WIRELESS OPEN-ACCESS
RESEARCH PLATFORM

Lab 2: Introduction to sysgen2opb

Charles Camp

Rice University CMC Lab

Document Revision 1.2

March 22, 2007

1 Introduction

In this lab, you will use Xilinx Platform Studio to specify and implement a base system on a WARP FPGA board. You will then implement a custom peripheral core using System Generator and import that core back into your base system. You will also write and compile software to control the core.

2 Specifying Your Base System

The first step in this lab exercise is to specify options for the base system that you will use for the remainder of this laboratory exercise.

1. Open Xilinx Platform Studio version 8.2i.
2. Choose the Base System Builder Wizard option.
3. Name the project file `C:\workshop\Lab2\xps\system.xmp`.
4. Choose to create a new design.
5. Choose Rice University WARP FPGA and Radio Boards.
6. Choose the PowerPC processor.
7. Change bus clock frequency to 50 MHz. Select 64 KBytes of on-chip data memory. Select 128 KBytes of on-chip instruction memory.
8. Deselect all additional IO interfaces EXCEPT RS232. Set RS232 parameters to 57600/8/NONE.
9. On the 'Add Additional Peripherals' screen, click 'Remove' to remove the PLB BRAM core from your base system.
10. Select the Memory Test sample application. Do not select the Peripheral Selftest application.
11. When presented with a system summary, click Generate.
12. Start Using Platform Studio!

3 Building Your Base System

1. View bus connectivity in the System Assembly View.
2. Click the Addresses radio button at the top of the System Assembly View.
3. Generate addresses for all system peripherals.
4. From the Hardware menu at the top of the XPS window, select Generate Bitstream.
5. This will compile the hardware component of your base system.
6. After several minutes, the compilation will finish (watch for `Bitstream generation is complete!` in the console window).

4 Testing Your Base System

1. From the Device Configuration menu at the top of the XPS window, select Update Bitstream.
2. This will compile the software component of your base system and insert the initial RAM values into the hardware configuration file.
3. Make sure that your WARP board has been powered on.
4. Open the Tera Term Pro software and select serial mode.
5. From the setup menu, verify that the serial port has been configured for COM1/57600/8/none/1/none.
6. From the Device Configuration menu at the top of the XPS window, select Download Bitstream.
7. The Tera Term window should display the result of your memory test. It will print something like this:

```
-- Entering main() --

-- Exiting main() --
```

8. Close Xilinx Platform Studio.

5 Converting a System Generator Model to an OPB Peripheral

1. Open MATLAB 7.1.
2. From the Matlab command window, change the working directory to `C:\workshop\Lab2\sysgen\`.
3. Choose File→ Open, and select the file `adder.mdl`.
4. This Simulink file is a System Generator model which implements a simple, unsigned, fixed-point adder with two 32-bit inputs and a 32-bit output.
5. Click the triangular play icon at the top of the model window.
6. Double-click the Simulink Scope block in the model.
7. Verify that the adder output waveform switches to value 12 at time $t = 1$.
8. Close the Scope window, then close the `adder` model.
9. In the Matlab command window, type `sysgen2opb('adder')`. This will prepare your model for conversion to an OPB-compliant peripheral.
10. Re-open the `adder` model.
11. Double click on the System Generator icon within the model. Verify that the System Generator paramters match those in Figure 1.
12. Click the Settings button, then the Folder icon, then navigate to `C:\workshop\Lab2\xps\system.xmp`. Click Open, then OK. This step automates the export of your peripheral to an existing XPS project.
13. Click Generate, and wait for the Generation Completed message.

