วงจรคอมไบเนชัน

Using MSI ICs

ชื่อบทเรียน

2.1 หลักการทำงานของวงจรคอมไบเนชัน

2.2 การสร้างวงจรคอมไบเนชัน

จุดประสงค์การสอน สัปดาห์ที่ 3

2.1 เข้าใจหลักการทำงานของวงจรคอมไบเนชัน

- 2.1.1 อธิบายวงจรเข้ารหัส
- 2.1.2 อธิบายวงจรถอดรหัส
- 2.1.3 อธิบายวงจรแปลงรหัส
- 2.1.4 อธิบายวงจรเปรียบเทียบ
- 2.1.5 อธิบายวงจรมัลติเพล็กซ์
- 2.1.6 อธิบายวงจรดีมัลติเพล็กซ์

2.2 ออกแบบวงจรคอมไบเนชัน

- 2.2.1 ใช้ลอจิกเกตพื้นฐาน
- 2.2.2 ใช้ MSI ICs
- 2.2.3 ใช้ PLD

วงจรคอมไบเนชัน

- **วงจรคอมไบเนชั่น (Combinational Logic Circuit)**

เป็นวงจรที่นำเอาอุปกรณ์ลอจิกหลายตัวมาต่อเข้าด้วยกัน และจะต้องไม่มีส่วนของการป้อนกลับสัญญาณจากเอาต์พุตมาสู่อินพุต เป็นผลให้การทำงานของวงจรประเภทนี้ มีเอาต์พุตขึ้นอยู่กับอินพุตที่ป้อนเข้ามาเท่านั้น

- **วงจรซีเควนเชียล (Sequential Logic Circuit)**

เป็นวงจรที่นำเอาสัญญาณเอาต์พุตป้อนกลับมาเป็นอินพุตของวงจร เพื่อจะได้มีสภาวะที่สัมพันธ์ต่อเนื่องกัน จึงทำให้การทำงานของวงจรประเภทนี้ มีเอาต์พุตที่ขึ้นอยู่กับอินพุตที่ป้อนเข้ามาและเอาต์พุตก่อนหน้าด้วย

ขั้นตอนการออกแบบวงจรคอมไบเนชั่น

Step 1 : Problem Statement / Specification :

บอกรายละเอียดของวงจรที่ต้องการ

ประกอบด้วย Inputs, Outputs, และฟังก์ชั่นการทำงานของวงจร

Step 2 : Conceptualization :

นำรายละเอียดการทำงานของวงจรมาเขียนเป็น Function Table หรือ Truth Table

Step 3 : Solution / Simplification :

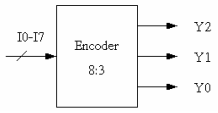
หาฟังก์ชั่นของแต่ละ O/P Sum Of Product (SOP) หรือ Product Of Sum (POS)

Step 4 : Realization :

เขียนวงจรจากฟังก์ชั่นที่ได้

วงจรเข้ารหัส (Encoder)

- หมายถึงวงจรที่เปลี่ยนรหัสใดๆ มาเป็นรหัสเลขฐานสอง เพื่อนำเข้าสู่ขั้นตอนของการประมวลผลในระบบดิจิตอล



วงจรเข้ารหัส (Encoder)

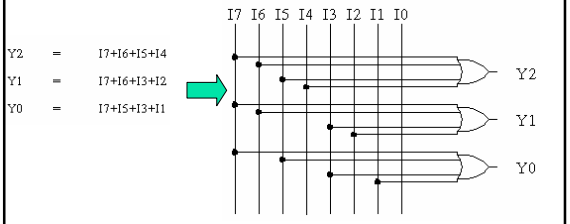
Input								Output		
I7	I6	I5	I4	I3	I2	I1	I0	Y2	Y1	Y0
1	X	X	X	X	X	X	X	1	1	1
0	1	X	X	X	X	X	X	1	1	0
0	0	1	X	X	X	X	X	1	0	1
0	0	0	1	X	X	X	X	1	0	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	0	0	1	0	0	0

Y2 = I7+I6+I5+I4

Y1 = I7+I6+I3+I2

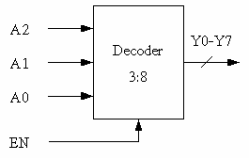
Y0 = I7+I5+I3+I1

วงจรเข้ารหัส (Encoder)



วงจรถอดรหัส (Decoder)

- หมายถึงวงจรที่เปลี่ยนรหัสเลขฐานสอง ไปเป็นรหัสในรูปแบบอื่นๆ เพื่อใช้ในการแสดงผล หรือการสื่อสารระหว่างระบบเป็นต้น



