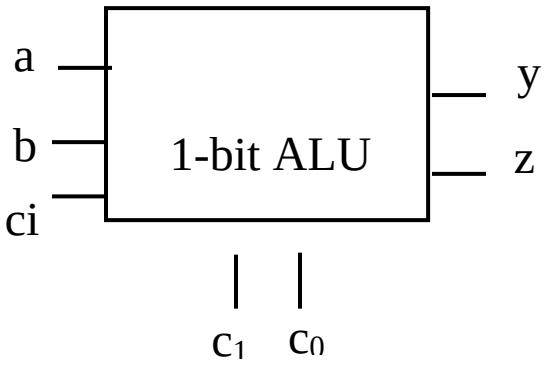
Lab 10 (Team Project) เตรียมพร้อมการสอบ Verilog

จงออกแบบวงจรและเขียน Verilog เพื่อตรวจสอบการทำงาน

ชื่อ-นามสกุล.....พงศกร รัตนพันธ์..... รหัสนักศึกษา...630610749..... ตอนที่.....001...
ชื่อ-นามสกุล..... รหัส
นักศึกษา..... ตอนที่.....
ชื่อ-นามสกุล..... รหัส
นักศึกษา..... ตอนที่.....
ชื่อ-นามสกุล..... รหัส
นักศึกษา..... ตอนที่.....

