การออกแบบวงจรเกี่ยวกับคณิตศาสตร์

การบวกลบเลขที่ไม่ใช่ฐานสิบ

การบวกลบเลขฐานสอง คิดเหมือนฐานสิบแต่ใช้หลักการดังนี้

การบวก	การลบ
0 + 0 = 0	0 - 0 = 0
0 + 1 = 1	0 - 1 = 1 ตัวยืมเท่ากับ 1
1 + 0 = 1	1 - 0 = 1
1 + 1 = 0 ตัวทดเท่ากับ 1	1 - 1 = 0

ตัวอย่างที่ 1.5 จงบวกเลขฐานสองต่อไปนี้

ก) 10111110 + 10001101

ข) 101101 + 00101100

ตัวอย่างที่ 1.6 จงลบเลขฐานสองต่อไปนี้

ก) 11100101 – 00101110

ข) 11010010 – 01101101

ตัวอย่างที่ 1.7 จงหาผลบวกของเลขฐานสิบหก $19B9_{16} + C7E6_{16}$

การแทนจำนวนลบด้วยตัวเลขฐานสอง

การแทนจำนวนลบด้วยเลขฐานสองสามารถทำได้ 3 วิธี คือ

- วิธีที่ 1 Signed-Magnitude
- วิธีที่ 2 One's Complement
- วิธีที่ 3 Two's Complement

วิธีที่ 1 SIGNED - MAGNITUDE

วิธีนี้เป็นวิธีที่ง่ายเพียงเติมบิตเครื่องหมาย ไว้ทางซ้ายของจำนวนโดย ให้บิตเครื่องหมายเป็น '0' ถ้าเป็นจำนวนบวก และบิตเครื่องหมายเป็น '1' ถ้าเป็นจำนวนลบ

เครื่องหมาย บวก

เช่น $+85_{10} = 0\ 1010101_2$

$-85_{10} = 1\ 1010101_2$

$+127_{10} = 0\ 1111111_2$

$-127_{10} = 1\ 1111111_2$

$+0_{10} = 0\ 0000000_2$

- $-0_{10} = 1\ 0000000_2$

1 0 1 0 1 0 1

1x2^0 = 1

1x2^2 = 4

1x2^4 = 16

1x2^6 = 64

85

จำนวนคอมพลีเมนท์ (COMPLEMENT NUMBER)

57 -

23

34

57 +

77

---

1 34

ดังนั้นสามารถกล่าวได้ว่า 77 เป็นค่า Complementของ 23 หรือ ในทางตรงกันข้าม 23 เป็นค่า Complementของ 77

จำนวนคอมพลีเมนท์ของเลขใดๆมี 2 แบบคือ

1. คอมพลีเมนต์ฐาน (Radix – Complement)
2. คอมพลีเมนต์ฐานลบหนึ่ง (Diminished Radix – Complement)

คอมพลีเมนต์ฐาน (RADIX – COMPLEMENT)

ถ้ากำหนดให้ D เป็นจำนวนเลขฐานใดๆ ที่มีขนาด n หลัก

ค่า Radix - Complement ของ D หาได้จาก

Radix - Complement ของ D $= r^n - D$ (1.1)

$= (r^n - 1) - D + 1$ (1.2)

โดย

r คือ ฐานของเลขจำนวน D เช่นถ้าเป็นฐานสิบ r = 10 และจะเรียกว่า 10's Complement ถ้าเป็นฐานสอง r = 2 และจะเรียกว่า 2's Complement

n คือ จำนวนหลักของ D

ตัวอย่างที่1.8 จงหาค่า Radix - Complement ของ 1879_{10}

ในที่นี้ D = 1879 r = 10 และ n = 4

ดังนัน 10's Complement ของ 1879 จะหาได้จาก

10's Complement ของ 1879 $= (10^4 - 1) - 1879_{10} + 1$

$= (10000 - 1) - 1879 + 1$

$= (9999_{10} - 1849_{10}) + 1$

$= 8151_{10}$

สรุป วิธีหา Radix - Complement ของเลขฐานสิบ ให้ใช้แต่ละหลักไปลบ 9 ได้เท่าไรแล้วบวก 1

ตัวอย่างที่ 1.9 จงหา 2's Complement ของ 1010_2

2's complement ของ 1010 $= (2^4 - 1) - 1010_2 + 1$

$= (10000_2 - 1) - 1010_2 + 1$

$= 1111_2 - 1010_2 + 1$

$= 0101_2 + 1$

$= 0110_2$

สรุป วิธีหา Radix - Complement ของเลขฐานสอง ให้กลับบิตแต่ละหลักเป็นตรงกันข้าม ได้เท่าไรแล้วบวก 1