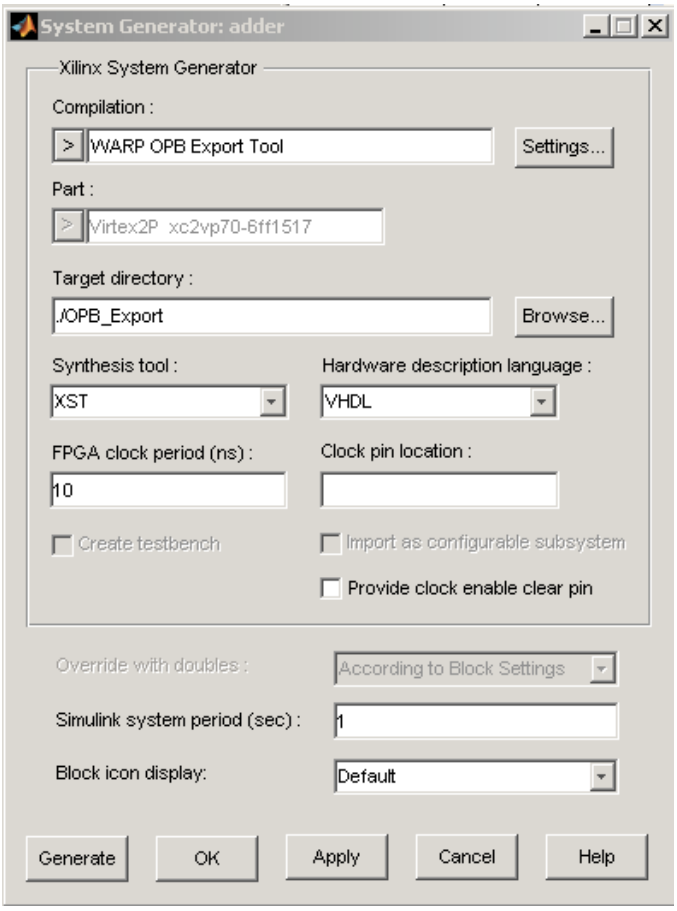

Figure 1: System Generator parameters for OPB core generation

6 Importing the OPB Peripheral

1. Open Xilinx Platform Studio and reopen the previous project `C:\workshop\Lab2\xps\system.xmp`.
2. In the IP Catalog tab of the Project Information Area, expand the Project Repository entry.
3. Right click on `adder_opbw` and select Add IP.
4. From the Bus Interface view of the System Assembly tab, expand the entry for `adder_opbw`. Connect the core to the OPB by clicking the hollow green circle on the left.
5. Repeat steps 3 and 4 to add two more adder cores (three total). When you've added three cores, your Bus Interface view should look like Figure 2.

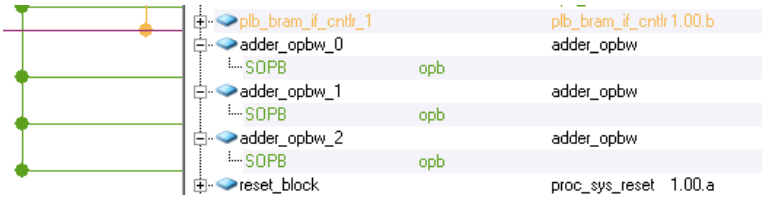

Figure 2: Adder cores attached to the OPB

6. Switch to the Addresses view and click Generate addresses. This constructs the hardware project's memory map.
7. From the 'Device Configuration' menu at the top of the XPS window, select 'Update Bitstream'.
8. This will compile the hardware component of your base system, including the three adders, and compile the board support libraries and peripheral drivers.
9. After several minutes, the compilation will finish (watch for `Memory Initialization completed successfully.` in the console window).

7 Using the OPB Peripheral

10. View file `adder_gateways.h` in directory `C:\workshop\Lab2\xps\drivers\adder_opbw\src`. This file defines macros that simplify the task of accessing registers in the adder core. In particular, we'll need the following macros from this file:

```
#define adder_WriteReg_Op1(BaseAddress, Value) \
  XIo_Out32((BaseAddress) + (adder_Op1_OFFSET), (Xuint32)(Value))

#define adder_WriteReg_Op2(BaseAddress, Value) \
  XIo_Out32((BaseAddress) + (adder_Op2_OFFSET), (Xuint32)(Value))

#define adder_ReadReg_Sum(BaseAddress) \
  XIo_In32((BaseAddress) + (adder_Sum_OFFSET))
```

11. View file `C:\workshop\Lab2\xps\ppc405_0\include\xparameters.h`. This file defines key parameters, including address locations, for each core in the system. We need this file to locate the base addresses of the three instances of our adder core. Look for lines like these (your address may be different):

```
#define XPAR_ADDER_OPBW_0_BASEADDR 0x7FC40000

#define XPAR_ADDER_OPBW_1_BASEADDR 0x7FC20000

#define XPAR_ADDER_OPBW_2_BASEADDR 0x7FC00000
```

12. Modify file `C:\workshop\Lab2\xps\TestApp_Memory\src\TestApp_Memory.c` to utilize the first adder in your system. The code below is a good starting point:

```
#include "adder_gateways.h"
#include "xparameters.h"

int main() {
  unsigned int result;
  unsigned int operand_a = 3;
  unsigned int operand_b = 5;

  adder_WriteReg_Op1(XPAR_ADDER_OPBW_0_BASEADDR, operand_a);

  adder_WriteReg_Op2(XPAR_ADDER_OPBW_0_BASEADDR, operand_b);

  result = adder_ReadReg_Sum(XPAR_ADDER_OPBW_0_BASEADDR);

  xil_printf("Adder Says : %d + %d = %d\r\n",operand_a,operand_b,result);

  return 0;

}
```

13. From the Device Configuration menu at the top of the XPS window, select Update Bitstream.
14. From the Device Configuration menu at the top of the XPS window, select Download Bitstream.
15. View results in Tera Term.

16. Try modifying your code to utilize all three adder cores. You should only have to modify the base address values in your C code to target different cores.