เนื้อหา

วงจรประมวลผลด้านคณิตศาสตร์

Evolution From Adder to Simple Arithmetic Unit

4 bit Binary Adder

มีหลายวิธีในการออกแบบวงจร 4 bit Binary Adder เช่น

- ออกแบบโดยวิธี 4 bit 2-level adder คือการออกแบบโดยตรงจาก Truth table
- ออกแบบโดยวิธีต่อ **cascade** วงจร adder 1 บิต จำนวน 4 วงจร
- ออกแบบโดยวิธี **carry look-ahead**

สำหรับวิธีแรก เราจะต้องใช้ Truth Table ที่มีถึง 9 อินพุตซึ่งหมายถึงมันจะมีอินพุตถึง 512 แบบ ซึ่งจะทำให้วงจรมีความซับซ้อนสูงมาก

ออกแบบวงจร 4 bit adder โดยการต่อ cascade วงจร FA 1 บิต 4 วงจร

การบวกเลข 4 บิต 2 จำนวน x,y และ Carry-in C0 สามารถแสดงให้เห็นได้ดังนี้

C0 Carry - in

x3 x2 x1 x0

y3 y2 y1 y0

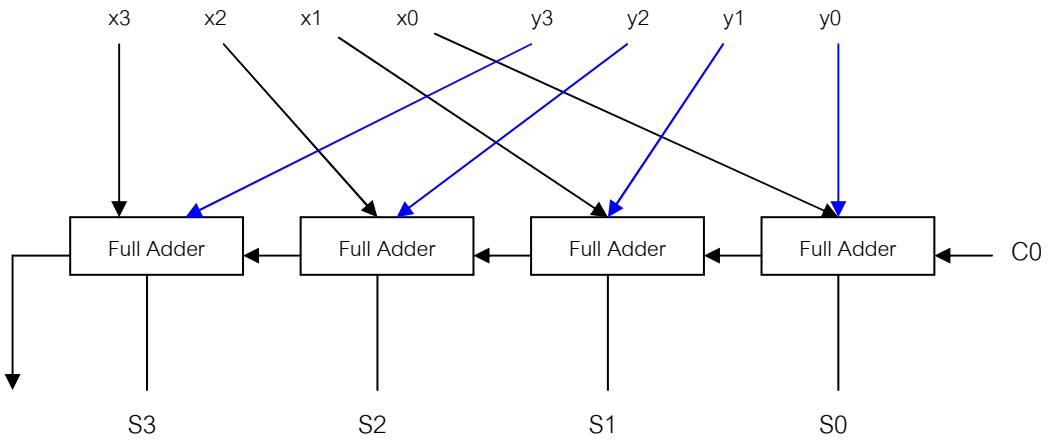
S3 S2 S1 S0 Sum

C4 C3 C2 C1 Carry-out

เราสามารถสร้างวงจร 4 bit adder แบบ ripple carry ได้โดยการนำ 1 bit FA มาต่อ cascade กัน 4 วงจร ดังรูปที่ 1 โดยการบวก LSB (c0, x0, y0) จะอยู่ที่ FA ตัวขวาสุด ซึ่ง Sum S0 และ Carry-out c1 หาได้จาก Boolean Equation ดังนี้

$$S0 = C0 \oplus (x0 \oplus y0)$$

$$C1 = x0\ y0 + C0 (x0 + y0)$$



รูปที่ 1 วงจร 4 bit adder ที่ได้จากการต่อ Cascade วงจร 1 bit FA 4 วงจร

Carry-out จะถูกนำไปเป็น Carry-in ของวงจรซ้ายมือ ซึ่งฟังก์ชั่นของ Sum และ Carry-Out ในทุก Block จะเป็นฟังก์ชั่นเดียวกัน ก็คือ

$$Si = Ci \oplus (xi \oplus yi)$$

$$Ci+1 = xi\ yi + Ci (xi + yi)$$

การต่อ cascade วงจร FA บิต 4วงจรนี้ ถึงจะเป็นวงจรที่ง่าย เนื่องจากทุก Block เป็นฟังก์ชั่นเดียวกันหมด แต่ก็ทำงานได้ช้า เพราะ Carry-in ของแต่ละ Block จะต้องรอ Carry Out จากวงจรทางขวามือ

