ModelSim Simulation

1 Week

Motivation and introduction

In this session, we will compile a C-code application and simulate the result in *ModelSim* and *Dlxsim*. The applications can be assembled or compiled using a compiler that is already generated by ASIPmeister in the previous session. ModelSim simulates your code binaries using the VHDL files generated from the ASIPmeister. While dlxsim only simulates the instruction one by one and it does not care about the hardware implementation of the instructions. 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 for 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 subdirectory in the "*Application*" directory (e.g. *LoopExampleC*), and copy your application from "*/home/asip00/Sessions/Session3/6_for.c*" to here.
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 Simulating the Application

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

2.2. 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.

2.3. In folder BUILD_SIM, look at the "*6_for.s*" which is generated. Another file "*startup.s*" is used along with the generated "*6_for.s*" to generate TestData.IM/DM files. Just understand and remember the structure of "6_for.s" files if you have to write your own .s file, and how it is being executed along with "*startup.s*".

2.4. Simulate your application in dlxsim simulator using "***make dlxsim***", just to verify the functionality.

2.5. In your project directory, go to the “ModelSim” directory and start the ModelSim using “***vsim***”

2.6. If ModelSim asks for “modelsim.ini” choose the default one like “/Software/ModelSim/ModelSim_6.6d/modeltech/modelsim.ini”

2.7. Open File Menu > New > Project and enter a project name (e.g. browstd32) and change the project location to the ModelSim directory in your project directory. Confirm the dialog with the OK button.

2.8. Choose “Add Existing File” button and browse to the meister/dlx_basis.syn directory of your ASIP Meister project and select all the VHDL files for synthesis.

2.9. Again, choose “Add Existing File” button and add the testbench files: tb_browstd32.vhd, MemoryMapperTypes.vhd, MemoryMapper.vhd, and Helper.vhd from the ModelSim directory of your current project.

2.10. [Optional] Configure the CPU Frequency for which you want to simulate your CPU, default is 50 MHz. Open the ModelSim testbench (“tb_ browstd32.vhd”), search for CLK _PERIOD, and change the value accordingly in “ns”.

2.11. Compile the project using Compile Menu > Compile Order > Auto Generate. Every file should have a green mark behind its name, showing that the compilation was successful.

2.12. Run the simulation using Simulate Menu > Start Simulation. Open the work library, mark the entry “***cfg***” (that is the VHDL configuration for the testbench) in the list and press OK. That will start the simulation and you will get another two tabs attached to the Workspace window (sim / Files).

2.13. [Optional] To load some predefined simulation settings choose Tools Menu > Tcl > Execute Macro and select the “wave_vhdl.do” file in your ModelSim directory and press OK to load it. The wave-window is filled with certain signals that are useful to evaluate the simulation of the program execution on the processor.

2.14. [Optional] If you want to dump VCD file of yor design for power estimation, you can enter following commands in ModelSim command prompt:

```
VSIM > vsim -t 1ns work.cfg
VSIM > vcd file test.vcd
VSIM > vcd add -r test/dut/*
```

2.15. Press the button “***Run all***” to run the simulation until it aborts. At the end of a simulation the message “Failure: Simulation End” is printed to show successful end of simulation. At the simulation end, the file “***TestData.OUT***” is created in your ModelSim directory. It contains the content of the simulated memory after the CPU finished working. Therefore, if your algorithm is storing the result in the memory you can find the values here.

a) How many cycles are required to execute this program DLXsim and ModelSim?
b) What are the contents of TestData.OUT? Are these correct? First value is the stack value; next 10 words belongs to array A, then 10 words belongs to array B, and C.
c) In the ModelSim waveform window, what is the starting address of PC after the reset? Moreover, after how many cycles your “**main**” function is started? In the waveform, look at the PC and IR values.

2.16. The default GCC compiler optimization is –O0. Try different optimization levels with dlxsim and ModelSim using e.g. "*make dlxsim GCC_PARAM=-O1*" or using "*make sim GCC_PARAM=-O1*".

2.17. Repeat this benchmarking for all compiler optimization-levels like O0, O1, O2, O3 and O4 for both dlxsim and ModelSim.

a) Does the application is executed successfully using different optimization levels? If yes, please fill the following benchmark table. These optimizations have distinct effects on the size of the code.

Optimization Level	**Executed?** [Yes/No]	**Cycle count** ModelSim	**Cycle count** dlxsim
-O0 (default)			
-O1			
-O2			
-O3			
-O4			

Next Session: Synthesis and Hardware Implementation
Readings for the next session: Laboratory Chapters 6