次のとおりです。

- **モジュール解析**: モジュールを残りのデザインから独立させて、リソース使用量解析およびタイミング解析を実行します。ラッパーまたはダミー ロジックは必要ありません。モジュールのみを合成、最適化、配置配線します。フル デザインの場合と同様に、リソース使用量解析を実行し、タイミング レポートを調べ、配置結果を確認します。

  モジュール解析フローでは、パーティション モジュールまたは IP コアはデザインの最上位から独立させて (OOC で) インプリメントします。モジュールは特定のパーツ/パッケージの組み合わせでデバイスの決まった位置にインプリメントされます。I/O バッファー、グローバル クロック、およびその他のチップ レベルのリソースは挿入されませんが、モジュール内にインスタンシエートすることはできます。この OOC インプリメンテーションの結果は、デザイン チェックポイント (DCP) ファイルとして保存できます。

- **モジュール再利用**: モジュール解析フローからの配置配線済みモジュールを最上位デザイン内で使用し、検証された結果をロックします。デザインの特定のセクションでタイミング クロージャおよびその他の目標を達成したら、結果をそのまま再利用できます。

  この OOC モジュールを再利用する場合、接続ロジックを正しくフロアプランするため、モジュール ピンとインターフェイス ロジックが配置された位置を把握しておく必要があります。インポートされた OOC モジュールの保持レベルは選択できるので、必要に応じて配置配線に少しの変更を加えることができます。このフローでは、OOC インプリメンテーション結果をデバイスのほかのエリアまたは別のデバイスに移動したり複製したりすることはサポートされていません。

  モジュール再利用フローには、モジュール制約の構築メカニズムが異なる 2 つの方法があります。最上位デザインと 1 つまたは複数の再利用モジュールを統合する際は、コンテキスト制約 (フル デザインでのモジュールの接続方法を定義) およびタイミング制約が重要です。

  モジュール再利用の 2 つの方法は、次のとおりです。

  - **ボトムアップ再利用**: 最上位デザインに関する知識がなくても、OOC インプリメンテーションを実行し、その OOC 結果を使用して最上位インプリメンテーションを実行できます。IP のような検証済みのモジュールを作成して配置配線まで実行し、1 つまたは複数の最上位デザインで再利用できます。このフローでは、最上位デザインの詳細はわかっていないので、コンテキスト制約はユーザーが指定する必要があります。コンテキスト制約では、モジュールの物理的な位置、モジュール I/O の配置の詳細、クロック ソースの定義、モジュールを出入りするパスのタイミング要件、および未使用 I/O に関する情報などを定義します。

- **トップ ダウン再利用**: 最上位デザインおよびフロアプランで OOC インプリメンテーション制約が作成され、OOC インプリメンテーションは最上位デザインにより駆動されます。この方法はチーム デザインで有益で、デザイン内の 1 つまたは複数モジュールの合成およびインプリメンテーションを並行して実行できます。チーム メンバーはそれぞれの担当部分を個別にインプリメントし、その結果を統合デザインで再利用します。このフローでは最上位デザインの詳細 (ピン配置、フロアプラン、タイミング要件) がわかっており、OOC インプリメンテーションをガイドするのに使用されます。これにより、OOC モジュールのピン制約、最上位の入力/出力タイミング要件、境界最適化制約すべてを最上位デザインから作成できます。

どちらの方法を使用しても、デザイン全体ではなく、デザインに含まれるモジュールの 1 つのみをインプリメントするので、インプリメンテーションの実行時間を短縮できます。これにより、1 日に実行可能なコンパイルの回数を増加できるので、設計時間を削減でき、モジュール ベースでタイミングを検証して満たすことができます。残りのデザインが完成していなかったり、使用可能でない場合でも、モジュールの作業を進めることができます。

重要: この資料の情報は、7 シリーズ デバイスにのみ該当し、UltraScale™ および UltraScale+™ デバイスには該当しません。UltraScale および UltraScale+ デバイスの階層デザインについては、『Vivado Design Suite ユーザー ガイド: パーシャル リコンフィギュレーション』(UG909) [参照 6] を参照してください。

設計に関する考慮事項

階層デザイン手法で最適な結果を得るには、特別な考慮が必要です。次のセクションでは、Vivado 階層デザイン フローのアーキテクチャ プランニング、設計、制約に関して考慮すべき点について説明します。

パフォーマンス目的のデザイン

モジュールをデザインの残りの部分から独立させてインプリメントすると、通常のトップ ダウン フローで実行されるモジュール間の最適化は実行されません。このような制限によるパフォーマンスの劣化を防ぐには、次のガイドラインに従ってください。

- インプリメントする OOC モジュールを注意して選択します。デザインのほかのロジックから論理的に独立しており、デバイスの連続したエリアに物理的に制約可能なモジュールを選択してください。
- 選択したモジュールを考慮して効率的な階層を構築します。独立したインプリメンテーション用に階層の構造を作成します。デザイン階層は、重要な考慮事項です。デザインをどのように分割するかは、結果の質に大きな影響を与える可能性があります。OOC インプリメンテーション用にモジュールを適切にグループ化するため、階層の追加または変更が必要な場合もあります。
- クリティカル パスは、モジュール (サブモジュールまたは最上位) 内に完全に含まれるようにします。
- モジュール内の最適化が最大限に実行されるようにし、配置配線を柔軟に実行できるようにするため、モジュール間の入力および出力にレジスタを付けます。
- コンテキスト制約を定義して、モジュールの使用方法に関する情報を供給します。コンテキスト制約では、最上位でモジュールをどのように接続するかが定義されるので、さらに多くの最適化および正確なタイミング解析を実行できます。詳細は、「アウト オブ コンテキスト コマンドおよび制約」セクションの「アウト オブ コンテキスト デザイン制約」を参照してください。
- 専用接続は、常に OOC モジュールの境界を越えて適切に処理されるとは限りません。関連するデザイン エレメントは、すべてパーティションにまとめる必要があります。たとえば、トランシーバーまたは IOLOGIC コンポーネントなどの専用接続が付いた I/O コンポーネントがその例です。詳細は、「コンポーネント間の専用接続」を参照してください。