ออกแบบวงจร 4 bit adder โดยใช้วิธี Carry look-ahead

ในการออกแบบวงจรนั้น เราต้องการให้วงจรมีราคาถูก, มีความซับซ้อนต่ำ, ทำงานเร็ว, และทำงานได้ถูกต้องตามฟังก์ชั่นที่ต้องการ เราจะมาออกแบบวงจร adder ที่มีความซับซ้อนต่ำกว่าแบบ 2-level direct adder และเร็วกว่า cascade ripple-carry adder โดยใช้เทคนิคที่เรียกว่า carry look-ahead

ลองพิจารณา Truth Table ของการบวกตัวเลข 1 บิต ต่อไปนี้

	InputS			OutputS		
	Ci	xi	yi	Ci+1	Si	
	0	0	0	0	0	
	0	0	1	0	1	
	0	1	0	0	1	
Carry-in = 0 →	0	1	1	1	0	← Carry-out = 1 (G)
	1	0	0	0	1	
	1	0	1	1	0	
	1	1	0	1	0	
Carry-in = 1 →	1	1	1	1	1	←Carry-out = 1 (P)

จะเห็นว่า

1. Carry-out ถูก Generate (G) ถ้า carry-in = 0 และ carry-out = 1
   Carry จะถูก Generate ถ้า xi, yi = 1, 1 ดังนั้น Gi = xi yi
2. Carry-out ถูก Propagate (P) ถ้า carry-in = 1 และ carry-out = 1
   Carry จะถูก Propagate ถ้า xi, yi = 0,1 หรือ 1,0 หรือ 1,1 ดังนั้น Pi = xi + yi

Carry-out จะเกิดขึ้นเมื่อ carry ถูก Gernerate หรือถูก Propagate ดังนั้น

$$C_{i+1} = G_i + C_iP_i$$

โดย $G_i = x_i y_i$ และ $P = x_i + y_i$

ในวงจร Adder หลายบิต Gi และ Pi จะถูกสร้างขึ้นอย่างเป็นอิสระต่อกันและในเวลาเดียวกันใน 1 delay time หลังจากมี อินพุต x,y เข้ามา และเราจะหา carry-out ของทุกบิตจาก carry generate Gi และ carry propagate Pi นี้

