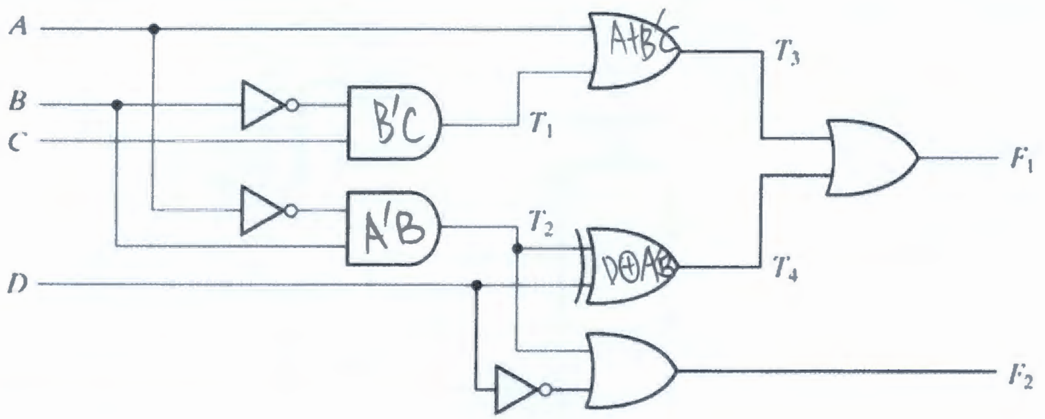
王廷基

102062129

許[illegible]韓

CS2102 Digital Logic Design
Exam 2 : 10:10 – 12:00, December 2, 2014

1. Consider the following circuit.



(a) Derive the Boolean expressions for T_1 through T_4. (4%)

(b) Write the outputs F_1 and F_2 as a function of the four inputs. (2%)

(c) List the truth table with 16 combinations of the four input variables. Then list the binary values for T_1 through T_4 and outputs F_1 and F_2 in the table. (6%)

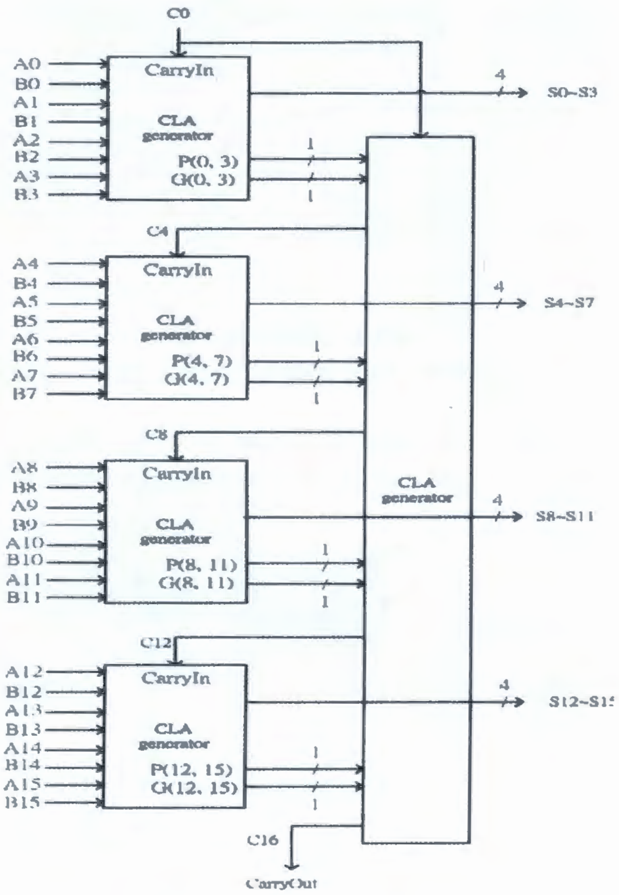
2. A majority circuit is a combinational circuit whose output is equal to 1 if the input variables have more 1's than 0's. The output is 0 otherwise. Design a 3-input majority circuit by finding the circuit's truth table and a logic diagram. Your logic diagram must be a simplified two-level AND-OR circuit. (8%)

