

2560

Seat Number

Name Student ID.....



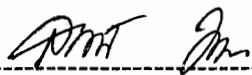
Final Examination Semester 2 Year 2559

Course: CPE 223 Digital System Design For computer Engineering Students Year

Date: 8, May, 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 8 pages including this page (6 problems, 30 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.)

ข้อสอบนี้ได้ผ่านการพิจารณาจากคณะกรรมการภาควิชาวิศวกรรมคอมพิวเตอร์



(รศ.ดร.พิรพล ศิริพงษ์อุดมกร)

ประธานหลักสูตร

28 เม.ย. 2560

วันที่.....เดือน.....พ.ศ.....

1. Complete the following clk_gen module, which generates a clock signal that initially goes to zero for 15 ns, then goes to one for 5 ns, and then repeats this pattern indefinitely. (4 points)

```
module clk_gen;  
    reg clock;  
    initial begin
```



```
        end  
    endmodule;
```

2. Below is a gate-level design for a circuit.

(6 points)

```
module Struct_circuit(x, y, a, b, c, d);  
    input a, b, c, d;  
    output x, y;  
    signal n0, n1, n2, n3;  
    or (n0, a, b);  
    not(n1, d);  
    or (n2, n1, c);  
    and (y, n0, n2);  
    xnor (x, a, b);  
endmodule;
```

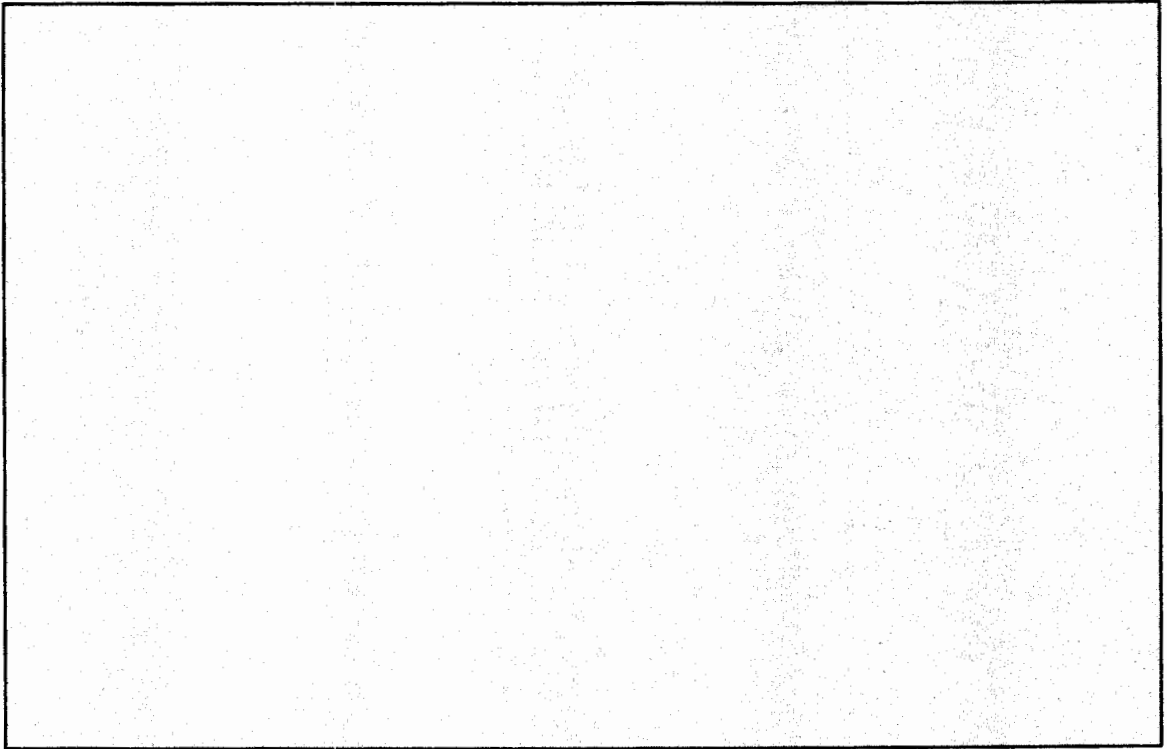
a) Draw a schematic diagram of this circuit and label all nets on the diagram. (2 points)

b) Rewrite the gate level circuit using behavioral design of Verilog. (4 points)

```
module Behave_circuit(x, y, a, b, c, d);
```

```
    input a, b, c, d;
```

```
    output x, y;
```



```
end module
```