เทอมของ carry-out C1, C2, C3, C4 สามารถหาได้โดยใช้ Co, Gi และ Pi เท่านั้น

เมื่อ I=0 $C1 = G0 + C0P0$

เมื่อ I=1; $C2 = G1 + C1P1$

$= G1 + (G0 + C0P0) P1$

$= G1 + G0P1 + C0P0P1$

เมื่อ I=2; $C3 = G2 + C2P2$

$= G2 + (G1 + G0P1 + C0P0P1) P2$

$= G2 + G1P2 + G0P1P2 + C0P0P1P2$

เมื่อ I=3; $C4 = G3 + C3P3$

$= G3 + (G2 + G1P2 + G0P1P2 + C0P0P1P2) P3$

$= G3 + G2P3 + G1P2P3 + G0P1P2P3 + C0P0P1P2P3$

ส่วนค่า Sum S3, S2, S1, S0 หาได้จากสมการต่อไปนี้

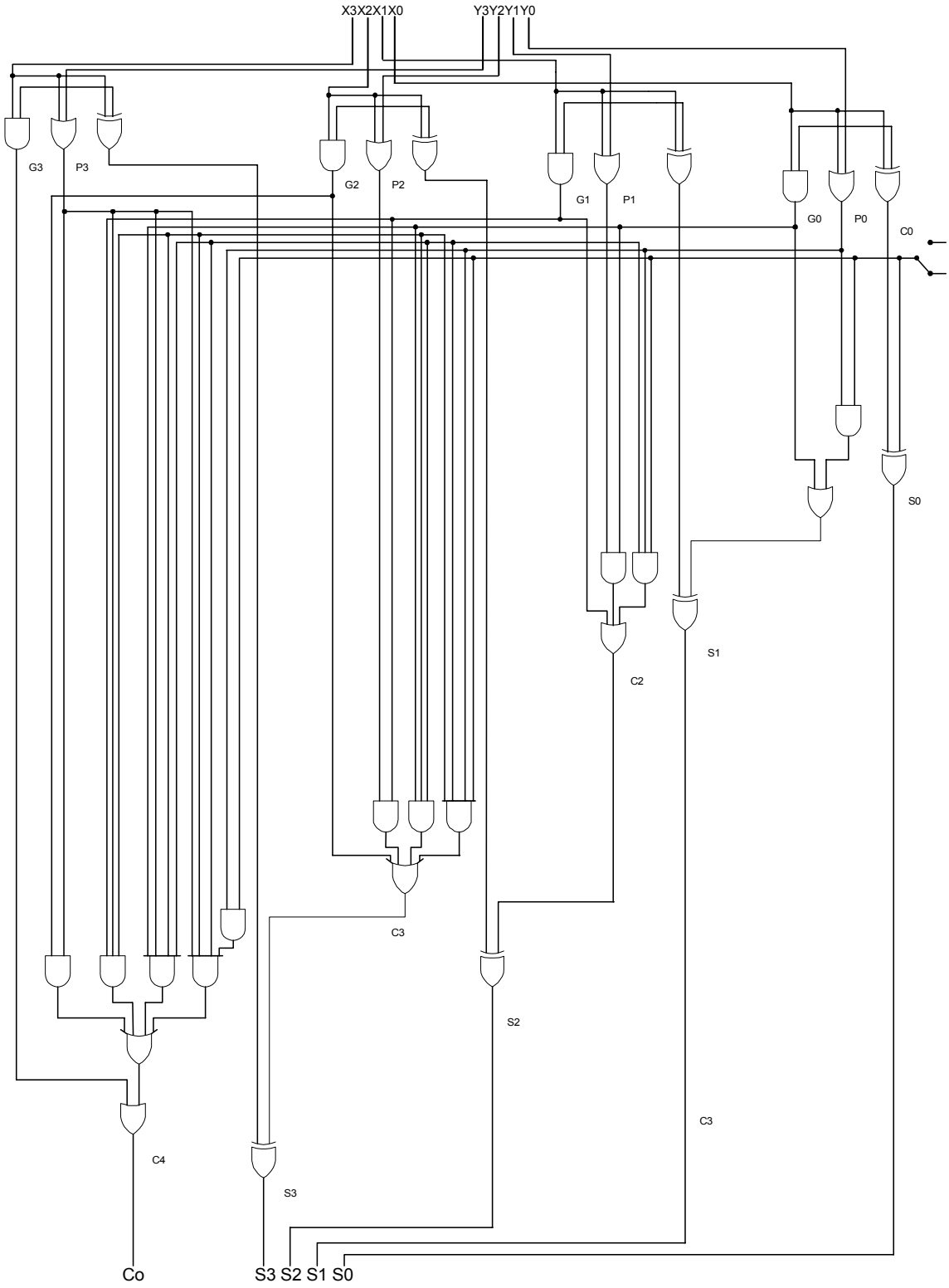
$S0 = C0 \oplus x0 \oplus y0$

$S1 = C1 \oplus x1 \oplus y1$

$S2 = C2 \oplus x2 \oplus y2$

$S3 = C3 \oplus x3 \oplus y3$

แต่ละ Ci จะคำนวณภายใน 3 propagation delay และ Si จะคำนวณได้ภายใน 1 propagation delay ดังนั้นในการบวกแต่ละครั้งจะใช้เพียง 4 delay ในขณะที่วงจร cascade จะใช้ถึง 8 delay logic diagram ของ carry look-ahead adder แสดงให้เห็นดังรูปที่ 2



