Vivado Design Suite
ユーザー ガイド

階層デザイン

UG905 (v2015.1) 2015 年 4 月 1 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
2015 年 4 月 1 日	2015.1	リリース番号をアップデート 15 ページの「合成」.の BUFFER_TYPE プロパティ名を IO_BUFFER_TYPE に変更

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、デザインを小さい管理可能なブロックにパーティションして個別に処理することができます。Vivado® Design Suite では、パーティションされたモジュールを残りのデザインから独立して (アウト オブ コンテキスト (OOC) で) インプリメントできます。次は、Vivado Design Suite での手法です。

- **モジュール解析 :** このフローでは、モジュールを残りのデザインとは別に解析して、リソース使用量を決定し、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。単にモジュールだけを合成、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 レポートを検証して、配置結果を確認します。

  このフローでは、パーティションされたモジュールまたは IP コアがデザインの最上位から独立して (OOC で) インプリメントされます。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 :** このフローでは、最上位デザイン内のモジュール解析フローからの配置配線されたモジュールを使用して、検証結果をロックします。デザインの特定のセクションを繰り返すことでタイミング クロージャーおよびその他の特定の目標を達成した後、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、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箇所を把握しておかないと、接続ロジックが正しくフロアプランできません。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、必要であれば配置配線を少し変更することができます。このフローでは、デバイスのほかのエリアや別のデバイスへの OOC インプリメンテーション結果の移動または複製はまだサポートされていません。

  モジュール再利用フローには、2 つのパターンがあります。この 2 つの違いはモジュール制約を構築するメカニズムにあります。最上位デザインを 1 つまたは複数の再利用モジュールと問題なく組み合わせるには、コンテキスト制約 (フル デザインでモジュールをどのように接続するか定義) およびタイミング制約が重要となります。

  モジュール再利用には、次のような方法があります。

  - **ボトムアップ再利用 :** この方法を使用すると、最上位デザインについてほとんど何も知らなくても OOC インプリメンテーションが実行され、再利用できるようになります。この OOC 結果を使用して、最上位インプリメンテーションが実行されます。配置配線を介して IP の一部のような検証済みのモジュールを構築でき、1 つまたは複数の最上位デザインで再利用できます。このフローの場合、最上位デザインの詳細はわからないので、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。これらは、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するために使用されます。
  - **トップダウン再利用 :** この方法を使用すると、最上位デザインおよびフロアプランが使用され、OOC インプリメンテーション制約が作成されます。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。これにより、チーム デザインで、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、アセンブルされたデザインでその結果を再利用して、それぞれの担当部分をインプリメントできます。このフローの場合、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) は既知の状態で、OOC インプリメンテーションをガイドするために使用されます。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、バウンダリ最適化制約すべてが最上位デザインから作成できるようになります。

これらどのフローを使用しても、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ことで、インプリメンテーション実行時間を削減できます。これにより、より多くコンパイルできるので、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。また、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。

設計に関する考慮事項

階層デザイン手法で最適な結果にするためには、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 フローでのアーキテクチャのプランニング、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

デザインの残りの部分から独立した (OOC) モジュールをインプリメントすると、通常トップダウン 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。こういった制限のためのパフォーマンス劣化を防ぐには、次のガイドラインに従ってください。

