Spartan-6 LX9 MicroBoard Embedded Tutorial

Tutorial 4

Embedded System Integration to ISE

Version 12.4.01

Revision History

Version	Description	Date
12.4.01	Initial release for EDK 12.4	3/17/2011

Table of Contents

Table of Figures

Overview

This is the forth tutorial in a series of training material dedicated to introducing engineers to creating their first embedded designs. These tutorials will cover all the required steps for creating a complete MicroBlaze design in the Spartan-6 LX9 MicroBoard. While dedicated to this platform, the information learned here can be used with any Xilinx FPGA.

The embedded system is currently a top-level module inside ISE. In order to add user logic, it is often necessary to have the embedded system as a sub-module in the FPGA design. This lab is divided into three main steps: Modifying the embedded system in ISE to be a sub-module, adding the executable located into internal BRAMs, and testing the system on the FPGA. The test application will reside in Block memory inside the FPGA. This lab uses the same MicroBlaze Hardware Platform used in the previous tutorial. The top-level module will look like this when the tutorial is completed:

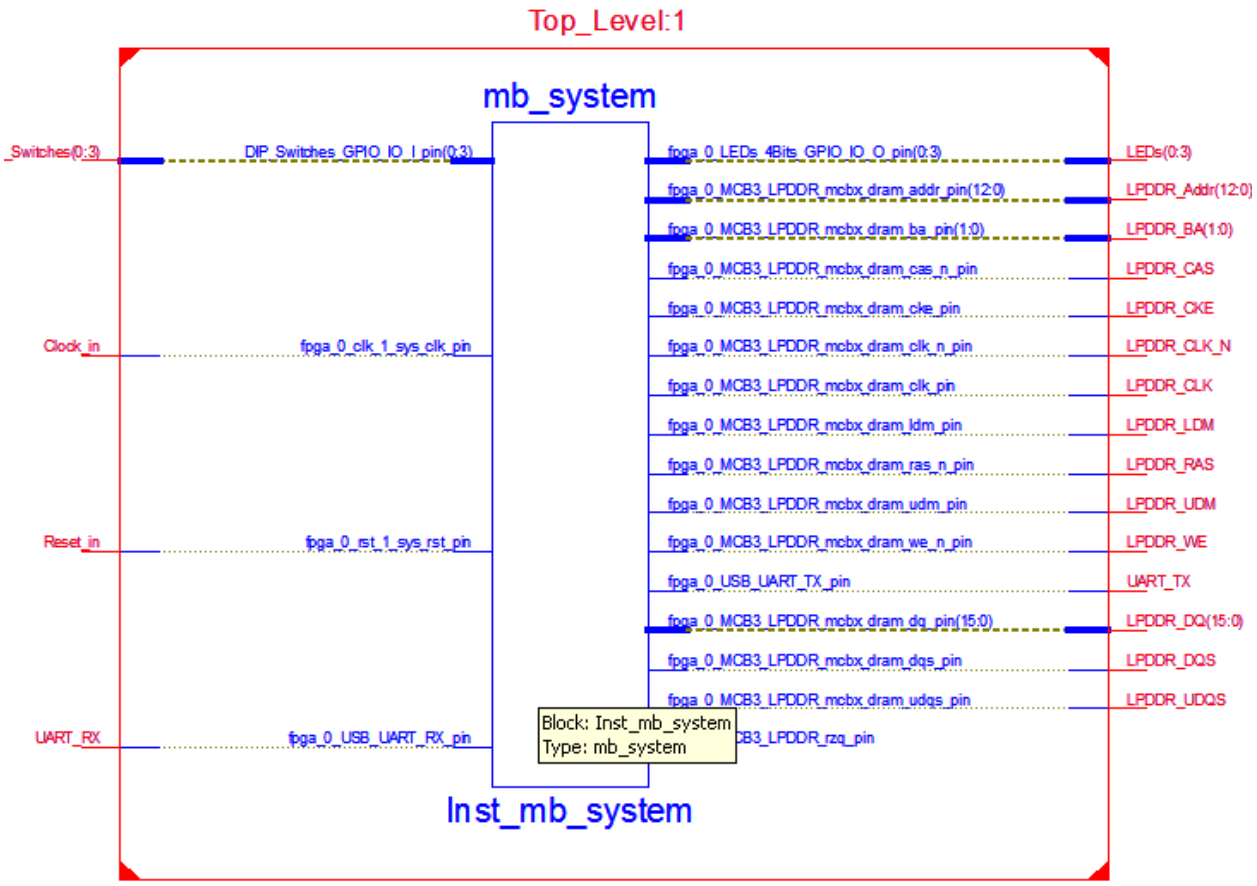

Figure 1 - MicroBlaze Processor as a Submodule in ISE

Objectives

This Tutorial demonstrates how to add a MicroBlaze embedded system as a sub-module into the ISE environment. The lab will show:

- How to instantiate an existing XPS project as a sub-module into Project Navigator
- How to modify the top level example instantiation file
- How to load the software executable into internal BRAMs

Requirements

The following items are required for proper completion of this tutorial.

- Completion of the Adding Custom IP to an Embedded System Tutorial

Software

The following software setup was used to test this reference design:

- WindowsXP 32-bit Service Pack 2
- Xilinx ISE WebPack with the SDK add-on or ISE Embedded Edition version 12.4
- Installed Digilent Adept and Xilinx 3rd-party USB Cable driver
- Installed Silicon Labs CP210x USB-to-UART Bridge Driver
- Installation of the Spartan-6 LX9 MicroBoard XBD files

Hardware

The hardware setup used by this reference design includes:

- Computer with a minimum of 300-900 MB (depending on O/S) to complete an XC6SLX9 design[1]
