XAPP1180 (v1.0) 2013 年 10 月 29 日

リファレンス システム : IP インテグレーターを用いた Kintex-7 MicroBlaze システム シミュレーション

著者 : James Lucero

概要

このリファレンス システムでは、IP インテグレーターをシミュレーションやハードウェアで使用して、Kintex®-7 デバイス アーキテクチャ上の MicroBlaze™ プロセッサ システムの機能を示します。このシステムには、システム デザインに重要なメイン メモリや RS232 などの一般的なペリフェラルが含まれます。

リファレンス システムにおけるペリフェラルの機能検証を目的として、スタンドアロンのソフトウェア アプリケーションがいくつか提供されています。アプリケーションには hello_uart および hello_mem が含まれます。

ここでは、システムのシミュレーション環境を設定する方法、Vivado® シミュレータまたは ModelSim® 環境のいずれかを使用してシミュレーションを実行する方法、およびハードウェアでデザインを実行する方法を説明します。Kintex-7 XC7K410TFFG900-2 FPGA を含む KC705 ボードをターゲットにして、シミュレーションおよびハードウェアでのデザイン実行について解説します。

含まれるシステム

このアプリケーション ノートにはリファレンス システムが含まれます。これは、`xapp1180.zip` ファイルから入手できます。

ファイルをローカル マシンで解凍すると、次のディレクトリ構造が使用されます。

```
mb_ddr_simulation/
    project_1.sdk/ - デザイン用のソフトウェア プロジェクトを含む
    project_1.srcs/ - エンベデッド デザイン用のファイルを含む
    sim/ - シミュレーションに必要なファイルを含む
    ready_for_download/ - このアプリケーション ノート用のプリコンパイル済みビットストリームおよびソフトウェア アプリケーションを含む
```

はじめに

このアプリケーション ノートでは、IP インテグレーター デザインを使用した、Kintex-7 デバイスを用いる MicroBlaze プロセッサ システムのシミュレーションおよびハードウェア生成について説明します。

必要な環境

シミュレーション、ソフトウェア、およびハードウェアの各要件を次に示します。

シミュレーション要件

VHDL および Verilog の協調シミュレーション機能または Vivado シミュレータを備える ModelSim SE/QuestaSim® Advanced Simulator 10.2a

ソフトウェア要件

Vivado 2013.3

ハードウェア要件

- ザイリンクス KC705 評価ボード (リビジョン C または D)
- Type-A/Mini-B の 5 ピン USB ケーブル 2 本

ハードウェア システム仕様

デザインの高性能な部分は次を使用しています。

- 200MHz での MicroBlaze プロセッサ
- 200MHz での AXI インターコネクト インスタンス
- 800MHz での AXI_7SERIES_DDRx メモリ コントローラー (データが幅が 64 ビットの DDR3 を備える)

