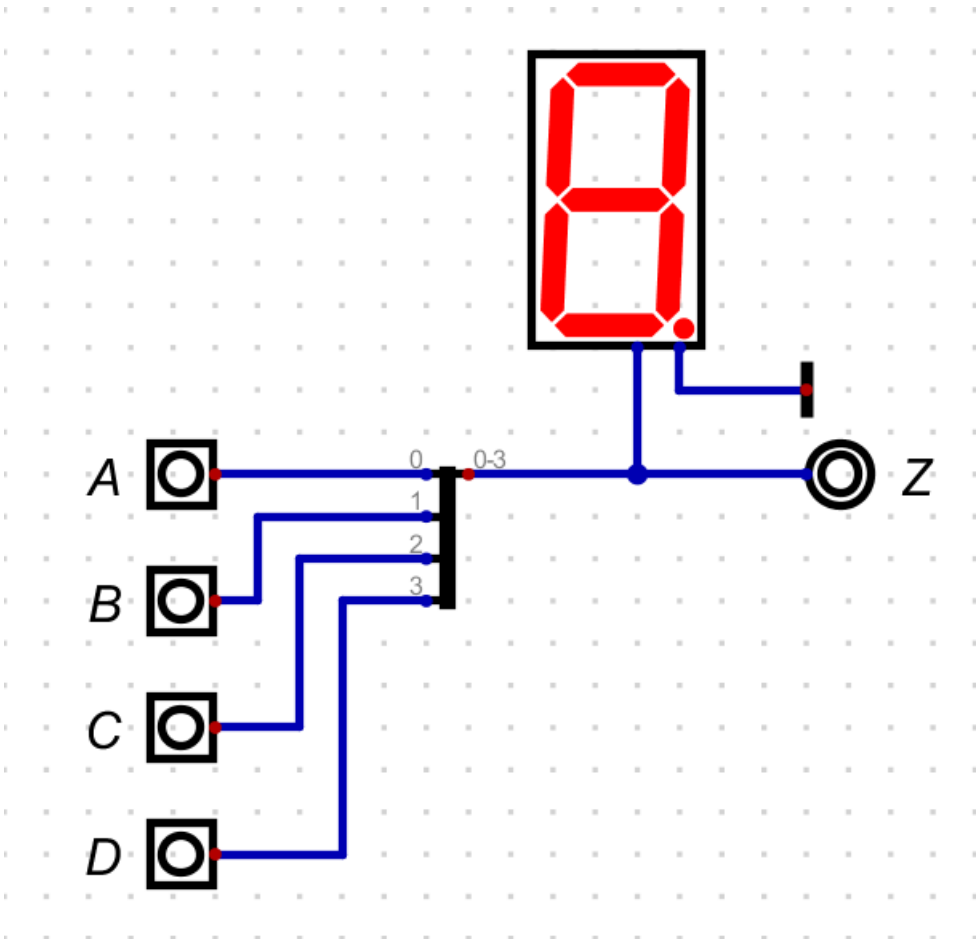
Wire Connection

ให้นิสิตสร้างวงจรที่มี Input คือ A, B, C, D ขนาด 1 Bit และ Output คือ Z ขนาด 4 บิท และ 7-Segment Hex Display โดยให้ใช้อุปกรณ์ Splitter/Merger รวม Input ให้มีขนาด 4 บิทโดยให้เป็นไปตามรูปแบบ DCBA ซึ่งมี D เป็น Most Significant Bit แล้วส่งผลลัพธ์ที่ได้ไปที่ Z และ 7-Segment Hex Display

ตัวอย่าง :



ข้อมูลนำเข้า

- A ขนาด 1 Bit
- B ขนาด 1 Bit
- C ขนาด 1 Bit
- D ขนาด 1 Bit

ข้อมูลส่งออก

- Z ขนาด 4 Bit
- 7-Segment Hex Display (optional)

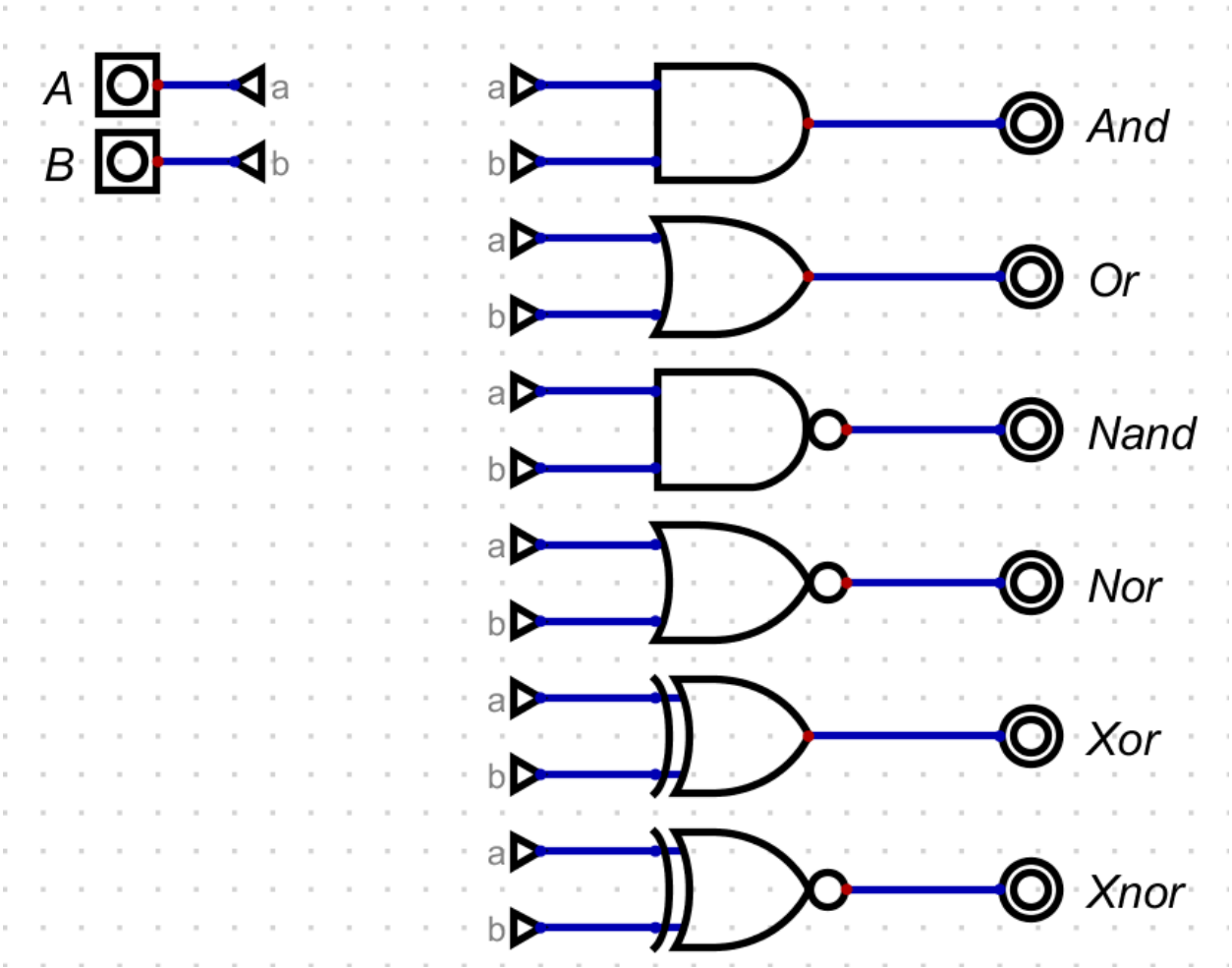
ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานได้ถูกต้องตาม Input ทุกรูปแบบ

Logic Gate

ให้นิสิตสร้างวงจรที่มี Input คือ A,B ขนาด 1 Bit และ Output คือ And, Or, Nand, Nor, Xor, Xnor ขนาด 1 Bit โดยที่ Output แต่ละอันเกิดจากการนำ Input A,B มาผ่าน Logic Gate ตามชื่อ Output นั้นๆ

ตัวอย่าง :



ข้อมูลนำเข้า

- A ขนาด 1 Bit
- B ขนาด 1 Bit

ข้อมูลส่งออก

- And ขนาด 1 Bit
- Or ขนาด 1 Bit
- Nand ขนาด 1 Bit
- Nor ขนาด 1 Bit
- Xor ขนาด 1 Bit
- Xnor ขนาด 1 Bit

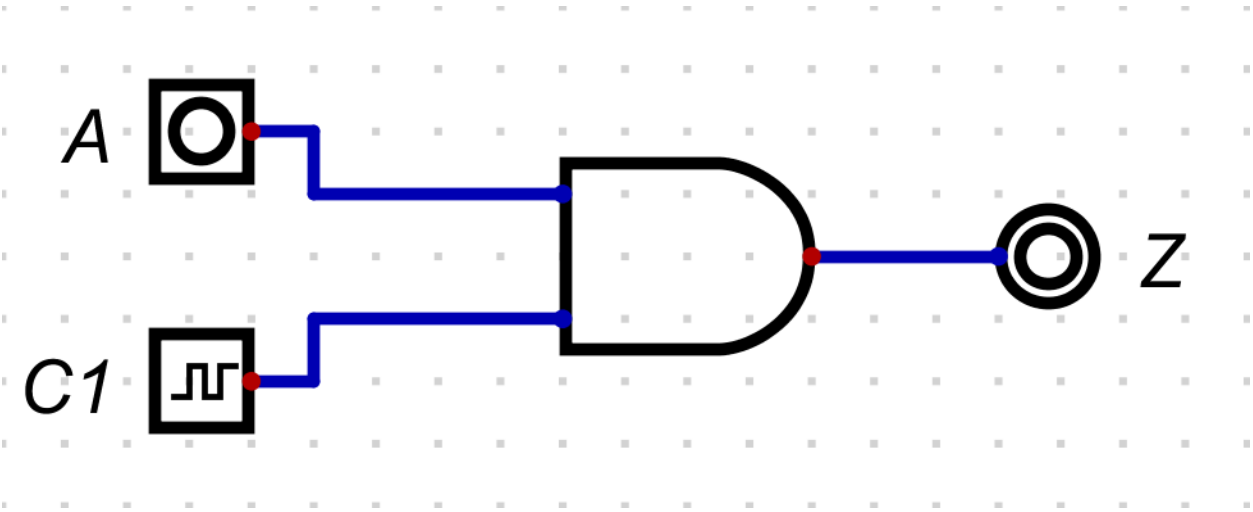
ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานได้ถูกต้องตาม Input ทุกรูปแบบ

Clock Gate

ให้นิสิตสร้างวงจรที่มี Input คือ A ขนาด 1 Bit, C1 เป็น Clock และ Output คือ Z ขนาด 1 บิท โดยให้นำ A และ C1 มา And กันแล้วส่งผลลัพธ์ไปออกที่ Z และให้นิสิตสังเกตุกราฟของผลลัพธ์ที่เกิดขึ้น

ตัวอย่าง :



