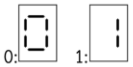
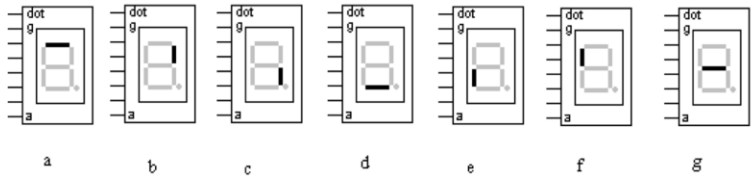
ข้อสอบปี 64

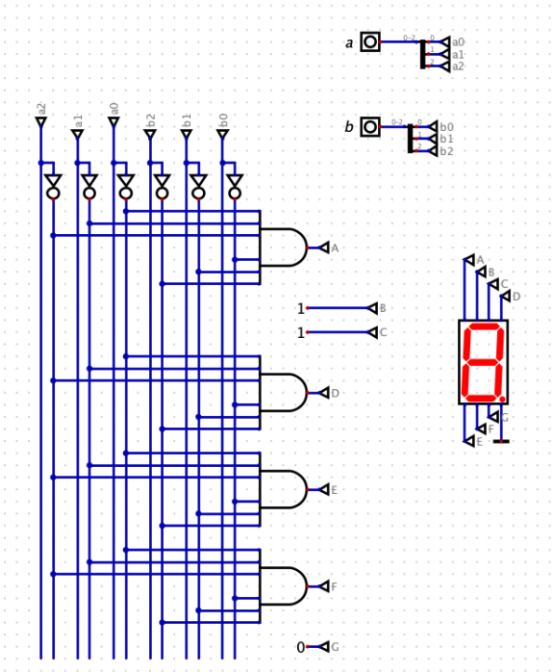
1. สร้างวงจรรวมเลข 3 บิต 2 ตัวโดยให้แสดงผลด้วย 7-Seg Disp เป็น 1 เมื่อ อินพุตของทั้งสองอันรวมกันแล้วมากกว่าตัวเลขตัวที่ 8 ของรหัสนิสิตของคุณ (เช่น 6XXXXXX521, คือเป็น 1 เมื่อผลบวกมากกว่า 5) และเป็น 0 ในกรณีอื่นๆ ระบุให้รับเลข Input 3 บิต 2 ตัวด้วย Binary Switch 6 ตัว และให้แสดงผลลัพธ์ด้วย 7-Seg Disp มีลักษณะดังนี้



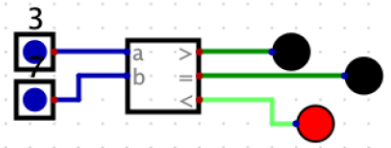
***** อนุญาตให้ใช้อุปกรณ์อะไรก็ได้ (PLA, Expresso, etc.) รวมถึง BinaryTo7Segment ******



ตรวจไปแล้ว □□□ครั้ง

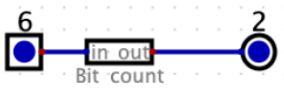


Useful Function



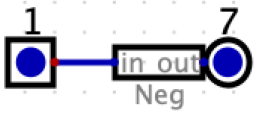
Comparator

เพื่อเปรียบเทียบจำนวนสองจำนวน (กี่บิตก็ได้)



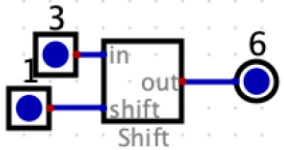
Bit Counter

นับเลข 1 ของแต่ละบิต



Negation

ส่งออกคำตอบเป็น $2^{bits} - input$



Barrel Shifter

เลื่อนบิตไปทางซ้าย และบิตที่เพิ่มมาทางขวาเป็น 0

Ex 011 -> 110 , 101 -> 010, 110 -> 100

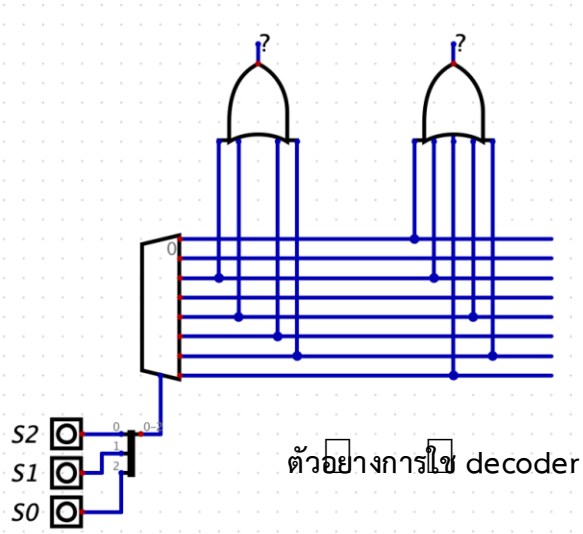
ข้อสอบปี 64

2. จงสร้างวงจรที่รับ Input จาก Binary Switch 3 อัน (S0, S1, S2) และ แสดงผลลัพธ์ O1 O2 ด้วย Binary Probe สองอัน ดังตารางต่อไปนี้ โดยใช้ 3:8 Decoder **(Decoder-8 Non Inv)**

