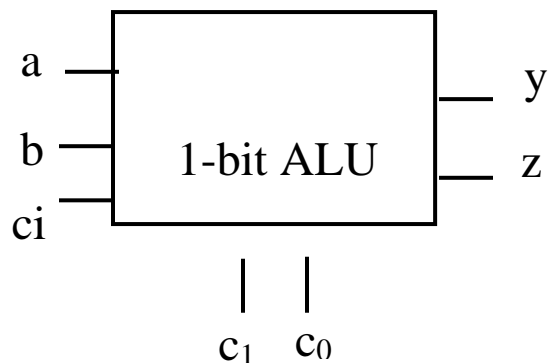


## Lab 10 (Team Project) เตรียมพร้อมการสอบ Verilog

### จงออกแบบวงจรและเขียน Verilog เพื่อตรวจสอบการทำงาน

ชื่อ-นามสกุล.....พงศกร รัตนพันธ์.....	รหัสนักศึกษา...630610749.....	ตอนที่.....001...
ชื่อ-นามสกุล .....ธนดล เดชประภากร.	รหัสนักศึกษา .. 630610734 ..	ตอนที่ 001
ชื่อ-นามสกุล .....ธนันท์ เชาวดี...	รหัสนักศึกษา.....630610735 .....	ตอนที่.... 001.....

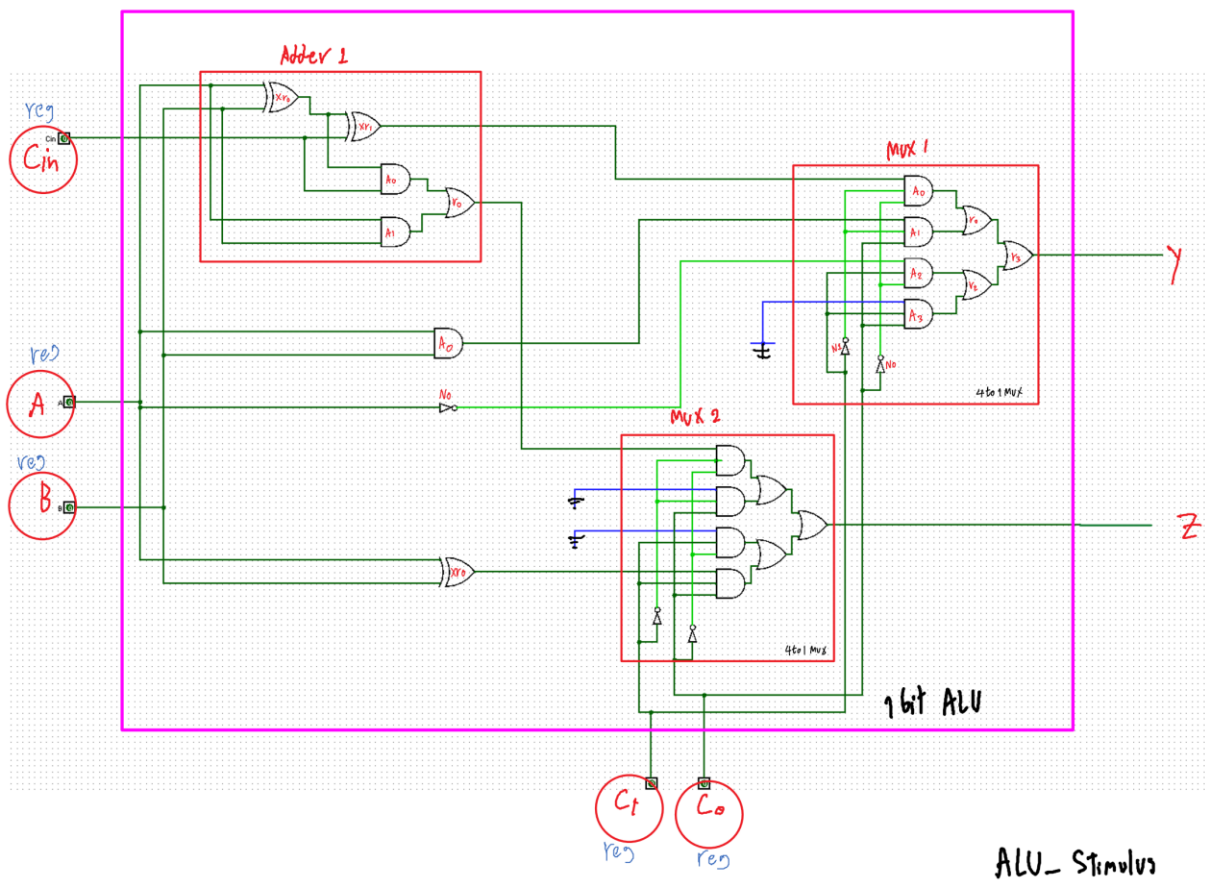
1) ออกแบบ 1-bit ALU ( Arithmetic Logic Unit) ที่รับอินพุต a และ b และเลือกปฏิบัติการตามสัญญาณควบคุม  $c_1$  และ  $c_0$



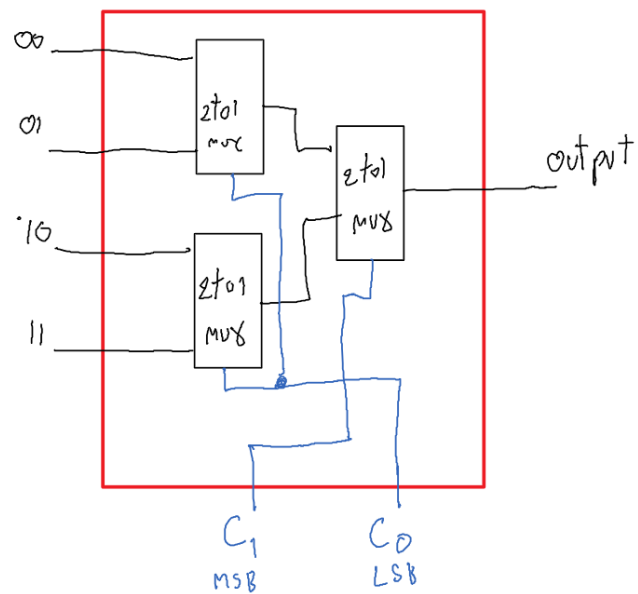
โดยมีตารางการปฏิบัติการดังนี้

$c_1$	$c_0$	y	z
0	0	sum of (a, b, ci)	Carry out
0	1	a AND b	0
1	0	NOT a	0
1	1	0	“0” if a=b “1” if a!=b

