Arithmetic Circuits

Digital Design. M. Morris Mano

Arithmetic Circuits

An arithmetic circuit consists of gates computing arithmetic operations: (addition, subtraction, multiplication …) with wires connecting the gates.

Half adder

The half adder is a combinational arithmetic circuit that adds two bits and produces a sum bit (S) and carry bit (C) as the outputs.

$$S = \overline{X}Y + X\overline{Y} = X \oplus Y$$

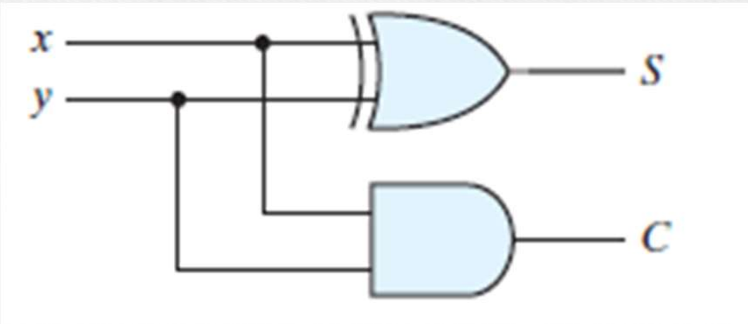
$$C = XY$$

Inputs		Outputs	
X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

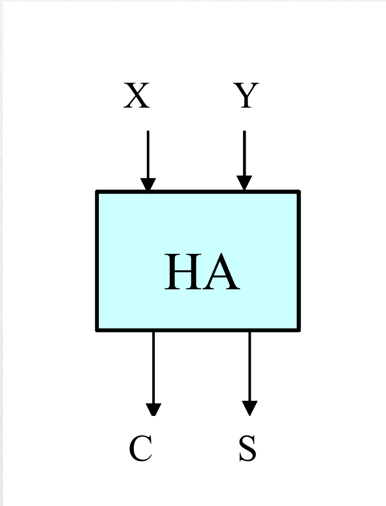
Half adder

$$S = \bar{X}Y + X\bar{Y} = X \oplus Y$$

$$C = XY$$



Logic Digram



Graphical Symbol

Full adder

The half adder is a combinational arithmetic circuit that adds three bits and produces a sum bit (S) and carry bit (C) as the outputs.

The three bits may be two inputs and the third input is the carry from previous stage.

