XAPP1230 (v1.0) 2015 年 6 月 9 日

UltraScale FPGA でのコンフィギュレーション リードバック キャプチャ

著者 : Stephanie Tapp

はじめに

このアプリケーション ノートでは、Vivado® Design Suite および UltraScale FPGA JTAG インターフェイスを使用して UltraScale™ FPGA 内部のコンフィギャラブル ロジック ブロック (CLB) レジスタのユーザー ステートをリードバック キャプチャするプロセスについて説明します。図 1 には、リードバック キャプチャ データを ASCII ファイル (.rdbk) に保存し、このファイルにあるデザイン エレメントをデザイン ロジック ロケーション ファイル (.ll) を用いて特定する手順を示しています。『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] に記載されている FPGA コンフィギュレーションに関する情報と、Vivado Design Suite の一般的な使用方法が基本的な知識として求められます。

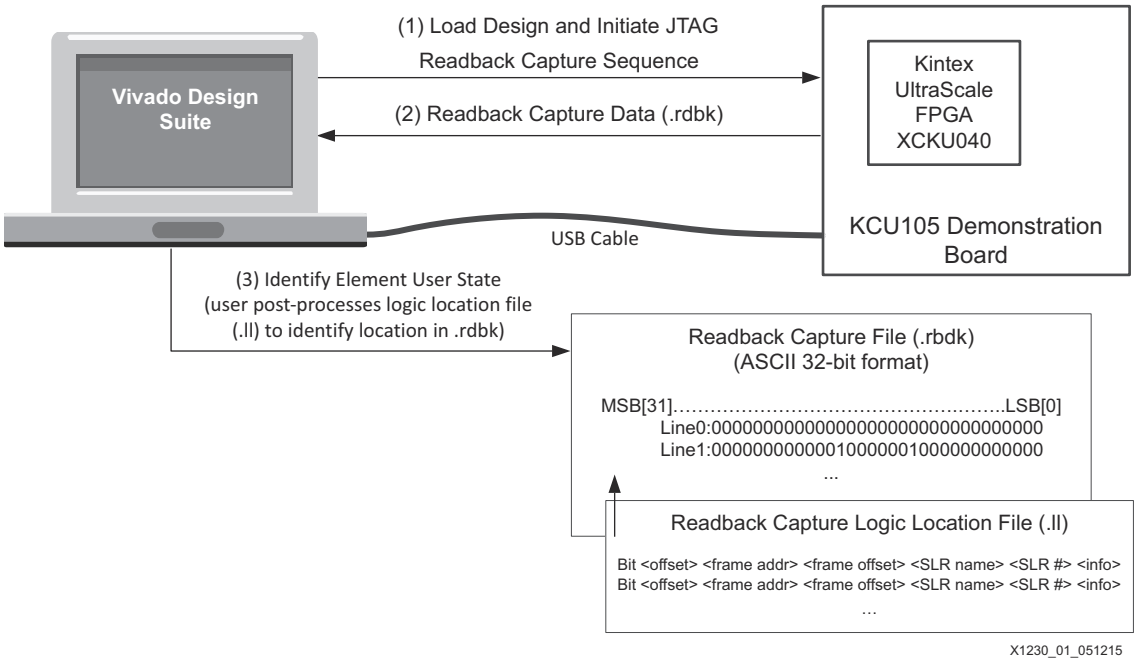

図 1 : JTAG インターフェイスを介する UltraScale FPGA リードバック キャプチャ フロー

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。デザイン ファイルの詳細は、「リファレンス デザイン」を参照してください。

概要

UltraScale FPGA の電源投入後のコンフィギュレーションでは、入力/出力ブロック (IOB)、CLB、分散 RAM、シフト レジスタ ロジック (SRL)、ブロック RAM の初期内容、そしてクロック リソース、ロジック リソース、および I/O 間のインターコネクト配線など、ユーザー デザインのプログラマブルな要素が定義されます。コンフィギュレーション ロジックの命令とコンフィギュレーション情報の組み合わせが、UltraScale FPGA のコンフィギュレーション用のビットストリームを構成します。ユーザー ビットストリームで FPGA がコンフィギュレーションされると、内部のコンフィギュレーション メモリのデータがリードバックと呼ばれるプロセスによって読み出し可能になります。

リードバックを実行するには、コマンド シーケンスをデバイスに送信します。リードバックが開始されると、UltraScale FPGA はコンフィギュレーション メモリの内容をサポートされているインターフェイスに送信します。UltraScale FPGA では 3 つのインターフェイス (コンフィギュレーション アクセス ポート (ICAP)、SelectMAP、JTAG インターフェイス) の 1 つからリードバックを実行可能です。

UltraScale FPGA はリードバック検証とリードバック キャプチャをサポートしています。リードバック検証は、コンフィギュレーション ロジックがプログラムされ、ユーザーの意図に沿っていることを確認するための比較機能です。リードバック キャプチャは、動作中のデザイン変更でユーザー ステート エレメントの内容を判断するために使用されます。

リードバック検証は、生成されたデザインのマスク ファイル (`.msk`/`.msd`) を使用してユーザー デザインのビットストリームとリードバック データを比較します。マスク ファイルは、ユーザー デザインで動的に変更される値を持つコンポーネントを判断し、比較中はそれらを無視します。このようなコンポーネントの例として、CLB や分散 RAM としてコンフィギュレーションされたルックアップ テーブル (LUT) が挙げられます。ユーザー デザインを FPGA に正しくプログラムする方法として、Vivado Design Suite のハードウェア マネージャーでは JTAG インターフェイスを介する検証動作がサポートされています。

このアプリケーション ノートは、リードバック キャプチャに焦点を当てて説明します。リードバック キャプチャはリードバック検証と同じプロセスを使用しますが、追加コマンドをリードバック シーケンス中に発行して、内部 CLB レジスタのユーザー ステートを読み出す必要があります。リードバック キャプチャの特定方法では、CLB レジスタ、ブロック RAM、分散 RAM として使用される LUT や SRL などのデザイン固有のリードバック キャプチャの結果がユーザー デザインの `.ll` ファイルによってマップされます。リードバック キャプチャでサポートされるデザイン エレメントには、固有のフレーム タイプ (タイプ 0、タイプ 1) があります (図 2 参照)。Vivado Design Suite はリードバック キャプチャを直接サポートしませんが、このアプリケーション ノートには Vivado Design Suite Tcl コマンドで実行可能な参照用スクリプトが付属しています。これを使用すると、ユーザー ステート データをリードバック キャプチャして読み出し可能な ASCII ファイル (`.rdbk`) にフォーマット変換できます。詳細は、「インプリメンテーション」を参照してください。

