เนื้อหา

วงจรประมวลผลด้านคณิตศาสตร์

Binary Arithmetic Logic Unit

คอมพิวเตอร์อย่างธรรมดาที่สุด ALU จะต้องมีความสามารถในการคำนวณ (บวก/ลบ), การกระทำทางลอจิก (AND, OR, XOR) และการเปรียบเทียบ (>, =, <)

Compare Operation and Comparator 1 – bit Comparator

การเปรียบเทียบตัวเลข 1 บิต 2 จำนวน สามารถเขียนเป็น Truth table ได้ดังนี้

		E	G	L
x	**y**	**x=y**	**x>y**	**x<y**
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

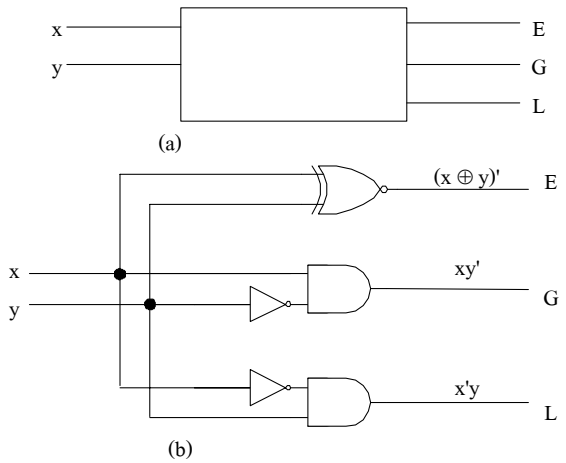
ฟังก์ชั่นของ E, G, L เป็นดังนี้

$E = \overline{x}\,\overline{y} + x\,y = \overline{x \oplus y}$

$G = x\,\overline{y}$

$L = \overline{x}\,y$

ซึ่งเราสามารถเขียน Block diagram และ Logic diagram ของวงจร Comparator ได้ดังรูปที่ 9



รูปที่ 9 a) block diagram b) Logic Diagram ของวงจร 1-bit comparator

2-bit Comparator

การเปรียบเทียบตัวเลข 2 บิต 2 จำนวน คือ x1x0 และ y1y0 ตัวเลขทั้ง 2 จำนวนจะเท่ากัน เมื่อ x1 = y1 , x0 = y0 ดังนั้น สามารถเขียนฟังก์ชั่นของ E (equal) ได้เป็น

$$E = \overline{(x1 \oplus y1)} \cdot \overline{(x0 \oplus y0)}$$

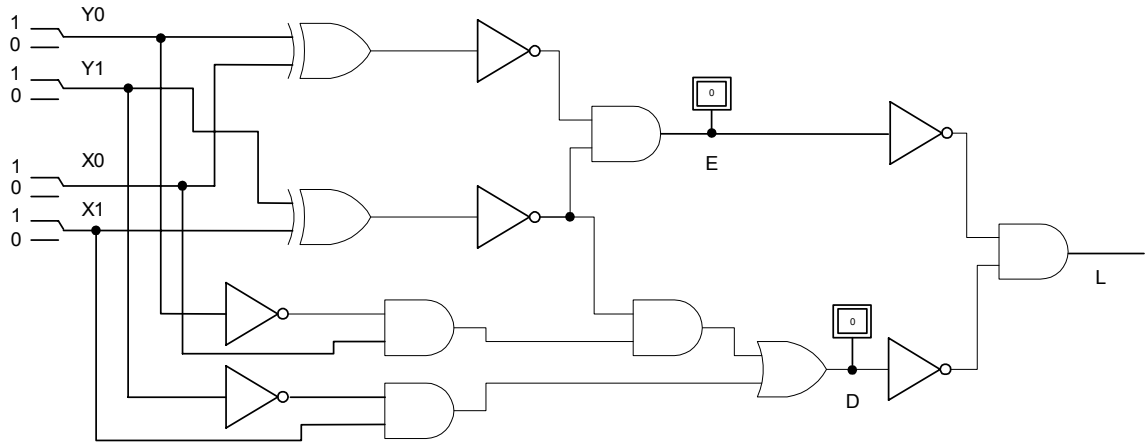
x > y เมื่อ x1> y1 หรือ x1 = y1 แต่ x0 > y0 สามารถเขียนฟังก์ชั่นของ G (Greater than) ได้เป็น

$$G = x1\,\overline{y1} + \overline{(x1 \oplus y1)}\,x0\,\overline{y0}$$

x < y เมื่อ x1< y1 หรือ x1 = y1 แต่ x0 < y0 สามารถเขียนฟังก์ชั่นของ L (Less than) ได้เป็น

$$G = \overline{x1}\,y1 + \overline{(x1 \oplus y1)}\,\overline{x0}\,y0$$

Logic Diagram แสดงให้เห็นดังรูปที่ 10



รูปที่ 10 Logic Diagram ของ 2-bit Comparator

4-bit Comparator

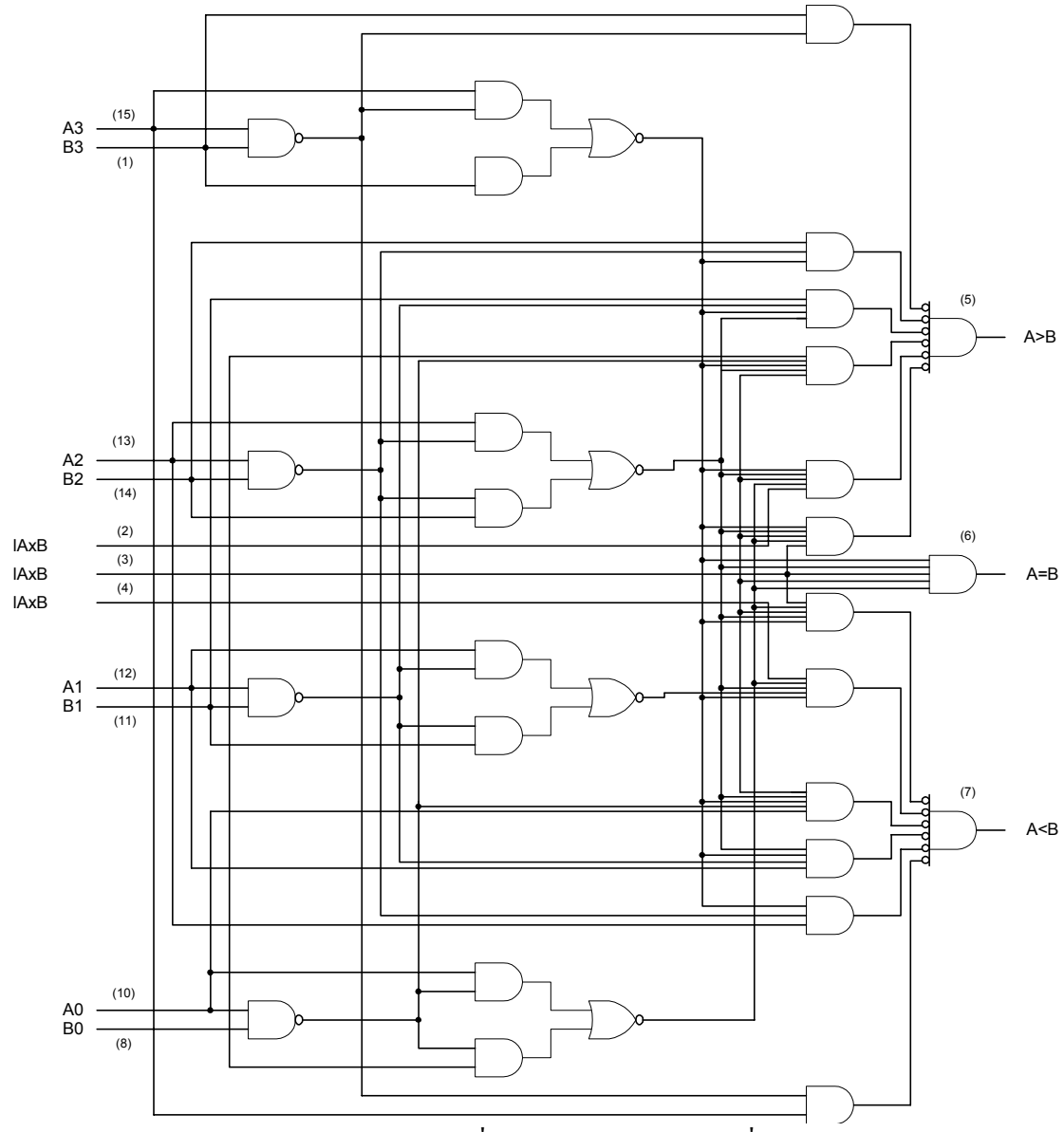
ฟังก์ชั่นของวงจรเปรียบเทียบ 4 บิต (x3x2x1x0 และ y3y2y1y0) สามารถหาได้โดยขยายจากรูปแบบของ 2–bit comparator ซึ่งจะได้เป็น