- 
- 
- ออกแบบ โครงสร้างของ Verilog



optional for 4to 1 mux



- เขียน Code ตัว

```

1 module adder (Sum, Cout, Cin, A, B);
2     input Cin, A, B;
3     output Sum, Cout;
4
5     wire w_xr0, w_A0, w_A1;
6
7     xor xr0(w_xr0, A, B);
8     xor xr1(Sum, Cin, w_xr0);
9
10    and A0(w_A0, w_xr0, Cin);
11    and A1(w_A1, A, B);
12
13    or r0(Cout, w_A0, w_A1);
14 endmodule

```

```

1 module mux4_1 (y, x0, x1, x2, x3, c1, c0);
2     input x0, x1, x2, x3, c1, c0;
3     output y;
4
5     wire w_A0, w_A1, w_A2, w_A3;
6     wire w_r0, w_r2;
7
8     wire w_N1, w_N0;
9
10    not N1(w_N1, c1);
11    not N0(w_N0, c0);
12
13    and A0(w_A0, x0, w_N1, w_N0);
14    and A1(w_A1, x1, w_N1, c0);
15    and A2(w_A2, x2, c1, w_N0);
16    and A3(w_A3, x3, c1, c0);
17
18    or r0(w_r0, w_A0, w_A1);
19    or r2(w_r2, w_A2, w_A3);
20    or r3(y, w_r0, w_r2);
21 endmodule

```

```

1 `include "adder.v"
2 `include "mux4_1.v"
3
4 module one_bit_ALU (y, z, Cin, A, B, c1, c0);
5     input Cin, A, B, c1, c0;
6     output y, z;
7
8     wire w_Sum, w_Cout;
9     wire w_A0, w_N0, w_xr0;
10
11    adder Adder1(w_Sum, w_Cout, Cin, A, B);
12    and A0(w_A0, A, B);
13    not N0(w_N0, A);
14    xor xr0(w_xr0, A, B);
15
16    mux4_1 mux1(y, w_Sum, w_A0, w_N0, 1'b0, c1, c0);
17    mux4_1 mux2(z, w_Cout, 1'b0, 1'b0, w_xr0, c1, c0);
18 endmodule

```

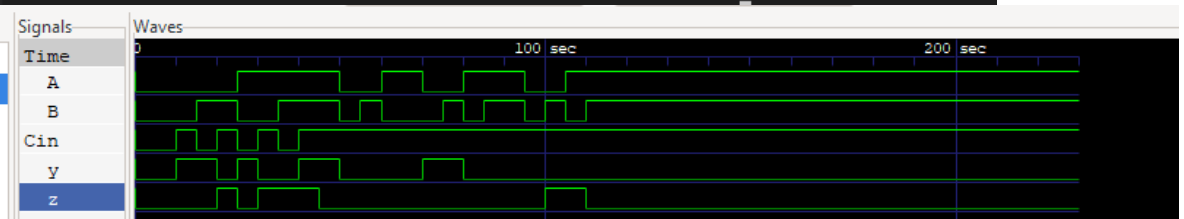
```

1  initial begin
2      $dumpfile("1bitALU_TimingDiagram.vcd");
3      $dumpvars(0, one_bit_ALU);
4      Cin = 1'b0;
5      A = 1'b0;
6      B = 1'b0;
7      c1 = 1'b0;
8      c0 = 1'b0;
9
10     // 0+0+0
11     #5 begin
12         A = 1'b0;
13         B = 1'b0;
14         Cin = 1'b0;
15     end
16     // 0+0+1
17     #5 begin
18         A = 1'b0;
19         B = 1'b0;
20         Cin = 1'b1;
21     end
22     // 0+1+0
23     #5 begin
24         A = 1'b0;
25         B = 1'b1;
26         Cin = 1'b0;
27     end
28     // 0+1+1
29     #5 begin
30         A = 1'b0;
31         B = 1'b1;
32         Cin = 1'b1;
33     end
34     // 1+0+0
35     #5 begin
36         A = 1'b1;
37         B = 1'b0;
38         Cin = 1'b0;
39     end
40     // 1+0+1
41     #5 begin
42         A = 1'b1;
43         B = 1'b0;
44         Cin = 1'b1;
45     end
46     // 1+1+0
47     #5 begin
48         A = 1'b1;
49         B = 1'b1;
50         Cin = 1'b0;
51     end
52     // 1+1+1
53     #5 begin
54         A = 1'b1;
55         B = 1'b1;
56         Cin = 1'b1;
57     end
58
59     #5 c0 = 1'b1;
60
61     #5 begin
62         A = 1'b0;
63         B = 1'b0;
64     end
65
66     #5 begin
67         A = 1'b0;
68         B = 1'b1;
69     end
70
71     #5 begin
72         A = 1'b1;
73         B = 1'b0;
74     end
75