คอมพลีเมนต์ฐานลบหนึ่ง (DIMINISHED RADIX – COMPLEMENT)

Radix - Complement ของ D $= r^n - D$ (1.1)

$= (r^n - 1) - D + 1$ (1.2)

จากสมการ Radix - Complement ที่ (1.2) ถ้าตัด +1 ด้านขวาออกจะได้เป็น

Diminished Radix - Complement $= (r^n - 1) - D$ (1.3)

Diminished Radix - Complement นี้ถ้าใช้กับเลขฐานสิบจะเรียกว่า "9's Complement"

และถ้าใช้กับเลขฐานสองจะเรียกว่า "1's Complement" ซึ่งมีวิธีจำง่ายๆคือ

ถ้าเป็นฐานสิบ เอาแต่ละหลักไปลบออกจาก 9

ถ้าเป็นฐานสอง กลับบิตทุกบิตเป็นตรงกันข้าม (จาก 1 เป็น 0 และจาก 0 เป็น 1)

การเขียนจำนวนเลขฐานสิบลบด้วยวิธี 9's Complement และ 10's – Complement ขนาด 4 หลัก

D	9's Complement	10's Complement
2067	7932	7933
100	9899	9900
7	9992	9993
0	9999	10000 (= 0000 = 0)

การเขียนจำนวนลบด้วยวิธี Two's – Complement ขนาด 8 บิต

+ 17 = 0 0010001 ทำ 2's complement 1 1101111 = -17

+ 127 = 0 1111111 1 0000001 = -127

การเขียนจำนวนลบด้วยวิธี One's -Complement

+ 17 = 0 0010001 ทำ 1's complement 1 1101110 = -17

+ 127 = 0 1111111 1 0000000 = -127

1.6 การบวกและลบเลขทูคอมพลีเมนต์

กฎการบวก (Addition Rules)

จำนวนใดๆ ถ้าแทนด้วยเลขฐานสองขนาด n บิตเมื่อบวกกันแล้วมีจำนวนบิตมากกว่า n ให้ตัดบิตที่เกินทิ้ง เช่น การบวกเลขขนาด 4 บิต ต่อไปนี้

ก) จงบวก 0011 กับ 0100

+3	0 011
+ +4	+ 0 100
+7	0 111

ข) จงบวก 1110 กับ 1010

-2	1 110
+ -6	1 010
-8	1 1 000

บิตนี้เกิน 4 บิต ตัดทิ้ง

วิธีแก้ไข คือ ต้องเพิ่มจำนวนบิต เช่น เปลี่ยนเป็นเลขฐานสองขนาด 5 บิต

+3		0 0011	
++6	+	0 0110	
+9		0 1001	ไม่เกิดโอเวอร์โฟลว์

กฎการลบ (Subtraction Rules)

ให้เปลี่ยนจากการลบเป็นการบวก เช่น A - B = A + (-B)

เช่น

	+4	0 100	0 100
-	+3	- 0 011	+ 1 101
	+1	ตัดทิ้ง ⟶	1 0 001

การบวกและลบเลขวันคอมพลีเมนต์

ทำเหมือนกับวิธี การบวกและการลบเลขแบบ Two's - Complement

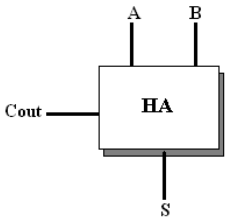
เพียงแต่ถ้ามีการทดเกินบิต ให้นำบิตนั้นกลับไปรวมอีกครั้ง เช่น

	+4		0 100		+6		0 110
+	-7	+	1 000	+	-3	+	1 100
	-3		1 100		+3	1	0 010
							1

มีตัวทดเกิน ให้นำกลับไปรวมเป็นคำตอบ 0 011

วงจรบวกเลขไบนารี (BINARY ADDITION CIRCUIT)

