MIDDLE EAST TECHNICAL UNIVERSITY

ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT

EE446

Computer Architecture II

Laboratory Manual

Course Instructors:
Prof. Dr. Gözde Bozdağı Akar
Assoc. Prof. Dr. Cüneyt Bazlamaçcı

Laboratory Assistants:
Ece Selin Böncü
Yeti Ziya Gürbüz
Kamil Sert

February 2018

Contents

1 Laboratory Regulations **3**
1.1 Rules . . . 3
1.2 Cheating . . . 4
1.3 Remarks and Evaluation . . . 5

2 General Information about Laboratory **6**
2.1 Experimental Setup . . . 7

3 Using ModelSim and Quartus Software **8**
3.1 Simulation in ModelSim Using Test Bench . . . 8
3.2 Embedding Verilog Designs to DE0-Nano FPGA Board . . . 9
3.3 Some Useful How-to Items . . . 10
3.3.1 How to create a Verilog module with parameters . . . 10
3.3.2 How to create a schematic design . . . 10
3.3.3 How to use your Verilog modules in schematic design . . . 11
3.3.4 How to add symbols(modules) to the work space in schematic design . . . 11
3.3.5 How to split bus in schematic editor . . . 11
3.3.6 How to create input and output ports for a schematic design . . . 12
3.3.7 How to create Verilog codes from your schematic designs . . . 12

Course Instructors

Name	e-mail	Room
Prof. Dr. Gözde Bozdağı Akar	bozdagi@metu.edu.tr	D-121/1
Assoc. Prof. Dr. Cüneyt Bazlamaçcı	cuneytb@metu.edu.tr	EA-410 / D-215 (Temporarily)

Laboratory Assistants

Name	e-mail	Room
Ece Selin Böncü	boncu@metu.edu.tr	EA-404 / ARC-202 / DB-14
Yeti Ziya Gürbüz	ygurbuz@metu.edu.tr	EA-404 / ARC-202
Kamil Sert	ksert@metu.edu.tr	EA-405

1 Laboratory Regulations

This laboratory is a very important part of *EE446 - Computer Architecture II* course in order to thoroughly understand the concepts given in the computer architecture lectures. By attending experiments and completing all the work, the key concepts and most of the abstract parts of the lectures can be grasped very easily. It also makes up 20% of all grades of the course. Thus, it is important to know the regulating rules of this laboratory for both a better understanding and for your grades.

There are rules for the regulation of *EE446 Laboratory*. These rules are strict; and by taking *EE446 - Computer Architecture II* course, you will be considered that you have understood and accepted all the rules stated below.

1.1 Rules

The rules for EE446 - Computer Architecture II Laboratory is given below, please read thoroughly:

1. The manual of an experiment will be available at least a week before the corresponding experiment.
2. A preliminary work which is to be detailed in the experiment manual has to be prepared for each experiment.
3. There will be a quiz for each experiment in the week of each experiment. The quizzes are to be held in the class hours of EE446 course on Wednesdays. **You will NOT be allowed** to attend the relevant laboratory session if you miss the quiz.
4. A quiz will have two parts each of which is to be evaluated out of 10. One part is for the related laboratory work and the other part is for the course material covered up to the date when the corresponding quiz is held. It is important to note that the laboratory work covers the related topics from the computer architecture courses and those topics are the ones that are exploited during the preliminary work preparation.
5. Depending on the quiz grade of the laboratory work part, the quizzes may penalize the performance grade of the related laboratory work. Students whose quiz grades for the laboratory work part are **less than 4** will get their performance grade from the related experiment multiplied by the ratio (quiz grade)/10. The performance grades of the students whose quiz grades are **greater than or equal to 4 and less than 6** will be **halved**. There will be no penalty for the students whose grades are **more than or equal to 6**. That is to say, assuming that your quiz grade for the laboratory work part is Q and your laboratory performance grade for the corresponding experiment is P, your penalized performance grade, $\hat{P}$, becomes:

$$\hat{P} = \begin{cases} \frac{Q}{10}\,P, & Q < 4 \\ 0.5\,P, & 4 \leqslant Q < 6 \\ P, & Q \geqslant 6 \end{cases} \tag{1}$$