```

- แสดงผลการทำงานที่ได้ และ

```
PS C:\Users\intel\github\Logic\work\Logic_Lab10\combination> vvp .\1bit_ALU_Test
VCD info: dumpfile 1bitALU_TimingDiagram.vcd opened for output.
  0 Output Cin=0 A=0 B=0 c1=0 c0=0 y=0 z=0
 10 Output Cin=1 A=0 B=0 c1=0 c0=0 y=1 z=0
 15 Output Cin=0 A=0 B=1 c1=0 c0=0 y=1 z=0
 20 Output Cin=1 A=0 B=1 c1=0 c0=0 y=0 z=1
 25 Output Cin=0 A=1 B=0 c1=0 c0=0 y=1 z=0
 30 Output Cin=1 A=1 B=0 c1=0 c0=0 y=0 z=1
 35 Output Cin=0 A=1 B=1 c1=0 c0=0 y=0 z=1
 40 Output Cin=1 A=1 B=1 c1=0 c0=0 y=1 z=1
 45 Output Cin=1 A=1 B=1 c1=0 c0=1 y=1 z=0
 50 Output Cin=1 A=0 B=0 c1=0 c0=1 y=0 z=0
 55 Output Cin=1 A=0 B=1 c1=0 c0=1 y=0 z=0
 60 Output Cin=1 A=1 B=0 c1=0 c0=1 y=0 z=0
 65 Output Cin=1 A=1 B=0 c1=1 c0=0 y=0 z=0
 70 Output Cin=1 A=0 B=0 c1=1 c0=0 y=1 z=0
 75 Output Cin=1 A=0 B=1 c1=1 c0=0 y=1 z=0
 80 Output Cin=1 A=1 B=0 c1=1 c0=0 y=0 z=0
 85 Output Cin=1 A=1 B=1 c1=1 c0=0 y=0 z=0
 90 Output Cin=1 A=1 B=1 c1=1 c0=1 y=0 z=0
 95 Output Cin=1 A=0 B=0 c1=1 c0=1 y=0 z=0
100 Output Cin=1 A=0 B=1 c1=1 c0=1 y=0 z=1
105 Output Cin=1 A=1 B=0 c1=1 c0=1 y=0 z=1
110 Output Cin=1 A=1 B=1 c1=1 c0=1 y=0 z=0
stimulus_1bit_ALU.v:128: $finish called at 230 (1s)
```



- อธิบายว่า 1-bit ALU ที่ได้ทำงานถูกต้องอย่างไร  
: จากผลลัพธ์ที่ได้ วงจรที่ออกแบบทำงานได้ถูกต้อง โดย เมื่อให้

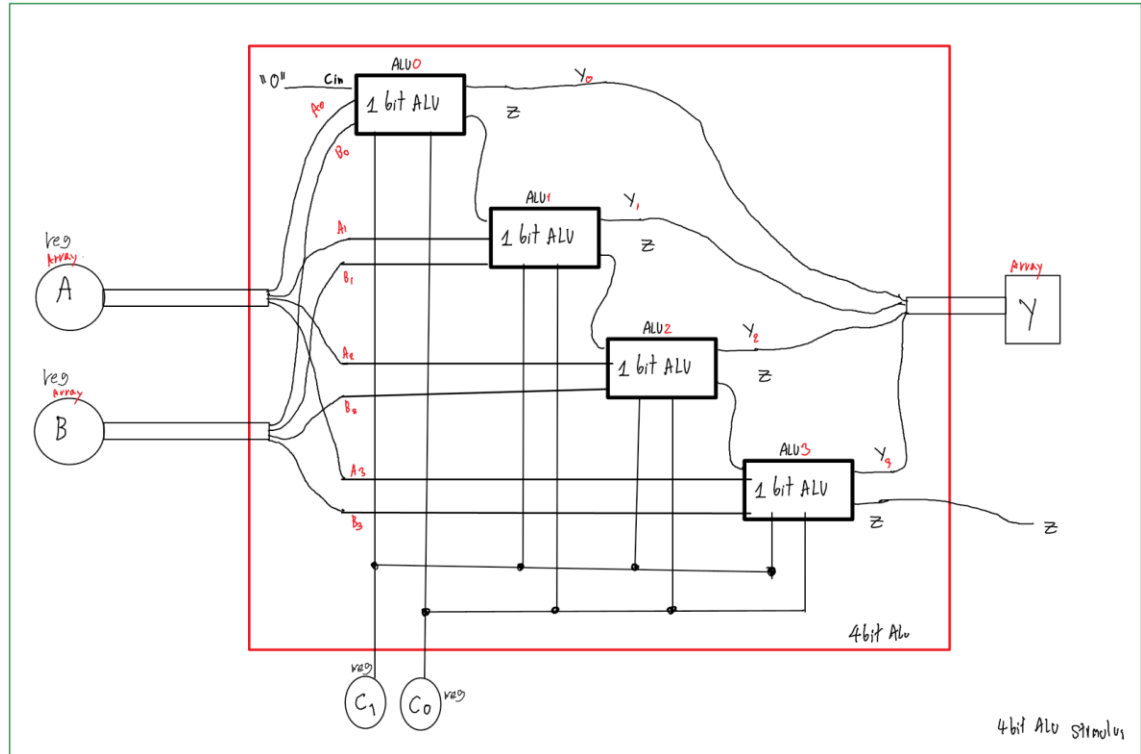
- **C1 = 0 และ C0 = 0** จะได้  $y = \text{sum}(s)$  และ  $z = \text{carry out}$  ของ full adder ที่มี inputs  $a$   $b$  และ  $c_{in}$  จะได้ output  $Y$ ,  $Z$  ตามตาราง

Inputs			Outputs	
A	B	C – IN	Sum	C – Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