ข้อมูลนำเข้า

- A ขนาด 1 Bit
- C1 เป็น Clock

ข้อมูลส่งออก

- Z ขนาด 1 Bit

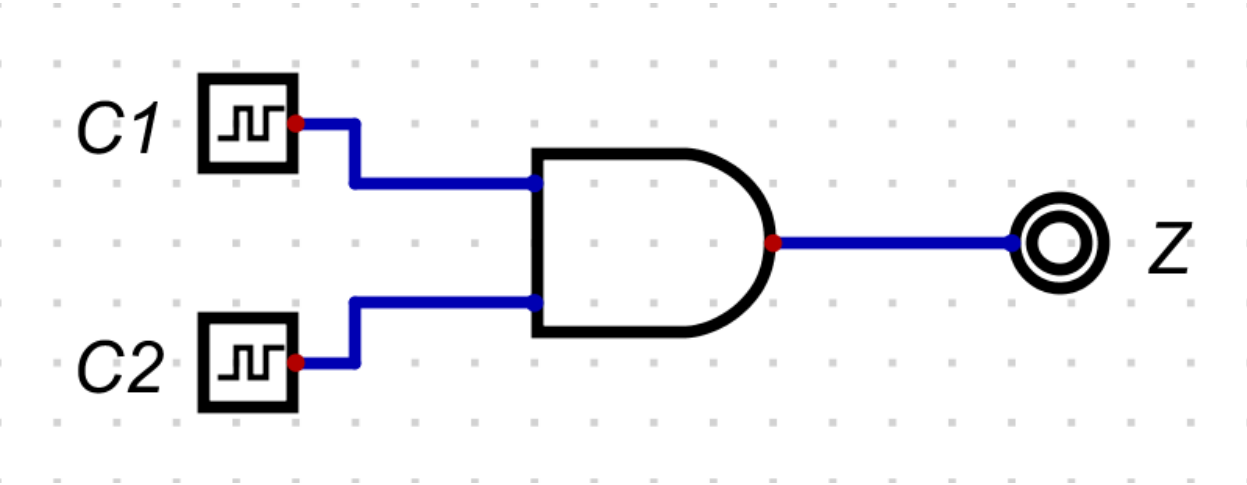
ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานได้ถูกต้องตาม Input ทุกรูปแบบ

Dual Clock

ให้นิสิตสร้างวงจรที่มี Input คือ C1,C2 เป็น Clock และ Output คือ Z ขนาด 1 บิท โดยให้นำ C1 และ C2 มา And กันแล้วส่งผลลัพธ์ออกไปที่ Z และให้นิสิตทดลองปรับความถี่ของ C1 และ C2 และสังเกตุกราฟของผลลัพธ์ที่เกิดขึ้น

ตัวอย่าง :



ข้อมูลนำเข้า

- C1 เป็น Clock
- C2 เป็น Clock

ข้อมูลส่งออก

- Z ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานได้ถูกต้องตาม Input ทุกรูปแบบ

XOR (Sum of Products)

ให้นิสิตสร้างวงจร XOR ที่มี Input คือ A, B ขนาด 1 Bit และ Output คือ Output ขนาด 1 บิท โดยให้สร้างวงจรโดยใช้สมการบูลีนแบบ Sum of Products เท่านั้น

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

รูปที่ 1 ตารางค่าความจริงของวงจร Xor

ข้อมูลนำเข้า

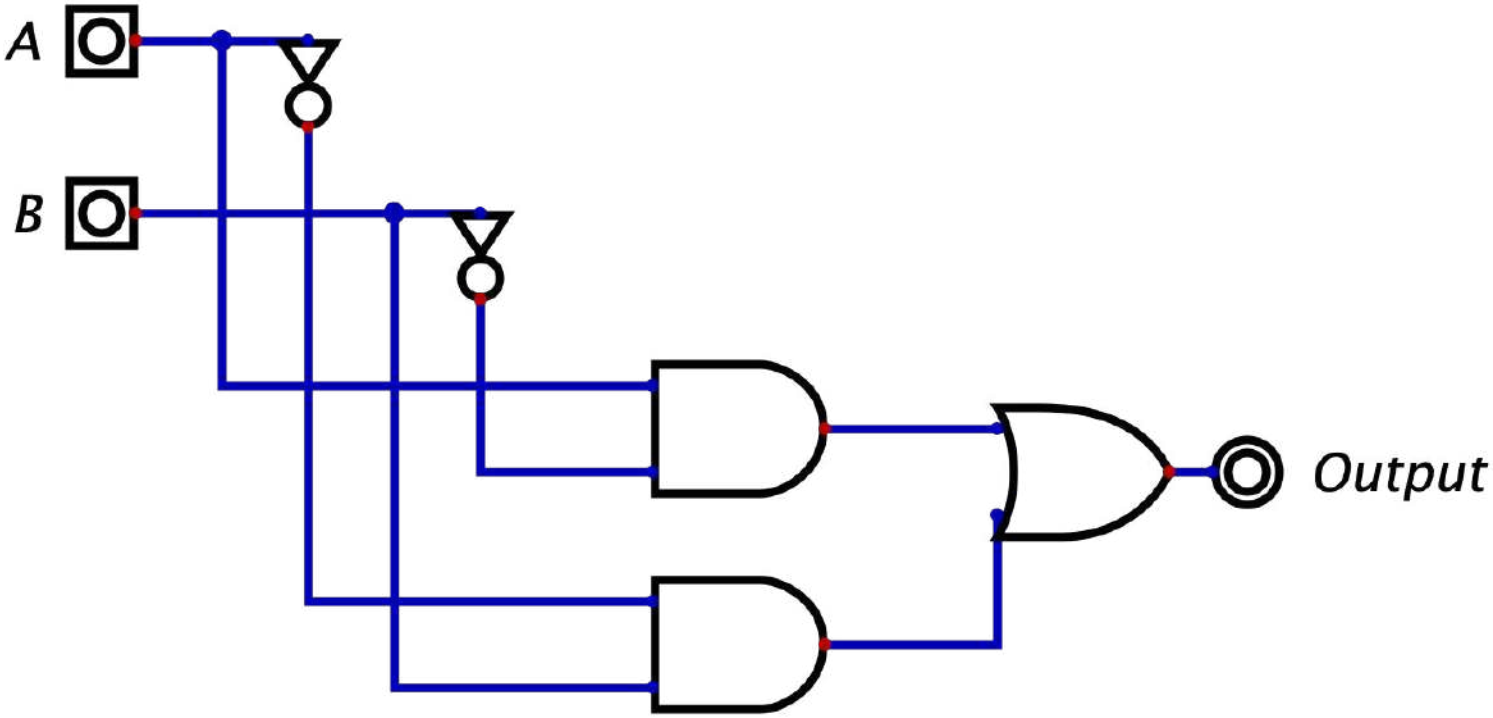
- A ขนาด 1 Bit
- B ขนาด 1 Bit

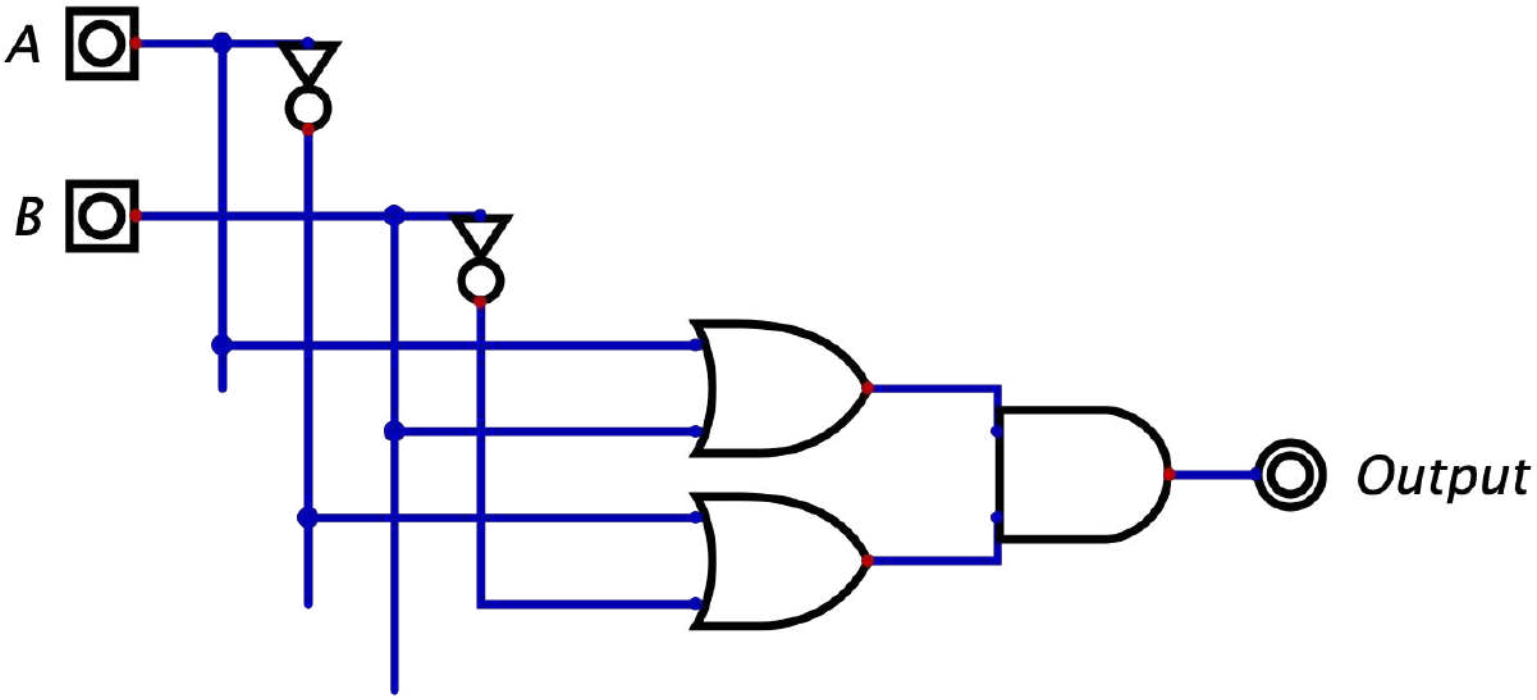
ข้อมูลส่งออก

- Output ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานถูกต้องตาม Input ทุกรูปแบบ


A
B
Output


A
B
Output

XOR (Product of Sums)

ให้นิสิตสร้างวงจร XOR ที่มี Input คือ A, B ขนาด 1 Bit และ Output คือ Output ขนาด 1 บิท โดยให้สร้างวงจรโดยใช้สมการบูลีนแบบ Product of Sums เท่านั้น

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

รูปที่ 1 ตารางค่าความจริงของวงจร Xor

ข้อมูลนำเข้า

- A ขนาด 1 Bit
- B ขนาด 1 Bit

ข้อมูลส่งออก

- Output ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานถูกต้องตาม Input ทุกรูปแบบ

FullAdder (Sum of Products)

ให้นิสิตสร้างวงจร FullAdder ขนาด 1 Bit ที่มี Input คือ A, B, Cin ขนาด 1 Bit และ Output คือ Sum, Cout ขนาด 1 Bit ซึ่ง Sum คือผลรวมของ Input ทั้งหมด สำหรับหลักปัจจุบันและ Cout คือตัวทดสำหรับหลักต่อไป โดยให้สร้างวงจรโดยใช้สมการบูลีนแบบ Sum of Products