デザインのより低い性能の部分は次を使用しています。

- 100MHz での AXI ペリフェラル IP (AXI4LITE)
- 100MHz での AXI インターコネクト インスタンス

リファレンス システムには次が含まれます。

- MicroBlaze
- AXI_INTERCONNECT
- MIG_7SERIES
- AXI_GPIO
- AXI_UART16550
- LMB BRAM

図 2 にブロック図を、表 1 にシステムのアドレス マップを示します。

ブロック図

図 2 に、リファレンス システムのブロック図を示します。

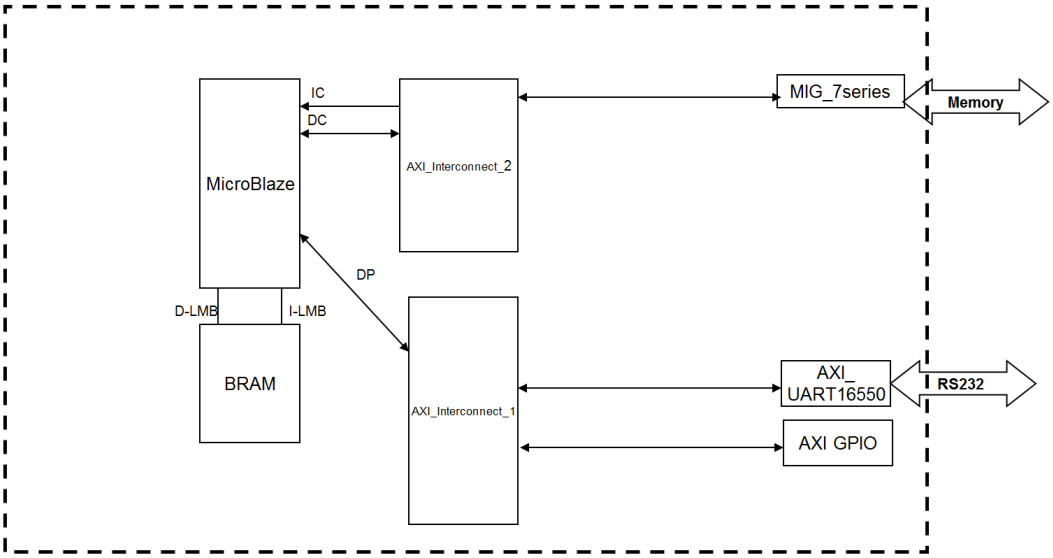

図2 : リファレンス システムのブロック図

アドレス マップ

表 1 に、リファレンス システムのアドレス マップを示します。

表 1 : リファレンス システムのアドレス マップ

ペリフェラル	インスタンス	ベース アドレス	上位アドレス
mig_7series	mig_7series_1	0x80000000	0xBFFFFFFF
axi_uart16550	axi_uart16550_1	0x40400000	0x40401FFF
axi_gpio	axi_gpio_1	0x42000000	0x4200FFFF
lmb_bram_if_cntlr	lmb_bram_if_cntlr_1	0x00000000	0x0000FFFF
lmb_bram_if_cntlr	lmb_bram_if_cntlr_2	0x00000000	0x0000FFFF

ソフトウェア アプリケーション

プロセッサの命令は、LMB (Local Memory Bus) BRAM (ブロック RAM) または MIG_7SERIES のいずれかで、SDK プロジェクトのリンカー スクリプトによって実行されます。

Hello UART ソフトウェア アプリケーション

このソフトウェア アプリケーションは AXI UART16550 の機能をテストします。シミュレーション済みの RS232 ターミナルに `Hello World!` が表示され、Vivado では Tcl コンソールに、ModelSim では [Transcript] ウィンドウに出力が表示されます。

このソフトウェア アプリケーションでは、MIG_7SERIES メモリ コントローラーを使用して DDR3 メモリが不足します。

Hello Mem ソフトウェア アプリケーション

このソフトウェア アプリケーションは MIG_7SERIES メモリ コントローラーの機能をメモリ テストを用いて検証します。GPIO 入力がメモリ コントローラーからの init_done 信号に接続されているため、ソフトウェア プリケーションは GPIO レジスタをポーリングし、メモリ コントローラーがいつキャリブレーションされるかを判断します。メモリのキャリブレーション後、ショート メモリ テスト (テスト パターンで 32 ワードの XIL_TESTMEM_INCREMENT) がメモリ コントローラーで実行されます。メモリ テストの結果に基づいて `PASSED!` または `FAILED!` メッセージがシミュレーション RS232 ターミナルに表示されます。

このソフトウェア アプリケーションでは、LMB BRAM を使用して内部メモリが不足します。

ModelSim/Questa シミュレータ向けにシミュレーション環境を設定する

ライブラリのコンパイル

1. `MODELSIM` (Linux 用) または `MODELTECH` (Windows 用) 変数が書き込み可能で有効な `modelsim.ini` ファイルを指定していることを確認します。
2. Vivado ツールを開きます。
3. Tcl コンソールで、EDK ライブラリを除くすべてのザイリンクス ライブラリをコンパイルするために次のコマンドを入力します。

   ```
   compile_simlib -simulator modelsim -library unisim -library simprim
   -library xilinxcorelib -language all -directory
   <PATH_TO_COMPILED_LIBRARIES>
   ```

   注記 : コンパイル時間はローカル マシンによって異なります。

Vivado プロジェクトでは、[Simulation] の [Compiled library location] でコンパイル済みのライブラリ ディレクトリを指定します。この手順は後述します。

シミュレーションのディレクトリおよびファイル

1 つのシミュレーション ディレクトリを Linux および Windows 環境の両方に使用します。これはプロジェクトのメイン ディレクトリの sim/ です。