Half Adder



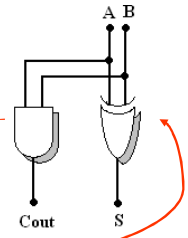
ตัวตั้ง A

ตัวบวก B

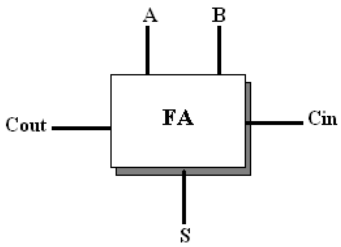
ผลบวก S

ตัวทดออก Cout

A	B	Cout	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



FULL ADDER



ตัวตั้ง A

ตัวบวก B

ผลบวก S

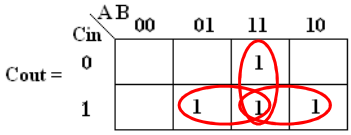
ตัวทดออก Cout

ตัวทดเข้า Cin

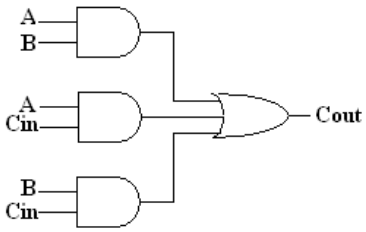
FULL ADDER

Cin	A	B	Cout	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

ตารางการทำงาน



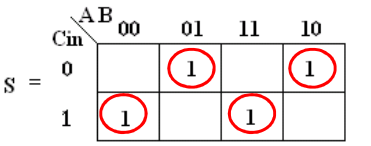
$$Cout = A.B + A.Cin + B.Cin$$



FULL ADDER

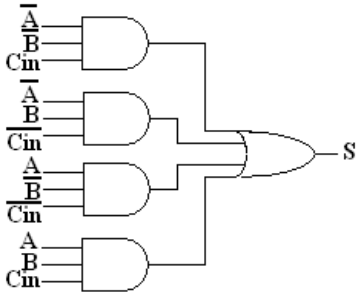
Cin	A	B	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

ตารางการทำงาน

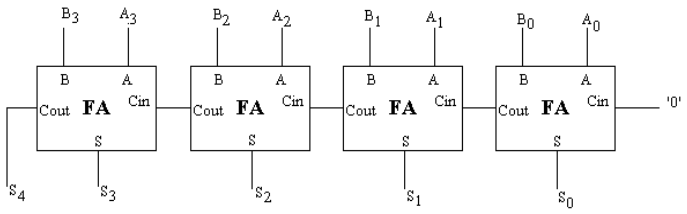


$$S = \bar{A}.\bar{B}.Cin + \bar{A}.B.\overline{Cin} + A.\bar{B}.\overline{Cin} + A.B.Cin$$

$$S = A \oplus B \oplus Cin$$



วงจรบวกแบบริปเปิล (RIPPLE ADDER)

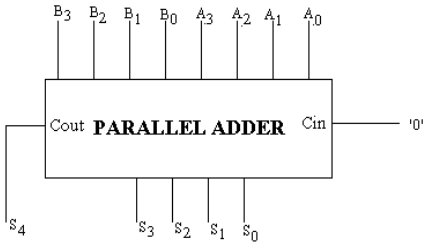


S = A + B

$A_3A_2A_1A_0$ +

$B_3B_2B_1B_0$

$S_4S_3S_2S_1S_0$



แผนผังบล็อกวงจรบวกแบบริปเปิลขนาด 4 บิต

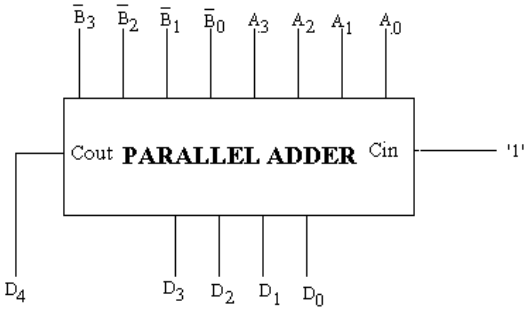
วงจรลบขนาด 4 บิต (4-BITS SUBTRACTORS)

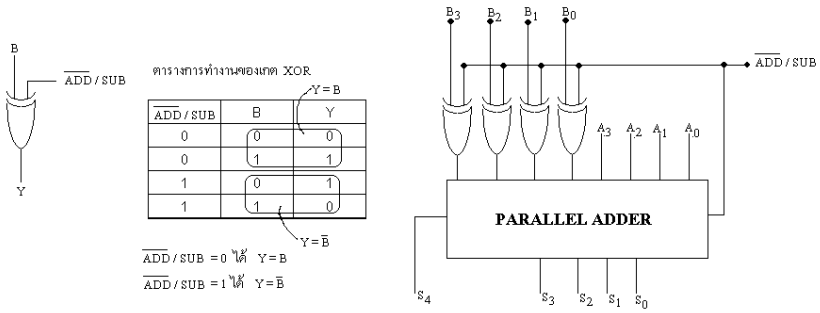
จาก D = A – B

D = A + (- B)

เลขลบคือค่า 2’ s complement ของเลขบวก ดังนั้น

D = A + (2’s complement ของ B) D = A + (1’scomplement ของ B + 1)