効率的なフロアプランの構築

OOC モジュールをインプリメントするには、次の要件に従う必要があります。

- 各モジュールのインプリメンテーションに Pblock 制約を使用し、配置を制御する必要があります。Pblock を使用しない場合、アセンブリ段階で配置が競合する可能性があります。
- すべての OOC Pblock に CONTAIN_ROUTING プロパティを追加します。このプロパティを設定しないと、配線に競合が発生する可能性があり、lock_design でインポートされたモジュールの配線をロックできません。
- 各 OOC モジュールの Pblock 範囲は重複しないようにします。最上位デザインに複数の OOC モジュールをインポートする場合は、モジュールがそれぞれデバイスの別の領域を使用している必要があります。
- 入れ子の Pblock (子 Pblock) は、その Pblock の範囲が親 Pblock の範囲に完全に含まれ、PARENT プロパティが正しく設定されていれば、OOC インプリメンテーションでサポートされます。詳細は、表 5 の「Pblock 制約」を参照してください。
- クロック バッファー (最上位と OOC モジュールの両方) は、すべてロックする必要があります。OOC モジュール内のバッファーには LOC 制約を設定し、最上位のバッファーの位置を HD.CLK_SRC 制約で指定する必要があります。HD.CLK_SRC 制約の詳細は、11 ページの「アウト オブ コンテキスト デザイン制約」を参照してください。
- OOC インプリメンテーション結果を再利用する場合は、HD.PARTPIN_RANGE または HD.PARTPIN_LOCS 制約を使用して OOC インプリメンテーション中に OOC モジュール ピンをロックすることを強くお勧めします。

  パーティション ピンは、OOC インプリメンテーションで主に次の 2 つの役割を果たします。

  - 関連するインターフェイス ロジックの配置をガイドするための物理ロケーションを作成します。配置に影響を与えるには、PartPin へおよび PartPin からのタイミング制約 (set_max_delay) が必要です。
  - 配線ツールに対してインターフェイス サブネットを配線するポイントを示します。PartPin がないと、インターフェイス ネットは配線できません。再利用中にインターフェイス ネットで必要とされるモジュール ブロック内の配線リソースを配線する OOC 結果を作成できます。これにより OOC ネットは保持されず、デザインが配線不可能な状態は発生しません。

  HD.PARTPIN_RANGE および HD.PARTPIN_LOCS 制約の詳細は表 6、set_max_delay の詳細は表 4 を参照してください。

コンポーネント間の専用接続

コンポーネントと専用接続は、デザインの同じパーティションに含めることが推奨されます。場合によっては、これは必須です。OOC モジュールの境界をまたがる専用接続があると、パフォーマンスが低下したり、インプリメンテーション エラーが発生することがあります。次に、専用接続を含むコンポーネントをリストします。

- **IOLOGIC および IOBUF**: ILOGIC または OLOGIC、IDDR、ODDR、ISERDES、および OSERDES に配置されたレジスタから IBUF、OBUF、IBUFDS、OBUFDS、IOBUF、および IOBUFDS などの I/O コンポーネントへの接続が含まれます。
- **GT トランシーバー コンポーネント**: GTX および GTP トランシーバーおよびそれらの専用 I/O 接続です。
- 双方向ポートは、できるだけ使用しないようにしてください。双方向ポートは PP_LOCS を受信できないので、双方向ポートの PP_RANGE または PP_LOCS 制約は OOC モジュールで自動的に削除されます。双方向ポートが必要な場合は、関連するインターフェイス ロジックをフロアプランして、再利用フロー中に OOC モジュールがインポートされる際のタイミング問題を回避するようにしてください。

デザインの異なるパーティションで相互接続される I/O コンポーネントを配置しないようにしてください。

IDELAYCTRL グループの使用

OOC モジュール内の IDELAYCTRL グループの使用はサポートされています。OOC インプリメンテーションで IDELAYCTRL が挿入され、その OOC インプリメンテーション結果が最上位にインポートされます。IDELAYCTRL グループを使用する場合は、次の規則が適用されます。

- それぞれに独自の IDELAYCTRL を含む複数の OOC モジュールで、同じクロック領域を共有することはできません。
- IDELAYCTRL を含む OOC モジュールは、100% 保持できません。これは、このバージョンの Vivado における既知の制限であり、今後の Vivado Design Suite リリースで修正される予定です。

I/O およびクロック バッファー

I/O およびクロック バッファーは OOC モジュール内でサポートされますが、使用方法によっては特別な注意が必要なこともあります。

- **I/O バッファー :** OOC ポートが最上位の I/O バッファーに直接接続される場合は、このバッファーを OOC モジュール内に移動すると結果が改善することがあります。ただし、すべての状況のおいてこれが可能なわけではありません。たとえば、OOC ポートが最上位の IBUF に直接接続されているのに、IBUF が OOC モジュール以外のその他のロジックも駆動する場合などです。このような場合、OOC モジュール内のロジックを `HD.PARTPIN_LOCS` 制約で制御する必要があります。詳細は、「アウト オブ コンテキスト コマンドおよび制約」を参照してください。
- **リージョナル クロック バッファー :** BUFR または BUFHCE が OOC モジュール内にある場合、特定の位置にロックする必要があります。これにより、バッファーで駆動されるロジックが適切に配置されるようになります。ただし、BUFR または BUFHCE が最上位デザインに含まれ、OOC Pblock がバッファーでアクセスできるよりも多くのクロック領域にまたがる場合、さらに多くの情報を供給する必要があります。入れ子の Pblock は、OOC Pblock で定義された範囲のサブセットである範囲で作成する必要があります。この入れ子の Pblock には、BUFR または BUFHCE で駆動されるすべてのセルが含まれ、次のコマンドで作成できます。

  ```
  create_pblock -parent <parent_pblock_name> <nested_pblock_name>
  add_cells_to_pblock <nested_pblock_name> -cells [get_cells -of [get_nets
  -segments -of [get_ports [list <clock_port> <clock_port>]]] -filter
  "(IS_PRIMITIVE)"]
  resize_pblock <nested_pblock_name> -add {SLICE_Xx1Yy1:SLICE_Xx2Yy2}
  ```

