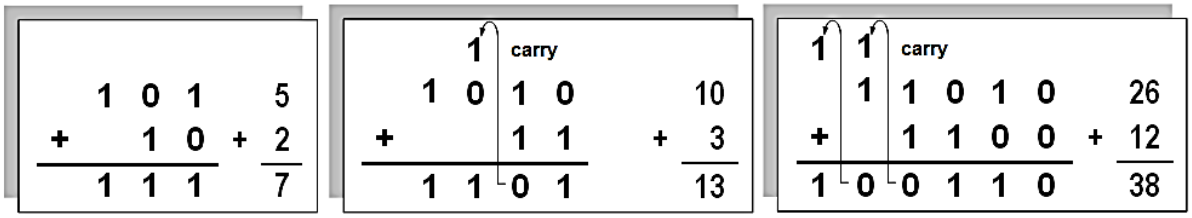
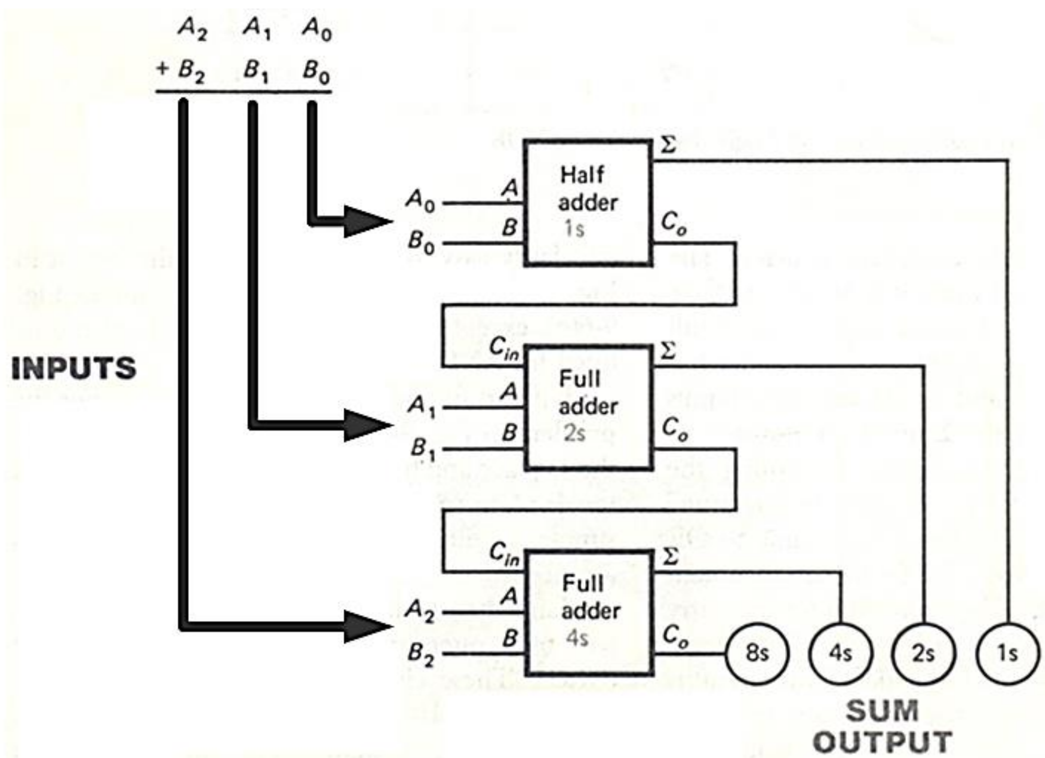
3-BIT PARALLEL ADDERS





BINARY SUBTRACTION

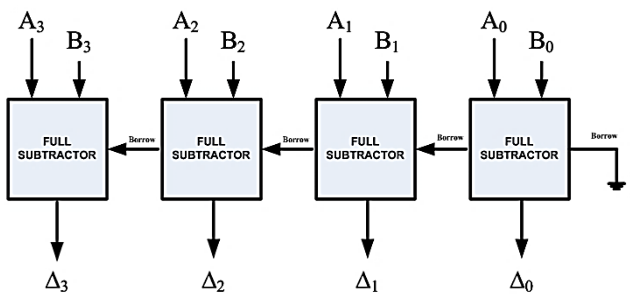
$$\begin{array}{rccccccl} & 32s & 16s & 8s & 4s & 2s & 1s & \\ & & 1 & & & & & \\ & & 10 & 10 & 0 & 10 & & \\ & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{1} & \cancel{0} & 1 & A \\ - & & & 1 & 0 & 1 & 0 & -B \\ \hline & & 1 & 1 & 0 & 1 & 1 & Di \end{array}$$

