Lab 10 (Team Project) เตรียมพร้อมการสอบ Verilog

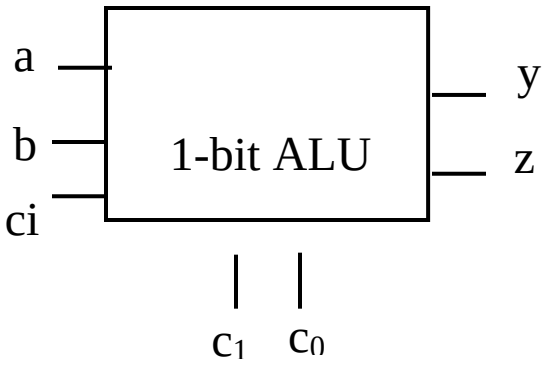
จงออกแบบวงจรและเขียน Verilog เพื่อตรวจสอบการทำงาน

ชื่อ-นามสกุล พงศกร รัตนพันธ์ รหัสนักศึกษา 630610749 ตอนที่ 001

ชื่อ-นามสกุล ธนดล เดชประภากร รหัสนักศึกษา 630610734 ตอนที่ 001

ชื่อ-นามสกุล ธนนันท์ เชาว์ดี รหัสนักศึกษา 630610735 ตอนที่ 001

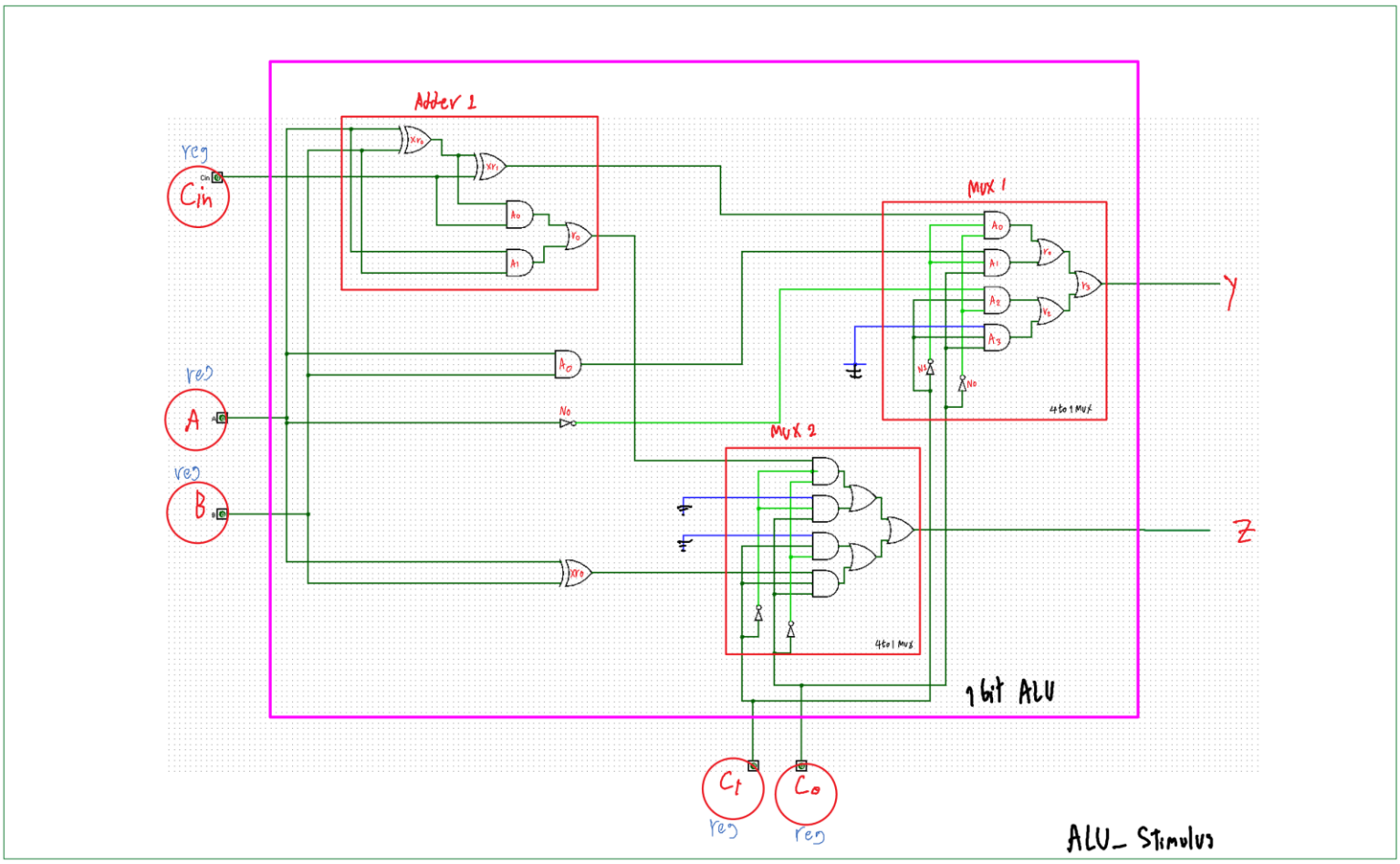
1) ออกแบบ 1-bit ALU (Arithmetic Logic Unit) ที่รับอินพุต a และ b และเลือกปฏิบัติการตามสัญญาณควบคุม c_1 และ c_0



