ELECTRICAL AND ELECTRONICS ENGINEERING
&
COMPUTER ENGINEERING

E E E 2 4 8 | C N G 2 3 2
L O G I C D E S I G N

2 1 | S p r i n g | 2 2

LABORATORIES

D r . G ü r t a ç Y e m i ş ç i o ğ l u
M b o n e a G o d w i n M j e m a
A h m e d M a m d o u h

4 TABLE OF CONTENTS

REGULATIONS

- Students are not permitted to perform an experiment without doing **the preliminary work** before coming to the laboratory. It is **not allowed** to do the preliminary work at the laboratory during the experiment. **Students, who do not turn in the complete preliminary work printout at the beginning of the laboratory session, cannot attend the lab. No "make-up" is given in that case.**

- No food or drink in the lab.

- There may be a quiz before each laboratory session, which will start promptly at the beginning of the lab. **Students who miss the quiz cannot attend the lab. No "make-up" is given in that case.** There won't be any extensions in the quiz time for latecomers. Therefore, students have to be at the lab on time for the quiz.

- Only the following excuses are valid for taking lab make-up:
  1. **Health Make-up:** Having a health report from METU Medical Center.
  2. **Exam Make-up:** Having an exam coinciding with the time of the laboratory session. The student needs to notify the instructor in advance if this is the case.

- Experiments will be done **individually**. The lab instructor will inform you of any part that you can do in groups.

- **Cheating or plagiarism is not tolerated. Plagiarism is a form of cheating as is using someone else's written word with minor changes and no acknowledgement. If you are caught cheating or plagiarising, you will at the very least receive a zero for the whole experiment. Disciplinary action may be taken.**

- Students who miss the lab **two times** without a valid excuse get zero as the laboratory portion of the course grade.

- *Those who fail to get a satisfactory score from the laboratory portion may fail the class. This score is expected to be 70% but may be adjusted up or down with the initiative of the course instructor.*

EXPERIMENT #4 & #5

SYNCHRONOUS BUILDING BLOCKS, SEQUENTIAL SYSTEMS AND TESTBENCHES

DUE WEEK 11

4.1 OBJECTIVE

The purpose of LAB 4 and 5 is to study basic synchronous sequential circuits (storage elements) to build more complex sequential circuits.

Students will use ModelSim® HDL simulation environment to test their design using testbenches. ModelSim® will be used to verify the functionality and the timing of synchronous designs and Altera Quartus II ISE 9.1 Project Navigator will be used to demonstrate your project on FPGA.

Students should prepare and submit a report before the LAB demo including all the codes, diagrams, simulation waveforms that they will demonstrate for each experiment mentioned below.

4.1.1 LAB 4

In the first part of the experiment simple synchronous storage elements with reset, load and shift functions will be studied using D type Flip-Flops (DFFs). These then will be used (as modules) to build various basic sequential circuits.

In the second part, a simple sequential system that can wrap-around and count up/down will be implemented.

4.1.2 LAB 5

In the third part of the experiment, knowledge and experience gained in the first part will be applied to design and test 3-bit counter using different type of flip-flops.

Then in the last part of the experiment you are required to design a simple sequential system that reads an 8-bit message from parallel input (user switches) into a shift-register and use the shift function together with a counter to determine the number of ‘1’s in the message. The number of 1’s will then be displayed on a 7segment display after getting decoded to a decimal.

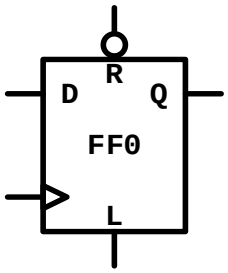
4.2 LAB 4

4.2.1 VERILOG MODELLING

Read through Section 4.4 to learn about Verilog HDL modelling of sequential circuits of synchronous type (with clock) and with asynchronous reset.

4.2.2 D FLIP-FLOP WITH ASYNCHRONOUS RESET AND SYNCHRONOUS LOAD

1. Draw a schematic to show how you would add combinational logic along with two new inputs (R and L) to a conventional D Flip-Flop to have the Reset and Load functions as shown in Figure 1**.** Note Load input take effect synchronously on the rising edge of the clock and Reset input is an active low input that takes effect asynchronously.