A	B	Cin	Sum	Cout
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

รูปที่ 1. ตารางค่าความจริงของ FullAdder

ข้อมูลนำเข้า

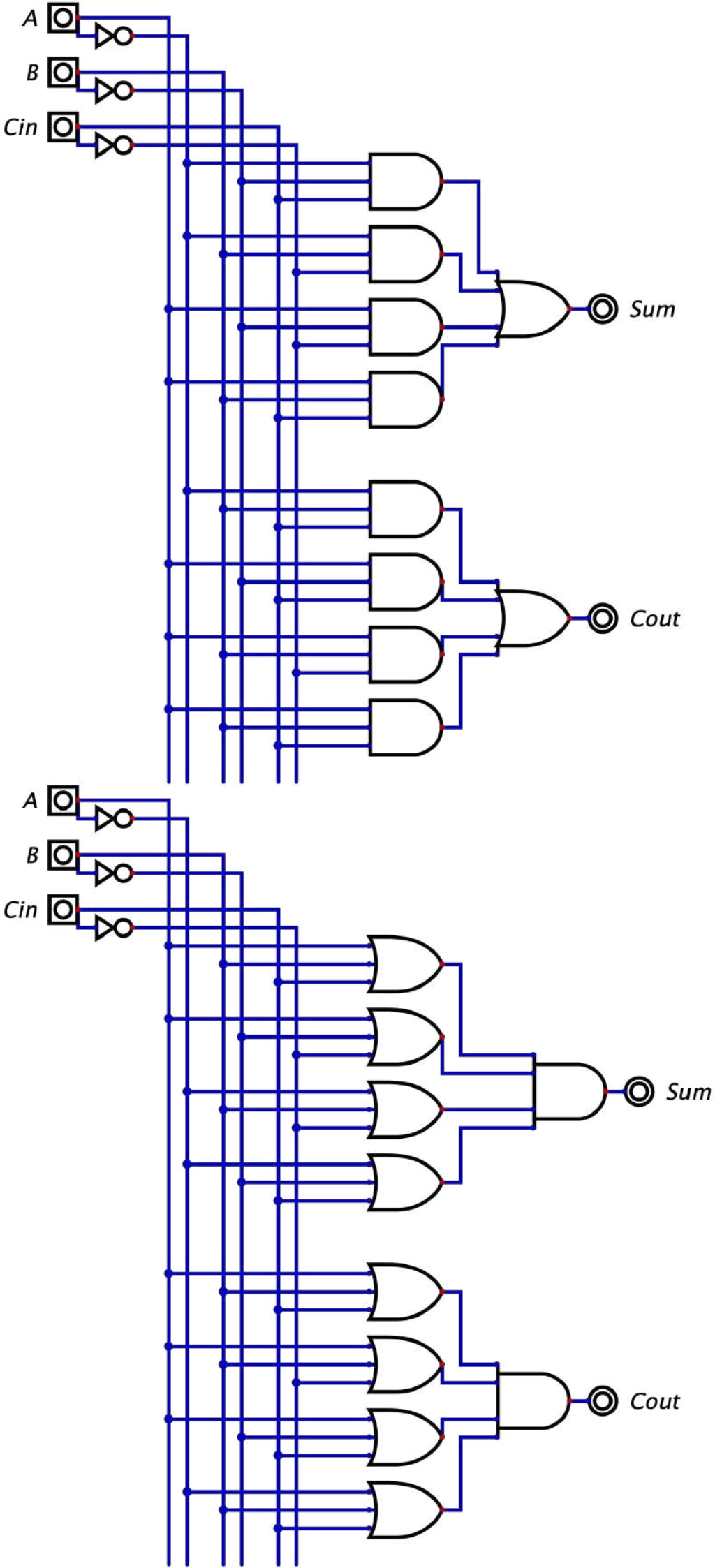
- A ขนาด 1 Bit
- B ขนาด 1 Bit
- Cin ขนาด 1 Bit

ข้อมูลส่งออก

- Sum ขนาด 1 Bit
- Cout ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 50% Cin มีค่าเป็น 0
- 50% Cin มีค่าเป็น 1


A
B
Cin
Sum
Cout
A
B
Cin
Sum
Cout

FullAdder (Product of Sums)

ให้นิสิตสร้างวงจร FullAdder ขนาด 1 Bit ที่มี Input คือ A, B, Cin ขนาด 1 Bit และ Output คือ Sum, Cout ขนาด 1 Bit ซึ่ง Sum คือผลรวมของ Input ทั้งหมด สำหรับหลักปัจจุบันและ Cout คือตัวทดสำหรับหลักต่อไป โดยให้สร้างวงจรโดยใช้สมการบูลีนแบบ Product of Sums

A	B	Cin	Sum	Cout
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

รูปที่ 1. ตารางค่าความจริงของ FullAdder

ข้อมูลนำเข้า

- A ขนาด 1 Bit
- B ขนาด 1 Bit
- Cin ขนาด 1 Bit

ข้อมูลส่งออก

- Sum ขนาด 1 Bit
- Cout ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 50% Cin มีค่าเป็น 0
- 50% Cin มีค่าเป็น 1

Comparator (Sum of Products)

ให้นิสิตสร้างวงจร comparator ที่มี 4 input คือ A, B, C และ D ขนาด 1 Bit และมี 3 output คือ Z1, Z2 และ Z3 ขนาด 1 Bit โดยที่ A และ B ประกอบเป็นค่าของเลขจำนวนที่หนึ่ง (N1) และ C และ D ประกอบเป็นค่าของเลขจำนวนที่สอง (N2) เช่น ถ้า AB = 10 เลข N1 ก็มีค่า 10 (เท่ากับ 2 ในฐานสิบ) ค่าของ Z แสดงผลการเปรียบเทียบขนาดเลขทั้งสองจำนวน โดย Z1 เป็น 1 เมื่อ N1 > N2 Z2 เป็น 1 เมื่อ N1 < N2 และ Z3 เป็น 1 เมื่อ N1 = N2 (จะสังเกตว่า ที่ input ใดๆ จะมีค่า Z เป็นหนึ่งเพียงตัวเดียวเท่านั้น) โดยให้สร้างวงจรโดยใช้สมการบูลีนแบบ Sum of Products

ข้อมูลนำเข้า

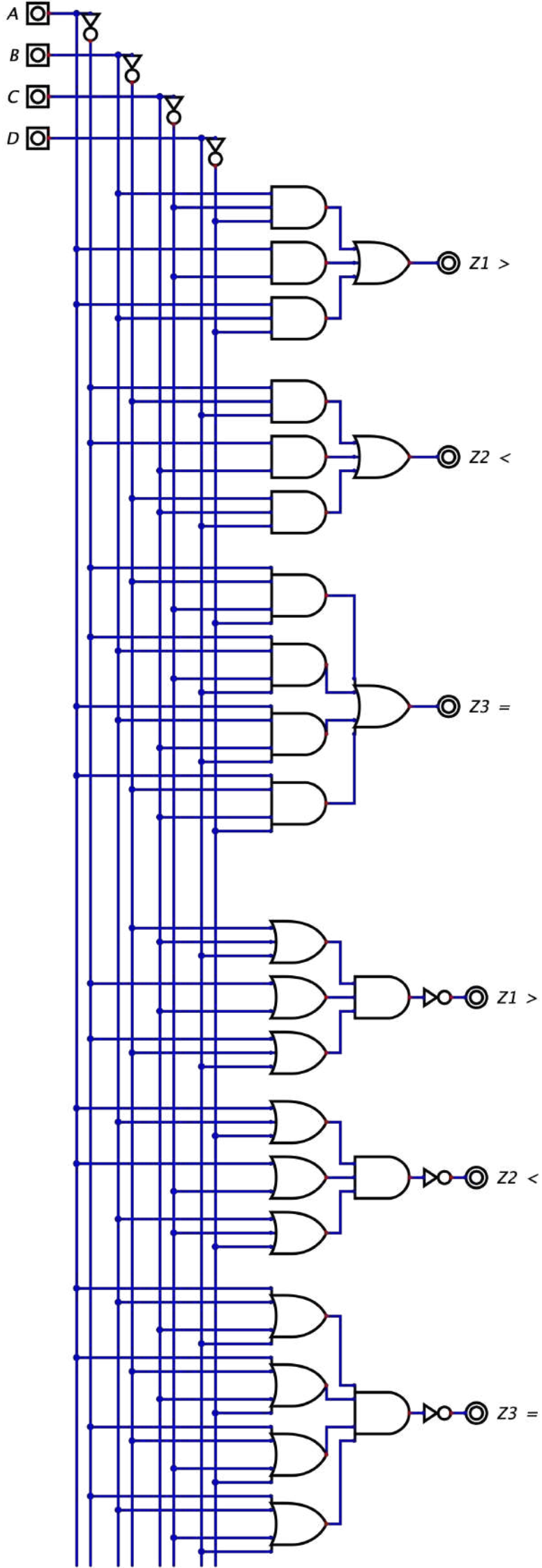
- A ขนาด 1 Bit
- B ขนาด 1 Bit
- C ขนาด 1 Bit
- D ขนาด 1 Bit

ข้อมูลส่งออก

- Z1 ขนาด 1 Bit
- Z2 ขนาด 1 Bit
- Z3 ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 33% N1 มีค่าเท่ากับ N2
- 33% N1 มีค่ามากกว่า N2
- 34% N1 มีค่าน้อยกว่า N2


A
B
C
D
Z1 >
Z2 <
Z3 =
Z1 >
Z2 <
Z3 =

Comparator (Product of Sums)

ให้นิสิตสร้างวงจร comparator ที่มี 4 input คือ A, B, C และ D ขนาด 1 Bit และมี 3 output คือ Z1, Z2 และ Z3 ขนาด 1 Bit โดยที่ A และ B ประกอบเป็นค่าของเลขจำนวนที่หนึ่ง (N1) และ C และ D ประกอบเป็นค่าของเลขจำนวนที่สอง (N2) เช่น ถ้า AB = 10 เลข N1 ก็มีค่า 10 (เท่ากับ 2 ในฐานสิบ) ค่าของ Z แสดงผลการเปรียบเทียบขนาดเลขทั้งสองจำนวน โดย Z1 เป็น 1 เมื่อ N1 > N2 Z2 เป็น 1 เมื่อ N1 < N2 และ Z3 เป็น 1 เมื่อ N1 = N2 (จะสังเกตว่า ที่ input ใดๆ จะมีค่า Z เป็นหนึ่งเพียงตัวเดียวเท่านั้น) โดยให้สร้างวงจรโดยใช้สมการบูลีนแบบ Product of Sums

ข้อมูลนำเข้า

- A ขนาด 1 Bit
- B ขนาด 1 Bit
- C ขนาด 1 Bit
- D ขนาด 1 Bit

ข้อมูลส่งออก

- Z1 ขนาด 1 Bit
- Z2 ขนาด 1 Bit
- Z3 ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 33% N1 มีค่าเท่ากับ N2
- 33% N1 มีค่ามากกว่า N2
- 34% N1 มีค่าน้อยกว่า N2

Multiplexer (Sum of Products)

ให้นิสิตสร้างวงจร ที่มี 3 input คือ X0, X1 และ Selector ขนาด 1 Bitและมี 1 output คือ Z ขนาด 1 Bit โดยที่ ค่าของ Z ควบคุมโดย input Selector คือ ถ้า input Selector เป็น 0 ค่าของ Z จะเป็น X0 แต่ถ้า input Selector เป็น 1 ค่าของ Z จะเป็น X1 วงจรนี้เรียกว่า Multiplexer เนื่องจากวงจรเลือก 1 input จาก 2 input จะเรียกสั้นๆว่า MUX 2:1 โดยให้สร้างวงจรโดยใช้สมการบูลีนแบบ Sum of Products

