April 4, 2013

COMPUTER ENGINEERING DEPARTMENT

COE 405

DESIGN & MODELING OF DIGITAL SYSTEMS

Midterm Exam

Second Semester (122)

Time: 1:00-3:30 PM

OPEN BOOK EXAM

Student Name : __KEY____

Student ID. : ____

Question	Max Points	Score
Q1	20	
Q2	15	
Q3	30	
Q4	25	
Q5	10	
Total	100	

Dr. Aiman El-Maleh

[20 Points]

(Q1) It is required to design an **iterative** combinational circuit that receives an **n-bit** number X and computes the result Y=3*X. The design should be based on a one-bit cell (i.e. processing one bit X_i) that can be copied n times to construct the n-bit design. (Hint: 3*X = X+X+X. Note that the maximum carry from one cell to the next is 2.)

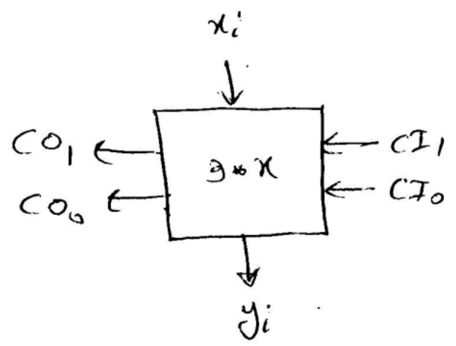
i) Determine the inputs and outputs for a one-bit cell.

ii) Derive the truth table for a one-bit cell.

iii) Derive minimized sum-of-product equations for a one-bit cell.

iv) Show the block diagram for a 4-bit design.

i)



ii) Truth Table :

x_i	CI_1	CI_0	CO_1	CO_0	y_i
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	x	x	x
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	x	x	x

iii)

x_i \ CI_1CI_0	00	01	11	10
0	0	0	x	0
1	0	1	x	1

$$CO_1 = x_i CI_1 + x_i CI_0$$

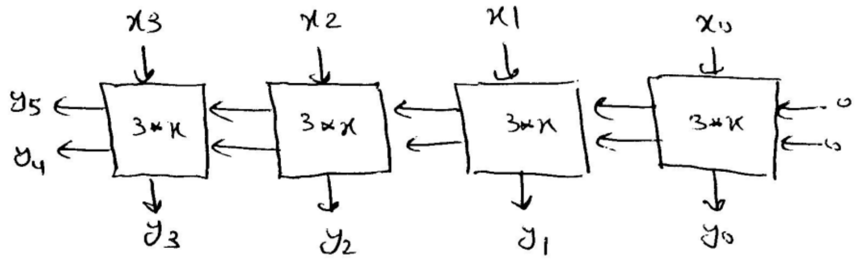
x_i \ CI_1CI_0	00	01	11	10
0	0	0	x	1
1	1	0	x	0

$$CO_0 = \overline{x}_i CI_1 + x_i \overline{CI_1}\, \overline{CI_0}$$

x_i \ CI_1CI_0	00	01	11	10
0	0	1	x	0
1	1	0	x	1

$$y_i = \overline{x}_i CI_0 + x_i \overline{CI_0}$$

iv)

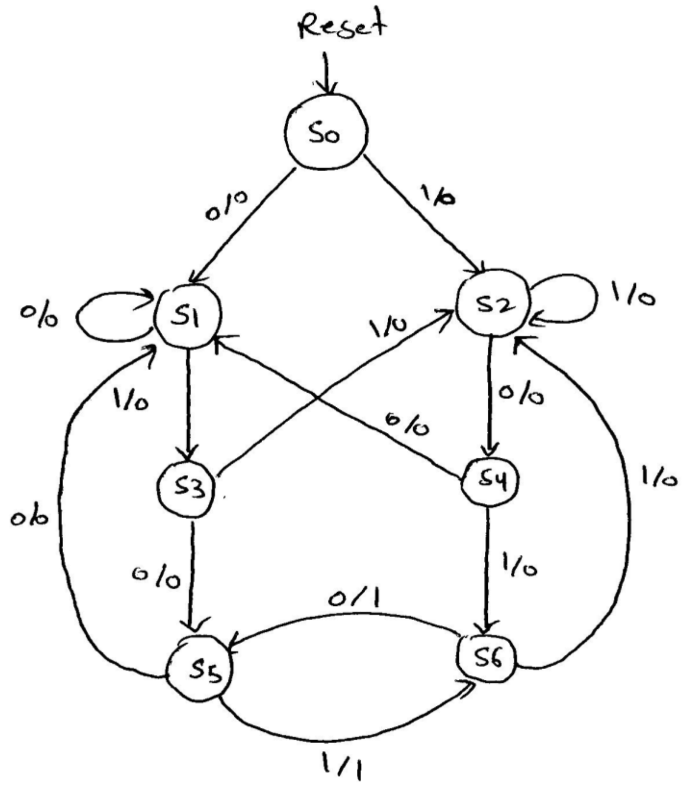


[15 Points]