このエリアにあるスクリプトは、リンカー スクリプトが DDR アドレス空間用に設定されている場合に ELF ファイルを、プロセッサ命令を用いてメモリ モデルをロード アップするファイルに変換できます。このプロセス実行中に、Data2MEM を使用して初期 .mem ファイルを生成し、プロセッサ命令でメモリ モデルを正しく読み込むフォーマットに .mem ファイルを perl スクリプトを使用して変換します。メモリ コントローラーがキャリブレーションされると、このシミュレーション テストベンチは .mem ファイルをメモリに読み込みます。

さまざまなサイズの DDR3 に応じてテストベンチ/スクリプトを変更する

各 .mem ファイルは 1 つの 16 ビット メモリ モデルに関連付けられています (たとえば、64 ビット DDR3 には 4x16 ビットのメモリ モデルが必要)。

テストベンチ (`system_tb.v`) は、DDR3 コンポーネントに必要なメモリ モデルの容量によって変更されます (16 ビットでは 1x、32 ビットでは 2x、64 ビットでは 4x)。

`gen_memfiles.sh/gen_memfiles.bat` コマンドは、使用する .mem ファイルに基づいて変更されます (1x sim/ddr3_0.mem、2x sim/ddr3_0.mem sim/ddr3_1.mem、4x sim/ddr3_0.mem sim/ddr3_1.mem sim/ddr3_2.mem sim/ddr3_3.mem)。

`memory_init.bmm` ファイルは、メモリのアドレス空間、メモリの幅、および使用する .mem ファイルに基づいて変更されます。

これらの変更用に、16 ビット DDR3 の例が `sim/example_16` ディレクトリにあります。

さまざまな UART 周波数に応じてテストベンチを変更する

UART コアのクロック周波数/AXI 周波数の変更に使用する命令は、テストベンチ (`system_tb.v`) のコメントにあります。

シミュレーション ファイル

sim/ ディレクトリには次の関連ファイルおよびディレクトリが含まれます。

- `512Mb_ddr3/` - Micron 社 DDR3 メモリ モデル
- `gen_memfiles.sh` - ELF を .mem ファイルに変換する Linux メイン スクリプト
- `gen_memfiles.bat` - ELF を .mem ファイルに変換する Windows メイン スクリプト
- `memory_init.bmm` - 初期 .mem ファイルを ELF ファイルから生成するために data2mem が使用するブロック メモリ マップ ファイル
- `ddr3_x.mem` - 外部メモリ モデルをロードするメモリ ファイル。提供されるファイルは hello_uart 外部メモリ ファイルを使用。
- `system_tb.v` - デザイン用の Verilog テストベンチ
- `uart_rcvr.v` - シミュレーション済みの RS232 ターミナル用
- `uart_rcvr_wrapper.v` - シミュレーション済みの RS232 ターミナル用

注記：ソフトウェア アプリケーションからの出力は、シミュレーション実行中に ModelSim/Questa シミュレータ ターミナル内部または Vivado Tcl コンソール内部でモニタリングされます。