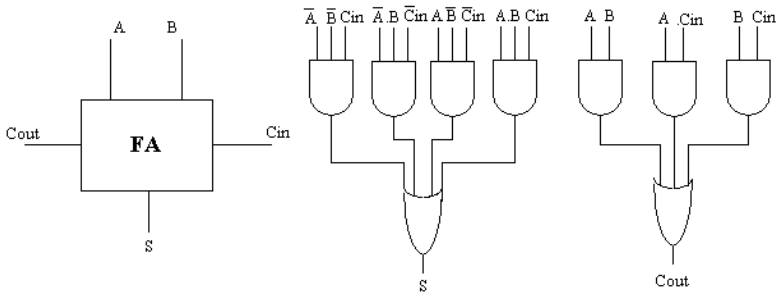




ถ้าให้ สัญญาณ $\overline{ADD/SUB} = 0$ จะได้สัญญาณ B และ $S = A + B + 0 = A + B$

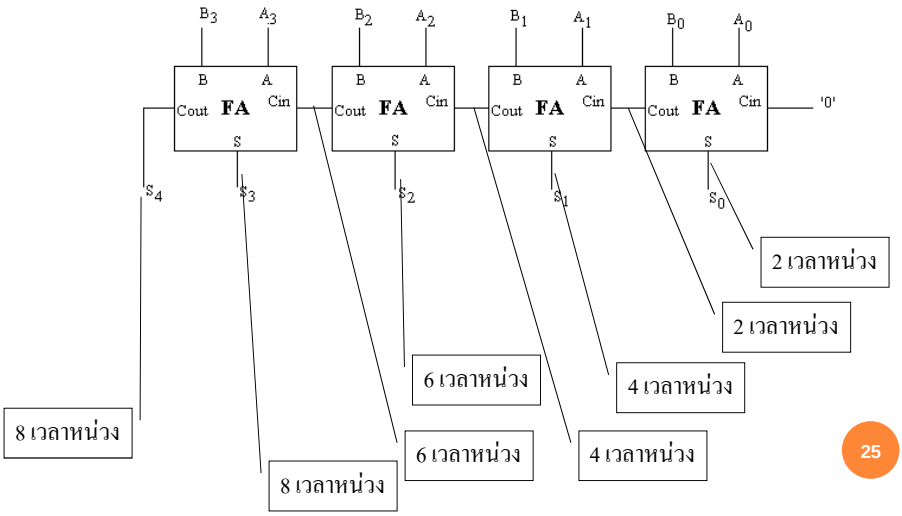
ถ้าให้ สัญญาณ $\overline{ADD/SUB} = 1$ จะได้สัญญาณ $\overline{B}$ และ $S = A + \overline{B} + 1 = A - B$

วงจรบวกแบบดูตัวทดล่วงหน้า (CARRY LOOKAHEAD ADDERS)



วงจร Full Adder ผลบวก S มีเวลาหน่วง เท่ากับ 2 ระดับ และตัวทดออก Cout ของวงจรมีเวลาหน่วงเท่ากับ 2 ระดับ

วงจรบวกแบบริปเปิลขนาด 4 บิต



วงจรบวกแบบ Carry Lookahead Adders ได้ใช้วิธีคำนวณหาค่าตัวทดออกโดยตรง ไม่ต้องรอผลของตัวทดอื่นๆ

จากสมการตัวทดออกของวงจรบวกแบบฟูล

Cout = AB + Acin + BCin

Cout = AB + (A+B)Cin

ถ้าเขียนเป็นสมการของตัวทดออกแต่ละหลักโดยให้

Ci+1 = Cout ตัวทดออกของบิตที่ i

Ci = Cin ตัวทดเข้าของบิตที่ i

Pi = Ai + Bi การ OR ระหว่างตัวตั้งกับตัวบวกบิตที่ i

Gi = Ai.Bi การ AND ระหว่างตัวตั้งกับตัวบวกบิตที่ i

ดังนั้น

Ci+1 = Gi + Pi.Ci

สมการของตัวทดออกของบิตที่ 0 ถึง บิตที่ 3 เขียนได้เป็น

$$C1 = G0 + P0.C0$$

$$C2 = G1 + P1.C1$$
$$= G1 + P1.(G0 + P0.C0)$$
$$= G1 + P1.G0 + P1.P0.C0$$

$$C3 = G2 + P2.C2$$
$$= G2 + P2.(G1 + P1.G0 + P1.P0.C0)$$
$$= G2 + P2.G1 + P2.P1.G0 + P2.P1.P0.C0$$

$$C4 = G3 + P3.C3$$
$$= G3 + P3.(G2 + P2.G1 + P2.P1.G0 + P2.P1.P0.C0)$$
$$= G3 + P3.G2 + P3.P2.G1 + P3.P2.P1.G0 + P3.P2.P1.P0.C0$$

