7 シリーズ FPGA マイグレーション

メソドロジ ガイド

UG429 (v1.0) 2011 年 3 月 17 日

XILINX®

本資料は英語版 (v1.0) を翻訳したもので、内容に相違が生じる場合には原文を優先します。
資料によっては英語版の更新に対応していないものがあります。
日本語版は参考用としてご使用の上、最新情報につきましては、必ず最新英語版をご参照ください。

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
2011 年 3 月 17 日	1.0	初版リリース

目次

このユーザー ガイドについて

ザイリンクス 7 シリーズ FPGA には、最適な電力、性能、コストが得られるように共通デザインが拡張でき、最も低い消費電力を実現する 3 つの統一された FPGA ファミリがあります。Artix™-7 は、最大規模の量産アプリケーション向けに、最も低いコストおよび絶対電力に最適化されたファミリです。Virtex®-7 ファミリは、最高のシステム性能と容量が得られるように最適化されています。Kintex™-7 ファミリは、コスト パフォーマンスに最も優れた新しいクラスの FPGA です。

このガイドでは、Virtex-6 デバイスを使用したデザインを 7 シリーズ FPGA に移行する方法を解説します。Virtex-6 デバイスと 7 シリーズ デバイスには多くの類似点があります。7 シリーズ FPGA デザインへのターゲットまたはリターゲットには、たいていの場合、大がかりな計画や既存デザインの変更は必要ありません。ただし、7 シリーズ アーキテクチャの基盤となる革新技術を最大限に活かすには、これらのデバイス間の変更点や差異について配慮する必要があります。このガイドで概説するガイドラインに従ってインプリメンテーションを進めることで、集積度 (コスト)、性能、消費電力を改善できます。

内容

このユーザー ガイドは、次の各章で構成されています。

- 第 1 章「7 シリーズ デバイスのターゲット/リターゲットに関する注意事項」
- 第 2 章「Virtex-6 FPGA からのリターゲットに関する注意事項」

その他のリソース

アンサー データベース、資料、ダウンロード、フォーラムなどのサポート リソースを利用するには、次のウェブサイトにアクセスしてください。

http://japan.xilinx.com/support

XILINX®

第 1 章

7 シリーズ デバイスのターゲット/リターゲットに関する注意事項

この章では、特定のテクノロジに依存しない、以前の FPGA デザインインプリメンテーションの移行について解説します。従来の FPGA または ASIC をターゲットとした既存のデザイン ソースを評価する場合、または 7 シリーズ デバイスで使用する新規コードを開発する場合の参考としてください。

HDL コーディング時の注意事項

制御信号の使用法

制御信号 (クロック、セット、リセット、クロック イネーブルなどの同期エレメントを制御する信号) の使用方法が、デバイスの集積度、使用率、性能に影響を及ぼすことがあります。これは、ほぼすべての FPGA テクノロジに当てはまることですが、7 シリーズ デバイスでは制御信号の選択や使用にあたって、そのトポロジの差異を考慮する必要があります。次に説明する事項に注意することは、デバイス使用率と性能の最適化に不可欠です。

レジスタまたはラッチでセットおよびリセットの両方を使用しないこと

Virtex®-6 および Spartan®-6 のアーキテクチャと同様に、7 シリーズ デバイスには REV ピンがありません。このため、フリップフロップにはロジックを追加しない限りセット信号とリセット信号の両方はインプリメントできません。したがって、同期セットおよび同期リセットの両方を使用するには、データ パスに信号を追加する必要があり、配置、ファンアウト、タイミング次第でエリアやタイミングに影響が及ぼす可能性があります。場合によっては、ロジック レベルを増加させることなく信号が追加でき、そのようなケースではデザインへの実質的な影響はほとんどありません。これに対して非同期のセットおよびリセットは、リソース使用率やタイミングへの影響がより顕著であるため、これらの併用は避ける必要があります。

非同期のリセット/セット信号および動的な値を持つ非同期制御信号の両方、またはいずれか一方を備えたレジスタのインプリメントは可能です。ただし、結果として得られる回路は目標よりも多くのリソースを消費し、タイミングおよび検証に当初の予想よりも大きな影響を与える可能性があります。

このコンフィギュレーションは RTL で記述するか、HDL、EDIF、または NGC フォーマットで FDCPE を用いてインスタンシエートできます。次の簡単なコード例は、非同期のセットとリセットを記述した Verilog コードです。このコードではリソースおよびタイミング パスが増加します。

```
always @(posedge reset, posedge set, posedge clk)
  if (reset)
    a_reg <= 1'b0;
  else if (set)
```

```
      a_reg <= 1'b1;
    else
      a_reg <= A;
```

2 つ目のコード例は、動的な値を持つ非同期の制御信号を記述した VHDL コードです。このコードではリソースおよびタイミング パスが増加します。

```
process (clk, initc) begin
   if initc='1' then
      data_reg <= init_signal;
   elsif (clk'event and clk='1') then
      data_reg <= data_in;
   end if;
end process;
```

ソフトウェアがこれらのコンフィギュレーションを検出すると、問題点を指摘する警告メッセージが表示され、該当するレジスタの一覧が示されます。合成に XST (Xilinx Synthesis Technology) を使用した場合、次のような警告が表示されます。

```
WARNING:Xst:3001 - This design contains one or more registers or latches
with an active asynchronous set and asynchronous reset. While this
circuit can be built, it creates a sub-optimal implementation in terms
of area, power and performance. For a more optimal implementation Xilinx
highly recommends one of these:
   1) Remove either the set or reset from all registers and latches if
   not needed for required functionality
   2) Modify the code in order to produce a synchronous set and/or reset
   (both is preferred)
   3) Use the -async_to_sync option to transform the asynchronous
   set/reset to synchronous operation (timing simulation highly
   recommended when using this option)

Please refer to http://www.xilinx.com search string "Virtex7
asynchronous set/reset" for more details.

List of register instances with asynchronous set and reset:
My_async_reg in unit <async_set_and_reset_module_or_entity>
```

サードパーティの合成ツールを使用した場合も、同様の警告が表示される可能性があります。デザインにネットリストまたはインスタンシエートされた FDCPE コンポーネントが含まれる場合は、マップで次のようなメッセージが表示されることがあります。

```
WARNING: MapLib:1182 - One or more latches or registers which have both
an active asynchronous set and reset have been found in the design. To
get this same functionality in the virtex7 architecture a sub-optimal
circuit must be created in terms of area, power and performance.
Therefore it is highly suggested to either remove one set or reset or
make the function synchronous in order to obtain a more optimal
implementation in this architecture.

List of register instances with set and reset:
  My_async_reg
```

これらの警告が表示された場合は、非同期のセットおよびリセット条件の両方を記述する必要のないコードに変更できないか検討することを推奨します。

レジスタの初期化

多くのエンジニアは、推論されたレジスタの初期化を明示的に指定することで、FPGA のレジスタおよびラッチがグローバル セット/リセット (GSR) 信号を介して初期化されるという本来の形を使用し、これによって、より信頼性が高く小型の回路を実現できます。初期化によって、リセット ステートから別のステートでのスタートアップが可能となり、初期化しない場合に比べてロジック消費量が少なくなる場合があります。たとえば、スタートアップ時には動作させる必要があっても、後続のリセットには不要となり得るステートを含むステート マシンがその例です。初期化のもう 1 つの利点として、実際の FPGA の動作により近い正確な RTL を記述できるため、回路表現の精度が高まります。このような点から、推論されたすべてのレジスタ、SRL、RAM に対して可能な限りレジスタの初期化を使用することを推奨します。

次のコード例では、レジスタ reg の値を 1 に初期化しています。

```
signal reg: std_logic := '1';
...
process (clk) begin
  if (clk'event and clk='1') then
    if (rst='1') then
      reg <= '0';
    else
      reg <= val;
    end if;
  end if;
end process;
```