(a)

RST	LOAD	D	Q	CLK	Q^+	Operation
0	X	X	X	X	0	Asynchronous reset
1	1	X	X	↑	D	Load Data
1	0	X	X	↑	Q	Retain Value
1	X	X	X	0	Q	Retain Value
1	X	X	X	1	Q	Retain Value
1	X	X	X	↓	Q	Retain Value

(b)

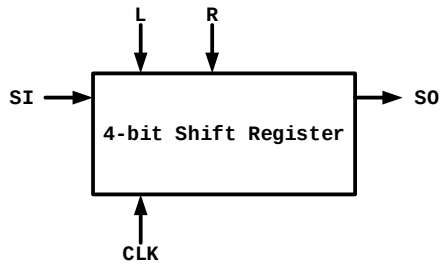
Figure 1: (a) New D Flip-Flop Symbol, (b) Condensed Truth Table for The New Flip-Flop.

2. Use your answer from 2.2.2.1 to modify the Verilog code provided for the D Flip-Flop in Section 4.4 to implement the new asynchronous reset and synchronous load functions.
3. Write a Verilog Testbench code to simulate and verify the functionality in ModelSim®.

4.2.3 4-BIT SHIFT REGISTER WITH ASYCHRONOUS RESET AND SYNCHRONOUS LOAD

1. Use the **new D Flip-Flop** in Figure 1 and design a 4-bit scalable bit-sliced synchronous shift register (with SI input and SO output) with synchronous Load and asynchronous Reset. The symbol and a table that describes the operation of the register are depicted in Figure 2.

2. The register has four input and one output. Write a structural (hierarchical) Verilog HDL code by explicitly declaring the D Flip-Flop created in 4.2.2. **Do not use if statement (behavioural) approach.**



(a)

R	SI	SO
1	X	0000
0	D_1	D_1000
0	D_2	D_2D_100
0	D_3	$D_3D_2D_10$
0	D_4	$D_4D_3D_2D_1$

(b)

Figure 2: (a) Symbol, (b) Table to Describe the 4-bit Shift Register Operation.

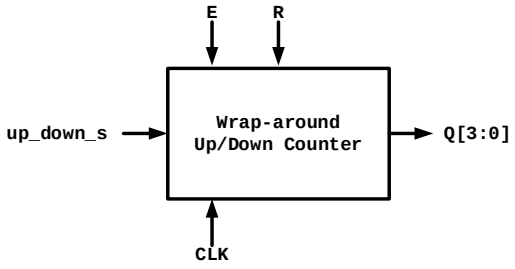
3. Use the following test patterns with the clock periods provided in Table 1. Write a Verilog Testbench code to simulate and verify the functionality in ModelSim®.

Table 1: Testbench Stimuli

R	L	SI	Clock Period
0	0	0	300 ns
1	1	1	300 ns
1	1	0	300 ns
1	1	1	300 ns
0	1	1	300 ns
1	1	0	300 ns
1	1	1	300 ns

4.2.4 4-BIT SYNCHRONOUS WRAP-AROUNG UP/DOWN COUNTER WITH SYCNHRONOUS UP/DOWN SELECT AND COUNT-ENABLE AND ASYNCHRONOUS RESET

1. Write a structural (hierarchical) Verilog HDL code by explicitly declaring the D Flip-Flop to create a scalable design of a 4-bit synchronous wrap-around up/down counter with synchronous up/down select and count-enable controls, and asynchronous reset as depicted in Figure 3. **Do not use if statement (behavioural) approach.**



(a)

R	up_down_s	E	Q[3:0]
1	X	X	0000
0	X	0	Q = Q
0	1	1	Q = Q+1
0	0	1	Q = Q-1

(b)

Figure 3: (a) Wrap-around Up/Down Counter Symbol, (b) Table to Describe the Wrap-around Up/Down Counter Operation.

2. Modify your code to connect a 4-bit output to 7-segment decoder designed and used in LAB 2&3.

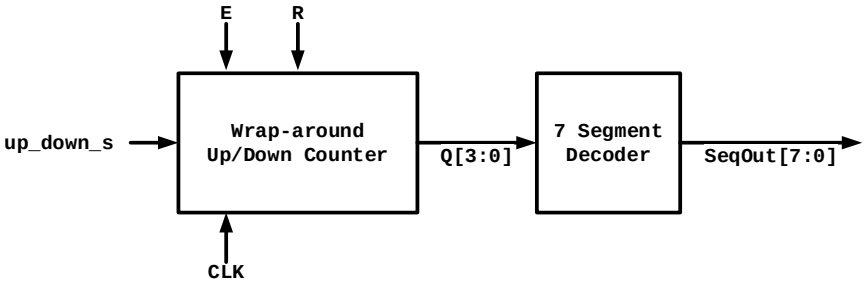


Figure 4: Block Diagram for The Modified Circuit Using 7 Segment Decoder.

