**Laboratory 3 Counters**

**Prelab**

1. Complete block diagram of the top modules
2. complete all Verilog files
3. compile your designs
4. simulations(optional)

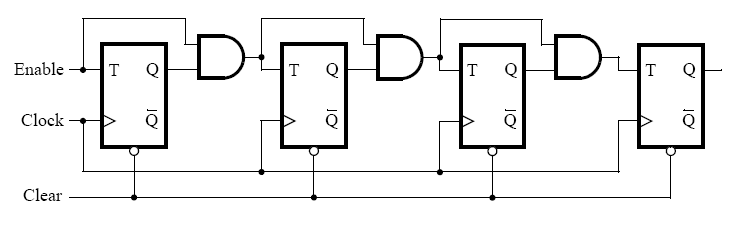
**Objectives**

1. Design counter using flip-flops.
2. Design counter using behavior statements.
3. Design a timer using clock frequency-division

**Part1 8-bit synchronous counter using T flip-flops and AND gates**

**Circuit description**

Consider the circuit in Figure 1. It is a 4-bit (**Q0 Q1 Q2 Q3**) synchronous counter which uses four T-type flip-flops. The counter increases its value on each positive edge of the clock if the Enable signal is asserted. The counter is reset to 0 by setting the Clear signal low. You are to implement an 8-bit counter of this type.



3

2

1

0

Figure 1. 4-bit synchronous counter (but you need to implement 8-bit counter in this lab)

Specific notes:

1. You should build a T flip-flop module tfliplfop (tff is a reserved key word, so it cannot be used as module name) first, and then instantiate **8 times** to create 8-bit counter with module name conter8.
2. Modify the 4 bit bcd to 7 segment display module BCD2LEDS from lab 2 to display 0 to F (instead of 0 to 9 for BCD numbers) with module name hex2seg7. Create two instances of hex2seg7 to display the 8-bit counter state Q7 Q6 … Q0 in double hex digits. One hex for Q7-4, and another for Q3-0
3. Use SW[1] for Enable, SW[0] for Clear, KEY[0] for clock.
4. The output of left TFF is Q[0] (LSB).
5. Counter state Q[7:0] also goes to red led display LEDR [7:0]
6. Q[7:0] should go through two 7-seg display so that the output of the 8-bit counter can be display on HEX1 and HEX0 as 00 to FF.

**prelab: complete the following Verilog files**

**Submodules**

1. T-flipflop: tflipflop.v

module tflipflop(resetn, t, clk, q); // available from lecture notes and textbook

// resetn: asynchronous low active reset control input

…

always @ (posedge clk, negedge resetn)

if (~resetn)

q<=0;

else

q<=t?~q: q;

…

endmodule

1. 8-bit counter: counter8.v

module counter8 (en, clear, clk, Q);

input en, clear, clk;

output [7:0] Q; // state of the 8 bit counter

…

// tflipflop x 8 instances

tflipflop u0(clear, T[0], clk, Q[0]); // TFF#0

…

// excitations

assign T[0]=en;

…

endmodule

1. HEX to 7-seg design: hex2seg7.v

to be modified from BCD to 7 segment display module available from Lab 2 (to display 0 to F instead of 0 to 9 only)

module hex2seg7(hex, seg7);

input [3:0] bin; // 4 bit binary input as one hex number

output reg [6:0] seg 7; // low active 7 seg LED output for 0 to F

always @ (\*)

case (bin)

4’b0000: seg7=?; // to display ‘0’

…

4’b1111: seg7=?; // to display ‘F’

endcase

endmodule

**Top Module**

module lab3part1(SW, KEY, LEDR, HEX1, HEX0);

input [1:0] SW; // SW[1] as enable input, SW[0] as asynchronous reset

input [0] KEY; // clock

output [7:0] LEDR; // binary counter state on LEDR

output [6:0] HEX1, HEX0; // two hex display of the counter state

wire [7:0] Q; // internal signals for stateQ7 Q6…Q0

// display 8 bit counter on LEDR directly

assign LEDR=Q;

// create instances of counter8 the 8 bit counter

//module counter8 (en, clear, clk, Q);

counter8(…);

// create 2 instances of display module

hex2seg7 H1(…); // display a hex # for bit 3-0 on HEX0

hex2seg7 H2(…); // bit 7-4 on HEX1

endmodule

**Implementation (put a check mark when you finish the step)**

1. \_\_\_\_\_\_Create a new project.
2. \_\_\_\_\_\_Write Verilog files (*lab3part1.v, counter8.v, tflipflop.v, hex2seg7.v*) required by the above description and notes and add them to the project.
3. \_\_\_\_\_\_Download *DE2\_pin\_assignments.qsf* from altera website.
4. In Quartus II, select **assignments 🡪 import assignments**, choose *DE2\_pin\_assignments.qsf* in the file box.
5. \_\_\_\_\_\_Compile your design.
6. **\_\_\_\_\_\_\_Tools🡪 programmer**, download the circuit into FPGA, and test the functionality.
7. \_\_\_\_\_\_\_Using Quartus II, find out how many logic elements (LEs) are used to implement your circuit.
8. \_\_\_\_\_\_\_Using Quartus II RTL viewer, find out how quartus II software synthesized your circuit.
9. Test the design a few times by changing SW[1], SW[0], KEY[0] (you are not expected to test 8 bit counter from 00 to FF manually) to observe the correct counting sequence on LEDR[7:0], and HEX1, HEX0.

**Trouble shooting tips:** you may need to test each submodule first (by creating a simple top module for testing purpose) before put everything together.