S0 S1 S2	O1 O2
0 0 0	0 1
0 0 1	0 0
0 1 0	1 1
0 1 1	0 0
1 0 0	1 1
1 0 1	1 0
1 1 0	1 1
1 1 1	0 1

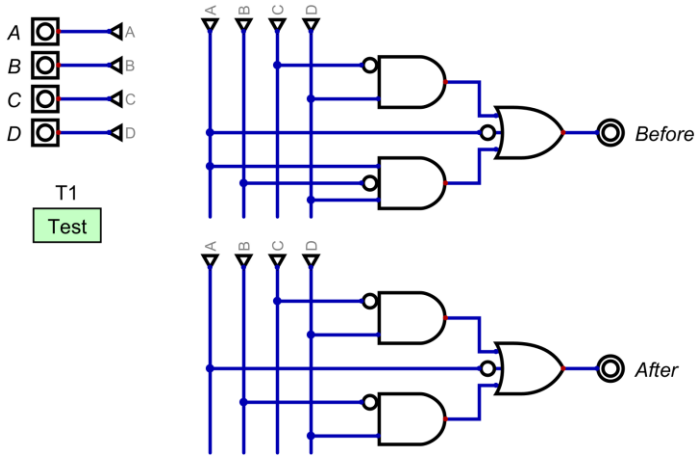
***** อนุญาตให้ใช้ Binary Switch, Binary Probe, Decoder-8 Non Inv และ Or gates เท่านั้นจะใช้กี่ตัวก็ได้******

ตรวจไปแล้ว □□□ครั้ง

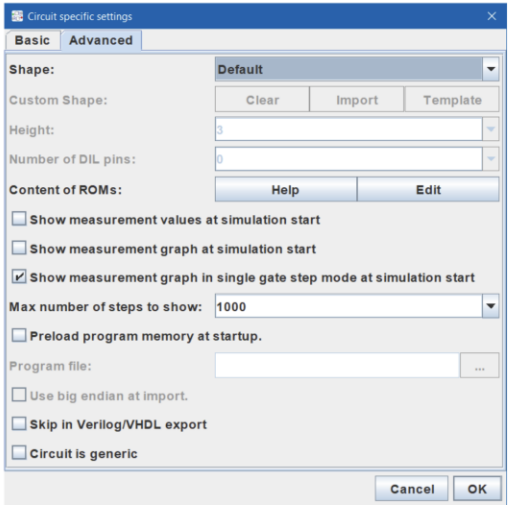


ตัวอย่างการใช้ decoder

แลป 5.1 (จริง ๆ ง่าย แต่เผื่อเค้าให้สร้าง testcase)



ก่อนอื่นให้ไปตั้งค่าที่ `Edit` > `Circuit specific settings` > `Advanced` > `Show measurement graph in single gate step mode at simulation start`



ตอนเปิด glitch ก็เปิดพร้อม double click ที่ input

จากนั้นสร้าง Testcases โดยการกด `Components` > `Misc.` > `Test case` แล้วก็วางที่ไหนก็ได้ของวงจรของเรา

เมื่อวางเสร็จแล้วให้ทำการกดคลิกขวาที่ `Test case` แล้วกด `Edit` จากนั้นก็กรอก testcases ที่เราต้องการจะตรวจสอบลงไป ดังนี้

```
A B C D Before
0 0 1 1 1
1 0 1 1 1
1 0 1 1 1
```

ข้อสอบปี 65

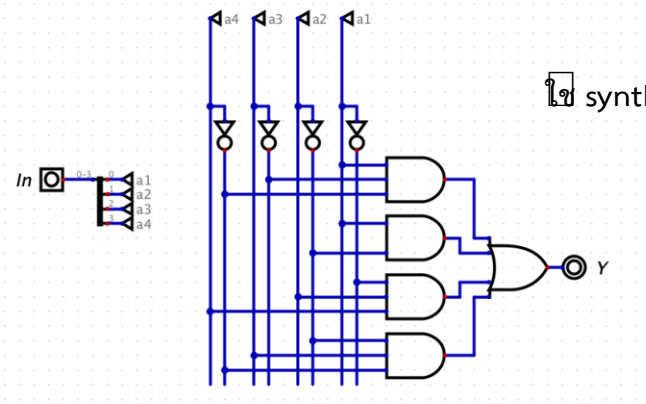
1. จงสร้างวงจรที่รับข้อมูลจาก Input (**in**) 4 Bit และ แสดงผลลัพธ์ด้วย Output (**out**) 1 Bit ดังตารางต่อไปนี้

in	out
0 0 0 0	0
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	0
0 1 1 1	0
1 0 0 0	0
1 0 0 1	1
1 0 1 0	1
1 0 1 1	0
1 1 0 0	0
1 1 0 1	1
1 1 1 0	1
1 1 1 1	0

