

Introduction

This application note describes a simple Excalibur™ system design that consists of software running on the processor and logic in the FPGA portion of the device, to demonstrate the features and flexibility of Excalibur devices. The design contains hardware logic that runs independently of the processor subsystem and also contains logic that is driven by the processor subsystem.

This document explains the design and shows you how to download it to the EPXA10 development board. It also describes how to run the design on the board and how to connect to it via a debugger.



For more details about the EPXA10 device, refer to the *Excalibur Devices Hardware Reference Manual*; and for more details about the EPXA10 development board, refer to the *Excalibur EPXA10 Development Board User Guide*.



EXCALIBUR™

The design downloaded to the development board in this guide exercises the processor subsystem, the FPGA logic, and demonstrates simple transfers of data between the processor subsystem and the FPGA peripherals.



Software Requirements

The following software is required to perform the procedures described in this document:

- Either of the following:
 - The Quartus® II software version 2.1
 - Excalibur utilities
- Altera-ARM Developer Suite (ADS)-Lite

The instructions in this document assume that:

- You are using a PC running Windows 98/NT/2000
- The software is installed on your PC in the default location



For more information on any of the software used in this document, refer to the documentation provided with the software or see the Altera website www.altera.com.

Quartus II Software

The Quartus II development software provides a comprehensive environment for SOPC design. The Excalibur MegaWizard® Plug-In, included with the Quartus II software, enables quick and intuitive setup and customization of the processor subsystem in the EPXA10 device. To create new projects or to modify embedded hardware in an existing EPXA10 project, the Quartus II development software is required.

Excalibur Utilities

The Excalibur utilities are applications required to start using the EPXA10 development board. The utilities included allow you to create programming files, set up the programming hardware, and download applications to flash memory. The utilities are installed automatically when you install the Quartus II software. If you do not need to modify the hardware portions of the design and do not require the Quartus II software, you can install the utilities separately, using the standalone installer available on the SOPC Builder CD-ROM.

Tools for Developing Embedded Software

To develop new embedded software, or modify existing applications, for the Excalibur family, you need to install software development tools such as the following:

- The Altera ADS-Lite toolset
- GNUPro Toolkit for ARM

Quartus II subscriptions include evaluation licences for ADS-Lite.

In this simple Excalibur EPXA10 system, you will use the AXD debugger supplied with ADS-Lite to observe the operation of the example design.



For more details on editing the hardware and software design, refer to the *EPXA10 Development Kit Getting Started User Guide*.

Design Example

The design example described in this document contains both hardware and software aspects that demonstrate some of the features and capabilities of the EPXA10 and the development board.