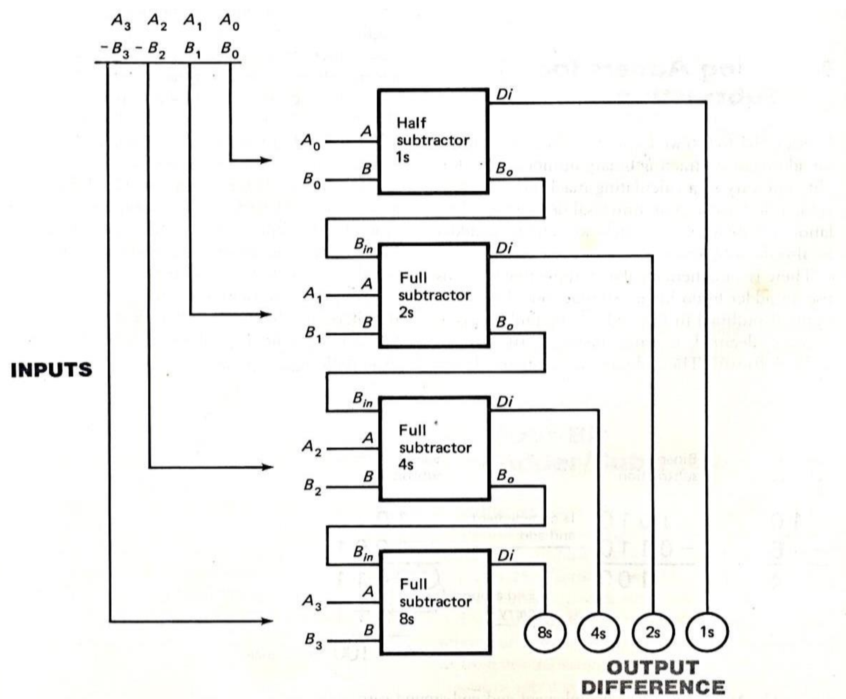
Subtract the binary number 00111 from 10101 and show the equivalent decimal subtraction.

$$\begin{array}{lr} \;\;111 & \\ \cancel{1}0\cancel{1}01 & 21 \\ \underline{00111} & \underline{\;\;7} \\ 01110 & =14 \end{array}$$

$$\begin{array}{rrrrrr} & 1 & 0 & 1 & & 5 \\ - & & 1 & 0 & - & 2 \\ \hline & 0 & 1 & 1 & & 3 \end{array} \qquad \begin{array}{rrrrrrr} & 1 & 0 & 1 & 0 & & 10 \\ - & & & 1 & 1 & - & 3 \\ \hline & & & & & & 07 \end{array} \qquad \begin{array}{rrrrrrrr} & 1 & 1 & 0 & 1 & 0 & & 26 \\ - & & 1 & 1 & 0 & 0 & - & 12 \\ \hline & & & & & & & 14 \end{array}$$

4-BIT PARALLEL SUBTRACTOR





1s Complement Method for Subtraction

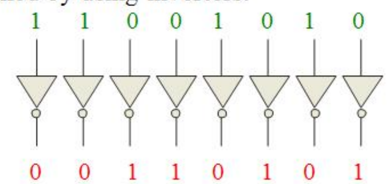
The 1's complement of a binary number is just the inverse of the digits.
To form the 1's complement, change all 0's to 1's and all 1's to 0's.