Full adder

Inputs			Outputs	
X	Y	Z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

X \ YZ	00	01	11	10
0		1		1
1	1		1	

$$S = X \oplus Y \oplus Z$$

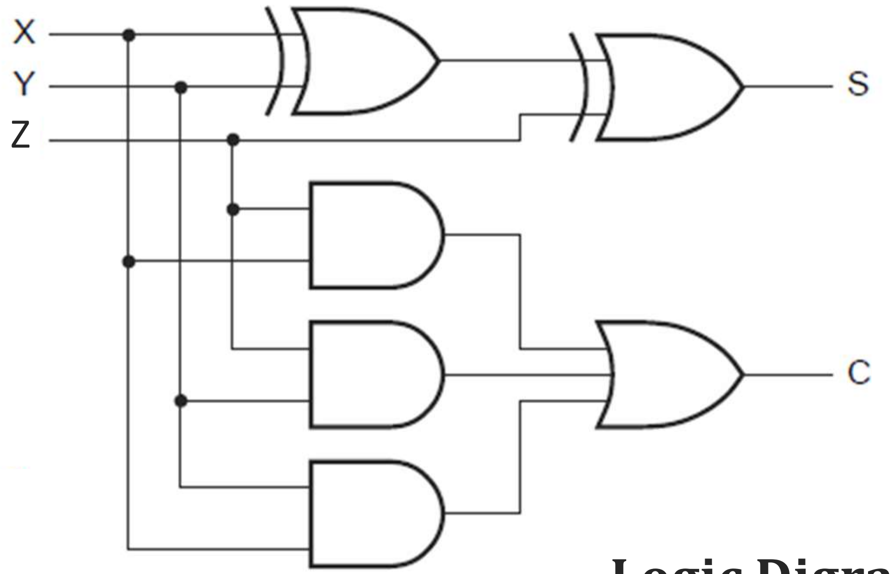
X \ YZ	00	01	11	10
0			1	
1		1	1	1

$$C = XY + YZ + XZ$$

Full adder

$S = X \oplus Y \oplus Z$

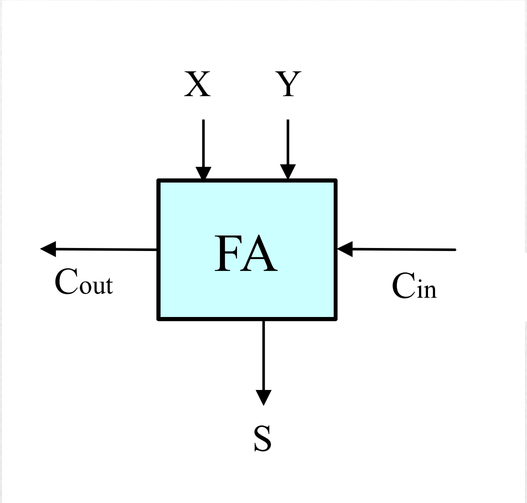
$C = XY + YZ + XZ$



Logic Digram

Full adder

As usually, the three bits are two inputs and the third input is the carry from previous stage, so the graphical symbol is usually drawn as shown.



Graphical Symbol

Implementing a Full Adder using Two Half Adders and an OR Gate

$S = X \oplus Y \oplus Z = (X \oplus Y) \oplus Z$

$C = XY + YZ + XZ$

$C = XY + YZ(X + \bar{X}) + XZ(Y + \bar{Y})$

$C = XY + XYZ + \bar{X}YZ + \cancel{XYZ} + X\bar{Y}Z$

$C = (XY + XYZ) + (\bar{X}YZ + X\bar{Y}Z)$

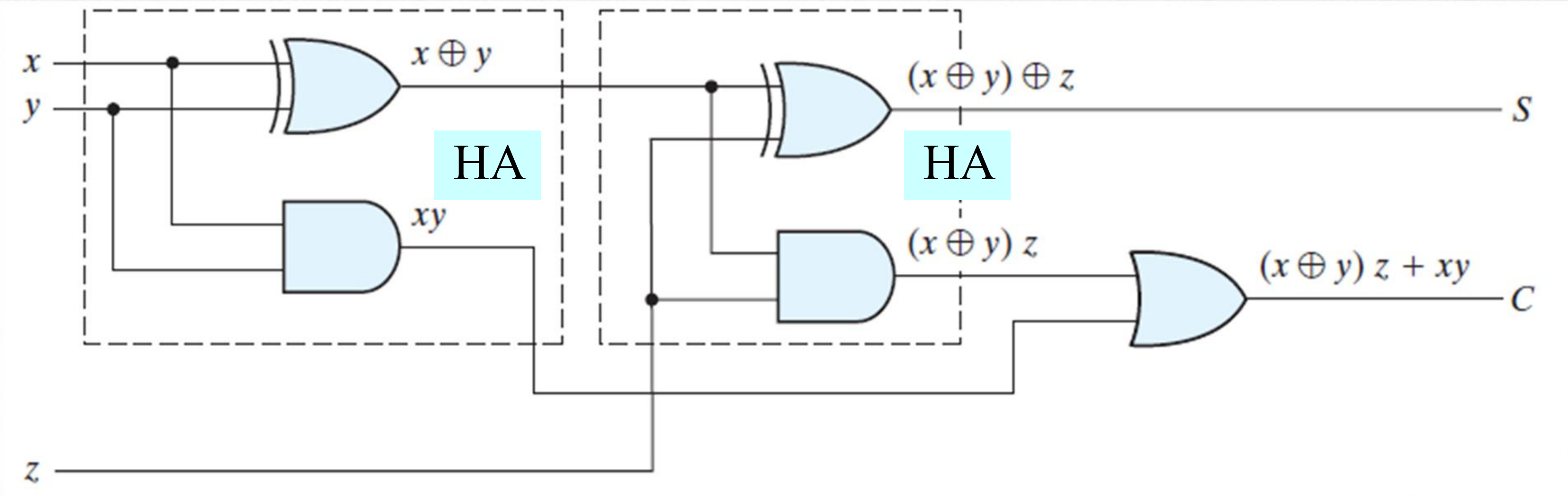
$C = XY(1 + Z) + Z(\bar{X}Y + X\bar{Y})$