ข้อมูลนำเข้า

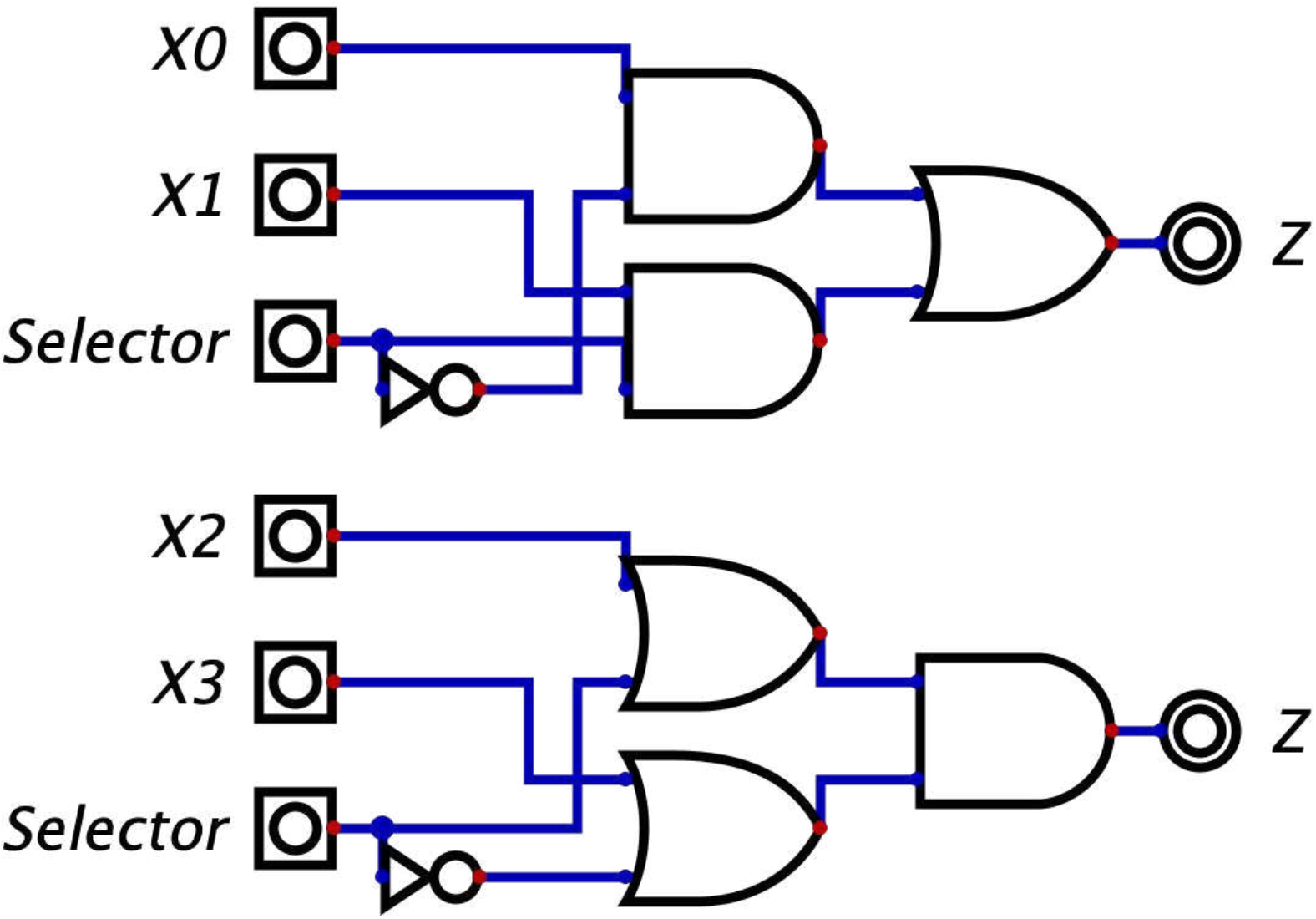
- X0 ขนาด 1 Bit
- X1 ขนาด 1 Bit
- Selector ขนาด 1 Bit

ข้อมูลส่งออก

- Z ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 50% Selector มีค่าเป็น 0
- 50% Selector มีค่าเป็น 1


X0
X1
Selector
Z
X2
X3
Selector
Z

Multiplexer (Product of Sums)

ให้นิสิตสร้างวงจร ที่มี 3 input คือ X0, X1 และ Selector ขนาด 1 Bitและมี 1 output คือ Z ขนาด 1 Bit โดยที่ ค่าของ Z ควบคุมโดย input Selector คือ ถ้า input Selector เป็น 0 ค่าของ Z จะเป็น X0 แต่ถ้า input Selector เป็น 1 ค่าของ Z จะเป็น X1 วงจรนี้เรียกว่า Multiplexer เนื่องจากวงจรเลือก 1 input จาก 2 input จะเรียกสั้นๆว่า MUX 2:1 โดยให้สร้างวงจรโดยใช้สมการบูลีนแบบ Product of Sums

ข้อมูลนำเข้า

- X0 ขนาด 1 Bit
- X1 ขนาด 1 Bit
- Selector ขนาด 1 Bit

ข้อมูลส่งออก

- Z ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 50% Selector มีค่าเป็น 0
- 50% Selector มีค่าเป็น 1

Mux4:1 (Sum of Products)

ให้นิสิตสร้างวงจร MUX 4:1 คือวงจรมี 4 input คือ X0, X1, X2 และ X3 ขนาด 1 Bit และมี input ที่ใช้ในการเลือก 2 input คือ S1 S0 ขนาด 1 Bit และ 1 output คือ Z ขนาด 1 Bit โดยค่าที่ออกมาที่ Z จะเป็น input ใดขึ้นกับค่าของ S1 S0 ตามตาราง

S1	S0	Z
0	0	X0
0	1	X1
1	0	X2
1	1	X3

ข้อแนะนำ : ใช้วงจรจากข้อ Multiplexer

ข้อมูลนำเข้า

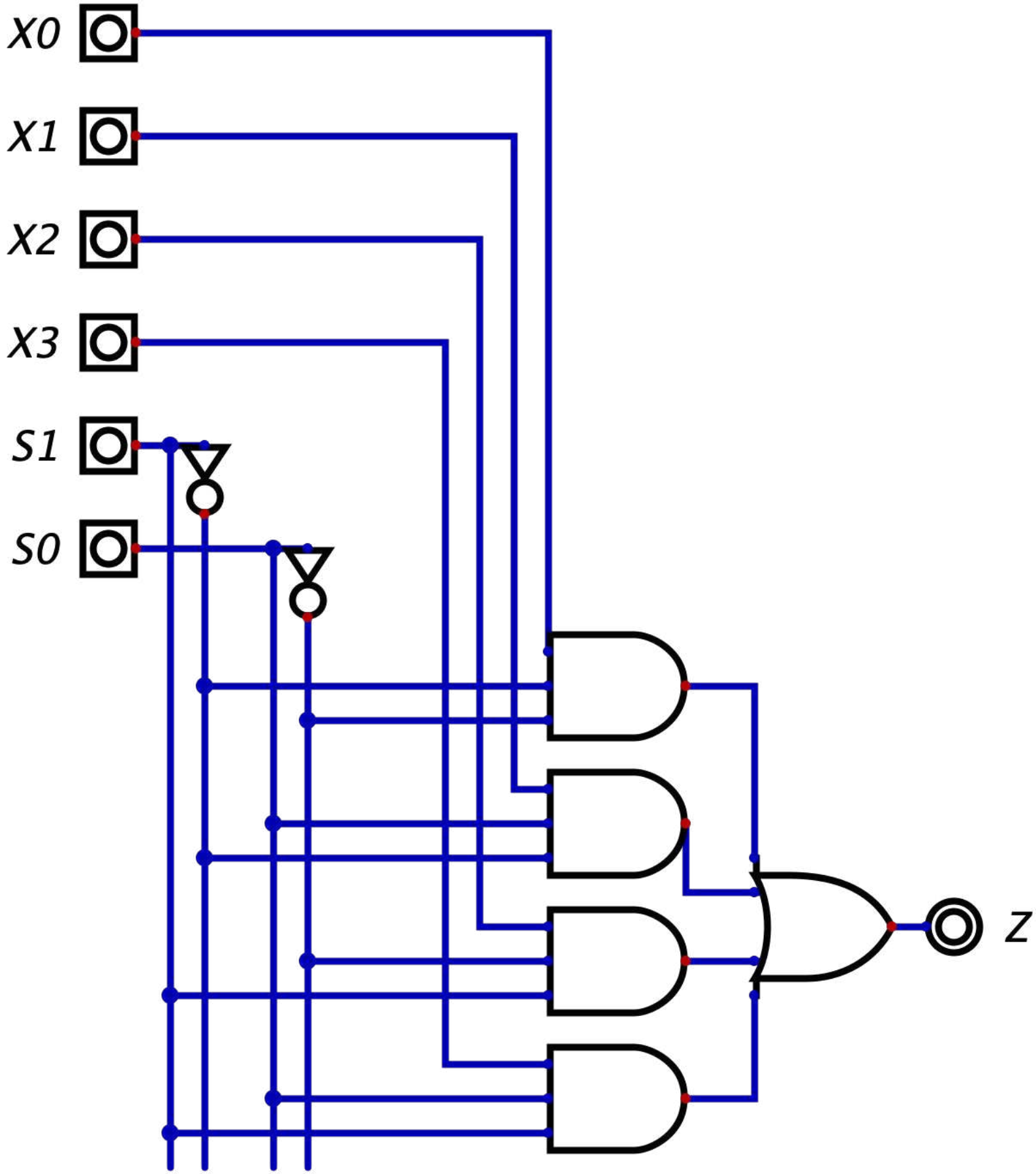
- X0 ขนาด 1 Bit
- X1 ขนาด 1 Bit
- X2 ขนาด 1 Bit
- X3 ขนาด 1 Bit
- S0 ขนาด 1 Bit
- S1 ขนาด 1 Bit

ข้อมูลส่งออก

- Z ขนาด 1 Bit

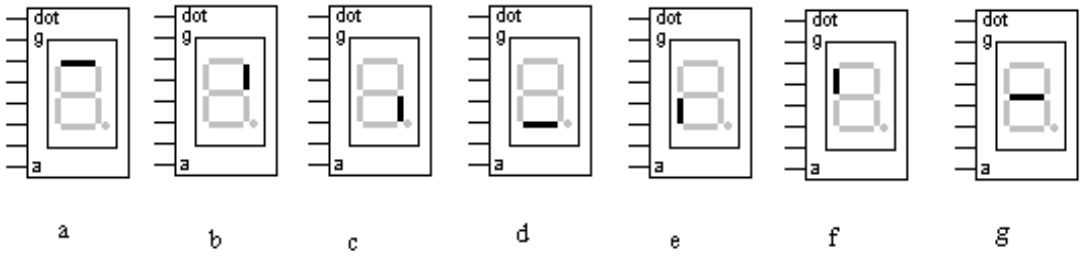
ชุดข้อมูลทดสอบ

- 25% S1 มีค่าเป็น 0 และ S2 มีค่าเป็น 0
- 25% S1 มีค่าเป็น 0 และ S2 มีค่าเป็น 1
- 25% S1 มีค่าเป็น 1 และ S2 มีค่าเป็น 0
- 25% S1 มีค่าเป็น 1 และ S2 มีค่าเป็น 1

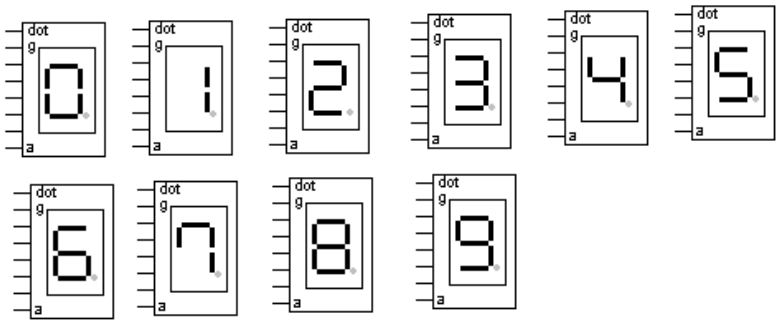

X0
X1
X2
X3
S1
S0
Z

BCD to Seven Segment Decoder

ให้นิสิตสร้างวงจร BCD to Seven Segment Decoder ที่มี Input คือ In ขนาด 4 bit และ Output คือ A,B,C,D,E,F,G ขนาด 1 Bit โดยให้แสดงค่าของ In ในรูปแบบเลขฐาน 10 ออกมาผ่านทางอุปกรณ์ Seven Segment (ให้แสดงแค่ช่วง 0-9) โดยที่ให้ A,B,C,D,E,F,G แทน Segment ต่างๆบนอุปกรณ์ Seven Segment ดังรูป