- インプリメントされる OOC モジュールを注意して選択します。デザインのほかのモジュールから論理的に独立し、デバイスの連続したエリアに物理的に制約が付けられたブロックを選択してください。
- 選択したモジュールを使用して効率的な階層を構築します。各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。デザインがパーティションされる箇所によっては、結果の質に多大な影響があることがあります。OOC インプリメンテーション用に適切なモジュールをグループ分けするために、階層を追加または変更する必要のあることもあります。
- モジュール内に完全に含まれるクリティカル パスは下位モジュールまたは最上位モジュールのいずれかに保持します。
- モジュール間の入力および出力にレジスタを付けて、モジュール内の最適化が最大限に実行されるようにし、最も柔軟な配置配線が実行できるようにします。
- 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提供します。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され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が実行できます。詳細は、「コマンドおよび制約」セクションの10 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。
- 専用接続は、常に OOC モジュール バウンダリをまたがって適切に処理されるわけではありません。関連するデザイン エレメントはすべて一緒にパーティションする必要があります。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 制約を含めて、配置を制御する必要があります。Pblock が使用されない場合、アセンブリ段階で配置が競合する可能性があります。
- `CONTAIN_ROUTING` プロパティをすべての OOC Pblock に追加します。このプロパティを付けない場合、配線に競合が発生する可能性があるので、`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なくなります。
- 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。

- 入れ子になった Pblock (子 Pblock) は、その入れ子になった Pblock の範囲が親 Pblock の範囲に完全に含まれていて、PARENT プロパティが正しく設定されている限り、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。
- すべてのクロック バッファー (最上位と OOC モジュールの両方) はロックされる必要があります。OOC モジュールの中のバッファーには LOC 制約を付ける必要があり、最上位のバッファーの位置は HD.CLK_SRC 制約で識別される必要があります。HD.CLK_SRC 制約の詳細については、10 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が使用される場合は、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 ピンをロックしておく必要があります。

  OOC インプリメンテーションでは、パーティション ピンに主に 2 つの役割があります。

  - 1 つ目は、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する役割です。配置に影響を与えるには、PartPin へ向かってまたは PartPin からのタイミング制約 (set_max_delay) が必要です。
  - 2 つ目のパーティション ピンの役割は、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すことにあります。PartPin がない場合、インターフェイス ネットは配線できません。作成した OOC 結果は、再利用中にインターフェイス ネットで必要とされる内部からモジュール ブロック配線リソースへ配線できます。これにより、配線不可能なデザインにならずに、OOC ネットすべてが保持されなくなります。

  HD.PARTPIN_RANGE および HD.PARTPIN_LOCS 制約の詳細は表 6 の「コンテキスト制約」、set_max_delay 詳細については、表 4 の「タイミング制約」を参照してください。

コンポーネント間の専用接続

専用接続のあるコンポーネントをデザインの同じパーティションに保持することが推奨されたり、必要とされることがあります。OOC モジュールのバウンダリにまたがる専用接続があると、パフォーマンスが落ちたり、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。次は、専用接続を含むコンポーネントのリストです。

- **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 を含む 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。

デザインの異なるパーティションに相互接続する I/O コンポーネントは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により、IDELAYCTRL が挿入されて、その結果が Top 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。

- 複数の OOC モジュール (それぞれに独自の IDELAYCTRL が含まれる場合) は、同じクロック領域を共有できません。
- IDELAYCTRL の付いた OOC モジュールは、100% 保持されません。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した方が結果が改善することがあります。インプリメンテーション ツールを使用すれば、I/O コンポーネントが完全に表示されるので、最も効率的に配置できます。ただし、すべての状況のおいてこれが実

行できる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)。このような場合、OOC モジュール内のロジックは 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。