วงจรถอดรหัส (Decoder)

EN	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	
0	X	X	X	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	1	$Y_0 = A'_2A'_1A'_0EN$
1	0	0	1	0	0	0	0	0	0	1	0	$Y_1 = A'_2A'_1A_0EN$
1	0	1	0	0	0	0	0	0	1	0	0	$Y_2 = A'_2A_1A'_0EN$
1	0	1	1	0	0	0	0	1	0	0	0	$Y_3 = A'_2A_1A_0EN$
1	1	0	0	0	0	0	1	0	0	0	0	$Y_4 = A_2A'_1A'_0EN$
1	1	0	1	0	0	1	0	0	0	0	0	$Y_5 = A_2A'_1A_0EN$
1	1	1	0	0	1	0	0	0	0	0	0	$Y_6 = A_2A_1A'_0EN$
1	1	1	1	1	0	0	0	0	0	0	0	$Y_7 = A_2A_1A_0EN$

วงจรถอดรหัส (Decoder)

$Y_0 = A'_2A'_1A'_0EN$

$Y_1 = A'_2A'_1A_0EN$

$Y_2 = A_2A'_1A'_0EN$

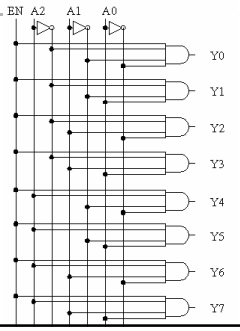
$Y_3 = A'_2A_1A_0EN$

$Y_4 = A_2A'_1A'_0EN$

$Y_5 = A_2A'_1A_0EN$

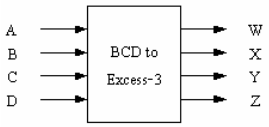
$Y_6 = A_2A_1A'_0EN$

$Y_7 = A_2A_1A_0EN$



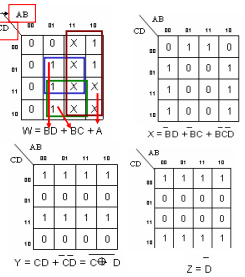
วงจรแปลงรหัส (Code Converter)

- หมายถึงวงจรที่เปลี่ยนรหัสในรูปแบบหนึ่งไปเป็นรหัสอีกรูปแบบหนึ่ง เพื่อให้เกิดความเหมาะกับการทำงานในระบบดิจิตอลที่ใช้รหัสต่างกัน

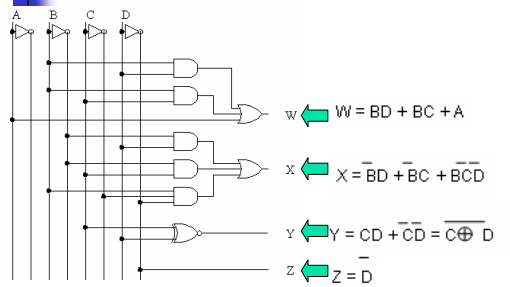


วงจรแปลงรหัส (Code Converter)

Input BCD-8421				Output Excess-3			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

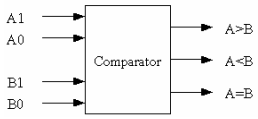


วงจรแปลงรหัส (Code Converter)



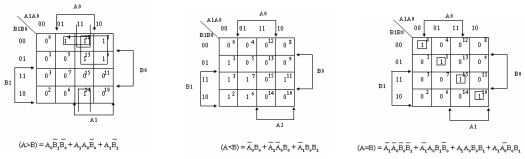
วงจรเปรียบเทียบ (Comparator)

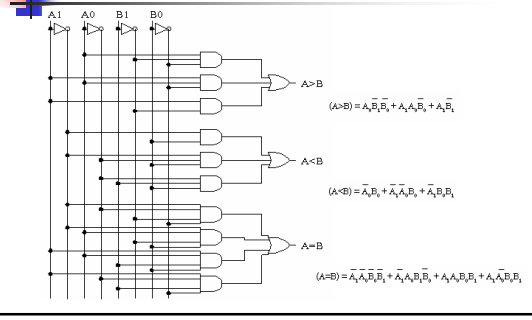
- เป็นวงจรที่ทำหน้าที่เปรียบเทียบอินพุตจำนวน 2 ชุดว่ามีค่าแตกต่าง