(Q2) It is required to design a synchronous sequential circuit that receives a serial input **X** and produces a serial output **Z**. The output Z will be 1 if the input has been **alternating for at least 3 clock periods**. Assume the existence of an asynchrnous reset input to reset the machine to a reset state. Draw the state diagram of the circuit assuming a **Mealy** model. *You are not required to derive the equations and the circuit*. The following is an example of some input and output streams:

Example:

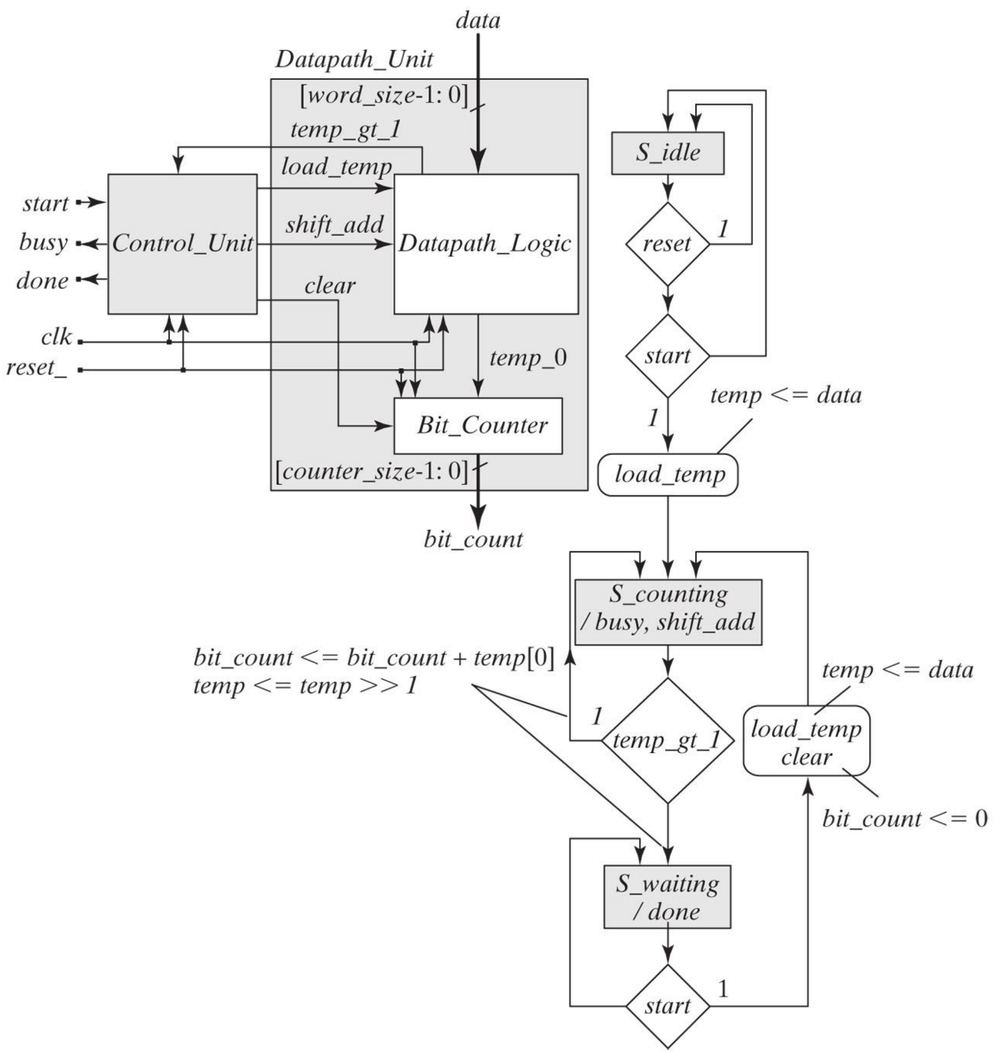
Input	**X**	1 0 1 0 1 0 1 1 0 1 0 1 0 0 1 0
Output	**Z**	0 0 0 1 1 1 1 0 0 0 1 1 1 0 0 0