3. Use the following test patterns with the clock periods provided in Table 2. Write a Verilog Testbench code to simulate and verify the functionality in ModelSim®.

Table 2: Testbench Stimuli

R	up_down_s	E	Clock Period
1	0	0	2000 ns
0	0	0	2000 ns
0	1	1	2000 ns
0	0	1	2000 ns

4.3 LAB 5

4.3.1 3-BIT DOWN COUNTER USING D-TYPE FLIP-FLOP

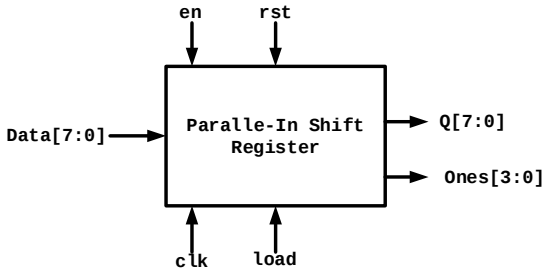
Design a 3-bit ***down*** counter with asynchronous clear using rising-edge triggered D-type flip-flops. The count should wrap around from “000” to “111”.

4.3.2 3-BIT DOWN COUNTER USING JK-TYPE FLIP-FLOP

Design a 3-bit down counter with asynchronous clear using negative-edge triggered JK-type flip-flops with asynchronous active low clear.

4.3.3 4-BIT SYNCHRONOUS PARALLEL LOAD SHIFT REGISTER WITH COUNTER AND ASYNCHRONOUS RESET

1. Write a structural (hierarchical) Verilog code to read an 8-bit message from parallel input (user switches) into a shift-register and use the shift function together with a counter to determine the number of '1's in the message. You have to create a top-level design and initialize shift register and counter explicitly. **Do not use if statement (behavioural) approach.** (no need for designing the 7-segment LED decoder here, the decoder will be used during the experimental work.)



(a)

rst	load	en	Q[7:0]	Ones[3:0]
1	X	X	0	0
0	X	0	0	0
0	1	1	Data	0
0	0	1	Data	#ones

(b)

Figure 5:(a) Parallel Load Shift Register Symbol, (b) table to describe the operation

2. Modify your code to connect a 4-bit output to 7-segment decoder designed and used in LAB 2&3.

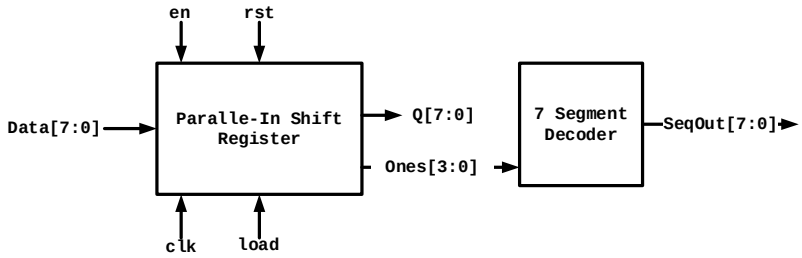


Figure 6: The Block Diagram for The Modified Circuit using 7-Segment Decoder.

3. Use the following test patterns with the clock periods provided in Table 2. Write a Verilog Testbench code to simulate and verify the functionality in ModelSim®.

Table 3: Testbench Stimuli

rst	load	en	Data	Delay
0	0	0	01010101	200 ns
1	1	1	01010101	200 ns
1	0	1	01010101	1000 ns
0	0	0	11010111	200 ns
1	1	1	11010111	200 ns
1	0	1	11010111	1000 ns
0	0	0	11000001	200 ns
1	1	1	11000001	200 ns
1	0	1	11000001	1000 ns

4.4 IMPLEMENTING SYNCHRONOUS LOGIC USING VERILOG HDL

4.4.1 D - LATCH

The *D* latch is said to be *transparent* because it responds to a change in data input with a change in the output as long as the enable input is asserted—viewing the output is the same as viewing the input. It has two inputs, *D* and *enables,* and one output, *Q*. Since *Q* is assigned value in behaviour, its type must be declared to be **reg.** Hardware latches respond to input signal *levels,* so the two inputs are listed without edge qualifiers in the sensitivity list following the @ symbol in the **always** statement. In this model, there is only one blocking procedural assignment statement, and it specifies the transfer of input *D* to output *Q* if enable is true (logic 1). Note that this statement is executed every time there is a change in *D* if *enable* is 1. Sample Verilog HDL code is given below.

```
// Description of D latch
module D_latch (Q, D, enable);

output Q;
input D, enable;
reg Q;

// level triggered
always @ (enable or D)
if (enable) Q <= D; // Same as: if (enable == 1)

endmodule
```