- 
- **C1 = 0 และ C0 = 1** จะได้  $y = a \text{ and } b$  และ  $z = 0$  ถูกต้องตามต้องการ ดังนี้
  - $a = 0 \ b = 0 \ y = 0 \ z = 0$
  - $a = 0 \ b = 1 \ y = 0 \ z = 0$
  - $a = 1 \ b = 0 \ y = 0 \ z = 0$
  - $a = 1 \ b = 1 \ y = 1 \ z = 0$
- **C1 = 1 และ C0 = 0** จะได้  $y = \text{not } a$  และ  $z = 0$  ถูกต้องตามต้องการ ดังนี้
  - $a = 0 \ b = 0 \ y = 1 \ z = 0$
  - $a = 0 \ b = 1 \ y = 1 \ z = 0$
  - $a = 1 \ b = 0 \ y = 0 \ z = 0$
  - $a = 1 \ b = 1 \ y = 0 \ z = 0$
- **C1 = 1 และ C0 = 1** จะได้  $y = 0$  และ  $z = a \text{ xor } b$  ถูกต้องตามต้องการ ดังนี้
  - $a = 0 \ b = 0 \ y = 0 \ z = 0$
  - $a = 0 \ b = 1 \ y = 0 \ z = 1$
  - $a = 1 \ b = 0 \ y = 0 \ z = 1$
  - $a = 1 \ b = 1 \ y = 0 \ z = 0$

2) จากโมดูล 1-bit ALU (Arithmetic Logic Unit) ที่ได้ให้นำมาพัฒนาต่อเป็น 4-bit ALU โดยรับอินพุต a0-a3 และ b0-b3 โดยผลลัพธ์จะมี y0-y3 และ z ซึ่งเป็นผลการปฏิบัติการของอินพุต ซึ่งถูกเลือกปฏิบัติการตามสัญญาณควบคุม  $c_1$  และ  $c_0$

- ออกแบบ โครงสร้างของ Verilog



- เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus

```

1  `include "1bit_ALU.v"
2
3  module four_bit_ALU (
4      y, z,
5      a, b,
6      c1, c0
7  );
8
9      input [3:0] a;
10     input [3:0] b;
11
12     output [3:0] y;
13     output z;
14
15     input c1, c0;
16
17     wire w_z0, w_z1, w_z2;
18
19     one_bit_ALU ALU0(y[0], w_z0, 1'b0, a[0], b[0], c1, c0);
20     one_bit_ALU ALU1(y[1], w_z1, w_z0, a[1], b[1], c1, c0);
21     one_bit_ALU ALU2(y[2], w_z2, w_z1, a[2], b[2], c1, c0);
22     one_bit_ALU ALU3(y[3], z, w_z2, a[3], b[3], c1, c0);
23
24 endmodule

```

```

1  `include "4bitALU.v"
2
3  module stimulus_4bit_ALU;
4      wire [3:0] y;
5      wire z;
6
7      reg [3:0] a;
8      reg [3:0] b;
9
10     reg c1, c0;
11
12     four_bit_ALU four_bit_ALU(y, z, a, b, c1, c0);
13
14     initial begin
15         $dumpfile("4bitALU_TimingDiagram.vcd");
16         $dumpvars(0, four_bit_ALU);
17         a = 1'b0;
18         b = 1'b0;
19         c1 = 1'b0;
20         c0 = 1'b0;
21     end
22
23     initial begin
24         repeat(4) begin
25             b = 1'b0;
26             repeat(4) begin
27                 #5 b = b + 1'b1;
28             end
29             #5 a = a + 1'b1;
30         end
31         a = 1'b0;
32
33         c0 = 1'b1;
34         repeat(4) begin
35             b = 1'b0;
36             repeat(4) begin
37                 #5 b = b + 1'b1;
38             end
39             #5 a = a + 1'b1;
40         end
41         a = 1'b0;
42
43         c1 = 1'b1;
44         c0 = 1'b0;
45         repeat(4) begin
46             b = 1'b0;
47             repeat(4) begin
48                 #5 b = b + 1'b1;
49             end
50             #5 a = a + 1'b1;
51         end
52         a = 1'b0;
53
54         c0 = 1'b1;
55         repeat(4) begin
56             b = 1'b0;
57             repeat(4) begin
58                 #5 b = b + 1'b1;
59             end
60             #5 a = a + 1'b1;
61         end
62
63         #400 $finish;
64     end
65
66     initial
67         $monitor($time, " Output c1=%d, c0=%d, a=%d, b=%d, y=%b=%d, z=%d", c1, c0, a, b, y, y, z);
68 endmodule
69

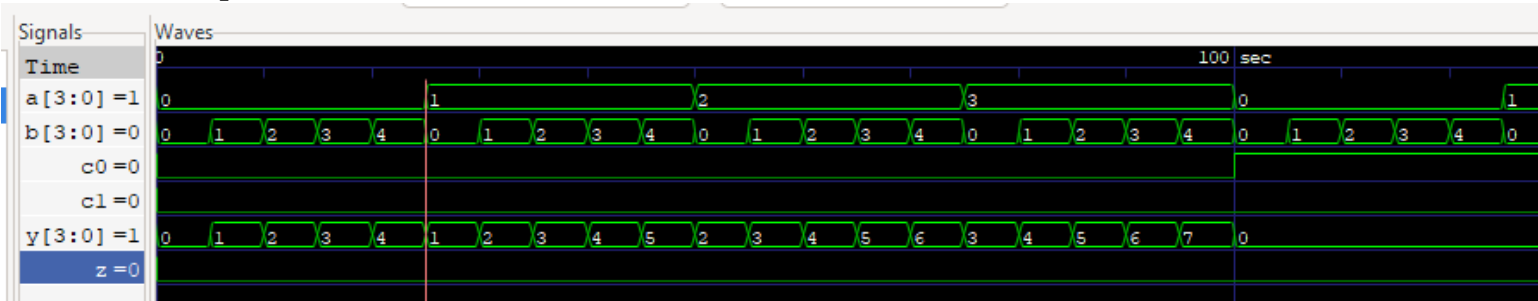
```



- แสดงผลการทำงานที่ได้

```
PS C:\Users\intel\github\Logic\work\Logic_Lab10\combination> vvp .\4bit_ALU_Test
VCD info: dumpfile 4bitALU_TimingDiagram.vcd opened for output.
```