$\overline{Gi}$

และในส่วนของผลบวก Si สามารถหาได้จาก

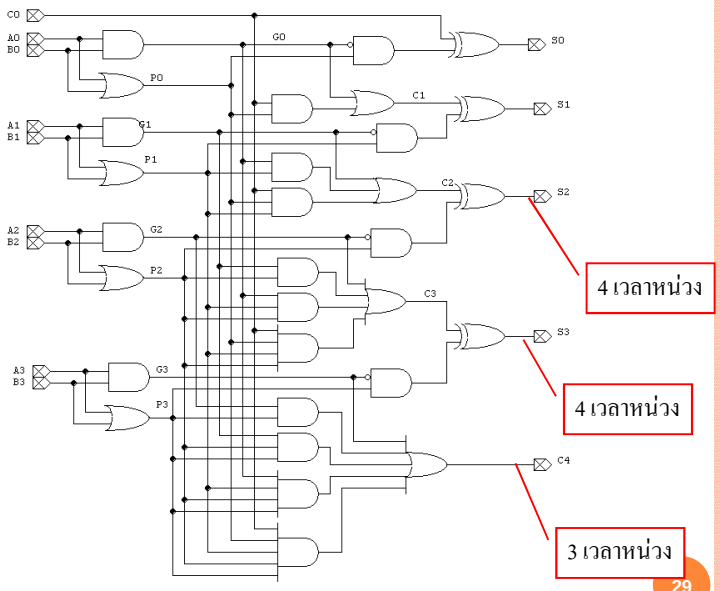
$$Si = Ai \oplus Bi \oplus Ci$$

$$= (A \oplus B) \oplus Ci$$

$$= (\overline{A}B + A\overline{B}) \oplus Ci$$

$$= \overline{(A.B.\ (A+B))} \oplus Ci$$

$$Si = (\overline{Gi.Pi}) \oplus Ci$$



29

วงจรบวกแบบ แบบแคร์รีลุคอะเฮด

โอเวอร์โฟลว์ (OVERFLOW)

ถ้าการคำนวณใดๆ ให้ผลลัพธ์มีค่าเกินกว่าจำนวนที่เลขฐานสองจะแทนได้ จะเกิดการผิดพลาดของการคำนวณขึ้นเรียกว่าเกิด "โอเวอร์โฟลว์" สาเหตุเนื่องจากใช้จำนวนบิตน้อยเกินไป เช่น ถ้าใช้เลขขนาด 4 บิต ซึ่งสามารถแทนค่าได้ตั้งแต่ +7 ถึง -8 ถ้าค่าเกินกว่านี้จะเกิด Overflow เช่น

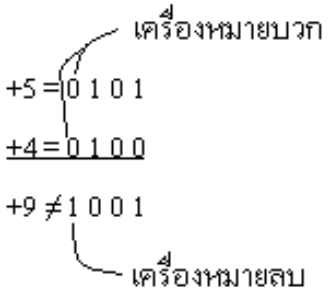
	+3		0 011	เครื่องหมายบวก
+	+6	+	0 110	เครื่องหมายบวก
	+9		1 001	เครื่องหมายลบ

1 001 คือ -7 ซึ่งไม่ถูกต้อง

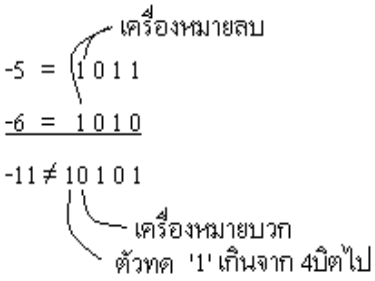
30

วงจรตรวจสอบโอเวอร์โฟลว์ (OVERFLOW)

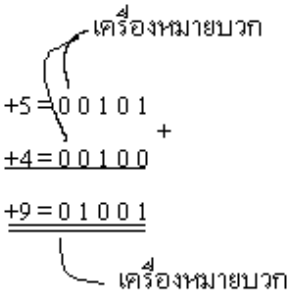
จงบวก +5 กับ +4 โดยใช้ข้อมูล 4 บิต

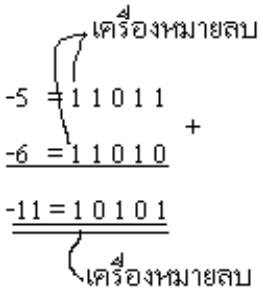


จงบวก –5 กับ –6 โดยใช้ 4 บิต

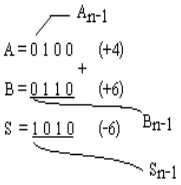


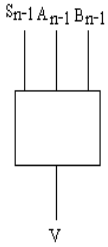
เปลี่ยนเลขฐานสองขนาด 5 บิต จะได้คำตอบที่ถูกต้องดังนี้