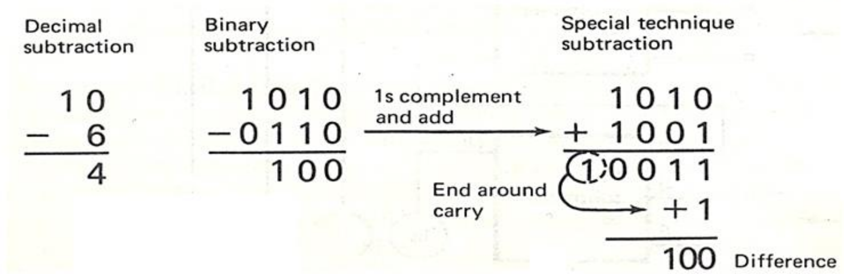
For example, the 1's complement of

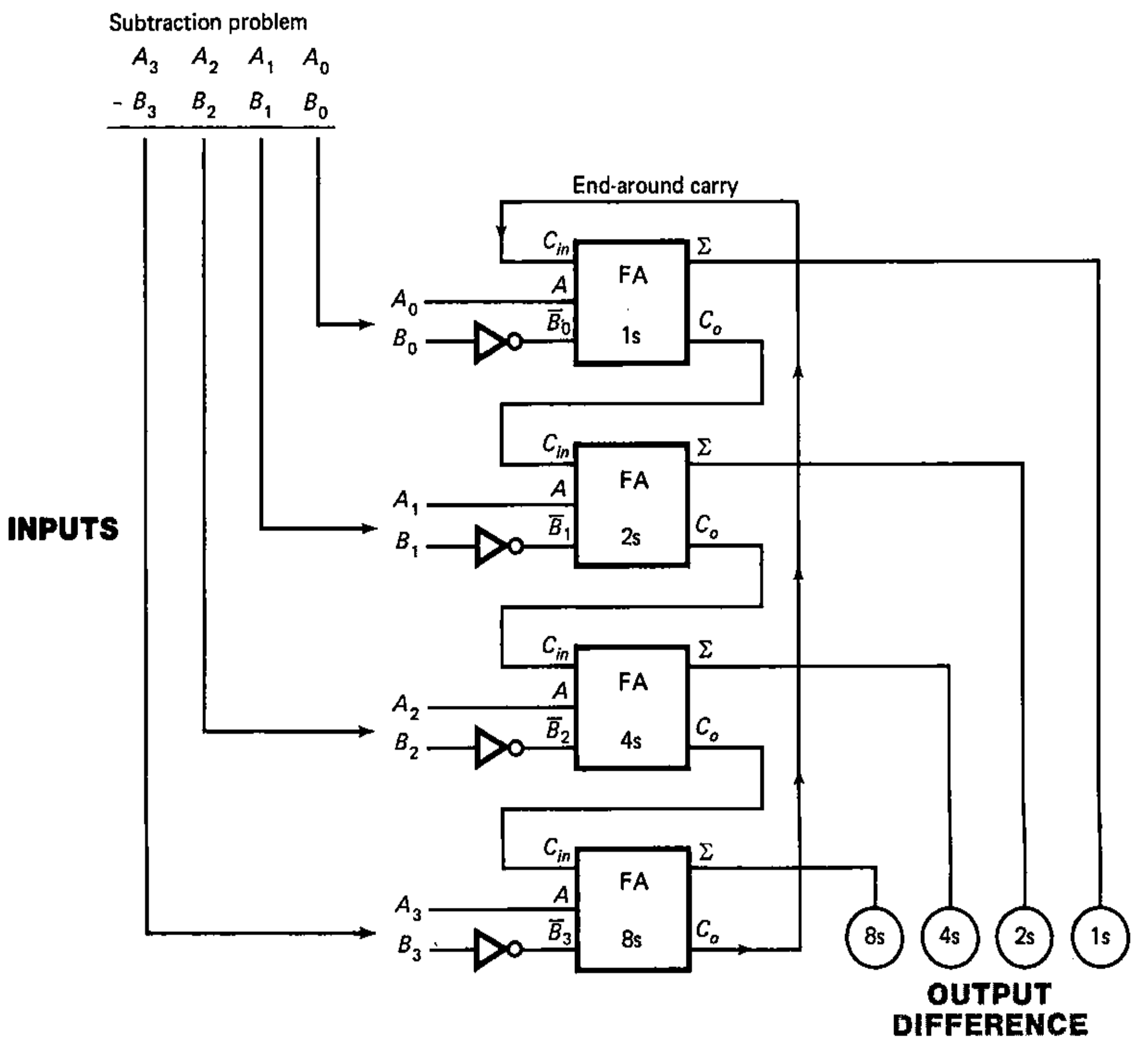
1 1 0 0 1 0 1 0

is 0 0 1 1 0 1 0 1

In digital circuits, the 1's complement is formed by using inverters:

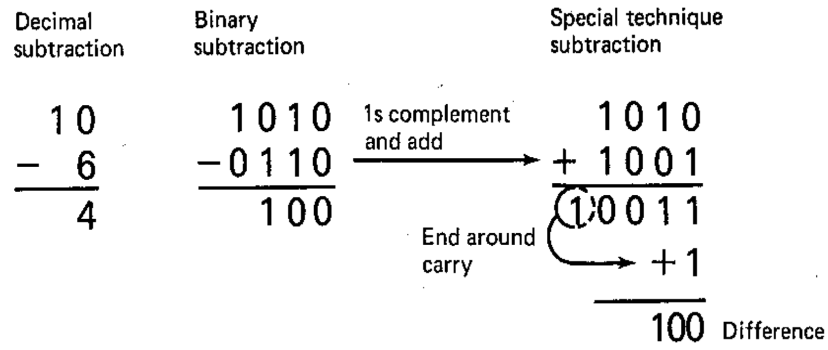


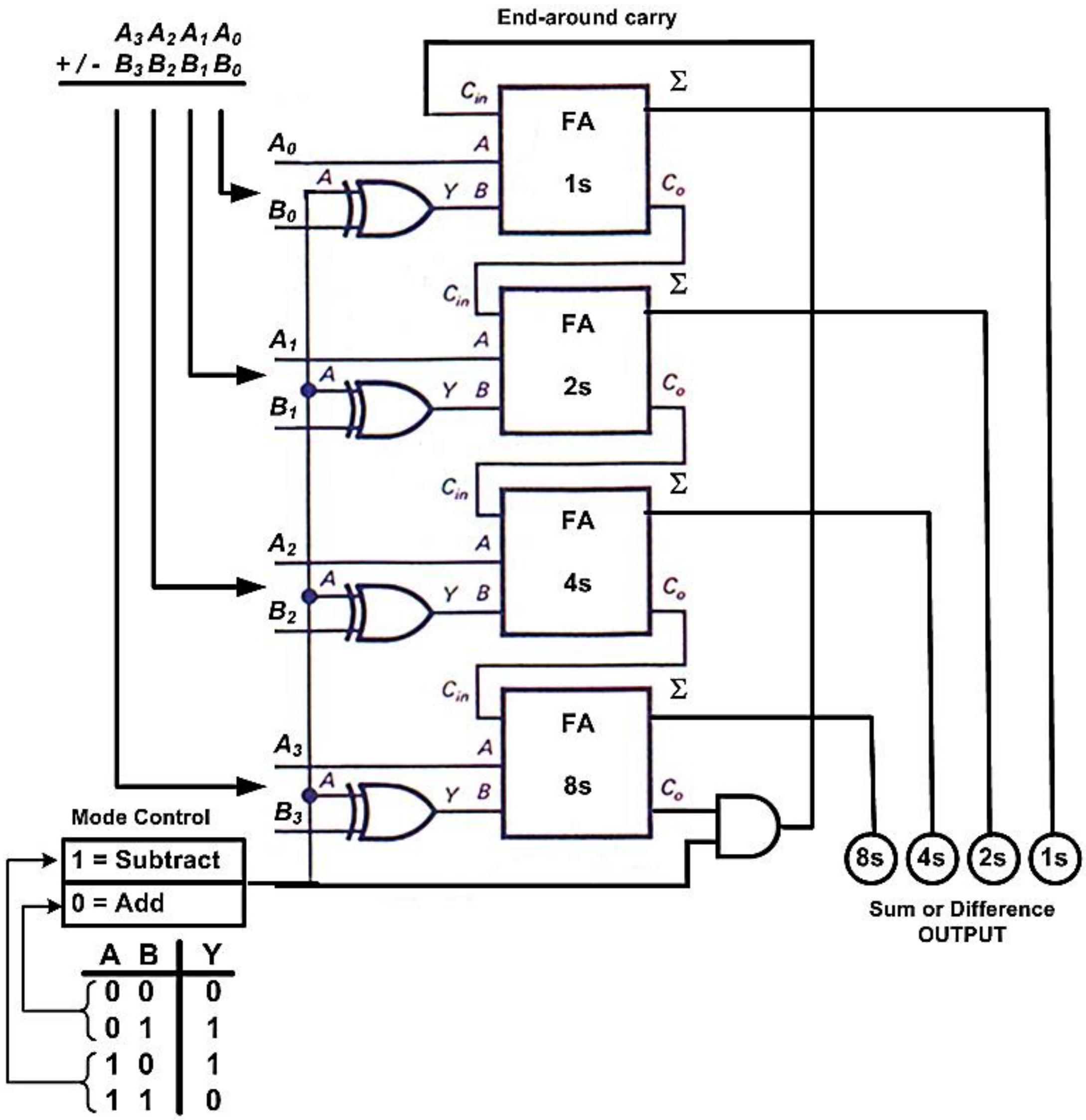




Bi-Functional 4-Bit Adder / Subtractor using 1s Complement Method

An example of 1s Complement and End-around Carry Subtraction





2s Complement Method

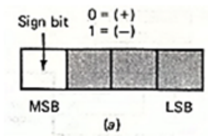
The 2s Complement of a binary number is found by adding 1 to the LSB of the 1s Complement.

Recall that the 1s Complement of 1 1 0 0 1 0 1 0

is 0 0 1 1 0 1 0 1 (1s Complement)

To form the 2s Complement, Add 1: +1

0 0 1 1 0 1 1 0 (2s Complement)

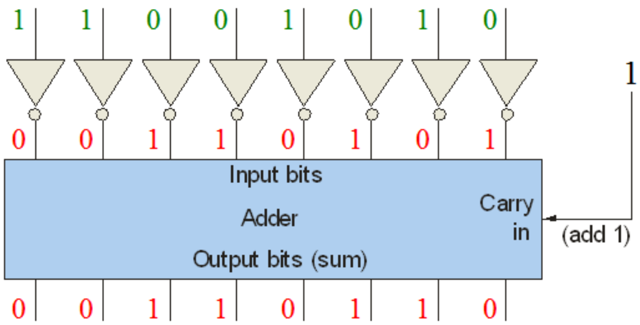


Signed decimal	4-bit 2s complement representation
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
−1	1111
−2	1110
−3	1101
−4	1100
−5	1011
−6	1010
−7	1001
−8	1000

Same as binary numbers

(b)

(9.15 2s Complement representations)



Signed Binary Numbers

There are several ways to represent signed binary numbers. In all cases, the MSB in a signed number is the sign bit, that tells you if the number is positive or negative.

