FPGA System Designs and Practices (ET5009701)

Midterm Exam (Closed books and notes)

Date: December 1, 2021
Place: IB-410-1 and IB-410-2
Time limit: 120 minutes

This exam has seven problems in total. Please read each problem carefully and write your answers on the answer sheet.

1. (30%) Answer each of the following short questions:

   (a) What are the two basic constructs (or blocks) used in the behavioral style?

   (b) What are the three timing parameters you need to consider when using a flip-flop?

   (c) Describe the meaning of the dataflow style.

   (d) What is the meaning of the sampling window of a *D*-type flip-flop? What would happen when the sampling window is not satisfied?

   (e) What are the three ways in Verilog HDL that can be used to define constants?

2. (10%) Explain and correct the following clock generator:

   (a) Using both **always** and **initial** constructs

```
always begin
   initial clock = 1'b0;
   #5 clock = ~clock;
end
```

   (b) Using the **always** construct

```
always begin
   #5 clock = ~clock;
end
```

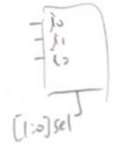
{!a[4],!a[3],a[2:0]}
{!a[4:3],a[2:0]}

3. (10%) Do the following two statements describe the same operations?

```
assign fifo_full = ({!gwr_ptr_next[n-1],!gwr_ptr_next[n-2],
                     gwr_ptr_next[n-3:0]} == grd_ptr);
assign fifo_full = ({!gwr_ptr_next[n-1:n-2],
                     gwr_ptr_next[n-3:0]} == grd_ptr);
```

4. (20%) Assume that we want to describe a 3-to-1 multiplexer with the following interface:

```
module mux_3to1 (
       input i0, i1, i2  // inputs
       input [1:0] sel,  // source selection inputs
       output out);      // output
// your answer is from here
......
endmodule
```



```
case (sel)
  2'b00: out=i0;
  2'b01: out=i1;
  2'b11: out=i2;
```

Write a Verilog HDL module to describe this 3-to-1 multiplexer according to the ways specified as follows:

(a) In behavioral style with a **case** statement

(b) In behavioral style with an **if-else** statement

5. (20%) Consider a modulo-3 counter that generates one output pulse per three input pulses. Assume that the input pulse (clk) is a 50-% duty-cycle signal, and the output pulse ($qout$) must be also 50-% duty-cycle, as depicted in Figure 1.

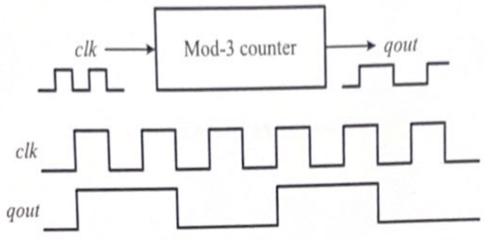


Figure 1: Block diagram and waveform of Problem 5.

(a) Design this modulo-3 counter and explain the rationale behind your design.

(b) Plot the resulting logic diagram of the modulo-3 counter.

(c) Write a Verilog HDL module to describe the resulting modulo-3 counter.

6. (10%) In this problem, you are asked to describe a register file with two read ports and one write port. The register file has the following features: The access to each read port is unconditional and asynchronous, and the access of the write port is controlled by a write enable signal, *wr_enable*, in synchronism with the clock (*clk*) signal. Each port regardless of read or write has its own access address.

```
// an N-word register file with one write port and two read ports
module register_file
       #(parameter M = 4,  // default number of address bits
         parameter N = 16, // default number of words,N=2**M
         parameter W = 8)( // default word width
         input  clk, wr_enable,
         input  [W-1:0] din,      // data input
         input  [M-1:0] rd_addra, rd_addrb, // read addresses
         input  [M-1:0]  wr_addr, // write address
         output [W-1:0] douta, doutb); // port_a and port_b outputs
reg [W-1:0] reg_file [N-1:0];
// your answer is from here

endmodule
```

7. (10%) Write a Verilog HDL module to describe a code converter that converts an excess-3 code to a BCD code. The code converter works as follows. It first checks to see whether the input codeword is a valid excess-3 code or not. If yes, it converts the input codeword into its equivalent BCD codeword by subtracting `4'b0011` from the input codeword and sets the `valid` signal true to indicate this situation; otherwise, it directly passes the input codeword to the output and sets the `valid` signal false.

```
module excess3_to_BCD(
        input  [3:0] x,  // excess-3 input
        output [3:0] y,  // BCD output
        output valid);   // indicate a valid excess-3 input
 // the body of the excess-3-to-BCD converter
 assign valid =   // complete this assignment using the ?: operator
 assign y =       // complete this assignment using the ?: operator
 endmodule
```

—— **End of exam** ——