  これを、最上位の BUFR または BUFHCE で駆動される各モジュール ポートに対して実行する必要があります。複数の OOC クロック ポートのロードが同じクロック領域に含まれる場合、該当するすべてのポートを上記の `add_cells_to_pblock` コマンドでリストできます。入れ子の Pblock の範囲は、最上位インプリメンテーションの BUFR または BUFHCE 位置に対応する必要があります。最上位のバッファー位置と OOC モジュールの対応する Pblock 範囲が一致しないと、最上位インプリメンテーション中に配線不可能な状態になることがあります。

- **グローバル クロック バッファー :** グローバル バッファーは OOC モジュール内でサポートされます。BUFR が OOC インスタンス内にあると、OOC インプリメンテーションでクロック ネットがグローバル配線に配線されます。OOC ポートが最上位のクロック ネットで駆動される場合、クロック ネットは OOC インプリメンテーション中には配線されず、クロック遅延/スキューを決定するのにタイミング見積もりが使用されます。この場合、タイミング見積もりを改善するため `HD.CLK_SRC` 制約を使用する必要があります。この制約により、ドライバーの位置およびタイプ (BUFG または BUFR など) がツールに示され、タイミング見積もりを改善するため CPR (Clock Pessimism Removal) が計算されます。CPR の詳細は、『Vivado Design Suite ユーザー ガイド: デザイン解析およびクロージャ テクニック』(UG906) [参照 1] を参照してください。

アウト オブ コンテキスト デザインのデザイン ルール

表 1 に、OOC デザインをインプリメントする際に実行されるデザイン ルール チェック (DRC) と、対応するデザイン ルールを示します。

表 1: 階層デザインのデザイン ルール

DRC 番号	重要度	デザイン ルール
HDOOC-1	エラー	リコンフィギャラブル モジュールに Pblock を定義する必要があります。
HDOOC-2	エラー	HD モジュールに特定の Pblock プロパティを定義する必要があります。
HDOOC-3	エラー	OOC モジュールのビットストリームは生成できません。
HDOOC-4	エラー	セルに Pblock 範囲または LOC がありません。

チェックポイント

階層デザイン フローでは、モジュール インプリメンテーションの結果をエクスポートおよびインポートするためにチェックポイントが使用されます。チェックポイントは、論理デザイン、物理デザイン、モジュール制約をアーカイブしたもので、デザインを完全に復元するのに必要な唯一のファイルです。

保存されたチェックポイントは、それを生成したときに指定していたパーツ/パッケージ/スピード グレードの組み合わせにのみ読み込むことができます。

推奨: 階層デザイン フローで読み込まれるすべてのデータがモジュール インターフェイスで完全に一致するようにするには、`read_checkpoint` コマンドに `-strict` オプションを使用することをお勧めします。チェックポイントの詳細は、『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904) [参照 2] のこのセクションを参照してください。

アウト オブ コンテキスト コマンドおよび制約

階層デザイン フローは、現在のところ、非プロジェクト モードのバッチ/Tcl インターフェイス (Vivado IDE (GUI) またはプロジェクト ベース コマンドなし) でのみサポートされています。サンプル デザインおよび関連スクリプトも、ご要望に応じて提供しています。詳細は、ザイリンクス サポートにお問い合わせください。

次のセクションでは、階層デザイン フローで使用されるアウト オブ コンテキスト コマンドおよび制約について説明します。階層デザイン フローを実行するためにこれらのコマンドを使用する例も示します。各コマンドの詳細は、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 3] を参照してください。

アウト オブ コンテキスト コマンド

残りのデザインから独立させたアウト オブ コンテキスト (OOC) モジュールを合成またはインプリメントするには、ツールをアウト オブ コンテキスト モードで実行する必要があります。それ以外は、アウト オブ コンテキスト フローを実行するためのコマンドはほかのフローと同じです。現在のところ、合成、最適化、またはインプリメンテーションでサポートされないコマンドはありません。

合成

このフローでは、複数の合成ツールおよび手法がサポートされます。サポートされるツールは、次のとおりです。

- **XST**: パーティション (PMXL ファイル) を使用したボトムアップ合成またはインクリメンタル合成。7 シリーズのみでサポートされます。

  注記: XST は、新しい Vivado デザイン、7 シリーズよりも新しいアーキテクチャをターゲットにしたデザインには推奨されません。

- **Synplify**: ボトムアップ合成またはコンパイル ポイント (階層プロジェクトを使用して各ネットリストを生成)
- **Vivado 合成**: アウト オブ コンテキスト (OOC) 合成のみ。

重要: OOC 合成 (ボトムアップ合成) では、各モジュールにそれぞれ合成 run があります。通常は、下位モジュールの自動 I/O バッファー挿入をオフにします。

この資料では、Vivado 合成フローのみについて説明します。Synplify フローの詳細は、Synopsys 社 Synplify の資料を参照してください。

このフローの Vivado 合成は、synth_design コマンドを使用してバッチ モードで実行します。

```
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top <top_module_name>
-part <part>
```

表 2: synth_design オプション

コマンド オプション	説明
-mode out_of_context	合成およびダウンストリーム ツールの I/O 挿入がオフになります。write_checkpoint を実行すると、チェックポイントにこのモードが保存されます。
-flatten_hierarchy rebuilt	-flatten_hierarchy に設定可能な値は複数ありますが、階層デザイン フローでは rebuilt が推奨されます。
-top	合成するモジュールのモジュール/エンティティ名を指定します。synth_design の前に set_property top <top_module_name> [current_fileset] を実行した場合は、このオプションを使用する必要はありません。
-part	ターゲットにするザイリンクス パーツ (例: xc7k325tffg900-3) を指定します。

synth_design コマンドは、デザインを合成してその結果をメモリに格納します。結果をファイルに書き出すには、write_checkpoint コマンドを実行する必要があります。

```
write_checkpoint <file_name>.dcp
```

上記のコマンドを実行すると、合成結果が DCP ファイルに保存され、インメモリ デザインを閉じることができます。合成結果を後で読み出すには、open_checkpoint を使用します。これにより、合成をそのたびに実行し直さなくても、合成結果に対して 1 つまたは複数のインプリメンテーション run を実行できます。