- Avnet Spartan-6 LX9 MicroBoard Kit
  - Avnet Spartan-6 LX9 MicroBoard
  - USB Extension cable (if necessary)
  - USB A-to-MicroB cable

Setup

- Install ISE Embedded Edition or ISE WebPack with the EDK add-on.
- Install Digilent Adept and Xilinx 3rd-party USB Cable driver (see Installation Guide on the DRC)

Recommended Reading

Available from Avnet: *http://em.avnet.com/s6microboard*

- The hardware used on the Spartan-6 LX9 MicroBoard is described in detail in Avnet document, *Spartan-6 LX9 MicroBoard User Guide*.
- An overview of the configuration options available on the Spartan-6 LX9 MicroBoard, as well as Digilent driver installation instructions can be found in the Avnet document, *Spartan-6 LX9 MicroBoard Configuration Guide.*
- Instructions on installing the Silicon Labs CP210x USB-to-UART drivers can be found in the Avnet document, *Silicon Labs CP210x USB-to-UART Setup Guide*.

Available from Xilinx: *http://www.xilinx.com/support/documentation/spartan-6.htm*

- Details on the Spartan-6 FPGA family are included in the following Xilinx documents:
  - *Spartan-6 Family Overview (DS160)*
  - *Spartan-6 FPGA Data Sheet (DS162)*
  - *Spartan-6 FPGA Configuration User Guide (UG380)*
  - *Platform Studio Help (available in tool menu)*
  - *Platform Studio SDK Help (available in tool menu)*
  - *MicroBlaze Reference Guide v.12.4 (UG081)*
  - *Embedded System Tools Reference Manual v.12.4 (UG111)*

[1] Refer to www.xilinx.com/ise/products/memory.htm

I. Embedded System sub-module in ISE

The embedded system is currently the top level module in ISE. However the majority of FPGA designs include some other non-embedded logic. We will make the transition from a top-module flow to a sub-module flow.

1) Start ISE Project Navigator and open the **Tutorial_01** project.

2) Select the **mb_system** module and expand the **Design Utilities** section in the **Processes** window.

3) Double-click on **View HDL Instantiation Template**.

4) The file created provides the component definition for the embedded system as well as the instantiation template. We will create a top level file and add the embedded system as a sub module.

5) Go to **Project > New Source…** Select **VHDL Module** and name it **Top_Level**. Click on **Next**.