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



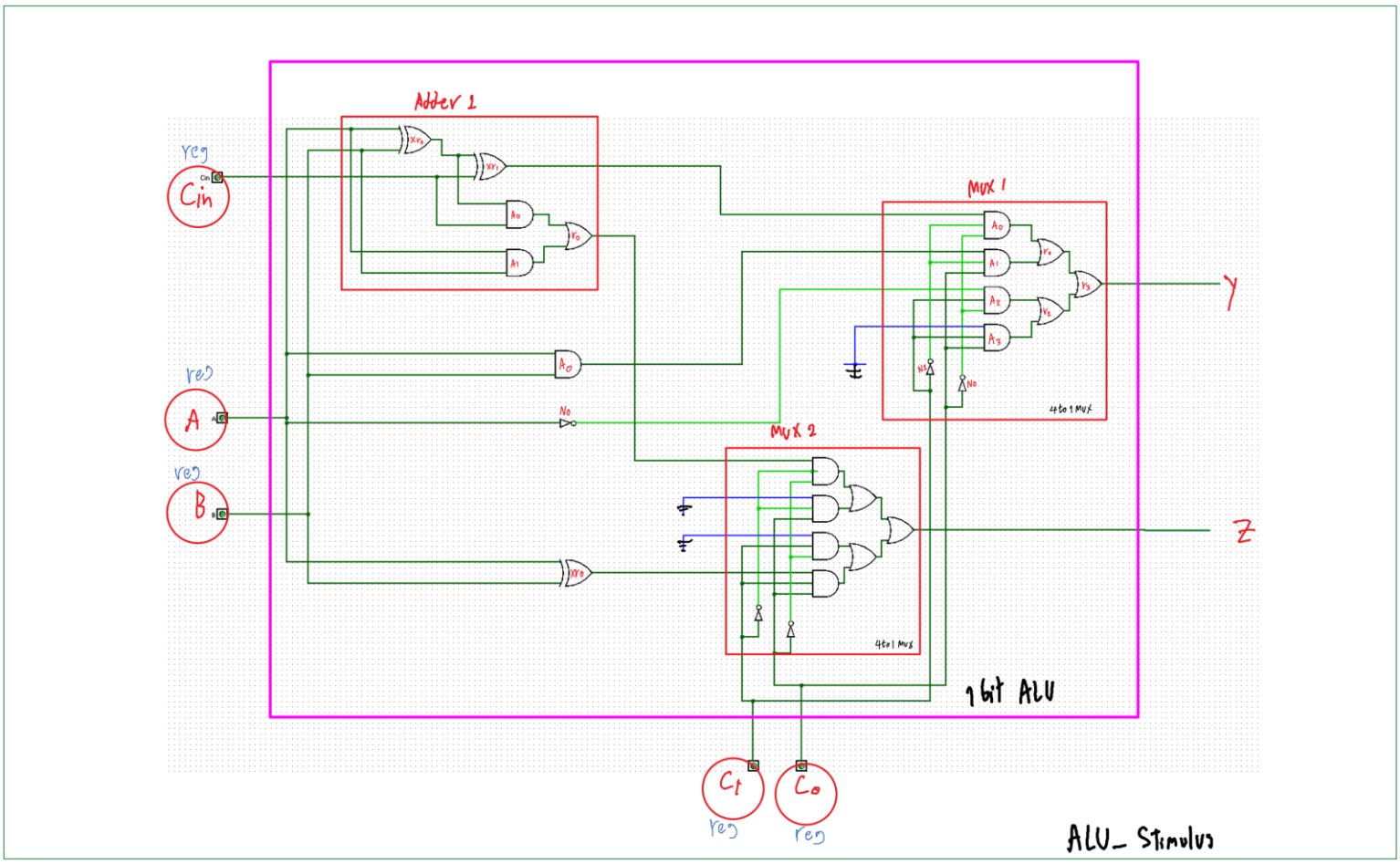
โดยมีตารางการปฏิบัติการดังนี้

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

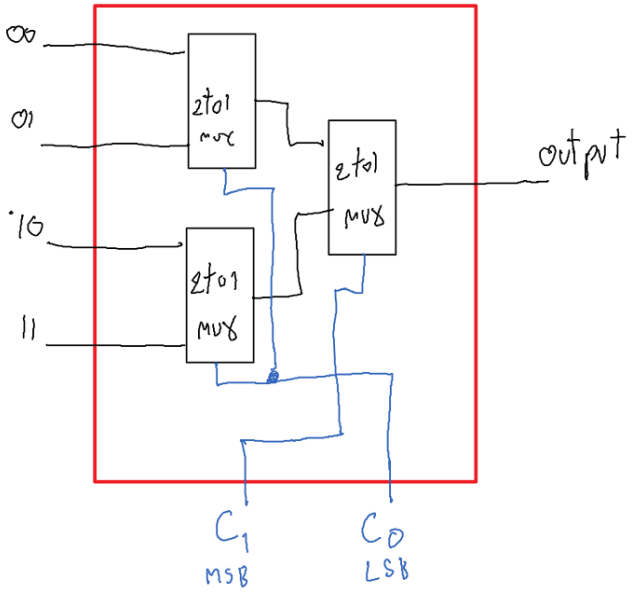
-

-

- ออกแบบ โครงสร้างของ Verilog



optional for 4to1 mux



- เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus

- **แสดงผลการทำงานที่ได้ และ**
-
-
-

- **อธิบายว่า 1-bit ALU ที่ได้ทำงานถูกต้องอย่างไร**

  : จากผลลัพธ์ที่ได้ วงจรที่ออกแบบทำงานได้ถูกต้อง โดย เมื่อให้

- **C1 = 0 และ C0 = 0 จะได้ y = sum(s) และ z = carry out ของ full adder ที่มี inputs a b และ cin 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 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 = 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 xor b ถูกต้องตามต้องการ ดังนี้**

