Seat Number .....

Name .....Student ID..... 20/12/2017

Final Examination Semester 1 Year 2017

Course: CPE 223 Digital Electronics and Logic Design **Group 31, 32**

Date: 4, December, 2017 13.00-16.00

Instruction

1. Calculator, books, documents, and notes are not allowed in the examination room.

2. Carefully read the explanation in each problem and then answer each question.

3. Do not take the examination sheets out of the examination room.

4. This examination has 12 pages including this page (6 problems, 40 points).

- Student must raise your hand to ask for permission before leaving the room.
- Student must not take the exam and booklet outside the room.
- The highest punishment can be applied if the cheating is discovered.

(Asst. Prof. Suthathip Maneewongvatana, Ph.D.)

This exam is approved by the Computer Engineering Department's committees

(Assoc. Prof. Dr. Natasha Dejdumrong)

Program chair

Date.....Month.....Year.....

1. Below is a gate-level design for a combinational circuit. **(4 points)**

```
module gate(
    input x,
    input y,
    output z
    );
        wire xx, yy, p,q;
        not(xx, x);
        not(yy, y);
        and(p, x,y);
        and(q, xx,yy);
        or(z,p,q);

endmodule
```

a) Draw a schematic diagram of this circuit and label all nets on the diagram. (2 points)

b) Rewrite the gate level circuit using dataflow design of Verilog. (2 points)

```
module gate_df(
    input x,
    input y,
    output z
    );
        // your code here




endmodule
```

2. Implementing the combinational circuit from multiplexers. **(6 points)**

a) Write a Verilog module for implementing a multiplexer-based represented in Figure1. (2 points)

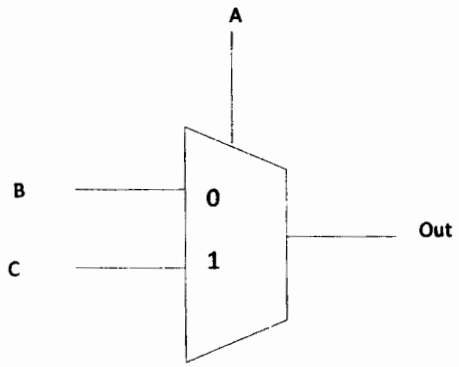

Figure 1

```
module mux(A, B, C, out);

        input A, B, C;
        output Out;

// your code here









endmodule
```

b) Given the truth table for a combinational circuit below. Implement the Verilog module for this combinational circuit using the mux module in a). (4 points)

w1	w2	w3	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

```
module combination(w1,w2,w3, f);

        input w1,w2,w3;
        output f;

// your code here









endmodule
```

3. Design the control unit in vending machine. This machine receives two types of coins 5฿ and 10 ฿ (logic 0 for 5฿ and logic 1 for 10฿). The machine receives one coin at a time. The output gate will open only when the total amount equals to 15฿.
(10 points)

a) Draw State Diagram for this control unit with Mealy Machine Design (4 points)

b) Write the Verilog code for this unit. (6 points)

```
module mealy( clk, inp, outp);
  input clk, inp;
  output outp;
  reg [_____] state;
  reg outp;
  always @( posedge clk)
    begin
// your code here
```

```
        end
endmodule
```

4. Determine the value of output x, y, and z after running these two Verilog modules. Given the initial values for inputs and outputs are a=0, b=0, c=1, x=1, y=0, z=1. **(4 points)**

module blocking(a,b,c,x,y,z);
input a,b,c;
output x,y,z;
reg x,y,z;
 	
always @ (a or b or c)
begin
x= b & c;
y= a | x;
z=y & x;
 	
end
 	
endmodule
	module nonblocking(a,b,c,x,y,z);
	input a,b,c;
	output x,y,z;
	reg x,y,z;
	
	always @ (a or b or c)
	begin
	x<= b & c;
	y<= a | x;
	z<=y & x;
	
	end
	
	endmodule

x= **y=** **z=**	**x=** **y=** **z=**

5. Given Asynchronous ripple counter implemented by JK flip flops in Figure 2. **(10 points)**

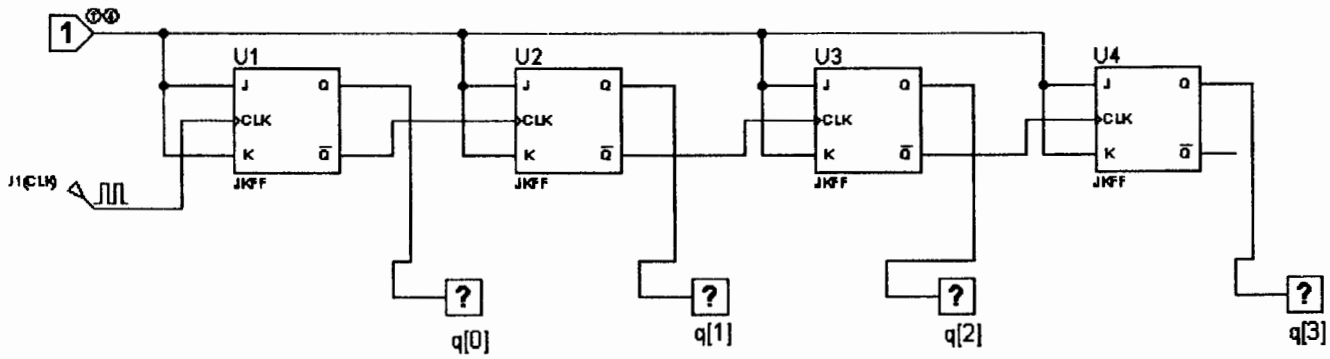

Figure 2.

a) Write Verilog module for a JK flip flop (5 points)

```
module JKFF (J,K, Clock, q, qbar);
        input J, K, Clock;
        output reg q;
        output qbar;
// your code here




endmodule
```

b) Write Verilog module for Asynchronous ripple counter using JK flip flop module above.

(5 points)

```
module ripplecounter (clock, b);
        input clock;
        output [3..0] q;
        reg [3..0] qb;
// your code here









endmodule
```

6. Given the input clock signal with frequency 8 Hz, write Verilog signal to generate the new clock signal with frequency 2 Hz. **(6 points)**

```
module clock2hz (clock);

        input clock10;

        output clock2;

        reg [2..0] count ;

  always @( posedge clock10)

        begin
// your code here









        end

endmodule
```