6) We will need to add all the ports for the UART, clock, reset, LEDs, DIP switches, and DDR. Add the ports needed for the board as shown below. Click **Next** then **Finish**.

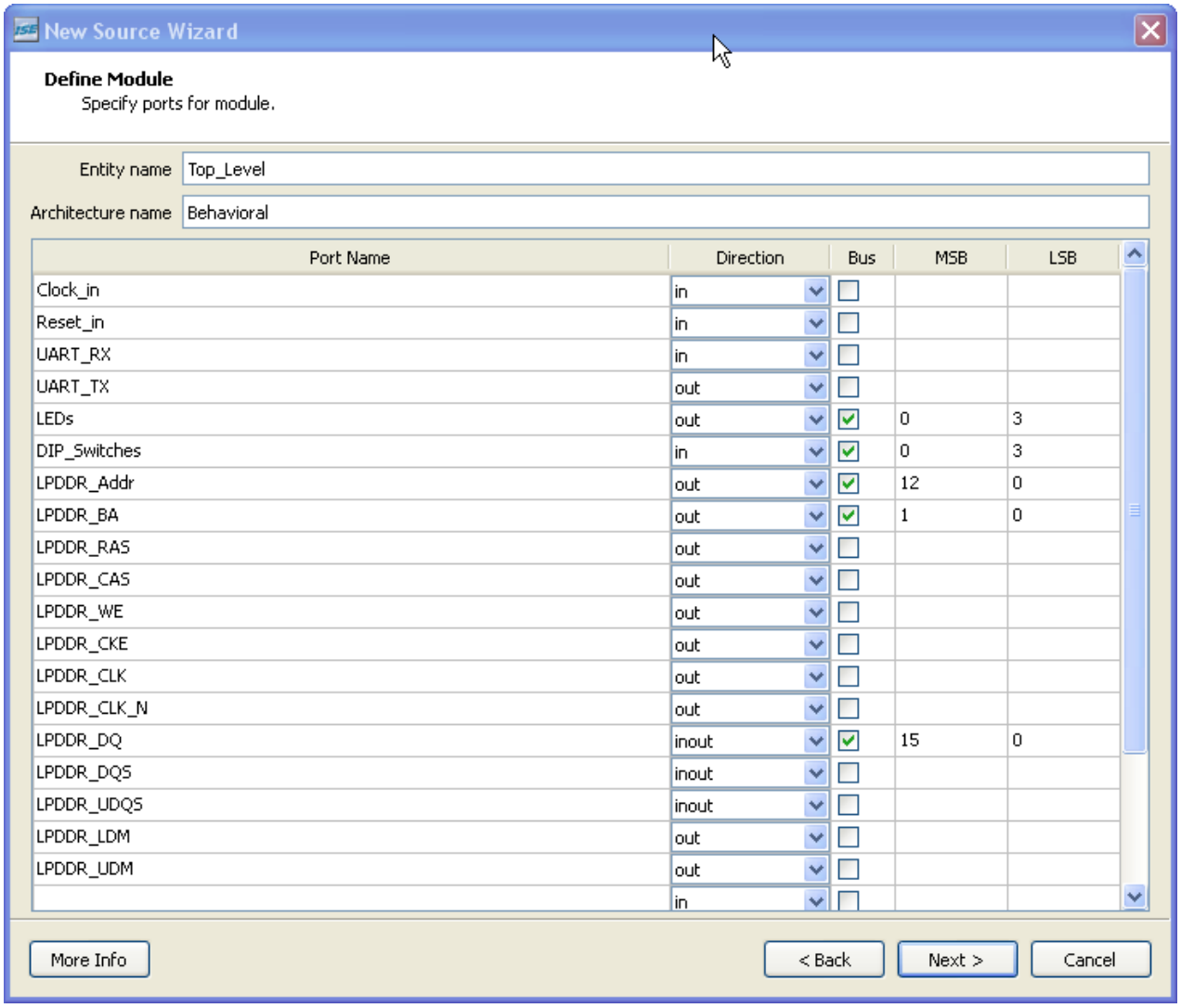

Figure 2 - New Source Wizard

7) To match the embedded system, in the new file, change the vector range for **LEDs** and **DIP_Switches** from *(*0 downto 3*)* to **(0 to 3):**

```
entity Top_Level is
    Port ( Clock_in : in  STD_LOGIC;
           Reset_in : in  STD_LOGIC;
           UART_RX : in  STD_LOGIC;
           UART_TX : out  STD_LOGIC;
           LEDs : out  STD_LOGIC_VECTOR (0 to 3);
           DIP_Switches : in  STD_LOGIC_VECTOR (0 to 3);
           LPDDR_Addr : out  STD_LOGIC_VECTOR (12 downto 0);
           LPDDR_BA : out  STD_LOGIC_VECTOR (1 downto 0);
           LPDDR_RAS : out  STD_LOGIC;
           LPDDR_CAS : out  STD_LOGIC;
           LPDDR_WE : out  STD_LOGIC;
           LPDDR_CKE : out  STD_LOGIC;
           LPDDR_CLK : out  STD_LOGIC;
           LPDDR_CLK_N : out  STD_LOGIC;
           LPDDR_DQ : inout  STD_LOGIC_VECTOR (15 downto 0);
           LPDDR_DQS : inout  STD_LOGIC;
           LPDDR_UDQS : inout  STD_LOGIC;
           LPDDR_LDM : out  STD_LOGIC;
```

Figure 3 - Change LEDs and DIP_Switches Port Declaration

8) Copy the **mb_system** component declaration and the two attribute lines from the **mb_sytem.vhi** file to the **Top_Level.vhd** file between the **architecture** and the **begin** line.