0	Output	c1=0,	c0=0,	a= 0,	b= 0,	y=0000= 0,	z=0
5	Output	c1=0,	c0=0,	a= 0,	b= 1,	y=0001= 1,	z=0
10	Output	c1=0,	c0=0,	a= 0,	b= 2,	y=0010= 2,	z=0
15	Output	c1=0,	c0=0,	a= 0,	b= 3,	y=0011= 3,	z=0
20	Output	c1=0,	c0=0,	a= 0,	b= 4,	y=0100= 4,	z=0
25	Output	c1=0,	c0=0,	a= 1,	b= 0,	y=0001= 1,	z=0
30	Output	c1=0,	c0=0,	a= 1,	b= 1,	y=0010= 2,	z=0
35	Output	c1=0,	c0=0,	a= 1,	b= 2,	y=0011= 3,	z=0
40	Output	c1=0,	c0=0,	a= 1,	b= 3,	y=0100= 4,	z=0
45	Output	c1=0,	c0=0,	a= 1,	b= 4,	y=0101= 5,	z=0
50	Output	c1=0,	c0=0,	a= 2,	b= 0,	y=0010= 2,	z=0
55	Output	c1=0,	c0=0,	a= 2,	b= 1,	y=0011= 3,	z=0
60	Output	c1=0,	c0=0,	a= 2,	b= 2,	y=0100= 4,	z=0
65	Output	c1=0,	c0=0,	a= 2,	b= 3,	y=0101= 5,	z=0
70	Output	c1=0,	c0=0,	a= 2,	b= 4,	y=0110= 6,	z=0
75	Output	c1=0,	c0=0,	a= 3,	b= 0,	y=0011= 3,	z=0
80	Output	c1=0,	c0=0,	a= 3,	b= 1,	y=0100= 4,	z=0
85	Output	c1=0,	c0=0,	a= 3,	b= 2,	y=0101= 5,	z=0
90	Output	c1=0,	c0=0,	a= 3,	b= 3,	y=0110= 6,	z=0
95	Output	c1=0,	c0=0,	a= 3,	b= 4,	y=0111= 7,	z=0
100	Output	c1=0,	c0=1,	a= 0,	b= 0,	y=0000= 0,	z=0
105	Output	c1=0,	c0=1,	a= 0,	b= 1,	y=0000= 0,	z=0
110	Output	c1=0,	c0=1,	a= 0,	b= 2,	y=0000= 0,	z=0
115	Output	c1=0,	c0=1,	a= 0,	b= 3,	y=0000= 0,	z=0
120	Output	c1=0,	c0=1,	a= 0,	b= 4,	y=0000= 0,	z=0
125	Output	c1=0,	c0=1,	a= 1,	b= 0,	y=0000= 0,	z=0
130	Output	c1=0,	c0=1,	a= 1,	b= 1,	y=0001= 1,	z=0
135	Output	c1=0,	c0=1,	a= 1,	b= 2,	y=0000= 0,	z=0
140	Output	c1=0,	c0=1,	a= 1,	b= 3,	y=0001= 1,	z=0
145	Output	c1=0,	c0=1,	a= 1,	b= 4,	y=0000= 0,	z=0
150	Output	c1=0,	c0=1,	a= 2,	b= 0,	y=0000= 0,	z=0
155	Output	c1=0,	c0=1,	a= 2,	b= 1,	y=0000= 0,	z=0
160	Output	c1=0,	c0=1,	a= 2,	b= 2,	y=0010= 2,	z=0
165	Output	c1=0,	c0=1,	a= 2,	b= 3,	y=0010= 2,	z=0
170	Output	c1=0,	c0=1,	a= 2,	b= 4,	y=0000= 0,	z=0
175	Output	c1=0,	c0=1,	a= 3,	b= 0,	y=0000= 0,	z=0
180	Output	c1=0,	c0=1,	a= 3,	b= 1,	y=0001= 1,	z=0
185	Output	c1=0,	c0=1,	a= 3,	b= 2,	y=0010= 2,	z=0
190	Output	c1=0,	c0=1,	a= 3,	b= 3,	y=0011= 3,	z=0
195	Output	c1=0,	c0=1,	a= 3,	b= 4,	y=0000= 0,	z=0
200	Output	c1=1,	c0=0,	a= 0,	b= 0,	y=1111=15,	z=0
205	Output	c1=1,	c0=0,	a= 0,	b= 1,	y=1111=15,	z=0
210	Output	c1=1,	c0=0,	a= 0,	b= 2,	y=1111=15,	z=0
290	Output	c1=1,	c0=0,	a= 3,	b= 3,	y=1100=12,	z=0
295	Output	c1=1,	c0=0,	a= 3,	b= 4,	y=1100=12,	z=0
300	Output	c1=1,	c0=1,	a= 0,	b= 0,	y=0000= 0,	z=0
305	Output	c1=1,	c0=1,	a= 0,	b= 1,	y=0000= 0,	z=0
310	Output	c1=1,	c0=1,	a= 0,	b= 2,	y=0000= 0,	z=0
315	Output	c1=1,	c0=1,	a= 0,	b= 3,	y=0000= 0,	z=0
320	Output	c1=1,	c0=1,	a= 0,	b= 4,	y=0000= 0,	z=0
325	Output	c1=1,	c0=1,	a= 1,	b= 0,	y=0000= 0,	z=0



อธิบายว่า 4-bit ALU ที่ได้ทำงานถูกต้องอย่างไร จากผลลัพธ์ที่ได้ วงจรที่ออกแบบทำงานได้ถูกต้อง โดย เมื่อให้

- **C1 = 0 และ C0 = 0** จะได้ output คือ  $y_0-y_3 = \text{sum}(s)$  ซึ่งเป็นผลรวมของ input a , b ในแต่ละหลัก เมื่อบวกเกินจะมีการพิเศษไปหลักถัดไป ซึ่งมี 4 bit โดย

