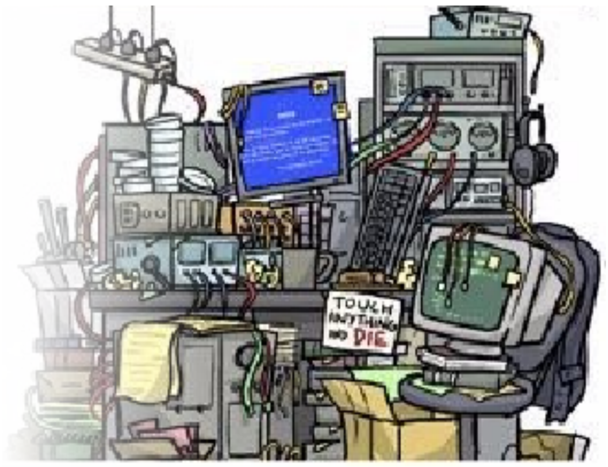
METU EE 446
Computer Architecture
Laboratory

**Warming Up
for
Computer Design**



Laboratory Work 1 - Warming Up for Computer Design

Objectives

The purpose of the first laboratory work is to construct a Verilog library composed of the fundamental modules to be used throughout the design of a computer. Moreover, simple datapath design is to be practiced through designing simple architectures from the modules in the constructed library to perform some simple tasks.

During this laboratory work, one will be familiar with designing modules with Verilog HDL and constructing architectures with schematic design. This laboratory work is a tool for getting familiar with the software-Quartus and Modelsim- to be used throughout the semester. Finally, one will practice embedding the designs to a development board, DE0-Nano, equipped with field programmable gate array (FPGA) and several peripheral units such as switch inputs, general purposed I/O pins, LED outputs etc.

1 Preliminary Work

To fulfill the requirements of this laboratory work, the following tasks should be performed.

1.1 Reading Assignment

The laboratory manual where the regulations and some other useful information exist is available at the ODTUClass course page. Read that manual thoroughly. If you feel yourself unfamiliar with Verilog HDL programming, please refer to the corresponding lecture notes of EE445 course, which are available at the course page.

1.2 Module Design with Verilog HDL (30% Credits)

For this part, you will implement fundamental modules in Verilog. These modules will be then used to construct more complex modules and computer architectures. More importantly, each module you design is to be added to your module library so that you will make use of them in the future laboratory works. Thus, consider this part of the preliminary work as building your own Verilog module library.

For each item in this part, you should submit hard copy of your design codes and the corresponding test bench codes together with test vectors. Additionally, remember to attach your explanation for the 2^{nd} item of the ALU design in **??**.

1.2.1 Constant Value Generator

Implement a module to put a constant value to a data bus. The module should have two configurable parameters one of which is for the data width and the other is for the value. If the bus width is specified as W, for example, the module should continuously output the specified constant value to a data bus of W-bit width.

1.2.2 Decoder

1. Implement a 2 to 4 decoder.
2. Write a test bench module to test your implementation.
3. Provide a vector table to be used by your test bench module to test your implementation.
4. Verify that your implementation is correct.

1.2.3 Multiplexers

1. Implement a W-bit 2 to 1 and a W-bit 4 to 1 multiplexers, where W is a parameter specifying the data width of the input.
2. Write test bench modules to test your implementations.
3. Provide vector tables to be used by your test bench modules to test your implementations.
4. Verify that your implementations are correct.

1.2.4 Arithmetic Logic Unit (ALU)

1. Implement a W-bit ALU for 2's complement arithmetic, where W is a parameter specifying the data width of its inputs. The ALU should be equipped with 8 operations controlled by 3 control signals. In addition to the N-bit result output, the ALU should have 4 other status output bits: Carry out (CO), overflow (OVF), negative (N) and zero (Z). Negative and zero bits are affected by all the ALU operations; whereas, carry out and overflow can only be affected by arithmetic operations. The specifications of the ALU operations and the ALU status outputs are provided in Table **??** and Table **??**, respectively.

Table 1: ALU Operation Control

ALU Control [2:0]	**ALU Operation**	**Symbol**
000	Addition	$A + B$
001	SubtractionAB	$A - B$
010	SubtractionBA	$B - A$
011	Bit Clear	$A \wedge \neg B$
100	AND	$A \wedge B$
101	OR	$A \vee B$
110	EXOR	$A \oplus B$
111	EXNOR	$\neg(A \oplus B)$

Table 2: ALU Status Descriptions

Status	**Description**
CO	1 if there is a Carry Out from add or subtract operations; 0 for logic operations
OVF	1 if the add or subtract operation results in overflow; 0 for logic operations
Z	1 if the result is zero
N	1 if the result is negative

2. Explain your method to detect overflow.
3. Write a test bench module to test your implementation.
4. Provide a vector table to be used by your test bench module to test your implementation. For this step, explicitly state to what operation a vector corresponds. In other words, if you are testing addition of the two numbers that will cause overflow, explicitly state which vector in the vector table is dedicated to that test.
5. Verify that your implementation is correct.