タイミング制約および物理制約は、OOC 合成に渡すことができます。物理制約は、合成では無視され、最終的な DCP に渡されます。タイミング制約の場合、モジュール レベルのコンテキスト制約すべてに USED_IN out_of_context を設定することが重要です。制約が OOC にのみ使用されることを指定する方法については、11 ページの「OOC 専用制約」を参照してください。

インプリメンテーション

このセクションでは、OOC フローでモジュール インスタンスをインプリメントするのに必要なコマンドについて説明します。モジュールの複数のインスタンスが最上位デザインにインスタンシエートされおり、アセンブリ フローを使用する場合は、必要なインプリメンテーション結果を生成するために、それぞれ独自の Pblock 制約が設定された OOC インプリメンテーションが複数必要です。

`synth_design -mode out_of_context` を既に実行しており、結果がまだメモリにある場合は、インプリメンテーションを直接実行できます。たとえば、次のインプリメンテーション コマンドを使用できます。

- `read_xdc`: すべての制約がまだ読み込まれていない場合に使用します。一部のモジュール レベルの XDC 制約を OOC インプリメンテーションのみに適用する必要のあることもあります。OOC のみの制約の詳細は、11 ページの「OOC 専用制約」を参照してください。
- `set_property HD.PARTITION 1 [current_design]`: セルをインプリメント済み OOC として指定し、再利用フローで DRC と必要なソフトウェア機能を有効にします。これは OOC インプリメンテーションには必要ありませんが、再利用フローで最終的な DRC が正しくインポートされることを確実にするには良い方法です。再利用フローでの HD.PARTITION の機能については、16 ページの「最上位再利用コマンド」を参照してください。
- `opt_design` (オプションだが推奨)
- `place_design`
- `phys_opt_design` (オプションだが推奨)
- `route_design`

メモリにデザインが読み込まれていない場合は、デザインを読み込む必要があります。これには、次のいずれかの方法を使用します。

- **方法 1: ネットリスト デザインの読み込み**

```
read_edif <file_name>.edf/edn/ngc
link_design -mode out_of_context -top <top_module_name> -part <part>
```

表 3: link_design オプション

コマンド オプション	説明
`-mode out_of_context`	ネットリスト デザインを OOC モードで読み込みます。ダウンストリーム ツールの特別チェックおよび最適化がオンになります。
`-part`	ターゲットにするザイリンクス パーツ (例: `xc7k325tffg900-3`) を指定します。
`-top`	インプリメンするモジュールのモジュール/エンティティ名を指定します。`link_design` の前に `set_property -top <top_module_name> [current_fileset]` を実行した場合は、このオプションを使用する必要はありません。

デザイン ネットリストを読み込んだ後に `link_design` に `-mode out_of_context` オプションを使用しない場合、その後のインプリメンテーション段階でデザインがフル デザインとして処理され、ソースのない信号およびロードのない信号はすべて削除されます。モジュール解析フローを実行するには、`synth_design` または `link_design` を実行するときに OOC モードを定義する必要があります。

- **方法 2: チェックポイントを開く**

  ```
  open_checkpoint <file_name>.dcp
  ```

重要: open_checkpoint コマンドには、-mode out_of_context オプションはありません。モードはチェックポイントの一部として保存されるので、チェックポイントを保存するときにモードが正しいことを確認してください。

- **方法 3: さまざまなファイル タイプを追加**

  add_files コマンドと link_design コマンドを使用すると、複数のファイルおよびさまざまなファイル タイプを含むモジュールを読み込むことができます。

  ```
  add_files <file_name>.dcp
  add_files <file_name>.edf
  add_files <file_name>.xdc
  link_design -mode out_of_context -top <top_module_name> -part <part>
  ```

アウト オブ コンテキスト デザイン制約

モジュール解析フローでは、次のどの制約も必須ではありません。タイミング解析をより正確なものにするには、HD.CLK_SRC および create_clock を使用してください。その他すべての制約はオプションです。

モジュール再利用フローを使用する場合は、これらのコンテキスト制約が重要になります。OOC モジュールを含むデザインを問題なく統合するには、これらの制約を使用して、物理リソースが適切に割り当てられ、クロック関連性が理解され、モジュール インターフェイスに関する情報が正確に設定されるようにします。各モジュールに制約を設定しないと、デザインの統合がより困難になります。

OOC 専用制約

OOC インプリメンテーションに必要な制約の中には、最上位デザインにインポートされると問題となるものがあります。このような状況を避けるため、別の XDC ファイルでこれらの制約を指定して、OOC でのみ使用されるように設定する必要があります。XDC ファイルを OOC フローにのみ使用されるように指定する方法は、2 つあります。XDC ファイルの制約が OOC フローでのみ使用されるように指定すると、それらにマーカーが追加され、OOC デザイン以外に読み込まれる際に制約が無視されるようになります。

- 方法 1: read_xdc を使用

  read_xdc コマンドで XDC ファイルを読み込む際に -mode out_of_context オプションを使用します。

  ```
  read_xdc -mode out_of_context <file>.xdc
  ```

ヒント: read_xdc コマンドはデザインを link_design で読み込む前または後に実行できます。

- 方法 2: USED_IN プロパティを使用

  add_files コマンドを使用してファイルを追加する場合、ファイルにプロパティを設定して OOC でのみ使用されるように指定できます。XDC ファイルが使用されるすべてのフロー (合成またはインプリメンテーション、あるいはその両方) を指定する必要があります。

  ```
  add_files <file>.xdc
  set_property USED_IN {synthesis implementation out_of_context} [get_files <file>]
  ```

重要: add_files コマンドは link_design でデザインを読み込むよりも前に実行する必要があります。既に読み込まれたデザインに対して add_files コマンドを使用しても、ファイルを追加はできません。

制約の構文

次の表に、アウト オブ コンテキスト インプリメンテーションで使用する必要のあるタイミング制約、配置制約、コンテキスト制約をリストします。これらの制約の多くは、どのデザイン フローでも使用できます。詳細は、『Vivado Design Suite ユーザー ガイド: 制約の使用』(UG903) [参照 4] を参照してください。

