*Name: Halwa

*Registration: Sooji Ka

Department of Computer Systems Engineering
University of Engineering & Technology Peshawar

Digital System Design

CSE 308

Midterm Examination Spring 2018

10 April 2018, Duration: 120 Minutes

Exam Rules

Please read carefully before proceeding.

1- This exam is CLOSED books/notes/computers/mobiles. However, one A4 size handwritten cheat sheet is allowed.
2- Write your name and registration number on the cheat sheet. Attach the cheat sheet with the answer sheet for up to 5 bonus points.
3- Sharing of cheat sheet and other materials during the exam is not permitted.
4- No calculators of any kind are allowed.
5- Answer all problems on the problem sheet.
6- There are 4 problems in total. Some problems are harder than others. Answer the easy ones first to maximize your score.
7- Questions will not be interpreted during the exam.
8- This exam booklet contains 8 pages, including this cover. Count them to be sure you have them all.

Problem 1	_____	(22 pts)
Problem 2	_____	(18 pts)
Problem 3	_____	(20 pts)
Problem 4	_____	(20 pts)
Exam Total	_____	(80 pts)

Good Luck!

Problem 1. (22 pts).

(a) (4 pts) Give the result of each Verilog expression (in binary) for the following inputs: A=4'b0011, B=3'b011 and C=3'b101. Assume A is a 4-bit wire and B and C are each 3-bit wires. Show your results using Verilog notation, such as 3'b101.

1. A + (B | C); 4'b1010
2. ~& A; 1'b1
3. (A = = B) ? B : C; 3'b011
4. {A, {2{C}}; 10'b0011101101

(b) (2 pts) Briefly explain how a **wand** differs from a **wor**.

A **wand** is forced to zero if any driver to it is zero, while a **wor** is forced to one if any driver to it is one.

(c) (2 pts) Write a Verilog statement that declares a 6-bit constant, C_24, with the decimal value 24.

```
parameter C_24 = 6'd24;
```

(d) (2 pts) Write Verilog code that declares an 8-bit register, R_H38, and initially assigns it the hexadecimal value 38.

```
reg [7:0] R_H38;
initial R_H38 = 8'h38;
```

(e) (4 pts) Write a single Verilog statement that declares a 12-bit by 16-word memory called mem1. Also, write a Verilog code segment that assigns the fifth word of mem1 the decimal value 127 on the positive edge of the signal **clock**.

```
reg [11:0] mem1 [0:15];
always @(posedge clock) mem1[4] <= 127;
```

(f) (4 pts) Give a simplified **Boolean equation** that is equivalent to the following Verilog statement, assuming that a, b, c and z are each one-bit wires.

```
assign z = c ? (a ? b : c) : b;
```

z = b | (~a)&c

z = c& (a&b | (~a)&c) | (~c)&b
z = a&b&c | (~a)&c | (~c)&b
z = b | (~a)&c

(g) (2 pts) Use structural Verilog to specify a 4-input AND gate named AND4 with inputs A, B, C, D, output Z, and a delay of 2 time units.

```
and #2 AND4(Z, A, B, C, D);
```

(h) (2 pts) What advantage is there to connecting ports by name, rather than position?

Many solutions accepted, but the primary reason is that it is more likely that wires will be connected to the correct port (i.e. it is harder to make a mistake).

Problem 2. (18 pts). ……(CLO-1)

For this problem, you will complete the Verilog code for a combinational 4-bit decrementer. Your decrementer must subtract 1 from the 4-bit number, x, to produce the 4-bit number, y, and a negative flag n, where n is 1'b1 when y is negative (i.e., when x = 4'b0000 and y = 4'b1111). Assume both x and y are unsigned binary numbers.

(a) (10 pts) For this part of the problem, you must use structural Verilog and instantiation on the module:

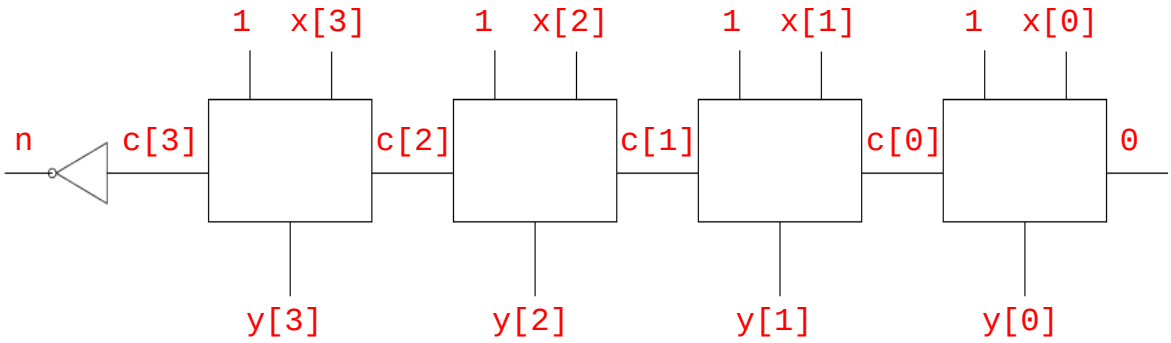
fulladder (input a, b, cin, output sum, carry)