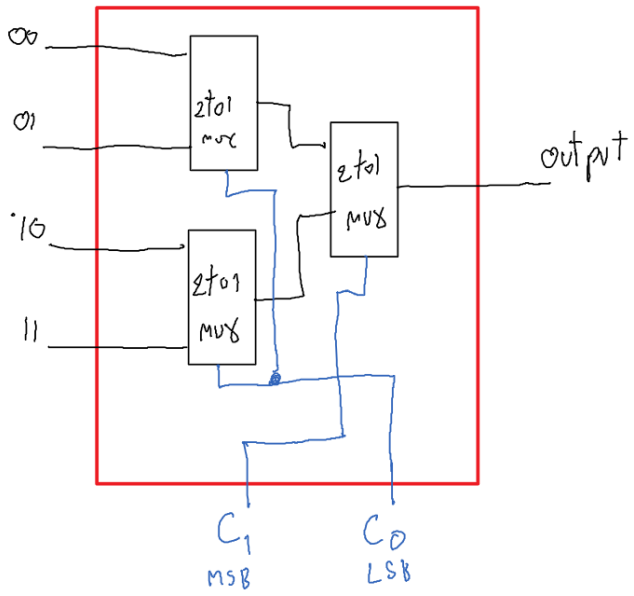
โดยมีตารางการปฏิบัติการดังนี้

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 4to1 mux



- เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus

```verilog
module adder (Sum, Cout, Cin, A, B);
    input Cin, A, B;
    output Sum, Cout;

    wire w_xr0, w_A0, w_A1;

    xor xr0(w_xr0, A, B);
    xor xr1 (Sum, Cin, w_xr0);

    and A0(w_A0, w_xr0, Cin);
    and A1(w_A1, A, B);

    or r0(Cout, w_A0, w_A1);
endmodule
```

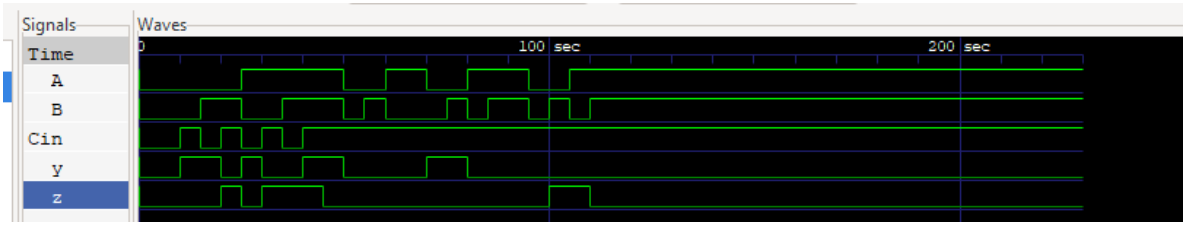
```verilog
module mux4_1 (y, x0, x1, x2, x3, c1, c0);
    input x0, x1, x2, x3, c1, c0;
    output y;

    wire w_A0, w_A1, w_A2, w_A3;
    wire w_r0, w_r2;

    wire w_N1, w_N0;

    not N1(w_N1, c1);
    not N0(w_N0, c0);

    and A0(w_A0, x0, w_N1, w_N0);
    and A1(w_A1, x1, w_N1, c0);
    and A2(w_A2, x2, c1, w_N0);
    and A3(w_A3, x3, c1, c0);

    or r0(w_r0, w_A0, w_A1);
    or r2(w_r2, w_A2, w_A3);
    or r3(y, w_r0, w_r2);
endmodule
```

```verilog
`include "adder.v"
`include "mux4_1.v"

module one_bit_ALU (y, z, Cin, A, B, c1, c0);
    input Cin, A, B, c1, c0;
    output y, z;

    wire w_Sum, w_Cout;
    wire w_A0, w_N0, w_xr0;

    adder Adder1(w_Sum, w_Cout, Cin, A, B);
    and A0(w_A0, A, B);
    not N0(w_N0, A);
    xor xr0(w_xr0, A, B);

    mux4_1 mux1(y, w_Sum, w_A0, w_N0, 1'b0, c1, c0);
    mux4_1 mux2(z, w_Cout, 1'b0, 1'b0, w_xr0, c1, c0);
endmodule
```

```v
`include "1bit_ALU.v"

module stimulus_1bit_ALU;
    reg Cin, A, B;
    reg c1, c0;

    wire y;
    wire z;

    // instantiate the design block
    one_bit_ALU one_bit_ALU(y, z, Cin, A, B, c1, c0);

    // control the signals that drives the design block
    initial begin
        $dumpfile("1bitALU_TimingDiagram.vcd");
        $dumpvars(0, one_bit_ALU);
        Cin = 1'b0;
        A = 1'b0;
        B = 1'b0;
        c1 = 1'b0;
        c0 = 1'b0;

        // 0+0+0
        #5 begin
            A = 1'b0;
            B = 1'b0;
            Cin = 1'b0;
        end
        // 0+0+1
        #5 begin
            A = 1'b0;
            B = 1'b0;
            Cin = 1'b1;
        end
        // 0+1+0
        #5 begin
            A = 1'b0;
            B = 1'b1;
            Cin = 1'b0;
        end
        // 0+1+1
        #5 begin
            A = 1'b0;
            B = 1'b1;
            Cin = 1'b1;
        end
        // 1+0+0
        #5 begin
            A = 1'b1;
            B = 1'b0;
            Cin = 1'b0;
        end
        // 1+0+1
        #5 begin
            A = 1'b1;
            B = 1'b0;
            Cin = 1'b1;
        end
        // 1+1+0
        #5 begin
            A = 1'b1;
            B = 1'b1;
            Cin = 1'b0;
        end
        // 1+1+1
        #5 begin
            A = 1'b1;
            B = 1'b1;
            Cin = 1'b1;
        end

        #5 c0 = 1'b1;

        #5 begin
           A = 1'b0;
           B = 1'b0;
        end

        #5 begin
           A = 1'b0;
           B = 1'b1;
        end

        #5 begin
            A = 1'b1;
            B = 1'b0;
        end

        #5 begin
            c1 = 1'b1;
            c0 = 1'b0;
        end
       #5 begin
            A = 1'b0;
            B = 1'b0;
        end
        #5 begin
            A = 1'b0;
            B = 1'b1;
        end
        #5 begin
            A = 1'b1;
            B = 1'b0;
        end
        #5 begin
            A = 1'b1;
            B = 1'b1;
        end

        #5 c0 = 1'b1;
        #5 begin
            A = 1'b0;
            B = 1'b0;
        end
        #5 begin
            A = 1'b0;
            B = 1'b1;
        end
        #5 begin
            A = 1'b1;
            B = 1'b0;
        end
        #5 begin
            A = 1'b1;
            B = 1'b1;
        end

        #120 $finish;
    end

    initial
        $monitor($time, " Output Cin=%d A=%d B=%d c1=%d c0=%d y=%d z=%d",
        Cin, A, B, c1, c0, y, z);
endmodule
```