กรุณาเริ่มจาก template_01.dig: ใน Template จะมี input ชื่อ in เป็นเลข 4 bit, และ output ชื่อ out เป็นเลข 1 bit. ในตัวอย่างหาก in มีค่าเป็น 1110 => out = 1

คะแนน

คะแนนเต็ม 100 คะแนน โดยมีจาก Grader 90 คะแนน และ ถ้าถูกต้องทุก Case ภายใน 1 ช.ม. จะได้อีก 10 คะแนน

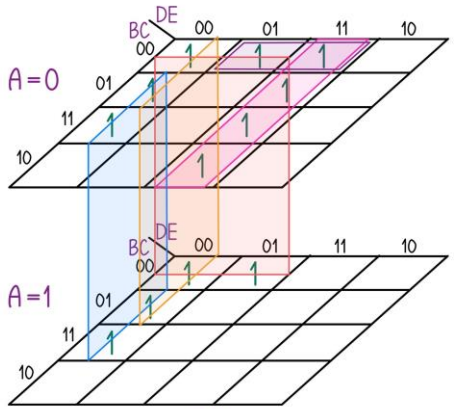


ใช้ synthesis แปปเดียวจบ

$$Before(A,B,C,D,E) = \Sigma m(0,1,3,4,7,11,12,15,16,17,20,28)$$

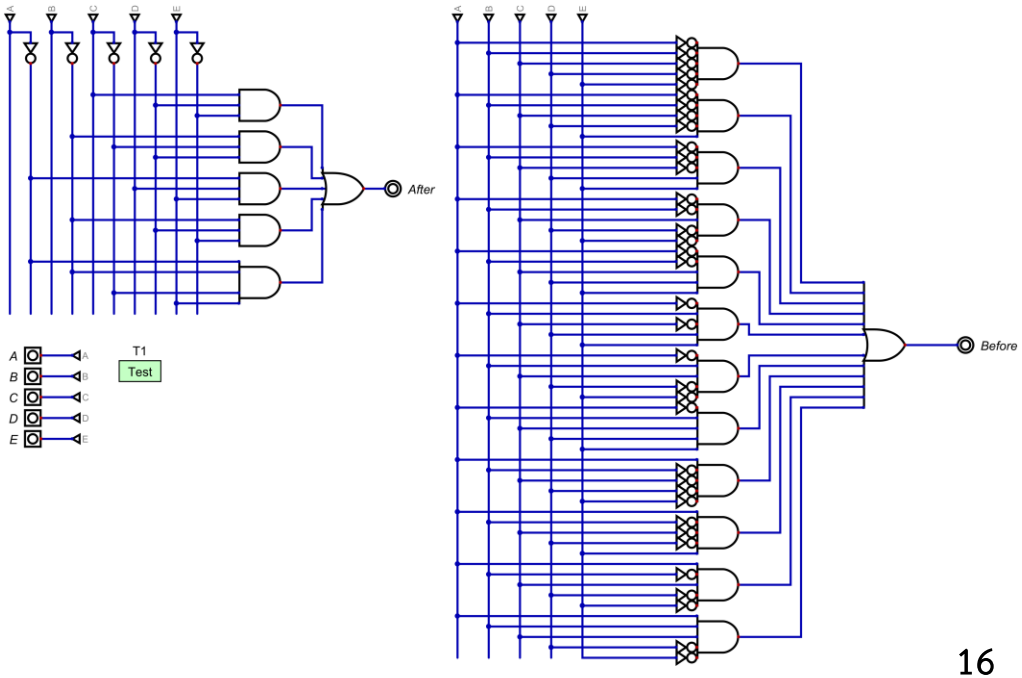
แลป 5.2 (จริง ๆ ง่าย แต่เผื่อเค้าให้สร้าง testcase)