4.4.2 D FLIP-FLOP

In synchronous sequential circuits, changes in flip-flops occur only in response to a transition of a clock pulse. The transition may be either a positive edge or a negative edge of the clock, but not both. Verilog HDL takes care of these conditions by providing two keywords: **posedge** and **negedge**. The *D*-type flip-flop is the simplest example of a sequential machine. It responds only to the clock. Verilog HDL code given below defines a positive-edge-triggered D flip-flop. Output *Q* must be declared as a **reg** data type in addition to being listed as an output. This is because it is a target output of a procedural assignment statement. The keyword **posedge** ensures that the transfer of input *D* into *Q* is synchronized by the positive-edge transition of *Clk*. A change in *D* at any other time does not change *Q*.

```
// D flip-flop without reset
module D_FF (Q, D, Clk);

output Q;
input D, Clk;
reg Q;

// on the rising edge of the clock
always @ ( posedge Clk)
Q <= D;

endmodule
```

4.5 EXPERIMENTAL WORK

4.5.1 EXPERIMENTAL SETUP

- Verify to make sure your workbench has all of the following items:
- A Personal Computer (PC) with Altera Quartus II ISE 13.0 Service pack 1 Project Navigator
- DEO Demo Board with Cyclone III EP3C16F484C6 FPGA installed on a card with 10 input toggle switches, three pushbuttons, and four 7-Segment LED displays, among other components.
- A cable connected between the demo board and the PC using USB interface in order to transfer design information from the PC to the Demo Board.

4.5.2 4-BIT SHIFT REGISTER WITH ASYNCHRONOUS RESET AND SYNCHRONOUS LOAD

1. Create a project in ModelSim®, simulate your Verilog design and the testbench and check your waveform to verify the functionality of your design.
2. **Demonstrate the functionality of your design in ModelSim® to the lab instructor before proceeding.**
3. Launch Quartus II Project Navigator
4. Create a project named "**Shift_Reg**", Copy-Paste your Verilog codes from ModelSim®.
5. Compile your code.
6. Assign package pins and implement your design. When doing this, assign the clock input to a push button.
7. Generate the programming file, upload your design to the onboard FPGA, and test using the toggle switches to control all inputs except the clock, which is controlled by a push-button.
8. **Demonstrate the hardware-tested functionality of your design to the lab instructor. Does pressing of the push button generates a falling edge or a rising edge? What change can you do in the top-level design schematic so that the pressing of the push button generates the other edge?**

4.5.3 4-BIT SYNCHRONOUS WRAP-AROUND UP/DOWN COUNTER WITH SYNCHRONOUS UP/DOWN SELECT AND COUNT-ENABLED AND ASYNCHRONOUS RESET

1. Create a project in ModelSim®, simulate your Verilog and testbench and check your waveform to verify the functionality of your design.
2. **Demonstrate the functionality of your design in ModelSim® to the lab instructor before proceeding.**
3. Launch Quartus II Project Navigator
4. Create a project named "**Counter**", Copy-Paste your Verilog codes from ModelSim®.
5. Compile your code.
6. Create a 7-segment symbol via your code.
7. Using Schematic Capture connect your design to 7-segment.
8. Do package pin assignments, place and route, and hardware programming for this design in Quartus.
9. **Demonstrate the hardware-tested functionality of your design to the lab instructor.**

4.5.4 4-BIT SYNCHRONOUS PARALLEL LOAD SHIFT REGISTER WITH COUNTER AND ASYNCHRONOUS RESET

1. Create a project in ModelSim®, simulate your Verilog design and the testbench and check your waveform to verify the functionality of your design.
2. **Demonstrate the functionality of your design in ModelSim® to the lab instructor before proceeding.**
3. Launch Quartus II Project Navigator
4. Create a project named "**Count_Ones**", Copy-Paste your Verilog codes from ModelSim®.
5. Compile your code.
6. Assign package pins and implement your design. When doing this, assign the clock input to a push button.
7. Generate the programming file, upload your design to the onboard FPGA, and test using the toggle switches to control all inputs except the clock, which is controlled by a push-button.
8. **Demonstrate the hardware-tested functionality of your design to the lab instructor.**

4.6 REFERENCES

- DE0 Board User Manual, Terasic Technologies Inc.
- Digital Design with An Introduction to the Verilog HDL, 5th Edition, M. Morris Mano & Michael D. Ciletti, Pearson
- Quartus II Introduction Using Schematic Design, ALTERA.