- **แสดงผลการทำงานที่ได้และอธิบายว่า 1-bit ALU ที่ได้ทำงานถูกต้องอย่างไร**

```
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)
```



จากผลลัพธ์ที่ได้ วงจรที่ออกแบบทำงานได้ถูกต้อง โดย เมื่อให้

- C1 = 0 และ C0 = 0: y = sum of (a, b, Cin), z = Cout ของ full adder ที่มีจะได้ 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: y = a and b, z = 0

```
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
```

ตามที่ต้องการ

Input a = 0 b = 0 output y = 0 z = 0

Input a = 0 b = 1 output y = 0 z = 0

Input a = 1 b = 0 output y = 0 z = 0

Input a = 1 b = 1 output y = 1 z = 0

- C1 = 1 และ C0 = 0: y = not a, 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
```

ตามที่ต้องการ

Input a = 0 b = 0 output y = 1 z = 0

Input a = 0 b = 1 output y = 1 z = 0

Input a = 1 b = 0 output y = 0 z = 0

Input a = 1 b = 1 output y = 0 z = 0

- C1 = 1 และ C0 = 1: y=0, z = a xor b

```
 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)
```

ตามที่ต้องการ

Input a = 0 b = 0 output y = 0 z = 0

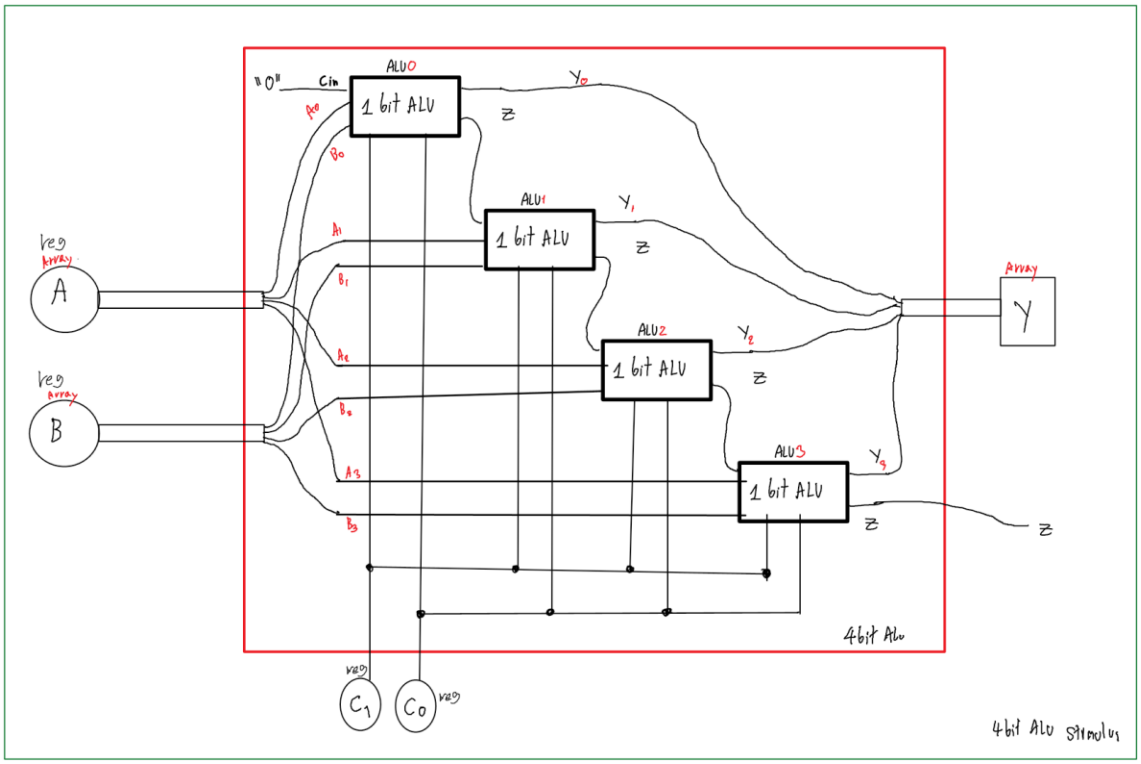
Input a = 0 b = 1 output y = 0 z = 1

Input a = 1 b = 0 output y = 0 z = 1

Input a = 1 b = 1 output 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

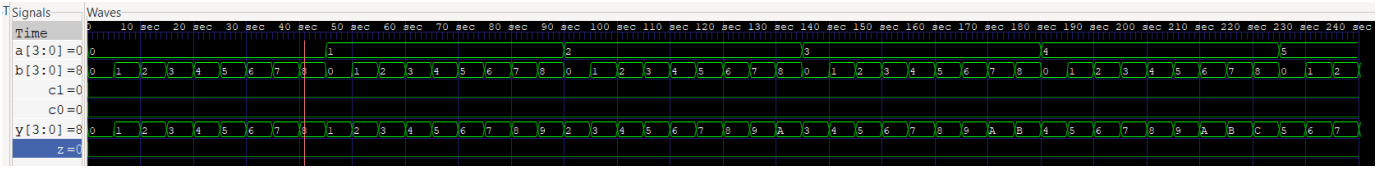


- เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus

```verilog
`include "1bit_ALU.v"

module four_bit_ALU (
    y, z,
    a, b,
    c1, c0
);

    input [3:0] a;
    input [3:0] b;

    output [3:0] y;
    output z;

    input c1, c0;

    wire w_z0, w_z1, w_z2;

    one_bit_ALU ALU0(y[0], w_z0, 1'b0, a[0], b[0], c1, c0);
    one_bit_ALU ALU1(y[1], w_z1, w_z0, a[1], b[1], c1, c0);
    one_bit_ALU ALU2(y[2], w_z2, w_z1, a[2], b[2], c1, c0);
    one_bit_ALU ALU3(y[3], z, w_z2, a[3], b[3], c1, c0);

endmodule
```

```
`include "4bitALU.v"

module stimulus_4bit_ALU;
    wire [3:0] y;
    wire z;

    reg [3:0] a;
    reg [3:0] b;

    reg c1, c0;

    four_bit_ALU four_bit_ALU(y, z, a, b, c1, c0);

    initial begin
        $dumpfile("4bitALU_TimingDiagram.vcd");
        $dumpvars(0, four_bit_ALU);
        a = 1'b0;
        b = 1'b0;
        c1 = 1'b0;
        c0 = 1'b0;
    end

    initial begin
        repeat(8) begin
            b = 1'b0;
            repeat(8) begin
                #5 b = b + 1'b1;
            end
            #5 a = a + 1'b1;
        end
        #5 a = a + 1'b1;
        a = 1'b0;

        c0 = 1'b1;
        repeat(8) begin
            b = 1'b0;
            repeat(8) begin
                #5 b = b + 1'b1;
            end
            #5 a = a + 1'b1;
        end
        #5 a = a + 1'b1;
        a = 1'b0;

        c1 = 1'b1;
        c0 = 1'b0;
        repeat(8) begin
            b = 1'b0;
            repeat(8) begin
                #5 b = b + 1'b1;
            end
            #5 a = a + 1'b1;
        end
        #5 a = a + 1'b1;
        a = 1'b0;

        c0 = 1'b1;
        repeat(8) begin
            b = 1'b0;
            repeat(8) begin
                #5 b = b + 1'b1;
            end
            #5 a = a + 1'b1;
        end
        #5 a = a + 1'b1;

        #1200 $finish;
    end

    initial
        $monitor($time, " Output c1=%d, c0=%d, a=%b=%d, b=%b=%d, y=%b=%d, z=%d", c1, c0, a, a, b, b, y, y, z);
endmodule

```

- **แสดงผลการทำงานที่ได้และอธิบายว่า 4-bit ALU ที่ได้ทำงานถูกต้องอย่างไร**



- C1 = 0 และ C0 = 0 จะได้ output คือ y0-y3 = sum(a, b) ซึ่งเป็นผลรวมของ input a , b ในแต่ละหลัก เมื่อบวกเกินจะมีการปัดเศษไปหลักถัดไป ซึ่งมี 4 bit โดย y3 เป็น MSB และ y0 เป็น LSB