- $y_3$  เป็น MSB และ  $y_0$  เป็น LSB

- **C1 = 0 และ C0 = 1** จะได้  $y = a$  and  $b$  และ  $z = 0$  ถูกต้องตามต้องการ ดังนี้

$$a_0-a_3 = 0 \ b_0-b_3 = 0 \ y_0-y_3 = 0 \ z = 0$$

$$a_0-a_3 = 0 \ b_0-b_3 = 1 \ y_0-y_3 = 0 \ z = 0$$

$$a_0-a_3 = 1 \ b_0-b_3 = 0 \ y_0-y_3 = 0 \ z = 0$$

$$a_0-a_3 = 1 \ b_0-b_3 = 1 \ y_0-y_3 = 1 \ z = 0$$

- **C1 = 1 และ C0 = 0** จะได้  $y = \text{not } a$  และ  $z = 0$  ถูกต้องตามต้องการ ดังนี้

$$a_0-a_3 = 0 \ b_0-b_3 = 0 \ y_0-y_3 = 1 \ z = 0$$

$$a_0-a_3 = 0 \ b_0-b_3 = 1 \ y_0-y_3 = 1 \ z = 0$$

$$a_0-a_3 = 1 \ b_0-b_3 = 0 \ y_0-y_3 = 0 \ z = 0$$

$$a_0-a_3 = 1 \ b_0-b_3 = 1 \ y_0-y_3 = 0 \ z = 0$$

- **C1 = 1 และ C0 = 1** จะได้  $y = 0$  และ  $z = a \text{ xor } b$  ถูกต้องตามต้องการ ดังนี้

$$a_0-a_3 = 0 \ b_0-b_3 = 0 \ y_0-y_3 = 0 \ z = 0$$

$$a_0-a_3 = 0 \ b_0-b_3 = 1 \ y_0-y_3 = 0 \ z = 1$$

$$a_0-a_3 = 1 \ b_0-b_3 = 0 \ y_0-y_3 = 0 \ z = 1$$

$$a_0-a_3 = 1 \ b_0-b_3 = 1 \ y_0-y_3 = 0 \ z = 0$$

-

-

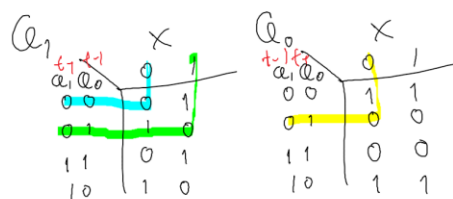
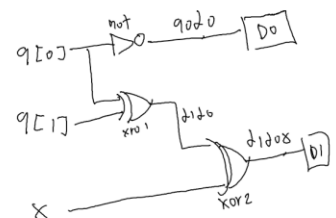
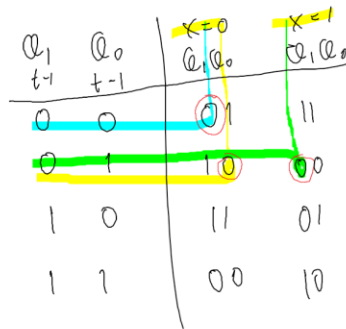
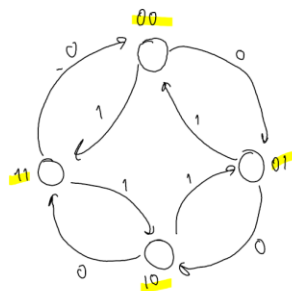
3) พัฒนาวงจรนับขึ้น,นับลง 2 บิต (00,01,10,11) โดยให้มีสัญญาณอินพุต (X) เป็นตัวควบคุมการนับขึ้นหรือนับลง โดยสัญญาณจะนับขึ้นก็ต่อเมื่อ X=0 และจะนับลงเมื่อ X=1 ออกแบบโดยใช้ Module D Flipflop

```

module D_FF(q,d,clk,reset);
    output q;
    input d,clk,reset;
    reg q;
    always @ (posedge reset or negedge clk)
        if(reset)
            q <= 1'b0;
        else
            q <= d;
endmodule

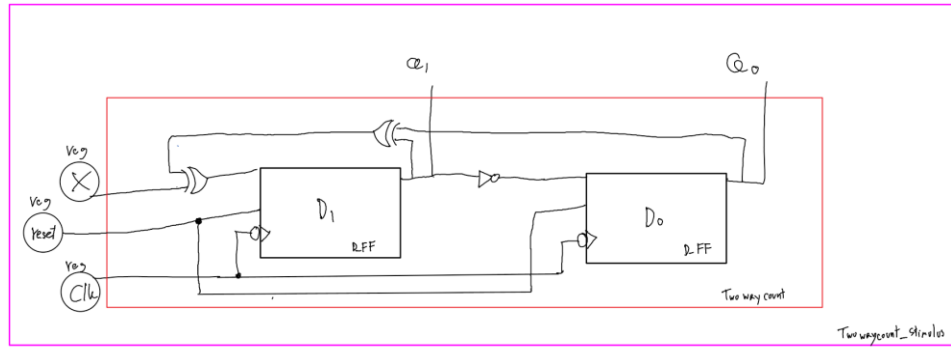
```