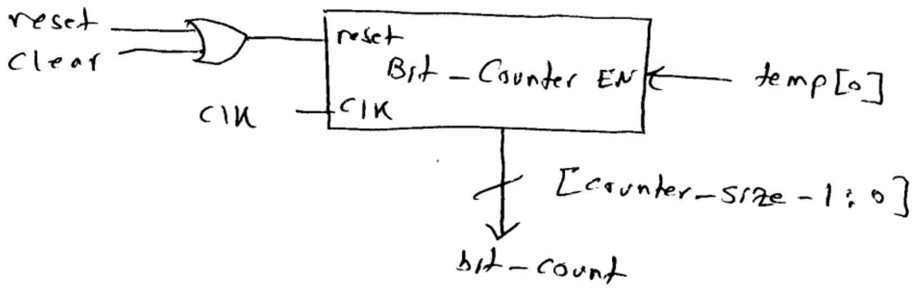
[30 Points]

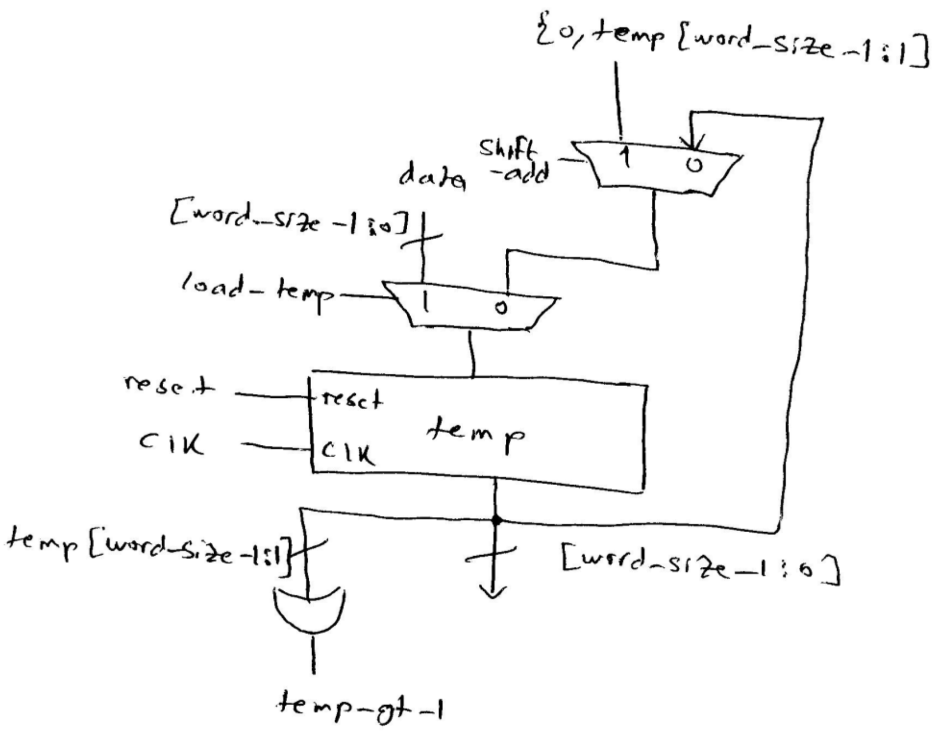
(Q3) The ASMD chart given below describes a state machine that counts 1's in a word and terminates activity as soon as possible. The machine remains in its reset state, *S_idle*, until an external agent asserts *start*. This action asserts the output, *load_temp*, which will cause *data* to be loaded into register *temp* when the state makes a transition to *S_counting* at the next active edge of *clk*. The machine remains in *S_counting* while *temp* contains a 1. Two actions occur concurrently at each subsequent clock: (1) *temp* is shifted towards its LSB and (2) *temp[0]* is added to *bit_count*. When *temp* finally has a 1 in only the LSB, the machine's state moves to *S_waiting*, where *done* is asserted. The state remains in *S_waiting* until *start* is reasserted. Assume that when the synchrnous *reset* input is asserted the machine is reset to the state *S_idle* and *bit_count* and *temp* are initialized to 0.

- **i)** Show the design of the data-path unit.
- **ii)** Show the design of the control unit using the following state assignment: *S_idle=00, S_counting =01*, and *S_waiting=10*.



i) **DataPath Unit**:





ii) Control Unit:

C.S.	Input		N.S.	Output				
	start	temp_gt_1		done	busy	load_temp	shift_add	clear
S_idle	0	x	*S_idle*	0	0	0	0	0
S_idle	1	x	*S_counting*	0	0	1	0	0
S_counting	x	1	*S_counting*	0	1	0	1	0
S_counting	x	0	*S_waiting*	0	1	0	1	0
S_waiting	1	x	*S_counting*	1	0	1	0	1
S_waiting	0	x	*S_waiting*	1	0	0	0	0