รูปที่ 2 Logic Diagram ของ 4 bit carry look-ahead adder

4 bit Adder/Subtracter

เราสามารถใช้วงจร binary adder มาทำการลบเลขได้ จากที่ การลบ A-B สามารถทำได้โดยการบวกค่า 2s complement ของ B เข้ากับ A ลองดูตัวอย่างการบวก/ลบ ดังนี้

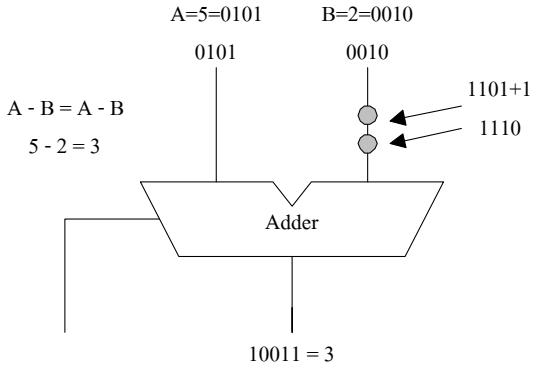
	บวก		ลบ
+5	0101	+5	0101
+ +2	0010	- +2	1110
+7	0111		1 0011
		Drop EAC	
+6	0110	+4	0100
+ -3	1101	- -3	0011
+3	1 0011	+7	0111

Drop EAC

โดยค่า 2s complement หาได้โดย การหา 1s complement แล้วบวกด้วย 1 ซึ่งการหา 1s complement ทำได้โดยการ invert ทุกบิตของตัวเลขนั้น ดังนั้นเราสามารถนำวงจร binary adder มาทำฟังก์ชั่น ของการลบเลข binary ได้โดยการ invert ทุกบิตของตัวลบ (เพื่อหา 1s complement) แล้วบวก 1 เข้าที่ carry-in ci = 1 (เพื่อหา 2s complement)

ดังนั้นการที่จะสร้างวงจร Adder/Subtracter เป็นวงจรเดียวกัน สามารถทำได้ตามหลักการต่อไปนี้

- การบวก เอา A (ตัวตั้ง) บวก B (ตัวบวก) โดยไม่มีการเปลี่ยนค่าใดก่อน
- การลบ เปลี่ยน B เป็น 2s complement ก่อนจะบวก B' เข้ากับ A เพื่อทำการลบ A – B



รูปที่ 3 แสดง process ของการลบ

การใช้ XOR เป็น Selective Complementer

การสร้างวงจร Adder/Subtracter เป็นวงจรเดียวกันนั้น จะเห็นว่าเราต้องมีการเลือกใช้ค่าจริง กับค่า 2s complement เนื่องจาก ถ้าทำการบวก เราจะใช้ค่าจริงของตัวบวก เข้าไปบวกกับตัวตั้ง ถ้าทำการลบ เราต้องใช้ค่า 2s complement เข้าไปบวกกับตัวตั้ง แต่ตัวบวกกับตัวลบนั้น ใช้ อินพุตเส้นเดียวกัน ดังนั้นเราต้องมีการ control ว่า เมื่อใดจะใช้ค่าจริง และเมื่อใดจะใช้ค่า 1s complement การ 2s complement นั้น เราจะต้องใช้วงจรเข้ามาควบคุม โดยใช้สัญญาณควบคุมให้ทำการ 1s complement อินพุต และทำการบวก 1 เข้าที่ carry-in ด้วย