9) Copy the **mb_system** component instantiation from the **mb_sytem.vhi** file to the **Top_Level.vhd** file between the **begin** and the **end Behavioral** line.

10) Map the top level ports to the embedded system ports:

```
Inst_mb_system: mb_system PORT MAP(
        fpga_0_USB_UART_RX_pin => UART_RX,
        fpga_0_USB_UART_TX_pin => UART_TX,
        fpga_0_LEDs_4Bits_GPIO_IO_O_pin => LEDs,
        fpga_0_MCB3_LPDDR_mcbx_dram_addr_pin => LPDDR_Addr,
        fpga_0_MCB3_LPDDR_mcbx_dram_ba_pin => LPDDR_BA,
        fpga_0_MCB3_LPDDR_mcbx_dram_ras_n_pin => LPDDR_RAS,
        fpga_0_MCB3_LPDDR_mcbx_dram_cas_n_pin => LPDDR_CAS,
        fpga_0_MCB3_LPDDR_mcbx_dram_we_n_pin => LPDDR_WE,
        fpga_0_MCB3_LPDDR_mcbx_dram_cke_pin => LPDDR_CKE,
        fpga_0_MCB3_LPDDR_mcbx_dram_clk_pin => LPDDR_CLK,
        fpga_0_MCB3_LPDDR_mcbx_dram_clk_n_pin => LPDDR_CLK_N,
        fpga_0_MCB3_LPDDR_mcbx_dram_dq_pin => LPDDR_DQ,
        fpga_0_MCB3_LPDDR_mcbx_dram_dqs_pin => LPDDR_DQS,
        fpga_0_MCB3_LPDDR_mcbx_dram_udqs_pin => LPDDR_UDQS,
        fpga_0_MCB3_LPDDR_mcbx_dram_udm_pin => LPDDR_UDM,
        fpga_0_MCB3_LPDDR_mcbx_dram_ldm_pin => LPDDR_LDM,
        fpga_0_clk_1_sys_clk_pin => Clock_in,
        fpga_0_rst_1_sys_rst_pin => Reset_in,
        DIP_Switches_GPIO_IO_I_pin => DIP_Switches
        );
```

11) Save the **Top_Level.vhd** file.

12) Since the names of the top level ports have changed, the constraint file also needs to be modified.