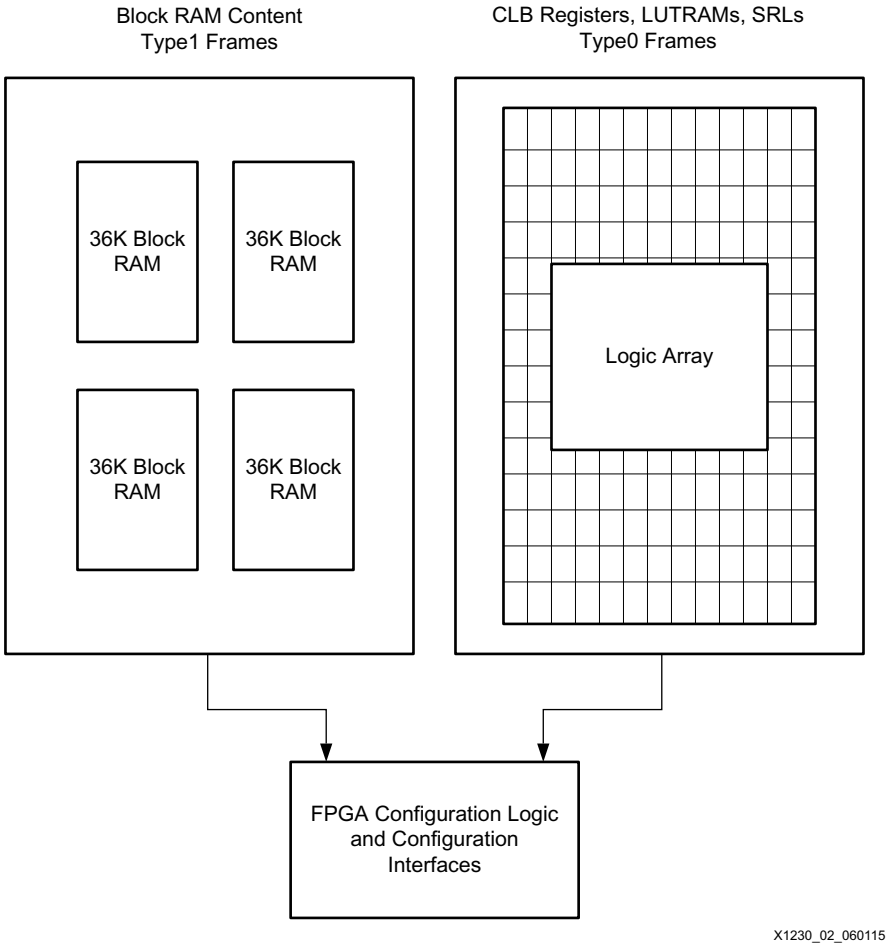

図 2 : リードバック キャプチャでサポートされる UltraScale FPGA デザイン エレメント

新しいデザインのデバッグには、Vivado のロジック解析が多くのアプリケーションにとって理想的なもう 1 つの強力なツールとなります。追加されたデバッグ IP ロジックがタイミング/リソースの問題とならない場合や、幅に制限のあるトレース (幅を増やすためにブロック RAM リソースが必要となる) が許容可能な場合、このツールの使用を検討します。数多くの信号を解析する必要があり、クロック制御が可能なハードウェア ASIC エミュレーションまたは協調シミュレーションでは、リードバック キャプチャ機能が優れたオプションとなります。この機能は、ユーザー デザインにロジックを加える必要がなく、デザイン ステートを確認するためのアクセスを容易にします。

7 シリーズ FPGA との違い

UltraScale FPGA には専用のキャプチャ メモリ セルがないため、7 シリーズ FPGA で利用可能な CAPTUREE2 プリミティブおよび GCAPTURE コマンドは必要なくなり、UltraScale FPGA でサポートされなくなりました。

7 シリーズ FPGA のリードバック キャプチャ シーケンスでは、ユーザー ステートの内容を読み出す前に GCAPTURE コマンドが必要でした。UltraScale FPGA では、マスク (MSK) レジスタおよび制御 1 (CTL1) レジスタ (CAPTURE bit[23]) へ値を書き込み、CLB レジスタのリードバック キャプチャを有効にする必要があります。UltraScale FPGA における JTAG リードバック キャプチャ コマンドのシーケンスを表 5 で説明します。

7 シリーズ FPGA でリードバック キャプチャを実行する場合、デバイスの現在のステートが確実にリードバック キャプチャされるように、GCAPTURE コマンドがロードされている間は関連するロジックのクロックを停止することが推奨されています。UltraScale FPGA では、リードバック キャプチャ シーケンス全体でターゲットとなっているユーザー ステート エレメントに関連するクロックを停止または無効にする必要があります。

表 1 に、7 シリーズと UltraScale FPGA 間におけるリードバック キャプチャの値の違いを示します。

表 1 : リードバック キャプチャの反転のまとめ

デザイン コンポーネント	UltraScale FPGA における リードバック キャプチャ データの反転	7 シリーズ FPGA における リードバック キャプチャ データの反転
CLB レジスタ	反転あり。 1 のとき 0 としてキャプチャされる値	反転あり。 1 のとき 0 としてキャプチャされる値
ブロック RAM レジスタ	反転なし。 0 のとき 0 としてキャプチャされる値	反転あり。 1 のとき 0 としてキャプチャされる値
分散 RAM または SRL	反転なし。 0 のとき 0 としてキャプチャされる値	反転なし。 0 のとき 0 としてキャプチャされる値

リードバック キャプチャ フローの概要

図 3 に、リードバック キャプチャ フローの概要図を示します。リードバック キャプチャが開始され、データがリードバックされると、目的のデザイン エレメントを割り出すためのビットマップが .ll ファイルで提供されます。ファイルの生成および特定方法の詳細は、「インプリメンテーション」で説明しています。

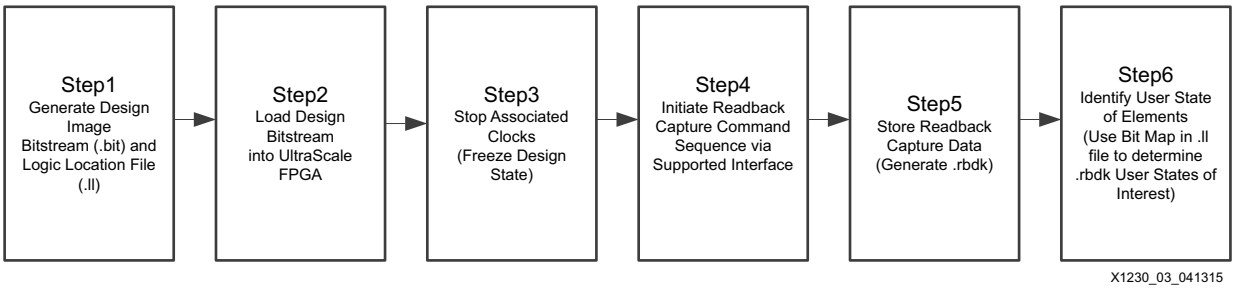

図 3 : リードバック キャプチャ フローの概要

インプリメンテーション

以降のセクションでは、開発ボードのセットアップ手順、ソフトウェアでのファイル生成手順、JTAG インターフェイスを介して UltraScale FPGA CLB レジスタのユーザー ステートを読み出すリードバック キャプチャ コマンド シーケンス、および KCU105 開発ボードでフローを実行するための Tcl 命令について解説しています。

デモのセットアップ

リードバック キャプチャ フローの例では次を使用してデモを実行します。

- Kintex UltraScale FPGA XCKU040 を搭載したザイリンクス KCU105 開発ボードと USB ケーブル
- Vivado Design Suite 2015.1
- アプリケーション参照用 Tcl スクリプト (`readback_capture.tcl`)

このアプリケーション ノートに含まれるリファレンス デザインでリードバック キャプチャを実行する前に、図 4 に示すようにボードをセットアップする必要があります。

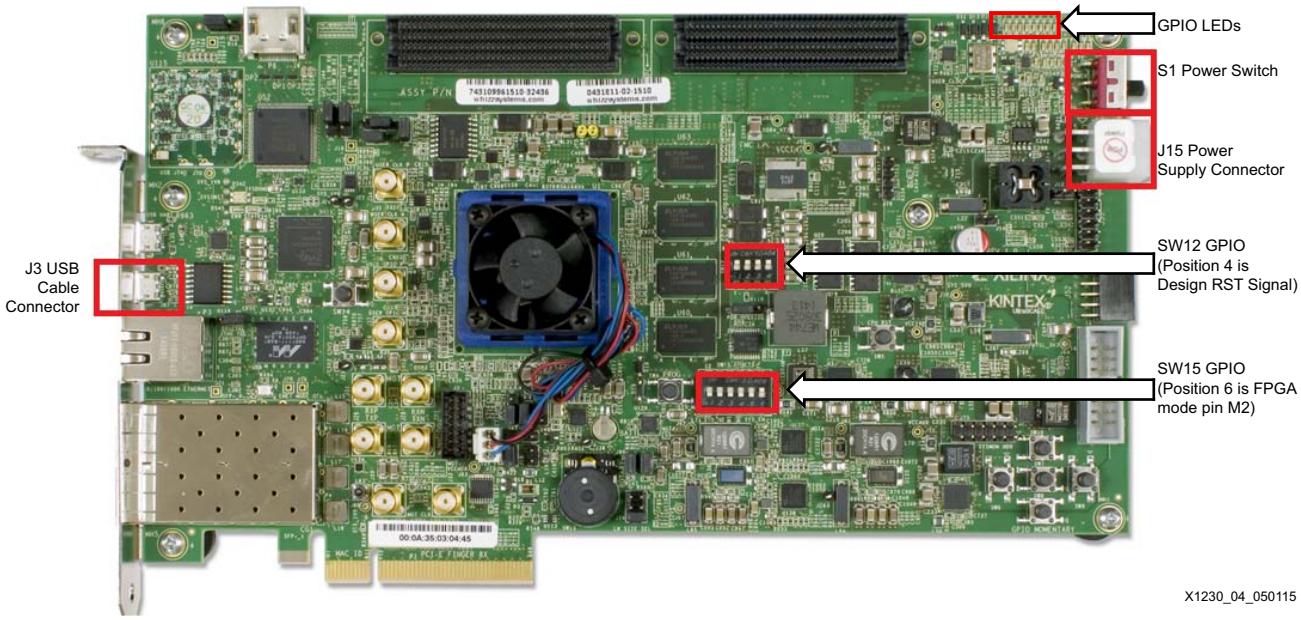

図 4 : KCU105 開発ボードのセットアップ

1. ザイリンクス プラットフォーム ケーブル USB II の一端をコンピューターの USB ポートに、もう一方の端をボードの USB ケーブル コネクター J3 に接続します。
2. 汎用 I/O (GPIO) スイッチ SW12 を `1111` に設定します (図 4 参照)。