コード例では、初期化によって、ロジック 1 の初期状態を作成するためだけのセット条件の指定が不要になっています。初期状態がリセット状態と同じ場合でも、レジスタの初期化を推奨します。これは初期化を使用することで、リセット状態を獲得しなくともシミュレーションのスタートアップ時のステートが FPGA の初期状態をより正確に反映したものになるためです。

アクティブ Low の制御信号の使用を控えること

推論またはインスタンシエートされたコンポーネントでアクティブ Low 信号を使用することは、次のような理由から推奨できません。

- 7 シリーズ デバイスではスライスの構成単位が粗いこと (スライスあたり 8 レジスタ)
- 7 シリーズ デバイスではスライス制御信号にプログラム可能な反転エレメントがないこと
- 階層デザイン手法では、階層の境界をまたいだ最適化 (KEEP_HIERARCHY、パーティション、パーシャル リコンフィギュレーション、ボトムアップ合成などの適用) を使用できないこと

アクティブ Low の制御信号があるために LUT を 1 つのインバーターとして使用したり、レジスタ パッキングの制約が増加したりする状況では、デバイス使用率が増加する場合があります。

タイミングもアクティブ Low 信号の影響を受ける可能性があります。HDL コードまたはインスタンシエートされたコンポーネントでは、可能な限りアクティブ High の制御信号を使用してください。制御信号が外部のプログラム不可のソースによって駆動されているなど、極性をデザイン内で制御できないときは、コードの最上位階層の信号を反転させ、極性 (機能) が同じになるようインバーターが駆動するアクティブ High の制御信号を記述してください。このような方法で記述することでインバーターを I/O ロジック内に吸収でき、追加ロジックまたは配線が不要となり、使用率、性能、消費電力が改善されます。

ファンアウトの少ない制御信号の使用を控えること

デザインで固有の制御信号は、デザインの機能および動作に必要なものに限って使用するようにします。ファンアウトが小さい固有の制御信号は、レジスタ、SRL、LUT RAM などでスライスの使用効率を低下させる場合があります。また、配置やタイミングにも悪影響を及ぼす可能性があります。通常、セット、リセット、クロック イネーブル信号は、デザインの機能性から積極的に必要とされる場合以外、コードでインプリメントしないようにしてください。

セットまたはリセットを多用しないこと

コード不必要なセットおよびリセットがあると、SRL、RAM (LUT RAM またはブロック RAM)、その他のロジック構造の推論を妨げる恐れがあります。アーキテクチャから最大限の効率を引き出すには、デザインの機能性から積極的に必要とされる場合にのみ、セットおよびリセットをコードで記述します。

不必要なセットおよびリセットはコード化しないでください。たとえば、レジスタはコンフィギュレーションの完了時に自動的に初期化されるため、初期化にのみ使用されるリセットは不要です。

リセットが不要なもう 1 つの例は、長期間にわたってアイドル状態となる回路です。このような回路では入力レジスタをリセットすれば、その他の回路のデータはいずれフラッシュされます。

最後の例は、複数クロック サイクル間リセットが保持される場合の、内部に位置するレジスタのリセットです。このようなレジスタはリセット中にフラッシュされるため、すべてのレジスタにリセットを付ける必要はありません。不要なセットまたはリセットを使用しないことで、デバイスの使用率、配置、性能、消費電力を改善できます。

DSP48E1 スライス レジスタの乗算器または加減算器のセット

7 シリーズの DSP48E1 スライスに含まれるレジスタにはリセット信号だけがあり、セット信号はありません。7 シリーズの DSP ブロックは、乗算、加減算、コンパレーター、カウンター、汎用ロジックなど幅広い機能を実行できます。このような追加リソースをデザインで柔軟に活用できるようにするため、機能は適切にマッピングされる必要あり、その機能にセット条件を付けることはできません。したがって、乗算器、加算器、カウンターなど DSP48E1 スライスにインプリメント可能なロジック周辺では、必要な場合を除いてセット信号 (値がロジック 1 になる信号) をコード化しないでください。

同期セット/リセットの使用法

回路の適切な動作にセットまたはリセットが必要な場合は、必ず同期リセットをコード化してください。同期セット/リセットはタイミング特性および安定性に優れ、FPGA 内のロジック規模を縮小して使用率を改善します。

同期セット/リセットは、数 LUT 分のロジック削減、パッキング時の制約の軽減、そして多くの場合に回路の高速化をもたらします。DSP48E1 や RAMB36E1/RAMB18E1 などのブロック内にあるレジスタには、非同期セット/リセットはインプリメントできません。したがって、RTL コード内に非同期のセットまたはリセットがある場合、これらのレジスタは等価機能のものとしては使用できません。同期リセットを使用すれば、スライス内のレジスタ共有に対する柔軟性が大幅に向上します。各スライスのリセットが共有され、互換性のない同期リセットを持つレジスタをデータパスにマッピングし直すことができるため、それらを同一スライス内に配置することも可能になります。これに対して非同期のリセットでは再マッピングできず、同じ機能を保つことはできません。したがって、異なる非同期リセットを持つ 2 つのレジスタ、あるいは非同期リセットを持つレジスタと持たないレジスタの 2 つを同一スライスへのマッピングは不可能です。これはデバイスの集積度に影響するだけでなく、配置が最適化されず、配線遅延やキャパシタンスが増加するため性能および消費電力に悪影響を及ぼします。

既存の非同期リセットを同期リセットにするコード修正を望まない場合、合成ツールに [Asynchronous To Synchronous] スイッチがあれば、これを使用して非同期リセットを同期リセットとして扱うことができます。ただし、このスイッチを使用した場合、RTL ハードウェア記述とインプリメントされたデザインが、必ずしもすべてのリセット条件で同じように動作しない可能性が生じます。このため、回路検証は十分慎重に行ってください。

合成ツールとして XST を使用する場合、[Asynchronous To Synchronous] スイッチは ISE® ソフトウェア Project Navigator のプロパティから選択できます。コマンドラインから実行する場合は、合成のオプションとして **`-async_to_sync`** スイッチを使用します。このオプションは同期セット/リセットを使用するようコードを変更するほどはリソース削減や性能向上点で効果的ではありません。ただし、オプションを適用しないときと比較するとレジスタのさらなるパッキングや最適化が可能であり、回路が小型化および場合によっては高速化されます。

クロック イネーブルの使用法

ファンアウトの大きいクロック イネーブルは手動で分割または複製せずに、単一のクロック イネーブルとしてコード化します。タイミングなどの理由で複製が必要になった場合は、合成ツールで処理してください。これによって、複製を多数回実行する場合でも柔軟に処理され、リソースや消費電力の増大と性能向上間でより適切なトレードオフ バランスが得られます。配置やその他の要因が変化することで、複製に対する要件が変化することがあります。この場合、合成およびインプリメンテーション ツールでこれを管理していれば、元のコードを修正し、変更箇所を再検証する必要がありません。

ファンアウトの大きいクロック イネーブルを単一のネットに抑えることのもう 1 つのメリットは、BUFGCE または BUFHCE などのその他の専用リソースへの再マッピングが容易になることです。これより、同じ機能を実現しながら、消費電力と配線リソース消費両方の大幅な削減が可能になります。

DSP その他の演算中心のコードの使用法

7 シリーズは多くの DSP デザインに非常に適したアーキテクチャを提供します。このアーキテクチャを活用するには、その基盤となる機能や性能を理解し、これらのリソースのメリットを最大限引き出すデザイン入力コードを作成する必要があります。