重要: トップダウン再利用フローを使用すると、このセクションに示す制約はすべて自動的に生成されます。これらの制約を生成するスクリプトおよび方法については、『Vivado Design Suite チュートリアル: 階層デザイン』(UG946) を参照してください。この資料の入手方法については、ザイリンクス サポートにお問い合わせください。

表 4: タイミング制約

制約名	説明
set_max_delay	入力および出力遅延を定義し、OOC モジュールで許容される時間を割り当てます。OOC インプリメンテーションの配置を制御し、最上位のタイミング クロージャを達成しやすくします。
create_clock	OOC モジュール ポートのクロックを定義します。create_clock 制約は、クロック バッファーが最上位にインスタンシエートされている場合でも、OOC モジュールにインスタンシエートされている場合でも、各クロック ポートごとに必要です。
set_clock_uncertainty	OOC モジュールへの入力クロックのばらつきを定義します。タイミング解析が正確になるよう、OOC モジュールのすべてのクロックに対して定義する必要があります。定義しないと、モジュールをインポートしたときにパスのタイミングが満たされなくなることがあります。
set_system_jitter	システム ジッター値を定義します。最上位デザインに基づいてユーザー クロックのばらつき (set_clock_uncertainty) を定義する場合は 0 に設定する必要があります。0 に設定しないと、システム ジッターが OOC インプリメンテーションのばらつき計算で考慮され、最終的な値がユーザーの定義した値とは異なるものになります。
set_clock_latency	OOC モジュールへの入力クロックのレイテンシを定義します。この制約は、クロック パス全体が既知ではない場合に、クロック遅延を正しくモデリングするために必要です。
set_clock_groups	非同期クロック (-asynchronous) または同じグローバル バッファーで駆動されるクロック (-physically_exclusive) を定義します。

タイミング制約の例:

- `create_clock -period 8.000 -name clk -waveform {0.000 4.000} [get_ports clk]`
- `set_max_delay -from [get_ports <ports>] -to [get_pins <synchronous pin>] -datapath_only <delay>`
- `set_max_delay -from [get_pins <synchronous pin>] -to [get_ports <ports>] -datapath_only <delay>`
- `set_system_jitter 0.0`
- `set_clock_latency -source -min 0.10`
- `set_clock_latency -source -max 0.20`

- `set_clock_groups -physically_exclusive -group [clk1] -group [clk2]`
- `set_clock_groups -asynchronous -group [clk1] -group [clk2]`

これらのタイミング制約は、その OOC モジュール自体に適用されます。OOC インプリメンテーションには、インスタンスへのタイミング、インスタンスからのタイミング、インスタンス内部のタイミングすべてが含まれる必要があります。これには、フォルス パスおよびマルチサイクル パスなどの特殊なパスも含まれます。

表 5: Pblock 制約

コマンド/プロパティ名	説明
`create_pblock`	各 OOC インスタンスの最初の Pblock を作成します。
`resize_pblock`	Pblock のサイト タイプ (SLICE、RAMB36 など) とサイト位置を定義します。
`add_cells_to_pblock`	Pblock に含めるインスタンスを指定します。通常これは、個別のインスタンスではなく階層レベルです。OOC インプリメンテーションでは、セル名を指定する代わりに、`-top` を使用して OOC モジュールの下のセルすべてを指定できます。
`CONTAIN_ROUTING`	Pblock に含まれない配線リソースが使用されないようにする Pblock プロパティです。デフォルト値は `FALSE` です。Pblock 範囲に完全に含まれるパスのみが含まれます。たとえば、BUFGMUX 範囲がない場合、BUFGMUX に入出力されるパスは含まれません。これは BUFGMUX のような多くのコンポーネントで適切な動作です。
`EXCLUDE_PLACEMENT`	定義された Pblock 範囲内の Pblock に含まれないロジックが配置されないようにするため使用する Pblock プロパティです。デフォルト値は `FALSE` です。このプロパティは OOC インプリメンテーションには影響しませんが、アセンブリ中の最上位ロジックの配置に影響します。アセンブリ中に最適な結果を得るには、値を `FALSE` のままにすることをお勧めします。
`PARENT`	Pblock 階層を識別する Pblock プロパティで、子 Pblock を完全に含む親 Pblock の名前を示します。OOC インプリメンテーションを含む入れ子の Pblock には、モジュール再利用で正しく動作するよう PARENT キーワードを使用する必要があります。

Pblock コマンドおよびプロパティの例:

- `create_pblock <pblock_name>`
- `add_cells_to_pblock [get_pblocks <pblock_name>] -top`
- `resize_pblock [get_pblocks <pblock_name>] -add {SLICE_X0Y0:SLICE_X100Y100}`
- `resize_pblock [get_pblocks <pblock_name>] -add {RAMB18_X0Y0:RAMB18_X2Y20}`
- `set_property CONTAIN_ROUTING true [get_pblocks <pblock_name>]`

Pblock に追加されるセルが `-top` を使用して指定されていることに注意してください。これは、OOC インプリメンテーションでは OOC インスタンスが最上位になり、OOC インスタンス全体が Pblock に含まれる必要があるからです。`-top` を使用することで、OOC モジュールが最上位デザインにインポートされる際に、Pblock が正しい階層レベルに適切に変換されるようもなります。

OOC モジュール内のロジックをフロアプランする場合は、Pblock を入れ子にできます。子 Pblock は完全に親 Pblock 内に含まれる必要があります。Pblock 間の親子関係は、次に示すように PARENT プロパティを使用して宣言します。

```
set_property PARENT <parent_pblock_name> [get_pblocks <child_pblock_name>]
```

Pblock プロパティ PARENT は別の Pblock を参照するので、制約が処理される順序が重要です。親 Pblock (-top を使用) は、それを参照する子 Pblock よりも前に定義する必要があります。

上記のタイミング制約および物理制約に加え、OOC インプリメンテーションのコンテキストを定義する制約もあります。コンテキスト制約では、OOC インプリメンテーションがインポートされる最上位の環境を定義します。