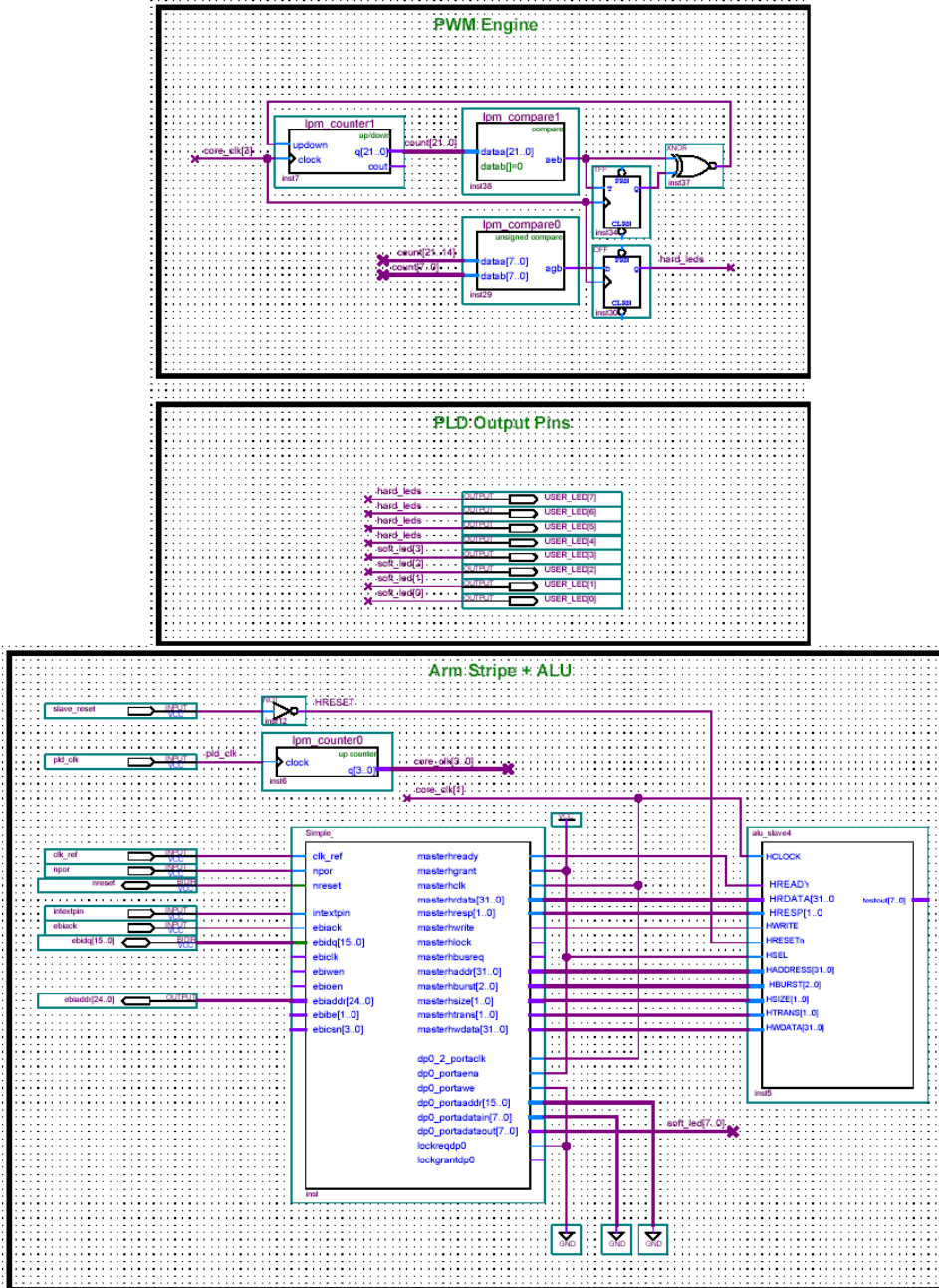
The design is configured to use a 50-MHz input clock. A PLL embedded in the EPXA10 device generates a 150-MHz clock to the processor. If a different-frequency input clock is used, the design must be modified accordingly and then recompiled. The frequency of the crystal installed on the EPXA10 development board is 50 MHz.



This application note focuses on observing a pre-compiled design. To follow the procedure for developing and compiling the design yourself, refer to the *EPXA10 Development Kit Getting Started User Guide*.

Figure 1 on page 4 shows a Quartus II representation of the design example.

Figure 1. Quartus II Schematic of the Design Example



Hardware

The hardware in the design consists of two parts: the processor subsystem, and the general-purpose FPGA logic.

The processor subsystem exercises the following five features of the product:

- *Processor subsystem*—the processor subsystem executes the software part of the design
- *Single-port SRAM*—the code running on the processor subsystem is stored in the on-chip single-port SRAM
- *Dual-port SRAM*—the on-chip dual-port SRAM is used as an interface between the processor subsystem and the FPGA
- *Stripe-to-PLD AHB master bridge*—this bridge gives the processor subsystem access to the 32-bit arithmetic-logic unit (ALU) located in the FPGA
- *External bus interface (EBI)*—EBI block 0 (EBI0) is used to connect the processor subsystem to the flash memory on the EPXA10 development board

The following three functions are implemented in the FPGA portion of the example design's hardware:

- *Pulse width modulation (PWM) circuit*—a simple pulse width modulator drives `USER_LED[7:4]`. A 24.4 KHz pulse is constantly sent to the LEDs. The width of the pulse is varied between 0 and 100% to give the illusion of the LEDs slowly becoming brighter, then slowly dimming again
- *DPRAM interface*—the PLD interfaces with one of the ports of DPRAM0. The four least-significant bits of DPRAM0 base address 0 are connected to I/O pins, which connect to `USER_LED[3:0]`
- *AHB-connected ALU*—a simple three-function 32-bit ALU is connected to the processor-to-PLD master bridge. The ALU can be operated from the processor to perform addition, subtraction, or multiplication

Software

The software in this design continually shifts a logic 1 back and forth through a local variable, writing the variable to DPRAM0 every time the value changes. A delay loop is executed after writing the variable to DPRAM. This delay loop slows the shifting sufficiently so that the changing patterns can be seen on the LEDs.

Using the AXD Debugger

The Altera ADS-Lite toolset includes the AXD software debugger. The following section outlines the steps required to exercise the design, including the FPGA ALU located in the hardware portion of the design, using AXD.