   スイッチ SW12 のポジション 4 は、リファレンス デザインの RST 信号で、リファレンス デザインのカウントをリセットするために使用できます。これはデフォルトでオフとなっており、図 4 に示すように設定する必要があります。

3. スイッチ SW15 を `000001` に設定します (図 4 参照)。

   FPGA モード ピン M2 はポジション 6 です。この設定では M[2:0] = `101` で JTAG モードが使用されます。

4. 電源を電源コネクタ J15 に接続します。
5. 電源スイッチ S1 をオンのポジションに設定します。

   注記 : GPIO LED は、リファレンス デザイン CLB レジスタの値を表示します。

リファレンス デザインの説明

このアプリケーション ノートには、ロジック ロケーション ファイルを使用してデザインの 8 つの CLB レジスタのユーザー ステートを特定する方法を示す、シンプルな 8 ビット カウンターのリファレンス デザインが含まれています。このリファレンス デザインでは、8 ビット カウンターが CLB レジスタでインプリメントされており、その値は図 4 の 8 つの GPIO LED で表示されます。

リファレンス デザインの 8 ビット カウンターのクロックは、次の条件が揃ったときに有効になります。

- UltraScale FPGA JTAG USER4 命令がアクティブ
- TMS が Low で、UltraScale FPGA JTAG TAP が RUNTEST 状態にある

このデザインでは、USB ケーブルを介して Vivado Design Suite で 2 つのコマンド (「カウントを有効にし、クロックを停止する」に記載の scan_ir_hw_jtag および runtest_hw_jtag) を送信することで、ユーザーがカウント値を指定できます。ユーザー側で scan_ir_hw_jtag コマンドを最初に送信して USER4 JTAG 命令をアクティブにすると、runtest_hw_jtag コマンドが発行されてカウンターが開始されます。RUNTEST コマンドが発行する TCK の数は、CLB レジスタに格納されているカウント値と同じです。たとえば、LED[7:0] = `10101110` をキャプチャしたい場合、TCK カウントに 10 進数に相当する 174 を選択することで CLB レジスタ qout_reg[7:0] に反映されます。SW12 ポジション 4 が `0` に設定され、runtest_hw_jtag -tck 1 コマンドが発行されると、RST 信号はカウント値をリセットできます。

リードバック キャプチャの実行先となるデザイン エレメントに関連するクロックは停止する必要があります。カウンター リファレンス デザインでは、別の JTAG コマンドで JTAG USER4 命令が置き換えられる場合、または RUNTEST 条件が満たされなくなったとき、カウンターへの内部 TCK が無効になります。リファレンス デザインの内部 TCK が無効になると、リードバック キャプチャが実行可能です。図 5 に、リファレンス デザインを示します。

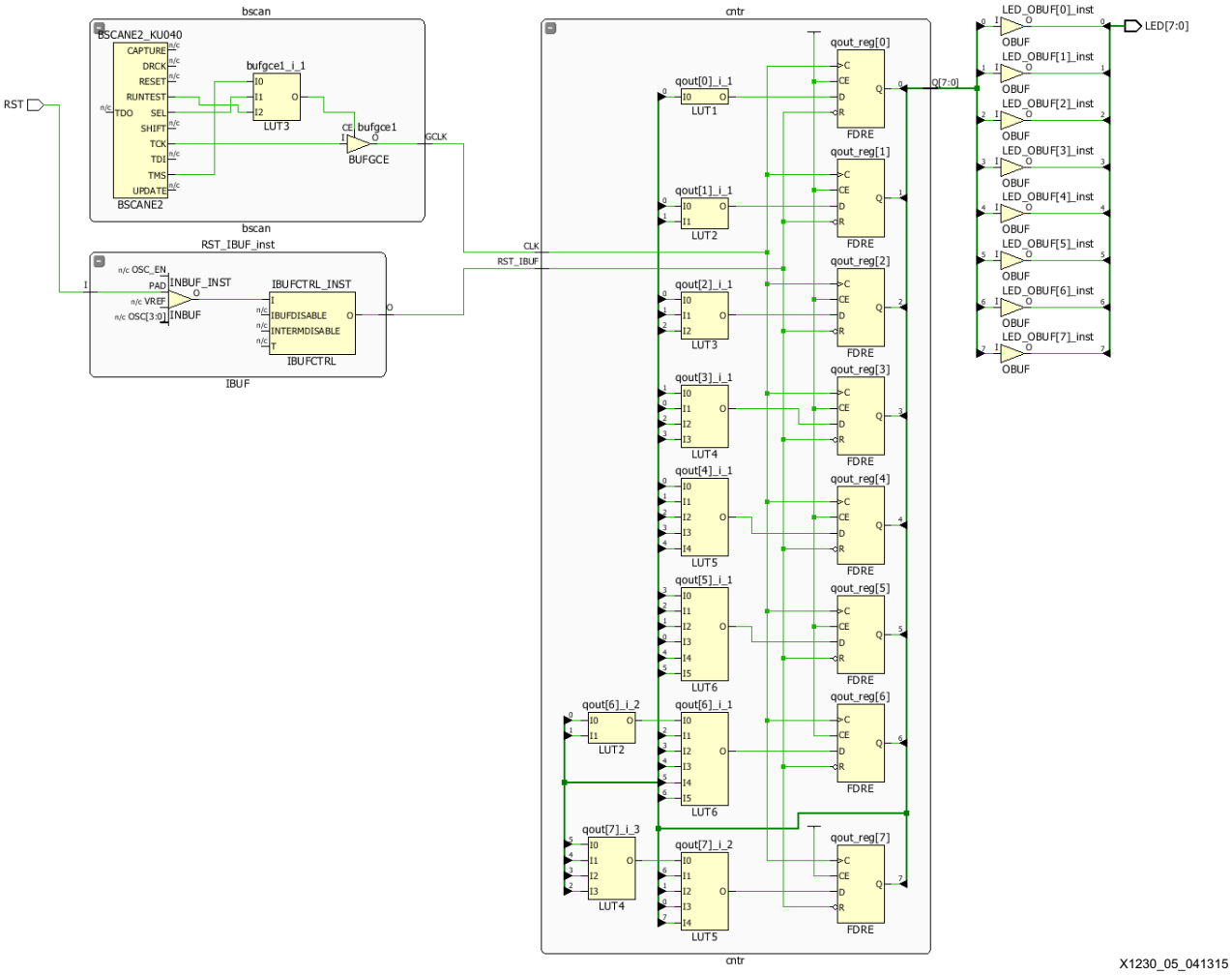

図 5 : 8 ビット カウンターのリファレンス デザイン

このアプリケーション ノートと共に提供されているリファレンス デザイン ファイルのディレクトリ構造は次のようになっており、デモ フローでも参照されます。

- デザイン
  - デザイン ソース ファイルはデモの実行に必要ありませんが、参照用として提供されています。
  - Verilog ファイル : `LED_Count.v`、`bscan.v`、および `cntr.v`.
  - 制約ファイル : `LED_Count.xdc`
- デモ ファイル
  - このアプリケーション ノートでは、デザイン ソース ファイルから作成されるビットストリームおよびロジック ロケーション ファイルを使用し、KCU105 評価ボードでリードバック キャプチャ フローのデモを実行します。
  - ビットストリーム ファイル : `LED_Count.bit` (または `LED_Count.rbt`).
  - ロジック ロケーション ファイル : `LED_Count.ll`
- Tcl スクリプト
  - ビット特定用に、KCU105 評価ボードでユーザー ステート データをリードバック キャプチャし、ASCII リードバック キャプチャ ファイル (`.rdbk`) を出力する Tcl スクリプトが提供されています。
  - スクリプト ファイル : `readback_capture.tcl`

Vivado Design Suite フロー

このセクションでは、Vivado Design Suite フローのプログラム段階で求められる、デザイン入力の考慮事項およびリードバック キャプチャ ビットストリームの設定について説明します。ビットストリームの設定の詳細は、『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908) [参照 2] を参照してください。

デザイン入力

リードバック キャプチャが実行されるデザインでは、ユーザー ステートを固定するために、ロジックに関連するクロックをユーザーが停止する必要があります。サンプルのリファレンス デザインで使用されるクロックは BSCANE2 TCK であり、これは USER4 がアクティブな JTAG 命令でない場合に無効になります。

ビットストリーム ファイルの生成

このセクションでは、考慮しなければならない write_bitstream 生成プロパティおよびファイル設定について説明します。表 2 に、リードバック キャプチャ用に、ビットストリームと共に生成する必要がある出力ファイル フォーマットの説明を示します。