表 6: コンテキスト制約

コマンド/プロパティ名	説明
HD.CLK_SRC	OOC インプリメンテーションで使用され、クロック バッファーが OOC モジュール外で使用されることを示します。値は、クロック バッファー インスタンスの位置です。このプロパティはクロック ポートに適用されるので、この制約が適用される前にポートにクロックを定義 (create_clock) しておく必要があります。
HD.PARTPIN_LOCS	配線される指定のポートにインターコネクト タイル (INT) を指定します。HD.PARTPIN_RANGE よりも優先されます。内部 OOC ロジックの配置配線に影響します。 クロック ポートに使用するとクロックにローカル配線が使用されるので、クロック ポートには使用しないでください。 専用接続には使用しないでください。
HD.PARTPIN_RANGE	指定したピン/ポートを配線するために使用可能なコンポーネント サイト (SLICE、DSP、ブロック RAM) またはインターコネクト タイル (INT) の範囲を定義します。 この制約は専用接続を持たないピンまたはポートに対してのみ有効です (最上位 I/O パッドへのクロックまたは直接接続など)。これらのピンまたはポートに適用される場合、制約は無視されます。
set_logic_unconnected	最上位で未接続のままになる指定した出力ポートに対して、追加で最適化を実行できるようにします。
set_logic_one	最上位の VCC で駆動される指定した入力ポートに対して、追加で最適化を実行できるようにします。
set_logic_zero	最上位の GND で駆動される指定した入力ポートに対して、追加で最適化を実行できるようにします。

重要: set_logic 境界最適化制約を誤って指定すると、ツールが正しく動作せず、エラーが発生する可能性があります。たとえば、出力ポートが最上位で使用されるのに OCC モジュールで未接続として定義すると、次のようなエラー メッセージが表示されます。

```
ERROR:[Opt 31-67] Problem: A LUT2 cell in the design is missing a connection on input pin I0, which is used by the LUT equation
```

コンテキスト制約の例:

- `set_property HD.CLK_SRC BUFGCTRL_X0Y16 [get_ports <port_name>]`
- `set_property HD.PARTPIN_LOCS INT_R_X0Y0 [get_ports <port_name>]`
- `set_property HD.PARTPIN_RANGE SLICE_X0Y1:SLICE_X1Y3 [get_ports <port_name>]`
- `set_logic_unconnected [get_ports <port_name>]`
- `set_logic_one [get_ports <port_name>]`
- `set_logic_zero [get_ports <port_name>]`

モジュール解析フローでは、デフォルトではインターフェイス ネット (OOC モジュール ポートに接続されるモジュール内のネット) は配線されません。これらのインターフェイス ネットが配線されるようにするには、`HD.PARTPIN` 制約を使用してモジュール ポートをロックする必要があります。モジュール ポート (またはパーティション ピン) をすばやく配置するには、`HD.PARTPIN_RANGE` を OOC モジュールの Pblock `SLICE` 範囲に設定します。これらのピンをさらに厳密に配置するには、`HD.PARTPIN_RANGE` により限定的な値を指定するか、`HD.PARTPIN_LOCS` を指定します。最適なサイトまたは範囲を判断するには、Vivado IDE で [Device] ウィンドウを開き、[Routing Resources] ボタン をクリックして配線リソースをオンにします。

拡大すると、図 1 に示すように INT の位置が表示されます。この図では見やすくするために配線リソースを非表示にしています。

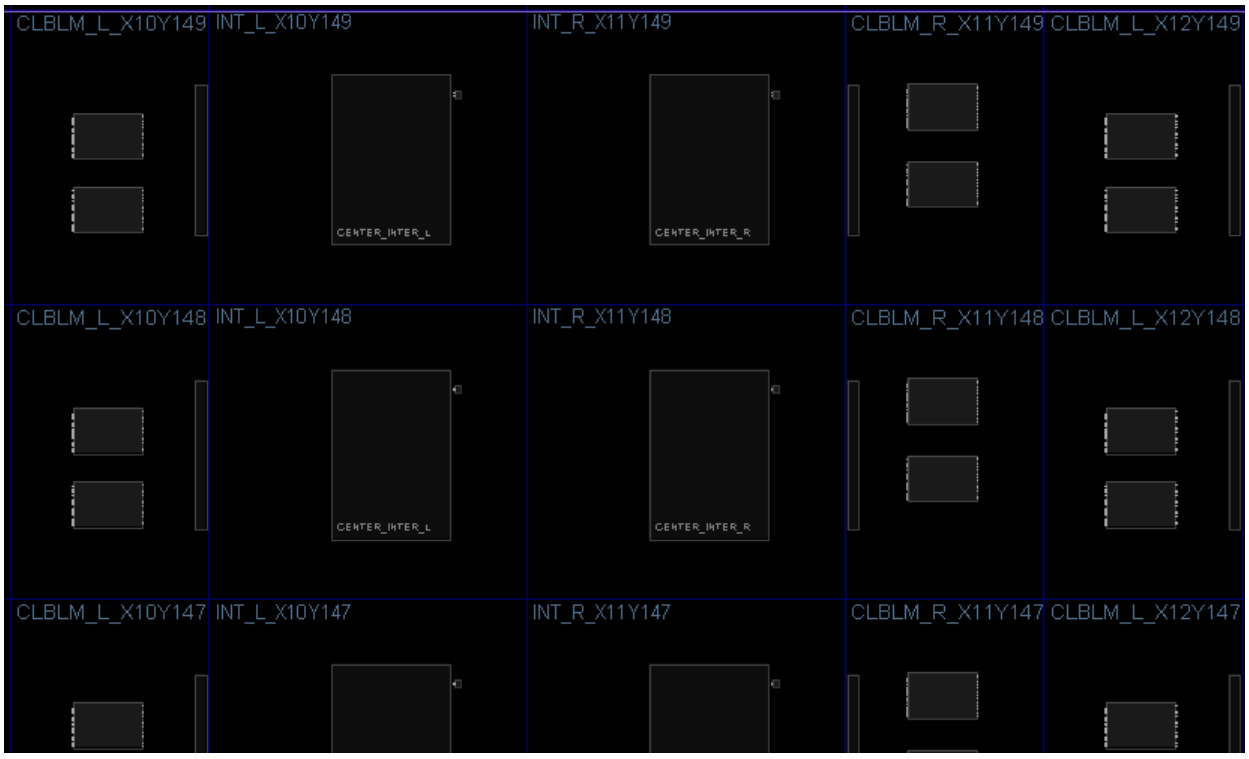

図 1: INT タイルの位置

最上位再利用コマンドおよび制約

次のセクションでは、アウト オブ コンテキスト インプリメンテーションをインポートする際に最上位で使用されるコマンドおよび制約について説明します。サンプル デザインおよび関連スクリプトも、ご要望に応じて提供しています。詳細は、ザイリンクス サポートにお問い合わせください。