  a = 0 b = 0 y = 0 z = 0

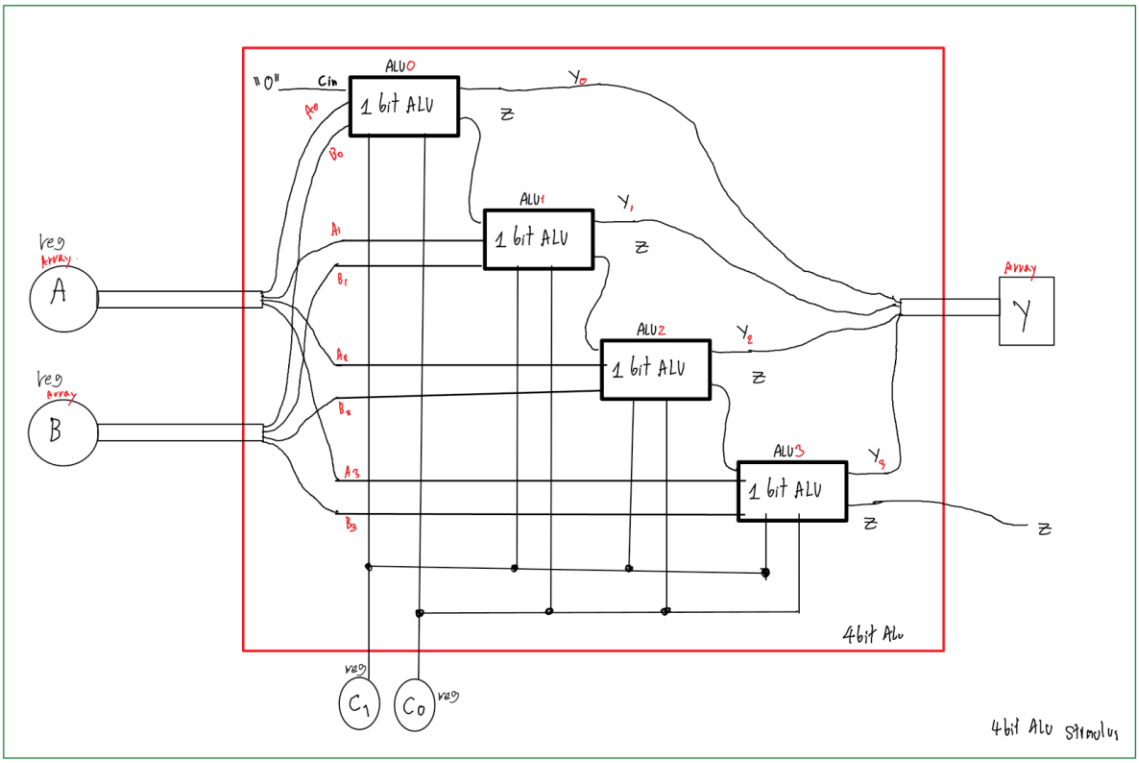
  a = 0 b = 1 y = 0 z = 1

  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
- **แสดงผลการทำงานที่ได้**
-
-
-
-
-
-
-

อธิบายว่า 4-bit ALU ที่ได้ทำงานถูกต้องอย่างไร จากผลลัพธ์ที่ได้ วงจรที่ออกแบบทำงานได้ถูกต้อง โดย เมื่อให้

- **C1 = 0 และ C0 = 0 จะได้ output คือ y0-y3 = sum(s) ซึ่งเป็นผลรวมของ input a , b ในแตะละหลัก เมื่อบวกเกินจะมีการปัดเศษไปหลักถัดไป ซึ่งมี 4 bit โดย**
- **y3 เป็น MSB และ y0 เป็น LSB**
- **C1 = 0 และ C0 = 1 จะได้ y = a and b และ z = 0 ถูกต้องตามต้องการ ดังนี้**