$C = XY + Z(X \oplus Y)$

Implementing a Full Adder using Two Half Adders and an OR Gate

$S = X \oplus Y \oplus Z = (X \oplus Y) \oplus Z$

$C = XY + YZ + XZ = XY + Z(X \oplus Y)$

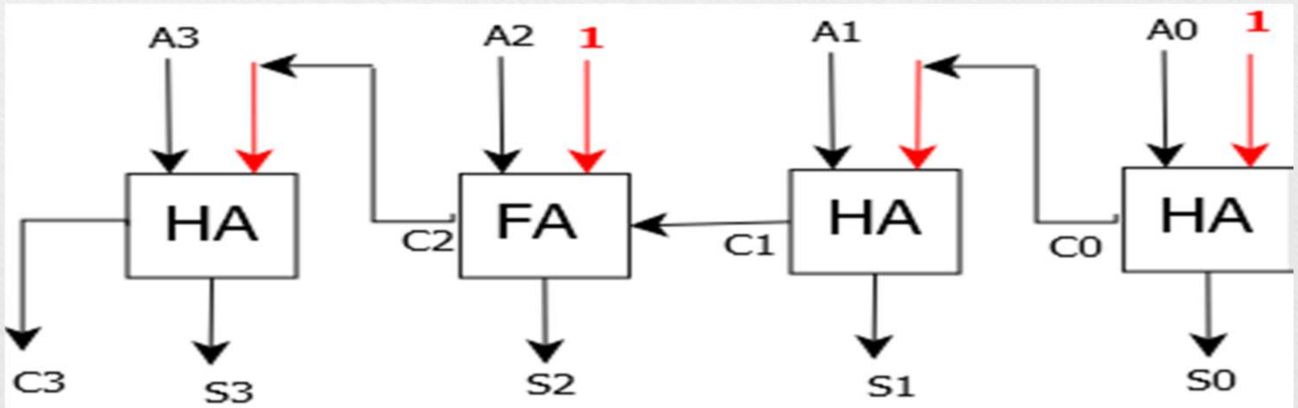


Building a circuit using HA(s) & FA(s)

Example 1:

Using the minimum number of half adder(s) and full adder(s), design a combinational circuit that adds five to a 4-bit binary number (A_3 A_2 A_1 A_0).

$$\begin{array}{ccccc} & C_3 & C_2 & C_1 & C_0 \\ & A_3 & A_2 & A_1 & A_0 \\ + & 0 & 1 & 0 & 1 \end{array}$$



Building a circuit using HA(s) & FA(s)

Exercise 1:

Using the minimum number of half adders & full adders to add ten to a 4-bit binary number $A_3A_2A_1A_0$

Binary Adder

A Binary Adder is a digital circuit that performs the arithmetic sum of two binary numbers provided with any length.

A Binary Adder is constructed using full-adder circuits connected in series, with the output carry from one full-adder connected to the input carry of the next full-adder.

Note that a half adder can be used for the addition of the least significant bits of the two numbers, but a full adder is used instead to be able to build cascaded circuits.

n-bit Binary Adder

An n-bit adder adds two numbers each is composed of n bits.

It has 2n+1 inputs; n augend bits + n addend bits + a carry from a pervious stage.

It has n+1 outputs; n represents the sum + a final carry.