シミュレーションの実行

IP インテグレーターからシミュレーションを実行する

次の手順に従って IP インテグレーターからシステム シミュレーションを実行します。

プロジェクトを開く/シミュレーション モードを設定する、およびデザイン ファイルを生成する

1. Vivado 内で <unzip dir>/mb_ddr_simulation/project_1.xpr を開きます。

   注記：デフォルトでは、Vivado シミュレータがターゲット シミュレータに選択されています。設計アシスト機能は実行しないでください。

2. Flow Navigator で、[IP Integrator] を展開して [Open Block Design] をクリックします。
3. design_1.bd を選択します。図 4 にブロック図を示します。

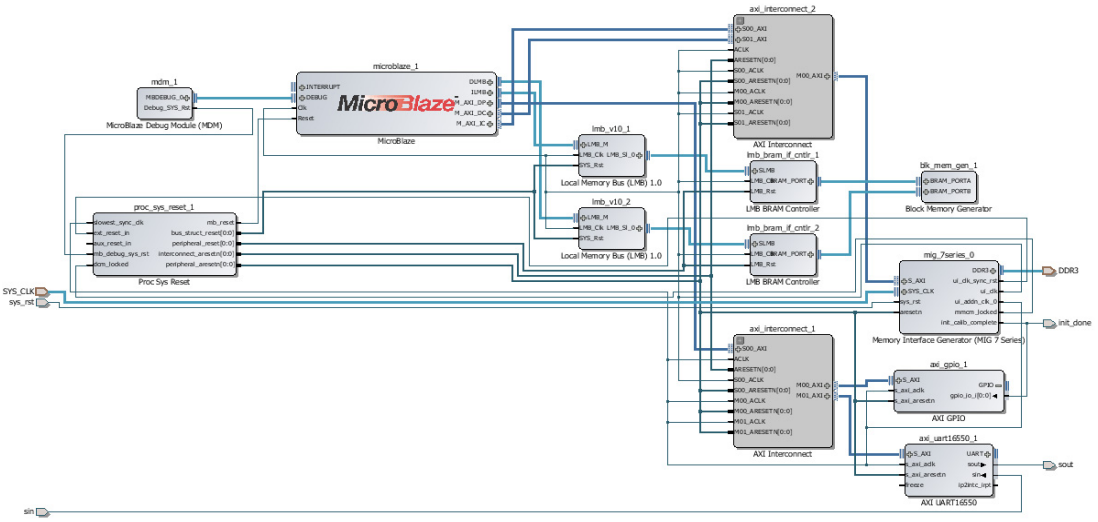

図4：IP インテグレーターのブロック図

4. [Sources] ビューで [design_1_wrapper] を展開します。
5. [design_1_i] を右クリックし、[Generate Output Products] をクリックします。
6. [Generate] をクリックしてデザイン ファイルとラッパーを生成します。
7. ModelSim/QuestaSim ユーザー対象
   a. Flow Navigator で、[Simulation] を展開して [Simulation Settings] をクリックします。

      [Project Settings] → [Simulation] タブを開きます。

   b. [**Target simulator**] をクリックして [**select QuestaSim/ModelSim**] を選択し、[**Target Simulator**] ダイアログ ボックスで [**YES**] をクリックします。
   c. [**Complied library location**] では、`<PATH_TO_COMPILED_LIBRARIES>` を選択します。
   d. [Simulation] タブをクリックして、[-noopt] が [More VSIM Options] に含まれていることを確認します。
8. Flow Navigator で [Project Manager] をクリックします。
9. [Sources] ビューで、[Simulation Sources] を展開した後、[sim_1] を展開します。
10. [system_tb] を右クリックし、[Associate ELF Files] をクリックします。[Simulation Sources] の microblaze_1 に対して、[Associated ELF File] 列の下にある [Browse] (...) ボタンをクリックします。

11. シミュレーションに使用するプロジェクトの [ELF File] を選択します。
12. IDDR3 メモリ空間にコードが存在する場合は、「Windows 用に外部メモリ ファイルを生成する」または「Linux 用に外部メモリ ファイルを生成する」の手順に従ってください。

    注記：追加された ELF ファイルが project_1.srcs/sim_1/imports/ にコピーされます。プロジェクトにほかの ELF ファイルを追加するには、[Project Manager] で [Add Sources] → [Add or Create Simulation Sources] → [Add Files] をクリックし、追加する ELF ファイルを選択します。デフォルトでは、hello_uart application が選択されています。

Windows 用に外部メモリ ファイルを生成する

1. コマンド プロンプトを開きます。ザイリンクス ツール インストール ディレクトリから settings32.bat または settings64.bat を実行します。一般的なインストール ディレクトリは C:\Xilinx\Vivado\2013.3 です。
2. 同じコマンド プロンプト内で次のパスを設定します。

   注記：パスおよび Windows のバージョンは、ローカル インストールによって異なり、下記に示す内容と異なる可能性があります。このパスのセットアップ ファイルの例は、<unzip dir>/mb_ddr_simulation/path64.bat または <unzip dir>/mb_ddr_simulation/path32.bat にあります。