```
    0 Output c1=0, c0=0, a=0000= 0, b=0000= 0, y=0000= 0, z=0
    5 Output c1=0, c0=0, a=0000= 0, b=0001= 1, y=0001= 1, z=0
   10 Output c1=0, c0=0, a=0000= 0, b=0010= 2, y=0010= 2, z=0
   15 Output c1=0, c0=0, a=0000= 0, b=0011= 3, y=0011= 3, z=0
   20 Output c1=0, c0=0, a=0000= 0, b=0100= 4, y=0100= 4, z=0
   25 Output c1=0, c0=0, a=0000= 0, b=0101= 5, y=0101= 5, z=0
   30 Output c1=0, c0=0, a=0000= 0, b=0110= 6, y=0110= 6, z=0
   35 Output c1=0, c0=0, a=0000= 0, b=0111= 7, y=0111= 7, z=0
   40 Output c1=0, c0=0, a=0000= 0, b=1000= 8, y=1000= 8, z=0
   45 Output c1=0, c0=0, a=0001= 1, b=0000= 0, y=0001= 1, z=0
   50 Output c1=0, c0=0, a=0001= 1, b=0001= 1, y=0010= 2, z=0
   55 Output c1=0, c0=0, a=0001= 1, b=0010= 2, y=0011= 3, z=0
   60 Output c1=0, c0=0, a=0001= 1, b=0011= 3, y=0100= 4, z=0
   65 Output c1=0, c0=0, a=0001= 1, b=0100= 4, y=0101= 5, z=0
   70 Output c1=0, c0=0, a=0001= 1, b=0101= 5, y=0110= 6, z=0
   75 Output c1=0, c0=0, a=0001= 1, b=0110= 6, y=0111= 7, z=0
   80 Output c1=0, c0=0, a=0001= 1, b=0111= 7, y=1000= 8, z=0
   85 Output c1=0, c0=0, a=0001= 1, b=1000= 8, y=1001= 9, z=0
```

```
  305 Output c1=0, c0=0, a=0110= 6, b=0111= 7, y=1101=13, z=0
  310 Output c1=0, c0=0, a=0110= 6, b=1000= 8, y=1110=14, z=0
  315 Output c1=0, c0=0, a=0111= 7, b=0000= 0, y=0111= 7, z=0
  320 Output c1=0, c0=0, a=0111= 7, b=0001= 1, y=1000= 8, z=0
  325 Output c1=0, c0=0, a=0111= 7, b=0010= 2, y=1001= 9, z=0
  330 Output c1=0, c0=0, a=0111= 7, b=0011= 3, y=1010=10, z=0
  335 Output c1=0, c0=0, a=0111= 7, b=0100= 4, y=1011=11, z=0
  340 Output c1=0, c0=0, a=0111= 7, b=0101= 5, y=1100=12, z=0
  345 Output c1=0, c0=0, a=0111= 7, b=0110= 6, y=1101=13, z=0
  350 Output c1=0, c0=0, a=0111= 7, b=0111= 7, y=1110=14, z=0
  355 Output c1=0, c0=0, a=0111= 7, b=1000= 8, y=1111=15, z=0
  360 Output c1=0, c0=0, a=1000= 8, b=1000= 8, y=0000= 0, z=1
```

- C1 = 0 และ C0 = 1: y = a and b, z = 0 - จะเห็นว่าหลักของ y เป็น 1 เมื่อหลักนั้น ๆ ของ a, b เป็น 1 เหมือนกัน ถูกต้องตามที่ต้องการ

```
440 Output c1=0, c0=1, a=0001= 1, b=0110= 6, y=0000= 0, z=0
445 Output c1=0, c0=1, a=0001= 1, b=0111= 7, y=0001= 1, z=0
450 Output c1=0, c0=1, a=0001= 1, b=1000= 8, y=0000= 0, z=0
455 Output c1=0, c0=1, a=0010= 2, b=0000= 0, y=0000= 0, z=0
460 Output c1=0, c0=1, a=0010= 2, b=0001= 1, y=0000= 0, z=0
465 Output c1=0, c0=1, a=0010= 2, b=0010= 2, y=0010= 2, z=0
470 Output c1=0, c0=1, a=0010= 2, b=0011= 3, y=0010= 2, z=0
475 Output c1=0, c0=1, a=0010= 2, b=0100= 4, y=0000= 0, z=0
480 Output c1=0, c0=1, a=0010= 2, b=0101= 5, y=0000= 0, z=0
485 Output c1=0, c0=1, a=0010= 2, b=0110= 6, y=0010= 2, z=0
490 Output c1=0, c0=1, a=0010= 2, b=0111= 7, y=0010= 2, z=0
495 Output c1=0, c0=1, a=0010= 2, b=1000= 8, y=0000= 0, z=0
500 Output c1=0, c0=1, a=0011= 3, b=0000= 0, y=0000= 0, z=0
505 Output c1=0, c0=1, a=0011= 3, b=0001= 1, y=0001= 1, z=0
510 Output c1=0, c0=1, a=0011= 3, b=0010= 2, y=0010= 2, z=0
515 Output c1=0, c0=1, a=0011= 3, b=0011= 3, y=0011= 3, z=0
520 Output c1=0, c0=1, a=0011= 3, b=0100= 4, y=0000= 0, z=0
525 Output c1=0, c0=1, a=0011= 3, b=0101= 5, y=0001= 1, z=0
530 Output c1=0, c0=1, a=0011= 3, b=0110= 6, y=0010= 2, z=0
535 Output c1=0, c0=1, a=0011= 3, b=0111= 7, y=0011= 3, z=0
```

- C1 = 1 และ C0 = 0: y = not a, z = 0 – bit ของ y จะตรงข้ามกับ bit ของ a ถูกต้องตามที่ต้องการ