Computers use a modified 2s Complement for signed numbers.

Positive numbers are stored in ***True*** form (with a 0 for the sign bit)

Negative numbers are stored in ***Complement*** form (with a 1 for the sign bit).

For example, the positive number 58 is written using 8-bits as 00111010 (true form).

Sign bit   Magnitude bits

Negative numbers are written as the 2's Complement of the corresponding positive number.

The negative number −58 is written as: −58 = 11000110 (Complement form)

Sign bit   Magnitude bits

An easy way to read a signed number that uses this notation is to assign the sign bit a column weight of −128 (for an 8-bit number). Then add the column weights for the 1's.

Assuming that the sign bit = −128, show that 11000110 = −58 as a 2's complement signed number:

Column weights: −128 64 32 16 8 4 2 1.

1 1 0 0 0 1 1 0

−128 +64 +4 +2 = −58

Arithmetic Operations with Signed Numbers

Using the signed number notation with negative numbers in 2s Complement form simplifies addition and subtraction of signed numbers.

Rules for Addition:

Add the two signed numbers. Discard any final carries. The result is in signed form.

Examples:

$$\begin{array}{r} 00011110 = +30 \\ 00001111 = +15 \\ \hline 00101101 = +45 \end{array} \qquad \begin{array}{r} 00001110 = +14 \\ 11101111 = -17 \\ \hline 11111101 = -3 \end{array} \qquad \begin{array}{r} 11111111 = -1 \\ 11111000 = -8 \\ \hline \cancel{1}11110111 = -9 \end{array}$$

Discard carry

Note that if the number of bits required for the answer is exceeded, overflow will occur. This occurs only if both numbers have the same sign. The overflow will be indicated by an incorrect sign bit. Two examples are:

$$\begin{array}{r} 01000000 = +128 \\ 01000001 = +129 \\ \hline 10000001 = \cancel{-126} \end{array} \qquad \begin{array}{r} 10000001 = -127 \\ 10000001 = -127 \\ \hline \cancel{1}00000010 = \cancel{+2} \end{array}$$

Discard carry

Wrong! The answer is incorrect and the sign bit has changed.

Rules for Subtraction:

2's complement the subtrahend and add the numbers. Discard any final carries. The result is in signed form.

Repeat the examples done previously, but subtract:

$$\begin{array}{rr} 00011110 & (+30) \\ -\ 00001111 & -(+15) \\ \hline \end{array} \qquad \begin{array}{rr} 00001110 & (+14) \\ -\ 11101111 & -(-17) \\ \hline \end{array} \qquad \begin{array}{rr} 11111111 & (-1) \\ -\ 11111000 & -(-8) \\ \hline \end{array}$$

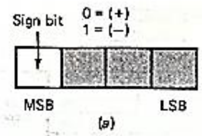
2's complement subtrahend and add:

$$\begin{array}{r} 00011110 = +30 \\ 11110001 = -15 \\ \hline \cancel{1}00001111 = +15 \end{array} \qquad \begin{array}{r} 00001110 = +14 \\ 00010001 = +17 \\ \hline 00011111 = +31 \end{array} \qquad \begin{array}{r} 11111111 = -1 \\ 00001000 = +8 \\ \hline \cancel{1}00000111 = +7 \end{array}$$

Discard carry

Discard carry

2s Complement Addition and Subtraction



(a)

Signed decimal	4-bit 2s complement representation
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
0	0000
–1	1111
–2	1110
–3	1101
–4	1100
–5	1011
–6	1010
–7	1001
–8	1000

Same as binary numbers (+7 to 0)

(b)

(9.15 2s Complement representations)

–4 (Decimal)
↓ Step ① Convert decimal to binary
0100 (Binary)
↓ Step ② 1s complement
1011 (1s complement)
↓ Step ③ Add +1 (1011 + 1 = 1100)
-4_{10} = 1100 (2s complement)

(9.16 (a) Signed binary number to 2s Complement conversion)

1100 (2s complement)
↓ Step ① 1s complement
0011 (1s complement)
↓ Step ② Add + 1 (0011 + 1 = 0100)
4_{10} = 0100 (Binary)

(9.16 (b) 2s Complement to Signed binary conversion)