5- variables K-map Example



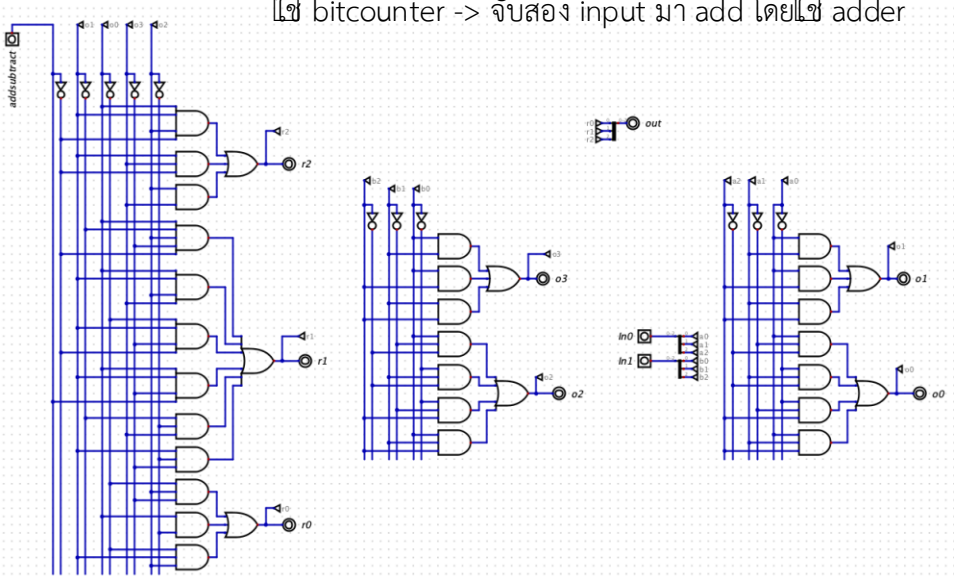
After = $CD'E' + B'C'D' + A'DE + B'D'E' + A'B'C'E$

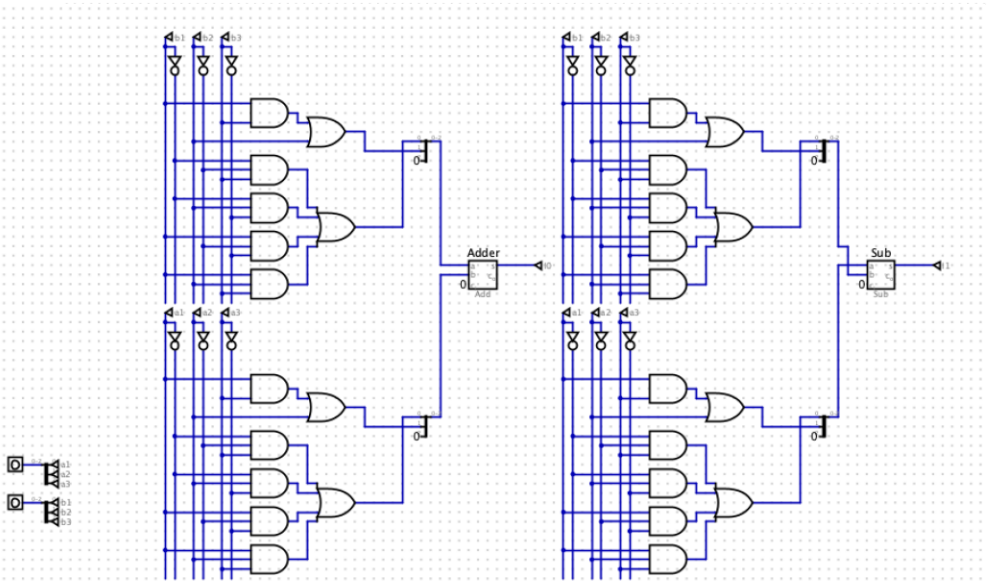
A	B	C	D	E	Before	After
1	1	1	0	0	1	1
1	0	1	0	0	1	1
1	0	0	0	0	1	1
1	0	0	0	1	1	1
0	1	1	0	0	1	1
0	0	1	0	0	1	1
0	0	0	0	0	1	1
0	0	0	0	1	1	1
0	0	0	1	1	1	1
0	0	1	1	1	1	1
0	1	1	1	1	1	1
0	1	0	1	1	1	1

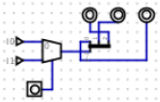


ข้อสอบปี 65 (ข้อสอง) นับ bit count แต่จริง ๆ ใช้ features Bitcounter จะง่ายกว่ามาก

ใช้ bitcounter -> จับสอง input มา add โดยใช้ adder





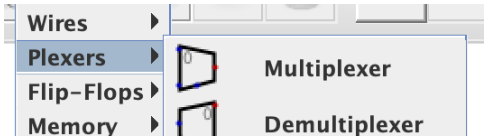


สามารถใช้ MUX 2:1 ได้ เอามาเป็นตัวอย่าง ปล. บนกับล่าง วงจรได้คำตอบเหมือนกัน แต่ทำกันคนละแบบ