DSP48E1 ブロックは符号付きの演算インプリメンテーションを使用します。HDL ソースでは、リソースの性能と最適に整合し、通常は最も効率的なマッピングを実現するために符合付きの値を使用してコード化することを推奨します。合成ツールでは、コード内で符号なしのバス値を使用しているリソースも使用できますが、符号なしから符号付きの値への変換によって、コンポーネントのビット精度が完全には保たれない可能性があります。7 シリーズ DSP48E1 スライスの乗算器は、符号付きデータの入力ビット精度が 18 ビット × 25 ビットです。したがって、符号なしデータの精度は 17 ビット × 24 ビットになります。Verilog コードでは、コード内で宣言しない限り、データはすべて符号なしの値とみなされます。

ターゲット デザインに加算器が多数含まれると予想される場合は、DSP48E1 スライスの前置加算器および後置加算器の使用数を増やすことができないかを検討することを推奨します。たとえば FIR フィルターの場合、複数の連続した加算ファンクション (加算器ツリー) ではなく、加算器のカスケード接続によってシストリック フィルターを構築できます。対称フィルターの場合は、専用の前置加算器を使用することで、機能をより少数の LUT やフリップフロップ、さらにはより少数の DSP スライスに統合できます (ほとんどの場合リソースが半減します)。

加算器ツリーが必要な場合は、6 入力の LUT アーキテクチャを活用すれば、単純な 2 入力加算と同じリソース量で 3 入力加算 (A + B + C = D) を効率的に構築できます。このような構成を取ることにより、必要に応じてキャリー ロジック リソースを削減、節約できます。多くの場合、これらの手法

は必要ありません。しかし、このような機能を知っておけば、あらかじめ適切なトレードオフを念頭に置いて RTL コードを作成でき、デザイン着手時からより円滑で効率的なインプリメンテーションが可能になります。

ほとんどの場合、DSP リソースは推論による実装が適しています。合成に XST を使用する場合は、『XST ユーザー ガイド』(UG627) の「XST ハードウェア記述言語 (HDL) コーディング手法」の章を参照することを推奨します。また、DSP48E1 スライスの機能と性能、およびこのリソースを活用してデザイン要件を効果的に満たす方法については『7 シリーズ FPGA DSP48E1 スライス ユーザー ガイド』(UG479) を参考にしてください。

RAM に関する注意事項

推論、プリミティブのインスタンシエーション、Unimacro、CORE Generator™ ソフトウェアによってブロック RAM および LUT RAM をリターゲットする場合、7 シリーズ アーキテクチャのブロック RAM と LUT を最大限に使用するために、いくつか注意すべき点があります。CORE Generator ソフトウェアで RAM を生成する場合は、合成時に適切に推論されるように 7 シリーズ デバイス向けに IP を再生成するか、RAM のコードを再作成する必要があります。

いずれの方法でも、使用率および性能に良い結果をもたらします。しかし、コードおよびシミュレーションがより理解しやすいものになり、今後のコード移植も容易になるため、メモリは可能な限り推論することを推奨します。

RAM のインスタンシエーション

このセクションでは、デザインで RAM プリミティブをインスタンシエートする場合、または 7 シリーズ デバイス向けに CORE Generator ソフトウェア IP を再生成できない場合の推奨事項を示します。これらは RAM を推論するコード、特に使用する RAM リソースを合成時の属性で指定するコードに対しても適用してください。推奨事項は、使用する RAM リソースを判断するうえで最も重要な要因であるワード数別に示します。

ワード数が 128 ビット未満の場合

7 シリーズ FPGA では LUT が大きく、LUT RAM のワード数も多いため、ブロック RAM と LUT RAM のいずれを選択するかの基準は前世代の FPGA とは異なる場合があります。一般的に、ターゲット デバイスのロジック リソース (LUT) および SLICEM (または、そのいずれか) が不足していない限り、64 ビット以下のメモリには、すべて LUT RAM を使用してください。

64 ビット以下のメモリでは、データ幅に関係なく LUT RAM を使用した方が、リソース、性能、消費電力の面で効率が向上します。64 ビットから 128 ビットのメモリの場合、最適なリソースの判断は次の要因に依存します。

1. 追加ブロック RAM の使用可否。ブロック RAM が使用できない場合は LUT RAM を使用します。
2. レイテンシの要件。非同期読み出し機能が必要な場合、LUT RAM を使用する必要があります。
3. データ幅。データ幅が 16 ビットを超える場合、可能であればブロック RAM を使用します。
4. 必要な性能要件。通常、レジスタ付きの LUT RAM の方がブロック RAM よりもクロックからの出力遅延が短く、配置に関する制約も少なくなります。デザインに 16 ビットを超えるワード数の LUT RAM が既にインスタンシエートされている場合は、よりワード数の多いプリミティブ (たとえば、RAM32X1S または RAM64X1S) を使用します。

RAM16X1S コンポーネントと MUXF5 コンポーネントまたはその他のロジックを併用していると、よりワード数の多い LUT への自動リターゲットが適切に行われません。その場合は、コードを変更して、ワード数の多いプリミティブが適切に使用されるようにする必要があります。

ワード数が 128 ビットを超える場合

ワード数が 128 ビットより多い RAM は、ほとんどの場合ブロック RAM をターゲットにします。7 シリーズ デバイスには 18Kb RAM (RAMB18E1) および 36Kb RAM (RAMB36E1) の 2 種類のブロック RAM があります。どちらを使用するかは、通常必要な幅とワード数によって決まります。表 1-1 に、TDP (True Dual-Port) および SDP (Simple Dual-Port) コンフィギュレーションにおける RAMB18 および RAMB36 の幅/ワード数の組み合わせを示します。

表 1-1 : メモリの幅、ワード数、タイプ (True-Dual Port または Simple-Dual Port) に応じたブロック RAM プリミティブの選択基準

ワード数	メモリ幅 (ビット)			
	RAMB18 TDP	RAMB18 SDP	RAMB36 TDP	RAMB36 SDP
512	18 ビット以下	36 ビット以下	19 ～ 36 ビット	37 ～ 72 ビット
1K	18 ビット以下	18 ビット以下	19 ～ 36 ビット	19 ～ 36 ビット
2K	9 ビット以下	9 ビット以下	10 ～ 18 ビット	10 ～ 18 ビット
4K	4 ビット以下	4 ビット以下	5 ～ 9 ビット	5 ～ 9 ビット
8K	2 ビット以下	2 ビット以下	3 ～ 4 ビット	3 ～ 4 ビット
16K	1 ビット	1 ビット	2 ビット	2 ビット
32K	N/A	N/A	1 ビット	1 ビット
64K	N/A	N/A	1 ビット(1)	1 ビット(1)

メモ :

1. カスケード モードで構成した 2 つの RAMB36 コンポーネントが必要です。

その他の RAM 機能の検討

新たにデザイン プロジェクトを開始する際に、専用 RAM 機能を使用すべきかどうか判断することも優れた結果を得る設計方法です。その場合は、次の項目を検討してください。

- FIFO : 7 シリーズ デバイスのブロック RAM には、FWFT (First Word Fall-Through) 機能や、しきい値をプログラム可能な ALMOST EMPTY および ALMOST FULL フラグの付いた同期 (同一クロック) またはデュアル クロック FIFO バッファーをインプリメントする専用ロジックがあります。デザインで FIFO をソフト ロジックから作成している場合は、専用ロジックを使用することで、デバイスの使用率、消費電力、性能、さらにこれらのコンポーネントの全体的な設計のしやすさが改善できないかを検討してください。
- ECC ロジック : 7 シリーズのブロック RAM には、内容のエラー検出および訂正専用のロジックがあります。データのエラー訂正機能が必要なデザインへの適用を検討してください。
- 出力レジスタ : 出力レジスタを使用するとブロック RAM の性能 (クロックから出力の遅延) が著しく向上すると同時に、消費電力およびデバイスの使用率も改善されます。以前のアーキテクチャから 7 シリーズ アーキテクチャにデザイン移行する場合、デザインに出力レジスタを含めることが可能かという点からコードを再検討してください。
- バイト ライト イネーブル : 7 シリーズ デバイスには、バイト ライト イネーブルがあります。この機能はブロック RAM のアクセスおよびデバイスの使用率を向上させます。これを使用することで、ブロック RAM およびその他のリソースをより高い効率で利用できるようになる場合があります。