รูปที่ 1 : Segment ต่างๆบนอุปกรณ์ Seven Segment



รูปที่ 2 : การแสดงผลของอุปกรณ์ Seven Segment

ข้อมูลนำเข้า

- In ขนาด 4 Bit

ข้อมูลส่งออก

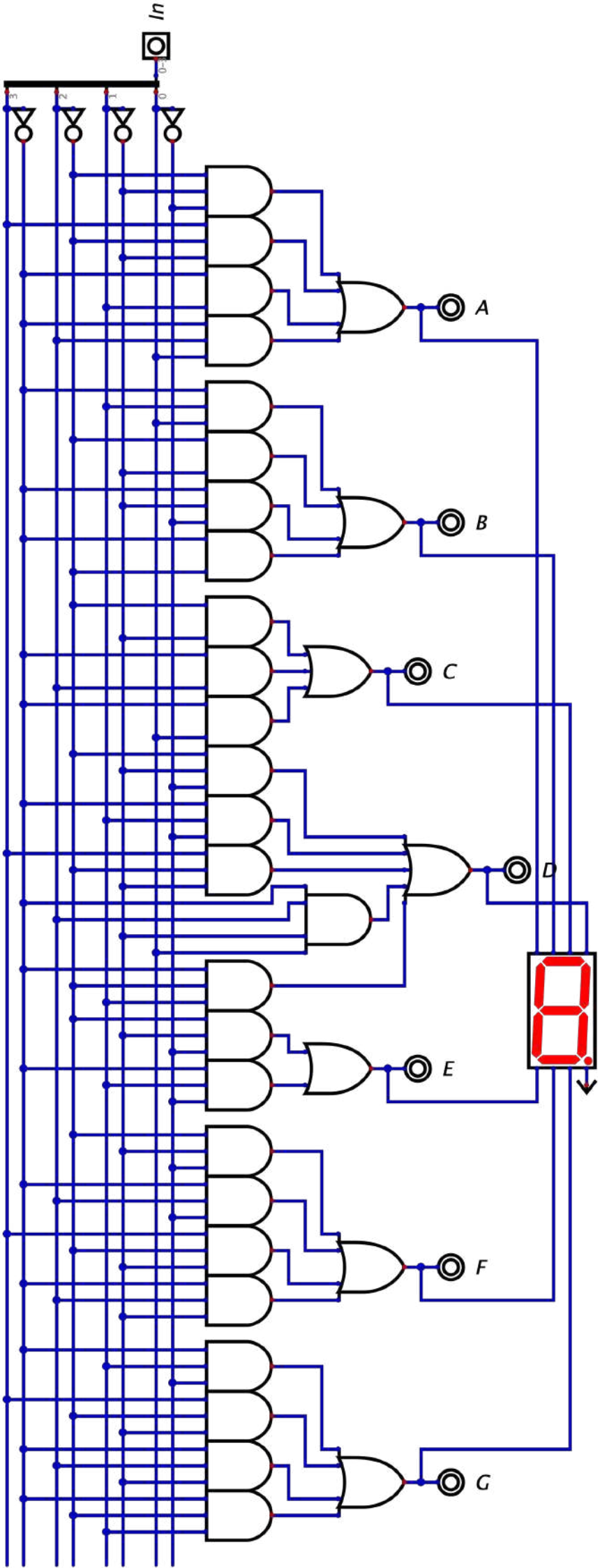
- A ขนาด 1 Bit
- B ขนาด 1 Bit
- C ขนาด 1 Bit
- D ขนาด 1 Bit
- E ขนาด 1 Bit
- F ขนาด 1 Bit
- G ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 10% In มีค่าเป็น 0
- 10% In มีค่าเป็น 1
- 10% In มีค่าเป็น 2
- 10% In มีค่าเป็น 3
- 10% In มีค่าเป็น 4
- 10% In มีค่าเป็น 5
- 10% In มีค่าเป็น 6

ชุดข้อมูลทดสอบ (ต่อ)

- 10% In มีค่าเป็น 7
- 10% In มีค่าเป็น 8
- 10% In มีค่าเป็น 9


In
3
2
1
0
A
B
C
D
E
F
G

DigLoLab ASCII

ให้นิสิตสร้างวงจร DigLoLab ASCII ที่มี Input คือ In ขนาด 3 bit และ Output คือ Z ขนาด 8 Bit โดยที่ Z คือค่า ASCII ของตัวอักษรลำดับที่ In ของคำว่า “DigLoLab” โดยที่ลำดับของตัวอักษรจะเริ่มจาก 0 ถึง 7 โดยจะเริ่มต้นลำดับ 0 ที่ตัวที่ตัว ‘D’ และจบลงที่ลำดับ 7 ที่ตัว ‘b’ ถ้าค่า In เป็น 0 ค่าของ Z ควรจะเป็น 44 ฐาน 16 (ตัว ‘D’)

ข้อมูลนำเข้า

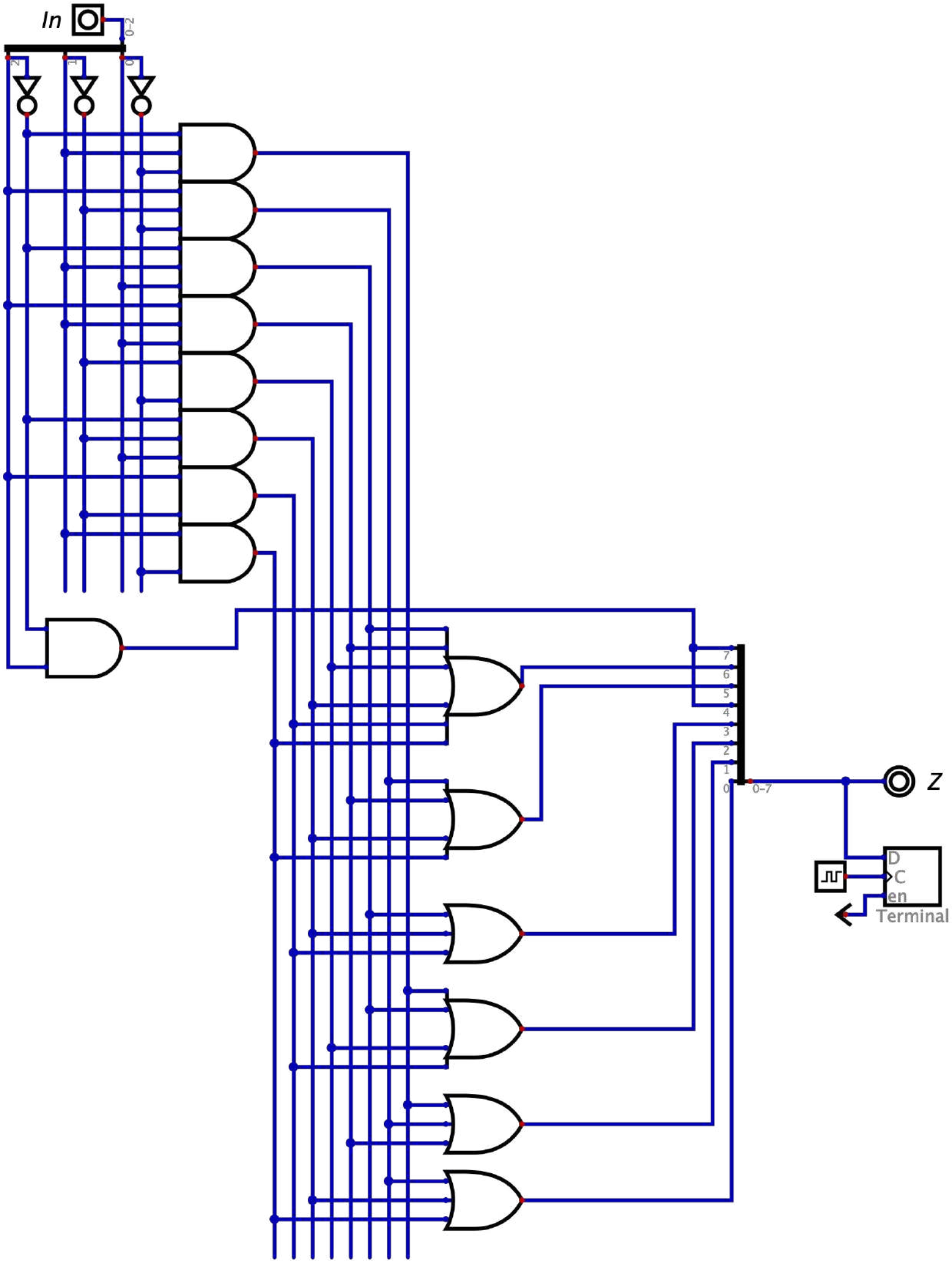
- In ขนาด 3 Bit

ข้อมูลส่งออก

- Z ขนาด 8 Bit

ชุดข้อมูลทดสอบ

- 12.5% In มีค่าเป็น 0
- 12.5% In มีค่าเป็น 1
- 12.5% In มีค่าเป็น 2
- 12.5% In มีค่าเป็น 3
- 12.5% In มีค่าเป็น 4
- 12.5% In มีค่าเป็น 5
- 12.5% In มีค่าเป็น 6
- 12.5% In มีค่าเป็น 7


In
0-2
2
1
0
7
6
5
4
3
2
1
0
0-7
Z
D
C
en
Terminal

Binary Encoder

ให้นิสิตสร้างวงจร Binary Encoder ที่มี Input คือ In ขนาด 4 Bit, Selector ขนาด 2 Bit และ Output คือ Output ขนาด 4 บิท โดยให้สร้างวงจรที่นำค่า A (มีค่าในช่วง 0 ถึง 9) ไปเข้ารหัสต่างๆตามค่า Selector โดยสามารถดูได้จากตารางด้านล่าง และส่งค่าที่เข้ารหัสไปแล้วออกมาที่ Output

ค่า **Selector**	Output
00	Excess-3
01	Cyclic
10	2 4 2 1 code
11	6 4 2 -3 code

รูปที่ 1 ค่า S และการเข้ารหัส

Decimal	Binary	Excess-3	Cyclic	2 4 2 1 code	6 4 2 -3 code
0	0 0 0 0	0 0 1 1	0 0 0 0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 1 0 0	0 0 0 1	0 0 0 1	0 1 0 1
2	0 0 1 0	0 1 0 1	0 0 1 1	0 0 1 0	0 0 1 0
3	0 0 1 1	0 1 1 0	0 0 1 0	0 0 1 1	1 0 0 1
4	0 1 0 0	0 1 1 1	0 1 1 0	0 1 0 0	0 1 0 0
5	0 1 0 1	1 0 0 0	0 1 1 1	1 0 1 1	1 0 1 1
6	0 1 1 0	1 0 0 1	0 1 0 1	1 1 0 0	0 1 1 0
7	0 1 1 1	1 0 1 0	0 1 0 0	1 1 0 1	1 1 0 1
8	1 0 0 0	1 0 1 1	1 1 0 0	1 1 1 0	1 0 1 0
9	1 0 0 1	1 1 0 0	1 1 0 1	1 1 1 1	1 1 1 1

รูปที่ 2 การเข้ารหัสแบบต่างๆ

ข้อมูลนำเข้า

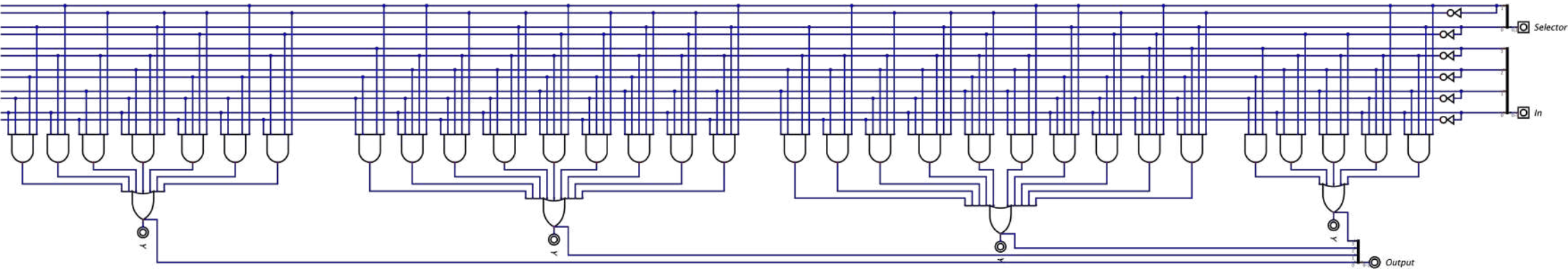
- In ขนาด 4 Bit
- Selector ขนาด 2 Bit

ข้อมูลส่งออก

- Output ขนาด 4 Bit

ชุดข้อมูลทดสอบ

- 25% Selector มีค่าเป็น 00
- 25% Selector มีค่าเป็น 01
- 25% Selector มีค่าเป็น 10
- 25% Selector มีค่าเป็น 11