$$G = x1\,\overline{y1} + \overline{(x0 \oplus y0)}\,\overline{x0}\,y0$$

$$E = \overline{(x1 \oplus y1)} \cdot \overline{(x0 \oplus y0)} \cdot \overline{(x1 \oplus y1)} \cdot \overline{(x0 \oplus y0)}$$

$$G = x3\,\overline{y3} + \overline{(x3 \oplus y3)}\,x2\,\overline{y2} + \overline{(x3 \oplus y3)}\,\overline{(x2 \oplus y2)}\,x1\,\overline{y1} + \overline{(x3 \oplus y3)}\,\overline{(x2 \oplus y2)}\,\overline{(x1 \oplus y1)}\,x0\,\overline{y0}$$

$$L = \overline{x3}\,y3 + \overline{(x3 \oplus y3)}\,\overline{x2}\,y2 + \overline{(x3 \oplus y3)}\,\overline{(x2 \oplus y2)}\,\overline{x1}\,y1 + \overline{(x3 \oplus y3)}\,\overline{(x2 \oplus y2)}\,\overline{(x1 \oplus y1)}\,\overline{x0}\,y0$$



ตัวอย่างของ 4- bit comparator คือ IC # 7485 ซึ่งมี logic diagram ดังรูปที่ 11

รูปที่ 11 logic diagram ของ IC 7485 (4-bit comparator)

Arithmetic and Logic Operation

โดยทั่วไป ALU จะต้องมีความสามารถในการทำ operation ทางคณิตศาสตร์ (บวก/ลบ) และ operation ทางลอจิก (AND, OR, XOR) ซึ่ง application ของ operation ทางลอจิกเหล่านี้ คือ การ selectively mask, merge และ complement

AND operation สามารถใช้ทำการ mask (disable) ส่วนของ word และ extract (enable) ส่วนที่เหลือ ดังตัวอย่างที่ 1

ตัวอย่างที่ 1 ถ้า 8-bit mask เป็น 11110000 mask นี้จะนำไป AND ด้วย data word (abcdefgh) ดังนี้

	a b c d e f g h	data word
AND	1 1 1 1 0 0 0 0	Mask
	a b c d 0 0 0 0	Result

จะเห็นว่า 4 บิตซ้ายจะถูก extract มาจาก data word แต่ 4 บิตขวาจะถูก mask (disable) เป็น 0

OR operation สามารถใช้ในการ merge (รวม) 2 data word เข้าด้วยกัน ผลของการ OR จะได้ word ที่มี 1 อยู่ในบิตที่มี 1 อยู่ใน I/P words ดังตัวอย่างที่ 2

ตัวอย่างที่ 2 มี 2 data word นำมา merge กัน

0 0 1 1 1 1 0 0	data word 1
0 1 0 1 0 1 0 1	data word 2
0 1 1 1 1 1 0 1	Result

XOR operation สามารถใช้ในการเลือก pass/complement แต่ละบิตของ data word ดังตัวอย่างที่ 3

ตัวอย่างที่ 3 ถ้า 8-bit control word เป็น 00001111 มันจะถูกนำไป OR กับ data word

1 1 0 0 0 1 0 1	data word
0 0 0 0 1 1 1 1	Control word
1 1 0 0 1 0 1 0	Result

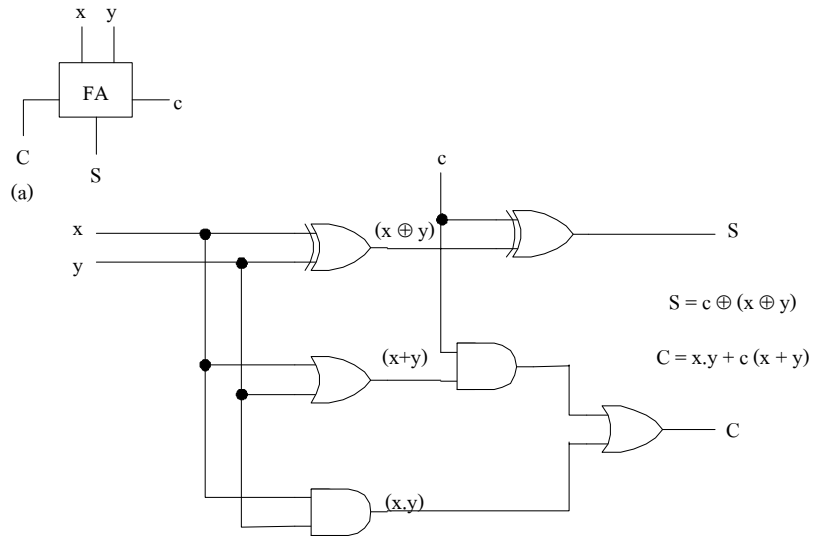
จะเห็นว่า 4 บิตซ้ายจะถูก Pass มาจาก data word โดยไม่มีการเปลี่ยนแปลง 4 บิตขวา จะถูก complement

Operation ทางลอจิกทั้ง 3 operation จะฝังอยู่ในฟังก์ชั่นของ Sum และ Carry-out ของ FA นั่นคือ