6. The preliminary works are to be collected before the quizzes.
7. Preliminary work is a crucial part of the laboratory work in both preparing for the experiment and understanding the concepts to be covered in the corresponding experiment. Thus, **You will NOT be allowed** to attend the relevant laboratory session without any preliminary work. Partially done preliminary work reports are acceptable if and only if at least the %30 of the total work is completed. Redrawings or rewritings of the given material will not be considered within the aforementioned %30.
8. You are expected to prepare your preliminary work on your own.
9. Experiments are to be performed individually.
10. Sharing any form of information during a laboratory session is strictly forbidden.

11. No extra time will be given to the latecomers and no one is allowed to take the laboratory session after 20 minutes past the beginning of the session.
12. Leaving laboratory room during a session is not allowed except for the emergency situations.
13. Transfer between laboratory sessions (e.g. from *Wednesday Afternoon* to *Thursday Afternoon 1*) will NOT be allowed.
14. Cheating is an important issue regarding ALL steps of the laboratory work that involve preliminary work and laboratory performance. Disciplinary action is to be taken in case of any cheating and cheating attempt. Please read the subsection 1.2 for detailed information about cheating.
15. Your codes have to be well commented. The codes lacking of comments will not be evaluated. Please read the subsection 1.3 for detailed information about coding and commenting.
16. Students with officially documented legal excuses will be allowed to take make-ups. Only academical permissions (given by University), signed confirmation from the instructors for any exam clashes, and METU Health and Counseling Center (MEDIKO) will be regarded as valid documents for the right to take make-ups. You may take at most 3 make-ups even if you have more than 3 officially documented legal excuses.
17. All your course related e-mails should be sent to ee446coordination@gmail.com address. If you have a general question that may interest the other students, please start a discussion in ODTUClass.
18. Students who do not attend **2 experiments** without legal excuses will get **ZERO** from the laboratory. Students who do not attend **greater than or equal to 3** experiments will get **N/A** from the EE446 course. Please note that missing the quiz of an experiment or insufficient preparation of the preliminary work of an experiment is equivalent to not attending the corresponding experiment.
19. This document supersedes any other previous documents.

1.2 Cheating

You are considered to be graduate and become engineers and colleagues in 1 or 2 years time. Hence, you are expected to act according to the professionalism that is required as a METU graduate.

Copying a work from any other resource (web-page, your friend's report, older resources you have found, etc.) or sharing information, know-how or code files during sessions are considered as cheating.

Helping your friends, studying together, or any form of cooperation is encouraged -since it both fosters your relationships with others and helps you learn the topic better- as well as YOU do your own work. Creating only one report is not studying together or is not cooperation, and will NOT be accepted.

1.3 Remarks and Evaluation

Your laboratory grade is to be composed of your preliminary work grade and your laboratory session performance grade. Preliminary work requires implementation of different tasks some of which are to be experimented during the laboratory sessions. The performance grade is according to the evaluation of the tasks that are considered in the laboratory session. The evaluation is based on the functionality of your implementations and comprehensiveness of your knowledge of the related laboratory topic. If a task is not implemented in your preliminary work, then you will get ZERO performance grade from corresponding step. Moreover, implementations without comments will not be considered as valid implementations of the tasks and the corresponding task will not be evaluated. You are expected to write down explanatory comments to your code. That does not mean you should write an explanation next to each code line. What is required is explaining the functionality of the code blocks and the functionality of the representative code lines where necessary.

You can - actually are expected to - practice your implementations before attending your laboratory session so that major bugs that may cost too much time to fix can be eliminated and you are on the safer side to complete the experimental work within the required time slot. You can practice and work on your implementations by using the laboratory at EA-409 that is to be open to access 7/24 (once you get your student ID card authorized).