- ออกแบบ โครงสร้างของ Verilog



$$D_1 = \bar{x}\bar{a}_1\bar{a}_0 + \bar{x}a_1\bar{a}_0 + x\bar{a}_1\bar{a}_0 + xa_1\bar{a}_0 = x \oplus a_1 \oplus a_0$$

$$D_0 = \bar{a}_1\bar{a}_0 + a_1\bar{a}_0 = \bar{a}_0$$



เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus

```

stimulus_two_way_count.v
1  `include "two_way_count.v"
2
3  module stimulus_two_way_count;
4      reg x;
5      reg clk, reset;
6      wire [1:0]q;
7
8      two_way_count c0(q, x, clk, reset);
9
10     initial begin
11         $dumpfile("two_way_count.vcd");
12         $dumpvars(0, c0);
13         clk = 1'b0;
14         x = 1'b0;
15         reset = 1'b0;
16     end
17     always
18         #5 clk = ~clk;
19     // TODO: Test two count
20     initial begin
21         // x = 1'b0;
22         reset = 1'b1;
23         x = 1'b0;
24         #50 reset = 1'b0;
25         #50 x = 1'b1;
26         #0 reset = 1'b1;
27         #25 reset = 1'b0;
28         #50 reset = 1'b0;
29         #50 $finish;
30     end
31
32     initial
33         $monitor($time, " input: x=%d reset=%d output: q=%d", x, reset, q);
34 endmodule

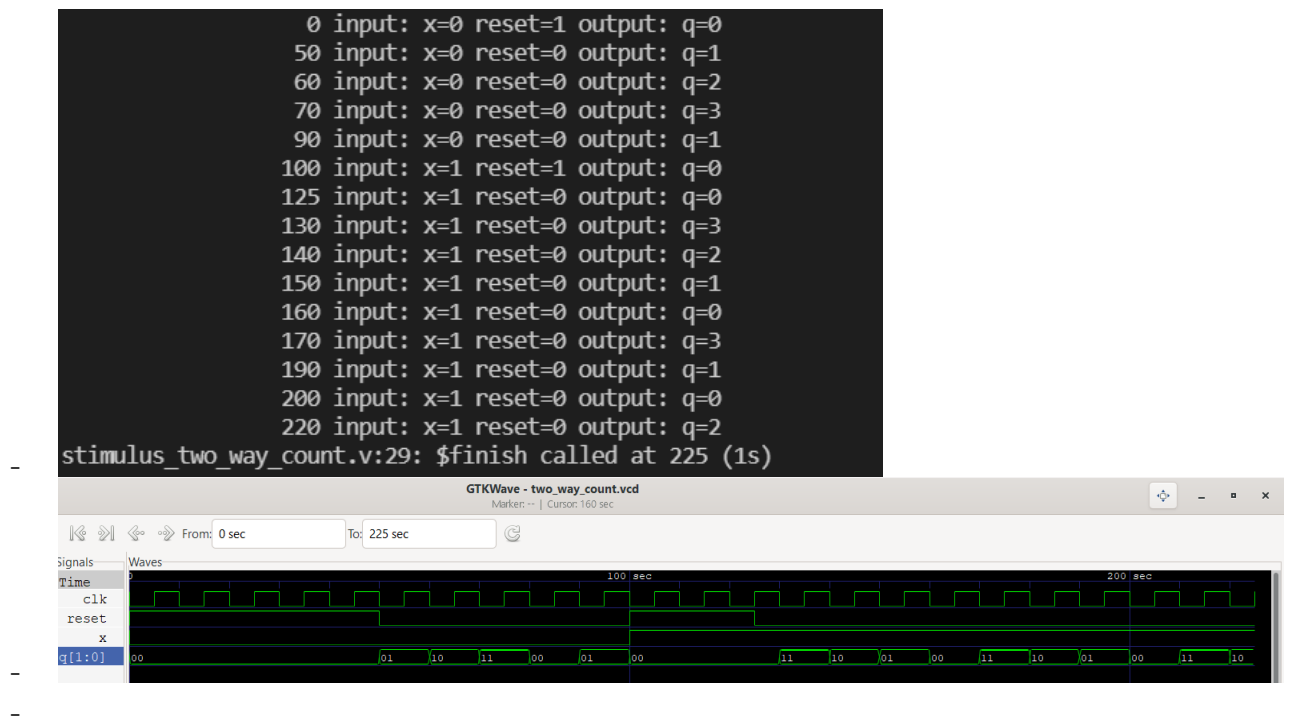
```

```

two_way_count.v
1  `include "D_FF.v"
2
3  module two_way_count(q, x, clk, reset);
4      input x, clk, reset;
5      output [1:0] q;
6
7      wire w_xor0, w_xor1, w_not0;
8
9      xor xor1(w_xor1, q[0], q[1]);
10     xor xor0(w_xor0, x, w_xor1);
11     D_FF D0(q[0], w_not0, clk, reset);
12
13     not not0(w_not0, q[0]);
14     D_FF D1(q[1], w_xor0, clk, reset);
15 endmodule