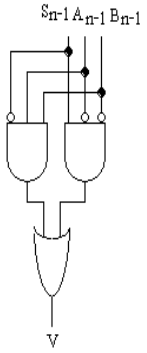


วงจรตรวจสอบโอเวอร์โฟลว์แบบแรกจะพิจารณาจากบิตเครื่องหมาย เมื่อให้ An-1 เป็นบิตเครื่องหมายตัวตั้ง Bn-1 เป็นบิตเครื่องหมายตัวบวก Sn-1 เป็นบิตเครื่องหมายผลรวม และ V สัญญาณโอเวอร์โฟลว์ โดยกำหนดให้มีค่าเป็น 1 เมื่อเกิดโอเวอร์โฟลว์ โดยจะเกิดเมื่อเครื่องหมายตัวตั้งและเครื่องหมายตัวบวกเหมือนกันแต่เครื่องหมายของผลรวมต่างจากเครื่องหมายของตัวตั้งหรือตัวบวก สามารถออกแบบวงจรได้ดังนี้

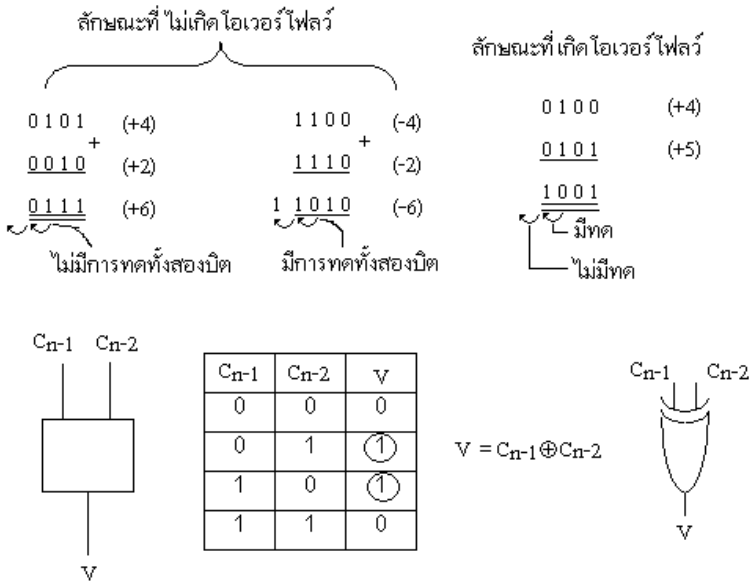




A_{n-1}	B_{n-1}	S_{n-1}	V
0	0	0	0
0	0	1	(1)
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	(1)
1	1	1	0



วงจรตรวจสอบโอเวอร์โฟลว์แบบที่สอง พิจารณาจากบิตตัวทด ถ้าการทดระหว่างบิตสูงสุดของขนาด (n-2) ไปยังบิตเครื่องหมาย และการทดของบิตเครื่องหมายเกิดขึ้นไม่พร้อมกัน จะเกิดโอเวอร์โฟลว์ ถ้ากำหนดให้ Cn-1 เป็นตัวทดจากบิตเครื่องหมาย Cn-2 เป็นตัวทดจากบิตสูงสุดของขนาด และ V เป็นสัญญาณโอเวอร์โฟลว์ ออกแบบวงจรได้ดังนี้



1.6 BINARY MULTIPLICATION

- วิธีคูณตามปกติ
- วิธีเลื่อนและบวก(Shift - and - add)

วงจรคูณเลขฐานสอง ขนาด 1 บิต x 1 บิต

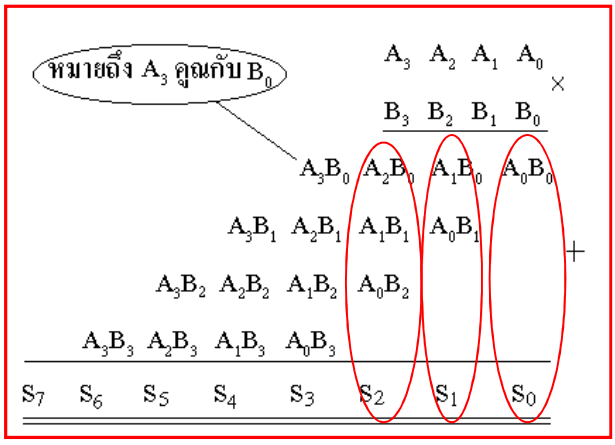
ตารางการทำงานของ Y = A คูณ B

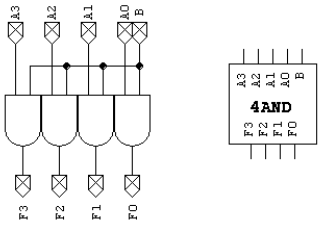
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