表 2 : Write_bitstream ファイル出力

ファイル拡張子	ファイルの種類	ファイルの説明
.bit/.rbt	バイナリ/ASCII	ユーザー デザイン ビットストリーム
.ll	ASCII	ロジック ロケーション ファイルには、レジスタ、分散 RAM、SRL、およびブロック RAM のロケーションが含まれます。
.rbd	ASCII	(オプション) リードバック ASCII .rbt などのフォーマットで、パディング オーバーヘッドがあるコンフィギュレーション データ (133 ワード) です。コンフィギュレーション コマンドは含まれません。このファイルは、INIT 値を用いるサンプル ファイルとして使用できます。詳細は、「デバッグ」を参照してください。

表 3 に、必要なリードバック キャプチャ ファイルを生成するために求められる write_bitstream の設定を示します。

表 3 : Write_Bitstream ファイル生成のオプション

オプション	説明
-verbose	write_bitstream の実行時に使用されるオプションのログ ファイル サマリを提供します。
-raw_bitfile	ASCII フォーマットのユーザー デザイン ビットストリーム (.rbt) を作成します。
-readback_file	(オプション) ファイル リードバックと一致させるためにオーバーヘッド パディングが追加された ASCII フォーマットの .rbd ファイルを作成します。コマンドは含まれません。このファイルは一般にリードバック検証用に .msd ファイルと共に使用されますが、オプションでリードバック キャプチャ シーケンスのデバッグ時に使用できます。詳細は、「デバッグ」を参照してください。
-logic_location	ブロック RAM、分散 RAM、SRL、および CLB レジスタのロケーションをビット マップするための、ASCII フォーマットの .ll ファイルを生成します。

ビットストリームのプロパティ セットでリードバックの禁止を解除しておきます。つまり、レベル 1 (リードバック無効) またはレベル 2 (リードバックとリコンフィギュレーションを共に無効) に設定されたリードバック セキュリティ set_property BITSTREAM.READBACK.SECURITY、あるいは暗号化プロパティは使用できません。

次のコマンド ラインは、Vivado Design Suite Tcl コンソールでユーザー プロジェクトからビットストリーム生成ランを実行する場合に必要な設定の一例です。

```
write_bitstream -verbose -raw_bitfile -logic_location_file -readback_file LED_Count
```

別の方法として、Vivado 統合設計環境 (IDE) を使用してビットストリームを生成する場合は、図 6 に示すように設定を選択します。ビットストリームの生成時、デバッグ向けの readback_file 設定はオプションです。詳細は、「デバッグ」を参照してください。

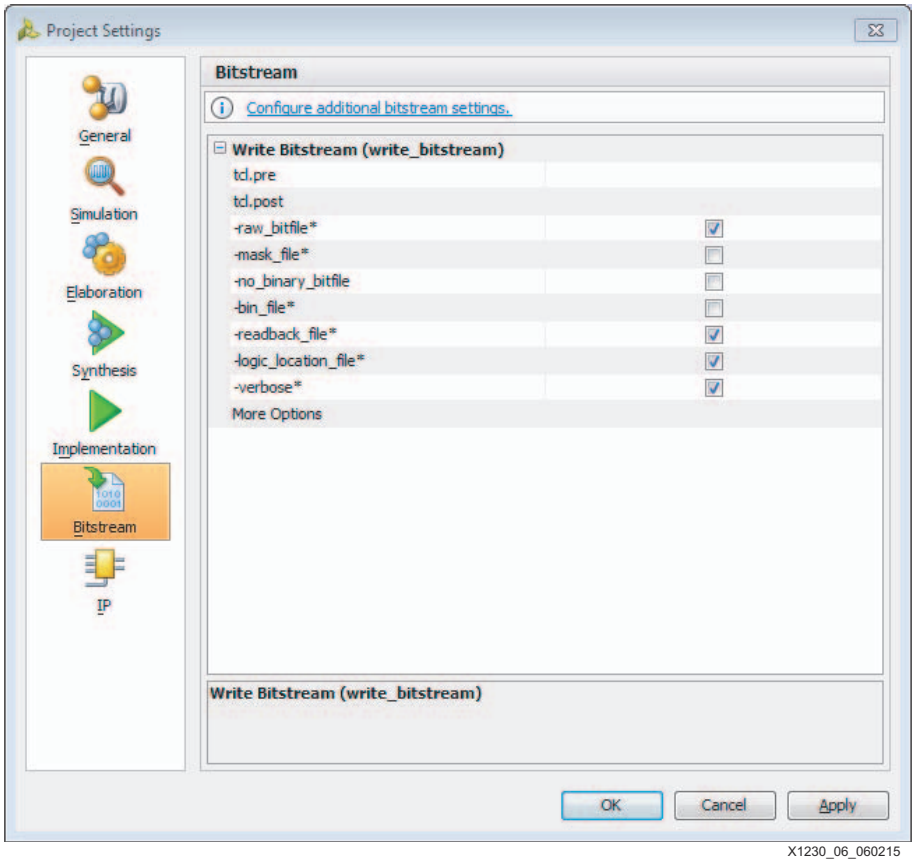

図 6 : Vivado Design Suite IDE のビットストリーム設定

ロジック ロケーション ファイル

.rdbk ファイルから適切なデザイン エレメントを特定するには、デザインの .ll ファイルからビット オフセットが必要です。このアプリケーション ノートに含まれる 8 ビット カウンター LED_Count.ll のサンプル ファイルを次に示します。CLB レジスタのビット ロケーションはこのファイルに示されています。

```
Revision 4
; Created by Vivado 2015.1 SW Build 1215546 at Fri May 08 13:54:31 2015
; Bit lines have the following form:
; <offset> <frame address> <frame offset> <SLR name> <SLR number> <information>
; <information> may be zero or more <key>=<value> pairs
; Block=<blockname>     specifies the block associated with this memory cell.
;
; Latch=<name>          specifies the latch associated with this memory cell.
;
; Net=<netname>         specifies the user net associated with this
;                       memory cell.
;
; Ram=<ram id>:<bit>    This is used in cases where a SLICE LUT is used
; Rom=<ram id>:<bit>    as RAM (or ROM) and for Block RAM.<Ram id> will
;                       be a letter indicating which LUT within the SLICE
;                       or a 'B' for Block RAM.For SLICE LUTs <bit> is a
;                       decimal number.For Block RAM this will be BIT or
;                       PARBIT indicating a data or parity bit and a decimal
;                       number.
;
; Info lines have the following form:
; Info <name>=<value>   specifies a bit associated with the FPGA
;                       configuration options, and the value of
;                       that bit.The names of these bits may have
;                       special meaning to software reading the .ll file.
;
Bit    30867264 0x00023204 1152 SLR0 0 Block=SLICE_X49Y78 Latch=AQ Net=cntr/Q[0]
Bit    30867280 0x00023204 1168 SLR0 0 Block=SLICE_X49Y78 Latch=AQ2 Net=cntr/Q[1]
Bit    30867268 0x00023204 1156 SLR0 0 Block=SLICE_X49Y78 Latch=BQ Net=cntr/Q[2]
Bit    30867284 0x00023204 1172 SLR0 0 Block=SLICE_X49Y78 Latch=BQ2 Net=cntr/Q[3]
Bit    30868128 0x00023204 2016 SLR0 0 Block=SLICE_X49Y90 Latch=AQ Net=cntr/Q[4]
Bit    30868144 0x00023204 2032 SLR0 0 Block=SLICE_X49Y90 Latch=AQ2 Net=cntr/Q[5]
Bit    30868132 0x00023204 2020 SLR0 0 Block=SLICE_X49Y90 Latch=BQ Net=cntr/Q[6]
Bit    30868148 0x00023204 2036 SLR0 0 Block=SLICE_X49Y90 Latch=BQ2 Net=cntr/Q[7]
```

.ll ファイルのヘッダーですべてのフィールドが説明されています。ファイルにはビット ポジションの列とロジック情報が含まれます。最初の列ではロジックがレジスタ、RAM、または ROM として使用されているかを示し、2 番目の列ではオフセットが提供されています。ビット オフセットのカウントは、リードバック データのビット ポジション フレーム アドレス 0 で開始します。フレーム オフセットは、指定されたフレーム アドレス内でビット オフセットを指定します。ネット名が適用できる場合、その名前がデザインの相関関係を示すために含まれます。このリファレンス デザインでは、ターゲット デザイン エレメントのユーザー ステートを特定するためにビット オフセット値を使用して正確なビット ロケーションが判断されます。