```
210 Output c1=1, c0=0, a=0000= 0, b=0000= 0, y=1111=15, z=0
215 Output c1=1, c0=0, a=0000= 0, b=0001= 1, y=1111=15, z=0
220 Output c1=1, c0=0, a=0000= 0, b=0010= 2, y=1111=15, z=0
225 Output c1=1, c0=0, a=0000= 0, b=0011= 3, y=1111=15, z=0
230 Output c1=1, c0=0, a=0000= 0, b=0100= 4, y=1111=15, z=0
235 Output c1=1, c0=0, a=0001= 1, b=0000= 0, y=1110=14, z=0
240 Output c1=1, c0=0, a=0001= 1, b=0001= 1, y=1110=14, z=0
245 Output c1=1, c0=0, a=0001= 1, b=0010= 2, y=1110=14, z=0
250 Output c1=1, c0=0, a=0001= 1, b=0011= 3, y=1110=14, z=0
255 Output c1=1, c0=0, a=0001= 1, b=0100= 4, y=1110=14, z=0
260 Output c1=1, c0=0, a=0010= 2, b=0000= 0, y=1101=13, z=0
265 Output c1=1, c0=0, a=0010= 2, b=0001= 1, y=1101=13, z=0
270 Output c1=1, c0=0, a=0010= 2, b=0010= 2, y=1101=13, z=0
275 Output c1=1, c0=0, a=0010= 2, b=0011= 3, y=1101=13, z=0
280 Output c1=1, c0=0, a=0010= 2, b=0100= 4, y=1101=13, z=0
290 Output c1=1, c0=0, a=0011= 3, b=0001= 1, y=1100=12, z=0
295 Output c1=1, c0=0, a=0011= 3, b=0010= 2, y=1100=12, z=0
300 Output c1=1, c0=0, a=0011= 3, b=0011= 3, y=1100=12, z=0
305 Output c1=1, c0=0, a=0011= 3, b=0100= 4, y=1100=12, z=0
310 Output c1=1, c0=0, a=0100= 4, b=0100= 4, y=1011=11, z=0
```

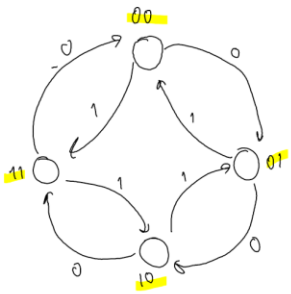
- C1 = 1 และ C0 = 1: y = 0, z = a3 xor b3 ถูกต้องตามที่ต้องการ

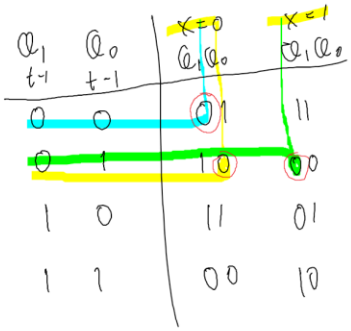
```
1400 Output c1=1, c0=1, a=0110= 6, b=0111= 7, y=0000= 0, z=0
1405 Output c1=1, c0=1, a=0110= 6, b=1000= 8, y=0000= 0, z=1
1410 Output c1=1, c0=1, a=0111= 7, b=0000= 0, y=0000= 0, z=0
1415 Output c1=1, c0=1, a=0111= 7, b=0001= 1, y=0000= 0, z=0
1420 Output c1=1, c0=1, a=0111= 7, b=0010= 2, y=0000= 0, z=0
1425 Output c1=1, c0=1, a=0111= 7, b=0011= 3, y=0000= 0, z=0
1430 Output c1=1, c0=1, a=0111= 7, b=0100= 4, y=0000= 0, z=0
1435 Output c1=1, c0=1, a=0111= 7, b=0101= 5, y=0000= 0, z=0
1440 Output c1=1, c0=1, a=0111= 7, b=0110= 6, y=0000= 0, z=0
1445 Output c1=1, c0=1, a=0111= 7, b=0111= 7, y=0000= 0, z=0
1450 Output c1=1, c0=1, a=0111= 7, b=1000= 8, y=0000= 0, z=1
1455 Output c1=1, c0=1, a=1000= 8, b=1000= 8, y=0000= 0, z=0
1460 Output c1=1, c0=1, a=1001= 9, b=1000= 8, y=0000= 0, z=0
```

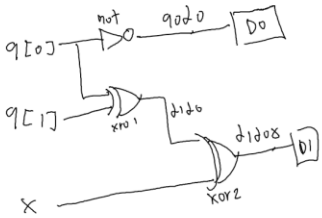
3) พัฒนาวงจรนับขึ้น,นับลง 2 บิต (00,01,10,11) โดยให้มีสัญญาณอินพุต (X) เป็นตัวควบคุมการนับขึ้นหรือนับลง โดยสัญญาณจะนับขึ้นก็ต่อเมื่อ X=0 และจะนับลงเมื่อ X=1 ออกแบบโดยใช้ Module D Flipflop