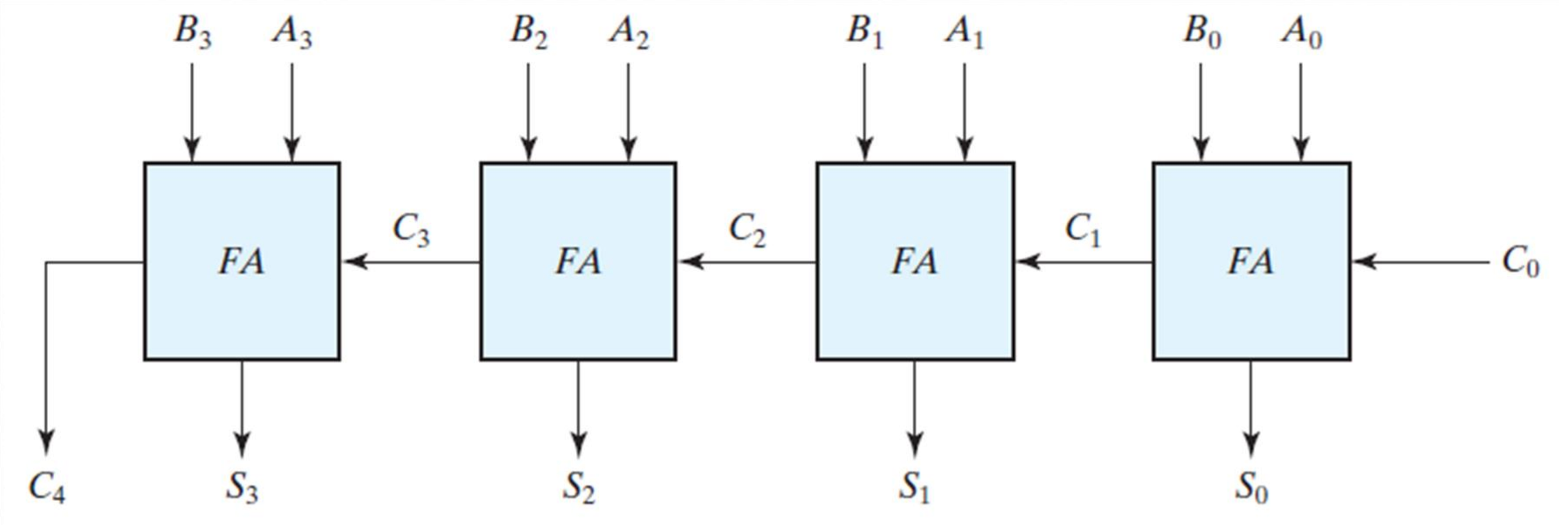
4-bit Binary Adder

The following figure represents a 4-bit adder adds two numbers each of 4 bits ($A_3A_2A_1A_0$) and ($B_3B_2B_1B_0$) and a previous carry (C_0).

In each stage, the bit A_i is added to the bit B_i and the carry C_i using a full adder, it produces a sum S_i and a Carry C_{i+1} feed to the next stage.

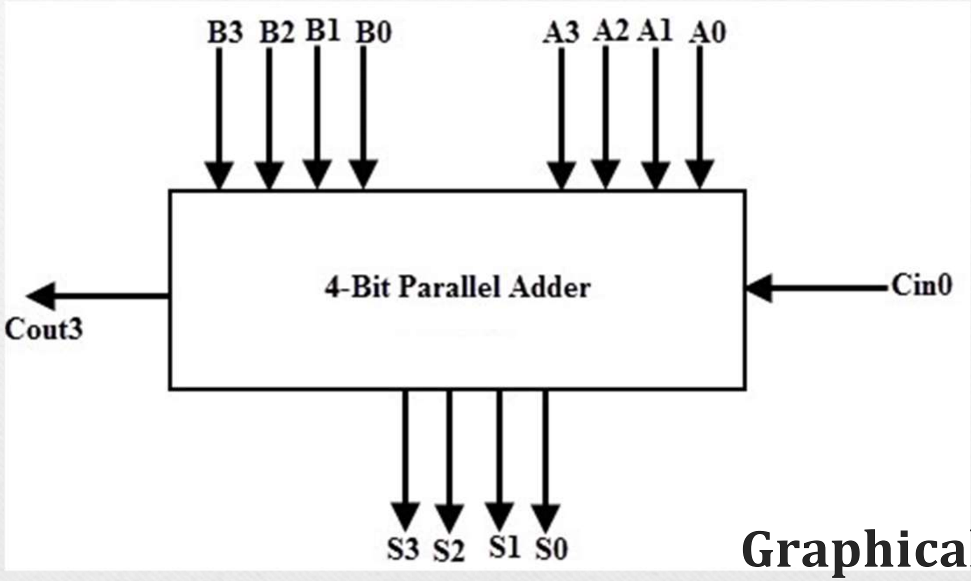
The final output is $S_3S_2S_1S_0$ and a final Carry C_4.