3. Let F(A, B, C) = Σ(1, 2, 3, 6).

(a) Implement F using one 3-to-8 decoder and one OR gate. (4%)

(b) Implement F using one 4-to-1 multiplexer. (4%)

4. In class we have introduced a two-level carry lookahead (CLA) adder as shown below. The inputs of the adder include two 16-bit numbers A = $A_{15}A_{14}\ldots A_0$ and B = $B_{15}B_{14}\ldots B_0$, and a carry-in bit C_0 in the least significant position. In each significant position i ($0 \leq i \leq 15$), let S_i and C_{i+1} respectively denote the sum bit and carry-out bit of $A_i + B_i + C_i$. Explain how to get C_{12}, C_{15}, and S_{15} from this circuit. (12%)

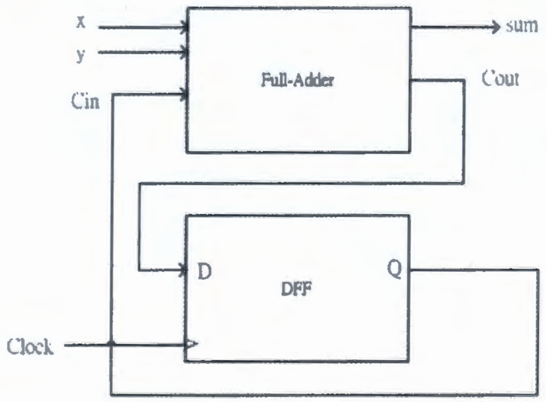


5. Consider the following sequential circuit that has two inputs x, y and one output sum. It consists of a full-adder and a D flip-flop, where C_{in} is the carry-in of the full-adder and C_{out} is the carry-out connected to the D flip-flop.

(a) Derive the input equations for the D flip-flop in terms of x, y, C_{in}. (2%)

(b) Derive the state equations for the D flip-flop in terms of x, y, C_{in}. (2%)

(c) Derive the state table of the sequential circuit. (6%)



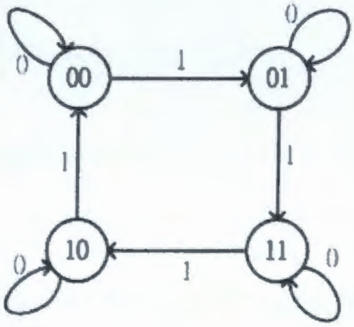
6. Consider the following state table

	Next State		Output	
Present State	X = 0	X = 1	X = 0	X = 1
a	f	b	0	0
b	d	~~c~~a	0	0
~~c~~	~~f~~	~~e~~~~b~~	~~0~~	~~0~~
d	g	a	1	0
~~e~~	~~d~~	~~e~~	~~0~~	~~0~~
~~f~~	f	b	1	1
g	g	~~h~~d	0	1
~~h~~	~~g~~	~~a~~	~~1~~	~~0~~

h = d
e = b
c = a

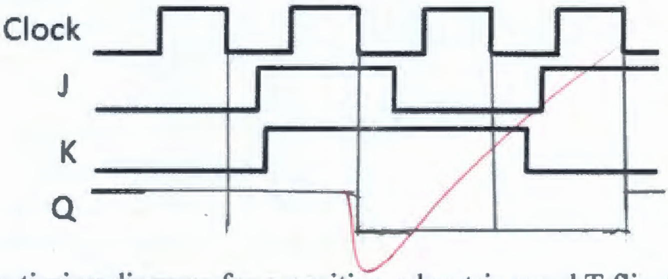
(a) Is it a Moore or Mealy circuit? Why? (2%)
(b) Perform state reduction to get the reduced state table. (4%)
(c) Draw the state diagram corresponding to the reduced state table. (4%)