  a0-a3 = 0 b0-b3 = 0 y0-y3 = 0 z = 0

  a0-a3 = 0 b0-b3 = 1 y0-y3 = 0 z = 0

  a0-a3 = 1 b0-b3 = 0 y0-y3 = 0 z = 0

  a0-a3 = 1 b0-b3 = 1 y0-y3 = 1 z = 0

- **C1 = 1 และ C0 = 0 จะได้ y = not a และ z = 0 ถูกต้องตามต้องการ ดังนี้**

  a0-a3 = 0 b0-b3 = 0 y0-y3 = 1 z = 0

  a0-a3 = 0 b0-b3 = 1 y0-y3 = 1 z = 0

  a0-a3 = 1 b0-b3 = 0 y0-y3 = 0 z = 0

  a0-a3 = 1 b0-b3 = 1 y0-y3 = 0 z = 0

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

  a0-a3 = 0 b0-b3 = 0 y0-y3 = 0 z = 0

  a0-a3 = 0 b0-b3 = 1 y0-y3 = 0 z = 1

  a0-a3 = 1 b0-b3 = 0 y0-y3 = 0 z = 1

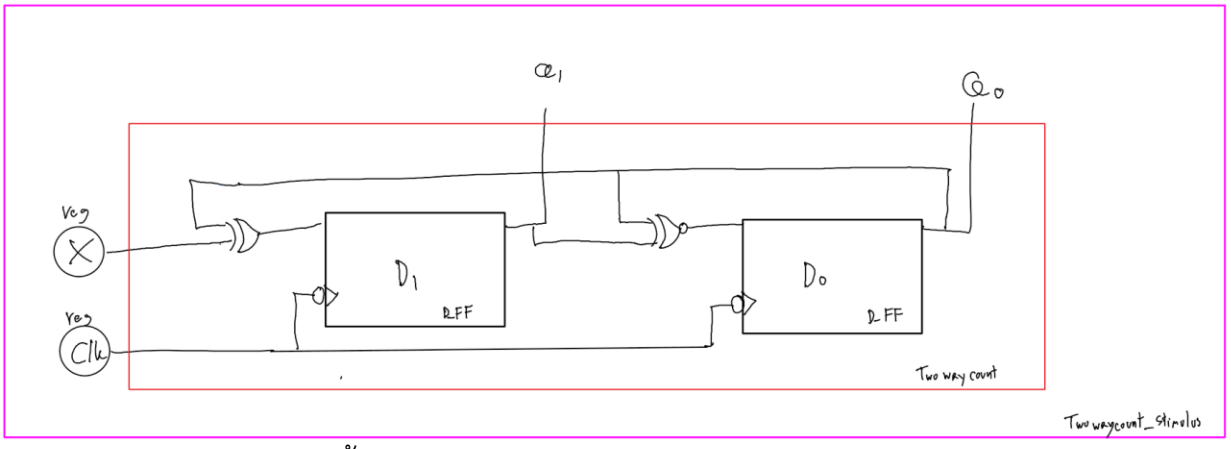
  a0-a3 = 1 b0-b3 = 1 y0-y3 = 0 z = 0

-

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



- เขียน Code ด้วย Verilog ทั้งส่วน Design และ ส่วน Stimulus
- **แสดงผลการทำงานที่ได้**
-
-
-

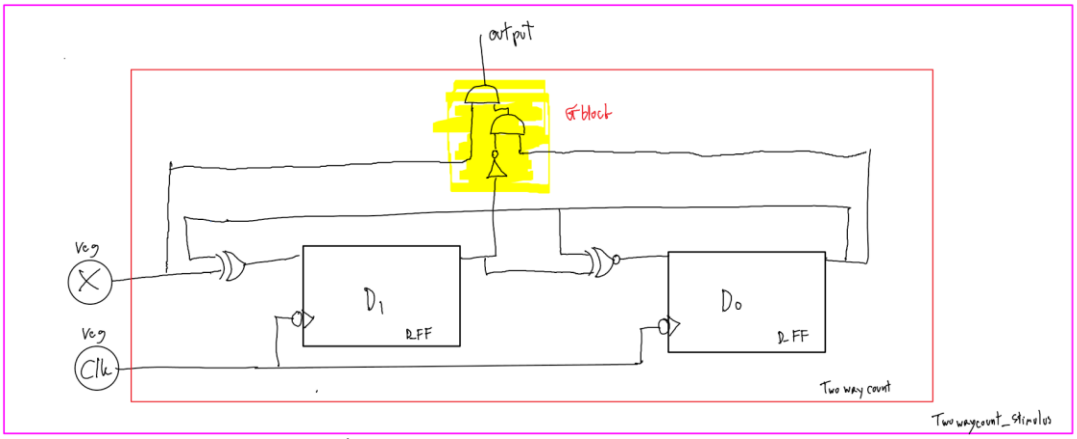
- **อธิบายว่าวงจรนับที่ได้ทำงานถูกต้องอย่างไร**

  เมื่อ D_FF เจอ negedge clk จะทำการเปลี่ยน state โดย มี D_FF เป็นตัวจำว่าตอนนี้อยู่ State ไหน 2 ตัว

-
- **เมื่อ x เป็น 0 วงจรจะนับขึ้นจาก 00 -> 01 -> 10 -> 11 -> 00 ไปเรื่อย ๆ**
- **เมื่อ x เป็น 1 วงจรจะลงจาก 00 -> 11 -> 10 -> 01 -> 00 ไปเรื่อย ๆ**
- **ตามที่เราต้องการเนื่องจากเรามี F Box ที่เป็น control box ที่คอยควบคุม logic ที่จะไปทำให้ D FF จำค่าต่างๆ**
- **โดย D_FF MSB จะถูกควบคุมด้วย X xor Q0 t-1**
- **และ D_FF MSB จะถูกควบคุมด้วย Q1 t-1 xor Q0 t-1**
- **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
- **แสดงผลการทำงานที่ได้ และอธิบายว่าวงจรนับที่ได้ทำงานถูกต้องอย่างไร**

จาก output ในแต่ละ state ของ วงจร จะมี 00 01 10 11

ซึ่งเมื่อนำ state not Q1 AND Q2 AND X output จะเป็น 1 เมื่อ

Q1 = 0 Q0 = 1 X = 1 กรณีเดียวที่ได้ “1” และกรณีอื่น จะได้ผลลัพเป็น “0”