เราจะใช้วงจร combination ที่มี W เป็น control I/P และ y เป็น Data I/P

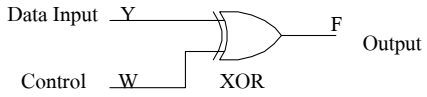
ถ้า W = 0, data I/P y จะผ่านไปยัง O/P

ถ้า W = 1, data I/P y จะถูก complement

วงจรนี้เรียกว่า selective Complementer สามารถเขียนเป็น Truth table ได้ดังนี้

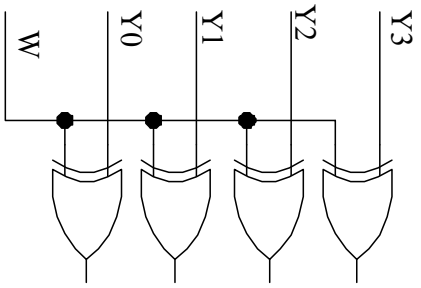
Inputs		Outputs	
Control	Data	F	
0	0	0	
0	1	1	Pass Y
1	0	1	
1	1	0	Complement Y

จาก Truth Table จะได้ $F = W y + W y = W \oplus y$ ดังนั้นเราจึงสามารถทำ selective Complementer ได้โดยใช้ XOR ดังรูปที่ 4



รูปที่ 4 ใช้ XOR เป็น Selective Complementer

Control I/P W นี้ สามารถจะใช้ ควบคุม อินพุต กี่เส้นก็ได้ เพื่อทำเป็น Selective Complementer (inverter) ซึ่งจะทำการเลือกว่าจะผ่าน อินพุต ไปทั้งหมด หรือจะทำการ invert ทุกๆ อินพุต ดังรูปที่ 5



รูปที่ 5 Complementer 4 bit โดยใช้ Control 1 เส้น

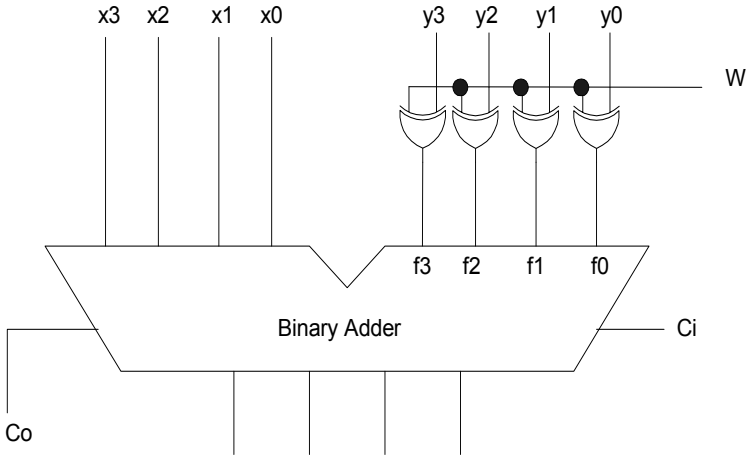
สร้างวงจร 4 bit binary Adder/Subtracter

วงจร 4 bit binary Adder/Subtracter สามารถสร้างขึ้นได้ดังรูปที่ 6 โดยที่

W = 0 เมื่อทำการบวก y จะถูก pass ไปทำการบวกกับ x

W = 1 เมื่อทำการลบ y จะถูก invert ทุกบิต เพื่อทำ 1s complement และ 1 (จาก W) จะถูกบวกเข้าไปที่

carry-in เพื่อทำเป็น 2s complement แล้วทำการบวกกับ x



รูปที่ 6 4 bit Adder/Subtracter

การตรวจจับ Overflow ในการบวก/ลบเลข binary

Overflow จะเกิดขึ้นเมื่อค่าของผลลัพธ์เกินกว่าค่าสูงสุดที่จะเป็นไปได้ตามจำนวนบิตที่มีอยู่จากวงจร Adder/Subtracter นี้ Overflow จะเกิดขึ้นเมื่อ carry-in และ carry-out ของ sign bit มีค่าต่างกัน ดังตัวอย่าง