Assuming the two flip flops F1 and F2 for storing the machine state and the state assignment: *S_idle=00, S_counting =01*, and *S_waiting=10*, the contrl signals will be as follows:

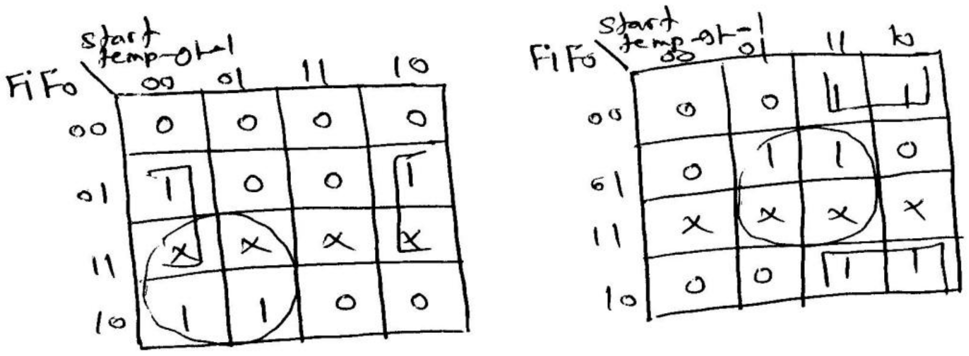
done = F1 F0’

busy = F1’ F0

shift_add = F1’ F0

load_temp = F0’ start

clear = F1 F0’ start



F1F0 \ start temp-gt-1	00	01	11	10
00	0	0	0	0
01	1	0	0	1
11	x	x	x	x
10	1	1	0	0

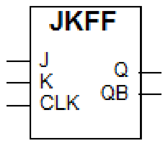
F1F0 \ start temp-gt-1	00	01	11	10
00	0	0	1	1
01	0	1	1	0
11	x	x	x	x
10	0	0	1	1

$$D1 = F1\ \overline{Start} + F_0\ \overline{temp\text{-}gt\text{-}1}$$

$$D_0 = \overline{F_0}\ start + F_0\ temp\text{-}gt\text{-}1$$

[25 Points]

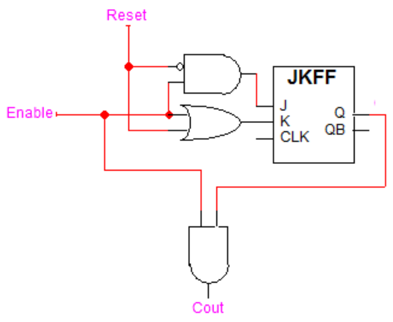
(Q4) A JK flip-flop with the following interface is given below:



i) Write a JKFF module with <u>behavioral</u> model for a rising-edge triggered JK flip-flop.

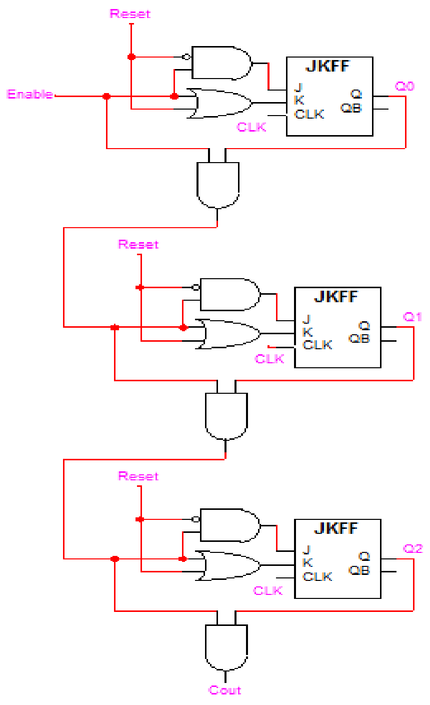
```
module JKFF (output reg Q, output QB, input J, K, CLK);
assign QB = ~Q;
always @( posedge CLK)
        case ({J,K})
                2'b00: Q <= Q;
                2'b01: Q <= 0;
                2'b10: Q <= 1;
                2'b11: Q <= ~Q;
        endcase
endmodule
```

ii) Using the JK flip-flop modeled in (i) model the 1-bit modular binary counter given below.



```
module OneBitCount (output  Q, Cout, input Reset, Enable, CLK);
        and (J, ~Reset, Enable);
        or (K, Reset, Enable);
        and (Cout, Enable, Q);
        JKFF F (Q, , J, K, CLK);
endmodule
```

iii) Based on the 1-bit modular binary counter modeled in (ii), build a 3-bit binary up counter as shown below. The counter does not count when the count enable is 0. Otherwise, it counts on the rising-edge of the clock.