$Y = A.B$

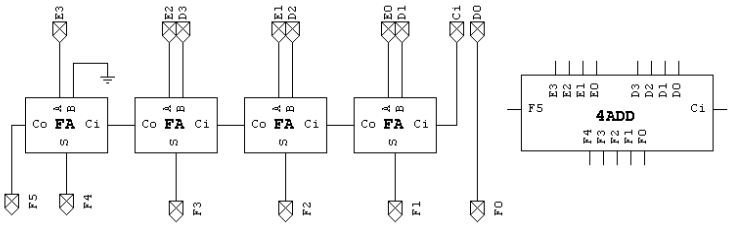
วงจรคูณเลขฐานสอง

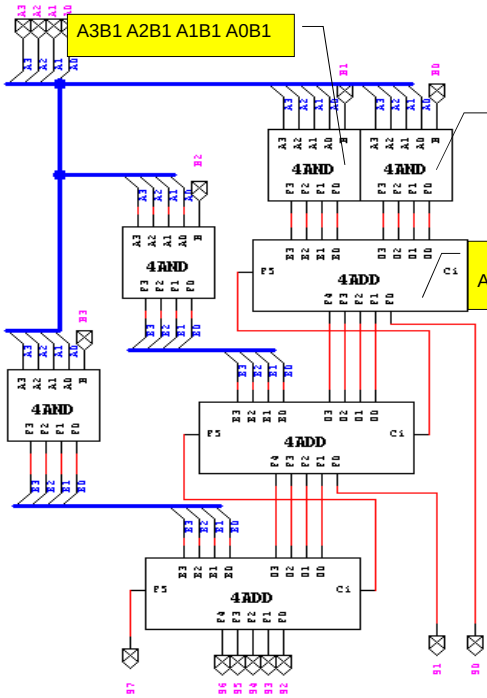
ให้ A3 A2 A1 A0 เป็นตัวตั้ง B3 B2 B1 B0 เป็นตัวคูณ





วงจรคูณ1 บิต เช่น B0x3 B0xA2 B0xA1 B0xA1





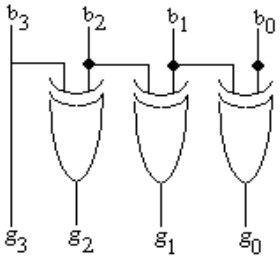
BINARY DIVISION

- วิธีหารตามปกติ
- วิธีเลื่อนและลบ(Shift - and - sub)

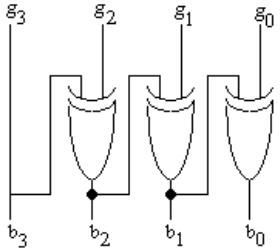
Binary	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

วงจรแปลงรหัส

Binary to Gray



Gray to Binary



วงจรเปรียบเทียบเลขฐานสอง (BINARY COMPARATOR)

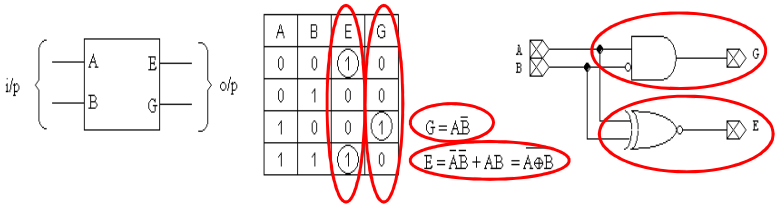
วงจรเปรยบเทียบข้อมูลขนาด 1 บิต

กำหนดให้ A เป็นตัวตั้ง และ B เป็นตัวเปรียบเทียบ

ถ้า A = B ให้ E = 1 G = 0

ถ้า A > B ให้ E = 0 G = 1

ถ้า A < B ให้ E = 0 G = 0



A	B	E	G
0	0	(1)	0
0	1	0	0
1	0	0	(1)
1	1	(1)	0

$G = A\bar{B}$

$E = \bar{A}\bar{B} + AB = \overline{A \oplus B}$

วงจรเปรียบเทียบข้อมูลขนาด 2 บิต

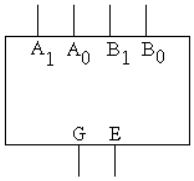
กำหนดให้ A1 A0 เป็นตัวตั้ง และ B1 B0 เป็นตัวเปรียบเทียบ