4-bit Binary Adder



Logic Digram

4-bit Binary Adder

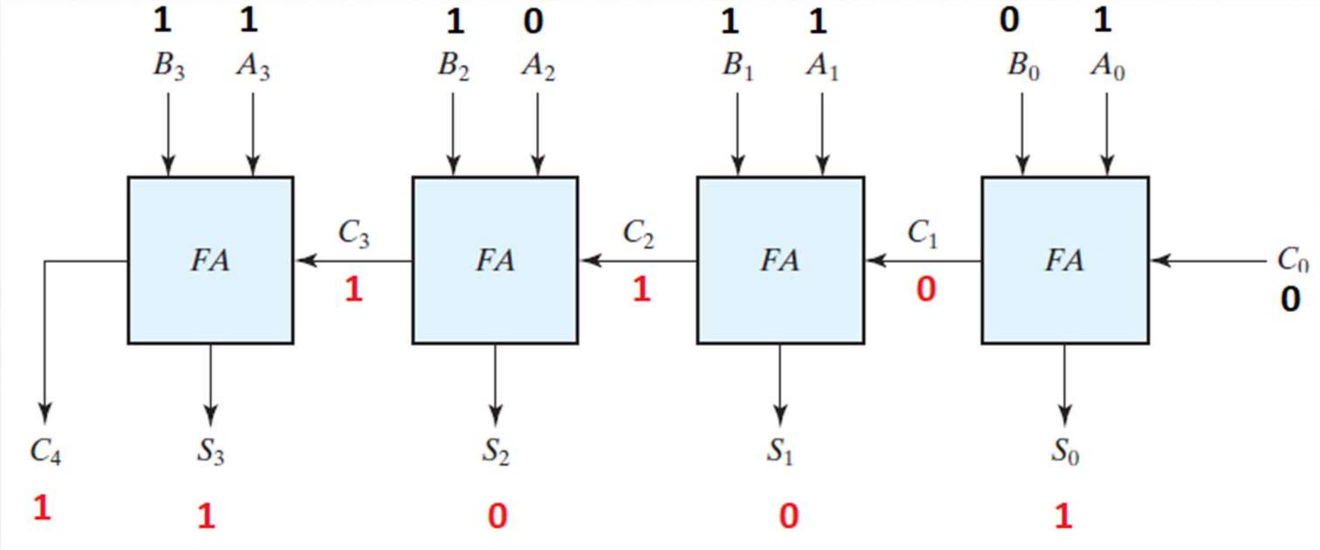


Graphical Symbol

4-bit Binary Adder

Example:

i	4	3	2	1	0
C	1	1	1	0	0
A		1	0	1	1
B		1	1	1	0
S		1	0	0	1



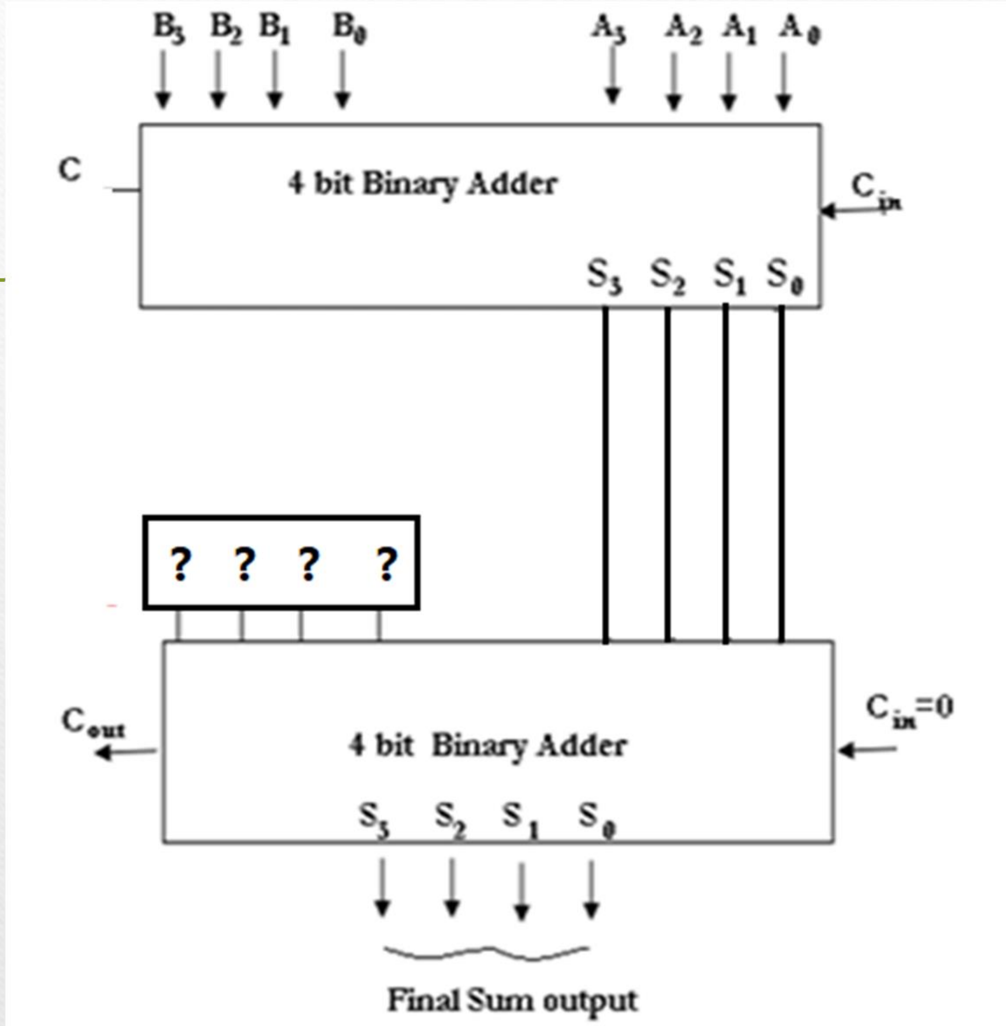
4-bit Binary Adder

BCD adder

<u>Example 2:</u>

Using two 4-bit binary adders, design a circuit that add two BCD numbers ($A_3A_2A_1A_0$) and ($B_3B_2B_1B_0$).

- **Naming the results S ($S_3S_2S_1S_0$) and carry (C).**
- **If S is <10, nothing must be done, otherwise we need to add 6 (0110) to S.**


B3 B2 B1 B0
A3 A2 A1 A0
C
4 bit Binary Adder
Cin
S3 S2 S1 S0
? ? ? ?
Cout
4 bit Binary Adder
Cin=0
S3 S2 S1 S0
Final Sum output

4-bit Binary Adder

BCD adder

- Using another 4-bit adder, the first input is ($S_3S_2S_1S_0$), while the other input may be 0000 or 0110
- In other word If C S_3 S_2 S_1 S_0 >9   add 0110

  else add 0000
- So the first and last bit are always 0, while the two other bits may be 0 or 1 according to the value of $CS_3S_2S_1S_0$. So we will design a function Z that is equal to 1 if $CS_3S_2S_1S_0>9$ and let it be the input of the middle bits, while the first and last bits are set to 0.

	C	S3	S2	S1	S0	Z
0	0	0	0	0	0	0
1	0	0	0	0	1	0
2	0	0	0	1	0	0
3	0	0	0	1	1	0
4	0	0	1	0	0	0
5	0	0	1	0	1	0
6	0	0	1	1	0	0
7	0	0	1	1	1	0
8	0	1	0	0	0	0
9	0	1	0	0	1	0
10	0	1	0	1	0	1
11	0	1	0	1	1	1
12	0	1	1	0	0	1
13	0	1	1	0	1	1
14	0	1	1	1	0	1
15	0	1	1	1	1	1
16	1	0	0	0	0	1
17	1	0	0	0	1	1
18	1	0	0	1	0	1
19	1	0	0	1	1	1
20	1	0	1	0	0	x
21	1	0	1	0	1	x
22	1	0	1	1	0	x
23	1	0	1	1	1	x
24	1	1	0	0	0	x
25	1	1	0	0	1	x
26	1	1	0	1	0	x
27	1	1	0	1	1	x
28	1	1	1	0	0	x
29	1	1	1	0	1	x
30	1	1	1	1	0	x
31	1	1	1	1	1	x