1.2.5 Registers

For this step, you will implement 3 different W-bit registers, where W is a parameter specifying the data width of the parallel input to the register. Note that the registers should have a clock input, it is not mentioned in the following items explicitly though.

1. Simple register with synchronous reset: Implement a positive edge triggered register with parallel load and synchronous reset. If the reset signal is 1, the contents of the register is cleared at the next rising edge of the clock. If the reset signal is 0, the contents of the register is loaded with the input data at the next rising edge of the clock. The specifications of the simple register with synchronous reset is provided in Table **??**.

Table 3: Simple Register (A) with Reset

Reset	Operation
0	$A \leftarrow DATA$
1	$A \leftarrow 0$

2. Register with synchronous reset and write enable: Implement a positive edge triggered register with parallel load, write enable and synchronous reset. If the reset signal is 1, the contents of the register is cleared at the next rising edge of the clock. If the reset signal is 0 and write enable signal is 1, the contents of the register is loaded with the input data at the next rising edge of the clock. Finally, if the reset signal is 0 and write enable signal is 0, the register retains its content. The specifications of the register with synchronous reset and write enable is provided in Table **??**.

Table 4: Register (A) with synchronous reset and write enable

Reset	Write Enable	Operation
0	0	Retain
0	1	$A \leftarrow DATA$
1	X	$A \leftarrow 0$

3. Shift register with parallel and serial load: Implement a positive edge triggered shift register with parallel and serial load. The register has three control signals: Synchronous reset, parallel/serial load select, shift left/right select. There are 3 input sources to the register: W-bit parallel input, 1-bit serial input left and 1-bit serial input right. If the reset signal is 1, the contents of the register is cleared at the next rising edge of the clock. If the reset signal is 0, the operation of the register is determined according to the parallel/serial and shift right/left select signals. If the parallel/serial select signal is 1, the contents of the register is loaded with the parallel input at the next rising edge of the clock. If the parallel/serial select signal is 0, the contents of the register is shifted and the left most or right most bit of the register is loaded with the corresponding serial input at the next rising edge of the clock. For the serial input operation, if the shift right/left select signal is 1, the contents are shifted right and the most significant bit is loaded with the serial input left; if the shift right/left select signal is 0, the contents are shifted left and the least significant bit is loaded with the serial input right. The summary of the specifications of the shift register with with parallel and serial load is provided in Table **??**.

Table 5: W-bit shift register (A) with parallel and serial load

Reset	Parallel/Serial	Right/Left	Operation
0	0	0	Shift Left A, $A[0] \leftarrow$ Serial Input Right
0	0	1	Shift Right A, $A[W-1] \leftarrow$ Serial Input Left
0	1	X	$A \leftarrow$ Parallel Input
1	X	X	$A \leftarrow 0$

4. For the aforementioned 3 registers, write test bench modules to test your implementation.
5. Provide vector tables to be used by your test bench modules to test your implementations.
6. Verify that your implementation is correct.

1.3 Register File(20% Credits)

For this part, you will use your modules available from Part **??** to design a W-bit register file of 4 registers, where W is a parameter specifying the data width of the registers. The register file will be the central storage of the computer you will design in the future laboratory works.

For the design of the register file, you will use your **decoder**, **multiplexer** and **register** implementations. The design should be according to the desired operation of the register file.

The register file has one data input and two data outputs. The sources of the outputs and the destination of the input can be one of the 4 registers in the register file. Therefore, there should be 3 address inputs of width 2: one for destination select and two for source select. Finally, a control signal is required to enable write operation and a synchronous reset signal is required to clear the contents of the all registers in the register file. Note that the register file should inherently have a clock input, thus it is not mentioned explicitly.

The content of a register in the register file should be able to be modified without affecting the contents of the other registers. To modify a register, it should be addressed and write enable control of the register file module should be 1. If write enable of the register file is 0, then the contents of the registers cannot be modified except for the reset condition. Note that the write operation is synchronous; however, read operation should be asynchronous so that the data outputs are available as soon as their sources are addressed.

According to the aforementioned desired operation of the register file:

1. Design and sketch (on a paper) a datapath for a register file design using your decoder, multiplexer and register modules. You may use additional gates wherever necessary. For your sketch, you may present your modules with boxes.
2. Implement your design in Verilog HDL.
3. Write a test bench module to test your implementation.
4. Provide a vector table to be used by your test bench module to test your implementation.
5. Verify that your implementation is correct.