```
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

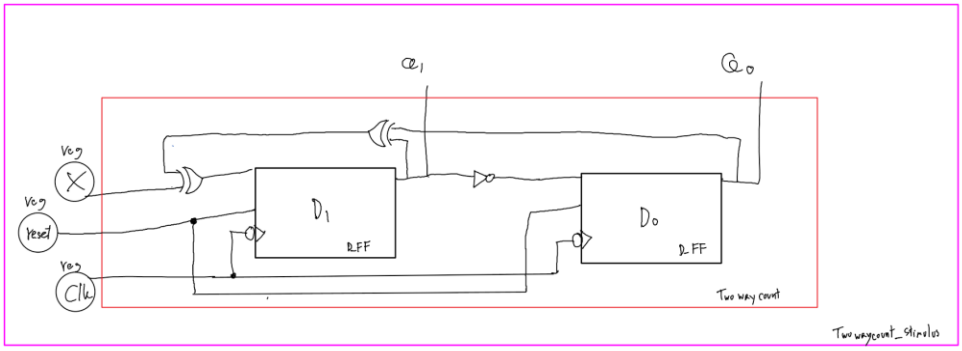






$$D_1 = \bar{X}\bar{Q}_1Q_0 + \bar{X}Q_1\bar{Q}_0 + X\bar{Q}_1\bar{Q}_0 + XQ_1Q_0 = X \oplus Q_1 \oplus Q_0$$

$$D_0 = \bar{Q}_1\bar{Q}_0 + Q_1\bar{Q}_0 = \bar{Q}_0$$



- เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus

stimulus_two_way_count.v

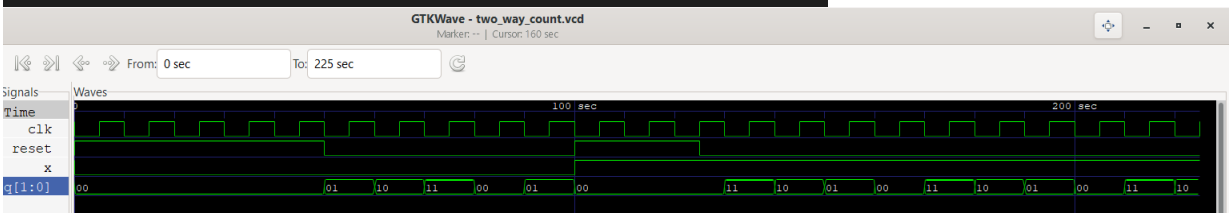
```
`include "two_way_count.v"

module stimulus_two_way_count;
    reg x;
    reg clk, reset;
    wire [1:0]q;

    two_way_count c0(q, x, clk, reset);

    initial begin
        $dumpfile("two_way_count.vcd");
        $dumpvars(0, c0);
        clk = 1'b0;
        x = 1'b0;
        reset = 1'b0;
    end
    always
        #5  clk = ~clk;
    // TODO: Test two count
    initial begin
        // x = 1'b0;
reset = 1'b1;
x = 1'b0;
#50 reset = 1'b0;
#50 x = 1'b1;
#0 reset = 1'b1;
#25 reset = 1'b0;
#50 reset = 1'b0;
#50 $finish;
    end

    initial
        $monitor($time, " input: x=%d reset=%d output: q=%d", x, reset, q);
endmodule
```

two_way_count.v

```
`include "D_FF.v"

module two_way_count(q, x, clk, reset);
    input x, clk, reset;
    output [1:0] q;

    wire w_xor0, w_xor1, w_not0;

    xor xor1(w_xor1, q[0], q[1]);
    xor xor0(w_xor0, x, w_xor1);
    D_FF D0(q[0], w_not0, clk, reset);

    not not0(w_not0, q[0]);
    D_FF D1(q[1], w_xor0, clk, reset);
endmodule
```

- **แสดงผลการทำงานที่ได้**

```
                   0 input: x=0 reset=1 output: q=0
                  50 input: x=0 reset=0 output: q=1
                  60 input: x=0 reset=0 output: q=2
                  70 input: x=0 reset=0 output: q=3
                  90 input: x=0 reset=0 output: q=1
                 100 input: x=1 reset=1 output: q=0
                 125 input: x=1 reset=0 output: q=0
                 130 input: x=1 reset=0 output: q=3
                 140 input: x=1 reset=0 output: q=2
                 150 input: x=1 reset=0 output: q=1
                 160 input: x=1 reset=0 output: q=0
                 170 input: x=1 reset=0 output: q=3
                 190 input: x=1 reset=0 output: q=1
                 200 input: x=1 reset=0 output: q=0
                 220 input: x=1 reset=0 output: q=2
stimulus_two_way_count.v:29: $finish called at 225 (1s)
```