$Si = Ci \oplus (Xi \oplus Yi)$

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

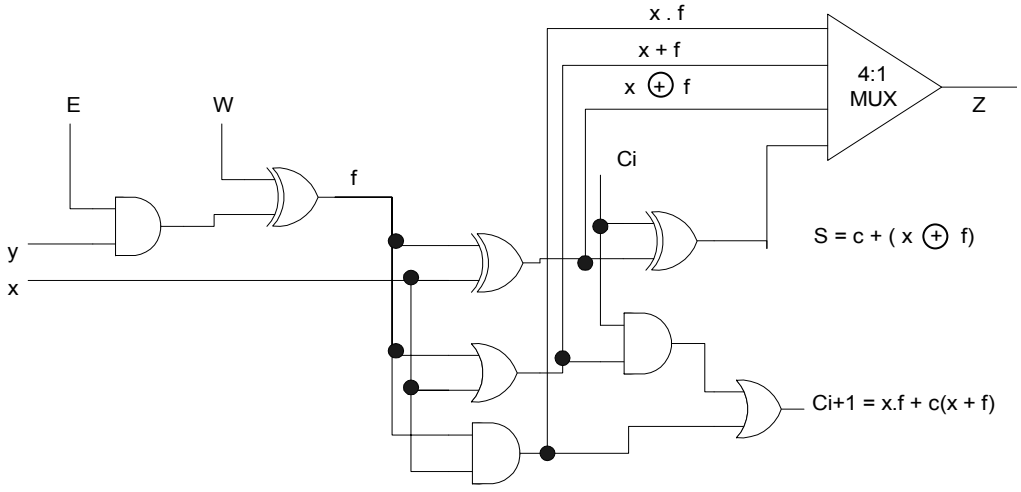
จะเห็นว่ามี operation AND (xi yi), OR (xi + yi), XOR (xi $\oplus$ yi) อยู่ในฟังก์ชั่นของ sum และ carry out ของ FA ซึ่งเราสามารถที่จะดึงเอา operation เหล่านั้นออกมาใช้ได้ ดังรูปที่ 12



รูปที่ 12 1-bit FA ที่มีฟังก์ชั่นทางลอจิกฝังอยู่

Design of Cascade (Bit slide) ALU

การออกแบบ ALU เราจะใช้ฟังก์ชั่นของ TC01 ต่อเข้ากับ adder เพื่อทำเป็น 1-bit ALU ดังรูปที่ 13 ซึ่งจะมีทั้ง operation ทางคณิตศาสตร์ ซึ่งได้ เอาต์พุต ที่ S และ C และ Operation ทางลอจิกที่จุดต่าง ๆ ดังรูป ซึ่งค่าที่ได้จาก แต่ละจุด ในแต่ละเงื่อนไขของ Control E และ W จะเป็นดังตารางที่ 1



รูปที่ 13 1-bit ALU อย่างง่าย

ตารางที่1 Operation ทางลอจิกของ ALU 1 บิต

	Logic Operations					Description
			AND	OR	XOR	
W E F	X.F	X + F	X ⊕ F	AND	OR	XOR
0 0 0	0	X	X	Clear	Pass X	Pass Y
0 1 Y	X.Y	X + Y	X ⊕ Y	X AND Y	X OR Y	X XOR Y
1 0 1	X	1	X'	Pass X	Set	Comp X
1 1 Y'	X.Y'	X + Y'	X ⊕ Y'	X AND Y'	X OR Y'	X XOR Y'

จากรูปที่ 13 จะใช้ 4:1 MUX ในการเลือก Data O/P ว่าเป็น Operation ใด (AND, OR, XOR หรือ SUM) โดยมีสัญญาณ a, b เป็น selector control

bit-slice ALU ขนาด 4 บิต สามารถทำได้โดยการต่อ 1-bit ALU มาต่อ cascade กันดังรูปที่ 14 ซึ่ง ALU ตัวนี้ จะทำงานตาม operation ของการคำนวณทางคณิตศาสตร์และลอจิ ดังตารางที่ 2

ตารางที่ 2 operation ทางคณิตศาสตร์และลอจิกของ 4-bit ALU อย่างง่าย

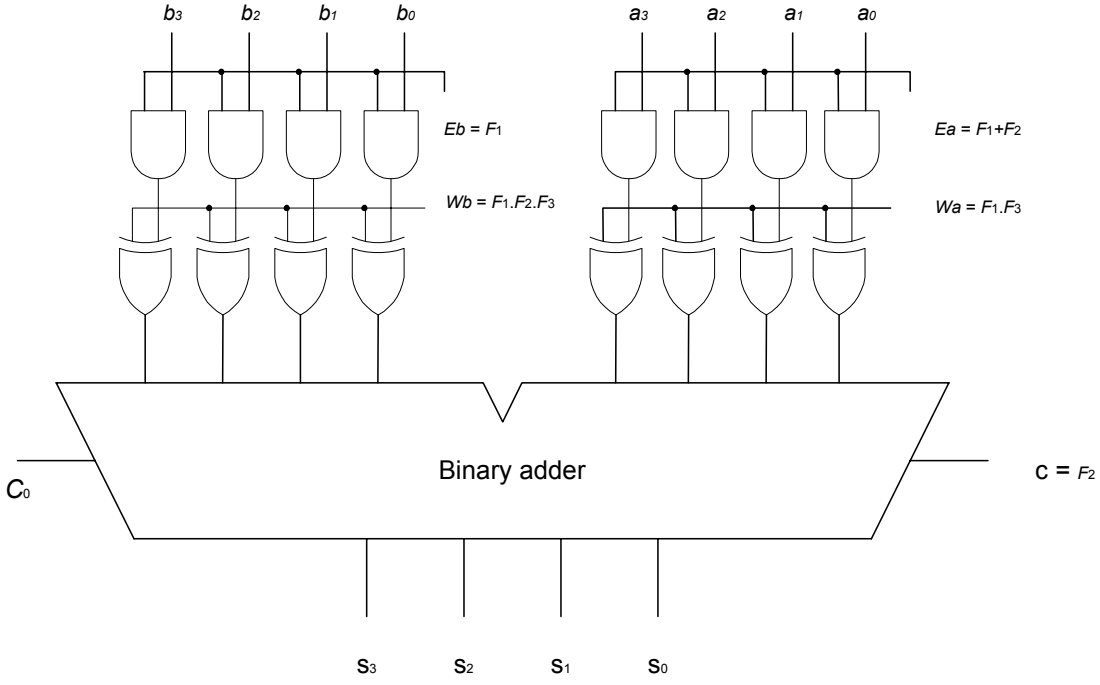
a b	Ci W E	F	Z	Function
0 0	- 0 0	0	0	Clear
	- 0 1	Y	X AND Y	AND x,y
	- 1 0	1	X	Pass x
	- 1 1	Y'	X AND Y'	AND x,y'
0 1	- 0 0	0	X	Pass x
	- 0 1	Y	X OR Y	OR x,y
	- 1 0	1	1	Set
	- 1 1	Y'	X OR Y'	Or x,y'
1 0	- 0 0	0	X	Pass x
	- 0 1	Y	X XOR Y	XOR x,y

	- 1 0	1	X'	Complement X
	- 1 1	Y'	X XOR Y'	XOR x,y'

1 1	0 0 0	0	X	Pass x
	0 0 1	Y	X + Y	Add x,y
	0 1 0	1	X + 1s	Decrement x
	0 1 1	Y'	X + Y'	X + 1s
	1 0 0	0	X + 1	complement y
	1 0 1	Y	X + Y + 1	Increment x
	1 1 0	1	X + 1s + 1	Add with carry
	1 1 1	Y'	X + Y' + 1	Pass x
				Subtract x,y