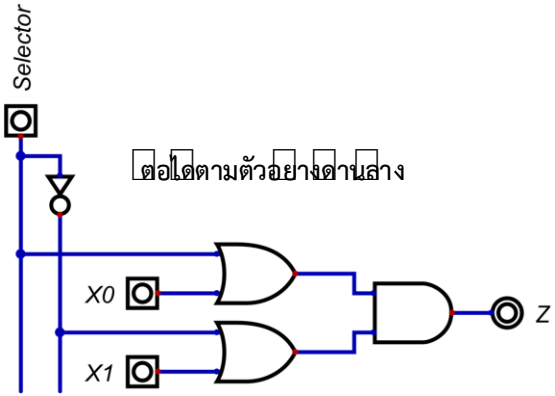
แลป 2.8 (สร้าง multiplexer แบบ POS)

INPUT			OUTPUT
X0	X1	Selection	Z
0	0	0	0
0	1		0
1	0		1
1	1		1
0	0	1	0
0	1		1
1	0		0
1	1		1

จริง ๆ ใช้ multiplexer ในโปรแกรมมาจะง่ายกว่าและซับซ้อนน้อยกว่า



สร้าง K-map ออกมาจาก truth-table ด้านบน จะได้ Z = (S + X0)(S' + X1)



ต่อได้ตามตัวอย่างด้านล่าง

LAB3.4 Hamming Code

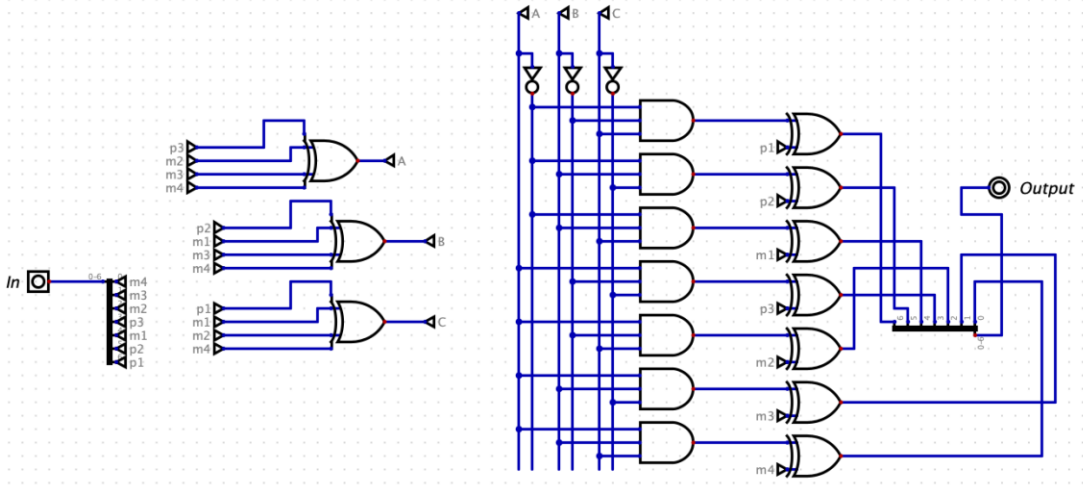
Hamming Code							
Bits	1	2	3	4	5	6	7
Variable	P1	P2	M1	P3	M2	M3	M4

Circuit A	
OUTPUT	EQUATION
C1	XOR(P3, M2, M3, M4)
C2	XOR(P2, M1, M3, M4)
C3	XOR(P1, M1, M2, M4)

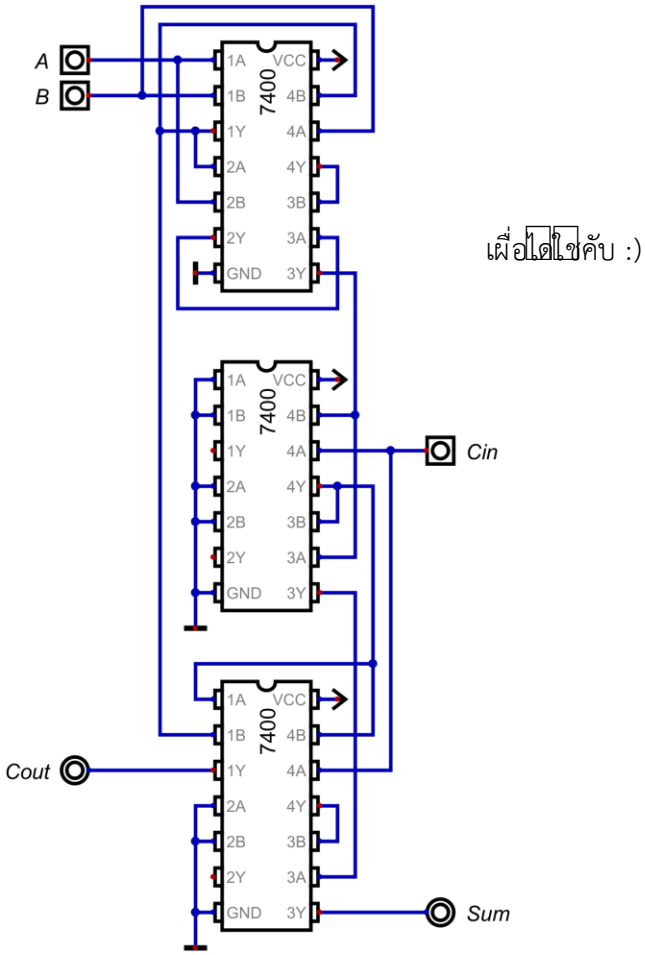
Circuit B										
INPUT				OUTPUT						
C	C1	C2	C3	B1	B2	B3	B4	B5	B6	B7
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0
2	0	1	0	0	1	0	0	0	0	0
3	0	1	1	0	0	1	0	0	0	0
4	1	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	1