**Part2 8-bit synchronous counter using Q <=Q+1**

**Circuit description**

The functionality of the circuit is the same as part 1. The counter is accomplished using the statement: Q <= Q+1;

**prelab: complete the following Verilog code)]**

Design a submodule 8 bit counter

module counter8new(resetn, clk,en, Q);

input resetn, clk, en;

output reg [7:0] Q;

// always block for 8 bit counter with asynchronous low active reset

// and synchronous high active enable

always @ (posedge clk, negedge resetn)

if (!resetn) // low active

… // reset

else if (en)

… // count up by 1

else

… // no change

endmodule

Design the top module

module lab3part2(SW, KEY, LEDR,HEX1, HEX0);

input [1:0] SW; // SW[1] as enable, SW[0] as asynchronous reset

input [0:0] KEY; // KEY[0] as clock

output reg [6:0] HEX1, HEX0; // to display 8 bits

// (2 hex numbers) on two 7-seg display

output [7:0] LEDR; // to display counter state on LEDR

wire [7:0] Q; // internal signals for counter state

assign LEDR=Q; // optional: to have counter state on LEDR directly

// instantiate 8 bit counter

couner8new U0(…);

// instantiate display module twice

hex2seg7 U1(…);

hex2seg7 U2(…);

endmodule

// always block for 8 bit counter with asynchronous low active reset

// and synchronous high active enable

**Implementation**

1. \_\_\_\_\_\_\_Create a new project.
2. \_\_\_\_\_\_\_Write Verilog files (*lab3\_part2.v, counter8new.v )* required by the above description. Use *hex2seg7.v* from previous work.
3. \_\_\_\_\_\_\_Download *DE2\_pin\_assignments.qsf* from altera website.
4. \_\_\_\_\_\_\_In Quartus II, select **assignments 🡪 import assignments**, choose *DE2\_pin\_assignments.qsf* in the file box.
5. \_\_\_\_\_\_\_Compile your design.
6. **Tools🡪 programmer**, download the circuit into FPGA, and test the functionality.
7. \_\_\_\_\_\_\_Using Quartus II, find out how many logic elements (LEs) are used to implement your circuit.
8. \_\_\_\_\_\_\_Using Quartus II RTL viewer, find out how quartus II software synthesized your circuit.
9. Test the design a few times by changing SW[1], SW[0], and KEY[0] (you are not expected to test 8 bit counter from 00 to FF manually) to observe the correct counting sequence.

**Part3 a simple timer of 8-bit counter updated every second**

**Circuit Description**

The circuit, driven by a 50MHz clock (CLOCK\_50) (on-board), will display two-digit hexadecimal numbers from 00 to FF on two 7-seg displays HEX1 and HEX0 at a refreshing rate of 1Hz. The circuit consists of 3 parts: 1) 1 Hz clock signal generator; 2) 8-bit binary counter with reset (SW[0]) and enable (SW[1]); and 3) binary-seg decoder for the display (seg7.v in lab2).

The Verilog code for a frequency divider is provided as follows:

module clock\_div(input clk\_in, input[31:0] clkscale, output reg clk\_out);

// output clk\_out frequency = input cclk\_in frequency / (2\* clkscale)

reg[31:0] clkq=0; // 2^32-1>25M

always@(posedge clk\_in)

begin

clkq=clkq+1;// add 1, every cycle of clk\_in

if(clkq==clkscale)

begin; // toggle every clckscale cycles

clk\_out=~clk\_out

clkq=0; // reset counter

end

end

endmodule

The frequency of output clk is equal to the frequency of input cclk divided by (2\*clkscale). For example, if cclk frequency is 50MHz, clkscale should be 25,000,000 to generate 1Hz clk.

Your Verilog top module will instantiate the module clock\_div with clk1Hz as output and hex2seg7 (see part 1), and also instantiate counter8 (see part 2) with reset SW[0], enable SW[1] and clk11Hz as inputs.

**[prelab: complete the following Verilog code]**

module lab3part3(SW, CLOCK\_50, LEDR, HEX1, HEX0);

//SW[1]: enable (high active)

//SW[0]: resetn (low active)

input [1:0] SW;

input CLOCK\_50; // DE2 50MHz clock

wire clk1Hz; // user derived 1Hz clock

wire [7:0] Q; // internal state for the coutner

output …

…

// instantiate module clock to generate 1 Hz clock

clock\_div U0 (CLOCK\_50, 25000000,clk1Hz); // 50MHz input, 25M scale,

//1Hz output

// instantiate 8 bit counter with clk1Hz as clock

// SW[0] as low active reset, SW[1] as high active enable

…counter8…

// instantiate display module twice

…hex2seg7…

…hex2seg7…

endmodule

**Implementation**

1. \_\_\_\_\_\_\_Create a new project.
2. \_\_\_\_\_\_\_Write Verilog files (*lab3part3.v, clock\_div.v,)* required by the above description.Use files *counter8.v hex2seg7.v* from previous work.
3. \_\_\_\_\_\_\_Download *DE2\_pin\_assignments.qsf* from altera website.
4. \_\_\_\_\_\_\_In Quartus II, select **assignments 🡪 import assignments**, choose *DE2\_pin\_assignments.qsf* in the file box.
5. \_\_\_\_\_\_\_Compile your design.
6. **Tools🡪 programmer**, download the circuit into FPGA, and test the functionality by change SW[1] and SW[0] values.

1. Test your design fully as you see the counter changes its display from 00 to FF (it will take over 4 minutes for the whole counting sequence. So yo may do two things: 1. Veryify the accuracy of the 1 Hz freq or 1sec period using an external digital watch; 2. Increase clock frequency by changing the frequency divider scale constant so that you may see the whole counting sequence within a more reasonable time say 1 min)