Design of a Second Binary ALU

ALU สามารถจะออกแบบได้นับหมื่นนับแสนแบบ ซึ่งขึ้นอยู่กับวิธีการผสมกันของ ตัวกระทำ (operation) ทางคณิตศาสตร์, ลอจิก และการเปรียบเทียบ เราจะลองมาดูการออกแบบ ALU อีกแบบหนึ่ง ซึ่งสร้างจาก 4-bit adder ร่วมกับ TC01 สองวงจร ดังรูปที่ 14



รูปที่ 14 block diagram ของ binary ALU แบบที่สอง

ถ้า Eb, Ea, Wb, Wa และ ci ซึ่งใช้เป็น control ใช้เป็นอิสระจากกัน ALU ตัวนี้จะมีถึง 32 operation แต่เวลานี้ เราต้องการฟังก์ชั่นง่ายๆ เพียง 8 ฟังก์ชั่นเท่านั้น เราจึงต้องทำ control ขึ้นมาใหม่ เพียง 3 ตัวเท่านั้น คือ F1, F2, F3 ซึ่งจะใช้ควบคุม Eb, Ea, Wb, Wa และ ci อีกที

ถ้ากำหนดให้ ALU ตัวนี้ มีการทำงานซึ่งควบคุมโดย F1, F2, F3 ดังตารางที่ 3

ตารางที่ 3 operation ทางคณิตศาสตร์ของ ALU แบบที่ 2

Opcode			**Adder Output**	**Operation**
F1	**F2**	**F3**		
0	0	0	0 ⟶ S	Clear
0	0	1	A' ⟶ S	Complement A
0	1	0	A + 1 ⟶ S	Increment A
0	1	1	A' + 1 ⟶ S	Negate A = 2s complement of A
1	0	0	B ⟶ S	Transfer B
1	0	1	A + B ⟶ S	Add without carry
1	1	0	A + B + 1 ⟶ S	Add with carry
1	1	1	A + B' + 1 ⟶ S	Subtract = A - B

กำหนดให้ g = O/P ที่ AND gate และ f = O/P ที่ XOR gate ของ TC01 ในแต่ละค่า ของ F1, F2,F3 จะต้องใช้ค่า g และ f ดังตารางที่ 4

ตารางที่ 4 ตารางการทำงานซึ่งแสดงค่า O/P g,f ของ TC01 ด้วย

Opcode								**Adder Output**	**Operation**
F_1	F_2	F_3	Ci	g_b	f_b	g_a	f_a	$Z = f_b + f_a + C$	
0	0	0	0	0	0	0	0	0 ⟶ S	Clear
0	0	1	0	0	0	A	A’	A' ⟶ S	Complement A
0	1	0	1	0	0	A	A	A + 1 ⟶ S	Increment A
0	1	1	1	0	0	A	A’	A’ + 1 ⟶ S	Negate A = 2s complement of A
1	0	0	0	B	B	0	0	B ⟶ S	Transfer B
1	0	1	0	B	B	A	A	A + B ⟶ S	Add without carry
1	1	0	1	B	B	A	A	A + B + 1 ⟶ S	Add with carry
1	1	1	1	B	B’	A	A	A + B’ + 1 ⟶ S	Subtract = A - B

พิจารณา F1, F2, F3 ซึ่งจะทำให้จะทำให้เกิด operation ดังตารางที่ 4

- ต้องการ B ที่ gb (ต้องให้ Eb = 1) เมื่อ F1 = 1 ดังนั้น Eb = F1
- ต้องการ A ที่ ga (ต้องให้ Ea = 1) เมื่อ F2 =1 หรือ F3 = 1 ดังนั้น Ea = F2 + F3
- ต้องการ A' ที่ fa (ต้องให้ Wa = 1) เมื่อ F1 =0 และ F3 = 1 ดังนั้น $\overline{Wa} = F1 . F3$
- ต้องการ B' ที่ fb (ต้องให้ Wb = 1) เมื่อ F1=1, F2=1, F3=1 ดังนั้น Wb = F1.F2.F3

ดังนั้น จุดควบคุม Eb, Wb, Ea, Wa จะถูก control โดย F1, F2, F3 ดังฟังก์ชันต่อไปนี้ซึ่งแสดงให้เห็นในรูปที่ 15

$Eb = F1$

$Ea = F2 + F3$

$Wa = \overline{F1} . F3$

$Wb = F1.F2.F3$

$Ci = F2$

BCD Arithmetic Unit

Design of a BCD adder/subtracter

Design of BCD adder

การบวกตัวเลข BCD 1 digit 2 จำนวน x และ y และ carry-in นั้น แต่ละ I/P จะมีค่าที่เป็นไปได้ดังต่อไปนี้

$Ci = \{ 0,1 \}$

$X = x3\ x2\ x1\ x0 = \{ 0,1,2,3,4,5,6,7,8,9 \}$

$Y = y3\ y2\ y1\ y0 = \{ 0,1,2,3,4,5,6,7,8,9 \}$

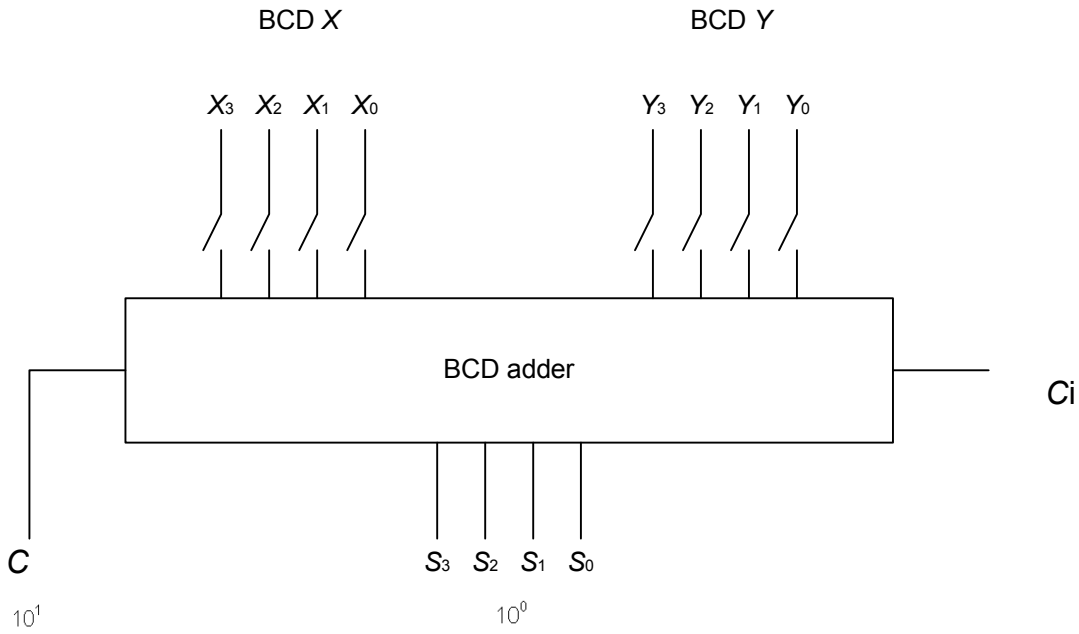
ดังนั้นการบวก X + Y + Ci จึงสามารถให้ผลได้ถึง 19 ค่า ดังตารางที่ 5