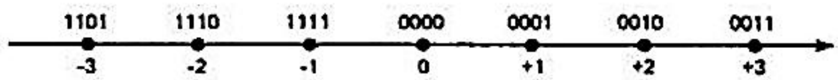
(a) (+4) + (+3) = $+7_{10}$: 0100 + 0011 = 0111 (2s complement SUM)

(b) (–1) + (–2) = -3_{10}: 1111 + 1110 = 1 1101 (2s complement SUM), Discard

(c) (+1) + (–3) = -2_{10}: 0001 + 1101 = 1110 (2s complement SUM)

(d) (+5) + (–4) = $+1_{10}$: 0101 + 1100 = 1 0001 (2s complement SUM), Discard

(9.17 Four examples of adding 2s complement numbers)



–4 (Decimal)
↓ Step ① Convert decimal to binary
0100 (Binary)
↓ Step ② 1s complement
1011 (1s complement)
↓ Step ③ Add +1 (1011 + 1 = 1100)
-4_{10} = 1100 (2s complement)

(9.16 (a) Signed binary number to 2s Complement conversion)

1100 (2s complement)
↓ Step ① 1s complement
0011 (1s complement)
↓ Step ② Add + 1 (0011 + 1 = 0100)
4_{10} = 0100 (Binary)

(9.16 (b) 2s Complement to Signed binary conversion)

(a) (+7) – (+3) = $+4_{10}$; (+3) = 0011 → Form 2s complement and ADD: 0111 + 1101 = 1 0100 (2s complement DIFFERENCE), Discard

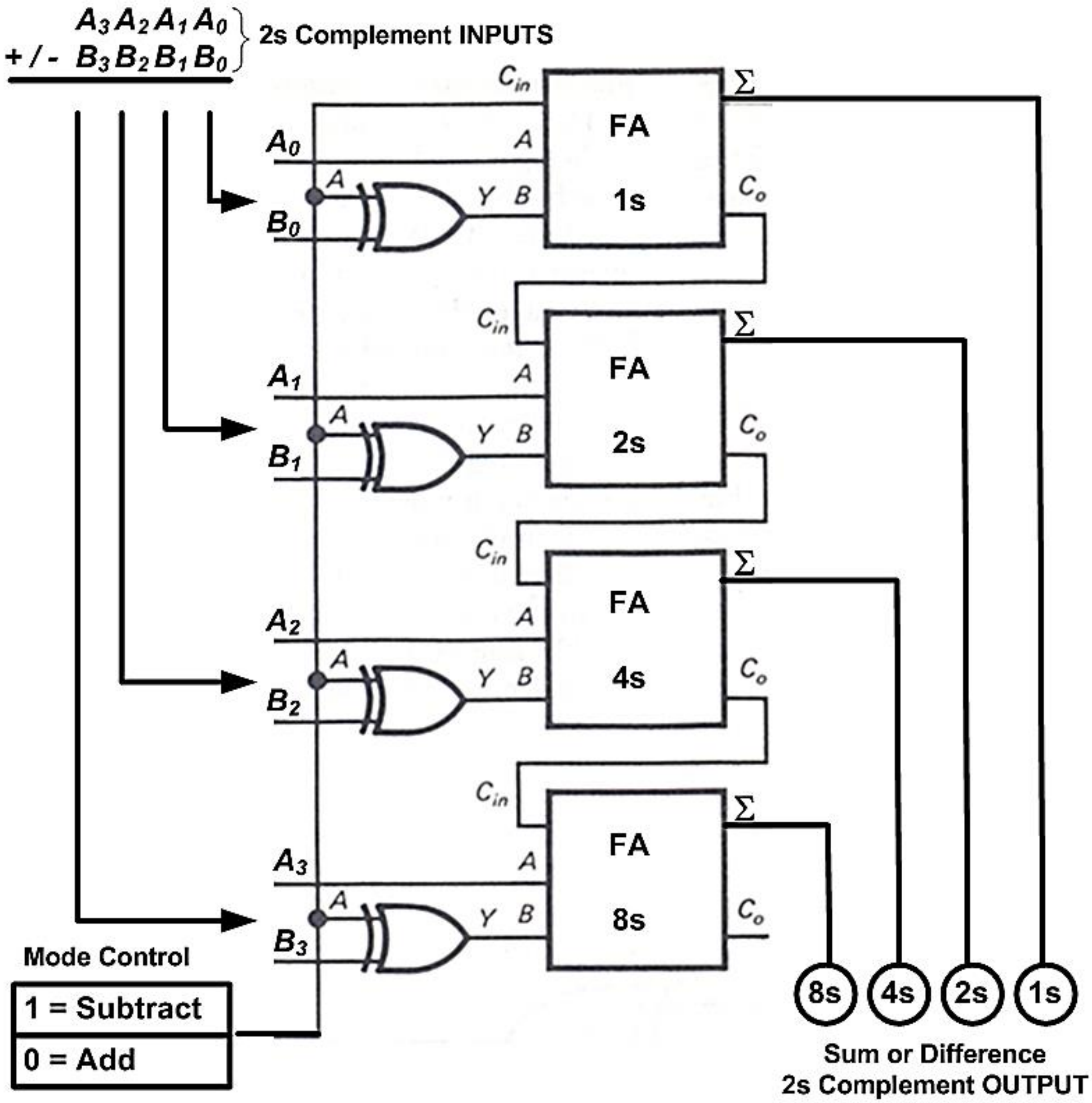
(b) (–8) – (–3) = -5_{10}; (–3) = 1101 → Form 2s complement and ADD: 1000 + 0011 = 1011 (2s complement DIFFERENCE)

