Synthesis and Hardware Implementation

1 Week

Motivation and introduction

In this session we synthesis our basis CPU with Xilinx ISE and execute a test application on real hardware. The synthesis reports tell us how much area and power is consumed by our CPU and what is the critical path of our design. In this session, we will compile a C-code application direct its output the LCD and UART with the help of some predefined libraries. This session also introduces about different peripheral where we forward our text/data, and how different libraries are used for different peripherals. For every part, that starts like "a)", "b)" … you have to mail the answers and asked files/tables to **sajjad.hussain@kit.edu** and use the topic "asipXX-Session3", with XX replaced by your group number.

Exercises

1. Preparing your project

1.1. You can use the same project as in the last session, and just create a separate application subdirectory the application. You can start a fresh project as in the Session 1, but this would be time consuming.

1.2. For the C application, you have to create two subdirectories in the "*Application*" directory e.g. "*Hello_SW*" and "*Hello_HW*". Copy your application from "*/home/asip00/Sessions/Session4/app.c*" to these. This is a simple example to direct a text to some peripheral devices like LCD or UART. "*Hello_SW*" is aimed for dlxsim and ModelSim simulations and "*Hello_HW*" is aimed for real hardware implementation.

1.3. Copy a "*Makefile*" file from the "*TestPrint*" application subdirectory to each application subdirectory.

1.4. Set proper parameters and settings in "*env_settings*.

1.5. For these exercises, we will be using pipeline forwarding (-pf1) option which is the default one.

1.6. Make sure that you already have VHDL files and GNU tools in your project's meister directory.

2. Compiling and ModelSim Simulation

2.1. First, you have to compile the application using gcc compiler to compare with the later results from dlxsim and ModelSim. For *gcc* you can forward the printed output to a file, e.g. "*a.out* > *output_gcc.txt*" ('*a.out*' is the default name of the binary that is created when you compile "*gcc arrayloop.c*" while '*output_gcc.txt*' then contains the printed array). To compile with GCC, comment the line "*#define ASIP*".

2.2. However, for compiling it, you first need to provide the required libraries from /home/asip00/epp/StdLib to your respective application, i.e. copy "*lib_lcd_dlxsim.c*", "*lib_uart.c*", "*loadStoreByte.c*", "*string.c*" and respective header files to "Hello_SW" directory. Also, copy "*lib_lcd_320.c*", "*lib_uart.c*", "*loadStoreByte.c*", "*string.c*" and respective header files to "Hello_HW" directory.

2.3. Go to your application subdirectory "Hello_SW", and type "***make clean***" clean this directory it there are previously generated files.

2.4. In the directory "Hello_SW", compile the C application using "***make sim***". A directory "***BUILD_SIM***" is created which contains different temporary files and a .dlxsim file to be simulated in dlxsim. In this directory, the files "***TestData.IM***" and "***TestData.DM***"are the file used during the ModelSim simulation.

2.5. Simulate your application in dlxsim simulator using "***make dlxsim***", just to verify the functionality. For dlxsim you can forward the LCD/UART output to a file, using the "-lf" and "-uf" parameters respectively, e.g. "make dlxsim DLXSIM_PARAM="-da0 –pf1 -lflcd.out -ufuart.out" writes output to the file "lcd.out" and "uart.out" in the application directory.

2.6. In your project directory, go to the "ModelSim" directory and start the ModelSim using "vsim". You can use the previous ModelSim project and simulate the application. Remember to generate VCD files required for power estimation while doing ModelSim simulation.

2.7. After compiling, simulate the application in dlxsim and ModelSim and compare whether the printed results are the same as expected. The dlxsim and ModelSim will print text to a virtual LCD/UART. While ModelSim automatically writes to the file "lcd.out" and "uart.out".

a) How many cycles are required to execute this program DLXsim and ModelSim?

3. Xilinx ISE Framework for Hardware Implementation

3.1. Go to your application subdirectory "Hello_HW", and type "***make clean***" clean this directory it there are previously generated files.

3.2. In the directory "Hello_HW", compile the C application using "***make sim***".

3.3. Go to the project directory and type "ise &" to start Xilinx ISE.

3.4. Create new project using File Menu > New Project with following project settings:

Project Name: ISE_Framework
Project Path: PATH_TO_YOUR_PROJECT/ ISE_Framework
Device Family: Virtex5
Device: xc5vlx110t
Package: ff1136

3.5. Add the design and framework files by selecting "Project Menu > Add Copy of Sources" then brows to:

3.5.1. "PATH_TO_YOUR_PROJECT/ ***ISE_Framework***" and select all the files

3.5.2. "PATH_TO_YOUR_PROJECT/ ***ISE_Framework/IP-Cores***" and select all the files

