

DIGITAL DESIGN LAB (EDA322)

LAB 3

Examiner: Prof. Ioannis Sourdis

TAs: Ahsen Ejaz, Ghaith Abou Dan, Fredrik Jansson, Konstantinos Sotiropoulos,
Magnus Östgren, Neethu Bal Mallya, Panagiotis Strikos

Last Edited: 2024-02-01

Deadline: Your respective lab session during Study Week 6

1 Introduction

The goal of this lab is to implement the controller for the ChAcc processor. The controller is the “brain” of the processor, which orchestrates the different modules to execute the operations on the processor. Before starting the lab, please prepare as described below.

1.1 Preparation

1. Complete Lab1 and Lab2.
2. Listen to the corresponding coding tutorials.
3. Study Sections 2 and 4 in the processor’s specification document (*processor.pdf*). A good understanding of the specifications is required to implement the controller in a reasonable amount of time.
4. Study the lecture material up to the previous study week.

1.2 Learning outcome

After completing this lab, you should be able to:

- Draw Finite-State Machines (FSM) for non-trivial problems like the controller of a simple processor.
- Implement an FSM in VHDL using dataflow or behavioral design style.
- Debug a complete design of the simple processor in VHDL.

2 Tasks

This lab **requires** you to do the following three tasks:

1. Design and implement the controller of the ChAcc processor.
2. Connect the controller to the datapath from Lab 2 and complete the ChAcc processor.
3. Simulate the whole processor by running the provided testbench.

2.1 Task 1: Controller

The controller is implemented as a synchronous sequential circuit using a Finite State Machine (FSM), where a state transition takes place when an executed instruction goes from one datapath stage into another. Recall that the datapath has five stages: FE, DE1, DE2, EX and ME. Each instruction might only use a subset of these stages, which is known when the controller decodes the opcode in the DE1 stage. Therefore, each instruction will require different states in the FSM, and the state transition may vary. In addition, the control signals (controller’s outputs) need to be set or reset in every state (stage). Refer to Section 4 in *processor.pdf* for further details. There are a few states (FE and DE1) that are shared by all the instructions. For the rest, there is one state sub-diagram per instruction.

There are two FSM design alternatives:

- Moore-type: The outputs are determined based on the current state and are independent of the inputs.
- Mealy-type: The output signals depend on both the input and the current state.

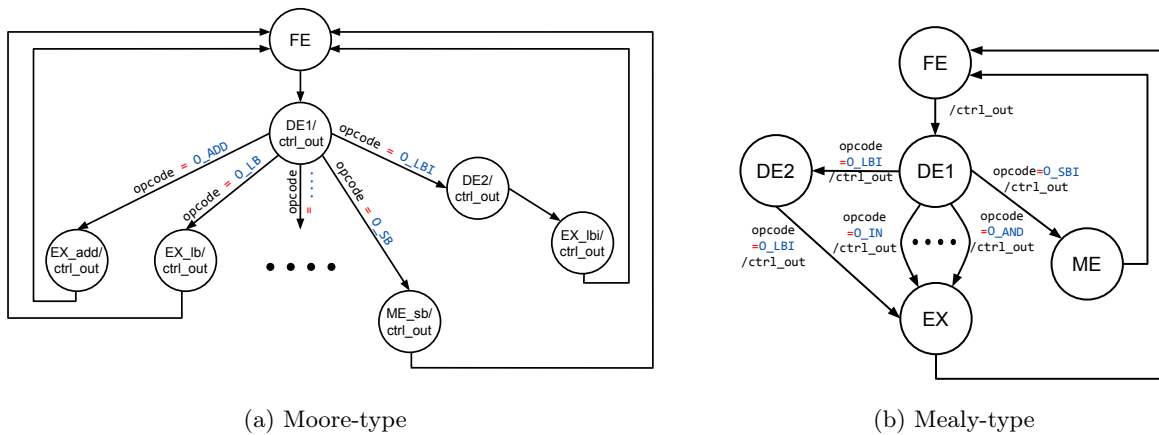


Figure 1: Part of the FSM controller

Figures 1 (a) and (b) depict an example part of the controller using a Moore-type and a Mealy-type FSM. `ctrl_out` in the figures represents the controller outputs in the particular state (for Moore) or the state transition (for Mealy).

To design and implement the controller, follow the steps below:

1. Draw the FSM on paper (or use any tool at your convenience). Include all the states and the outputs in each state. For the Moore machine, if possible, combine similar states. You can use either design alternative, but the tips given below assume a Mealy machine.

As part of the lab's examination, be prepared to discuss your choice, the advantages and disadvantages of each type of FSM, and the different states the other option will need to have.

2. You now have all the essential information to implement the controller of the ChAcc processor. The entity of the controller module is given in `proc_controller.vhd`. The controller's interface (input-s/outputs) is detailed in the processor's specification document (Section 4.1, *processor.pdf*). Moreover, you can refer to Table 2 in *processor.pdf* for the datapath stages used by the different instructions.