- イネーブル/リセットの優先順位：7 シリーズ FPGA ではイネーブルとリセットの優先順位が変更できるため、スライスおよび I/O レジスタに比べて出力レジスタ制御の一貫性が改善されます。

詳細は、『7 シリーズ FPGA メモリ リソース ユーザー ガイド』に記載されています。

合成および物理的制約の使用法

多くの場合、コードまたは合成制約ファイルには、前のインプリメンテーションやアーキテクチャで所望の結果が得られるように組み込まれた合成の属性、制約、および指示子が含まれます。これらの要素は、結果の質を低下させたり、今後のインプリメンテーションにとって最適な選択ではない場合があるため、コメント アウトするか削除することを推奨します。

既存デザインのコード、ネットリスト、または UCF ファイルに組み込まれた、LOC、RLOC、BEL 制約、その他の物理的制約は、7 シリーズ FPGA へのリターゲットの前にすべて削除しておく必要があります。従来のアーキテクチャにおける最適配置が、機能ブロック、デバイスのフロアプラン、およびタイミングが異なる 7 シリーズ FPGA でも最適となる可能性は低いためです。レイアウトや座標の違いによってエラーが発生する場合もあります。エラーが発生しなかったとしても、物理的制約を新しいアーキテクチャに合わせて変更、削除、または更新しない限り、タイミング、集積度、および消費電力が最適化されません。

タイミング制約の指定

合成のタイミング制約は、まず現実的なタイミング目標に関連する部分から指定します。合成ソフトウェアは、性能目標を満たしてもタイミングに過剰な余裕がある領域では、エリア節約のアルゴリズムを適用します。一方、タイミングに余裕がない領域ではタイミング最適化のアルゴリズムを適用可能です。タイミング制約が指定されていないと、合成ツールはデザインのあらゆる部分でタイミングを最適化することになり、しばしばエリアが犠牲になります。さらに、本来ならばさらに最適化できるパスをそのままに、最適化が不要な領域で最適化 (ロジック レベル削減) を実行することがあります。タイミング制約を適用することで、ツールはデザイン上で制約が必要な領域のタイミングを集中的に最適化し、それ以外の不要な場所ではタイミングを緩和します。

目的とする I/O およびクロック タイミングだけでなく、マルチサイクル パス、疑似パス、タイミング無視 (TIG) などのタイミング例外も反映したタイミング制約を適用する必要があります。現実的で完全なタイミング制約を適用すると、多くの場合、結果が改善されるだけでなく、タイミング クロージャおよびデバッグに要する時間、さらにはランタイムとメモリ必要量も削減されます。ただし必要以上にデザインに制約を加えると、ランタイムやメモリ必要量が増し、適切な制約を課したデザインほどよい結果が得られない場合があるため、制約の過剰な適用は避けてください。UCF ファイルの作成に時間を費やしたとしても、タイミング クロージャ プロセス全体では、優れた UCF ファイルによってはるかに多くの時間を節約できます。

ソフトウェア オプション

7 シリーズ FPGA のソフトウェア アルゴリズムは、デバイス エリア (つまりコスト)、消費電力、および性能間で適切なバランスが得られるように設計されています。ISE ソフトウェアのオプションにより、性能と引き換えにデバイス エリアを縮小したり、デバイス エリアと引き換えに性能を向上させたりできます。消費電力を削減するオプションもあります。オプションは多くの場合、性能、エリア、およびソフトウェアのランタイム (または、これらのいずれか) とのトレードオフになります。デフォルト設定でバランスの取れたデザイン目標が達成できない場合は、ソフトウェア オプションを指定できます。ただし、これらのオプションは、デフォルトのアルゴリズムでデザイン目標を達成できないかを判断してから使用することを推奨します。このため、最初のデザインではソ

フトウェアのオプションはデフォルトで実行してください。そして、その結果を分析した後に、必要な結果が得られるように特定のオプションを適用してアルゴリズムを調整します。

ルートスルーとしての LUT 利用

7 シリーズ FPGA の LUT 使用率を解析する際には、ルートスルーとして使用されている LUT に注意する必要があります。マップ レポートに示される LUT ルートスルーは、スライスへの入力パスがほかにない場合、最も一般的にはバイパス入力 (AX、BX、CX、DX) がすべて使用不可の場合に、内部のスライス ロジックまたはレジスタにアクセスする必要があると作成されます。LUT ルートスルーは、LUT への入力を 1 つ使用してスライスにアクセスします。ルートスルーが必要になる状況として、次のような場合が考えられます。

1. フリップフロップ、RAM、その他の LUT ではないソースがフリップフロップを駆動する場合 (スライス内に既にレジスタが 4 つ存在するなどの理由でバイパス ラインが占有されている)
2. フリップフロップ、RAM、その他の LUT ではないソースがスライス内の MUXF7/MUXF8 のデータ入力を駆動する場合
3. フリップフロップ、RAM、その他の LUT ではないソースが CARRY4 のセレクト ラインまたはデータ ライン (MUXCY のセレクト ラインと XORCY の DI または、そのいずれか一方) を駆動する場合

7 シリーズ デバイスは、性能とエリアの両方にメリットをもたらす、スライスあたり 8 つのレジスタを備えています。高速で、高度にパイプライン化された多くのデザインでは、豊富なレジスタによってロジックを追加せずに最高性能を実現できます。また、LUT の組み合わせを改善できるため、性能を損なわずに 6 入力 LUT のデュアル出力構造を活用できます。また、リセットの使用やシフトレジスタに多数のタップ ポイントが必要であるために、シフトレジスタ LUT (SRL) を使用できない場合は、レジスタによって、より効率的にシフトレジスタをインプリメントできます。

レジスタが 8 個あるため、多くの場合、LUT ではないソース (ブロック RAM、ほかのフリップフロップ、DSP スライスなど) によって駆動されるレジスタが消費するスライス数が削減されます。ザイリンクスのソフトウェア ツールでは、性能、エリア、消費電力の面で最適な特性が実現されるように、スライス レジスタの配置が調整されます。しかし、1 つのスライスに 4 個を超えるレジスタを配置する場合はしばしば LUT ルートスルーが必要となり、リソース使用率のレポートに表示されます。

デザインによっては 7 シリーズ デバイスで LUT の使用数が増加しているようにレポートされる場合がありますが、必要なスライス数はルートスルーにより減少します。通常はスライスではなく LUT の使用率が比較されるため、使用する LUT 数が増加して見えると誤解を招く可能性があります。残っているロジックを考える際には、ルートスルーは LUT の 1 入力 (通常、A6 入力ではなく、O5 出力にアクセスするための下位入力の 1 つ) および 1 出力を必要とし、残りの 4 入力 1 出力は任意のファンクションに使用できることを踏まえてください。つまり、必要なルートスルーを使用したとしても、4 入力 LUT に相当するロジックは未使用で残ります。

通常の状況では、必要に応じてレジスタを複数のスライスに分散させることで、ルートスルーを一切使用しない構成も可能です。こうすればロジック用に 6 入力 LUT 全体を確保できますが、レジスタの分散によって消費するスライス数が増加したり、性能へのペナルティが生じたりする可能性があります。ルートスルーは 1 つの LUT をすべて使用するかのように報告されますが実際にはその一部のみで、配置が異なれば LUT を 1 つも使用しない場合もあります。次に、マップ ファイル (`.mrp`) 中の、ルートスルーの使用状況とその目的を示したレポートの抜粋を示します。

```
Number used exclusively as route-thrus:    117
  Number with same-slice register load:    106
  Number with same-slice carry load:        11
  Number with other load:                    0
```