C1C2C3 ใช้บอกตำแหน่งบิตที่ผิด และต้องทำการกลับบิตนั้น สามารถกลับโดยเอาบิตที่ผิดไป XOR ด้วย 1

ตัวอย่าง ในแลป 3.4 เรื่องการกลับบิต ของบิตที่ผิด



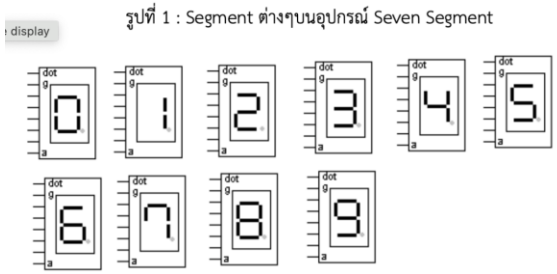
LAB 4.2 ต่อ full adder (ต่อจากหน้า 9)



เผื่อได้ใช้คับ :)

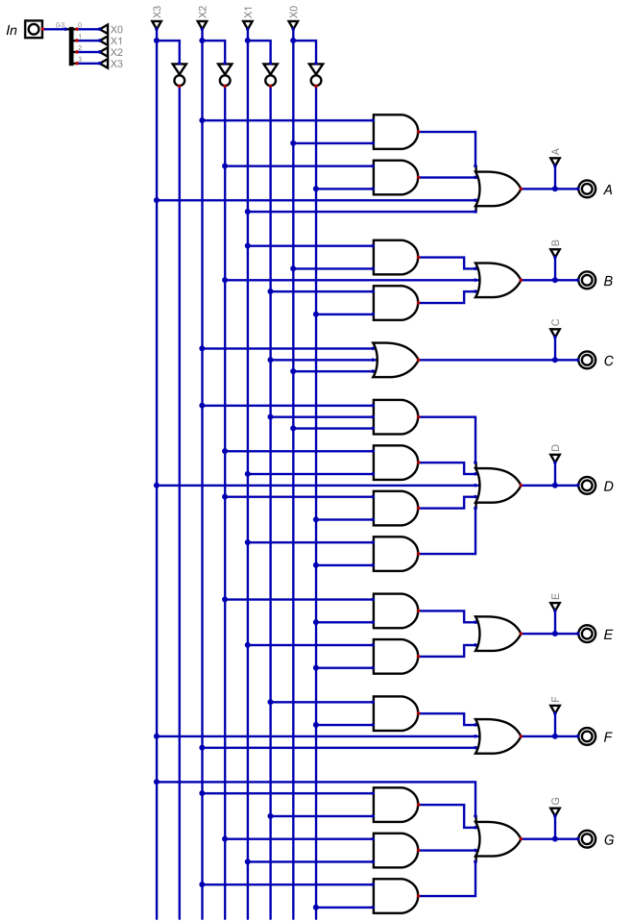
LAB 3.1 Seven Segment Display

รูปที่ 1 : Segment ต่างๆบนอุปกรณ์ Seven Segment

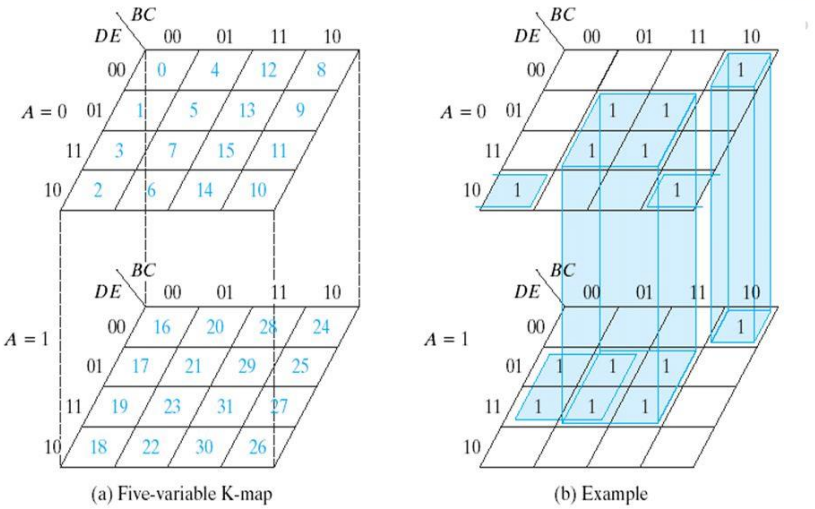


รูปที่ 2 : การแสดงผลของอุปกรณ์ Seven Segment

INPUT					OUTPUT						
In	X3	X2	X1	X0	A	B	C	D	E	F	G
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	1	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	1	0	1	0	DON'T CARE						
11	1	0	1	1							
12	1	1	0	0							
13	1	1	0	1							
				0							
				1							