1. Overflow เมื่อทำการบวกเลขบวก 2 จำนวน

	0 1 0 0	Carries	
+4	0 1 0 0		
+ +4	0 1 0 0		
+8	1 0 0 0	=	- 0 Erroneous

2. Overflow เมื่อบวกเลขลบ 2 จำนวน

	1 0 0 0	Carries	
-5	1 0 1 1		
+ -4	1 1 0 0		
-9	0 1 1 1	=	+7 Erroneous

3. Overflow เมื่อลบเลขที่เครื่องหมายต่างกัน

	1 0 0 1	Carries	
-7	1 0 0 1		
- +3	1 1 0 1		
-4	0 1 1 0	=	+6 Erroneous

การตรวจจับ Overflow สามารถทำได้โดย

$$\text{Overflow} = Ci \oplus Co$$

เมื่อ Ci, Co คือ carry-in และ carry-out ของ sign bit วงจรตรวจสอบ Overflow จะใช้สำหรับแจ้งว่ามี Overflow เกิดขึ้น

Design of 4 bit 8 Function Arithmetic Unit

AU อย่างง่ายๆ สามารถออกแบบได้โดยการรวมเอาวงจรควบคุมบางอย่างเข้าไว้กับวงจร Adder/Subtracter เราจะลองมาออกแบบ AU ที่ทำงานได้หลาย Function คือ Pass, Add, Subtract, Decrement, Increment

จากวงจร Adder/Subtracter ซึ่งมี อินพุต X,Y และ Ci ที่สามารถทำการบวกและลบเลข binary ได้ เราจะนำมาเพิ่มเติมฟังก์ชั่นบางฟังก์ชั่นลงไปโดยมีวิธีการดังนี้

Add X,Y ทำได้โดย X+Y

Subtract X,Y ทำได้โดย X+Y'+1

Pass X ทำได้โดย X+0s หรือ X+1s+1

X = 0 0 1 1 = 3	X = 0 0 1 1 = 3
F = 0 0 0 0	F = 1 1 1 1
Ci = 0	Ci = 1
Z = 0 0 1 1 = 3	Z = 1 | 0 0 1 1 = 3

Decrement X ทำได้โดย X + 1s

X = 0 1 0 1 = 5

F = 1 1 1 1

Ci = 0

Z = 1 | 0 1 0 0 = 4

Increment X ทำได้โดย X + 0s +1

X = 0 1 1 0 = 6

F = 0 0 0 0

Ci = 1

Z = 0 1 1 1 = 7

จะเห็นว่าค่า F ที่จะนำมาบวกกับ X นั้นมีได้หลายค่า คือ Y, Y', 0s, 1s ซึ่งเราจะต้องทำการออกแบบวงจรควบคุมในส่วนนี้ว่า ค่าใดที่จะนำไปบวกกับ X และ Carry-in

ออกแบบฟังก์ชั่นของ True/Complement/Zero/One

ฟังก์ชั่นของ True/Complement/Zero/One หรือ TC01 จะเอาไว้สำหรับนำไปรวมกับ Adder/Subtracter เพื่อให้ทำงานตามฟังก์ชั่นที่ต้องการ

TC01 นี้ จะประกอบไปด้วย

I/P Y หลายบิต

O/P F หลายบิต

สัญญาณควบคุม 2 เส้น (E = Enable, W = complement)

ผลที่ได้ทาง เอาต์พุต F จะเป็นค่า true หรือ Complement หรือ Zero หรือ One เราจะใช้ E,W เป็นตัวควบคุมดังนี้

Control		Output
W	E	F
0	0	0s
0	1	Y
1	0	1s
1	1	Y’