ถ้า A1A0 = B1B0 ให้ E = 1 G = 0

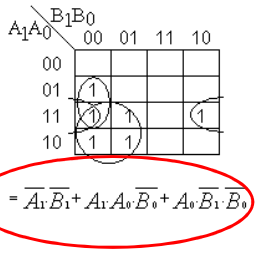
ถ้า A1A0 > B1B0 ให้ E = 0 G = 1

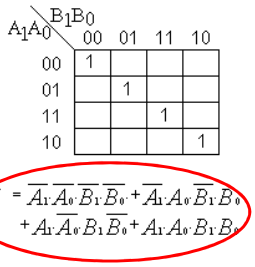
ถ้า A1A0 < B1B0 ให้ E = 0 G = 0

วงจรเปรียบเทียบข้อมูลขนาด 2 บิต

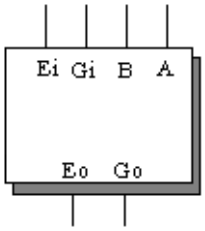


A_1	A_0	B_1	B_0	E	G
0	0	0	0	1	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	0	0
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	0





วงจรเปรียบเทียบข้อมูลขนาด 1 บิตแบบมีบิตควบคุม



กำหนดให้

A เป็นบิตตัวตั้ง

B เป็นบิตตัวเปรียบเทียบ

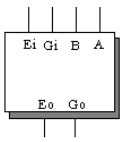
Ei เป็นบิตควบคุม ถ้าบิตที่มีนัยสำคัญสูงกว่า (บิตซ้ายมือ) มีค่าเท่ากัน จะป้อน 1 ให้แก่บิตนี้

Gi เป็นบิตควบคุม ถ้าบิตที่มีนัยสำคัญสูงกว่า (บิตซ้ายมือ) A มากกว่า B จะป้อน 1 ให้แก่บิตนี้

Eo เป็นบิต เท่ากัน ถ้า Ei = 1 Gi = 0 และ A = B จะให้ Eo = 1

Go เป็นบิตมากกว่า ถ้า Gi = 1 จะได้ Go = 1 หรือ Ei = 0 Gi = 0 แต่ จะได้ Go = 0

วงจรเปรียบเทียบข้อมูลขนาด 1 บิตแบบมีบิตควบคุม



Gi	Ei	A	B	Go	Eo
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	1
0	1	0	1	0	0
0	1	1	0	1	0
0	1	1	1	0	1
1	x	0	0	1	0
1	x	0	1	1	0
1	x	1	0	1	0
1	x	1	1	1	0

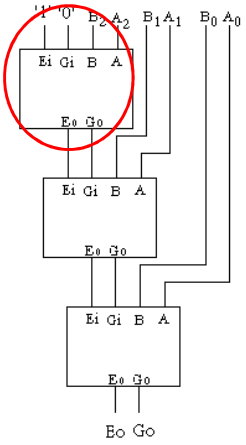
GiEi \ AB	00	01	11	10
00				
01				1
11	1	1	1	1
10	1	1	1	1

$$G_o = G_i + E_i A \cdot \overline{B}$$

GiEi \ AB	00	01	11	10
00				
01	1		1	
11				
10				

$$E_o = \overline{G}_i E_i \overline{A} \cdot \overline{B} + \overline{G}_i E_i A \cdot B$$

วงจรเปรียบเทียบข้อมูลขนาด 3 บิต ใช้วงจรเปรียบเทียบข้อมูล 1 บิตต่อเรียงกัน



หัวข้ออื่นๆ

- Even parity generaror
- Parity Checking
- Hazard

พาริตีบิต (PARITY BIT)

- พาริตีบิต หรือ บิตภาวะคู่หรือคี่ [1] (อังกฤษ: parity bit) หรืออาจเรียกเพียงแค่ พาริตี หมายถึงบิตที่เพิ่มเข้าไปในข้อมูล โดยไม่จำเป็นว่าจะต้องนำไปต่อท้ายหรือขึ้นต้น เพื่อทำให้แน่ใจว่าบิตที่เป็นค่า 1 ในข้อมูลมีจำนวนเป็นเลขคู่หรือเลขคี่ การใช้พาริตีบิตเป็นวิธีที่ง่ายอย่างหนึ่งในการตรวจจับและแก้ไขความผิดพลาด
- พาริตีมีสองชนิดคือ พาริตีคู่ (even parity) กับ พาริตีคี่ (odd parity)
- พาริตีคู่ หมายถึงจำนวนของเลข 1 ในข้อมูลมีเป็นจำนวนคู่
- พาริตีคี่ หมายถึงจำนวนของเลข 1 ในข้อมูลเป็นจำนวนคี่

วงจรสร้างบิตพาริตีคู่ ของข้อมูลขนาด 4 บิต

- แพริตีบิตคู่ จะมีค่าเป็น 1 เมื่อจำนวนของเลข 1 ในข้อมูลเป็นจำนวนคี่ (ซึ่งจะทำให้จำนวนเลข 1 ทั้งหมดเป็นจำนวนคู่ เมื่อรวมกับบิตนี้)

B3	B2	B1	B0	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

$$Y = B3 \oplus B2 \oplus B1 \oplus B0$$

แผนผังลอจิก