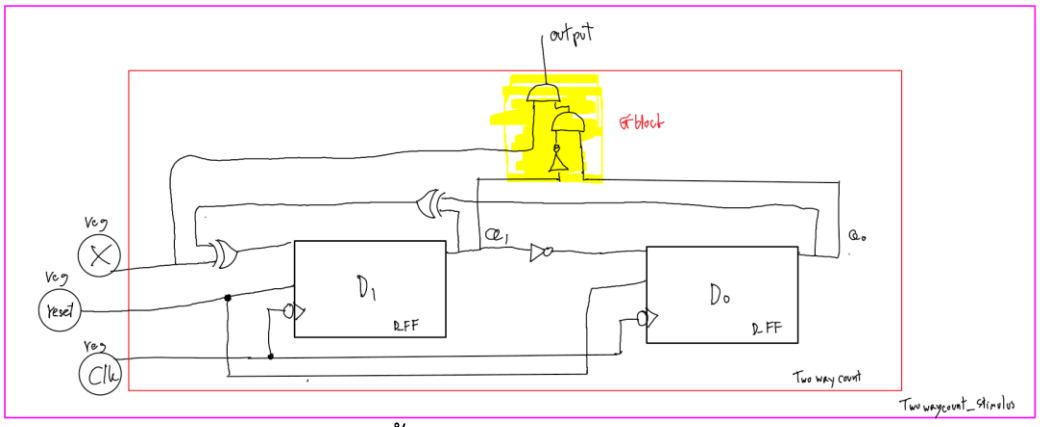


อธิบายว่าวงจรนับที่ได้ทำงานถูกต้องอย่างไร

- เมื่อ D_FF เจอ negedge clk จะทำการเปลี่ยน state โดย มี D_FF 2 ตัว เป็นตัวจำว่าตอนนี้อยู่ State ไหน
- เมื่อ x เป็น 0 วงจรจะนับขึ้นจาก 00 -> 01 -> 10 -> 11 -> 00 ไปเรื่อย ๆ
- เมื่อ x เป็น 1 วงจรจะลงจาก 00 -> 11 -> 10 -> 01 -> 00 ไปเรื่อย ๆ ตามที่เราต้องการ เนื่องจากเรามี F Box ที่เป็น control box ที่คอยควบคุม logic ที่จะไปทำให้ D FF จำค่าต่างๆ โดย D_FF1 Q1 MSB จะถูกควบคุมด้วย [Q1 t-1 xor Q0 t-1] xor x และ D_FF0 Q0 LSB จะถูกควบคุมด้วย not Q1 t-1 เป็นไปตามที่ต้องการ

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

- ออกแบบ โครงสร้างของ Verilog



- เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus

sequential > 4 > two_way_count_g.v

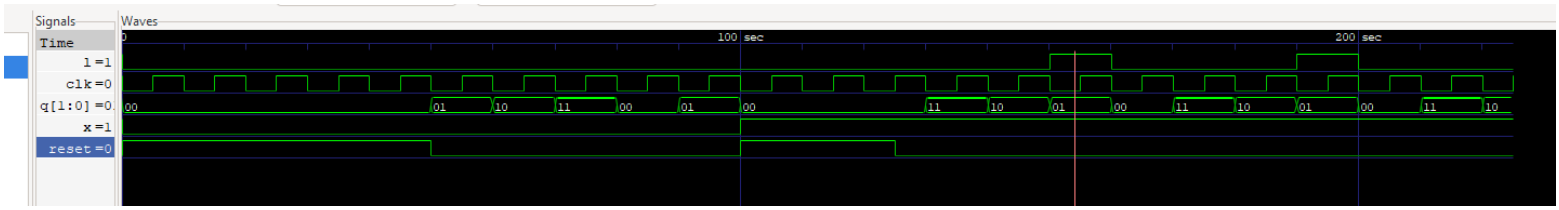
```verilog
`include "D_FF.v"

module two_way_count_g(l ,x, clk, reset);
    input x, clk, reset;
    wire [1:0] q;

    wire w_xor0, w_xor1, w_not0;

    xor xor1(w_xor1, q[0], q[1]);
    xor xor0(w_xor0, x, w_xor1);
    D_FF D0(q[0], w_not0, clk, reset);

    not not0(w_not0, q[0]);
    D_FF D1(q[1], w_xor0, clk, reset);

    output l;
    wire w_not1, w_and0;

    not not1(w_not1,q[1]);
    and and1(w_and0, w_not1,q[0]);
    and and2(l,x,w_and0);

endmodule
```

sequential > 4 > stimulus_two_way_count_g.v

```v
`include "two_way_count_g.v"

module stimulus_two_way_count_g;
    reg x;
    reg clk, reset;


    wire l;
    assign L = l;

    two_way_count_g c0( l, x, clk, reset);

    initial begin
        $dumpfile("two_way_count_g.vcd");
        $dumpvars(0, c0);
        clk = 1'b0;
        x = 1'b0;
        reset = 1'b0;
    end
    always
        #5  clk = ~clk;
    // TODO: Test two count
    initial begin
        // x = 1'b0;
reset = 1'b1;
x = 1'b0;
#50 reset = 1'b0;
#50 x = 1'b1;
#0 reset = 1'b1;
#25 reset = 1'b0;
#50 reset = 1'b0;
#50 $finish;
    end

    initial
        $monitor($time, " output L = %d   reset = %d x = %d",L, reset, x);
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 not Q1 AND Q2 AND X output(l) จะเป็น 1

เมื่อ Q1 = 0 Q0 = 1 X = 1 จะเป็นกรณีเดียวที่ได้ “1” และกรณีอื่น จะได้ผลลัพธ์เป็น “0”