JTAG を介した FPGA のプログラム

リードバック キャプチャを実行する前に、ユーザー デザインのビットストリーム イメージを UltraScale FPGA にロードする必要があります。これには、JTAG またはサポートされているほかのコンフィギュレーション インターフェイスが使用可能です。このデモでは、JTAG を介して FPGA リファレンス デザインがロードされます。Vivado Design Edition または Vivado Lab Edition ツールで、ハードウェア マネージャーを開きます。Tcl コンソール ウィンドウで、次のコマンドを実行して FPGA をリファレンス デザインでプログラムし、ユーザー パスにはデザイン ファイルを指定します。

```
>connect_hw_server
>open_hw_target [lindex [get_hw_targets -of_objects [get_hw_servers localhost]] 0]
>current_hw_device [lindex [get_hw_devices] 0]
>refresh_hw_device -update_hw_probes false [lindex [get_hw_devices] 0]
```

```
>set_property PROGRAM.FILE {C:/XAPP1230/DemoFiles/LED_Count.bit} [lindex
[get_hw_devices] 0]
>program_hw_devices [lindex [get_hw_devices] 0]
>close_hw_target [lindex [get_hw_targets -of_objects [get_hw_servers localhost]] 0]
```

カウントを有効にし、クロックを停止する

FPGA がリファレンス デザインで問題なくコンフィギュレーションされたら、カウント値を選択し、関連するクロックを停止してキャプチャされるデザイン ステートを固定する必要があります。リファレンス デザインでは、Vivado Design Suite Tcl コンソールで JTAG 命令を用いて任意のカウント値を選択できます。コマンドの使用方法は、『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。

CLB レジスタの特定のカウント値を選択してそれを GPIO LED で表示させるには、Vivado ハードウェア マネージャーを開いて TCL コンソールに次の jtag_mode を入力します。

JTAG モードでターゲットを開くには次のように入力します。

```
>open_hw_target -jtag_mode 1
```

次に、Vivado Design Suite Tcl scan_ir_hw_jtag コマンドと共に USER4 命令 (オペコード 23) を発行し、RUNTEST コマンドを使用してカウント値を選択してカウンターを有効にします。使用する Vivado ハードウェア マネージャー JTAG コマンドを次に示します。

```
scan_ir_hw_jtag <ir length> -tdi <instruction>
runtest -tck <number of tcks>
```

USER4 オペコードを JTAG 命令レジスタにシフトします。

```
>scan_ir_hw_jtag 6 -tdi 23
```

RUNTEST コマンドの TCK カウントが LED に表示され、その値が CLB レジスタに格納されます。この例では、174 TCK の RUNTEST が選択され、16 進数に相当する LED[7:0] = 10101110 のカウント値が表示されます。

```
>runtest_hw_jtag -tck 174
```

USER4 JTAG 命令 (オペコード 23) を BYPASS (オペコード 3f) に置き換え、クロックが確実に停止するようにします。

```
>scan_ir_hw_jtag 6 -tdi 3f
```

CLB レジスタのリードバック キャプチャの JTAG シーケンス

JTAG インターフェイスを介してフレーム データ レジスタ アウト (FDRO) レジスタからデータをリードバック キャプチャする手順は、その他のレジスタから読み出す場合と基本的には同じです。表 4 に、リードバック キャプチャ実行中に使用するレジスタと各レジスタの 32 ビット コマンド リファレンスを示します。これらレジスタの詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] で説明されています。

表 4 : リードバック キャプチャに使用するコンフィギュレーション レジスタ コマンドの例

レジスタの説明	32 ビット コンフィギュレーション コマンド
FDRO 読み出しパケット ヘッダー (タイプ 1)	0x28006000
FDRO 読み出しパケット ヘッダー	0x4800CF00
動作なし (NOOP) 1 ワード	0x20000000
同期ワード	0xAA995566
MSK 書き込みパケット ヘッダー	0x3000C001
CTL1 書き込みパケット ヘッダー	0x30030001
CMD 書き込みパケット ヘッダー	0x30008001
フレーム アドレス レジスタ (FAR) 書き込みパケット ヘッダー	0x30002001

表 4 : リードバック キャプチャに使用するコンフィギュレーション レジスタ コマンドの例 (続き)

レジスタの説明	32 ビット コンフィギュレーション コマンド
CMD 書き込みパケット データ - リセット巡回冗長検査 (RCRC) コマンド	0x00000007
CMD 書き込みパケット データ - 読み出しコンフィギュレーション (RCFG) コマンド	0x00000004
MSK 書き込みパケット データ (CTL1 CAPTURE ビット)	0x00800000
CTL1 書き込みパケット データ (CAPTURE ビットは有効)	0x00800000
タイプ 2 FAR 書き込みパケット データ (NULL)	0x00000000

ほとんどのアプリケーションでは、読み出し可能なユーザー ステート データすべてをリードバック キャプチャすることが目標となるため、フレーム アドレス 0 からデバイス全体のリードバックが実行されます。表 5 に、JTAG インターフェイスを介した、フレーム アドレス 0 から開始する CLB レジスタの読み出し手順を示します。CLB レジスタに対してリードバック キャプチャを実行するための基本的な手順を次に示します。

1. ユーザー デザイン ステートに関連するクロックを停止し、ステートを固定します。

   注記 : キャプチャされるユーザー ステートに関連するクロックのみを停止させる必要があります。

2. キャプチャを有効にするために、コンフィギュレーション ロジック レジスタ MSK[23]/CTL1[23] に 1/1 を書き込みます。
3. CTL1 CAPTURE ビットが 1 に設定されている状態で FAR レジスタ (0x00000000) からフレームを読み出します。
4. リードバック キャプチャの実行後は、コンフィギュレーション ロジック レジスタ MSK[23]/CTL1[23] に 1/0 を書き込んで、デフォルト値にリセットします。
5. クロックを有効にし、関連するユーザー デザイン ステートの固定を解除します。

表 5 : XCKU040 で全フレームをリードバック キャプチャする JTAG シーケンスの例

手順	説明	手順の詳細	TDI	TMS	TCK の数
1	TAP コントローラーをリセットする	5 クロックの間、TMS に 1 を入力してデバイスを Test-Logic-Reset (TLR) ステートにする	X	1	5
		Run-Test/Idle (RTI) ステートに移行する	X	0	1
		Select-IR ステートに移行する	X	1	2
		Shift-IR ステートに移行する	X	0	2
2	CFG_IN をシフトインする	CFG_IN 命令の最初の 5 ビットを LSB から順にシフトする	00101 (CFG_IN)	0	5
		Shift-IR ステートから出る間に、CFG_IN 命令の MSB をシフトする	0	1	1
		Select-DR ステートに移行する	X	1	2
		Shift-DR ステートに移行する	X	0	2

表 5 : XCKU040 で全フレームをリードバック キャプチャする JTAG シーケンスの例 (続き)

手順	説明	手順の詳細	TDI	TMS	TCK の数
3	CFG_IN にパケットをシフトする	Shift-DR の間に、コンフィギュレーション パケットを MSB から順に CFG_IN データ レジスタにシフトする	パケット データ [31:0] : 0xFFFFFFFF (ダミー ワード) 0xAA995566 (同期ワード) 0x20000000 (NOOP) 0x30008001 (CMD 書き込み) 0x00000000 (NULL) 0x3000C001 (MSK 書き込み) 0x00800000 (CAPTURE ビット [23]) 0x30030001 (CTL1 書き込み) 0x00800000 (CAPTURE ビット [23]) 0x20000000 (NOOP) 0x20000000 (NOOP) 0x20000000 (NOOP) 0x20000000 (NOOP) 0x20000000 (NOOP) 0x20000000 (NOOP) 0x20000000 (NOOP) 0x20000000 (NOOP) 0x30002001 (FAR 書き込み) 0x00000000 (アドレス 0x0) 0x30008001 (CMD 書き込み) 0x00000004 (RCFG) 0x28006000 (FDRO 書き込み) 0x483D0E2B (読み出すワードの数を示すタイプ 2 パケット、つまり、XCKU040 32350 フレーム + 1 フレーム + 10 ワード) 0x20000000 (NOOP)	0	767
		Shift-DR ステートから出る間に、最後のコンフィギュレーション パケットの LSB をシフトする	0	1	1
		Select-IR ステートに移行する	X	1	3
		Shift-IR ステートに移行する	X	0	2
4	CFG_OUT をシフトインする	CFG_OUT 命令の最初の 5 ビットを LSB から順にシフトする	00100 (CFG_OUT)	0	5
		Shift-IR ステートから出る間に、CFG_OUT 命令の MSB をシフトする	X	1	2
		SELECT-DR ステートに移行する	X	1	2
		SHIFT-DR ステートに移行する	X	0	2