各コマンドの詳細は、『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835) [参照 3] を参照してください。

最上位再利用コマンド

アセンブリでは、モジュール解析フローでインプリメントしたモジュールを読み込みます。各パーティション インスタンスのチェックポイントは、必ず存在している必要があります。各チェックポイントは、OOC モジュールをブラック ボックスとして含む最上位デザインに読み込まれます。OOC インプリメンテーション結果は、ブラック ボックスではないセルには読み込むことはできません。これで、標準的なインプリメンテーション コマンドを使用して、まだ配置配線されていないデザイン部分 (最上位) をインプリメントできるようになります。

合成

各パーティション インスタンスに対応するブラック ボックスを含む最上位ネットリストが必要です。これには、最上位合成に各パーティション インスタンスのモジュール/エンティティ宣言を含め、ロジックは含まれないようにする必要があります。

最上位合成では、通常すべての最上位ポートに I/O バッファーが推論されますが、I/O バッファーが OOC モジュールにインスタンシエートされている場合は、最上位合成でポートごとに I/O バッファー挿入をオフにする必要があります。Vivado 合成でこれを設定する属性は `IO_BUFFER_TYPE = "none"` です。`IO_BUFFER_TYPE` およびその他の合成属性の詳細は、『Vivado Design Suite ユーザー ガイド: 合成』(UG901) [参照 5] のこのセクションを参照してください。

インプリメンテーション

最上位インプリメンテーションは、標準デザインと同じように実行できますが、次をさらに実行する必要があります。

1. OOC インプリメンテーション結果をインポートする各ブラック ボックス セルに `HD.PARTITION` プロパティを設定します。

   注記: `HD.PARTITION` プロパティは、OOC インプリメンテーションで設定されていれば OOC モジュールと共にインポートされますが、セルが確実に正しくマークされるようにするため、最上位ブラック ボックス インスタンスに設定しておくと有益です。

2. 各パーティション インスタンスの OOC チェックポイントを読み込みます。
3. 保持レベル (logical、placement、routing) を選択します。

セルをパーティションとしてマーク

セルをパーティションとしてマークするには、HD.PARTITION セル プロパティが必要です。HD.PARTITION を設定すると、次が実行されます。

- 指定したセルに DONT_TOUCH が設定され、HD 境界上および HD 境界を越える最適化が実行されないようになります。
- HD 特有の DRC が実行されます。
- read_checkpoint -cell で特定のコードがイネーブルになり、PartPin がクリーンアップされて余分な Pblock が削除されます。

```
set_property HD.PARTITION 1 [get_cells <cell_name>]
```

OOC チェックポイントの読み込み

OOC チェックポイントを読み込むには、read_checkpoint コマンドを -cell オプションを指定して実行します。最上位デザインが既に開いており、各パーティション インスタンスがブラック ボックスとして含まれている必要があります。

```
read_checkpoint -cell <cell_name> <file> [-strict]
```

表 7: read_checkpoint オプション

オプション名	説明
-cell	OOC モジュールの完全な階層名を指定します。
-strict	セルを置き換えるのにポートが完全に一致している必要があることを指定し、パーツ、パッケージ、スピード グレード値が同一であることを確認します。
<file>	読み込む OOC チェックポイントを指定します。

OOC モジュール インプリメンテーションの結果をインポートしてロックする際、インターフェイス ネットは保持されません。また、初期のトップダウンまたは OOC インプリメンテーション中に生成された PartPin ロケーションは理想的なロケーションではない可能性があり、配線が不必要に制限されることがあります。このため、read_checkpoint -cell コマンドでは自動的にインターフェイス ネットの配線が解除され、セルのチェックポイントからの PartPin はすべて削除されます。

保持レベルの設定

OOC チェックポイントを読み込んだら、このモジュールの保持レベルを定義する必要があります。

インポートした OOC チェックポイントの配置および配線をロックするには、lock_design コマンドを使用します。

```
lock_design [-level <value>] [-unlock] [<cell>]
```

このコマンドでは、-level オプションで次の値を指定することにより保持レベルを指定できます。

- logical: 論理デザインを保持します。配置または配線情報も使用されますが、ツールで結果を改善できる可能性がある場合は変更可能です。
- placement (デフォルト): 論理および配置デザインを保持します。配線情報も使用されますが、ツールで結果を改善できる可能性がある場合は変更可能です。
- routing: 論理、配置、および配線デザインが保持されます。内部配線は保持されますが、インターフェイス ネットは保持されません。配線を保持するには、OOC インプリメンテーションで Pblock に CONTAIN_ROUTING プロパティを使用している必要があります。これにより、OOC インプリメンテーションが再利用される際に配線が競合することはなくなります。

保持レベルに関係なく、配線も含めて物理データベース全体が読み込まれますが、ツールで結果を改善できると判断されない限りは変更されません。

表 8: `lock_design` の引数

引数名	説明
`-level`	保持レベルを指定します。指定可能な値は `logical`、`placement`、または `routing` で、デフォルト値は `placement` です。
`-unlock`	セルのロックを解除します。セルを指定しない場合は、デザイン全体のロックが解除されます。ロックを解除するには、`-level` オプションを指定する必要があります。この場合、ロックする際とは逆に機能し、`routing` を指定すると配線のロックのみが解除され、`placement` を指定すると配置配線のロックが解除されます。
`<cell>`	ロックする階層セル名を指定します。セルを指定しない場合は、デザイン全体がロックされます。