3.5.3. "PATH_TO_YOUR_PROJECT/ ***meister/browstd32.syn***" and select all the files

3.6. Select top level modules "**dlx_toplevel**", and now you can synthesize, implement and generate programming file for the design using the following respectively:

3.6.1. Processes Menu > Synthesize XST

3.6.2. Processes Menu > Implement Design

3.6.3. Processes Menu > Generate Programming File

3.7. Once the design is implemented you can see different reports using:

3.7.1. Processes Menu > Place & Route > Generate Post Place & Route Static Timing > Detailed Reports > Place and Route Report

3.7.2. Processes Menu > Place & Route > Generate Post Place & Route Static Timing > Detailed Reports > Post PAR Static Timing Report

3.7.3. Processes Menu > Place & Route > Analyze Post Place & Route Static Timing > Timing Constraints

3.8. In the project directory and type "hterm &" to start HyperTerminal to see the UART output if there is any output. and adjust its settings like: Baud rate=115200, Stop bit=1, Data bits=8, Parity=None, COM Port=ttyUSB0 (for example), Newline at=CR+LF,

3.9. In the application subdirectory and type "make fpga", it will combine the generate DM/IM file with your ISE generated bitstream. Finally, a new bitstream file containing your hardware CPU along with corresponding IM/DM files of your application will be generated in the folder "BUILD_FPGA". This bitstream will be used to configure the FPGA.

3.10. For hardware implementation you need to connect to i80labpc10 only. Connect your FPGA to PC the i80labpc10, power the board. In the application subdirectory type "make upload": to upload the existing bitstream to the FPGA

4. Xilinx ISE Framework for Benchmarking

4.1. To accurately measure the critical path and area of the ASIPmeister CPU, you can use ISE_Benchmark folder instead of ISE_Framework folder.

4.2. Go to the project directory and type "ise &" to start Xilinx ISE.

4.3. Create new project using File Menu > New Project with following project settings:

Project Name: ISE_BenchMark
Project Path: PATH_TO_YOUR_PROJECT/ ISE_ BenchMark
Device Family: Virtex5
Device: xc5vlx110t
Package: ff1136

4.4. Add the design and framework files by selecting "Project Menu > Add Copy of Sources" then brows to:

4.4.1. "PATH_TO_YOUR_PROJECT/ ISE_ BenchMark" and select all the files

4.4.2. "PATH_TO_YOUR_PROJECT/ ***meister/ browstd32.syn***" and select all the files

4.5. Now you can synthesize, implement and generate programming file for the design as before.

4.6. Once the design is implemented you can see different reports as before.

b) Reports: P&R Report > Check for "Device Utilization Summary" to see #Slices and #LUT consumed.
c) Post PAR Static Timing Report: Check for "Timing Summary" to see the minimum period and maximum frequency supported by the processor architecture.

5. Xilinx ISE Framework for XPower Power Estimation

5.1. To accurately measure the power consumption of the ASIPmeister CPU, you can create another folder ISE_XPower.

5.2. Go to the project directory and type “ise &” to start Xilinx ISE.

5.3. Create new project using File Menu > New Project with following project settings:

Project Name: ISE_XPower
Project Path: PATH_TO_YOUR_PROJECT/ ISE_ XPower
Device Family: Virtex5
Device: xc5vlx110t
Package: ff1136

5.4. Add only design files by selecting “Project Menu > Add Copy of Sources” then brows to “PATH_TO_YOUR_PROJECT/ ***browstd32.syn***” and select all the files.

5.5. Now you can synthesize and implement the design as before.

5.6. Once the design is implemented you can open XPower tool using Processes Menu > Place & Route > Analyze Power Distribution (xPower Analyzer)

5.7. Then in XPower Tool, select “File Menu > Open Design” and set the properties as follows:

5.7.1. Design File: PATH_TO_YOUR_PROJECT/ISE_ XPower/ BrownieSTD32.ncd

5.7.2. Physical Constraint File: PATH_TO_YOUR_PROJECT/ ISE_ XPower/ BrownieSTD32.pcf

5.7.3. Simulation Activity File: PATH_TO_YOUR_PROJECT/test.vcd

5.8. After analyzing the activity file, the CPU power is estimated. You can see total and dynamic power of the FPGA. In addition, you can confirm that the VCD file is loaded properly by verify the clock value in XPower.

d) What is the total power consumed? What is the distribution of power i.e., dynamic and static power?
e) Why static/quiescent/leakage power is so high?
f) What is the power distribution among On-Chip **1**) Clocks, **2**) Logic, **3**) Signals, **4**) BRAMS, **5**) IOs?

Next Session: Adding Custom Instructions
Readings for the next session:
Laboratory Chapters 8.2.3, 3.2.2, 3.2.3, 4.4,
ASIPmeister Tutorial
ASIPmeister User Manual