7. Use D flip-flops and other necessary gates to design a sequential circuit whose state diagram is shown below. You need to simplify the input equation of each D flip-flop using the K-map method to build your circuit (10%)

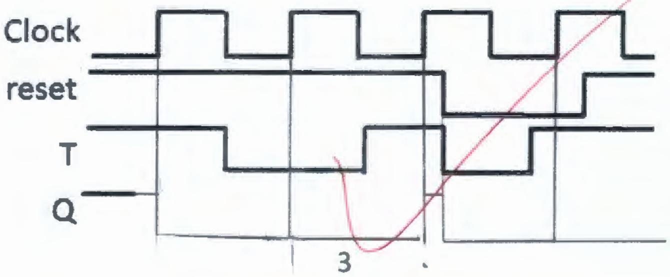


$JQ' + K'Q$

8. (a) Complete the following timing diagram for a negative-edge-triggered J-K flip-flop. Assume Q begins at 1, and ignore the propagation delay of the J-K flip-flop. (4%)



(b) Complete the following timing diagram for a positive-edge-triggered T flip-flop with an asynchronous active-low reset input. Assume Q begins at 1, and ignore the propagation delay of the T flip-flop. (6%)



9. Answer TRUE or FALSE. (6%)
   (a) A Verilog module can be defined within another Verilog module.
   (b) The instance name of a Verilog module is required.
   (c) The first entry in the port list of a Verilog predefined primitive is an input.

10. Use each of the following styles to complete a Verilog model that describes one-bit comparator whose output is 1 if in_1 is greater than or equal to in_2, where in_1, in_2 are the inputs, and comp_out is the output. Assume that each input has the value 0 or 1. (9%)

(a) Gate-level modeling

```
module test_a (comp_out, in_1, in_2);
        output comp_out;
        input in_1, in_2;
        /* complete your code here */
endmodule
```

(b) Dataflow modeling

```
module test_b (comp_out, in_1, in_2);
        output comp_out;
        input in_1, in_2;
        /* complete your code here */
endmodule
```

(c) Behavioral modeling

```
module test_c (comp_out, in_1, in_2);
        output comp_out;
        input in_1, in_2;
        /* complete your code here */
endmodule
```

11. Explain why the following Verilog module does not correctly model a 4-to-1 multiplexer. (5%)

```
module mux4 (a, b, c, d, select, y_out);
        input a, b, c, d;
        input [1:0] select;
        output y_out;

        reg y_out;

        always @ (select)
                if (select == 0) y_out = a;
                else if (select == 1) y_out = b;
                else if (select == 2) y_out = c;
                else y_out = d;
endmodule
```

0|

(b) $F_1 = T_3 + T_4 = A + B'C + A'BD' + AD + B'D$

$F_2 = T_2 + D' = A'B + D'$

(c)

A	B	C	D	T_1	T_2	T_3	T_4	F_1	F_2
0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1	0
0	0	1	0	1	0	1	0	1	1
0	0	1	1	1	0	1	1	1	0
0	1	0	0	0	1	0	1	1	1
0	1	0	1	0	1	0	0	0	1
0	1	1	0	0	1	0	1	1	1
0	1	1	1	0	1	0	0	0	1
1	0	0	0	0	0	1	0	1	1
1	0	0	1	0	0	1	1	1	0
1	0	1	0	1	0	1	0	1	1
1	0	1	1	1	0	1	1	1	0
1	1	0	0	0	0	1	0	1	1
1	1	0	1	0	0	1	1	1	0
1	1	1	0	0	0	1	0	1	1
1	1	1	1	0	0	1	1	1	0

