GCD: S1H — A Uni-Processor Hw/Sw Solution

(Laboratory Session 5, EDAN15)

Flavius.Gruian@cs.lth.se

March 8, 2013

1 Introduction

In this laboratory session you will have to integrate the hardware developed in the previous session into a larger system, similar to one you first built. Furthermore, you should write software containing the necessary communication and computations, to obtain a functioning hardware/software solution for the *gcd* of N-numbers problem. In addition you will also have to evaluate your design and report the device utilization, time (clock cycles), power and energy consumption, in a similar way you did for labs 2 and 3. The development and testing will be carried out using the *EDK 14.x* and the *Digilent Nexys-3* evaluation board.

2 System Architecture

The architecture for this laboratory session is centered on a MicroBlaze cooperating with your *gcd* accelerator core. Starting from a typical uni-processor system (MicroBlaze, LMB controllers, local BRAM, system bus, Uart, and timer) you should connect your hardware into the system and get it running together with the processor.

XPS supports custom IP cores, granted they are specified in a standard format. To be able to rapidly use your core in the XPS, an FSL-based template is available to you on the course web-page (`fsl_hw_template.zip`). Unpack the archive inside your XPS `project/pcores` directory. It should now contain a new directory called `fsl_hwa_v1_00_a`.
Overwrite the `fsl_hwa_v1_00_a/hdl/vhdl/user_logic.vhd` with your own

`user_logic.vhd` to add the functionality you implemented in the previous laboratory session. After selecting in XPS **Project → Rescan User Repositories**, the XPS should contain now a new core *fsl_hwa* among the others, ready to place in your design. You find it in the *IP Catalog* in the left XPS frame, under *Project Repository*, *Project Local pcores/USER*. Also add a duplex FSL (meaning one FSL processor port, and two FSL buses) and connect it to the MicroBlaze and your *fsl_hwa* instance. Remember to connect all the necessary reset and clock signals as well.

All you have left to do now is to write the software that uses the core connected to the FSL. To exchange data with the *fsl_hwa* use the already familiar **getfsl()** and **putfsl()** macros.

3 Design Evaluation

You should use the same procedure for evaluating your system as you employed in labs two and three. For a fair comparison, use the same data sets used in the aforementioned labs.

4 Assignment

To wrap it up, during this laboratory session you should:

1. Create the support architecture with your core in it. Make sure you set the core parameters right and that all signals are connected as they should.
2. Synthesize the whole design (generate bitstream and export to XSDK), keeping an eye on the synthesis report to make sure it goes through alright. Note the device utilization data.
3. Ensure that the core works properly inside the system. Write the software part of the *gcd* application.
4. Carry out time measurements and record the number of cycles. Use the same data sets you used to evaluate your designs in labs 2 and 3.
5. Record the power and compute the energy consumption for the same data sets and compiler setups as used in the previous labs.
6. Include all the above data in the final report.

5 Final Remarks & Hints

- Verify with a few lines of code that the hardware receives/sends data before writing the whole program.

- You might need to reset the system (push the "BTN0" – connected to *reset* – button on the board) before inputing data, to make sure your FSMs start from a correct state.