Selector
In
Output

Hamming Code

ให้นิสิตสร้างวงจร Hamming Code ที่มี Input คือ In ขนาด 7 Bit และ Output คือ Output ขนาด 7 บิท โดยวงจรจะทำหน้าที่ตรวจสอบและแก้ไข Hamming code ที่เข้ามาทาง In (ซึ่งให้ถือว่าผิดพลาดได้ไม่เกิน 1 บิต) และให้หา Hamming Code ที่ถูกต้องออกมาทาง Output

Decimal digit	Position	1 p_1	2 p_2	3 m_1	4 p_3	5 m_2	6 m_3	7 m_4
0		0	0	0	0	0	0	0
1		1	1	0	1	0	0	1
2		0	1	0	1	0	1	0
3		1	0	0	0	0	1	1
4		1	0	0	1	1	0	0
5		0	1	0	0	1	0	1
6		1	1	0	0	1	1	0
7		0	0	0	1	1	1	1
8		1	1	1	0	0	0	0
9		0	0	1	1	0	0	1

ตารางที่ 1 Hamming Code for BCD

ข้อมูลนำเข้า

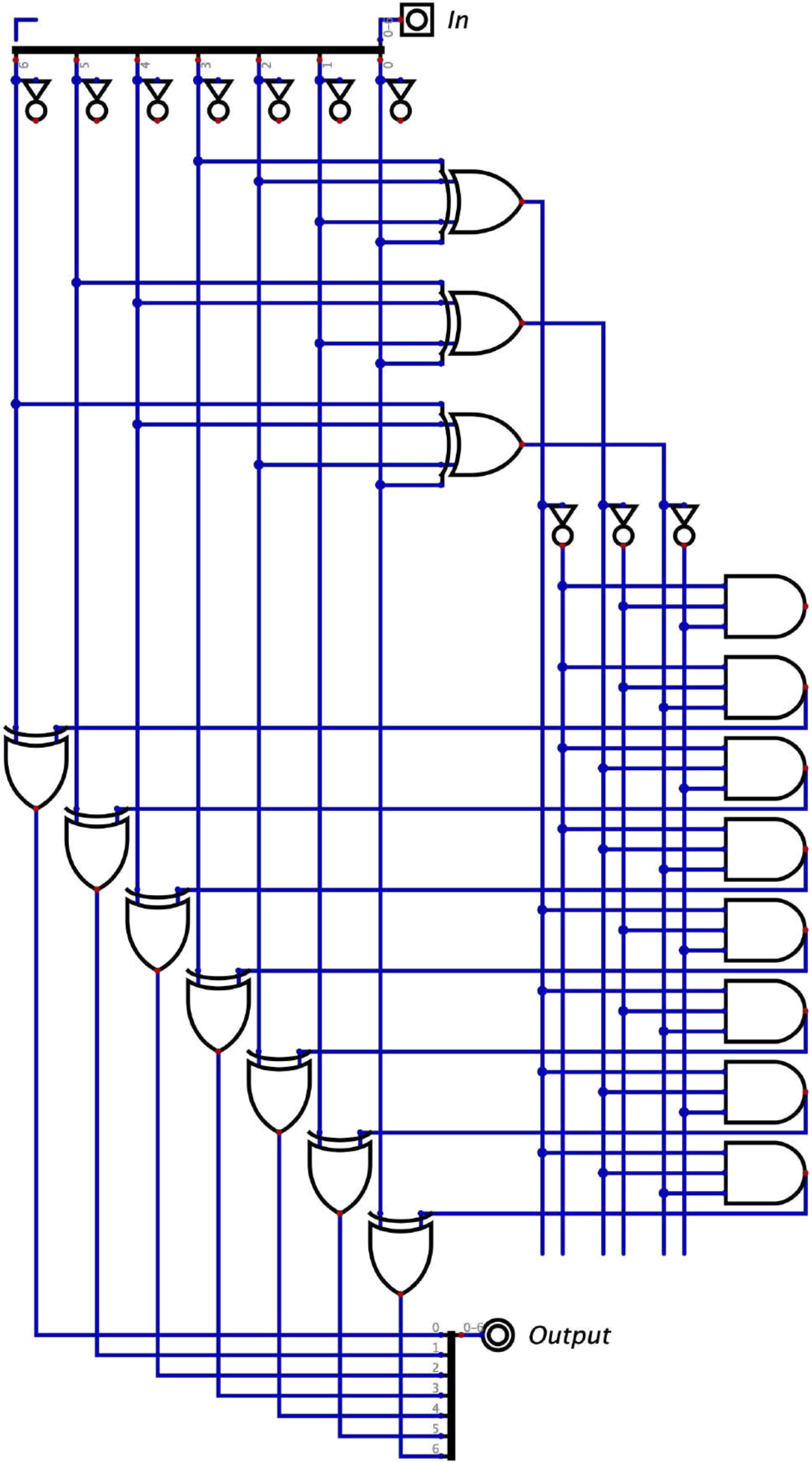
- In ขนาด 7 Bit

ข้อมูลส่งออก

- Output ขนาด 7 Bit

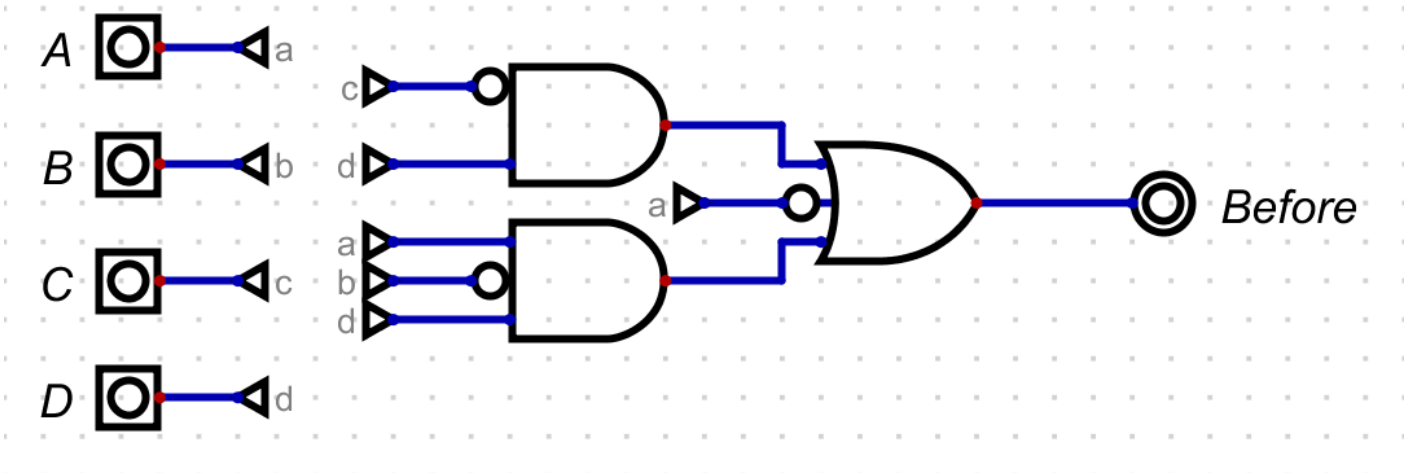
ชุดข้อมูลทดสอบ

- 10% In เป็น Hamming code ที่มีค่า 0 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 1 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 2 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 3 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 4 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 5 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 6 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 7 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 8 (จะผิดพลาดไม่เกิน 1 bit)
- 10% In เป็น Hamming code ที่มีค่า 9 (จะผิดพลาดไม่เกิน 1 bit)


In
0-6
6
5
4
3
2
1
0
0
0-6
Output
0
1
2
3
4
5
6

Glitch_I

จากสมการบูลลีน F(A,B,C,D) = A’ + C’D + AB’D จงสร้างวงจรก่อนและหลังกำจัด Hazard แล้วเขียน test vector ที่แสดง Glitch เปรียบเทียบให้เห็นชัดเจนว่าวงจรก่อนกำจัด hazard มี glitch ส่วนวงจรหลังกำจัดไม่มี ให้แสดง K-Map ให้ผู้ตรวจดูด้วย ในวงจรประกอบด้วย Input คือ A,B,C,D ขนาด 1 Bit และ Output คือ Before,After ขนาด 1 Bit โดยที่ Before คือ Output ของวงจรก่อนการแก้ไข Glitch (สร้างวงจรตามสมการบูลลีนที่กำหนดให้) และ After คือ Output ของวงจรหลังการแก้ไข Glitch แล้ว



รูปที่ 1. วงจรก่อนแก้ glitch

ข้อมูลนำเข้า

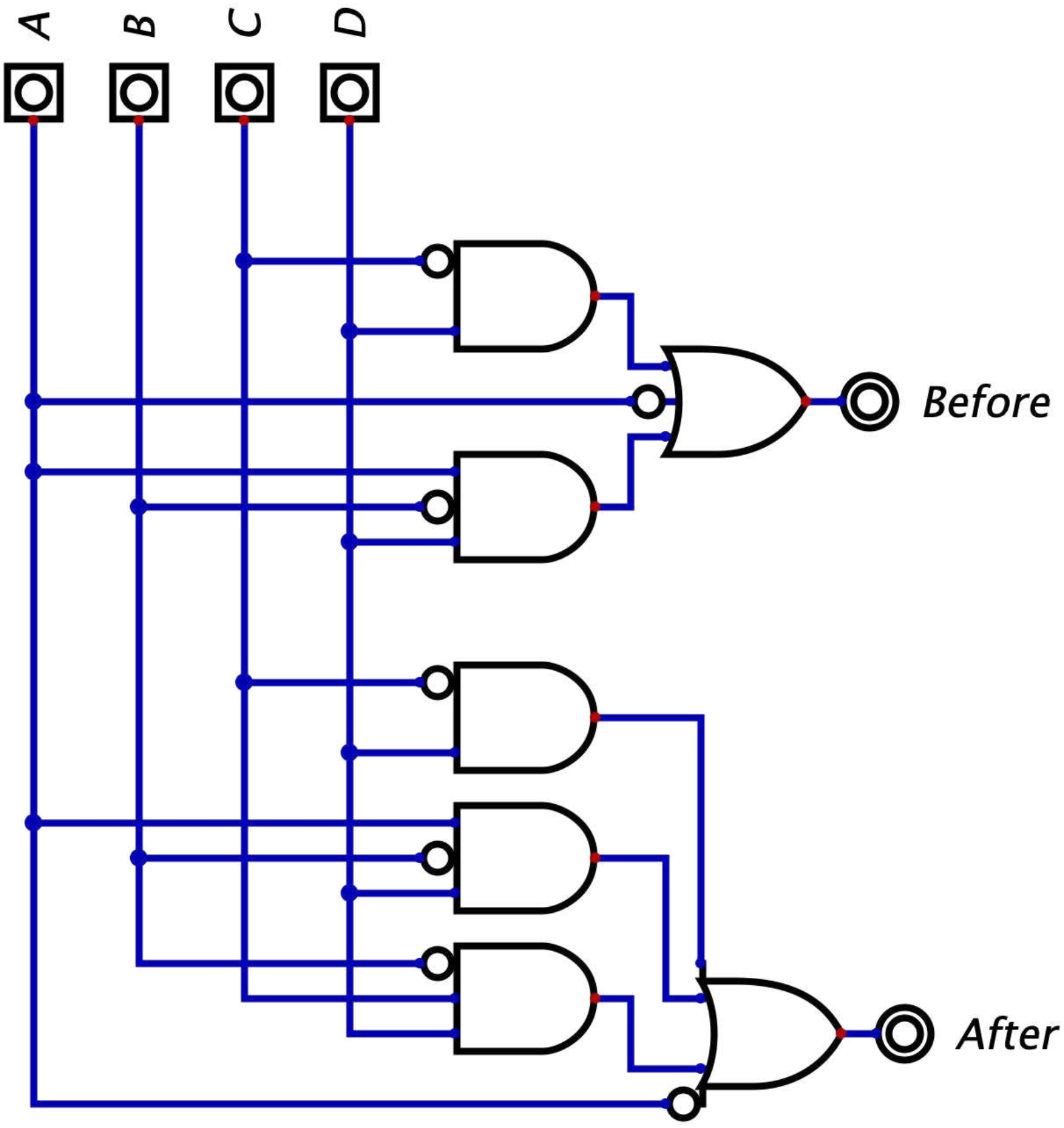
- A ขนาด 1 Bit
- B ขนาด 1 Bit
- C ขนาด 1 Bit
- D ขนาด 1 Bit