   ```
   set PATH=C:\Xilinx\Vivado\2013.3\bin;C:\Xilinx\Vivado\2013.3\lib\win64.o;
   C:\Xilinx\Vivado\2013.3\tps\win64\jre\bin\server;C:\Xilinx\Vivado\2013.3\
   tps\win64\jre\bin;C:\Xilinx\SDK\2013.3\bin\nt64;C:\Xilinx\Vivado\2013.3\
   ids_lite\EDK\bin\nt64;C:\Xilinx\Vivado\2013.3\ids_lite\EDK\lib\nt64;C:\
   Xilinx\Vivado\2013.3\ids_lite\ISE\bin\nt64;C:\Xilinx\Vivado\2013.3\ids_
   lite\ISE\lib\nt64;%PATH%
   ```
3. 同じコマンド プロンプト内で、Vivado プロジェクトのメイン ディレクトリである <unzip dir>\mb_ddr_simulation\ にディレクトリを変更します。
4. 次のコマンドを実行します。

   ```
   sim\gen_memfiles.bat <location of ELF file>
   ```

   注記：この実行には 20 秒ほどかかり、コマンド ウィンドウがいくつか開き、閉じます。メモリ ファイル (.mem) は sim/ ディレクトリにあります。

Linux 用に外部メモリ ファイルを生成する

1. Vivado プロジェクトの現在のディレクトリが mb_ddr_simulation/ であることを確認します。
2. sim/ ディレクトリの実行権限があることを確認します。

   ```
   exec sim/gen_memfiles.sh <location of ELF file>
   ```

注記：メモリ ファイル (.mem) は sim/ ディレクトリにあります。

シミュレーションを実行する

1. Flow Navigator で、[Simulation] → [Run Simulation] → [Run Behavioral Simulation] をクリックしてシミュレータを起動します。
2. Vivado シミュレータまたは Modelsim/Questa がデザインのコンパイルを完了したら、必要な信号をデザインに追加し、選択したソフトウェア アプリケーション向けに指定された実行時間で run コマンドを実行します。たとえば、[**run 200us**] です。

ソフトウェア アプリケーションを実行する

Hello UART ソフトウェア アプリケーションを実行する

このシミュレーションの実行には ~240µs かかります。Vivado シミュレータの実行時間は、1 ～ 2 時間ほどかかる可能性があります。

AXI UART16550 は、シミュレーション済みの RS232 ターミナルおよび/または ModelSim トランスクリプト ウィンドウ、または Vivado シミュレータ Tcl コンソールへの出力送信を開始します。図 5 にその出力を示します。

```
UART    0 Transmitted Out Char = 0x48 = "H" @ Time = 173475000.0 ps
UART    0 Transmitted Out Char = 0x65 = "e" @ Time = 179715000.0 ps
UART    0 Transmitted Out Char = 0x6c = "l" @ Time = 185475000.0 ps
UART    0 Transmitted Out Char = 0x6c = "l" @ Time = 191235000.0 ps
UART    0 Transmitted Out Char = 0x6f = "o" @ Time = 196995000.0 ps
UART    0 Transmitted Out Char = 0x20 = " " @ Time = 202755000.0 ps
UART    0 Transmitted Out Char = 0x55 = "U" @ Time = 208995000.0 ps
UART    0 Transmitted Out Char = 0x61 = "a" @ Time = 214755000.0 ps
UART    0 Transmitted Out Char = 0x72 = "r" @ Time = 220515000.0 ps
UART    0 Transmitted Out Char = 0x74 = "t" @ Time = 226275000.0 ps
UART    0 Transmitted Out Char = 0x0d = <special char> @ Time = 232515000.0 ps
```

図5 : Hello UART 出力

Hello Mem ソフトウェア アプリケーションをシミュレーションする

このシミュレーションの実行には ~142µs かかります。Vivado シミュレータの実行時間は、1 時間ほどかかります。

メモリ テストが問題なく完了したら、ModelSim/Questa トランスクリプト ウィンドウまたは Vivado シミュレータ Tcl コンソールに出力が表示されます。図 6 にその出力を示します。

```
UART    0 Transmitted Out Char = 0x50 = "P" @ Time = 102135000.0 ps
UART    0 Transmitted Out Char = 0x41 = "A" @ Time = 106935000.0 ps
UART    0 Transmitted Out Char = 0x53 = "S" @ Time = 111735000.0 ps
UART    0 Transmitted Out Char = 0x53 = "S" @ Time = 116535000.0 ps
UART    0 Transmitted Out Char = 0x45 = "E" @ Time = 121335000.0 ps
UART    0 Transmitted Out Char = 0x44 = "D" @ Time = 126135000.0 ps
UART    0 Transmitted Out Char = 0x21 = "!"@ Time = 130935000.0 ps
UART    0 Transmitted Out Char = 0x0d = <special char> @ Time = 135735000.0 ps
```