2 General Information about Laboratory

As it is mentioned before, this laboratory is a very important part of *EE446 - Computer Architecture II* course. The laboratory work helps you understand most of the concepts given in the computer architecture lectures. Also, materializing abstract parts of the lectures will be much more easy.

The experiments will be carried out in Microprocessor Laboratory at EA-407 which is located on the 4^{th} floor of A Block of our department. EA-409 laboratory will be open to access 7/24 for you to practice. Your student ID card has to be authorized for the entrance. You may get your ID authorized by visiting the staff responsible for that. Once you are authorized, you may test and debug your preliminary work prior to your laboratory session. You may also use EA-407 laboratory within the work hours except for laboratory sessions.

Figure 1: Microprocessor and Computer Architecture Laboratory, A Building, Room 407

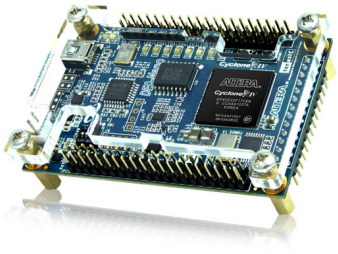
There are different sessions of laboratory, which are provided by Table 1. You are assigned to one of the sections to perform your experiments.

Table 1: Laboratory Sessions of 2016 Fall Semester

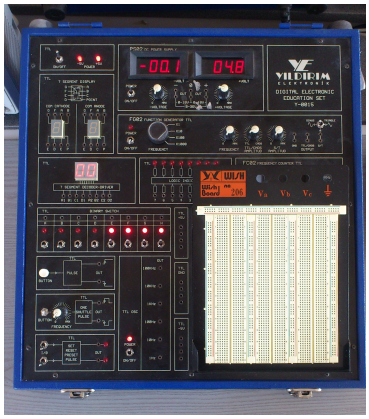
Time	**Wednesday**	**Thursday**	**Friday**
09.00 - 10.50			FM
11.00 - 12.50			
13.40 - 15.30	WM	THM1	FA1
15.40 - 17.30		THM2	FA2

There will be a total of 6 experiments. These experiments are based on constructively practicing the design of computers via Verilog hardware description language and will be in the following subjects:

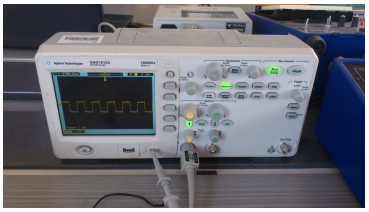
- Fundamental modules for computer design
- Arithmetic logic processor
- Single cycle computer
- Multi cycle computer: Instruction set architecture and data-path design
- Multi cycle computer: Controller design
- Introduction to multitasking and process scheduling



(a) DE0-Nano Board



(b) DEES



(c) Digital Oscilloscope

Figure 2: Devices used in the laboratory

2.1 Experimental Setup

The experimental setup of EE446 Laboratory is composed of the following items:

- A notebook computer
  - You may login as student, which does not require any password
- DE0-Nano Development and Education Board - A board containing Altera Cyclone IV EP4CE22F17C6N FPGA
  - DE0-Nano is connected to the notebook via USB data cable.
  - DE0-Nano is programmed via Quartus installed in the notebook (to be explained in Section 3.2).
  - For simulation purposes, ModelSim is installed in the notebook (to be explained in Section 3.1).
- DEES: Digital Electronics Education Set - A set that includes LED displays, switches, and a breadboard in order to construct external circuitry
- Digital Oscilloscope - Oscilloscope for debugging purposes

3 Using ModelSim and Quartus Software

This section introduces ModelSim and Quartus software to be used throughout the laboratory work to perform behavioral simulation for the Verilog design codes and to embed the designs to the FPGA, respectively. In the scope of this section, only walk-through to use ModelSim and Quartus software for the laboratory work purpose is covered. For more general usage tutorial of those software, one can refer to the tutorials available in EE446 ODTUClass course page.