(c) (+3) – (–3) = $+6_{10}$; (–3) = 1101 → Form 2s complement and ADD: 0011 + 0011 = 0110 (2s complement DIFFERENCE)

(d) (–4) – (+2) = -6_{10}; (+2) = 0010 → Form 2s complement and ADD: 1100 + 1110 = 1 1010 (2s complement DIFFERENCE), Discard

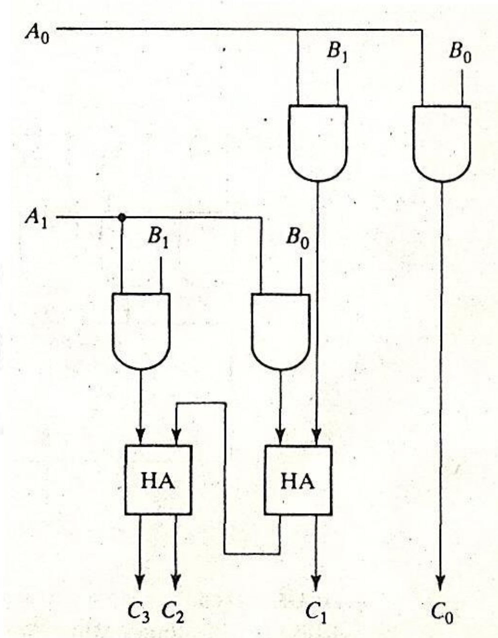
(9.18 Four examples of Signed Subtraction using 2s complement numbers)