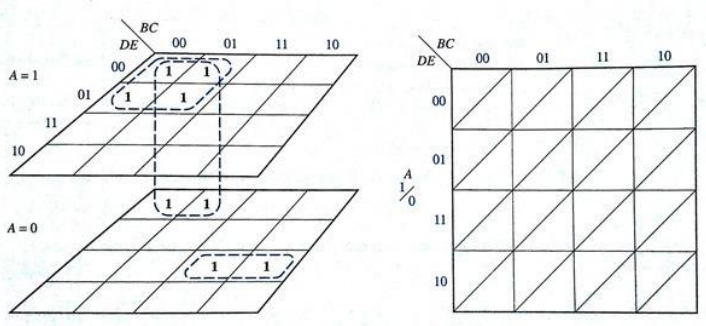


ตัวอย่าง 5 Variables K-map เผื่อได้ใช้



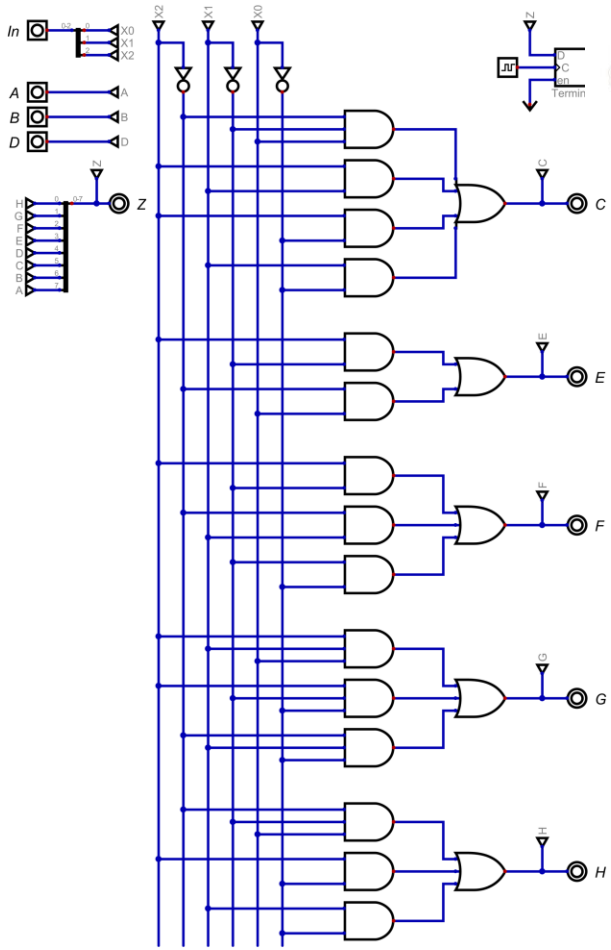
(a) Five-variable K-map (b) Example

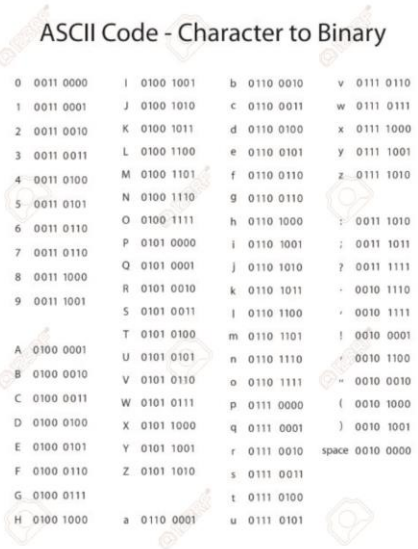
Five-variable K-map and example.