レポートのこのセクションは、ルートスルーに必要とされる LUT 数と、その理由を示しています。これは情報提供のみを目的としており、デバイス使用率を詳細に評価する際に必要に応じて使用できます。

図 1-1 は、スライス内の 8 個のレジスタすべてにアクセスするためにルートスルーを使用している例を FPGA Editor 上で示したものです。

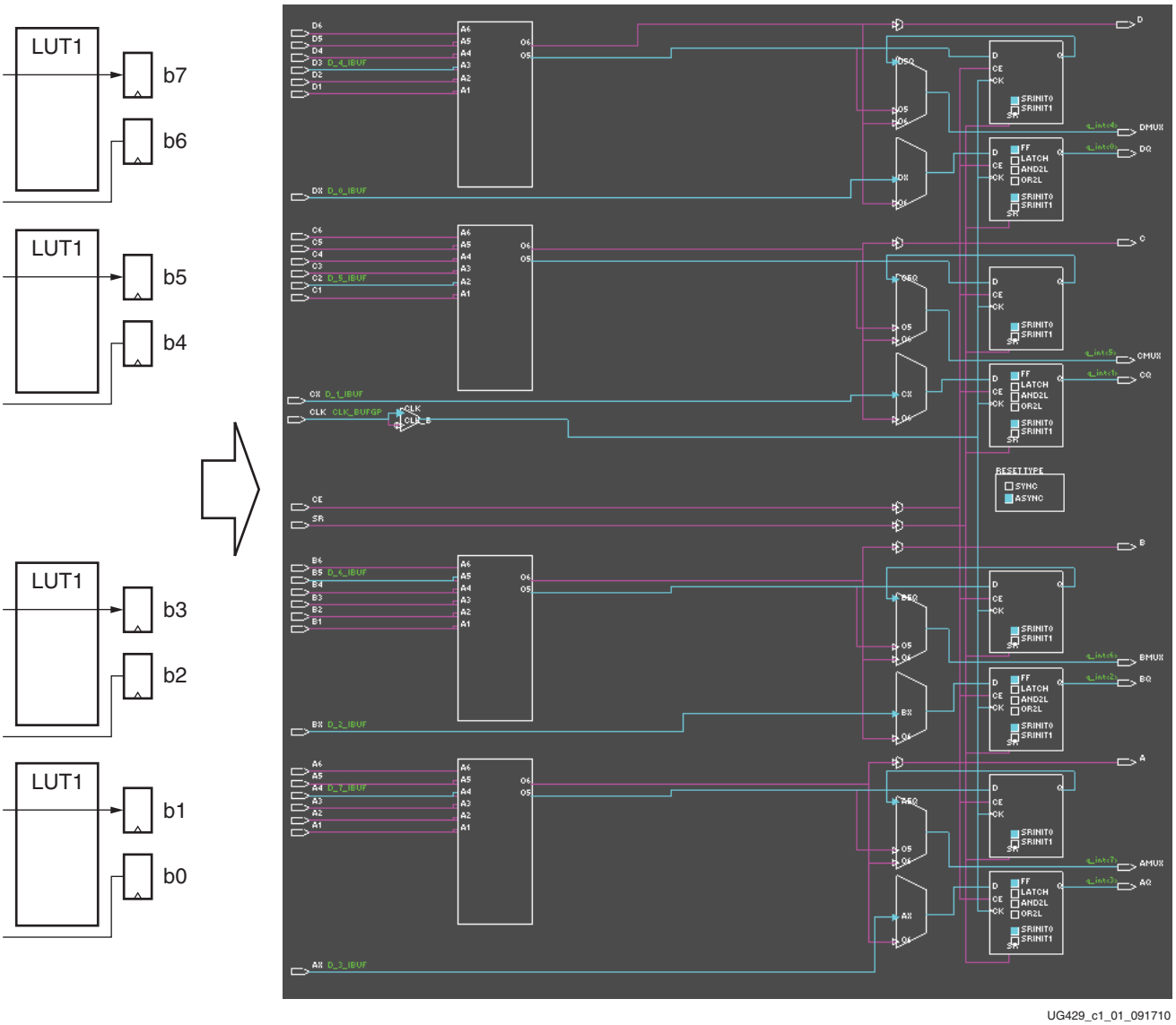

図 1-1 : ルートスルーを適用した FPGA Editor の画面

XILINX®

第 2 章

Virtex-6 FPGA からのリターゲットに関する注意事項

この章では、7 シリーズ デバイスに移行する際の、Virtex®-6 FPGA デザイン固有の注意点について解説します。

7 シリーズ デバイスの選定

リソースおよびデバイス サイズの点では、7 シリーズのアーキテクチャは Virtex-6 ファミリと類似しています。DSP、ブロック RAM、スライスの構造はほとんど変更されていないため、同タイプのデザイン構造およびコードは、Virtex-7 デバイスでも同タイプと量のリソースをターゲットとします。Virtex-6 FPGA と同様に、デバイスの型番にはそのアレイに含まれるロジック セル数が 1000 を単位に表されています。たとえば Virtex-6 XC6VLX550T には約 550,000 のロジック セルが搭載されています。7 シリーズ デバイスでも同じマトリックスと表記法を使用しており、若干サイズが大きくなったデバイス Virtex-7 XC7V585T のロジック セル数は 585,000 ということになります。このような型番表記により、ファミリ間でのロジック集積度の比較が容易になっています。Virtex-7 デバイスのロジック セル数は 285,000 以上であるため、XC6VLX240 よりも小さい Virtex-6 デバイスのリターゲットには、Kintex™-7 ファミリの方が適しているでしょう。

デバイス選定の基準はロジック集積度のみではなく、その他の必要なリソースについても評価する必要があります。まず『7 シリーズ FPGA の概要』(DS180) に掲載された機能一覧表で、関連するあらゆるリソースを比較することを推奨します。次に、7 シリーズ デバイスの選定にあたって検討すべきその他の留意点を示します。

- ブロック RAM/DSP — ブロック RAM または DSP を多く使用するデザイン用には、Virtex-6 SXT デバイスに相当する Virtex-7 拡張機能デバイス (XC7VXT) が、特定のアレイ サイズにおいて、より多くのブロック RAM と DSP を搭載しています。
- I/O — 特定アレイ サイズでの I/O 数、バンク サイズ、配置、機能は Virtex-6 ファミリとは異なります。新規デザインの開始時に、より多くの機能をデバイスに盛り込むことができないかを検討することは有益です。これによって、必要な I/O 数を抑えるだけでなく、システム全体の性能、コスト、消費電力、機能を向上できます。
- スピード グレード — Virtex-6 アーキテクチャをターゲットとしたときと同一スピード グレードにすることを推奨します。デザインの一部のブロックまたは領域で性能の向上が見られる場合でも、インプリメント後のデザインの正確な性能特性を解析できるまでは、システム全体の性能はそれほど変わらないことを前提としてください。

既存のソフト IP、EDIF、または NGC ネットリストの使用法

デザインに含まれる以前のソフト IP またはブラック ボックス ネットリストは、7 シリーズ デバイスにインプリメントする前に、再生成または再合成することを強く推奨します。Virtex-6 アーキテクチャをターゲットにしたほとんどのネットリストは 7 シリーズ デバイスをターゲットにした場合もエラーなしでインプリメントできます。しかし、7 シリーズ デバイスに最も効率よくリターゲットするために、旧アーキテクチャをターゲットにしたネットリストまたはコアは、すべて再生成してください。