ตารางที่ 5 BCD addition table

Sum	Y	Ci = 1									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
X											
0		1	2	3	4	5	6	7	8	9	10
1		2	3	4	5	6	7	8	9	10	11
2		3	4	5	6	7	8	9	10	11	12
3		4	5	6	7	8	9	10	11	12	13
4		5	6	7	8	9	10	11	12	13	14
5		6	7	8	9	10	11	12	13	14	15
6		7	8	9	10	11	12	13	14	15	16
7		8	9	10	11	12	13	14	15	16	17
8		9	10	11	12	13	14	15	16	17	18
9		10	11	12	13	14	15	16	17	18	19

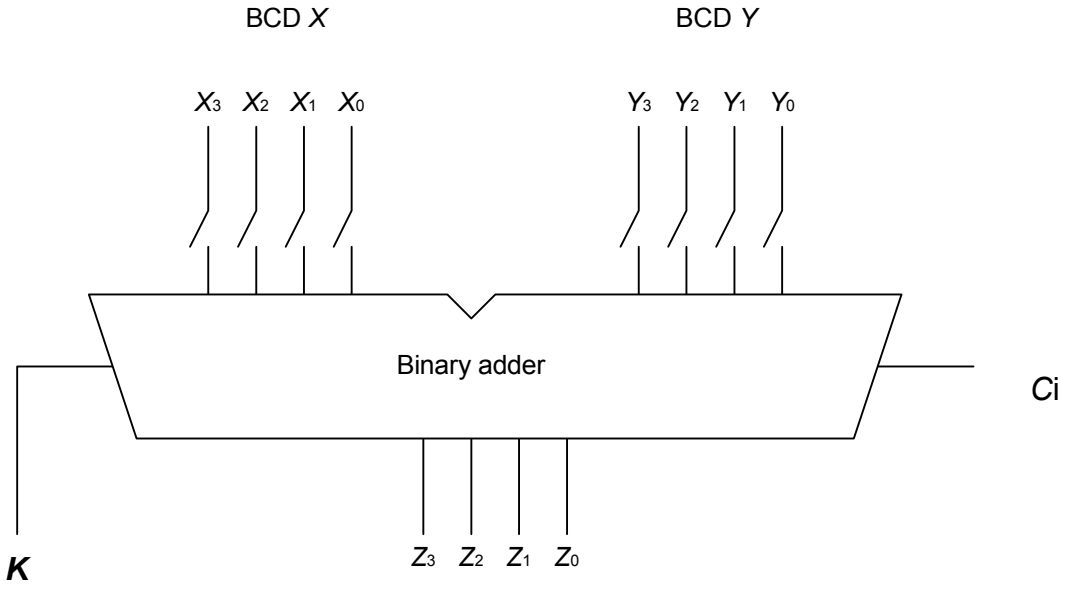
Sum	Y	Ci = 0									
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
X											
0		0	1	2	3	4	5	6	7	8	9
1		1	2	3	4	5	6	7	8	9	10
2		2	3	4	5	6	7	8	9	10	11
3		3	4	5	6	7	8	9	10	11	12
4		4	5	6	7	8	9	10	11	12	13
5		5	6	7	8	9	10	11	12	13	14
6		6	7	8	9	10	11	12	13	14	15
7		7	8	9	10	11	12	13	14	15	16
8		8	9	10	11	12	13	14	15	16	17
9		9	10	11	12	13	14	15	16	17	18

Step 1 : Problem statement : ออกแบบ 1-stage BCD adder ให้ทำการบวกตัวเลข BCD 1 digit 2 จำนวน X, Y และ Carry-in ดังรูปที่ 16



รูปที่ 15 Block diagram ของ BCD adder

Step 2 : Conceptualization : เราจะออกแบบโดยใช้วงจร 4-bit binary adder ดัง Bock diagram รูปที่ 15 มา modify เพื่อทำการบวก โดยให้มี I/P X, Y เป็นเลข BCD (0000 – 1001) และ Carry in Ci 1 บิต (0,1) และให้ผลลัพธ์ Z และ carry-out K เป็น binary ซึ่งผลลัพธ์นี้จะต้องนำมาปรับให้ได้ O/P S และ Co อยู่ในรูปของ BCD



รูปที่ 16 Block Diagram ของ binary adder ที่มี I/P เป็น BCD

Step 3 : Solution/Simplification : จากตารางที่ 5 ผลลัพธ์ที่ได้จากการบวกเลข BCD 1 digit 2 จำนวนกับ Carry-in จะมีค่าตั้งแต่ 0 (0+0+0) ถึง 19 (9+9+1) แต่ในการบวกนั้น เราจะใช้วงจร binary adder จึงทำให้ผลลัพธ์ออกมาในรูปของ binary เช่น $10_{10} = 1010_2$ เราจำเป็นต้องนำผลลัพธ์นี้มาปรับให้อยู่ในรูปของ BCD เช่น $10_{10} = (1\ \ 0000)_{BCD}$

ดังนั้น ค่าผลลัพธ์ที่เกิน range ของ BCD คือผลลัพธ์ที่มีค่า 10 – 19 จะต้องนำมาปรับให้เป็นรูปของ BCD โดยการบวกด้วย 6 ดังตัวอย่างที่ 4,5 และ ตารางที่ 6

ตัวอย่างที่ 4

5	=	0 1 0 1	
+ 3	=	0 0 1 1	
8	=	1 0 0 0	ผลลัพธ์เป็น BCD ถูกต้อง ●

ตัวอย่างที่ 5

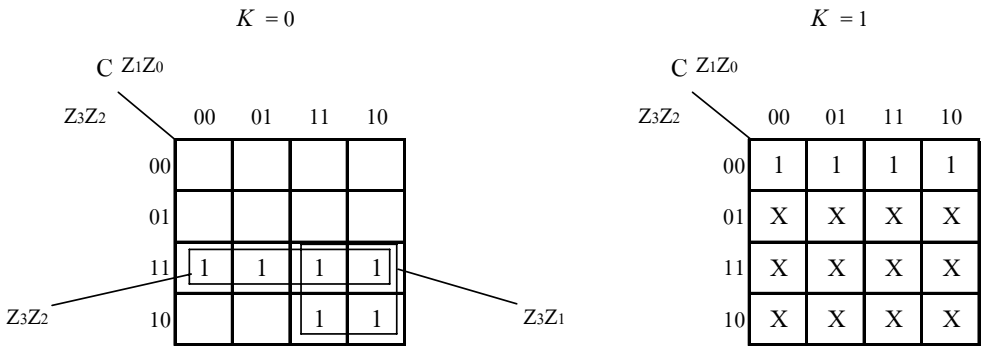
7	=	0 1 1 1	
+ 5	=	0 1 0 1	
12	=	1 1 0 0	ผลลัพธ์เป็น binary ซึ่งมากกว่า 9
		+ 0 1 1 0	บวก 6 เนื่องจากผลลัพธ์มากกว่า 9
		1 0 0 1 0	12 ได้ผลลัพธ์ BCD ที่ถูกต้อง ●