図6 : Hello Mem 出力

ハードウェア上でのシステムの実行

このセクションでは、リファレンス デザインをハードウェアで実行する手順について説明します。

1. USB ケーブルをホスト PC から USB JTAG ポートに接続します。適切なデバイス ドライバーがインストールされていることを確認します。
2. 別の USB ケーブルをホスト PC から USB UART ポートに接続します。USB-UART ドライバーがインストールされていることを確認します。
3. 電源をオンにします。
4. ホスト PC 上で、次の設定でハイパーターミナルなどの端末プログラムを開始します。
   - ボー レート : 9600
   - データ ビット : 8
   - パリティ : なし
   - ストップ ビット : 1
   - フロー制御 : なし

構築済みビットストリームとコンパイル済みソフトウェア アプリケーションを使用してリファレンス システムを実行する

`<unzip_dir>/mb_ddr_simulation/` ディレクトリの `ready_for_download/` ディレクトリにあるファイルを使用してシステムを実行する手順は、次のとおりです。

1. Xilinx Microprocessor Debugger (XMD) ツールを起動します。
2. `ready_for_download` ディレクトリへ移動します。
3. XMD にビットストリームをダウンロードします。

   ```
   XMD% fpga -f design_1_wrapper.bit
   ```
4. XMD でプロセッサに接続します。

   ```
   XMD% connect mb mdm
   ```
5. プロセッサ コード (ELF) ファイルをダウンロードします。

   ```
   XMD% dow hello_uart.elf
   ```

   または

   ```
   XMD% dow hello_mem.elf
   ```
6. ソフトウェアを実行します。

   ```
   XMD% run
   ```

「Hello UART」または「PASSED!」がハイパーターミナル画面に表示されます。

ハードウェアの構築

このセクションでは、Vivado を用いたハードウェア デザインの再構築について説明します。

注記：生成されたビットストリームは次の場所にあります。

```
<unzip dir>/mb_ddr_simulation/project_1.runs/impl_1/design_1_wrapper.bit
```

1. Vivado 内の <unzip dir>/mb_ddr_simulation/project_1.xpr を開きます。
2. Flow Navigator で次を実行します。
   a. [IP Integrator] を展開し、[Open Block Design] をクリックします。
   b. [design_1.bd] を選択します。
3. [Sources] ビューで次を実行します。
   a. [design_1_wrapper] を展開します。
   b. [design_1_i] を右クリックし、[Generate Output Products] をクリックします。

c. [Generate] をクリックしてデザイン ファイルおよびラッパーを生成します。

4. Flow Navigator で [Project Manager] をクリックします。
5. [Log] ビューで [Design Runs] タブをクリックします。
6. [impl_1] を右クリックして [Launch Run] をクリックし、[Launch Selected Runs] ダイアログ ボックスで [OK] をクリックします。
7. [Missing Synthesis Results] ボックスで [OK] をクリックしてネットリストを生成します。

   注記：この手順は、マシンによっては 30 分～ 60 分ほどかかります。
8. 完了したら、[Implementation Completeded] ダイアログ ボックスで [Generate Bitstream] をクリックします。

   注記：特にデザインが変更された場合は、インプリメンテーションのタイミングが満たされていることを確認します。
9. [Design Runs] ウィンドウで、[impl_1] を右クリックして [Open Implemented Design] をクリックします。
10. [Project Manager] の [Sources] ビューで次を実行します。
    a. [design_1_i] を右クリックし、[Export Hardware for SDK] をクリックします。
    b. [Export Hardware] および [Include bitstream] をクリックします。
    c. [OK] をクリックします。