クロッキングに関する注意事項

7 シリーズ アーキテクチャは、Virtex-6 アーキテクチャと同様のクロック構造を持ちます。いずれのアーキテクチャにも 32 個のグローバル バッファー (BUFG)、リージョナル バッファー (BUFIO と BUFH)、および I/O バッファー (BUFIO) があります。また、共に MMCM (Mixed-Mode Clock Manager)、クロック ゲート (BUFGCE と BUFHCE)、クロック マルチプレクサー (BUFGCTRL) も備えています。多くの場合、Virtex-6 デバイスから 7 シリーズ デバイスへの移行する際にクロッキングのトポロジを変更する必要はありません。しかし、一部の差異によって変更が必要になる場合もあります。また、7 シリーズ アーキテクチャに追加された新機能を評価し、デザイン全体の性能を向上させることができるかを検討する必要もあります。

クロック接続に BUFG を使用している Virtex-6 FPGA デザインの更新は必要ありません。7 シリーズ デバイスは、Virtex-6 FPGA と同様に 32 個のグローバル バッファー コンポーネントを持ち、類似したコネクティビティを備えています。任意のクロック領域で BUFG を 12 個までしか使用できない制限も同じですが、7 シリーズのクロック領域は Virtex-6 FPGA よりも大きくなっています。この点が問題になるとは考えられませんが、7 シリーズ デバイスへのリターゲットの際にクロック領域内で配置変更が生じる可能性はあります。BUFGCE を介してグローバル クロッキングのクロック イネーブルまたはクロック ゲーティング機能を使用する場合や BUFGCTRL または BUFGMUX でクロックをマルチプレクスする場合、これらの機能および使用法は両アーキテクチャ間で同様です。水平クロック バッファー (BUFH または BUFHCE) の使用方法も同じです。

BUFR を使用する場合、バッファー機能は 7 シリーズと Virtex-6 FPGA で同様です。BUFR の大きな違いは、Virtex-6 FPGA の BUFR は隣接する領域も駆動でき、7 シリーズ デバイスの BUFR はそれが属する領域しか駆動できない点です。したがって、BUFR が複数のクロック領域を駆動していた場合は、そのクロッキング構造を再評価する必要があります。ただし、7 シリーズ デバイスのクロック領域は Virtex-6 FPGA よりも大きいため、Virtex-6 デバイスで 2 つのクロック領域を駆動していたものが 7 シリーズ デバイスでは 1 つのクロック領域に収まる可能性があり、その場合デザインの変更は不要です。7 シリーズ デバイスをターゲットとしても複数のクロック領域を駆動する必要があるときは、デザインを変更する必要があります。リージョナル クロックによって複数の領域を駆動できますが、7 シリーズ デバイスでは、BUFMR が垂直方向に隣接する最大 3 つのクロック領域に含まれる BUFR を駆動できます。

BUFR と同様に、BUFIO によって 1 つのバンクを駆動する場合は、Virtex-6 デバイスとその機能は変わりません。7 シリーズ デバイスの BUFIO 接続は 1 つのバンクに限られます。複数のバンクを駆動する場合は、BUFMR を使用して隣接するバンクにある複数の BUFIO を駆動できます。BUFMR は BUFIO と BUFR の両方を駆動できるため、高速の I/O クロックや、複数の隣接クロック領域で位相の揃った低速の分周クロックが可能になります

Virtex-6 FPGA デザインで使用されていた MMCM は、ほとんどの場合 7 シリーズ デザインにリターゲットすれば同様に使用できます。ただし、DRP ポートを使用している場合は、プログラミングの観点で変更を与えるマッピングやデータが変更されている可能性があるため、同じ目的を達成するには修正が必要となることがあります。また、7 シリーズ デバイスには MMCM を直接カスケードする接続方法はないため、Virtex-6 FPGA デザイン内で直接カスケード接続されていた

MMCM はそのまま使用することはできません。それ以外の MMCM は自動的にリターゲット可能であり、Virtex-6 デバイスと同様の方法で使用できます。

7 シリーズ デバイスの新しいクロッキング機能も検討してください。たとえば、PLLE2 と呼ばれる新しいクロッキング コンポーネントがあります。この機能は、MMCM と同様に 2 つの UNISIM コンポーネント、PLLE2_BASE と PLLE2_ADV で表されます。たいたていのケースは PLLE2_BASE で対応できますが、より高度な状況では PLLE2_ADV を使用します。PLLE2 よりも豊富な機能を持つ MMCM が使用される場合がほとんどですが、高速の I/O クロッキングには PLL の方が有利なことがあります。7 シリーズ デバイスの BUFHCE コンポーネントは、イネーブルを非同期にゲーティングできます。この機能は、BUFH でクロック以外のファンアウトの大きな信号を駆動したり、クロックが停止あるいは失われた場合に有用です。7 シリーズの BUFH コンポーネントには CE_TYPE と呼ばれる新しい属性があります。この属性値を「SYNC」に設定すると、BUFH の動作は Virtex-6 デバイスの場合と同じになり、出力にクロック グリッチを発生させることなく同期でクロックを停止できます。「ASYNC」に設定した場合は新しい動作、つまりクロック ソース入力に関係なくクロックのゲーティングのオン/オフが可能になります。

グローバル バッファーによるクロック以外の負荷の駆動

ファンアウトの大きな信号をグローバル バッファー (BUFG) に接続して、必要なローカル配線を減らすことは珍しいことではありません。この手法は 7 シリーズ デバイスでも適用できます。Virtex-6 FPGA と同様に、クロック以外の目的に使用する BUFG の数は、競合を回避するために 2 個以下とすることを推奨します。

その他のプリミティブのリターゲットに関する注意事項

次に列挙するプリミティブは Virtex-6 と 7 シリーズ デバイスで同じです。したがって、動作、接続、一般的な使用法に変更は不要です。

AND2B1L、BUFG、BUFGCE、BUFGCTRL、BUFGMUX、BUFGMUX_1、BUFGMUX_CTRL、BUFGP、BUFH、BUFHCE、BUFIO、CARRY4、CFGLUT5、DCIRESET、DNA_PORT、DSP48E1、EFUSE_USR、FDCE、FDPE、FDRE、FDSE、IBUF、IBUFDS、IBUFDS_DIFF_OUT、IDDR、IDDR_2CLK、IDELAYCTRL、IOBUF、IOBUFDS、KEEPER、LDCE、LDPE、LUT1、LUT2、LUT3、LUT4、LUT5、LUT6、MUXF7、MUXF8、OBUF、OBUFDS、OBUFTDS、ODDR、OR2L、PULLDOWN、PULLUP、RAM128X1D、RAM256X1S、RAM32M、RAM32X1S、RAM64M、RAM64X1D、RAM64X1S、SRL16E、SRLC32E

Virtex-6 アーキテクチャ向けにインスタンシエートされたデバイス プリミティブの中には 7 シリーズ アーキテクチャに自動的にリターゲットされないものや、ターゲットされたとしても注意すべき変更を伴うものがあります。UNISIM コンポーネントの詳細は、『HDL デザインのためのザイリンクス 7 シリーズ FPGA ライブラリ ガイド』(UG768) を参照してください。また上記の各ブロックを 7 シリーズ FPGA にリターゲットする場合の詳細情報は、該当するユーザー ガイドに記載されています。

ここでは、これらのコンポーネントの一部について説明します。

BSCAN_VIRTEX6

7 シリーズ デバイスのバウンダリ スキャン コンポーネントは、Virtex-6 FPGA のコンポーネントと同じインターフェイスおよび機能を備えています。コンポーネント名は Virtex-6 FPGA を指す部分を削除して BSCANE2 に変更されましたが、使用方法は以前のコンポーネントと同様です。デザイン内の BSCAN_VIRTEX6 コンポーネントは、自動的に BSCANE2 コンポーネントにリターゲットされますが、時間が許せば、新しいコンポーネント向けにコードを更新することを推奨します。