3.1 Simulation in ModelSim Using Test Bench

This part covers only simulation of the designs via test bench codes and test vectors. It is assumed that a test bench code and the corresponding test vectors are available for a design to be tested. To recall how to write test bench codes for the designs, one can refer to the related lecture notes of EE445 course, which are available as a supplementary material in EE446 ODTUClass course page.

The following walk-through is to demonstrate how to perform behavioral simulation of a user defined module with its test bench and test vectors. A video demonstration of this walk-through is available in EE446 ODTUClass course page.

1. Create a project in ModelSim
   - Open ModelSim
   - If you see a pop-up window, just click '*Jumpstart*'
   - You should see a welcome pop-up window
   - Click '*Create a Project*' link
   - Another pop-up window to create a project will appear and from this window, set your project name, provide a directory for the project and set a library name
   - Click '*OK*' when you are done then a window appears to add files to the project; select adding existing files
   - Browse the location of your codes and make sure that '*All Files (*.*)*' should be selected as file type to add test vectors with '.tv' extension
   - Select your files to be added and and proceed
   - Before leaving the '*Add file to Project*' window, make sure that '*Copy to project directory*' option is selected
2. Now, from '*Compile*' menu, compile your codes with '*Compile All*'
3. Start simulation
   - From '*Simulate*' menu, go to '*Start Simulation...*'
   - A window will appear and from '*Design*' tab, find your library name and select your test bench code under that library
4. A GUI for simulation should appear, from the '*Objects*' box, select all the variables and '*Add Wave*' by right-clicking
5. Run the simulation from either '*Simulate*' menu or the available buttons in the GUI

3.2 Embedding Verilog Designs to DE0-Nano FPGA Board

This part only focuses on programming the FPGA chip of DE0-Nano Board. In general, to embed an HDL design to a board equipped with an FPGA chip, I/O and clock mappings to the pins of the board should be specified. Quartus software has GUI tools to perform those I/O mappings. However, it is prone to errors as well as time consuming to perform pin mappings for each design from scratch. Therefore, a hierarchical design procedure is to be followed, in which, there exists a top level module whose inputs and outputs correspond to the physical pins of the board. Any user designed module can be integrated under that top level module. Thus, pin mappings are to be performed only for the top module.

To introduce re-usability of the pin mappings, a template project including the top level module for DE0-Nano board and its pin mappings is created. In this way, pin mappings are to be performed only for the top module of this template project once and then by using that template, any design file can be embedded to the board easily by writing few lines of code. The template project is readily available in the notebook computers in the laboratory and in EE446 ODTUClass course page. The name of the folder name of the template project is '*empty_project*'. The following simple walk-through explains how to embed a design by using this '*empty_project*' template.

1. Copy the '*empty_project*' folder to a working directory of your choice
2. Copy your design files-if any- inside the '*empty_project*' folder
3. Open '*de0nano_embedding.qpf*' Quartus project file
4. Now, from the '*Project*' menu, add your design files to the project
    - If you do not have your design files yet, you may proceed with adding new files to your project
    - Please refer to the tutorial on Quartus software in EE446 ODTUClass course page if you are unfamiliar with adding existing or new files to a project
5. According to your designed modules, modify the top level module file '*de0nano_embedding.v*'
6. Finally, compile the project and load the program to the board
    - Please refer to the tutorial on Quartus software in EE446 ODTUClass course page if you are unfamiliar with compiling a project and loading the program to the board

DE0-Nano board has switches, keys, LEDs, general purpose I/O pins, clock and SDRAM. If the '*de0nano_embedding.v*' file is examined, it can be observed that the aforementioned pins are associated with variable names. To illustrate, assuming that you have a module named '*my_module*' with two inputs x0, x1 and an output; if you want to connect the inputs to the keys and the output to one of leds then you should simply add the following line to the top level design file: '*my_module mm(.x0(KEY[0]), .x1(KEY[1]), .y(LED[0]));*'. Similarly, if you want to use the clock of the DE0-Nano board as the clock source of your design, you simply assign '*CLOCK_50*' variable to the clock variable of your module. An example of a modified top level module design file is available in EE446 ODTUClass course page.