表 5 : XCKU040 で全フレームをリードバック キャプチャする JTAG シーケンスの例 (続き)

手順	説明	手順の詳細	TDI	TMS	TCK の数
5	FDRO からフレーム データをシフトする	FDRO レジスタの内容を CFG_OUT データ レジスタからシフトする	…	0	リードバックのビット数 -1
		Shift-DR ステートから出る間に、FDRO レジスタの最後のビットを CFG_OUT データ レジスタからシフトする	X	1	1
		Select-IR ステートに移行する	X	1	3
		Shift-IR ステートに移行する	X	0	2
6	TAP コントローラーをリセットする	テスト アクセス ポート (TAP) コントローラーを TLR ステートにして完了する	X	1	3
この段階で、リードバック キャプチャは問題なく完了しています。以降の手順によって、CTL1 レジスタが元のステートにリセットされます。					
7	RTI に移行する	RTI ステートに移行する	X	0	
		Select-IR ステートに移行する	X	1	2
		Shift-IR ステートに移行する	X	0	2
8	CFG_IN をシフトインする (キャプチャ ビットをアップデート)	CFG_IN 命令の最初の 5 ビットを LSB から順にシフトする	00101 (CFG_IN)	0	5
		Shift-IR ステートから出る間に、CFG_IN 命令の MSB をシフトする	0	1	1
		Select-DR ステートに移行する	X	1	2
		Shift-DR ステートに移行する	X	0	2
9	CFG_IN にパケットをシフトする	Shift-DR ステートの間に、コンフィギュレーション パケットを MSB から順に CFG_IN データ レジスタにシフトする	パケット データ [31:0] : 0xFFFFFFFF (ダミー ワード) 0xAA995566 (同期ワード) 0x20000000 (NOOP) 0x3000C001 (MSK 書き込み) 0x00800000 (CAPTURE ビット [23]) 0x30030001 (CTL1 書き込み) 0x00000000 (CAPTURE ビット [23]) 0x20000000 (NOOP) 0x20000000 (NOOP)	0	288
		Shift-DR ステートから出る間に、最後のコンフィギュレーション パケットの LSB をシフトする	0	1	1
		Select-IR ステートに移行する	X	1	3
		Shift-IR ステートに移行する	X	0	2
10	TAP コントローラーをリセットする	TAP コントローラーを TLR ステートにして完了する	X	1	3

リードバック キャプチャの実行時に考慮する必要があるパイプライン データがあります。FDRO レジスタからのコンフィギュレーション データは、図 9 に示すように 10 のダミー データ ワードに加えてフレーム バッファーを通るため、リードバック キャプチャ ファイルにあるワードをこの数の分だけパイプラインとして破棄する必要があります。次のセクションでは、.rdbk ファイルに対して readback_capture.tcl スクリプトを使用して FAR 0x00000000 から XCKU040 フレーム データをすべて読み出す方法を示し、その後 .ll ファイルの <offset> フィールドを用いて CLB レジスタのユーザー ステートを判断する方法についても説明します。

より短時間でのリードバック キャプチャが求められ、特定の CLB レジスタしか読み出す必要のないアプリケーションでは、リードバック キャプチャの性能最適化が 1 つのフレームになる可能性があります。1 つのフレームをリードバックする場合、ビット <offset> フィールドの代わりに .ll ファイルの <frame_address> フィールドと <frame_offset> フィールドが使用されます。

リードバック キャプチャ - Tcl スクリプトの使用法

このアプリケーション ノートには readback_capture.tcl スクリプトが参照用として含まれており、Vivado Design Suite Tcl コンソールから実行できます。ボードとケーブルは「デモのセットアップ」に記載のとおりに接続し、FPGA はリファレンス デザインでプログラムされ、リードバック キャプチャの実行前に関連するクロックは停止します。次に、Vivado Design Suite Tcl コンソールで readback_capture.tcl ファイルを source コマンドで実行する必要があります (図 7 参照)。この例では、c:/XAPP1230/TclScript/readback_capture.tcl というパスを使用しています。

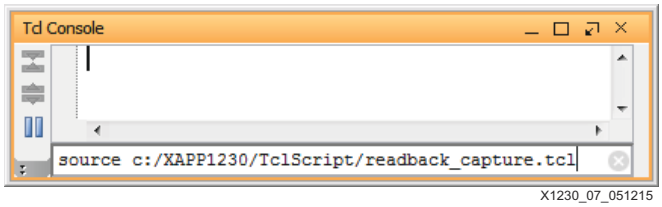

図 7 : readback_capture.tcl スクリプトを source コマンドで実行する

Tcl スクリプトには、JTAG を介してリードバック シーケンスを開始するプロセスや、.ll ファイルで想定された結果と照合するために、キャプチャされた内容を ASCII 32 ビット フォーマットに変換するプロセスが含まれます。リードバック シーケンスは、Vivado ハードウェア マネージャーの JTAG コマンドで開始されます。これらを含むコマンドの詳細は、『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908) [参照 2] を参照してください。

readback_capture.tcl ファイルが source コマンドで問題なく実行されたら、rdbk_jtag プロセスを Vivado Design Suite Tcl コンソールで実行する必要があります。rdbk_jtag プロセスのオプションを次に示します。

```
rdbk_jtag <path to save readback capture data .rdbk> <total frame count>
>rdbk_jtag c:/XAPP1230/LED_Count.rdbk 32530
```

表 6 に、デバイス全体の内容をリードバックするために使用されるフレーム総数が 32530 の場合の XCKU040 デバイス情報を示します。

表 6 : UltraScale コンフィギュレーション フレーム カウントの例

UltraScale FPGA	コンフィギュレーション フレームの総数	フレームあたりのワード数	オーバーヘッド ワード数 (パイプライン ワード カウント : 1 フレーム + 10 ワード)
XCKU040	32530	123	133

注記 :

1. その他のファミリ メンバーについては、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] を参照してください。

Tcl スクリプトは、3 つの主要セクションに分割されます。

- 定義 : コマンドの定義を提供します。
- リードバック キャプチャのプロセス : リードバック シーケンスや CAPTURE ビットをセットするためのコンフィギュレーション レジスタ コマンドの書き込みなど、setup_rdbk_cmd および rdbk_jtag プロセスが含まれます。
- データ フォーマットの変換プロセス : JTAG シリアル データを ASCII 32 ビット ワード フォーマット ファイルに変換します。呼び出されるプロセスには setHexRevTbl、revHexData、および conv2hex が含まれます。

リードバック キャプチャ データの解析

readback_capture.tcl スクリプトおよび記載されたプロセスが実行されると、ASCII 32 ビット ワード フォーマットの .rdbk ファイルが生成されます。.ll ファイルをビット マップとして使用して、目的のデザイン エレメントの .rdbk 値を判断することができます。リファレンス カウンター デザインには特定可能なカウンターが 8 つあります。このセクションでは、.ll ファイルがどのように使用されるかを説明します。