ข้อมูลส่งออก

- Before ขนาด 1 Bit
- After ขนาด 1 Bit

ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานถูกต้องตาม Input ทุกรูปแบบและไม่มี Glitch


A
B
C
D
Before
After

Glitch_II

จากสมการบูลลีน F(A,B,C,D,E,F) = Σm(0,1,3,4,7,11,12,15,16,17,20,28) จงสร้างวงจรก่อนและหลังกำจัด Hazard แล้วเขียน test vector ที่แสดง Glitch เปรียบเทียบให้เห็นชัดเจนว่าวงจรก่อนกำจัด hazard มี glitch ส่วนวงจรหลังกำจัดไม่มี ให้แสดง K-Map ให้ผู้ตรวจดูด้วย ในวงจรประกอบด้วย Input คือ A,B,C,D,E,F ขนาด 1 Bit และ Output คือ Before,After ขนาด 1 Bit โดยที่ Before คือ Output ของวงจรก่อนการแก้ไข Glitch (สร้างวงจรตามสมการบูลลีนที่กำหนดให้) และ After คือ Output ของวงจรหลังการแก้ไข Glitch แล้ว

ข้อมูลนำเข้า

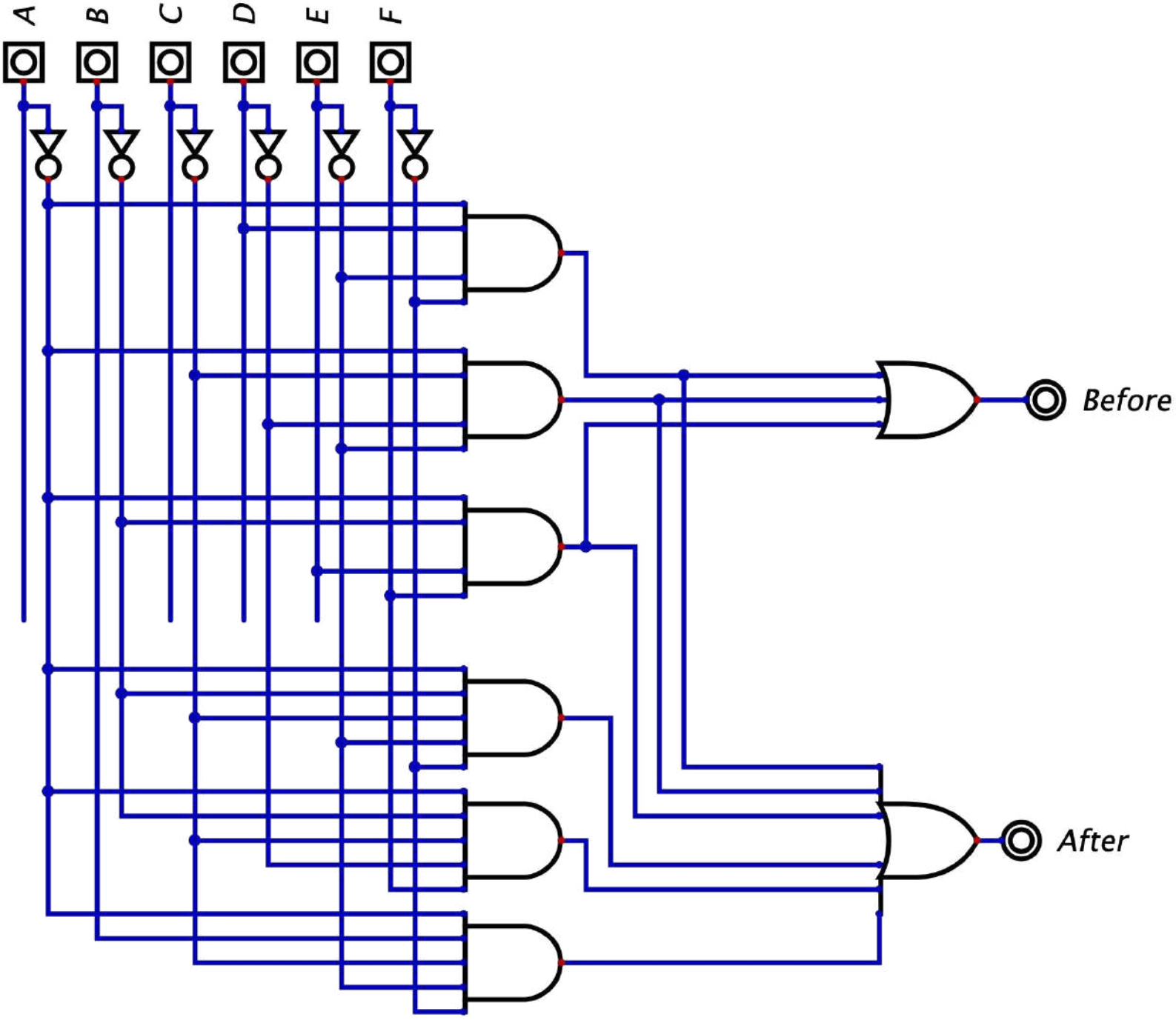
- A ขนาด 1 Bit
- B ขนาด 1 Bit
- C ขนาด 1 Bit
- D ขนาด 1 Bit
- E ขนาด 1 Bit
- F ขนาด 1 Bit

ข้อมูลส่งออก

- Before ขนาด 1 Bit
- After ขนาด 1 Bit

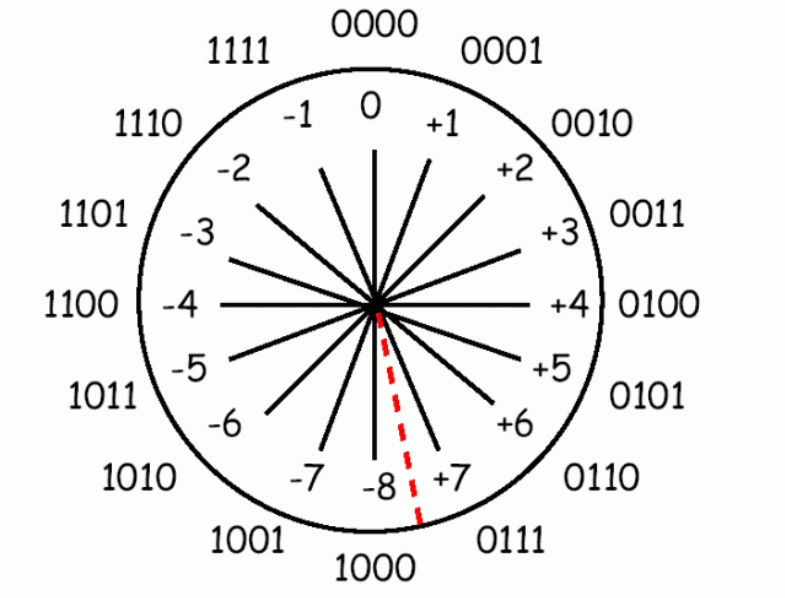
ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานถูกต้องตาม Input ทุกรูปแบบและไม่มี Glitch


A
B
C
D
E
F
Before
After

2’s Complement Number

ให้นิสิตสร้างวงจร 2’s Complement Number ที่มี Input คือ In ขนาด 8 Bit และ Output คือ Output ขนาด 9 Bit โดยให้แปลง In ซึ่งเป็นจำนวนเต็มบวกในระบบเลขฐาน 2 ขนาด 8 Bit ไปเป็นจำนวนเต็มลบที่มีค่าเป็น -In ในระบบ 2’s Complement (ถ้า In มีค่าเป็น 10 แล้ววงจรต้องแปลงไปเป็นเลขที่มีค่า -10) แล้วส่งออกไปที่ Output



รูปที่ 1. เลขในระบบ 2’s Complement ขนาด 4 bit

ข้อมูลนำเข้า

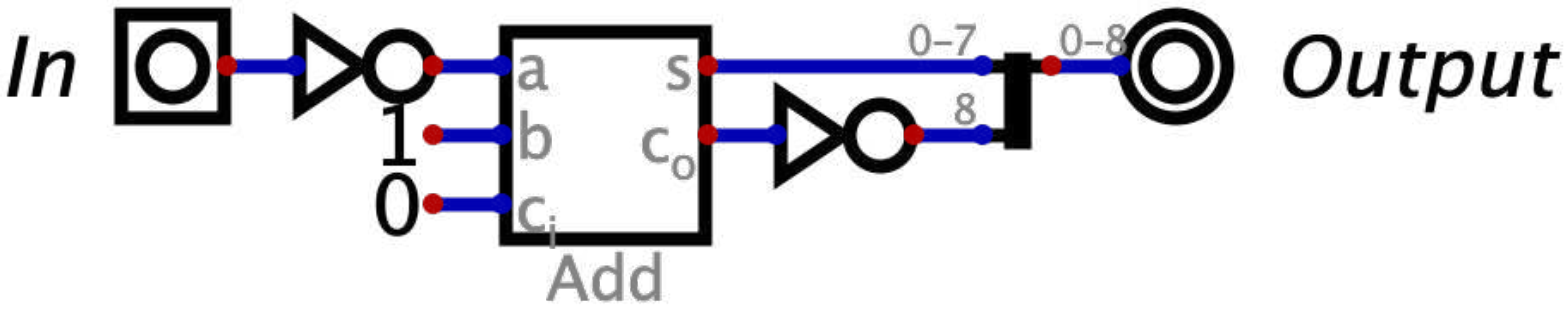
- In ขนาด 8 Bit

ข้อมูลส่งออก

- Output ขนาด 9 Bit

ชุดข้อมูลทดสอบ

- 100% โปรแกรมทำงานถูกต้องตาม Input ทุกรูปแบบ


In
1
0
a
b
c_i
s
c_o
Add
0–7
8
0–8
Output

Conversion (2’s Complement)

ให้นิสิตสร้างวงจร Conversion ที่มี Input คือ In ขนาด 4 Bit และ Output คือ Output ขนาด 4 Bit โดยให้แปลง In ซึ่งเป็นจำนวนเต็มในระบบ Sign and magnitude ไปเป็นจำนวนเต็มในระบบ 2’s Complement แล้วส่งออกไปที่ Output

Decimal	Signed Magnitude	Signed One's Complement	Signed Two's Complement
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001