ตารางที่ 6 ตารางการแปลงผลบวก binary เป็นผลบวก BCD

Decimal	Binary Sum						BCD Sum					BCD Sum Weights	
Sum	***K***	**Z3**	**Z2**	**Z1**	**Z0**		***C***	**S3**	**S2**	**S1**	**S0**	**10**	**1**
0	0	0	0	0	0		0	0	0	0	0	0	0
1	0	0	0	0	1		0	0	0	0	1	0	1
2	0	0	0	1	0		0	0	0	1	0	0	2
3	0	0	0	1	1		0	0	0	1	1	0	3
4	0	0	1	1	0	+0=	0	0	1	0	0	0	4
5	0	0	1	1	1		0	0	1	0	1	0	5
6	0	0	1	0	0		0	0	1	1	0	0	6
7	0	0	1	0	1		0	0	1	1	1	0	7
8	0	1	0	0	0		0	1	0	0	0	0	8
9	0	1	0	0	1		0	1	0	0	1	0	9
10	0	1	0	1	0		1	0	0	0	0	1	0
11	0	1	0	1	1		1	0	0	0	1	1	1
12	0	1	1	1	0		1	0	0	1	0	1	2
13	0	1	1	1	1		1	0	0	1	1	1	3
14	0	1	1	0	0	+6=	1	0	1	0	0	1	4
15	0	1	1	0	1		1	0	1	0	1	1	5
16	1	0	0	0	0		1	0	1	1	0	1	6
17	1	0	0	0	1		1	0	1	1	1	1	7
18	1	0	0	1	0		1	1	0	0	0	1	8
19	1	0	0	1	1		1	1	0	0	1	1	9

การแปลง binary sum (Co, z3 z2 z1 z0) เป็น BCD sum (Co, S3 S2 S1 S0) จะเห็นว่าต้องปรับค่า Co เป็น 1 และบวก S ด้วย 0110 เมื่อ Z มีค่าเป็น 1010 ขึ้นไป

ฟังก์ชั่นของ Co สามารถหาได้จากตารางที่ 6 โดยใช้ K-map ดังรูปที่ 11 (K หมายถึง Co ตัวเดิม)



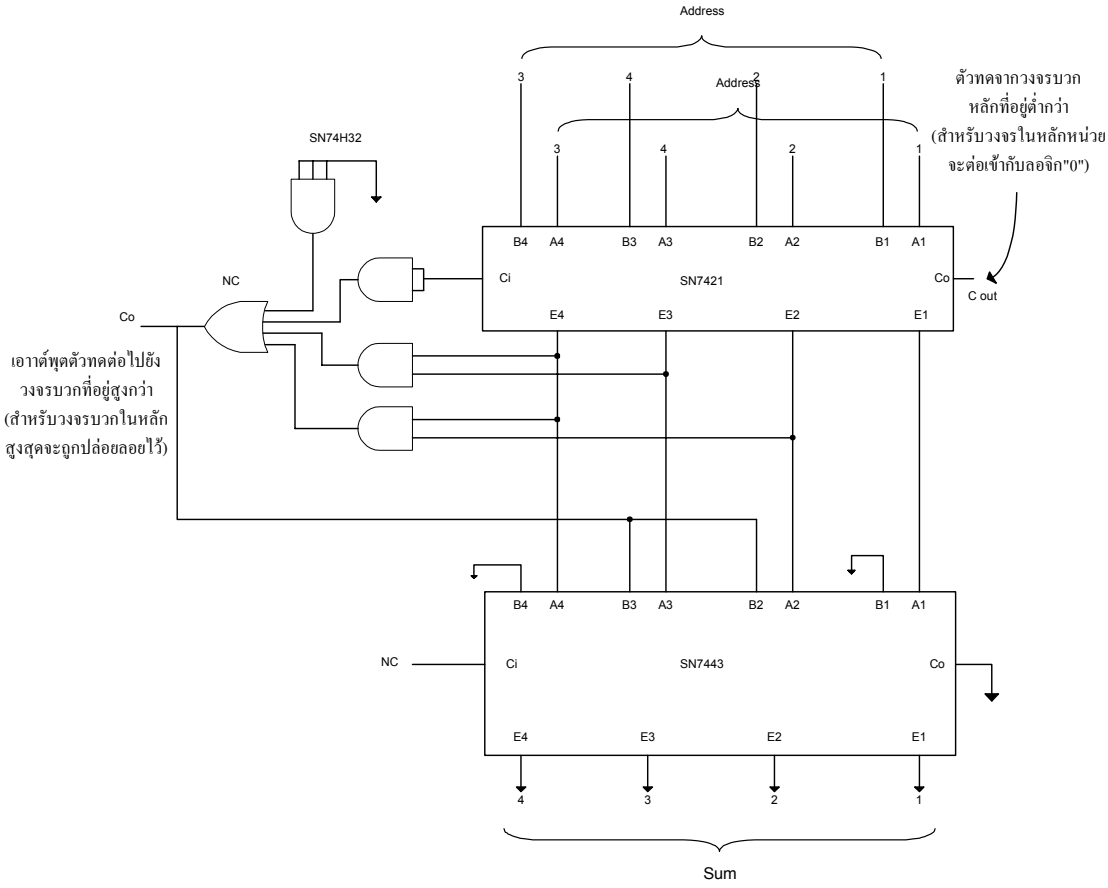
รูปที่ 17 K-map for BCD correction

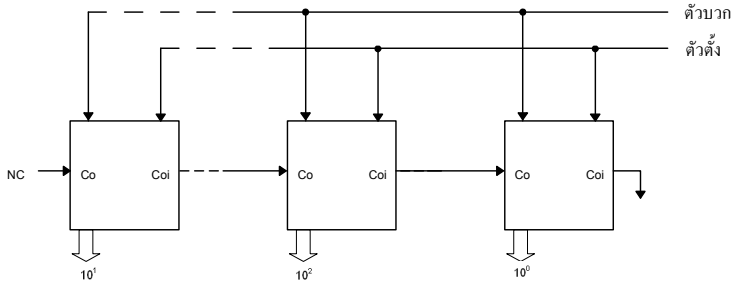
จะได้ฟังก์ชั่นของ Co ดังนี้

$$Co = K + z3z1 + z3z2$$

และเมื่อ Co = 1 เราจะบวก Z ด้วย 0110

Step 4 : Realization : วงจร 1-stage BCD adder สามารถสร้างได้จาก วงจร 4 bit binary adder ที่ใช้ทำการบวก BCD I/P เข้าด้วยกัน และ ฟังก์ชั่นของ Co ตัวใหม่ และ 4-bit binary adder อีก 1 วงจร สำหรับบวก Z เข้ากับ 0110 เมื่อ Co ตัวใหม่เท่ากับ 1 ดังรูปที่ 18





รูปที่ 18 1-stage BCD adder ที่สร้างจาก 4-bit binary adder

BCD Subtraction

การลบเลข BCD X-Y สามารถทำได้โดยการบวก X ด้วยค่า 9s complement ของ Y

$$X - Y = X + Y'$$

ค่า 9s complement ของ Y หาได้จาก 9 – Y ดังตารางที่ 7

$$Y' = 9 - Y$$

Decimal		BCD	
Y	**Y = 9s Complement**	*Y* *abcd*	**Y = 9s Complement**
0	9	0000	1001
1	8	0001	1000
2	7	0010	0111
3	6	0011	0110
4	5	0100	0101
5	4	0101	0100
6	3	0110	0011
7	2	0111	0010
8	1	1000	0001
9	0	1001	0000