4-bit Binary Adder

BCD adder

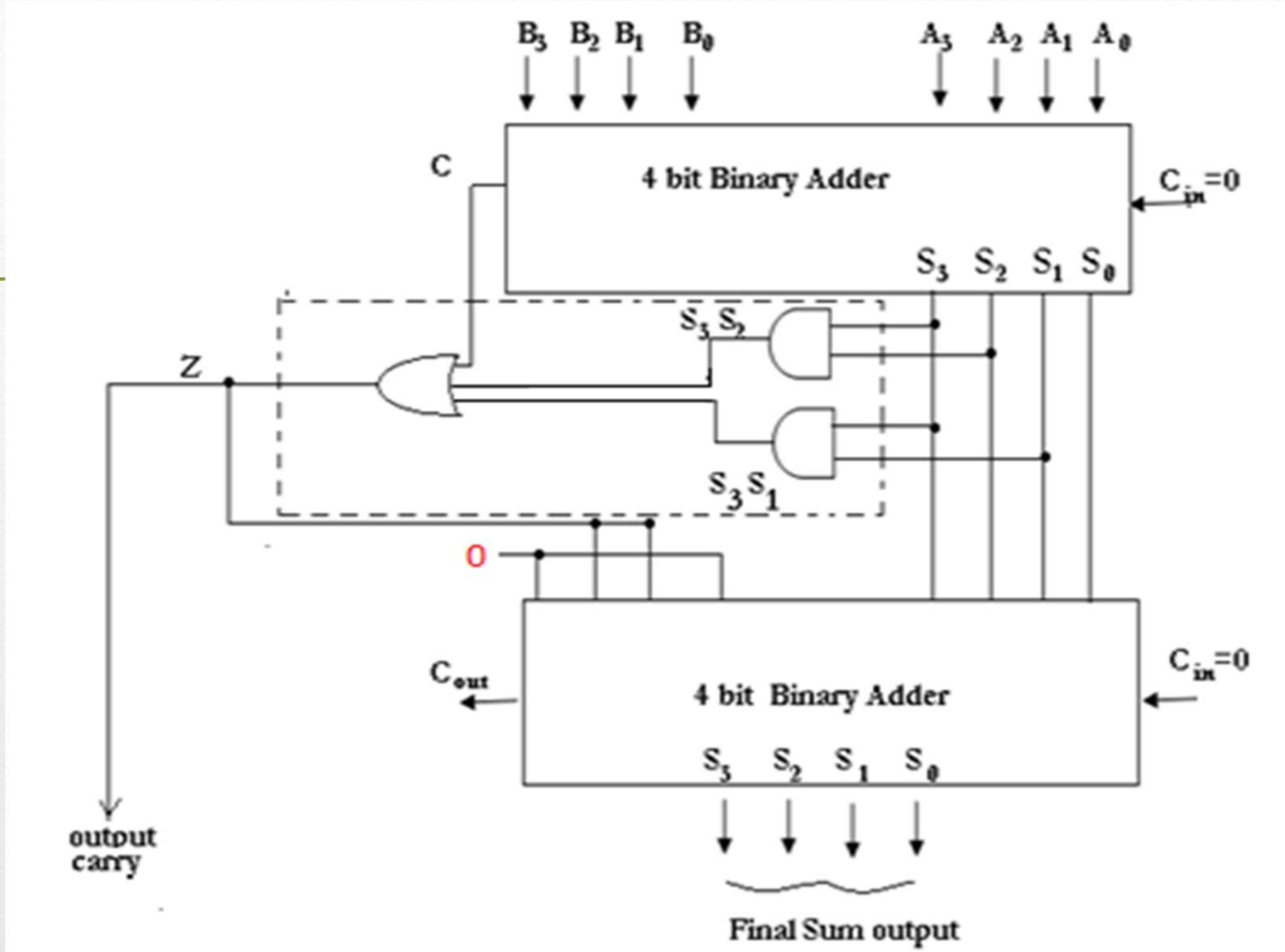
C=0

S3S2 \ S1S0	00	01	11	10
00				
01				
11	1	1	1	1
10			1	1

C=1

S3S2 \ S1S0	00	01	11	10
00	1	1	1	1
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