図 8 に、リファレンス デザインから CLB レジスタがリードバック キャプチャされる様子を図示します。

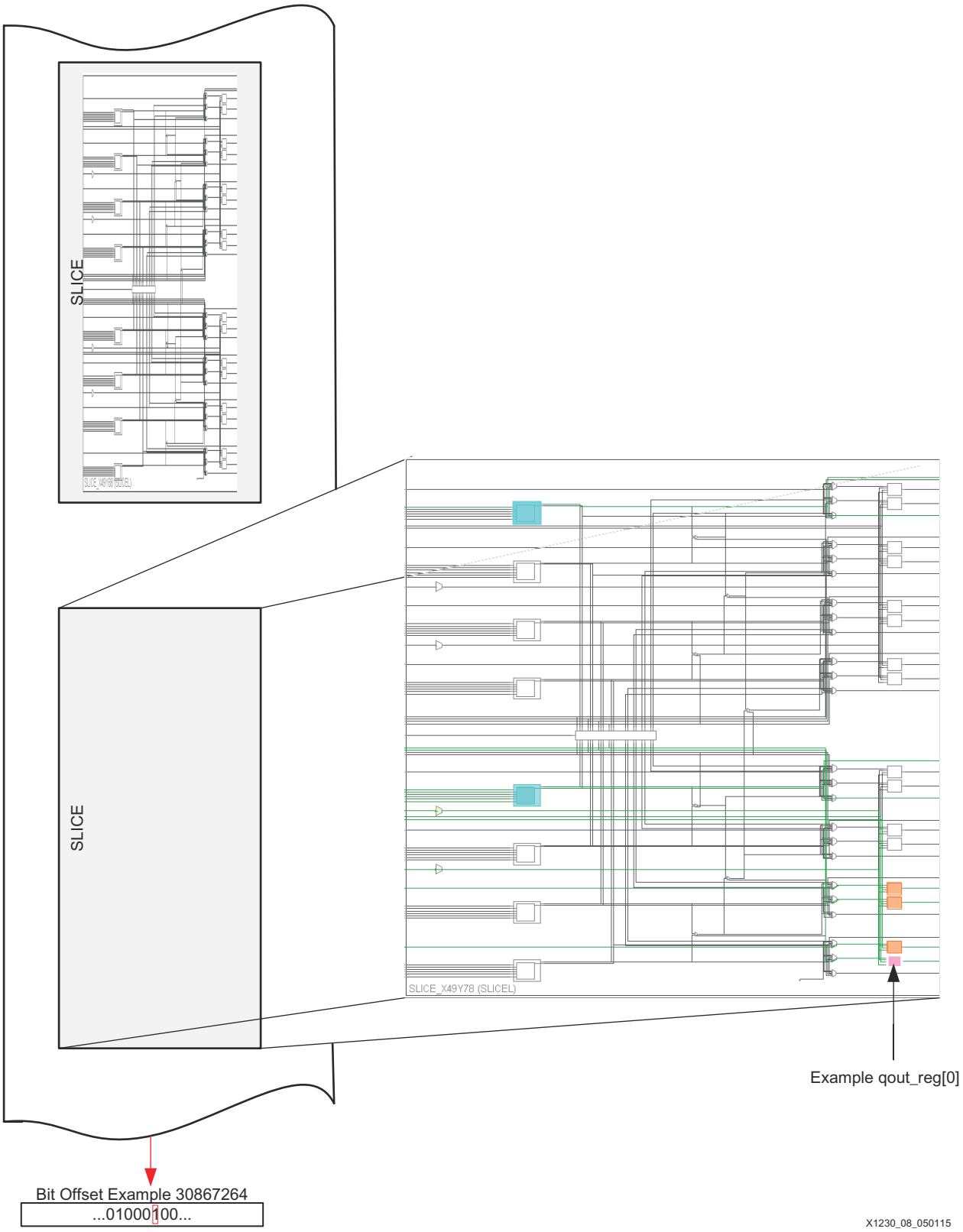

図 8 : リファレンス デザインから CLB レジスタがリードバック キャプチャされる様子

リファレンス サンプルの .ll ファイルの内容は、表 7 を参照してください。最初のレジスタ Q[0] では、FAR 0x00000000 からの 30867264 のビット オフセットを用いて .rdbk ファイルにおける CLB レジスタ ビット ロケーションが計算されます。このアプリケーション ノートでは、デバイス全体のリードバック キャプチャのデモ フローについて説明しているため、ビット オフセットが計算に必要な唯一の列となります。

表 7 : リファレンス デザインの cntr.ll からの CLB レジスタのまとめ

ビット オフセット	<frame_address>	<frame_offset>	<SLR name>	<SLR number>	備考
30867264	0x00023204	1152	SLR0	0	ブロック = SLICE_X49Y78 ラッチ = AQ ネット = cntr/Q[0]
30867280	0x00023204	1168	SLR0	0	ブロック = SLICE_X49Y78 ラッチ = AQ2 ネット = cntr/Q[1]
30867268	0x00023204	1156	SLR0	0	ブロック = SLICE_X49Y78 ラッチ = BQ ネット = cntr/Q[2]
30867284	0x00023204	1172	SLR0	0	ブロック = SLICE_X49Y78 ラッチ = BQ2 ネット = cntr/Q[3]
30868128	0x00023204	2016	SLR0	0	ブロック = SLICE_X49Y90 ラッチ = AQ ネット = cntr/Q[4]
30868144	0x00023204	2032	SLR0	0	ブロック = SLICE_X49Y90 ラッチ = AQ2 ネット = cntr/Q[5]
30868132	0x00023204	2020	SLR0	0	ブロック = SLICE_X49Y90 ラッチ = BQ ネット = cntr/Q[6]
30868148	0x00023204	2036	SLR0	0	ブロック = SLICE_X49Y90 ラッチ = BQ2 ネット = cntr/Q[7]

Tcl スクリプトは、リードバック キャプチャ データとパイプライン/パッド フレーム (133 ワード) を含む .rdbk ファイルを生成します。ビット オフセット値にはパイプライン オーバーヘッドが含まれないため、ロジック ロケーション ファイルを後処理する際にこれを反映させる必要があります。

CLB レジスタ値に対してワード ラインを求める式は、(int(bit offset/32) + パイプライン ワード カウント + エディター ワード) = ワード ラインです。エディターがワード ライン 1 で開始する場合のエディター ワードは 1 となり、エディターがワード ライン 0 で開始する場合のエディター ワードは 0 となります。次に、ワード ライン オフセットがビット オフセットのモジュロによって計算されます ((ビット オフセット mod 32) = ワード ライン オフセット)。たとえば、ワード ライン オフセット (30867264 mod 32) = 12 で、Q[0] CLB レジスタ ビットはワード ライン (int(30867264/32) + 133 + 1) = 964736 上にあります (図 9 参照)。

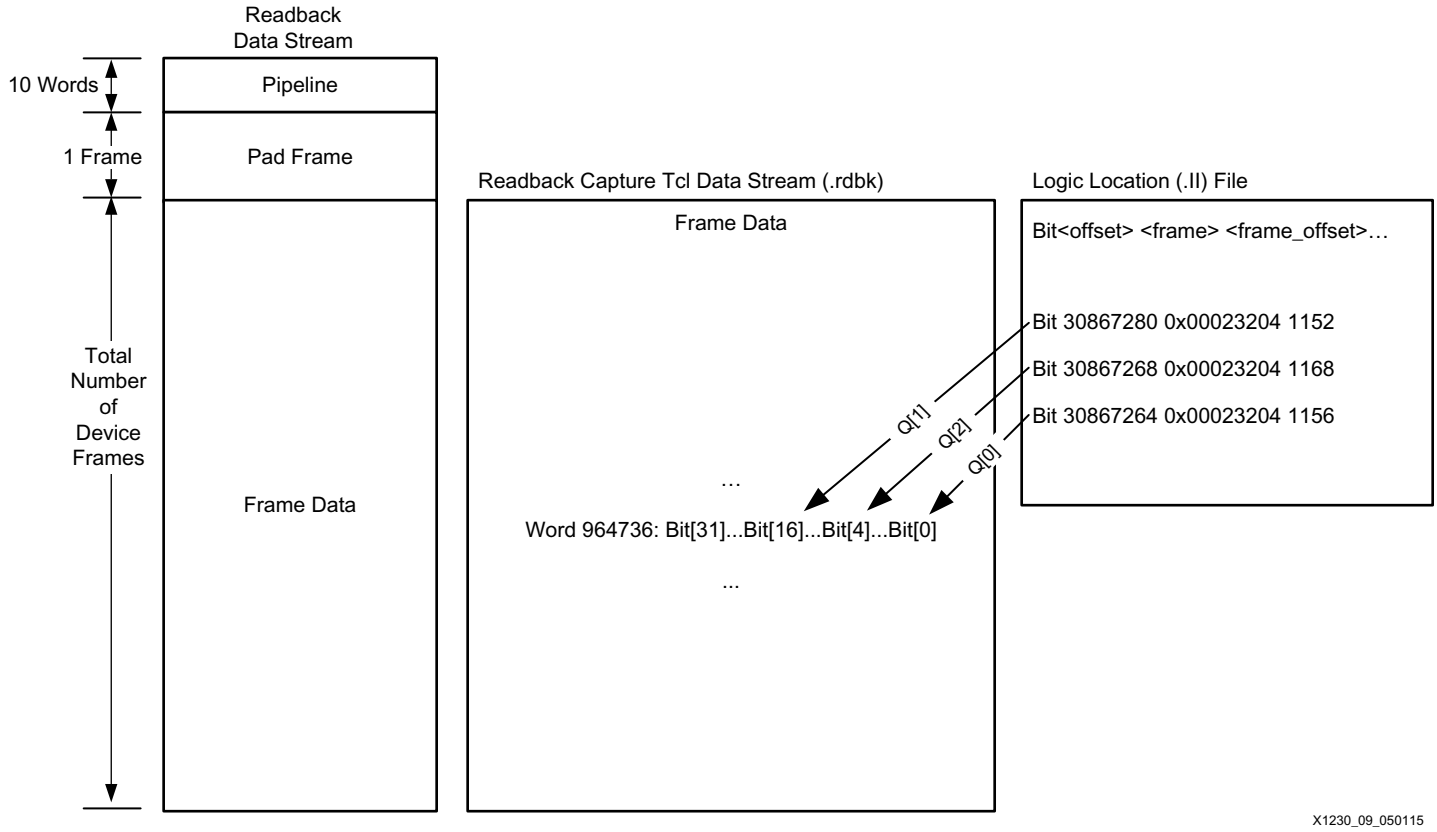

図 9 : `.ll` ファイルを使用してデザイン エレメントのロケーションを特定する例

リードバック キャプチャの実行時、すべての CLB レジスタで反転があります。CLB レジスタはキャプチャされるときに反転されるため、`0` はリードバック キャプチャ ファイルでは `1` と示されるはずです。UltraScale FPGA のブロック RAM、分散 RAM、あるいは SRL のキャプチャでは反転はありません。