to create the decrementer, where sum = a ^ b ^ cin and carry = (a & b) | (a & cin) | (b & cin). Fill in the module given below. In addition to using instantiation, you may also instantiate primitives. Draw the hardware for your decrementer in the space after the module.

```
module decrementer(input [3:0] x, output [3:0] y, output n);

    wire [3:0] c;
    fulladder FA0 (1'b1, x[0], 1'b0, y[0], c[0]);
    fulladder FA1 (1'b1, x[1], c[0], y[1], c[1]);
    fulladder FA2 (1'b1, x[2], c[1], y[2], c[2]);
    fulladder FA3 (1'b1, x[3], c[2], y[3], c[3]);
    not(n, c[3])

endmodule;
```



(b) (8 pts) For this part of the problem, you must use behavioral Verilog to create the decrementer. Fill in the module given below.

```
module decrementer(input [3:0] x, output reg [3:0] y, output reg n);

     always@(x) begin
          y = x-1;
          n = (y == 4'b1111);
     end

endmodule;
```

Problem 3. (20 pts). ……(CLO-1)

Below is an RTL (or dataflow) description for a circuit.

```
module RTL_circuit(x, y, a, b, c, d);
     parameter x_delay = 3, y_delay = 7;
     input a, b, c, d;
     output x, y;
     assign #y_delay y = (a | b) & (c | ~d);
     assign #x_delay x = a ^~ b;
endmodule;
```

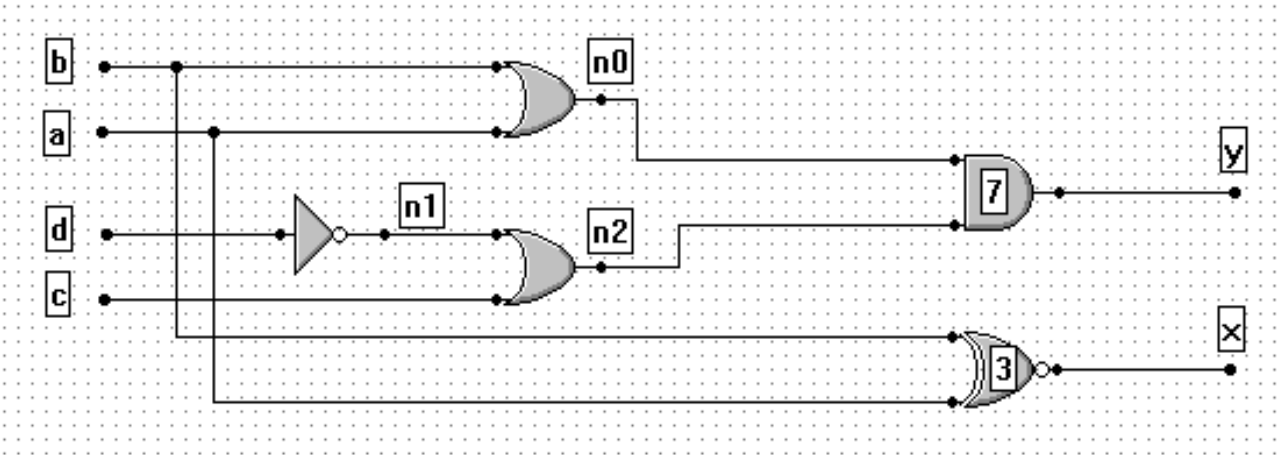
(a) (3 pts) Give a Verilog statement that instantiates the above RTL_circuit, with the instance name RTC, so that x has a delay of 7 time units and y has a delay of 5 time units. When you instantiate the circuit, use the same names for wires as is used in the module port list.

```
RTL_circuit #(7, 5) RTC (x, y, a, b, c, d);
```

(b) (6 pts) Rewrite the RTL_circuit using Verilog built-in primitives and structural Verilog. Your design should have the same overall delays (from module inputs to module outputs) as the original code. It is fine to let some gates have zero delay. Part of the module is done for you.

```
module struct_circuit(x, y, a, b, c, d);
     parameter x_delay = 3, y_delay = 7;
     input a, b, c, d;
     output x, y;
     wire n0, n1, n2;
     or (n0, a, b);
     not(n1, d);
     or (n2, n1, c)
     and #y_delay (y, n0, n2);
     xnor #x_delay (x, a, b);
endmodule;
```

(c) (3 pts) Draw a gate-level diagram for your module in **(b)**. Label all nets on the diagram and write the delay of each gate inside the gate.



Answers vary based on design in **(b)**.