SDK ツールによるソフトウェアのコンパイルとデザインの実行

このセクションでは、SDK ツールを用いたソフトウェアのコンパイル、そしてその後に続く、SDK からのハードウェアとソフトウェアの実行について説明します。

SDK ツールでソフトウェアをコンパイルする

1. SDK を起動します。Linux の場合は xsdk と入力します。
2. Workspace Launcher で、次のワークスペースを選択します。

   ```
   <unzip dir>\project_1.sdk\SDK_Workspace
   ```
3. [OK] をクリックします。

   注記：手順 4 ～手順 7 は、空の SDK_Workspace にのみ必要です。
4. ハードウェア プラットフォームの BSP およびソフトウェア アプリケーションをインポートする必要があります。[File] → [Import] → [General] → [Existing Projects into Workspace] をクリックします。
5. [Next] をクリックします。
6. <unzip dir>\project_1.sdk を選択して [OK] をクリックします。
7. standalone_bsp_0、hw_platform_0、および関連するソフトウェア アプリケーションを含む、すべてのチェック ボックスがオンになっていることを確認してください。
8. [Finish] をクリックします。

   注記：この時点で BSP およびソフトウェア アプリケーションがコンパイルされます。これには 2 ～ 5 分かかります。その他のソフトウェア アプリケーションについてのエラー メッセージが表示された場合、[Project Explorer] でソフトウェア アプリケーション プロジェクトを右クリックして [Change Referenced BSP] をクリックし、[standalone_bsp_0] をクリックします。[OK] をクリックします。
9. これで、SDK で既存のソフトウェア アプリケーションの変更やソフトウェア アプリケーションの新規作成が可能になりました。

注記：デフォルトでは、含まれているソフトウェア アプリケーションは -DSIM = 1 に設定されています。このオプションを変更するには、ソフトウェア アプリケーション上で右クリックして [C/C++ Build

Settings] → [MicroBlaze gcc complier] → [Miscellaneous] をクリックします。ハードウェアでソフトウェアを実行する場合は、このオプションを削除します。

SDK ツールによってハードウェアおよびソフトウェアを実行する

1. SDK_Workspace プロジェクトが開いた状態で、[Xilinx Tools] → [Program FPGA] をクリックします。
2. [Program] をクリックします。
3. [Project Explorer] で、(Software_Application_Name) → [Run As] → [Launch on Hardware] を右クリックします。

システムの変更

システムを変更する手順は、次のとおりです。

1. インスタンスを次のように接続します。

インスタンス	接続先
AXI4Lite プロトコルを用いる AXI マスター /スレーブ インスタンス	axi_interconnect_1
AXI4 プロトコルを用いる AXI マスター /スレーブ インスタンス	axi_interconnect_2

2. ブロックの変更および保存後、既存の SDK_Workspace 向けに新しい SDK_Export ディレクトリを生成し、デザインを再生成します。
3. [Project Manager] の [Sources] ビューで、[design_1_wrapper] を展開します。

   注記：8 ページの「ハードウェアの構築」を参照して新しいビットストリームを生成してください。
4. [design_1_i] を右クリックし、[Generate Output Products] をクリックします。
5. [Generate] をクリックしてデザイン ファイルおよびラッパーを生成します。
6. [design_1_i] を右クリックし、[Export Hardware for SDK] をクリックします。
7. [Module Already Exported] ダイアログ ボックスで [YES] をクリックします。
8. SDK プロジェクトで、[hw_platform_0] を右クリックして [Change Hardware Platform Specification] をクリックします。
9. 開く警告ダイアログ ボックスで [Yes] をクリックします。
10. 新しく作成された design_1.xml へのリンクを提供します。このファイルは、<unzip dir>\project_1.sdk\SDK\SDK_Export\hw にあります。
11. [OK] をクリックします。
12. 必要に応じて「シミュレーションの実行」セクションの手順を再実行してください。

参考資料

この資料のデザインファイルは、https://secure.xilinx.com/webreg/clickthrough.do?cid=344919 にあります。

次の文書は、このアプリケーション ノートの補足資料です。

- UG081 :『MicroBlaze プロセッサ リファレンス ガイド』
- UG900 :『Vivado Design Suite ユーザー ガイド : ロジック シミュレーション』
- UG994 :『Vivado Design Suite ユーザー ガイド : IP インテグレーターを使用した IP サブシステムの設計』
- UG898 :『Vivado Design Suite ユーザー ガイド : エンベデッド ハードウェア デザイン』
- UG810 :『Kintex-7 FPGA 用 KC705 評価ボード ユーザー ガイド』

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
2013 年 10 月 2 日	1.0	初版

Notice of Disclaimer