- **リージョン 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が、BUFR または BUFHCE が最上位デザインに含まれる場合、OOC Pblock がバッファーのアクセスよりも多くのクロック領域にまたがるので、さらに多くの情報を提供する必要があります。入れ子になった 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される必要があります。このような Pblock には、BUFR または BUFHCE で駆動されるすべてのセルが含まれます。この Pblock は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}
```

これは、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 ポートに同じクロック領域のロードが含まれる場合、すべての適用可能なポートは上記の add_cells_to_pblock コマンドでリストできます。入れ子になった 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対応する OOC モジュールの Pblock 範囲間が一致しないと、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、クロック ネットは OOC インプリメンテーションで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ません。クロック遅延/スキューはタイミング概算により判断されます。この場合、HD.CLK_SRC 制約を使用するとタイミング概算が改善できます。この制約により、ドライバーの位置およびタイプがツールに伝わり、CPR (Clock Pessimism Removal) が計算されてタイミング概算が改善します。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 は、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) とその DRC をテストするためのデザイン ルールについて示しています。

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	リコンフィギャブル モジュールには、定義済みの Pblock が含まれる必要があります。
HDOOC-2	エラー	HD モジュールでは、一部の Pblock プロパティが定義されている必要があります。
HDOOC-3	エラー	OOC モジュールの場合、ビットストリームは生成できません。
HDOOC-4	エラー	セルに対する 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、これだけでデザインを完全に復元することができます。

保存されたチェックポイントは、元々生成されていたのと同じパーツ/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。チェックポイントに関する詳細は、『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「スナップショットの保存と復元」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 ベースのコマンドなし) でのみサポートされています。スクリプト例およびこのフローの設定手順については、『Vivado Design Suite チュートリアル : 階層デザイン』(UG946) [参照 6] を参照してください。

次のセクションでは、階層デザイン フローで使用される特定のアウト 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のこれらのコマンドの使用例を示します。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したアウト 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、ツールがアウト オブ コンテキスト モードで実行される必要があります。これ以外の場合、アウト オブ コンテキスト フローを実行するために使用されるコマンドはほかのフローと同じになります。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。次は、サポートされるツールのリストです。

- [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。これは 7 シリーズのみでサポートされています。

  注記 : XST は新しい Vivado デザインや、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 synthesis] : 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) は、各モジュールにそれぞれ合成 run がある合成フローです。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。

この文書では、Vivado 合成フローのみについて説明します。Synplify フローに関する詳細は、Synopsys 社の 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 モードで実行されます。

```
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。このモードは、write_checkpoint が使用されるとチェックポイントに保存されます。
-flatten_hierarchy rebuilt	-flatten_hierarchy に使用できる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	合成され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 より前に set_property top <top_module_name> [current_fileset] が実行されている場合、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。open_checkpoint を使用すると、合成結果は後で読み出すことができます。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 合成に渡されます。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 コンテキスト制約はすべて正しく USED_IN out_of_context とマークされていることが重要です。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、10 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。最上位デザインにインスタンシエートされたインスタンスが複数このモジュールに含まれ、アセンブリ フローが使用される場合、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 制約の付いた OOC インプリメンテーションが複数必要になります。

synth_design -mode out_of_context が以前に実行され、結果がまだメモリに含まれる場合は、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にのみ適用する必要のあることもあります。OOC のみの制約の詳細については、10 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 として識別し、再利用フロー中に DRC と必要なソフトウェア機能を有効にします。OOC インプリメンテーションに必須なわけではありませんが、再利用フロー中に最終的な DRC が正しくインポートされるように設定されているかどうか確認する良い方法です。HD.PARTITION の再利用フローでの役割については、15 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 (オプションですが、推奨)
- place_design
- phys_opt_design (オプションですが、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。これは、次の方法のいずれかで実行できます。

- **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 ツールの特定チェックおよび最適化をオンにします。
-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	インプリメントされ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] が実行されている場合、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 オプションを付けない場合、その後のインプリメンテーション段階でデザインが完全デザインとして処理され、ソースのない信号やロードのない信号がすべて削除されます。synth_design または link_design のを実行中には、OOC モジュールを定義して、モジュール解析フローを実行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 オプションは使用できません。このモジュール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書き出す際にはツールが正しいモードであるかどうかを必ず確認してください。

- **方法 3 : さまざまなファイル タイプを追加**

add_files コマンドを link_design と一緒に使用すると、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に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を使用したデザインの場合、次の制約のいずれも絶対に必要というわけではありません。より正確なタイミング解析には、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ってきます。OOC モジュールを含むデザインを問題なくアセンブルするには、これらの制約を使用して、物理リソースが適切に分配され、クロック関係が理解され、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。各モジュールの制約を設定しておかないと、アセンブリ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 使用目的にのみ設定する