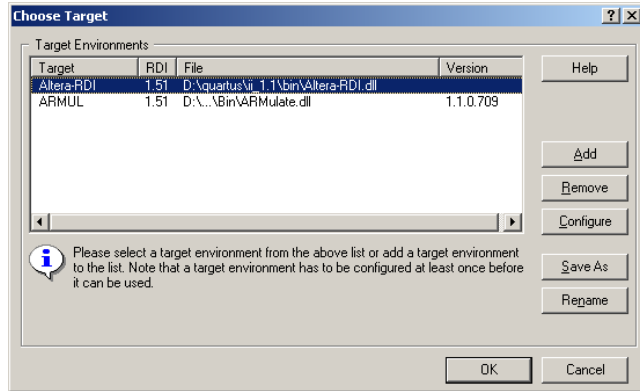
For more information on the AXD Debugger, refer to the *ARM Developer Suite—Debuggers Guide*.



The Excalibur utilities include the RDI (**Altera_RDI.dll**). The RDI is the standard application interface between ARM processors and ARM-supported debuggers. If your debugger supports the RDI from ARM Limited and the ARM922T processor, you can use your existing debugger with an EPXA10 device.

To set up the AXD Debugger and target the EPXA10's embedded processor, perform the following steps.

1. Start the AXD Debugger by selecting **Start > Programs > ARM Developer Suite > AXD Debugger** (Start menu).
2. Select **Configure Target** (Options menu).
3. If Altera-RDI is listed as a target, click it to highlight it (see [Figure 2 on page 7](#)), then click **OK** to connect to the embedded processor. If Altera-RDI is not listed as a target, you must add it, by performing the following steps:
 - a. Click **Add** in the **Choose Target** window.
 - b. Browse to the directory in which the Quartus II software is installed.
 - c. Navigate to the **bin** directory.
 - d. Select **Altera-RDI.dll**. Click **Open**.
 - e. Click on **Altera-RDI** in the **Choose Target** window, click **OK** to connect to the embedded processor (see [Figure 2](#)).

Figure 2. Selecting Altera-RDI

The AXD Debugger launches and pauses the embedded processor. The illumination pattern in `USER_LED [3 : 0]` stops.



`USER_LED [7 : 4]` will continue to illuminate intermittently. The LEDs are controlled by PLD logic that is not affected by pausing the embedded processor.

Stepping through the Code

To step through the software code running on the embedded processor, select **Execute > Step** (AXD menu) or press **F10**. A window displays the disassembled software code with a blue arrow pointing to the location indicated by the program counter. Each time **F10** is pressed, the blue arrow advances (or branches) to the next instruction.

To resume normal embedded processor operation, select **Execute > Go** or press **F5**. The paused LEDs again illuminate intermittently.

To pause the embedded processor again, select **Execute > Stop** or press **Shift+F5**.

Manipulating the LEDs

`USER_LED[3:0]` can be manually controlled with the AXD debugger. Because the LEDs are controlled by the embedded processor through the dual-port RAM, they can also be controlled by the debugger writing to dual-port RAM. To change the value displayed on `USER_LED[3:0]`, perform the following steps:

1. Pause the embedded processor (select **Execute > Stop** or press **Shift+F5**).
2. Select **Processor Views > Memory** (AXD menu).
3. In the memory window, enter `0x40000` in **Start Address** and press **Enter**. This is the address at which DPRAM0 is mapped.
4. Right-click the array of memory values and select **Size > 32 bit** to show the memory as 32-bit values.
5. Double-click on the value at address `0x00040000` and enter `0xF`. Press **Enter**.



This extinguishes `USER_LED[3:0]` because they are active-low signals.

Writing `0x0` illuminates the LEDs.

Operating the Custom Peripheral

The 32-bit custom peripheral is mapped at address `0x10000000`. Its registers are as shown in [Table 1](#).

Memory Location	Register Name
<code>0x10000004</code>	Operand A
<code>0x10000008</code>	Operand B
<code>0x1000000C</code>	Operator
<code>0x10000010</code>	Result[31:0]
<code>0x10000014</code>	Result[63:32]

The valid operators are as follows:

- 0x05, add
- 0x06, subtract
- 0x07, multiply

To exercise the peripheral, change the values of operand A, operand B, and the operator. Bit 2 of the operand register is write-only. The value entered as the operand is masked when the debugger reads the operand register. For example, if 0x5 is entered as the operand, the value 0x1 is displayed in the debugger. After entering the operator and operands, refresh the memory window to view the result of the operation (either right-click the memory array and click **Refresh**, or start and stop the embedded processor).



For more information on the custom peripheral, refer to the *Excalibur Hardware Design Tutorial*, which describes its implementation and verification.



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
<http://www.altera.com>
[Applications Hotline:](#)
(800) 800-EPLD
[Literature Services:](#)
lit_req@altera.com

Copyright © 2002 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, mask work rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001