You should submit hard copy of the sketch of your design, the design code and the corresponding test bench code together with test vectors.

1.4 Datapath Design for an Architecture(50% Credits)

In this part, you will design a datapath for an architecture so that you can perform several tasks by applying proper control signals. The architecture to be completed is provided in Figure ??. There are two simple 8-bit registers, R0 and R1, and two shift registers with parallel and serial load, Acc and Q. External data input is directly connected to the inputs of the R0 register and the output of the R0 register is directly connected to the B input of the ALU. Assuming that **you do not have explicit access to**

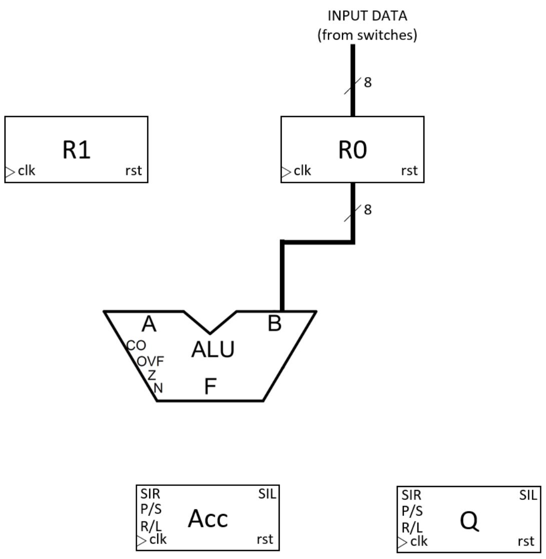


Figure 1: Architecture to which a datapath is to be designed

the output of the R0 register, you should complete the architecture by designing a datapath so that the following tasks can be performed with the desired constraints:

1. LOAD: The content of the R0 register is loaded to the R1 register in 1 cycle. Namely, R1 ← R0 operation takes 1 cycle.
2. INCREMENTED LOAD: The content of the R0 register plus one is loaded to the R1 register in 1 cycle. Namely, R1 ← R0 + 1 operation takes 1 cycle.
3. REVERSED LOAD: The reversed content of the R0 register is loaded to the R1 register after application of a proper set of control signals. For instance, if the content of the R0 is $d_7d_6d_5d_4d_3d_2d_1d_0$, then the content of the R1 should be $d_0d_1d_2d_3d_4d_5d_6d_7$ after the proper set of controls signals are applied. Recall that, you cannot connect anything to the outputs of the R0 register.

According to the aforementioned tasks with specified constraints, design a datapath by using your modules implemented in Part ?? and additional gates, if you need. You will implement your design in **Schematic Editor**. Thus, make sure that your Verilog modules are exported as symbols (see Laboratory Manual for how-to).

Considering your design, answer the following questions:

- How many control pins for the control signals does your architecture have?
- How many different control signals does your architecture use to perform the desired tasks?
- Can you reduce the number of the control pins? Why not or how?
- Write down the sequence of the control signals for REVERSED LOAD operation. How many cycle does this operation take?

For this part, should submit hard copy of the sketch of your datapath design and your answers to the questions.

2 Experimental Work

2.1 Register File (35% Credits)

Load the register file module designed in the Preliminary Work Part **??** to the DE0-Nano board as a 4-bit register file. There are 4 switches in the DE0-Nano board and 8 switches in the DEES. You may use those switches as data and address inputs. For the other control signals and the clock, you may use the push buttons from either DE0-Nano board or DEES. For the two data outputs, you may use the LEDs of any module.

Verify the operation of the register file and demonstrate it to your lab instructor.

2.2 Datapath Design (65% Credits)

Load the custom architecture designed in the Preliminary Work Part **??** to the DE0-Nano board. Use the 8 switches of the DEES as the data input to the R0 register and use the 8 LED of any module to display the content of the R1 register. For your control signals and the clock, you may use the push buttons from either DE0-Nano board or DEES and the rest of the switches available. In case you need more switches or buttons, you may use external modules upon request.

Verify the operation of your design by performing the LOAD, INCREMENTED LOAD and REVERSED LOAD operations and demonstrate it to your lab instructor.

!!! Important Notice !!! When using DEES in your experiments, **DO NOT CONNECT VCC pins of DE0-Nano Board** to DEES's VCC terminal. Also, to make a common voltage reference, CONNECT GND of DE0-Nano Board to DEES's GND terminal. Please make sure that you turn all supplies OFF while making any connection throughout laboratory work.

3 Parts List

```
DE0-Nano Board
DEES
Oscilloscope
```