ตารางที่ 7 แสดง Y และ Y' ในรูปของ Decimal และ BCD

ขั้นตอนการลบ BCD โดยวิธีการทำ 9s complement แล้วบวกโดยวงจร BCD adder แสดงให้เห็นผลลัพธ์ดังตัวอย่างที่ 6,7

<u>ตัวอย่างที่ 6</u> กรณี X> Y ผลบวกจาก BCD Adder จะเกิด Carry-out = 1 เป็น EAC แสดงว่าผลลัพธ์เป็นบวก ให้นำ EAC ไปบวกกับผลบวกที่ได้ จะได้ เอาต์พุต ที่ถูกต้อง

X = 5 = 0 1 0 1

Y = 2 = <u>0 1 1 1</u> 9s complement ของ 2

X – Y = X + Y' = 1 1 0 0 Note: X + Y' = X + (9 – Y)

= 9 + (X - Y) > 9

	+ 0 1 1 0		บวกด้วย 6
	1 0 0 1 0		ผลลัพธ์จาก BCD adder
EAC →	+ 1		
=	1 0 0 1 1	= 3	●

<u>ตัวอย่างที่ 7</u> กรณี X < Y ผลบวกจาก BCD Adder จะได้ Carry-out = 0 แสดงว่าผลลัพธ์เป็นลบ ให้ทำ 9s complement ผลลัพธ์นั้น จะได้ เอาต์พุต ที่ถูกต้อง

X	=	5	=	0 1 0 1	
Y	=	9	=	<u>0 0 0 0</u>	9s complement ของ 9
X – Y	=	X + Y'	=	0 0 1 0 1	ผลลัพธ์จาก BCD adder

Carry-out = 0 ทำ 9s complement

0 0 1 0 0=-4(Carry-outเป็น0แสดงว่าเป็นค่าลบ)●

Design of BCD Complementer Using a Binary Adder

9s complement ของเลข BCD Y สามารถหาได้โดย

1. ลบ Y ออกจาก 15 โดย invert ทุกบิตของ BCD
2. ลบออก 6 โดยนำมาบวกด้วย 10_{10} (1010_2 = 2s complement ของ 6) ซึ่งจะเกิด Carry-out $=16_{10}$
3. ลบออก 16_{10} (10000_2) โดยการตัด Carry-out ออกไป

นั่นคือ

9 – Y	=	15 – Y – 6
	=	15 – Y + 10 – 16
	=	25 – Y - 16

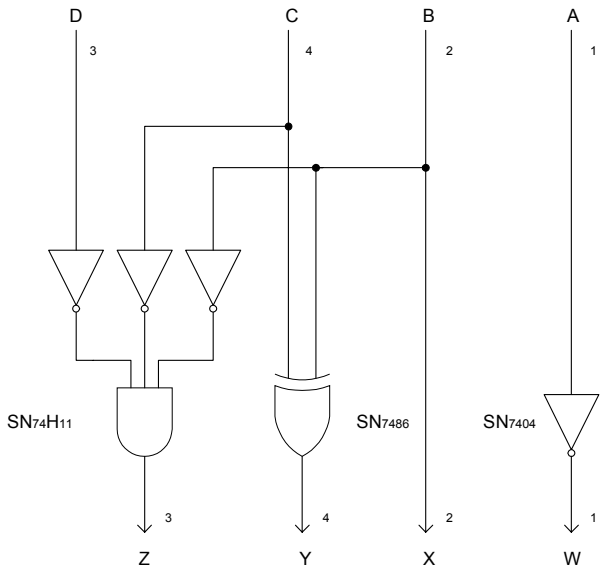
พิจารณาตัวอย่างต่อไปนี้

BCD number	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1
Invert	1 1 1 1	1 1 1 0	1 1 0 1	1 1 0 0
Add 1010	1 0 1 0	1 0 1 0	1 0 1 0	1 0 1 0
Result	1 1 0 0 1	1 1 0 0 0	1 0 1 1 1	1 0 1 1 0
Drop Carry-out	1 0 0 1	1 0 0 0	0 1 1 1	0 1 1 0
				= 9s Complement

Decimal *N*	BCD abcd *N*	Bitwise Complement *15 - N*	+	Add 1010 **10**	=	Re 01 *25 - N* *16 + N′*	Ignore carry-out = 9s Complement *N′*
0	0000	1111		1010		11001	1001
1	0001	1110		1010		11000	1000
2	0010	1101		1010		10111	0111
3	0011	1100		1010		10110	0110
4	0100	1011		1010		10101	0101
5	0101	1010		1010		10100	0100
6	0110	1001		1010		10011	0011
7	0111	1000		1010		10010	0010
8	1000	0111		1010		10001	0001
9	1001	0110		1010		10000	0000

ตารางที่ 8 การหา 9s complement ของ เลข BCD

จะสามารถสร้างวงจร BCD complementer ได้ดัง Block Diagram รูปที่ 19



รูปที่ 19 BCD 9s complementer สร้างจาก binary adder

Implement a BCD Adder/Subtracter by Combining the BCD Adder and Complementer

เราสามารถนำวงจร BCD adder รวมกับ BCD Complementer เพื่อสร้างเป็น BCD Adder /Subtracter โดยที่จะต้องมีการ Control ว่าจะทำการ บวก หรือลบ ในที่นี้จะใช้ W เป็นสัญญาณควบคุม

ทางอินพุต

- ถ้า W = 0 ทำการบวก BCD I/P Y จะ pass เข้าไปทำการบวกโดยไม่มีการเปลี่ยนค่า
- ถ้า W = 1 ทำการลบ BCD I/P Y จะถูกทำ 9s complement เป็น Y' ไปทำการบวกกับ X

Carry-in

- ถ้า W = 0 ทำการบวก Ci ใช้เป็น carry-in
- ถ้า W = 1 ทำการลบ Ci รับค่าจาก Carry-out ในกรณีที่ผลลัพธ์เป็นค่าบวก

Carry-out

- ถ้า W = 0 ทำการบวก Co ใช้เป็นค่า 10^1
- ถ้า W = 1 ทำการลบ Co ส่งไปควบคุมการทำ 9s complement ผลลัพธ์ และใช้เป็นบิตเครื่องหมาย

  Co = 1 ผลลัพธ์เป็นบวก

  Co = 0 ผลลัพธ์เป็นลบ

ทาง เอาต์พุต

- ถ้า W = 0 ผลลัพธ์จาก BCD adder จะ pass ไปยัง เอาต์พุต โดยไม่มีการเปลี่ยนแปลงค่า
- ถ้า W = 1