(d) (6 pts) Rewrite the RTL_circuit using behavioral Verilog. Your design should have the same delays as the original code. Part of the module is done for you.

```
module behave_circuit(x, y, a, b, c, d);
     parameter x_delay = 3, y_delay = 7;
     input a, b, c, d;
     output x, y;
     reg x, y;
     always @(a, b, c, d) begin
          #y_delay y <= (a | b) & (c | ~d);
          #x_delay x <= a ^~ b;
     end
endmodule;
```

(e) (2 pts) Show how to use the ‘timescale directive so that the x_delay and y_delay correspond to 0.3 ns and 0.7 ns, respectively and the simulator time step is 1 ps.

```
‘timescale 100 ps / 1 ps
```

Problem 4. (20 pts). ……(CLO-2)

(a) (10 pts) Below is Verilog code for a common digital circuit.

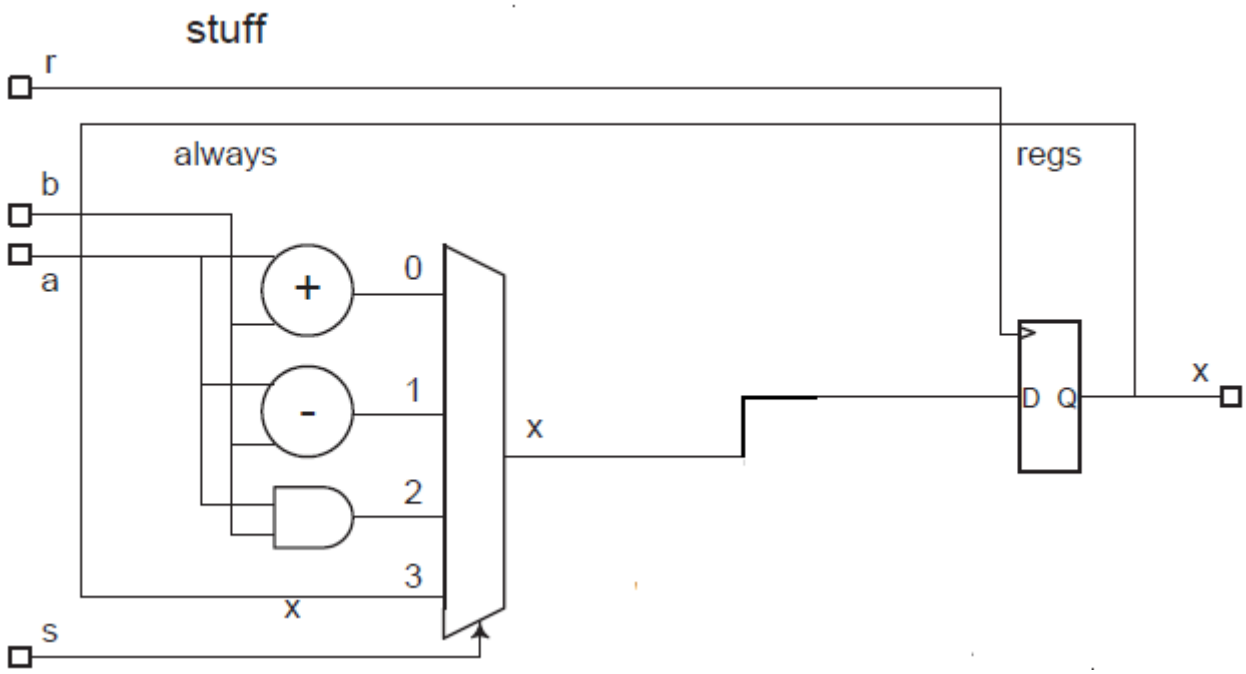
```
module circuit(clk, rst, en, q);
     input clk, rst, en;
     output [3:0] q;
     reg [3:0] q;
     always@(posedge rst or posedge clk) begin
          if (rst == 1’b1)
               q <= 4’b0000;
          else if ((clk == 1’b1) and (en == 1’b1))
               q <= q + 4’b0001;
          end
endmodule;
```

Briefly tell what the circuit is or does. Are the rst and the en signals synchronous or asynchronous?

This circuit corresponds to a positive edge-triggered 4-bit up counter with enable and reset signals. The rst signal is asynchronous and en signal is synchronous.

(b) (10 pts) Show the hardware that would be synthesized for the module below. Be sure to show the module ports and registers, if any.

```
module stuff(x, s, r, a, b);
     input [7:0] a, b;
     input [1:0] s;
     input r;
     output x;
     reg [7:0] x;
     always @( posedge r ) begin
          case( s )
               0: x = a + b;
               1: x = a - b;
               2: x = a & b;
          endcase
     end
endmodule;
```



<This page is intentionally left blank. This page can be used for scratch work or as extra space. If you write work here that you want me to grade, be sure to clearly indicate which problem(s) the work corresponds to!>