3. Give the behavioral design of a circuit below. (4 points)

```
module circuit(clk, rst, en, q);  
    parameter delay = 6;  
    input clk, reset, en;  
    output [3:0] q;  
    reg [3:0] q;  
    always@(posedge reset or posedge clk) begin  
        if (reset == 1'b1)  
            q <= #delay 4'b0000;  
        else if ((clk == 1'b1) and (en == 1'b1))  
            q <= #delay q + 4'b0001;  
        end  
    endmodule;
```

a) Draw the top level structure of this circuit representing the input and output signals. You need to specify the data size of each signal. (1 points)

b) Explain the function of this circuit. (3 points)

4. Write a Verilog logic implemented as a multiplexer-based represented in Figure1.
(4 points)

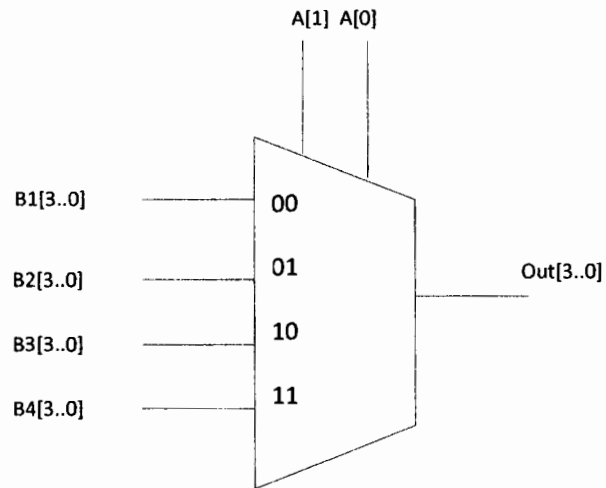


Figure 1

```
module mux(A, B, out);  
    input[1..0] A;  
    input[3..0] B1,B2,B3,B4;  
    output[3..0] Out;
```

```
endmodule
```

5. Give the Verilog code for an FSM below.

(5 points)

```
module Fido(CLK,RESET,MOVE,STATE);
    input CLK,RESET,MOVE;
    output reg [2:0] STATE;
    always @(posedge CLK) begin
        if (RESET) STATE <= 3'b000;
        else case (STATE)
            3'b000: STATE <= MOVE ? 3'b001 : 3'b000;
            3'b001: STATE <= MOVE ? 3'b100 : 3'b010;
            3'b010: STATE <= MOVE ? 3'b100 : 3'b011;
            3'b011: STATE <= MOVE ? 3'b000 : 3'b011;
            3'b100: STATE <= MOVE ? 3'b101 : 3'b111;
            3'b101: STATE <= MOVE ? 3'b110 : 3'b011;
            3'b110: STATE <= MOVE ? 3'b110 : 3'b111;
            3'b111: STATE <= MOVE ? 3'b101 : 3'b001;
            default: STATE <= 3'b000;
        endcase
    end
endmodule
```

a) Draw the state transition diagram that represents the current state, the input, and next state. (4 points)

b) If the initial state is "000" and the sequence of MOVE for each clock cycle is 00111, what is the current state after finish the end of MOVE sequence. (1 points)

6. Write the Verilog code for 4 bits adder that can present flags for detecting overflow and zero result. Given a and b are 4 bit-input, c is the 4 bit-result, o is flag for overflow detection, and z is flag for zero detection. All flags will be 1, if the overflow or zero is detected. (7 points)

module fourbit_adder(c,o,z,a,b);

endmodule