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。特定の XDC ファイルの制約が OOC フローにのみ使用されるように指定すると、これらの制約にマーカーが追加され、それらが OOC 以外のデザインに読み込まれた場合に無視されるようになります。

- **方法 1 :** `read_xdc` の使用

  `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

  ファイルが `add_files` コマンドで追加される際に、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します。XDC ファイルが使用されるフローすべて (合成やインプリメンテーションなど) 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` を使用しても、ファイルを追加はできません。

制約の構文

次の表は、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べきタイミング制約、配置制約、コンテキスト制約をリストしています。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、このセクションの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) [参照 6] を参照してください。

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するのに使用して、OOC モジュールで許容される時間を概算します。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するのに使用します。`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ても、OOC モジュールにインスタンシエートされていても、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になるクロックのばらつきを定義するために使用します。OC モジュールのすべてのクロックに対して定義して、タイミング解析が正確になるようにします。定義しないと、モジュールがインポートされたときに、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 に設定しておかないと、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の要素に入れられ、最終値がユーザーの定義した値とは異なってしまいます。

表 4 : タイミング制約

制約名	説明
set_clock_latency	OOC モジュールへの入力になる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の範囲は、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などすべてが含まれます。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 を作成するコマンドです。
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 に含まれるインスタンスを指定します。通常、個別インスタンスに対する階層レベルを指定します。OOC インプリメンテーションの場合は、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します。
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。デフォルトの値は false です。Pblock 範囲に完全に含まれるパスのみが含まれます (例 : BUFGMUX の範囲がない場合、BUFGMUX を出入りす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必要とされるビヘイビアーです)。
EXCLUDE_PLACEMENT	定義された Pblock 範囲内で 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となるので、値を false のままにしておくことをお勧めします。
PARENT	Pblock 階層を識別するために使用される Pblock プロパティで、値は子 Pblock が完全に含まれる親 Pblock の名前になります。OOC インプリメンテーションを含むネストされた Pblock はすべて PARENT キーワードを使用してモジュール再利用中の動作を正しくしておく必要があります。

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、入れ子になった Pblock を使用できます。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 プロパティの `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。

前述のタイミング制約および物理制約だけでなく、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、クロック バッファーが OOC モジュール外で使用される場合に、それをインプリメンテーション ツールに伝えるため使用します。値は、クロック バッファー インスタンスの位置になります。これはクロック ポートに適用され、ポートにはこの制約が適用されるよりも前に定義済みクロック (`create_clock`) が含まれる必要があります。
`HD.PARTPIN_LOCS`	配線される指定ポートにインターコネクト タイル (`INT`) を定義します。`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、BRAM) 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 (例 : クロックまたは最上位 I/O パッドへの直接接続など)。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、追加で最適化が実行できるようになります。
`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、追加で最適化が実行できるようになります。
`set_logic_zero`	最上位で GND で駆動される指定した入力ポートに対して、追加で最適化が実行できるようになります。

重要 : `set_logic` バウンダリ最適化制約を間違って指定すると、不正なビヘイビアおよびツール エラーが発生する可能性があります。たとえば、出力ポートを OCC モジュールで未接続として定義したのに、実際にはそれが最上位で使用される場合、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の場合、デフォルトではインターフェイス ネット (OCC 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をロックする必要があります。配置配線モジュール ポート (またはパーティション ピン) の配置を素早く取得するには、HD.PARTPIN_RANGE に OOC モジュールの Pblock SLICE 範囲の値を付けます。これらのピンのさらに詳細な配置を取得するには、さらに厳密な HD.PARTPIN_RANGE 値を使用するか、明示的な HD.PARTPIN_LOCS 値を指定します。最適なサイトや範囲を決定するには、Vivado IDE で [Device] ビューを開いて、次の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 のように 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)。