BUFIODQS

BUFIODQS の DQSMASK 機能は、7 シリーズ デバイスでは直接サポートされません。これは MIG IP コアによるメモリ インターフェイス ストローブ用の機能でしたが、専用の Phaser 回路に移行されました。デザイン内の MIG で生成したコアにこのエレメントがある場合、その回路を適切に置き換えるために、7 シリーズ デバイス向けにコアを再生成してください。デザインに DQSMASK ピンをグランドに固定し、DQSMASK_ENABLE 属性を FALSE に設定した BUFIODQS がインスタンシエートされている場合は、駆動するバンクが 1 つだけならばこのコンポーネントを安全に BUFIO コンポーネントに置き換えることができます。複数の隣接バンクを駆動する場合は、このコンポーネントを BUFMR によって置き換え、その後に複数の BUFIO を接続します。DQSMASK 入力および DQSMASK_ENABLE 属性を使用する回路は 7 シリーズ デバイスではサポートされていないため、これらが必要な場合は代替回路を構築する必要があります。

BUFR

7 シリーズ デバイスの BUFR プリミティブは、Virtex-6 デバイスの同プリミティブと機能的に等価です。BUFR の大きなの違いは、Virtex-6 FPGA の BUFR は隣接する領域も駆動でき、7 シリーズ デバイスの BUFR はそれが属する領域しか駆動できない点です。したがって、Virtex-6 FPGA デザインの BUFR が 1 つのクロック領域にのみ使用されていた場合は、変更は不要です。リージョナル クロックによって複数の領域を駆動できますが、7 シリーズ デバイスでは、BUFMR が垂直方向に隣接する最大 3 つのクロック領域に含まれる BUFR を駆動できます。ターゲット アーキテクチャを適切に反映するために BUFR プリミティブの SIM_DEVICE 属性を「7SERIES」に変更してください。ただし、この属性を「VIRTEX6」に設定しても同じ機能が適用されます。この属性が Virtex-6 FPGA より以前のアーキテクチャに設定されているする場合は、その値を「7SERIES」に更新することを強く推奨します。これによってデバイスの回路がシミュレーションでの動作に正確に反映されます。

CAPTURE_VIRTEX6

7 シリーズ デバイスのキャプチャ コンポーネントは、Virtex-6 FPGA のコンポーネントと同じインターフェイスおよび機能を備えています。コンポーネント名は Virtex-6 FPGA を指す部分を削除して CAPTUREE2 に変更されましたが、使用方法は以前のコンポーネントと同様です。デザイン内の CAPTURE_VIRTEX6 コンポーネントは、自動的に CAPTUREE2 コンポーネントにリターゲットされますが、新しいコンポーネント名を使用してコードを更新することを推奨します。

FIFO18E1/FIFO36E1

7 シリーズ デバイスのハード FIFO コンポーネントは、Virtex-6 の FIFO と同じインターフェイスおよび機能を備えています。注意すべき唯一の変更点は、FIFO の動作前に新たなリセット要件が加えられたことです。回路が確実に初期化されるように、電源投入時にリセット状態を保持するクロック サイクル数が Virtex-6 FPGA の 3 サイクルから、7 シリーズ FPGA では 5 サイクルに増やされました。リセットのアサート時間が 5 クロック サイクルよりも短く設計された回路では、上記の要件

を満たすようデザインを変更する必要があります。これらのコンポーネントには、デフォルトで Virtex-6 FPGA の初期化動作を観察する SIM_DEVICE と呼ばれる属性があります。7 シリーズ デバイスに対して適切に動作するように、この値を「7SERIES」に変更してください。変更しない場合は、初期化に関して回路が適切に動作することを確認するためにタイミング シミュレーションを実行することを強く推奨します。

FRAME_ECC_VIRTEX6

7 シリーズ デバイスのコンフィギュレーション フレーム エラー検出/訂正回路コンポーネントは、Virtex-6 FPGA のコンポーネントと同じインターフェイスおよび機能を備えています。コンポーネント名は Virtex-6 FPGA を指す部分を削除して FRAME_ECCE2 に変更されましたが、使用方法は以前のコンポーネントと同様です。デザイン内の FRAME_ECC_VIRTEX6 コンポーネントは、自動的に FRAME_ECCE2 コンポーネントにリターゲットされますが、時間が許せば、新しいコンポーネント向けにコードを更新することを推奨します。

GTHE1_QUAD、GTXE1、IBUFDS_GTHE1、IBUFDS_GTXE1

GTHE1_QUAD と GTXE1 の両プリミティブおよびこれらに関連するバッファーや一般的な回路は 7 シリーズ デバイスではサポートされていません。7 シリーズ デバイスではシリアル トランシーバー回路の基本的な機能が変更されたため、リターゲットされません。これらの機能は、GTXE2 および GTHE2 コンポーネントで置き換えられました。デザインにこれらのコンポーネントが含まれている場合は、該当する 7 シリーズ デバイスをターゲットとして CORE Generator ウィザードで生成した、同等のインターフェイスで置き換えることを推奨します。

ICAP_VIRTEX6

7 シリーズ デバイスの内部コンフィギュレーション アクセス ポートは、Virtex-6 FPGA のコンポーネントと同じインターフェイスおよび機能を備えています。ただし、コンフィギュレーション ロジックやレジスタで読み出し/書き込まれるデータには注意すべき違いがあります。コンポーネント名は Virtex-6 FPGA を指す部分を削除して ICAPE2 に変更され、ICAP_VIRTEX6 コンポーネントにあった BUSY ピンは 7 シリーズの ICAPE2 コンポーネントからは削除されました。ICAPE2 コンポーネントは確定的なタイミングを使用するため BUSY ポートをポーリングするのではなく、このタイミングに従って ICAP インターフェイスを変更することを推奨します。デザインに ICAP_VIRTEX6 プリミティブが含まれている場合は、デザイン内の接続およびシミュレーション動作を適切なものにするために ICAPE2 に置き換えてください。ICAPE2 コンポーネントの使用法の詳細は、『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) を参照してください。

IODELAYE1

Virtex-6 FPGA の IODELAYE1 コンポーネントは、特定のコンフィギュレーションでは自動的にリターゲットされます。7 シリーズ デバイスには IODELAYE1 コンポーネントはありません。同等の機能は 2 つの独立したファンクション IDELAYE2 と ODELAYE2 に分割されました。デザインに出力専用の遅延として構成された IODELAYE1 コンポーネント (DELAY_SRC = 'O') がある場合は、ODELAYE1 コンポーネントに自動的にリターゲットされます。入力専用の遅延の場合は (DELAY_SRC = 'I')、IDELAY_TYPE 属性が「DEFAULT」に設定されていなければリターゲット可能です。双方向の遅延として使用されている IODELAY コンポーネントは自動的にはリターゲットされません。同等の機能が必要な場合は、IDELAYE2 を入力パスに、ODELAYE2 を出力パスにインスタンシエートします。

ISERDESE1

Virtex-6 FPGA の ISERDESE1 コンポーネントは、7 シリーズ デバイスでは ISERDESE2 コンポーネントに置き換えられました。ISERDESE2 コンポーネントには ISERDESE1 の機能のほとんどが含まれています。主な変更点は 1 つの ISERDESE2 コンポーネントで最大 8 ビットのデータ幅をサポートし、2 つの ISERDESE2 コンポーネントをカスケード接続することで最大 14 ビットのデータ幅をサポートできるように、Q7 ピンと Q8 ピンが追加されたことです。ほとんどの場合、デザイン内の ISERDESE1 コンポーネントはリターゲットされ、変更は不要です。時間が許せば、ISERDESE2 プリミティブに置き換えることを推奨しますが、7 シリーズ デバイスがターゲットの場合はこの作業は不要です。