13) Expand the project files in the **Hierarchy** window. Select **mb_system.ucf** and select **Edit Constraints (Text)** in the **User Constraints** selection.

14) Name the nets in the UCF file based on the new port names from the **Top_Level.vhd** file.

```
fpga_0_clk_1_sys_clk_pin becomes Clock_in
fpga_0_rst_1_sys_rst_pin becomes Reset_in
fpga_0_USB_UART_RX_pin becomes UART_RX
fpga_0_USB_UART_TX_pin becomes UART_TX
DIP_Switches_GPIO_IO_I_pin becomes DIP_Switches
fpga_0_LEDs_4Bits_GPIO_IO_O_pin becomes LEDs
```

Note: The LPDDR connections are assigned inside the MicroBlaze system. Since Spartan-6 FPGA's have a hard-core MCB, the I/O locations for each are fixed. Thus, I/O constraints for the LPDDR connections do not need to be made in this UCF.

15) Save and close the constraints file.

16) Select the top level module in the **Hierarchy** window.

17) Cleanup the project files. Go to **Project > Cleanup Project Files**. Click **OK**.

18) Double-click on **Generate Programming File** in the **Processes** window to update the design.

II. Integrate the Software Application

While the embedded system is now a sub-module, no changes were made to the hardware. We can use the Tutorial_Test application from the previous tutorial to test the new design.
We have been using SDK to run our application but for applications using internal BRAM memory it is possible to update the Project Navigator bitstream with the software executable. The new bitstream can then be downloaded using iMPACT.

1) In Project Navigator, double-click on the **mb_system** module to start XPS.

2) Select the **Applications** tab on the left side.

3) Double-click on **Add Software Application Project…**

4) Enter **Tutorial_Test** for the project name. Click on the **Project is an ELF-only Project** box. Browse to the **\Tutorial_01\WorkSpace\Tutorial_Test\Debug** directory and select **Tutorial_Test.elf**. Click **OK**.

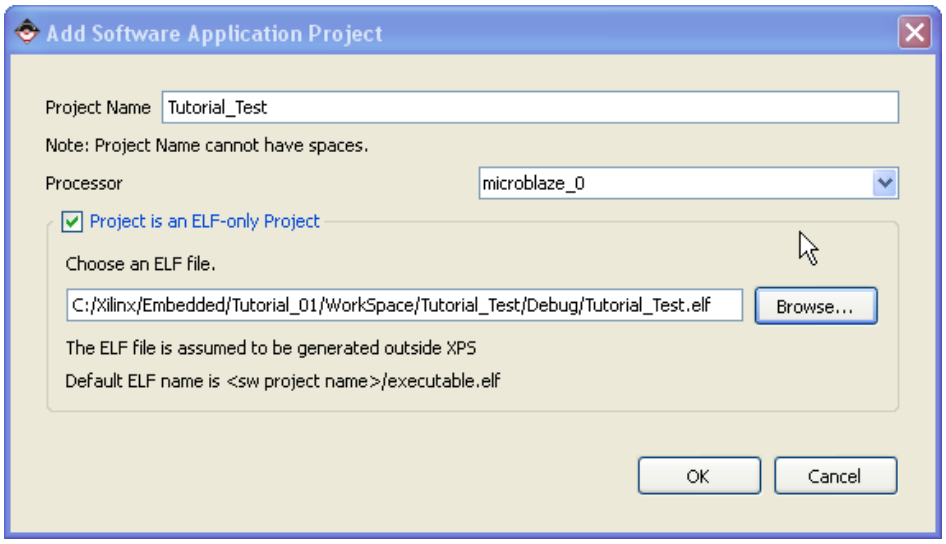

Figure 4 - Add Software Application

5) Right-click on the **Tutorial_Test** application and select **Mark to Initialize BRAMs**. The application will now be selected for the Update Bistream process.

6) Close **XPS**.

7) In Project Navigator, select the top level module.

8) Double-click on **Update Bitstream with Processor Data.**

III. Test the Generated System with the Test Application

Instead of using SDK to run the code we will directly program the FPGA using Project Navigator and iMPACT

.