อาดับ		อินพุต				เอาท์พุต		
A	B	A_1	A_0	B_1	B_0	A>B	A<B	A=B
0	0	0	0	0	0	0	0	1
0	1	0	0	0	1	0	1	0
0	2	0	0	1	0	0	1	0
0	3	0	0	1	1	0	1	0
1	0	0	1	0	0	1	0	0
1	1	0	1	0	1	0	0	1
1	2	0	1	1	0	0	1	0
1	3	0	1	1	1	0	1	0
2	0	1	0	0	0	1	0	0
2	1	1	0	0	1	1	0	0
2	2	1	0	1	0	0	0	1
2	3	1	0	1	1	0	1	0
3	0	1	1	0	0	1	0	0
3	1	1	1	0	1	1	0	0
3	2	1	1	1	0	1	0	0
3	3	1	1	1	1	0	0	1

วงจรเปรียบเทียบ (Comparator)

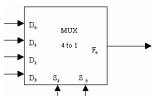




วงจรมัลติเพล็กซ์ (Multiplexer)

Data = 2^n เป็นอินพุต

Selector = n



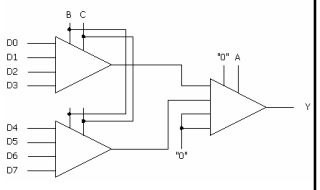
S_1	S_0	F_0
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$F0 = D0\bar{S}1\bar{S}0 + D1\bar{S}1S0 + D2S1\bar{S}0 + D3S1S0$

$= (D0\bar{S}0 + D1S0)\bar{S}1 + (D2\bar{S}0 + D3S0)S1$

วงจรมัลติเพล็กซ์ (Multiplexer)

A	B	C	Y
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7



สร้าง 8 to 1 Mul โดยใช้ 4 to 1 Mul 3 ตัว

วงจรดีมัลติเพล็กซ์ (Demultiplexer)

- เป็นวงจรที่ทำงานตรงกันข้ามกับวงจรมัลติเพล็กซ์
- ใช้เป็นตัวรับสัญญาณในระบบสื่อสารดิจิตอลที่ทำหน้าที่แยกสัญญาณจากสายเส้นเดียวเป็นเอาต์พุตหลายเส้น
- ระหว่างฝ่ายรับและฝ่ายส่งจะต้องทำงานเข้าจังหวะกันจึงจะได้ข่าวสารที่ถูกต้องตรงกัน

วงจรดีมัลติเพล็กซ์ (Demultiplexer)

A	B	C	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

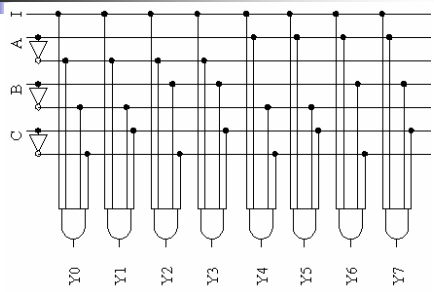
$Y0 = \bar{A}\bar{B}\bar{C}I$ $Y4 = A\bar{B}\bar{C}I$

$Y1 = \bar{A}\bar{B}C I$ $Y5 = A\bar{B}C I$

$Y2 = \bar{A}B\bar{C}I$ $Y6 = AB\bar{C}I$

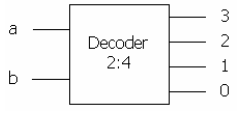
$Y3 = \bar{A}BCI$ $Y7 = ABCI$

วงจรดีมัลติเพล็กซ์ (Demultiplexer)



Implementation Using Decoder

- Decoder ขนาด n : 2^n จะมีอินพุตจำนวน n บิต
- มีเอาต์พุตซึ่งแต่ละเอาต์พุตจะ Active เมื่อมีการป้อนค่าอินพุตที่สอดคล้องกับ Minterm

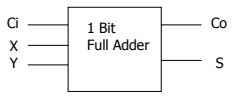


a	b	0	1	2	3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Implementation of 1 bit Full Adder Using 3 : 8 Decoder

Step 1 : Specification :

ออกแบบวงจร Full Adder ใช้สำหรับบวกเลขขนาด 1 บิต x , y ร่วมเข้ากับตัวทดเข้า Ci เพื่อให้ได้ เอาต์พุต เป็นผลบวกเป็น S และตัวทดออกเป็น Co



Implementation of 1 bit Full Adder Using 3 : 8 Decoder

Step 2 : Conceptualization :

Ci	x	y	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Implementation of 1 bit Full Adder Using 3 : 8 Decoder

Step 3 : Simplification :