最上位再利用制約

最上位デザインで OOC モジュールを再利用する際、標準的なデザイン制約すべてを適用できます。OOC インプリメンテーションに使用される OOC 制約は、チェックポイントに保存され、デザインに適用可能なときに適用されます。

Tcl スクリプト

このフローの Tcl スクリプトは、ご要望に応じて提供しています。詳細は、ザイリンクス サポートにご連絡ください。

既知の問題

このセクションでは、現在のリリースの階層デザイン フローに関する既知の問題を説明します。

グローバル クロック配線の制限

最上位のバッファーで駆動されるクロックは、OOC インプリメンテーション中は配線されません。配線の見積もりが使用されます。`HD.CLK_SRC` を使用すると配線の見積もりを改善できます。OOC モジュール内のクロック バッファーは OOC インプリメンテーション中に配線されます。OOC クロックに最適な制約を設定すると、このタイミング見積もりは正確なタイミング結果を得るのに十分なものになります。

IDELAYCTRL の設定された OOC モジュールの制限

IDELAYCTRL の設定された OOC モジュールをインポートすることはできますが、ロックすることはできません。OOC モジュールは 100% 保持されません。

非プロジェクト モードのサポート

Vivado Design Suite の階層デザインは、現時点ではプロジェクト モードではサポートされていません。

XILINX
ALL PROGRAMMABLE™

付録 A

その他のリソースおよび法的通知

ザイリンクス リソース

アンサー、資料、ダウンロード、フォーラムなどのサポート リソースは、ザイリンクス サポート サイトを参照してください。

ソリューション センター

デバイス、ツール、IP のサポートについては、ザイリンクス ソリューション センターを参照してください。デザイン アシスタント、デザイン アドバイザリ、トラブルシューティングのヒントなどが含まれます。

参考資料

1. 『Vivado Design Suite ユーザー ガイド: デザイン解析およびクロージャ テクニック』(UG906)
2. 『Vivado Design Suite ユーザー ガイド: インプリメンテーション』(UG904)
3. 『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835)
4. 『Vivado Design Suite ユーザー ガイド: 制約の使用』(UG903)
5. 『Vivado Design Suite ユーザー ガイド: 合成』(UG901)
6. 『Vivado Design Suite ユーザー ガイド: パーシャル リコンフィギュレーション』(UG909)
7. Vivado Design Suite の資料

トレーニング リソース

ザイリンクスでは、この資料に含まれるコンセプトを説明するさまざまなトレーニング コースおよび QuickTake ビデオを提供しています。次のリンクから関連するトレーニング リソースを参照してください。

1. Vivado Design Suite QuickTake ビデオ チュートリアル
2. トレーニング コース: Vivado Design Suite を使用した FPGA の設計 4

お読みください: 重要な法的通知

本通知に基づいて貴殿または貴社 (本通知の被通知者が個人の場合には「貴殿」、法人その他の団体の場合には「貴社」。以下同じ) に開示される情報 (以下「本情報」といいます) は、ザイリンクスの製品を選択および使用することのためにのみ提供されます。適用される法律が許容する最大限の範囲で、(1) 本情報は「現状有姿」、およびすべて受領者の責任で (with all faults) という状態で提供され、ザイリンクスは、本通知をもって、明示、黙示、法定を問わず (商品性、非侵害、特定目的適合性の保証を含みますがこれらに限られません)、すべての保証および条件を負わない (否認する) ものとします。また、(2) ザイリンクスは、本情報 (貴殿または貴社による本情報の使用を含む) に関係し、起因し、関連する、いかなる種類・性質の損失または損害についても、責任を負わない (契約上、不法行為上 (過失の場合を含む)、その他のいかなる責任の法理によるかを問わない) ものとし、当該損失または損害には、直接、間接、特別、付随的、結果的な損失または損害 (第三者が起こした行為の結果被った、データ、利益、業務上の信用の損失、その他あらゆる種類の損失や損害を含みます) が含まれるものとし、それは、たとえ当該損害や損失が合理的に予見可能であったり、ザイリンクスがそれらの可能性について助言を受けていた場合であったとしても同様です。ザイリンクスは、本情報に含まれるいかなる誤りも訂正する義務を負わず、本情報または製品仕様のアップデートを貴殿または貴社に知らせる義務も負いません。事前の書面による同意のない限り、貴殿または貴社は本情報を再生産、変更、頒布、または公に展示してはなりません。一定の製品は、ザイリンクスの限定的保証の諸条件に従うこととなるので、https://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。IP コアは、ザイリンクスが貴殿または貴社に付与したライセンスに含まれる保証と補助的条件に従うことになります。ザイリンクスの製品は、フェイルセーフとして、または、フェイルセーフの動作を要求するアプリケーションに使用するために、設計されたり意図されたりしていません。そのような重大なアプリケーションにザイリンクスの製品を使用する場合のリスクと責任は、貴殿または貴社が単独で負うものです。https://japan.xilinx.com/legal.htm#tos で見られるザイリンクスの販売条件を参照してください。

自動車用のアプリケーションの免責条項

オートモーティブ製品 (製品番号に「XA」が含まれる) は、ISO 26262 自動車用機能安全規格に従った安全コンセプトまたは余剰性の機能 (「セーフティ設計」) がない限り、エアバッグの展開における使用または車両の制御に影響するアプリケーション (「セーフティ アプリケーション」) における使用は保証されていません。顧客は、製品を組み込むすべてのシステムについて、その使用前または提供前に安全を目的として十分なテストを行うものとします。セーフティ設計なしにセーフティ アプリケーションで製品を使用するリスクはすべて顧客が負い、製品の責任の制限を規定する適用法令および規則にのみ従うものとします。

この資料に関するフィードバックおよびリンクなどの問題につきましては、jpn_trans_feedback@xilinx.com まで、または各ページの右下にある [フィードバック送信] ボタンをクリックすると表示されるフォームからお知らせください。フィードバックは日本語で入力可能です。いただきましたご意見を参考に早急に対応させていただきます。なお、このメール アドレスへのお問い合わせは受け付けておりません。あらかじめご了承ください。