```

แสดงผลการทำงานที่ได้

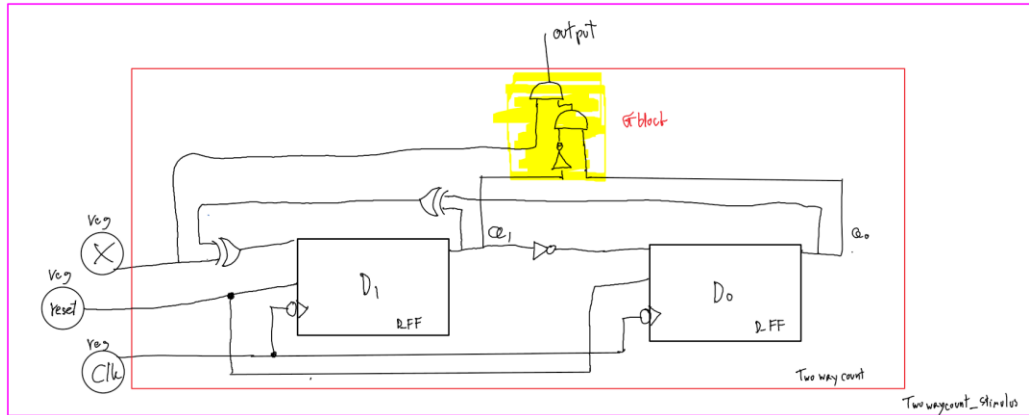


อธิบายว่าวงจรนับที่ได้ทำงานถูกต้องอย่างไร

- เมื่อ D\_FF เจอ **negedge clk** จะทำการเปลี่ยน **state** โดยมี D\_FF เป็นตัวจำว่าตอนนี้อยู่ **State** ไหน 2 ตัว
- เมื่อ x เป็น 0 วงจรจะนับขึ้นจาก 00 -> 01 -> 10 -> 11 -> 00 ไปเรื่อยๆ
- เมื่อ x เป็น 1 วงจรจะลงจาก 00 -> 11 -> 10 -> 01 -> 00 ไปเรื่อยๆ
- ตามที่เราต้องการเนื่องจากเรามี F Box ที่เป็น control box ที่คอยควบคุม logic ที่จะไปทำให้ D FF จำค่าต่างๆ
- โดย D\_FF Q1 MSB จะถูกควบคุมด้วย  $\text{not } Q0_{t-1}$
- และ D\_FF Q0 LSB จะถูกควบคุมด้วย  $[Q1_{t-1} \text{ xor } Q0_{t-1}] \text{ xor } x$
- เป็นไปตามที่ต้องการ

4) จากวงจรนับขึ้น, นับลง 2 บิต ที่ได้ในข้อ 3 ให้เพิ่มสัญญาณเอาต์พุตโดยให้สัญญาณเอาต์พุตเป็น 1 ก็ต่อเมื่ออยู่ที่สถานะ 01 และค่าอินพุต(x) เป็น 1 เท่านั้นในกรณีอื่นสัญญาณเอาต์พุตนี้จะเป็น 0

- ออกแบบ โครงสร้างของ Verilog



- เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus

```
sequential > 4 > two_way_count.gv
1  `include "D_FF.v"
2
3  module two_way_count_g1(x, clk, reset);
4      input x, clk, reset;
5      wire [1:0] q;
6
7      wire w_xor0, w_xor1, w_not0;
8
9      xor xor1(w_xor1, q[0], q[1]);
10     xor xor0(w_xor0, x, w_xor1);
11     D_FF D0(q[0], w_not0, clk, reset);
12
13     not not0(w_not0, q[0]);
14     D_FF D1(q[1], w_xor0, clk, reset);
15
16     output l;
17     wire w_not1, w_and0;
18
19     not not1(w_not1, q[1]);
20     and and1(w_and0, w_not1, q[0]);
21     and and2(l, x, w_and0);
22
23 endmodule
```

```

sequential > 4 > stimulus_two_way_count_g.v
1  `include "two_way_count_g.v"
2
3  module stimulus_two_way_count_g;
4      reg x;
5      reg clk, reset;
6
7
8      wire l;
9      assign L = l;
10
11     two_way_count_g c0( l, x, clk, reset);
12
13     initial begin
14         $dumpfile("two_way_count_g.vcd");
15         $dumpvars(0, c0);
16         clk = 1'b0;
17         x = 1'b0;
18         reset = 1'b0;
19     end
20     always
21     #5 clk = ~clk;
22     // TODO: Test two count
23     initial begin
24         // x = 1'b0;
25         reset = 1'b1;
26         x = 1'b0;
27         #50 reset = 1'b0;
28         #50 x = 1'b1;
29         #0 reset = 1'b1;
30         #25 reset = 1'b0;
31         #50 reset = 1'b0;
32         #50 $finish;
33     end
34
35     initial
36     $monitor($time, " output L = %d reset = %d x = %d", L, reset, x);
37 endmodule

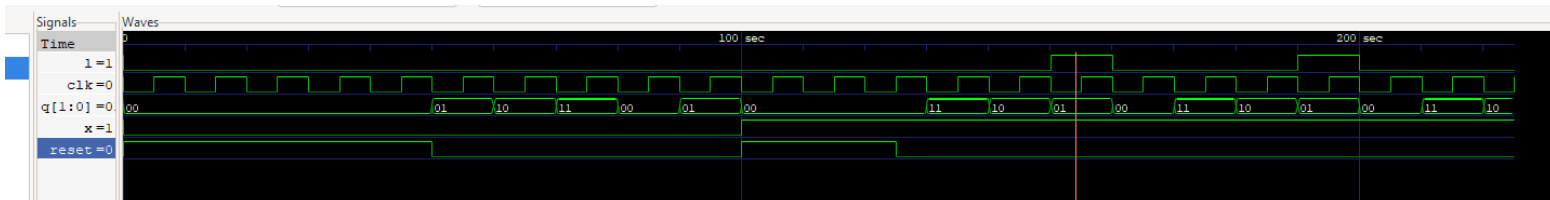
```

- แสดงผลการทำงานที่ได้

```

PS C:\Users\intel\github\Logic\work\Logic_Lab10\sequential\4> vvp .\two_way_count_test_g
VCD info: dumpfile two_way_count_g.vcd opened for output.
      0 output L = 0   reset = 1 x = 0
      50 output L = 0   reset = 0 x = 0
     100 output L = 0   reset = 1 x = 1
     125 output L = 0   reset = 0 x = 1
     150 output L = 1   reset = 0 x = 1
     160 output L = 0   reset = 0 x = 1
     190 output L = 1   reset = 0 x = 1
     200 output L = 0   reset = 0 x = 1
stimulus_two_way_count_g.v:32: $finish called at 225 (1s)

```



- อธิบายว่าวงจรนับที่ได้ทำงานถูกต้องอย่างไร

จาก output ในแต่ละ state ของ วงจร จะมี 00 01 10 11

ซึ่งเมื่อนำ state  $\text{not } Q1$  AND  $Q2$  AND  $X$  output จะเป็น 1 เมื่อ

$Q1 = 0$   $Q0 = 1$   $X = 1$  กรณีเดียวที่ได้ “1” และกรณีอื่น จะได้ผลลัพธ์เป็น “0”