จาก Truth Table สามารถเขียนเป็นฟังก์ชั่นของ S และ CO ในรูปของ Canonical Sum Of Product ได้ดังนี้

$S = \sum m\,(1,2,4,7)$

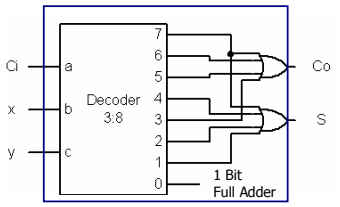
$Co = \sum m\,(3,5,6,7)$

Ci	x	y	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Implementation of 1 bit Full Adder Using 3 : 8 Decoder

Step 4 : Realization :

จากฟังก์ชั่นที่ได้ สามารถสร้างวงจรโดยใช้ 3:8 Decoder

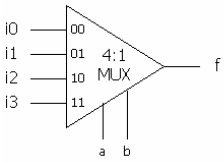


$Co = \sum (3,5,6,7)$

$S = \sum (1,2,4,7)$

Implementation Using Multiplexer

- Multiplexer ขนาด 2^n : 1 จะมี Data Input จำนวน 2^n อินพุต
- มี เอาต์พุต เพียง 1 เอาต์พุต
- มี Selector จำนวน n บิต ที่ใช้สำหรับเลือกเอา Data Input เส้นใด เส้นหนึ่งออกไปทางเอาต์พุต

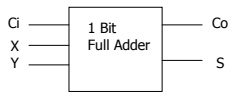


a	b	f
0	0	i0
0	1	i1
1	0	i2
1	1	i3

Implementation of 1 bit Full Adder Using 8 : 1 Multiplexer

Step 1 : Specification :

ออกแบบวงจร Full Adder ใช้สำหรับบวกเลขขนาด 1 บิต x , y ร่วมเข้ากับตัวทดเข้า Ci เพื่อให้ได้ เอาต์พุต เป็นผลบวกเป็น S และตัวทดออกเป็น Co



Implementation of 1 bit Full Adder Using 8 : 1 Multiplexer

Step 2 : Conceptualization :

Ci	x	y	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Implementation of 1 bit Full Adder Using 8 : 1 Multiplexer

Step 3 : Simplification :

จาก Truth Table สามารถเขียนเป็นฟังก์ชั่นของ S และ CO ในรูปของ Canonical Sum Of Product ได้ดังนี้

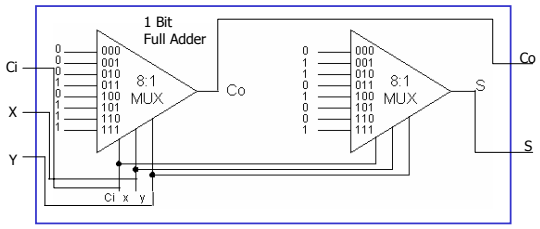
$S = \sum (1,2,4,7)$

$Co = \sum (3,5,6,7)$

Implementation of 1 bit Full Adder Using 8 : 1 Multiplexer

Step 4 : Realization :

จากฟังก์ชั่นที่ได้ สามารถสร้างวงจรโดยใช้ 8:1 MUX



เขียนเป็น Variable – Entering Map

X3	X2	X1	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

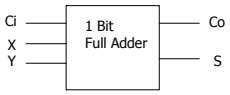
ขุม X3

F	X2X1			
	00	01	11	10
	X3	$\overline{X3}$	1	$\overline{X3}$

Implementation of 1 bit Full Adder Using 4 : 1 MUX

Step 1 : Specification :

ออกแบบวงจร Full Adder ใช้สำหรับบวกเลขขนาด 1 บิต x , y ร่วมเข้ากับตัวทดเข้า Ci เพื่อให้ได้ เอาต์พุต เป็นผลบวกเป็น S และตัวทดออกเป็น Co



Implementation of 1 bit Full Adder Using 4 : 1 MUX

Step 2 : Conceptualization :

Ci	x	y	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Implementation of 1 bit Full Adder Using 4 : 1 MUX

Step 3: Simplification :

จาก Truth Table สามารถเขียน Variable – Entering Map ของ S และ Ci

Ci	x	y	Co	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Co

XY	00	01	11	10
	0	Ci	Ci	1

S

XY	00	01	11	10
	Ci	$\overline{Ci}$	Ci	$\overline{Ci}$

Implementation of 1 bit Full Adder Using 4 : 1 MUX

Step 4 : Realization : สามารถสร้างวงจรโดยใช้ 4:1 MUX