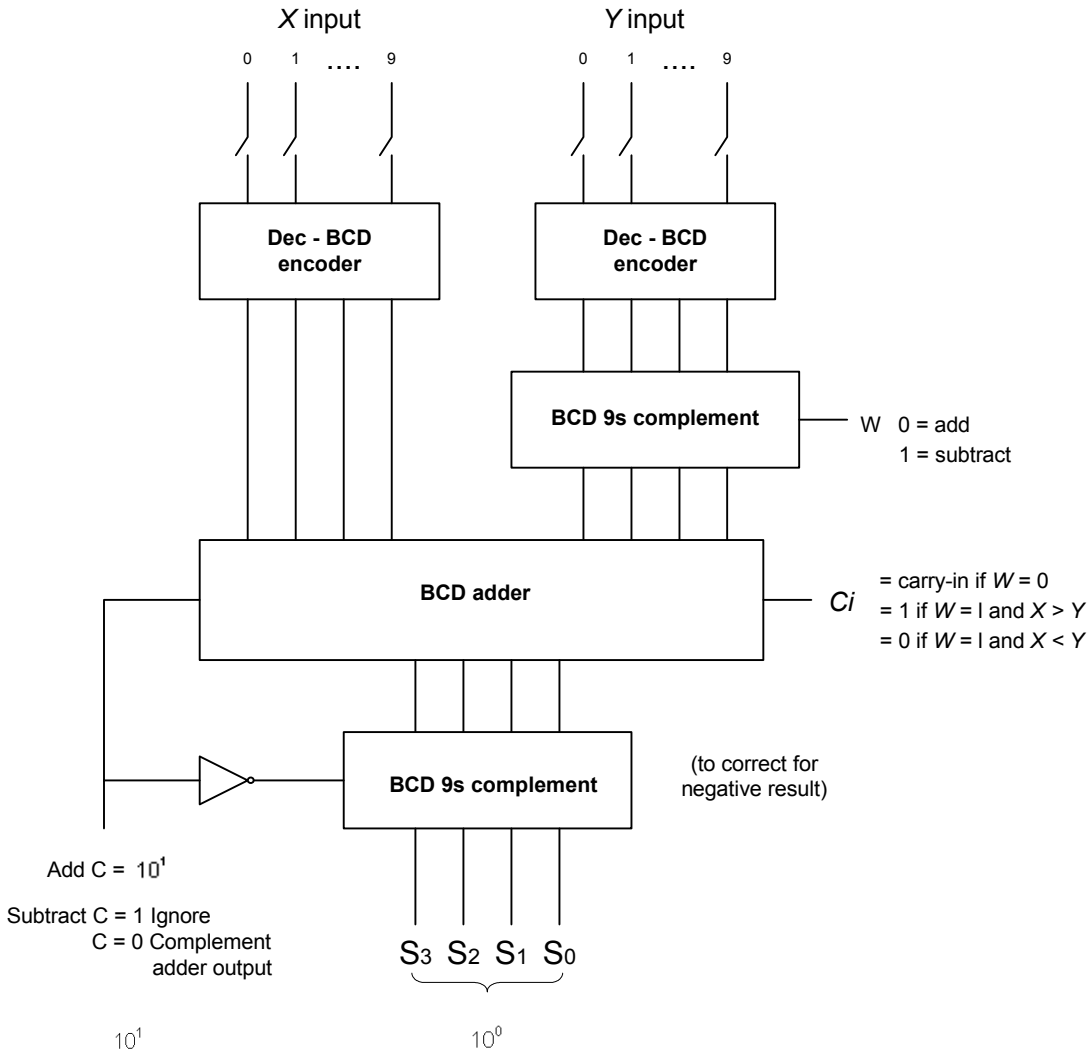
  Co = 1 ผลลัพธ์จาก BCD adder จะ pass ไปยัง เอาต์พุต โดยไม่มีการเปลี่ยนแปลงค่า

  ผลลัพธ์เป็นค่าบวก

  Co = 0 ผลลัพธ์จาก BCD adder จะ ถูก ทำ 9s complement ได้ผลที่ถูกต้องทางเอาต์พุต

  ผลลัพธ์เป็นค่าลบ

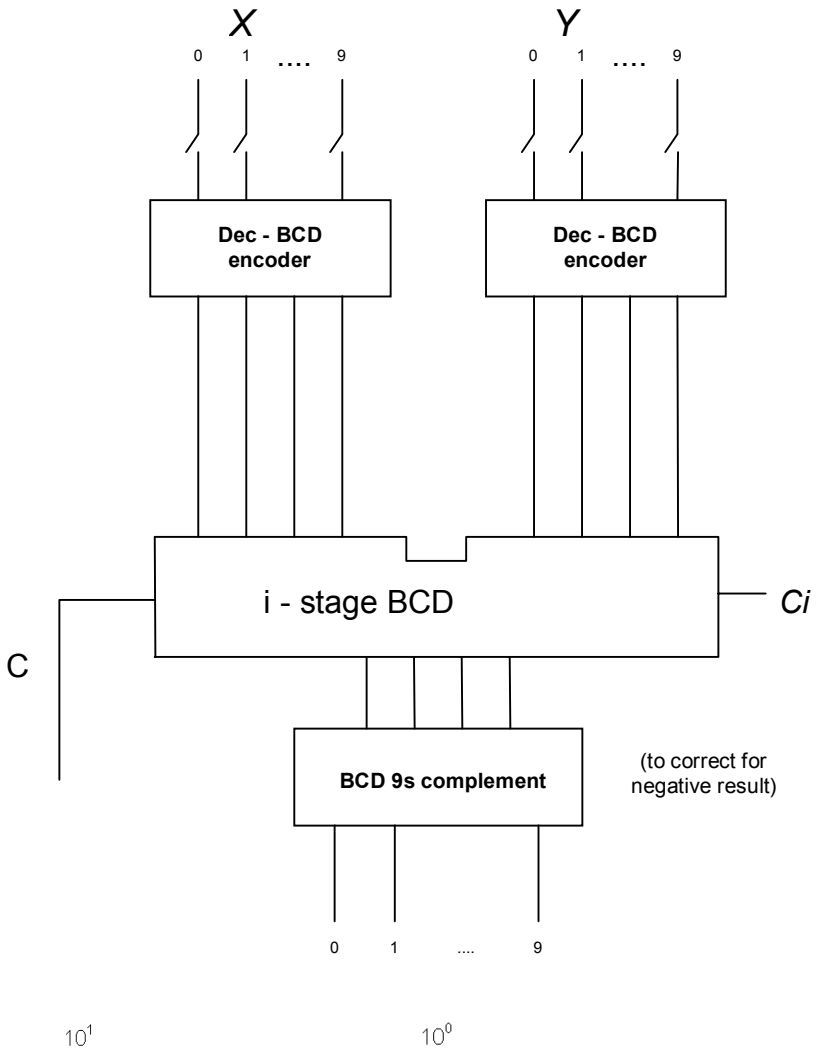
Block Diagram ของวงจร แสดงให้เห็นดังรูปที่ 20



รูปที่ 20 Block diagram ของ BCD adder/subtracter

Design of BCD Arithmetic Unit

ในการบวก/ลบตัวเลข Decimal สองจำนวน ตัวเลข decimal นั้นจะต้องถูกเข้ารหัสให้เป็นรหัส BCD เพื่อทำการบวกด้วย BCD adder/subtracter และที่ เอาต์พุต จะต้องทำการถอดรหัสจาก BCD ออกมาเป็น Decimal ดังรูปที่ 21



รูปที่ 21 Block diagram ของ 1-stage BCD arithmetic unit