JTAG_SIM_VIRTEX6、SIM_CONFIG_V6、SIM_CONFIG_V6_SERIAL

JTAG_SIM_VIRTEX6 および SIM_CONFIG_V6 コンポーネントは、7 シリーズ デバイスではそれぞれ JTAG_SIME2 と SIM_CONFIGE2 に置き換えられました。これらの機能は変更されているため、適切な動作が得られるようにテストベンチまたはシミュレーション ファイルで該当する新しいコンポーネントに置き換えることを推奨します。SIM_CONFIG_V6_SERIAL コンポーネントに相当する等価のコンポーネントは 7 シリーズ デバイスには用意されていません。

MMCM_BASE および MMCM_ADV

7 シリーズでは MMCM コンポーネント名が MMCME2_BASE と MMCME2_ADV に変更されました。機能の点では、7 シリーズの MMCM は Virtex-6 FPGA の MMCM のスーパーセットです。したがってほとんどの場合、MMCM は自動的にリターゲットされます。直接リターゲットできない可能性があるのは、DRP ポートを使用して動作パラメーターを変更している場合や 2 つの MMCM を直接カスケード接続している場合のみです。DRP ポートを使用している場合は、インターフェイスに変更はないものの、アドレス マッピングとレジスタ データが変更されているため、同じ機能を実行するには異なるデータをロードするようにコードを変更する必要があります。一方、直接カスケードされた MMCM がリターゲットされないのは、Virtex-6 FPGA が持つ専用の接続方式が 7 シリーズ FPGA にはないためです。

OSERDESE1

ほとんどの場合、Virtex-6 FPGA の OSERDESE1 コンポーネントは、7 シリーズ デバイスの OSERDESE2 に自動的にリターゲットされます。DDR メモリ インターフェイス回路が新しい Phaser 回路への移行によって削除されたことで、以前の回路に存在した関連するピンや属性がなくなりました。メモリ インターフェイスでこのブロックを使用している場合は、新しい OSERDES および関連する回路を使用して MIG IP ツールでコードを再生成して回路を実装してください。DDR メモリ インターフェイスのピンおよびオプションなしで OSERDES を使用する場合は、自動的にリターゲットされます。

PCIE_2_0

Virtex-6 FPGA の PCI Express® 用統合ブロックは、7 シリーズ デバイスでは PCIE_2_1 ブロックに置き換えられました。この新しいブロックにはいくつかの新機能がありますが、直接リターゲットはできません。この IP を含むすべての Virtex-6 FPGA デザインは、CORE Generator ツールで該当する 7 シリーズ デバイスをターゲットとして再生成することを推奨します。PCIE_2_1 コンポーネントの詳細は、『7 シリーズ FPGA PCI Express 用インテグレイテッド ブロック ユーザー ガイド』(UG477) を参照してください。

STARTUP_VIRTEX6

7 シリーズ デバイスのスタートアップ コンポーネントは、Virtex-6 FPGA のコンポーネントと同じインターフェイスおよび機能を備えています。コンポーネント名は Virtex-6 FPGA を指す部分を削除して STARTUPE2 に変更されましたが、ほとんどの場合、使用方法は以前のコンポーネントと同様です。STARTUP_VIRTEX6 コンポーネントの TCKSPI と DINSPI ポートは、7 シリーズの STARTUPE2 コンポーネントではなくなりました。これらのピンは専用コンフィギュレーション ピンから多目的ピンに変更され、STARTUP コンポーネントを使用せずに直接 HDL コードに接続できるようになりました。コンフィギュレーション SPI PROM にコンフィギュレーション後にインターフェイスする場合は、『7 シリーズ FPGA コンフィギュレーション ユーザー ガイド』(UG470) を参照してください。デザイン内の STARTUP_VIRTEX6 コンポーネントは、自動的に STARTUPE2 コンポーネントにリターゲットされますが、時間が許せば、新しいコンポーネント向けにコードを更新することを推奨します。

SYSMON

Virtex-6 FPGA の SYSMON コンポーネントは、7 シリーズ デバイスに直接リターゲットできます。ネイティブ コンポーネント XADC に再マッピングされます。XADC コンポーネントは精度と機能が改善されています。新しい XADC 機能を導入するためにコードを変更することが有利であるかどうか、再評価することを推奨します。

TEMAC_SINGLE

Virtex-6 FPGA の TEMAC_SINGLE コンポーネントは、7 シリーズ デバイスではサポートされていません。この機能が必要な場合は、適切なソフト IP から生成してください。

USR_ACCESS_VIRTEX6

7 シリーズ デバイスのユーザー アクセス コンポーネントは、Virtex-6 FPGA のコンポーネントと同じインターフェイスおよび機能を備えています。コンポーネント名は Virtex-6 FPGA を指す部分を削除して USR_ACCESSE2 に変更されましたが、使用方法は以前のコンポーネントと同様です。デザイン内の USR_ACCESS_VIRTEX6 コンポーネントは、自動的に USR_ACCESSE2 コンポーネントにリターゲットされますが、時間が許せば、新しいコンポーネント向けにコードを更新することを推奨します。

Unimacro の使用法

7 シリーズ デバイスは、Virtex-6 FPGA デザインのすべての Unimacro に対応しています。DEVICE 属性を「7SERIES」に変更することを推奨しますが、必須ではありません。

I/O に関する注意事項

入力ホールド タイム

これまでは入力レジスタの駆動に MMCM を使用せずにグローバル クロックを使用している場合、正のホールド タイム要件が発生しないように、遅延が自動的に挿入されていました。7 シリーズ デバイスではこの機能がサポートされていないため、このような状況では注意が必要です。ホールド タイムの要件が必要にならないように、次の手順に従ってください。

1. クロック挿入遅延をなくすために MMCME2 または PLLE2 を使用します。
2. データを適切にキャプチャするようにデータ パスに IDELAYE2 を挿入して十分なデータ遅延を確保します。
3. 入力レジスタは ILOGIC ではなくスライス アレイに配置します。
4. 外部タイミングが正のホールド タイムを許容していることを確認します。

適切なタイミング解析では、I/O タイミングに関連する問題点はすべて設計者に警告されます。セットアップ タイムとホールド タイムの両方が確実に満たされるように、低速のものも含めてすべての I/O パスにタイミング制約を適用することを推奨します。

バンク タイプ

Virtex-6 デバイスのバンク構造は均質ですが、Virtex-7 および Kintex-7 デバイスには、異なる I/O 規格と回路をサポートする 2 種類のバンクがあります。7 シリーズ デバイスの 2 つのバンクは、HP (High Performance) バンクおよび HR (High Range) バンクと呼ばれます。HP バンクと HR バンクの最も顕著な違いは、サポートする I/O 規格です。HR バンクは V_{CCO} が最大 3.3V までのほとんどの I/O 規格をサポートするのに対し、HP バンクは V_{CCO} が 1.8V 以下の I/O 規格をサポートします。Virtex-6 FPGA ではすべてのバンクが最大 2.5V までの I/O 規格をサポートしているため、デザイン内の 2.5V 規格の I/O は HR バンクに割り当てる必要があります。また、高速、低電圧 (1.8V 以下) の I/O 規格はいずれも HP バンクに割り当て、ジッターその他の性能が最適化されるようにすることを推奨します。より大きな Virtex-7 デバイスには HP バンクしかありません。したがって、1.8V を超える I/O 規格を使用するデザインでは、その電圧に対応するためにレベル シフターなどの外部回路が必要になる可能性があります。

HP バンクと HR バンクのもう 1 つの違いは、ODELAYE2 コンポーネントが HP バンクにしかないことです。したがって、Virtex-6 FPGA において IODELAYE1 コンポーネントの出力遅延機能を使用していたか、あるいは出力パスに同様の遅延制御が必要な場合は、それらの出力を HP バンクに割り当てる必要があります。