LAB 3.2 ASCII

INPUT				OUTPUT									
In	X2	X1	X0	Char	Hex	A	B	C	D	E	F	G	H
0	0	0	0	D	44	0	1	0	0	0	1	0	0
1	0	0	1	i	69	0	1	1	0	1	0	0	1
2	0	1	0	g	67	0	1	1	0	0	1	1	1
3	0	1	1	L	4C	0	1	0	0	1	1	0	0
4	1	0	0	o	6F	0	1	1	0	1	1	1	1
5	1	0	1	L	4C	0	1	0	0	1	1	0	0
6	1	1	0	a	61	0	1	1	0	0	0	0	1
7	1	1	1	b	62	0	1	1	0	0	0	1	0





LAB 3.3 Encoder

Binary Encoder																			
INPUT					Excess-3				Cyclic				2 4 2 1 Code				6 4 2 -3 Code			
In	X3	X2	X1	X0	A3	A2	A1	A0	B3	B2	B1	B0	C3	C2	C1	C0	D3	D2	D1	D0
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
10	1	0	1	0	DON'T CARE				DON'T CARE				DON'T CARE				DON'T CARE			
11	1	0	1	1																
12	1	1	0	0																
13	1	1	0	1																
14	1	1	1	0																
15	1	1	1	1																

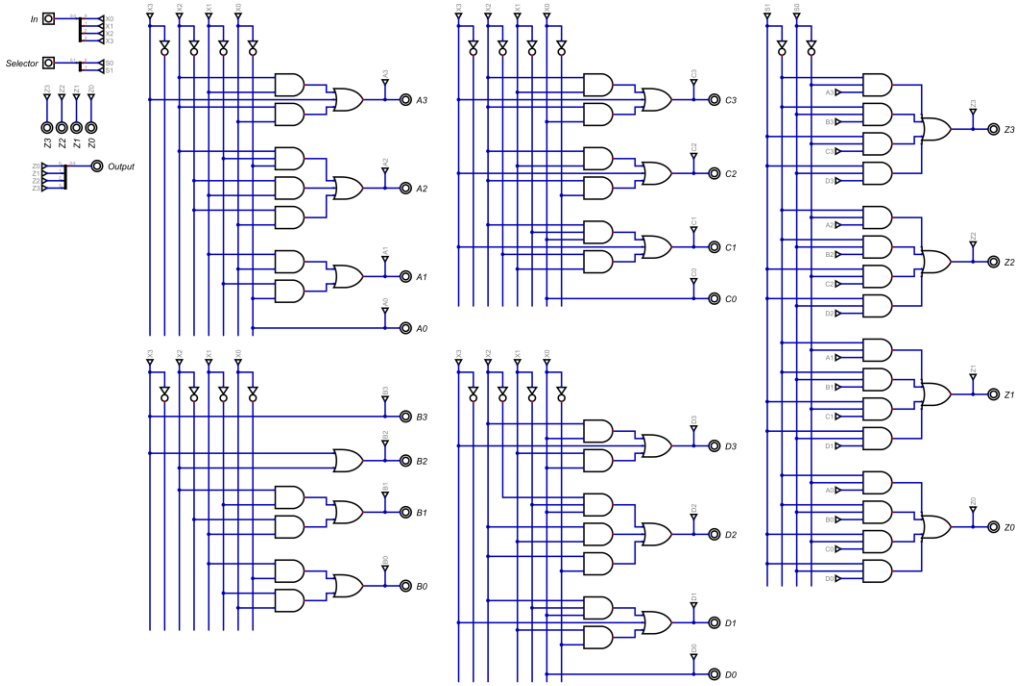
MUX-4-to-1 (A0 - D0)			
INPUT		OUTPUT	EQUATION
S1	S0	Z0	
0	0	A0	S1'S0'A0
0	1	B0	S1'S0B0
1	0	C0	S1S0'C0
1	1	D0	S1S0D0

MUX-4-to-1 (A1 - D1)			
INPUT		OUTPUT	EQUATION
S1	S0	Z1	
0	0	A1	S1'S0'A1
0	1	B1	S1'S0B1
1	0	C1	S1S0'C1
1	1	D1	S1S0D1

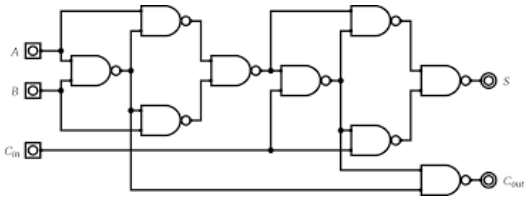
MUX-4-to-1 (A2 - D2)			
INPUT		OUTPUT	EQUATION
S1	S0	Z2	
0	0	A2	S1'S0'A2
0	1	B2	S1'S0B2
1	0	C2	S1S0'C2
1	1	D2	S1S0D2

MUX-4-to-1 (A3 - D3)			
INPUT		OUTPUT	EQUATION
S1	S0	Z3	
0	0	A3	S1'S0'A3
0	1	B3	S1'S0B3
1	0	C3	S1S0'C3
1	1	D3	S1S0D3

LAB 3.3 Encoder (ต่อ)



LAB 4.2 ต่อ full adder



Full Adder มี Input สามค่า A, B, Cin และมี Output 2 ค่าคือ Sum (ผลรวม) และ Cout (ตัวทด)

Input			Output	
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