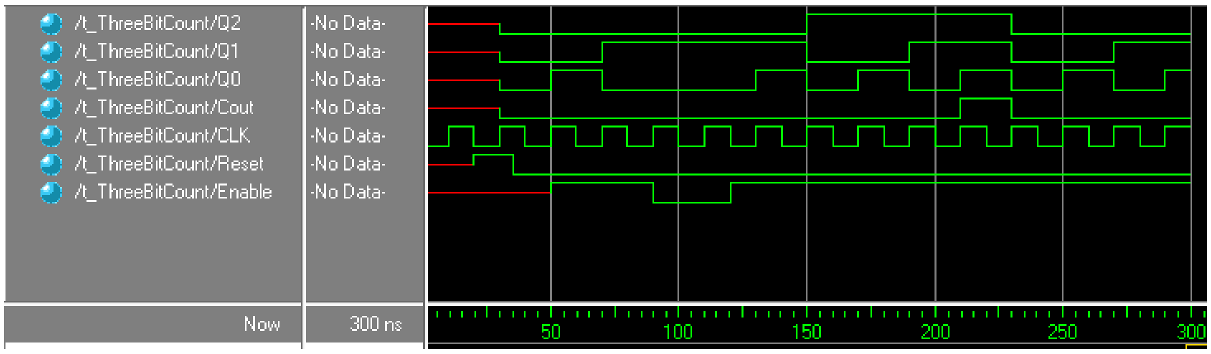
```
module ThreeBitCount (output  Q2, Q1, Q0, Cout, input Reset, Enable, CLK);
    OneBitCount F0 (Q0, Cout0, Reset, Enable, CLK);
    OneBitCount F1 (Q1, Cout1, Reset, Cout0, CLK);
    OneBitCount F2 (Q2, Cout, Reset, Cout1, CLK);
endmodule
```

iv) Write a test bench to test the 3-bit **counter** using the data given below. You need to generate the clock signal inside the test bench with a period of 10 and 50% duty cycle. The test bench should simulate the circuit for 300 delay units.

Time	Action
20	Reset=1
35	Reset=0
50	Enable=1
90	Enable=0
120	Enable=1

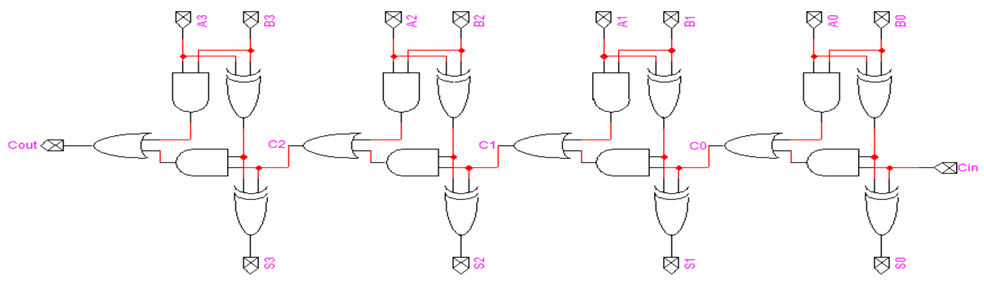
```
module t_ThreeBitCount ();
    wire Q2, Q1, Q0, Cout;
    reg CLK, Reset, Enable;
    ThreeBitCount C0 (Q2, Q1, Q0, Cout, Reset, Enable, CLK);
    initial #300 $finish;
    initial begin
    CLK = 0;
    forever #5 CLK = ~CLK;
    end
    initial fork
    #20 Reset = 1;
```

```
            #35 Reset = 0;
            #50 Enable = 1;
            #90 Enable = 0;
            #120 Enable = 1;
            join
        endmodule
```



[10 Points]

(Q5) It is required to model a generic n-bit Ripple Carry Adder (RCA). A 4-bit RCA is shown below:



Model a behavioral n-bit RCA using repetitive constructs.

```
module RCA #(parameter n=4)
(output reg [n-1:0] Sum, output Cout, input [n-1:0] A, B, input Cin);

reg [n-1:0] Carry;
integer i;
assign Cout = Carry[n-1];
always @(A, B, Cin) begin
        Sum[0] = A[0] ^ B[0] ^ Cin;
        Carry[0] = A[0] & B[0] | Cin & (A[0] ^ B[0]);
        for (i=1; i < n ; i=i+1) begin
           Sum[i] = A[i] ^ B[i] ^ Carry[i-1];
           Carry[i] = A[i] & B[i] | Carry[i-1] & (A[i] ^ B[i]);
        end
end
endmodule
```