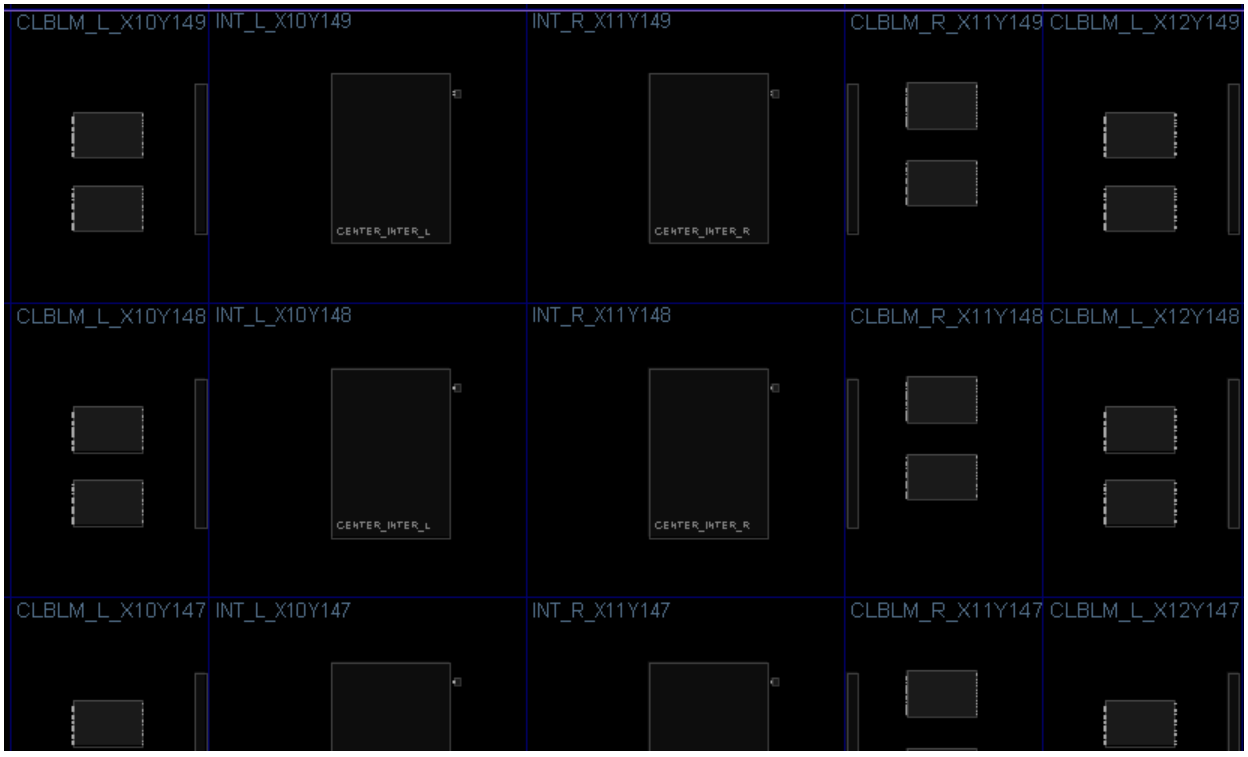

図 1 : INT タイルの位置

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。これらのコマンドおよび制約を使用したスクリプト例およびこのフローの設定手順については、『Vivado Design Suite チュートリアル : 階層デザイン』(UG946) [参照 6] を参照してください。

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、前にインプリメントしたモジュールをモジュール解析フローから読み込む必要があります。パーティション分割された各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。また、それぞれのチェックポイントは最上位デザインに読み込まれ、OOC モジュールごとにブラック ボックスが含まれます。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ができません。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 (Top) をインプリメントできるようになります。