I/P Y จะถูก Disable เมื่อ E = 0 : O/P F จะเป็น 0 ทั้งหมด หรือ 1 ทั้งหมด

I/P Y จะถูก Enable เมื่อ E = 1 : O/P F จะเป็น Y หรือ Y’

นั่นคือ ฟังก์ชั่นของ TC01 จะออกแบบ Enable control ได้โดยใช้ AND gate ซึ่ง E จะเป็นตัว Enable ว่าจะให้ Y ผ่านไปได้หรือไม่นั่นเองเอาต์พุตจากการ Disable หรือ Enable จะถูกนำมาควบคุมโดย W อีกครั้ง

E	y	G = E.y	O/P	
0	0	0	0	Disable Gate
0	1	0	0	
1	0	0	y	Enable Gate
1	1	1	y	

ถ้า อินพุต ถูก Disable W จะเป็นตัวกำหนดว่า O/P F จะเป็น 0 หรือ 1 ทั้งหมด

ถ้า อินพุต ถูก Enable W จะเป็นตัวกำหนดว่า O/P F จะเป็น Y หรือ Y'

	W E Y	G = E.y	F = W $\oplus$(E.y)	
	0 0 0	0	0	0
	0 0 1	0	0	
Pass	0 1 0	0	0	True
	0 1 1	1	1	
	1 0 0	0	1	1
	1 0 1	0	1	
Complement	1 1 0	0	1	Complement
	1 1 1	1	0	

ถ้า E = 0 , Disable Data , G =0

เมื่อ W =0, G = 0 จะ pass เข้าไปที่ F

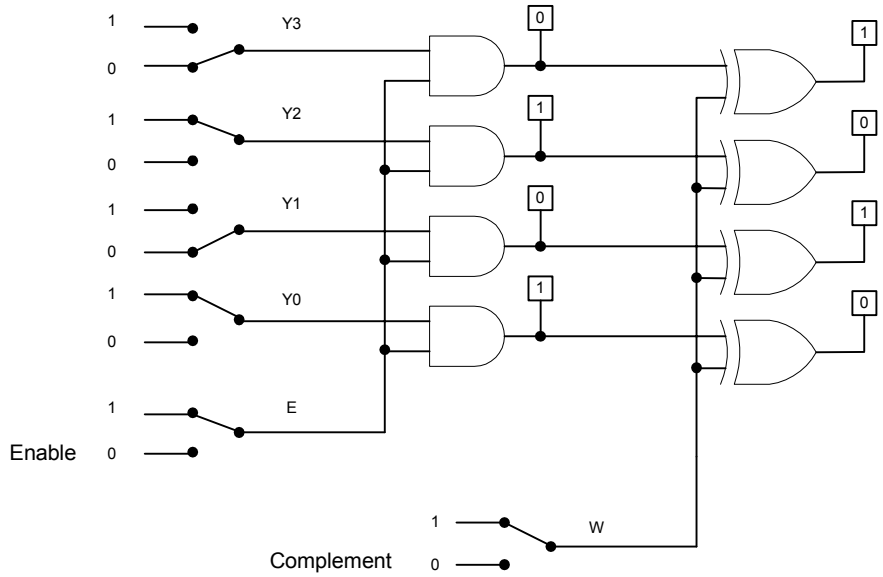
เมื่อ W =1, G จะ ถูก Complement เป็น 1 ไปที่ F