3.3 Some Useful How-to Items

In this section some tips and tricks on using Quartis software and Verilog programming are to be presented. Those can be helpful for your preliminary work tasks.

3.3.1 How to create a Verilog module with parameters

There are many cases where designing a module should be generalized. For example, a register design can be used for different data widths. It is cumbersome to implement the same design to support different data widths. To overcome this, some parts of the module can be parametrized so that modules with different properties of the same design can be instantiated. For the register example, one can make the data width a parameter so that by varying the parameter, registers of different data width can be created from the same design.

In Verilog, a module can be easily parametrized. You will just add a parameter block before defining the arguments of the module. An example of a module with parameters is given below:

```
// an example module with parameters
// the module is to split a data bus into two

module bus_split #(parameter W=8, S=4) (bus_in, split_out0, split_out1);

        input wire [(W - 1):0] bus_in;

        output wire [(W - S - 1):0] split_out1;
        output wire [(S - 1):0] split_out0;

        assign bus_out0 = bus_in[(S - 1):0];
        assign bus_out1 = bus_in[(W - 1):S];

endmodule
```

Note that, the default values of the parameters should be provided so that the module can be instantiated with the default parameters if no parameter setting is performed. To create a module with a specified parameter, the values of the parameters should be supplied to the module in the order they are defined:

```
// assume you want to split the content of the instruction register
// say the name of the output of the instruction register is reg_out

...

wire [3:0] instr;       // most significant 4 bits are for instruction
wire [11:0] oprnd;      // the rest is for operand

bus_split #(16, 12) split_instr(.bus_in(reg_out), .split_out0(oprnd), .split_out1(instr)

...
```

3.3.2 How to create a schematic design

Designing modules with Verilog HDL is very convenient; however, sometimes schematic design is more preferable for the sake of clarity. Especially for the datapath designs, where the most of the task is connecting wires between modules, schematic design can be less painful than its Verilog counterpart.

To create a schematic file in Quartus:

1. Go to <<File→New>> in Quartus, or press CTRL+N
2. Select <<Block Diagram/Schematic File>> under Design Files
3. A .bdf file should be created; save that file to add it to your project

3.3.3 How to use your Verilog modules in schematic design

For computer architecture and its datapath designs, you will mostly use schematic editor to make connections among your Verilog modules. At that point, you will need your Verilog modules as schematic symbols so that they can be placed onto work space of the editor.

To create symbols for your Verilog module in Quartus:

1. Open your Verilog(.v) file where your modules are implemented
2. Go to <<File→Create / Update→Create Symbol Files for the Current File >> in Quartus
3. It will compile your codes and if everything is all right, the symbols are created for the modules in that file

3.3.4 How to add symbols(modules) to the work space in schematic design

To add a module to your work space in schematic editor:

1. Find <<Symbol Tool>> button in the toolbar of the editor (a button with AND gate icon)
2. Upon your press on that button, a window will appear
   - You may find your modules under <<Project>> library
   - You may also find some useful modules under Quartus libraries

3.3.5 How to split bus in schematic editor

There will be cases where you need to split data bus into sub buses. For instance, you may need to connect the most significant bit of the output of a register to the least significant bit of the input of another register. However, your inputs and outputs are available as wire arrays, namely, buses. It is straightforward splitting buses in Verilog HDL by coding accordingly. On the other hand, it is also very easy in schematic editor once you know the tricks.

To split a bus, you just need to create a bus and name it to the portion of the bus you want to split. For instance, assume that you have a module with an output of width 10 and the name of that port is my_out. Whenever you make a connection to that port with bus tool, the name of that connection bus inherently becomes my_out[9..0]. Thus, if you want to split that bus, you just need to rename it as my_out[3..0] to make a connection to the least significant 4 bits only. Splitting two different buses and connecting them is a bit more tricky; since, the same connections with different names will cause name conflicts. To overcome this, schematic editor has WIRE symbol in its built-in library. You should use WIRE module when connecting two buses with different names. Long story short, to split buses:

