Design Project FDSD: November – December 2023

Design, implement, and simulate a simple 4-bit processor using the Xilinx design tool and simulator. The processor executes the following instruction set:

Instruction Mnemonic	Operation	Description
LDA #*value*	A ← *value*	Load register A with *value*
LDB #*value*	B ← *value*	Load register B with *value*
SHLA	A ← A << 1	Shift-left with 1 bit the value of register A
SHRA	A ← A >> 1	Shift-right with 1 bit the value of register A
MAB	B ← A	Move the value of reg. A to reg. B
ADDAB	A ← A + B	Add reg. A to reg. B and store the result in reg. A
NOTA	A ← A'	Complement reg. A
JC #*value*	PC ← *value*	Jump to address = *value*, **if carry flag is set**

The top-level schematic of the processor is given in the **Appendix file "myProcessor.pdf"**. Use this architecture when you make your design. The processor consists of the following components:

1. Two general purpose 4-bit registers – **register_A** and **register_B**;
2. One 4-bit Arithmetic and Logic Unit (**ALU**) that executes 4 operations:
   - **Addition:** When **S0 = 0** and **S1 = 0** then **res(3:0) ← A(3:0) + B(3:0)**;
   - **Shift-left:** When **S0 = 0** and **S1 = 1** then **res(3:0) ← (A(3:0) shifted-left with one bit)**;
   - **Shift-right:** When **S0 = 1** and **S1 = 0** then **res(3:0) ← (A(3:0) shifted-right with one bit)**;
   - **Complement:** When **S0 = 1** and **S1 = 1** then **res(3:0) ← A(3:0)'**;
3. Two multiplexers 2-to-1 4-line MUXs - **MUX4_2to1**;
4. 1-bit **Status Register** – stores the carry flag of the ALU after ADDAB instruction is executed.
5. One 4-bit **Program Counter (PC)** register that is used as an address generator for the program memory of the processor;
6. One program memory **ROM16x7** that has size 16 memory locations each 7 bits wide. In this memory the program is stored. Every instruction in the program occupies one memory location. The format of each instruction in the memory is as follows:

6 … 4	3 … 0
Operation Code (OPC)	**Immediate operand** *(value)*

   OPC is the code of the instruction. *value* is the operand that is used when instructions LDA, LDB, and JC are executed (see the instruction table above).
7. One **Instruction Decoder** component. It takes as an input an OPC and decodes the corresponding instruction. The outputs of the decoder are control signals that configure the data-path of the processor (register_A, register_B, the two MUX4_2to1, Program Counter, Status Register, and ALU) in accordance with the instruction that has to be executed.

Project Schedule and Deliverables

The project described above will be split and implemented in five small tasks.

1. **Task1:** Design, implementation, and simulation of components **MUX4_2to1**, **register_A**, and **register_B**.
   To be delivered **no later than 22.11.2023;**
2. **Task2:** Design, implementation, and simulation of components **ALU** and **Status Register.**
   To be delivered **no later than 29.11.2023;**
3. **Task3:** Design, implementation, and simulation of components **ROM16x7** and **Program Counter**.
   To be delivered **no later than 06.12.2023;**
4. **Task4:** Design, implementation, and simulation of component **Instruction Decoder**.
   To be delivered **no later than 13.12.2023;**
5. **Task5:** Connect all components designed in Tasks 1 to 4 in order to design the simple processor. Write **one small program** that computes the **absolute value** of the expression |X/2 – Y|. The result has to be in **register B**. The operation "/" is integer division. When the program is written run it 2 times. First, load the program in the memory with X = 12 and Y = 5. Second, load the same program with X = 10 and Y = 11. The simulations should show that your processor works correctly.
   To be delivered **no later than 31.12.2023**.

NOTE1: **You have to work on the project tasks in a group of max 2 students!** So, make sure that you find a partner to work with and **you should not change your partner throughout the project**.

NOTE2: **When you design your specific processor components, you are allowed to (re-)use all pre-designed components that you can find in the Xilinx project library!** So, study all components that are available in the library and try to use them as much as possible in order to save your time and effort.

NOTE3: You have to deliver each task in electronic form **by e-mail to the assistants at "dite.liacs@gmail.com"**. To do this you have to **automatically cleanup (ask the assistants how)** and **zip** the whole Xilinx project in which you do and simulate the tasks. After you zip the project try to **unzip** it on another computer in rooms 302, 306, 303, 307, or 309 and to run it in order to check if you have included all needed files. **If sending the zip file via e-mail fails because of SPAM filters, etc., then upload the zip file on GoogleDrive and send us a link to the zip file such that we can download it. Make the link accessible to everyone and keep the link with each zip file active/available at least 90 days after you send it to us!**
IMPORTANT: If we cannot run and simulate the project, you have sent, we will assume that you have not completed your task. So, be careful to send us a working/correct project (including all needed testbench files) such that we can properly run, simulate, and test/check your project using your testbench files.
Together with each zip file you have to send us a text file/report (max 2 pages) explaining what you did and how to run and simulate the project that is in the zip file. The text file/report is important and its quality will be graded as well in terms of clarity, completeness of explanation, etc.

NOTE4: It will be very beneficial for you to attend the **on-campus Design Project hands-on classes every Wednesday from 15:15h to 17:00h in computer rooms 302, 306, 303, 307, or 309 in Snellius**. There, you can do parts of the project together with us, you can ask questions, and you will receive help from us related to the tasks above. **However, if needed, you have to spend sufficient amount of extra time on the project in the computer rooms or at home in order to complete it successfully!**

Grading of Your Project

- To receive grade 6.0 you have to ***complete successfully*** Task1, Task2, Task3, and Task4;
- Receiving a grade higher than 6.0 will depend on how you perform Task5;

IMPORTANT: Completing a task successfully means:

1. You have delivered the task on time according to the schedule above. If you fail to do this we will assume that you have not completed your task. **So, take the schedule seriously!**
2. The delivered task contains working/correct Xilinx implementation together with the testbench files you have used for the simulation of your implementation.
3. You have delivered a text file/report that explains what you have done in the task and how to simulate your implementation.