ถ้า E = 1 , Enable Data , G = y

เมื่อ W =0, enable data (E.y) จะ pass เข้าไปที่ F

เมื่อ W =1, enable data (E.y) จะ ถูก Complement เข้าไปที่ F

Function จาก truth Table นี้ สามารถสร้างขึ้นโดยใช้วงจรดังรูปที่ 7



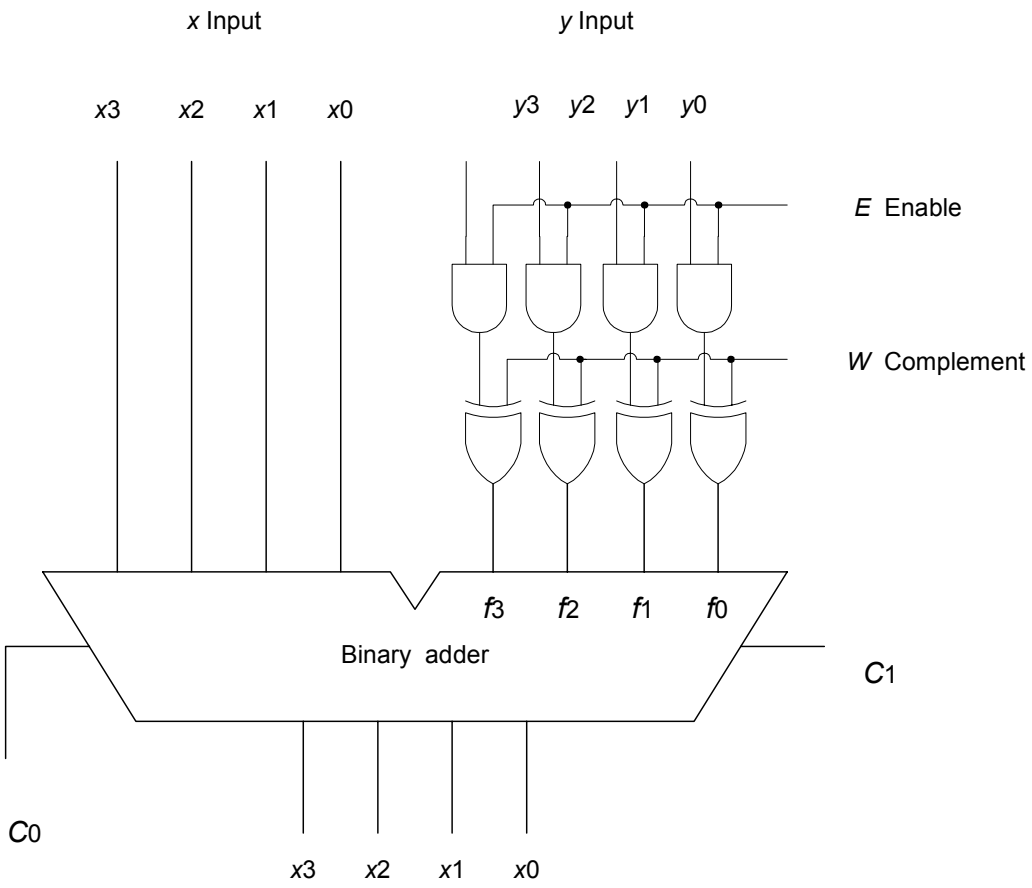
รูปที่ 7 การใช้เกตเพื่อทำการ Enable และ Control

ตอนนี้เราก็ได้วงจรของ TC01 แล้ว ซึ่งจะได้นำมาต่อกับ Adder เพื่อใช้สร้าง Arithmetic unit ต่อไป

Implement of 4 bit – 8 Function Arithmetic Unit

เมื่อนำวงจร TC01 มาต่อควบคุม Binary Adder ดังรูปที่ 8 จะได้ AU ที่สามารถทำการคำนวณได้ตามฟังก์ชั่นใน Truth Table ต่อไปนี้

Ci W E	F หรือ Y	Z	Function
0 0 0	0s	X	Pass x
0 0 1	Y	X + Y	Add
0 1 0	1s	X + 1s	Decrement X
0 1 1	Y’	X + Y’	X+1s Complement
1 0 0	0s	X + 1	Increment X
1 0 1	Y	X + Y + 1	Add with carry-in
1 1 0	1s	X + 1s + 1	Pass X
1 1 1	Y’	X + Y’+ 1	Subtract



รูปที่ 8 รวม ANDs, XORs และ Adder เข้าด้วยกันเป็น AU