2、

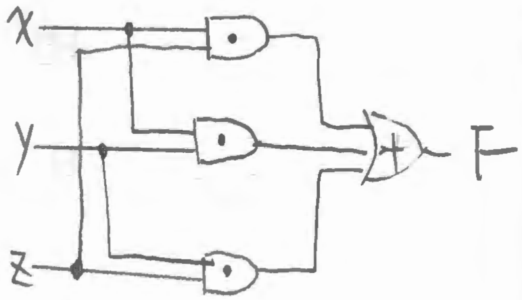
x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x \ yz	00	01	11	10
0	0	0	1	0
x 1	0	1	1	1

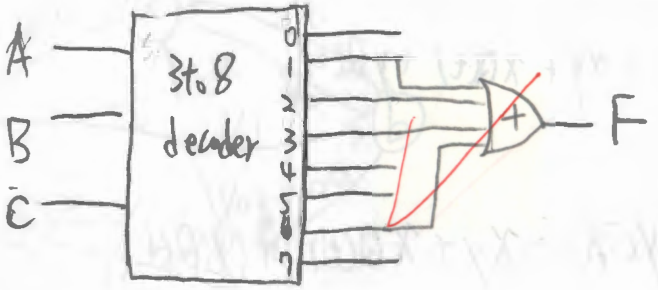
y

z

F = xz + yz + xy

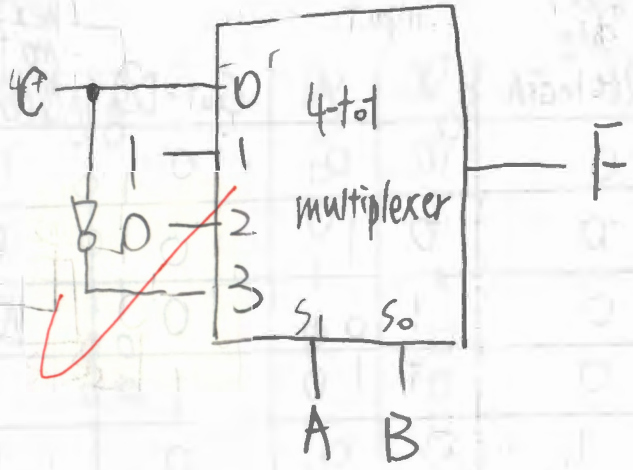


3, (a)



(b)

A	B	C	T	
0	0	0	0	C
0	0	1	1	
0	1	0	1	1
0	1	1	1	
1	0	0	0	0
1	0	1	0	
1	1	0	1	C'
1	1	1	0	



4,

Let $p_i = A_i \oplus B_i$

$g_i = A_i \cdot B_i$

$$C_{12} = P_{(8,11)} C_8 + G_{(8,11)} = P_{(8,11)}(P_{(4,7)} C_4 + G_{(4,7)}) + G_{(8,11)}$$

$$= P_{(8,11)} G_{(4,7)} + G_{(8,11)} + P_{(8,11)} P_{(4,7)}(P_{(0,3)} C_0 + G_{(0,3)})$$

$$= P_{(8,11)} P_{(4,7)} P_{(0,3)} C_0 + P_{(8,11)} P_{(4,7)} G_{(0,3)}$$

$$+ P_{(8,11)} G_{(4,7)} + G_{(8,11)}$$

$$C_{15} = p_{14} C_{14} + g_{14} = p_{14}(p_{13} C_{13} + g_{13}) + g_{15} = p_{14} p_{13}(p_{12} C_{12} + g_{12}) + p_{14} g_{13} + g_{14}$$

$$= p_{14} p_{13} p_{12} C_{12} + p_{14} p_{13} g_{12} + p_{14} g_{13} + g_{14}$$

$$S_{15} = A_{15} \oplus B_{15} \oplus C_{15}$$

5、(a)

$$D = xy + xcin + ycin = xy + xQ(t) + yQ(t)$$

(b)

$$Q(t+1) = D = xy + xcin + ycin = xy + xQ(t) + yQ(t)$$

(c)