合成

パーティション分割されたインスタンスそれぞれに対するブラック ボックスを含む最上位ネットリストが必ず必要です。これには、最上位合成に、パーティション分割済みのインスタンスのモジュール/エンティティ宣言が含まれ、ロジックは含まれないようにする必要があります。

最上位合成は、通常すべての最上位ポートで I/O バッファーを推論しますが、I/O バッファーが OOC モジュールに特にインスタンシエートされる場合は、ポートごとに最上位合成で I/O バッファー挿入をオフにする必要があります。Vivado 合成ではこの属性は `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「合成属性」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で回避されるブラック ボックス セルそれぞれに 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、最上位ブラック ボックス インスタンスに設定すると、セルが正しくマークされているかどうか確認しやすくなります。

2. パーティション分割されたインスタンスごとに OOC チェックポイントを読み込みます。

2. 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。

- 特定のセルに DONT_TOUCH を設定すると、HD バウンダリ中または HD バウンダリ上で不正な最適化がされないようになります。
- 階層デザイン用の DRC を開始します。
- `read_checkpoint -cell` で特定のコードをイネーブルにし、PartPin を一掃して余分な Pblock を削除します。

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

OOC チェックポイントの読み込み

OOC チェックポイントは、read_checkpoint コマンドに -cell オプションを付けて読み込みます。最上位デザインは既に開いており、パーティション分割されたインスタンスにそれぞれブラック ボックスが含まれる必要があります。

```
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り、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、最初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 ロケーションはすべて理想的なロケーションではない可能性があり、配線ツールで不必要な制限が課せら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 が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>]
```

このコマンドを使用する場合、-level オプションで次の値を指定すると、保持レベルが指定できます。

- **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が保持されます。配置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中に CONTAIN_ROUTING プロパティが Pblock に使用される必要があります。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しなくなります。

必要な保持レベルに関係なく、物理データベース全体は配線も含めてまだ読み込まれますが、ツールで結果が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。使用できる値は、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、デザイン全体の固定が解除されます。ロック ビヘイビアーと正反対の動作になるロック解除には、-level パラメーターを指定する必要があります。routing を指定すると、配線のロックを解除するのみですが、placement を指定すると配置配線のロックが解除されます。
<cell>	ロックされる階層セル名です。セルが指定されない場合は、全デザインがロックされます。

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場合は、すべての標準的なデザイン制約を適用できます。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

このフローを実行するには、『Vivado Design Suite チュートリアル : 階層デザイン』(UG946) [参照 6] に含まれる Tcl スクリプトを使用します。これらのスクリプトは、チュートリアルを実行する際に使用するデザイン ファイルを含む ZIP ファイルに含まれます。提供されているスクリプトの詳細は、*<Extract_Dir>*/Tcl ディレクトリにある ZIP ファイルの <Extract_Dir>/Tcl ディレクトリの README.txt ファイルを参照してください。

既知の問題

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について説明します。

UltraScale サポート

UltraScale デバイスは、現時点ではデザイン解析しかサポートされていません。再利用フローは、このリリースではベータ版になります。既知の制限などはありませんが、フローはまだ完全には保証されていません。

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。配線の概算が使用されます。HD.CLK_SRC を使用すると配線の概算は改善されます。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付けると、このタイミング概算にタイミングの正確な結果が表示されるようになります。必要なインターフェイス制約の例およびそれらの制約を自動生成させる方法については、『Vivado Design Suite チュートリアル : 階層デザイン』(UG946) [参照 6] を参照してください。

IDELAYCTRL の付いた OOC モジュールの制限

IDELAYCTRL の付いた OOC モジュールはインポートはできますが、ロックできませんので、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ません。

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、次のザイリンクス 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。トピックには、デザイン アシスタント、アドバイザリ、トラブルシュート 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 チュートリアル : 階層デザイン』(UG905)
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、本書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 コースおよびオンライン 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 ビデオ チュートリアル
2. Vivado でのアドバンスド FPGA 設計トレーニング コース

法的通知

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。