- It is strongly recommended that you implement the FSM as a sequential circuit that consists of two parts: a) combinational and b) storage element that is actually implemented using registers. The internal state transitions of the controller should take into account the `master_load_enable` signal.
- The FSM state should be stored in a register. For this, you will need to implement a process that uses two signals for the register's input and output. The process will be similar to the one you used for the register in Lab 2.
- After you have implemented the state register and depending on the coding style of your choice, you need one or two more processes for the next state and output of the FSM
- When reset (`resetn`) is enabled ('0'), a state transition from any state to FE must take place and all the output control signals must be reset. When `resetn` is released, the program starts from the beginning as PC is set to 0.
- Use behavioral or dataflow design style, but note that state assignment and minimization are required if you select the dataflow design style.

Tips: To keep your code clean and readable, we recommend not using raw bit vectors for this but instead named constants. These are already provided for you in the `chacc_pkg` package(`chacc_pkg.vhd`).

Hints: For the FSM state register, you can introduce a new type using the following syntax:

```
type state_type is (FE, DE1, DE2, EX, ME);
signal curr_state : state_type;
signal next_state : state_type;
```

2.2 Task 2: Complete your ChAcc Processor

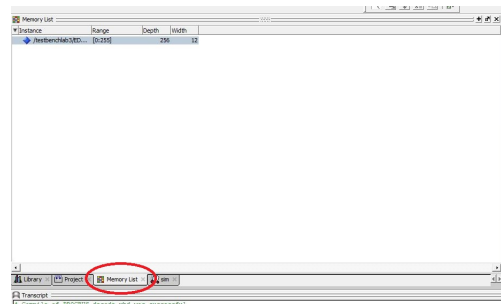
To complete your ChAcc processor, replace the mock controller used in Lab 2, and connect the new controller in `EDA322_processor.vhd`. Compile with VHDL files and fix syntax errors if any.

2.3 Task 3: Test your ChAcc Processor

You are provided with a program that calculates the first 12 numbers of the Fibonacci sequence. When the simulation completes, you should see numbers 89 and 144 in the addresses 1 and 2 of the data memory.

To verify the correctness of your ChAcc processor, follow the steps below: (Note: The files mentioned below are available in Canvas: *Files > Labs > Lab 3 > lab3_files*).

- Initialize the memories using `i_memory_lab3.mif` and `d_memory_lab3.mif` files. The test program within `i_memory_lab3.mif` is provided in `lab3code.txt` for your reference.
- Simulate and run the provided testbench (`testbench_lab3.vhd`) in ModelSim/ QuestaSim for at least **4000 ns**.
- After completing the test run, check the contents of the memory. If the test fails, debug your design and fix the error(s).
 - In ModelSim, you can check the memories and their contents from the *Memory List* window. You can access the window using either of the following:
 - * Menu item: *View > Memory List*
 - * Command: `view memory list`



Tips: Several issues might cause your design not to perform as expected. Here is a list of things that might help you with **debugging your design**:

- First, check your controller. Load its inputs, outputs, and the state signals to the waveform and re-run the testbench.
 - Check that the state transitions happen correctly for each simulated instruction and the right output signals are generated based on the provided specifications.
 - In the controller, state transition should not start before the reset signal is released.
 - Upon the arrival of a new opcode, in which state should your controller be?
 - The `e_flag` and `z_flag` to the controller should come from the outputs of the E and Z registers.
- If your controller operates as expected, the problem may be on the datapath. Verify the value of the processor's main output signals and/or the internal signals by adding them to the waveform.
- If there are **red** lines in your waveforms, focus on them before going into a detailed examination of other signals' values.
- Examine the test program (`lab3code.txt`) and understand what it does in every step. Try to understand the input instructions and the expected sequence of outputs. Spot the element of the memory that is wrong at the end of the simulation.

- Identify the instruction responsible for writing the particular value in the data memory. Check whether the specific instruction was executed correctly.
- If all of the above is correct, it is possible that either (a) some previous instruction produced a wrong result and updated the memory, or (b) that some datapath component in the execution of the instruction is not functionally correct.
- Keep tracing back the error to the actual cause.

3 Demonstration and Evaluation

The lab will be evaluated according to the checked aspects in the table below. To demonstrate your successful completion of Lab 3, keep all the essential files and simulation results ready to be presented to a TA.

Also, as mentioned earlier, be prepared to discuss your preferences regarding the implementation of the FSM.

Task#	Files	Coding Style	Simulation
1	proc_controller.vhd	✓	
2	EDA322_processor.vhd	✓	
3		✓	✓ (Using testbench_lab3.vhd, given .mif files, and by examining the contents of the data memory)

- The demo must be completed during your registered lab session.
- Note that there is no code upload required for this lab.