1) Plug the MicroBoard board into the PC.
2) Plug the USB UART cable between the MicroBoard and the PC.
3) Start HyperTerminal and create a new connection for the USB-UART COM port at 9600 Baud.
4) In Project Navigator, double-click on **Configure Target Device**. Click **OK**.
5) In iMPACT, double-click on **Boundary Scan**.
6) Go to **Output > Cable Setup**
7) In the **Cable Plug-in**, check the box then type **digilent_plugin**. Click **OK**

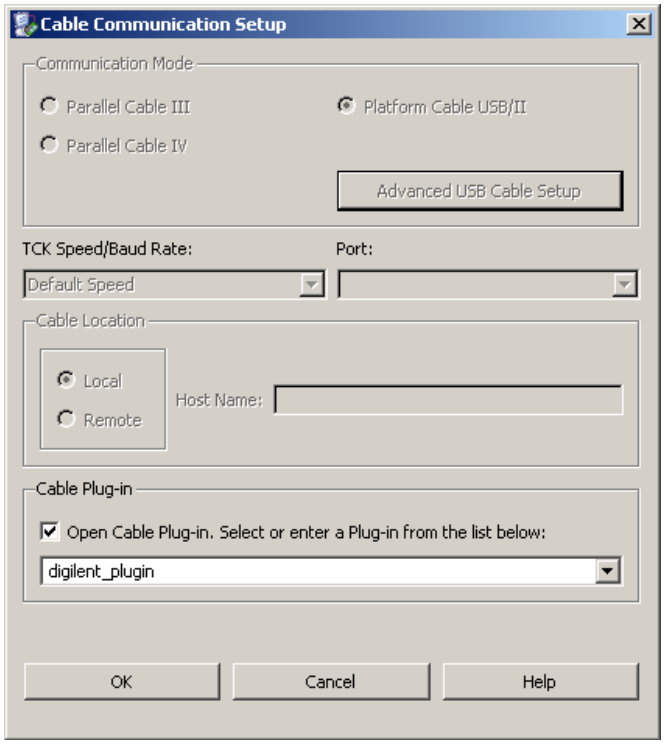

Figure 5 - JTAG Cable Setup

8) Right-click in the **Boundary Scan** window and select **Initialize Chain**.
9) Click **Yes** to assign a file.
10) Browse to the ISE project location, **C:\Xilinx\Embedded\Tutorial_01** and select **top_level_download.bit**. Click **Open**.
11) The top_level.bit file is the bitstream without the application pre-loaded into the BRAMs.

12) Click **No** for the **Attach SPI or BPI PROM** window.

13) Click **OK**.

14) Double-click on **Program** in the iMPACT Processes window.

15) Verify that the Console window is showing the print statement.

16) Carefully modify the DIP switches positions to change the intensity of the LEDs.

17) Close iMPACT.

18) The top_level_download.bit file could be used to create a PROM file for the SPI Flash to configure the FPGA at power-on.

That concludes this tutorial. We now have instantiated the MicroBlaze processor as a submodule of the ISE project.

Getting Help and Support

Evaluation Kit home page with Documentation and Reference Designs

http://em.avnet.com/s6microboard

Avnet Spartan-6 LX9 MicroBoard forum:

http://community.em.avnet.com/t5/Spartan-6-LX9-MicroBoard/bd-p/Spartan-6LX9MicroBoard

For Xilinx technical support, you may contact your local Avnet/Silica FAE or Xilinx Online Technical Support at www.support.xilinx.com. On this site you will also find the following resources for assistance:

- Software, IP, and Documentation Updates
- Access to Technical Support Web Tools
- Searchable Answer Database with Over 4,000 Solutions
- User Forums
- Training - Select instructor-led classes and recorded e-learning options

Contact Avnet Support for any questions regarding the Spartan-6 LX9 MicroBoard reference designs, kit hardware, or if you are interested in designing any of the kit devices into your next design.

- http://www.em.avnet.com/techsupport

You can also contact your local Avnet/Silica FAE.