1. Create a bus to represent the whole bus and name it with the name of the port it corresponds
   - For instance, if you are to split an 8-bit output port named p_out[7..0], create a bus, name it to p_out[7..0], and connect it to the port
2. Create a bus to represent a sub bus and name it with the name of the bus it corresponds; however, specify the wires you want to include then connect it to its ancestor bus
   - For instance, if you are to split the least significant 4 bits of an 8-bit bus named p_out[7..0], create a bus, name it to p_out[3..0], and connect it to the bus

- If you want to split a non contiguous wires from the bus, you can include the name of all the portions you want to split with commas (E.g. p_out[7..5], p_out[3], p_out[0])

3. If you want to connect your sub bus to a port whose pin order and pin count match your sub bus naming convention, you may directly connect it to that port
4. If you are to connect your sub bus to another sub bus, add a WIRE symbol in between two connections.

In Figure 3, an example of the aforementioned procedure is depicted. The colored buses are the ones that are sub buses of different buses. The most significant 4 bits of p_out port are connected to the least significant 4 bits of in port and vice-verse.

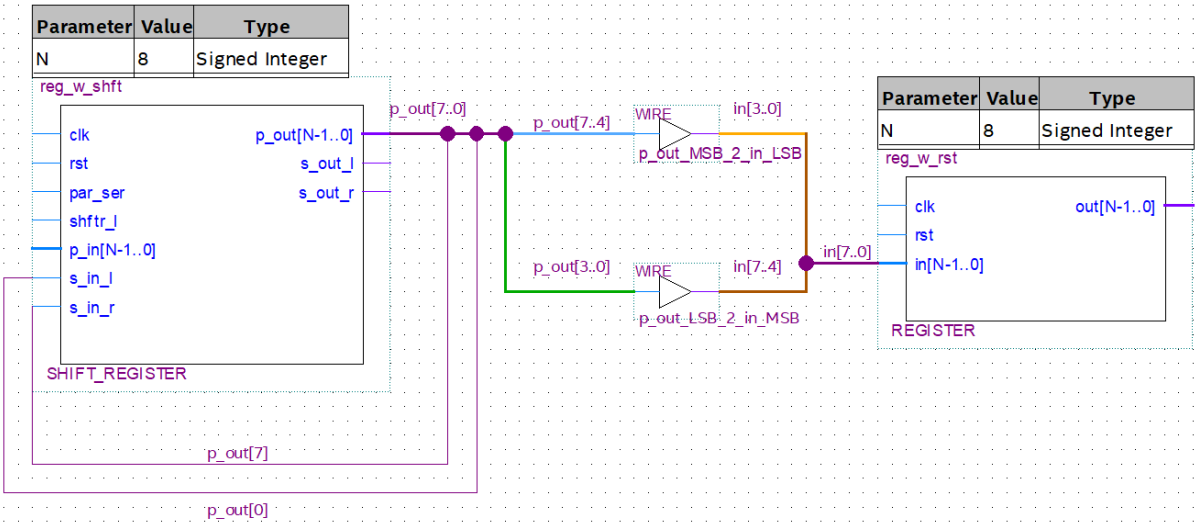


Figure 3: An example of bus splitting in schematic editor

3.3.6 How to create input and output ports for a schematic design

It is important to add input and output pins to your schematic design so that you may use your design as a block in other designs.

To add input and output pins to a schematic design:

1. Find <<Pin Tool>> button in the toolbar of the editor (a button with 3 pins and an 'in' string icon)
2. From drop-down menu of that button, select the pin you want to add

3.3.7 How to create Verilog codes from your schematic designs

You may need to create Verilog HDL code of a schematic design. To create Verilog HDL codes from your schematic design files in Quartus:

1. Open your Schematic Design(.bdf) file where your design is implemented
2. Go to <<File→Create / Update→Create HDL Design File from Current File >> in Quartus
3. In the next pop-up window, select Verilog as the HDL
4. It will compile your codes and if everything is all right, the Verilog HDL is created for the schematic design in that file