2s Complement Addition and Subtraction



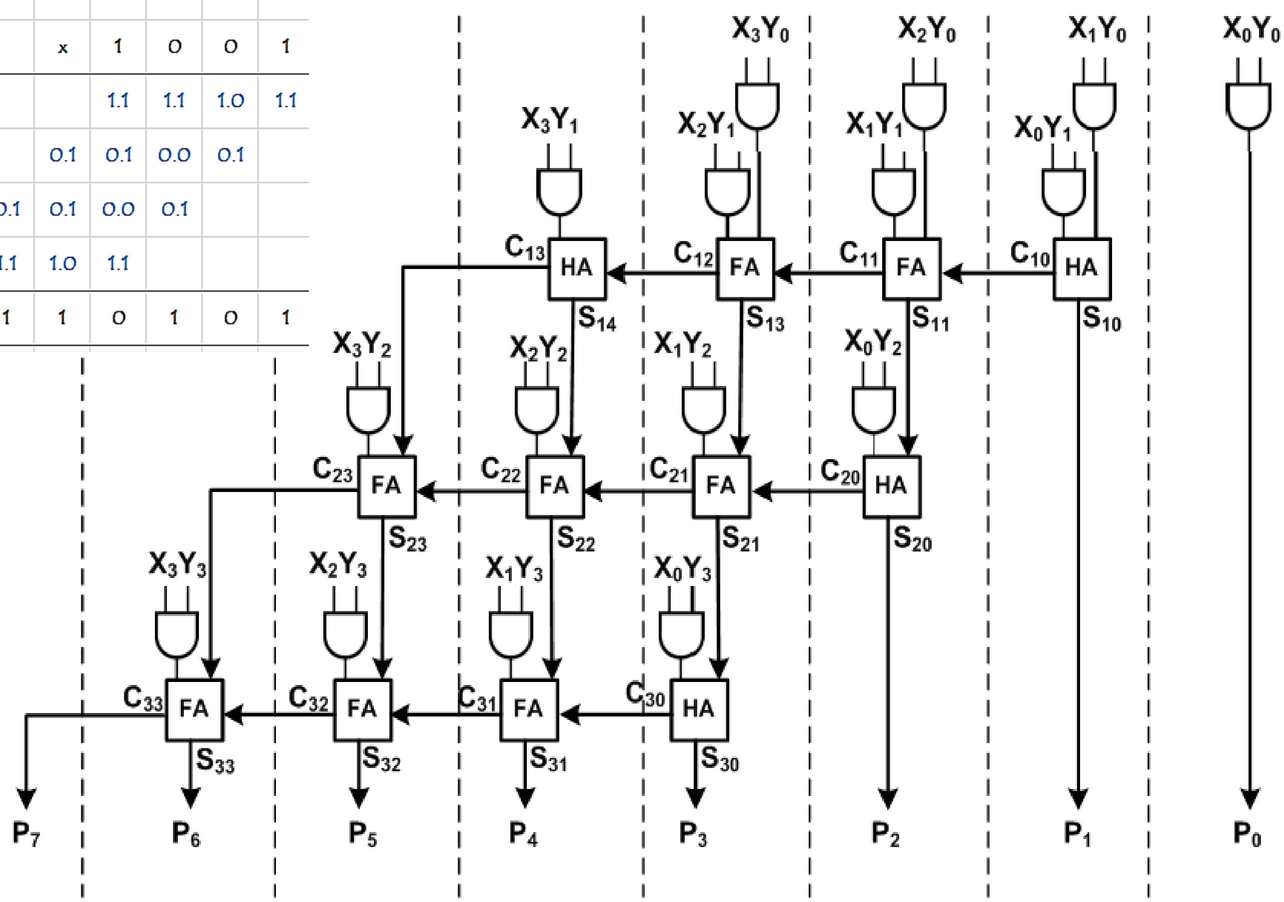
BINARY MULTIPLICATION

		B_1	B_0
		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
C_3	C_2	C_1	C_0



N x N BIT BINARY MULTIPLICATION

			1	1	0	1
		x	1	0	0	1
			1.1	1.1	1.0	1.1
		0.1	0.1	0.0	0.1	
	0.1	0.1	0.0	0.1		
1.1	1.1	1.0	1.1			
1	1	1	0	1	0	1



Multiplication using Booth Algorithm
Click for Reference Link: Booth Algorithm

Booth's multiplication algorithm is an algorithm that multiplies two signed binary numbers in two's complement notation. The algorithm was invented by Andrew Donald Booth in 1951. Booth's algorithm involves repeatedly adding one of two pre-determined values A and S to a product P, then performing a rightward arithmetic shift on P. Let m and r be the multiplicand and multiplier, respectively; and let x and y represent the number of bits in m and r.

1. Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to $(x + y + 1)$.
   - i. A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining $(y + 1)$ bits with zeros.
   - ii. S: Fill the most significant bits with the value of $(-m)$ in two's complement notation. Fill the remaining $(y + 1)$ bits with zeros.
   - iii. P: Fill the most significant x bits with zeros. To the right of this, append the value of r. Fill the least significant (rightmost) bit with a zero.
2. Determine the two least significant (rightmost) bits of P.
   - i. If they are 00, do nothing. Use P directly in the next step.
   - ii. If they are 01, find the value of $P + A$. Ignore any overflow.
   - iii. If they are 10, find the value of $P + S$. Ignore any overflow.
   - iv. If they are 11, do nothing. Use P directly in the next step.
3. Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.
4. Repeat steps 2 and 3 until they have been done y times.
5. Drop the least significant (rightmost) bit from P. This is the product of m and r.

4. EXAMPLE

Find the value of $(-3) \times 2$ in decimal system?

Let $m = -3$ and $r = 2$, and $x = 4$ and $y = 4$:

- i. $m = 1101$
- ii. $m' = 0011$ (2s Complement of m)
- iii. $r = 0010$

Therefore,

- i. $A = 1101\ 0000\ 0$
- ii. $S = 0011\ 0000\ 0$
- iii. $P = 0000\ 0010\ 0$

Perform the loop four times:

1) $P = 0000\ 0010\ \mathbf{0}$. The last two bits are 00.
   $P = 0000\ 0001\ 0$. Do nothing, Arithmetic right shift.
2) $P = 0000\ 0001\ \mathbf{0}$. The last two bits are 10.
   $P = 0011\ 0001\ 0$. $P = P + S$.
   $P = 0001\ 1000\ 1$. Arithmetic right shift.
3) $P = 0001\ 1000\ \mathbf{1}$. The last two bits are 01.
   $P = 1110\ 1000\ 1$. $P = P + A$.
   $P = 1111\ 0100\ 0$. Arithmetic right shift.
4) $P = 1111\ 0100\ \mathbf{0}$. The last two bits are 00.
   $P = 1111\ 1010\ 0$. Do nothing, Arithmetic right shift.

Drop the least significant (rightmost) bit from P. We obtained 1111 1010. This is the product of $(-3) \times 2$, which is equivalent to -6 decimal. [6].