Present state Q(t)=cin	input x	 y	 Cout=D	next state Q(t+1)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

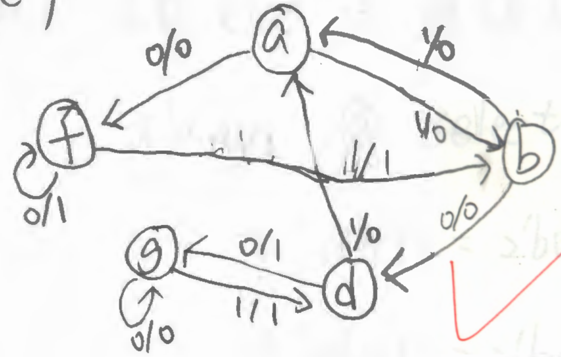
Sum - 2

6, (a) Mealy, 由 edge-triggered 發生前的情況判斷 output

(b)

Present state	Next state x=0	 x=1	Output x=0	 x=1
a	f	b	0	0
b	d	a	0	0
d	g	a	1	0
f	f	b	1	1
g	g	d	0	1

(c)



7.

Present state A B	Input x	next state A B	DA	DB
0 0	0	0 0	0	0
0 0	1	0 1	0	1
0 1	0	0 1	0	1
0 1	1	1 1	1	1
1 0	0	~~0~~1 0	~~0~~1	0
1 0	1	0 0	0	0
1 1	0	1 1	1	0
1 1	1	1 0	1	0

-4

DA

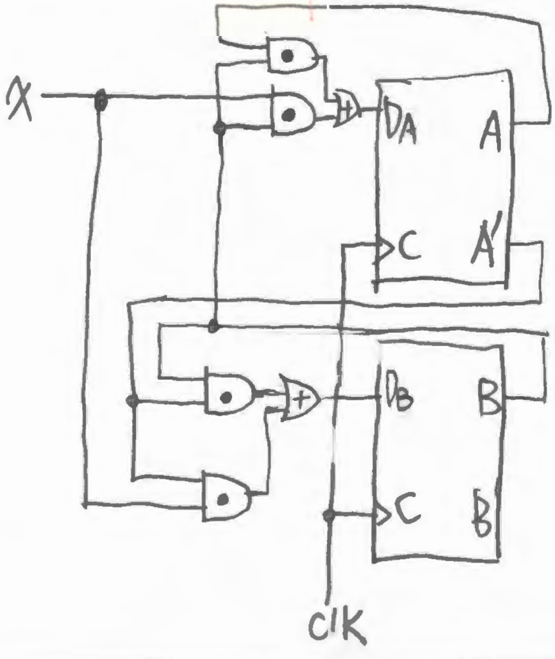
A \ Bx				
	0	0	1	0
	~~0~~1	0	1	1

DB

A \ Bx				
	0	1	1	1
	0	0	0	~~0~~1

$D_A = Bx + A\cancel{B}x'$

$D_B = \cancel{AB} + A'x + Bx'$



8. -0

9. (a) False

(b) True

(c) Faluse

✓

10.

(a)

```
wire  in_2_not, equal, greater;
not G1(in_2_not, in_2);
and G2(greater, in_1, in_2_not);
xor G3(equal, in_1, in_2);
or G4(comp_out, greater, equal);
```

-3

(b)

```
assign  comp_out = (in_1>in_2)||(in_1==in_2);
```

✓

(c)

```
always  @ (in_1, in_2) begin
 if (in_1 > in_2)
       comp_out=1'b1;
 if (in_1 == in_2)
       comp_out=1'b1;
 if (in_1 < in_2)
       comp_out=1'b0;
end
```

-1

11、没有 else if 應该成

-3

```
always @ (select)
        if (select == 2'b00) y-out=a;
        if (select == 2'b01) y-out=b;
        if (select == 2'b10) y-out=c;
        if (select == 2'b11) y-out=d;
```