ตารางที่ 1. ตารางแสดงเลข 4 Bit ในระบบต่างๆ

คำแนะนำ : สร้างวงจรโดยใช้ความรู้ในการแปลงเลข 2’s Complement แทนการใช้ K-map

ข้อมูลนำเข้า

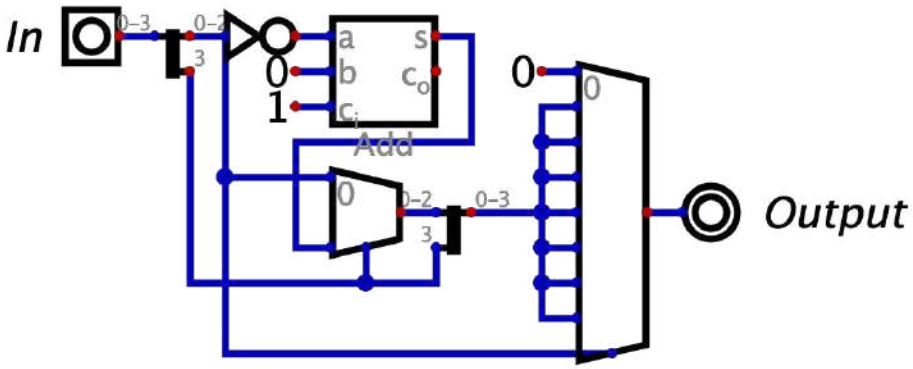
- In ขนาด 4 Bit

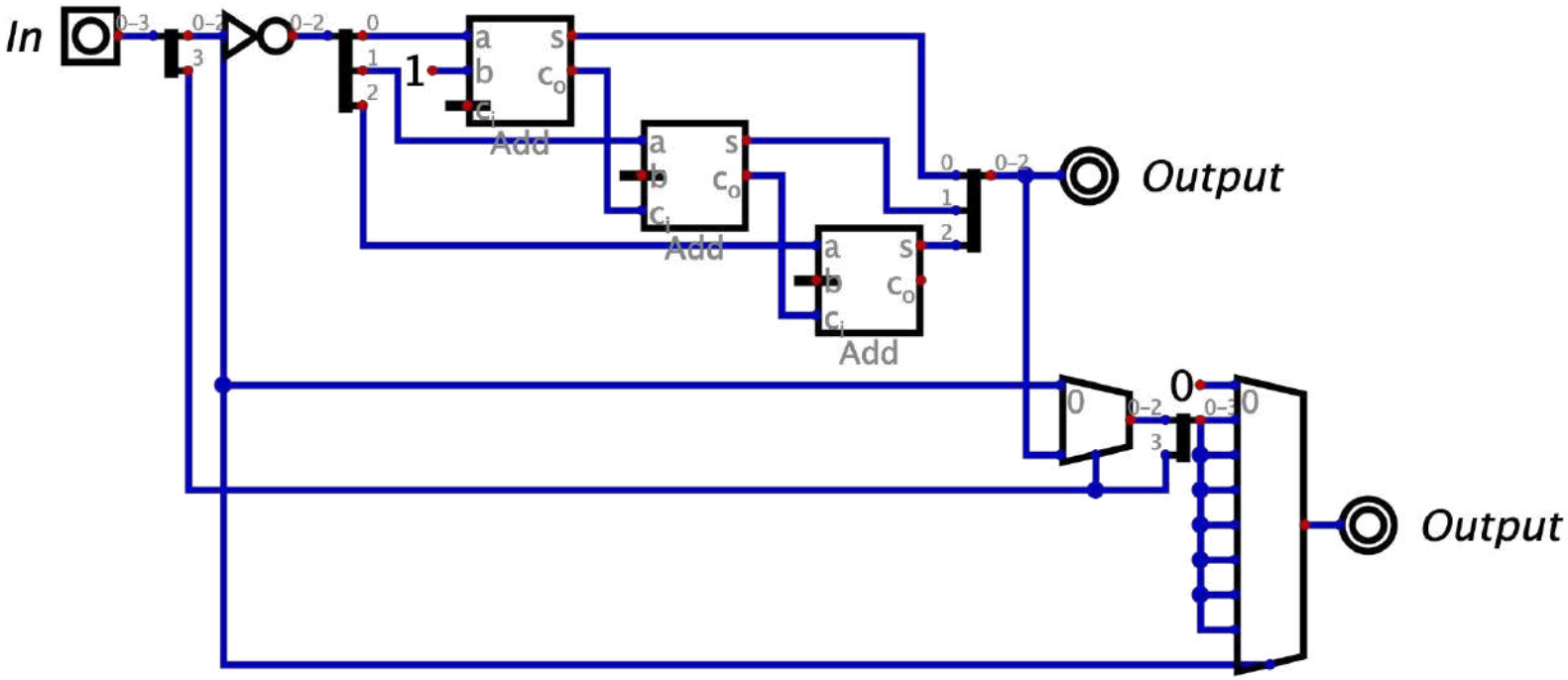
ข้อมูลส่งออก

- Output ขนาด 4 Bit

ชุดข้อมูลทดสอบ

- 50% In เป็นจำนวนเต็มบวก และจำนวนเต็ม 0
- 50% In เป็นจำนวนเต็มลบ


In
0–3
0–2
3
0
1
a
b
c_i
s
c_o
Add
0
0–2
3
0–3
0
0
Output


In
0–3
0–2
3
0–2
0
1
2
1
a
b
c_i
s
c_o
Add
Add
Add
0
1
2
0–2
Output
0
0
0–2
3
0–3
0
Output

ALU

ให้นิสิตสร้างวงจร ALU ที่มี Input เป็นเลข Binary แบบ 2's complement 4 bit 2 จำนวน คือ A และ B, มี Function Selector ขนาด 3 bit คือ S, Mode selector ขนาด 1 bit คือ M และ Carry in ขนาด 1 bit คือ C. มี Output คือ F ขนาด 4 bit และ Cout ขนาด 1 bit โดยวงจรจะมีการทำงานดังนี้

S	M=0 (Logical Functions)	M=1 (Arithmetic functions)	
		C=0	C=1
000	Not A	A	A+1
001	A nand B	A-1	A
010	A nor B	A+B	A+B+1
011	A xnor B	A-B-1	A-B
100	A and B	-A-1	-A
101	A or B	B	B+1
110	A and (Not B)	Shift A left one bit	Not used
111	A	Shift A right one bit	Not used

ตารางที่ 1. ตารางแสดงการทำงานของวงจร

ในการ Shift A left one bit ให้ทิ้ง bit ซ้ายสุดของ A ไปและให้ bit ขวาสุดของ A เป็น 0

ในการ Shift A right one bit ให้ทิ้ง bit ขวาสุดของ A ไปและให้ bit ซ้ายสุดของ A เป็น 0

สุดท้ายให้ส่งผลลัพธ์ของการทำงานของวงจรออกไปที่ F และในส่วน Cout นิสิตสามารถ implement แบบใดก็ได้

ข้อแนะนำ : เริ่มสร้างจาก ALU ขนาด 1 bit ก่อนแล้วจึงค่อยนำ ALU 1 bit มาสร้าง ALU 4 bit

ข้อห้าม : ห้ามใช้ Subtractor และ Shifter

ข้อมูลนำเข้า

- A ขนาด 4 Bit
- B ขนาด 4 Bit
- S ขนาด 3 Bit
- M ขนาด 1 Bit
- C ขนาด 1 Bit

ข้อมูลส่งออก

- F ขนาด 4 Bit
- Cout ขนาด 1 Bit

ชุดข้อมูลทดสอบ

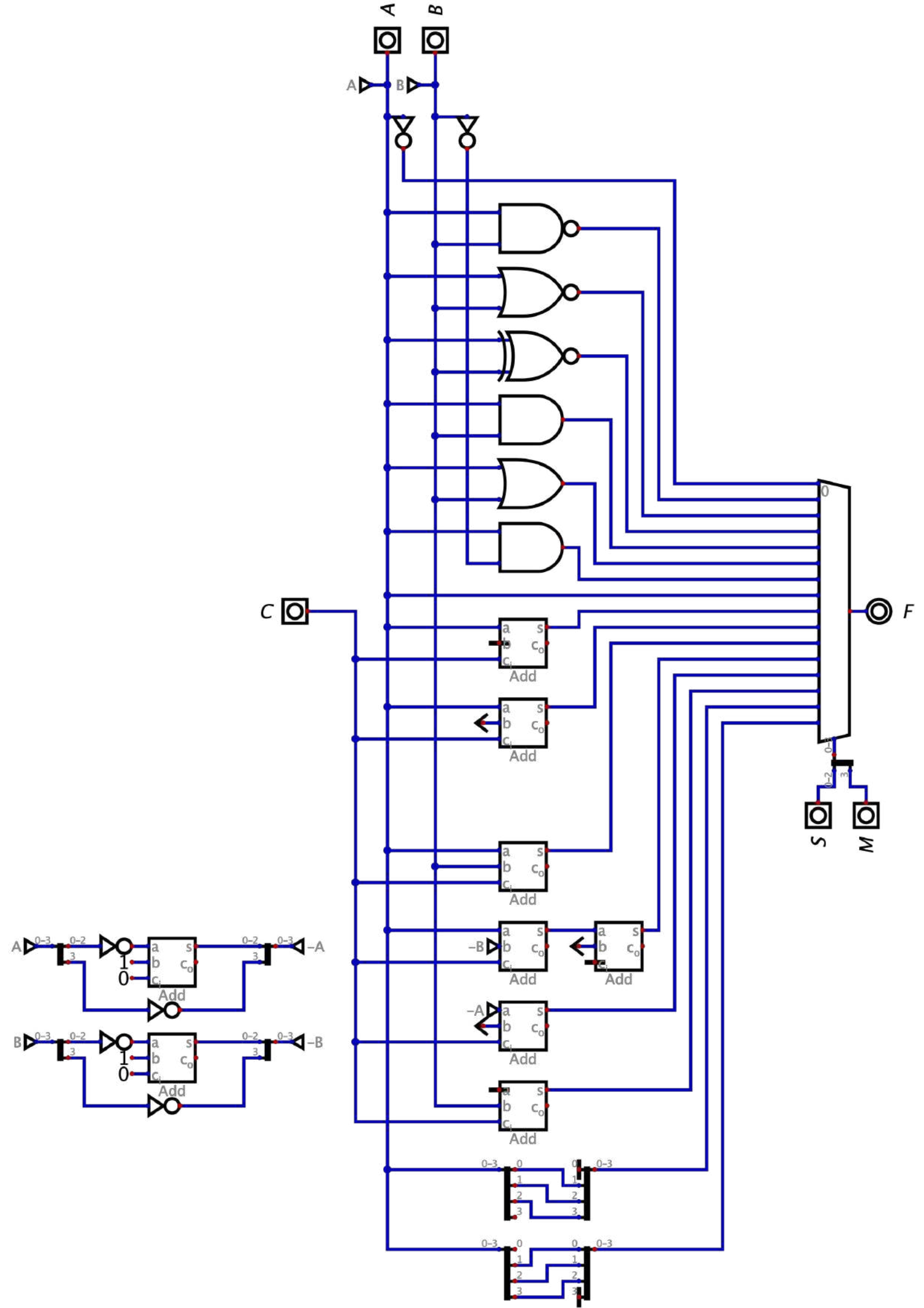
- 50% M มีค่าเป็น 0
  - 6.25% S มีค่าเป็น 000
  - 6.25% S มีค่าเป็น 001
  - 6.25% S มีค่าเป็น 010
  - 6.25% S มีค่าเป็น 011
  - 6.25% S มีค่าเป็น 100

- 6.25% S มีค่าเป็น 101
- 6.25% S มีค่าเป็น 110

ชุดข้อมูลทดสอบ (ต่อ)

- 6.25% S มีค่าเป็น 111

- 50% M มีค่าเป็น 1
  - 6.25% S มีค่าเป็น 000
  - 6.25% S มีค่าเป็น 001
  - 6.25% S มีค่าเป็น 010
  - 6.25% S มีค่าเป็น 011
  - 6.25% S มีค่าเป็น 100
  - 6.25% S มีค่าเป็น 101
  - 6.25% S มีค่าเป็น 110
  - 6.25% S มีค่าเป็น 111


A
B
C
F
S
M
Add
-A
-B