この例では、TCK 値が 174 と入力されることで 8 ビット カウンター値が設定され、この値が `10101110` の LED[7:0] に反映されています。この値が使用されると、想定されるデザイン値は `10101110` となり、表 8 に示すように CLB レジスタのリードバック値は反転されます。

表 8 : リファレンス デザインのサンプル リードバック キャプチャ

リファレンス デザインのコンポーネント	ビット オフセット	想定されるデザイン値	想定される CLB レジスタ リードバック値 (反転)
Q[0]	ワード ライン 964736 のビット 0	0	1
Q[1]	ワード ライン 964736 のビット 16	1	0
Q[2]	ワード ライン 964736 のビット 4	1	0
Q[3]	ワード ライン 964736 のビット 20	1	0
Q[4]	ワード ライン 964763 のビット 0	0	1
Q[5]	ワード ライン 964763 のビット 16	1	0
Q[6]	ワード ライン 964763 のビット 4	0	1
Q[7]	ワード ライン 964763 のビット 20	1	0

チェックリスト

このセクションでは、デザインのセットアップ時に参考となるチェックリストを提供します。

ボード デザイン/回路図のチェックリスト

1. コネクティビティについては、使用するインターフェイスに基づいて『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) [参照 1] の記載に従います。
2. 提供される Tcl は、1 つの XCKU040 モノリシック デバイス用です。SSI テクノロジで構築されたデバイスまたは複数デバイスの JTAG チェーンの場合、JTAG 命令を変更する必要があります。

ソフトウェア フローのチェックリスト

1. デザイン入力 : リードバック キャプチャを用いてロジックを固定する方法 (関連するクロックを無効にする) を確認します。
2. Write_bitstream :
   a. `.ll` ファイルが生成されることを確認します。
   b. リードバック キャプチャを実行する場合は、次のビットストリーム設定/プロパティを回避します。
      - JTAG を無効
      - 暗号化
   c. SelectMAP インターフェイスを介したリードバック キャプチャの場合のみ、ビットストリーム プロパティで Persist が有効となっていることを確認します。
3. Vivado デバイス プログラマ :
   a. ターゲット デザインが問題なくコンフィギュレーションされていることを確認します。リファレンス デザインの場合は、KCU105 の DONE と INIT の両 LED が点灯していることを確認します。
   b. デザインが正常にロードされていない場合は、UltraScale デバイスのステータス レジスタがデバッグ情報を提供する有用な情報源となることがあります。詳細は、『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570) を参照してください。
4. TCL の使用 :
   a. リードバック キャプチャ プロセス コマンドの発行前に Tcl ファイルが source コマンドで実行されていることを確認します。
   b. JTAG コマンドが受信可能となるように、Tcl コンソールが open_hw_target -jtag_mode 1 となっていることを確認します。
   c. FPGA のプログラムを繰り返して実行する必要がある場合は、Tcl コマンド disconnect_hw_server を実行して -jtag_mode を終了させます。

デバッグ

リードバックの Tcl スクリプトを souce コマンドで再実行し、CLB レジスタのステートの変更を照合できます。たとえば、リファレンス デザインを用いた最初のリードバック キャプチャ実行後、反転されたパターン LED[7:0] = `01010001` を次の手順に従ってロードできます。

1. KCU105 ボードの SW12 RST 信号を `0` に設定して CLB レジスタ パターンをリセットします。次のコマンドを実行します。

```
>scan_ir_hw_jtag 6 -tdi 23
>runtest_hw_jtag -tck 1
```

2. CLB レジスタの反転されたパターンを次のコマンドでロードします。

```
>scan_ir_hw_jtag 6 -tdi 23
>runtest_hw_jtag -tck 81
>scan_ir_hw_jtag 6 -tdi 3f
>rdbk_jtag c:/XAPP1230/LED_Count_inverted.rdbk 32530
```

CLB レジスタ ビット ロケーションは、LED_Count.rdbk ファイルから反転される必要があります。

ユーザー フローを照合し、.ll ファイルでビット ロケーションをチェックするために、既知のステートに初期化されたレジスタ値を用いてデザインを作成できます。write_bitstream readback_file を選択し、キャプチャされたレジスタに INIT XDC 制約を使用する必要があります (つまり、set_property INIT 1'b1 [get_cells FDRE_1])。.ll ファイルに対して INIT ステートが指定された .rbd ファイルを使用すると、リードバックの実行前に、適切なロケーションがチェックされ、ユーザーの後処理用スクリプトが想定どおりに動作していることを確認できます。

SelectMAP インターフェイス

リードバックをより高速に実行する必要があるアプリケーションでは、SelectMAP または ICAP インターフェイスを介したリードバック キャプチャが選択可能です。SelectMAP または ICAP インターフェイスには JTAG TAP コマンドのオーバーヘッドが必要ないため、表 5 に示す手順の一部のみを適用できます。SelectMAP インプリメンテーションに必要な表 5 の手順は次のとおりです。

- 手順 3 : CAPTURE ビットのセット、開始フレーム アドレスの設定、読み出されるワード数の決定を担う、パケット データ情報。
- 手順 5 : シフトアウトされるリードバック ビット カウントを提供します。
- 手順 9 : CAPTURE ビットをデフォルト値にリセットするパケット データを提供します。

まとめ

リードバック キャプチャによって、内部 CLB レジスタ、ブロック RAM、分散 RAM、SRL 内容の現在のユーザー ステートを読み出すことで、デザインの機能が適切かどうかをチェック可能になります。確認する信号が多数あるデザイン デバッグのハードウェア エミュレーションまたは協調シミュレーションでは、リードバック キャプチャが重要な機能となります。この機能を使用すると、最小限の準備でデザイン ロジック リソースを追加することなく、デザイン ステートを簡単に確認できるようになります。KCU105 評価ボードを使用したリードバック キャプチャのデモ フローが含まれており、CLB レジスタのユーザー ステートが特定されます。

リファレンス デザイン

このアプリケーション ノートのリファレンス デザイン ファイルは、ザイリンクスのウェブサイトからダウンロードできます。

表 9 に、リファレンス デザインの詳細を示します。

表 9 : リファレンス デザインの詳細

パラメーター	説明
全般	
開発者	Stephanie Tapp
ターゲット デバイス	Kintex UltraScale FPGA XCKU040
ソース コードの提供	あり
ソース コードの形式	Verilog
既存のザイリンクス アプリケーション ノート/リファレンス デザイン、またはサードパーティからデザインへのコード/IP の使用	N/A
シミュレーション	
論理シミュレーションの実施	N/A
タイミング シミュレーションの実施	N/A
論理シミュレーションおよびタイミング シミュレーションでのテストベンチの利用	N/A
テストベンチの形式	N/A
使用したシミュレータ/バージョン	N/A
SPICE/IBIS シミュレーションの実施	N/A
インプリメンテーション	
使用した合成ツール/バージョン	Vivado Design Suite 2015.1
使用したインプリメンテーション ツール/バージョン	Vivado Design Suite 2015.1
スタティック タイミング解析の実施	N/A
ハードウェア検証	
ハードウェア検証の実施	あり
使用したハードウェア プラットフォーム	KCU105 評価ボード

参考資料

1. 『UltraScale アーキテクチャ コンフィギュレーション ユーザー ガイド』(UG570 : 英語版、日本語版)
2. 『Vivado Design Suite ユーザー ガイド : プログラムおよびデバッグ』(UG908 : 英語版、日本語版)
3. 『Vivado Design Suite Tcl コマンド リファレンス ガイド』(UG835 : 英語版、日本語版)
4. 『Kintex UltraScale アーキテクチャ データシート : DC 特性および AC スイッチ特性』(DS892 : 英語版、日本語版)
5. 『Virtex UltraScale アーキテクチャ データシート : DC 特性および AC スイッチ特性』(DS893 : 英語版、日本語版)
6. 『KCU105 評価キット クイック スタート ガイド』(XTP391)
7. 『KCU105 ボード ユーザー ガイド』(UG917)

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	内容
2015 年 6 月 9 日	1.0	初版

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。