$$Z(C, S_3, S_2, S_1, S_0) = C + S_3S_2 + S_3S_1$$

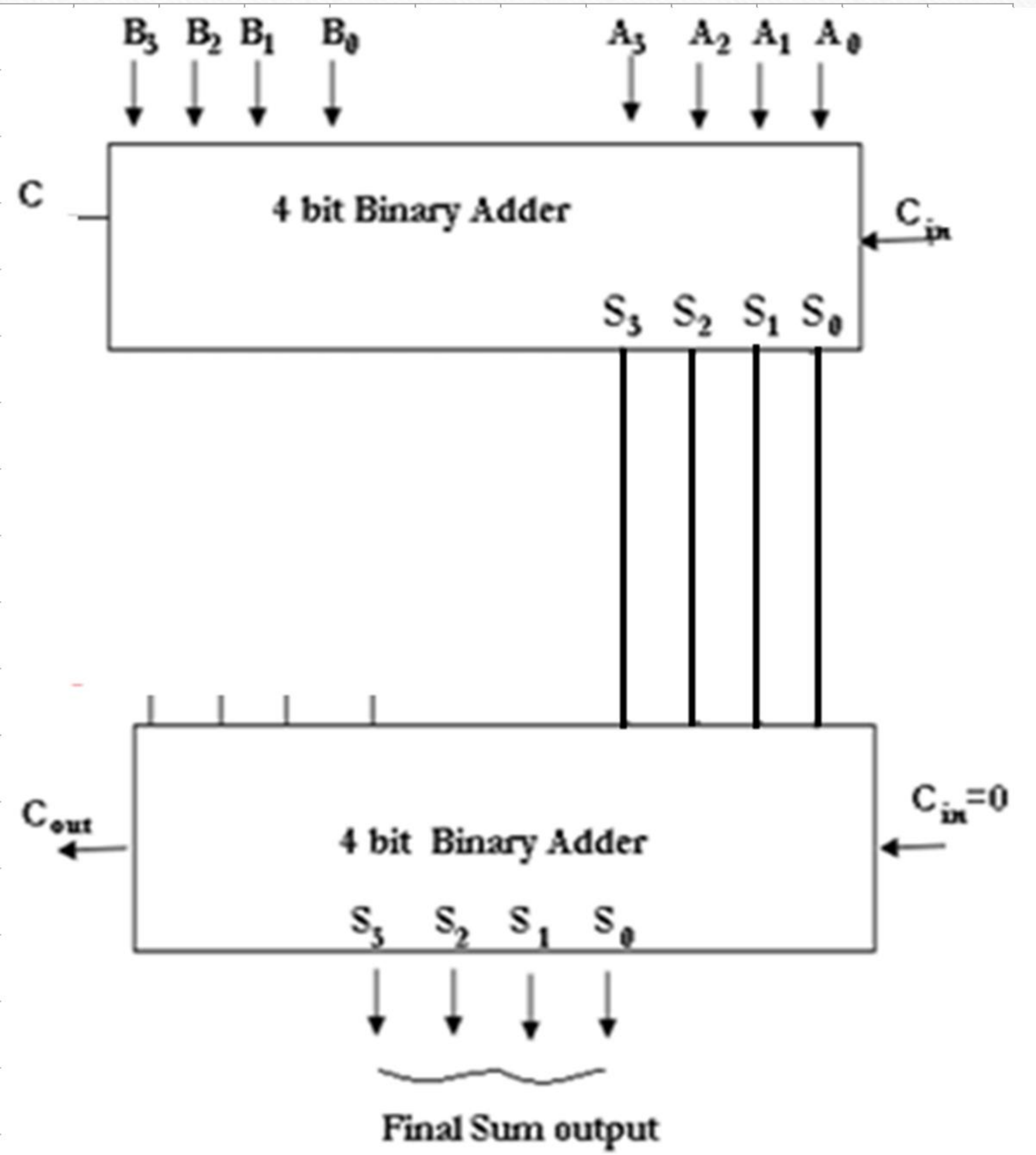

B3 B2 B1 B0
A3 A2 A1 A0
C
4 bit Binary Adder
Cin=0
S3 S2 S1 S0
S3 S2
Z
S3 S1
0
Cout
4 bit Binary Adder
Cin=0
S3 S2 S1 S0
output carry
Final Sum output

4-bit Binary Adder

<u>Exercise 2:</u>

Use only two 4-bit adders and an inverter, design a circuit that add two numbers represented in excess-3 code, known that the correction after adding the two digits with a 4-bit binary adder is as follows:

- The output carry is equal to the carry from the binary adder.
- If the output carry = 1, then add 0011.
- If the output carry = 0, then add 1101.

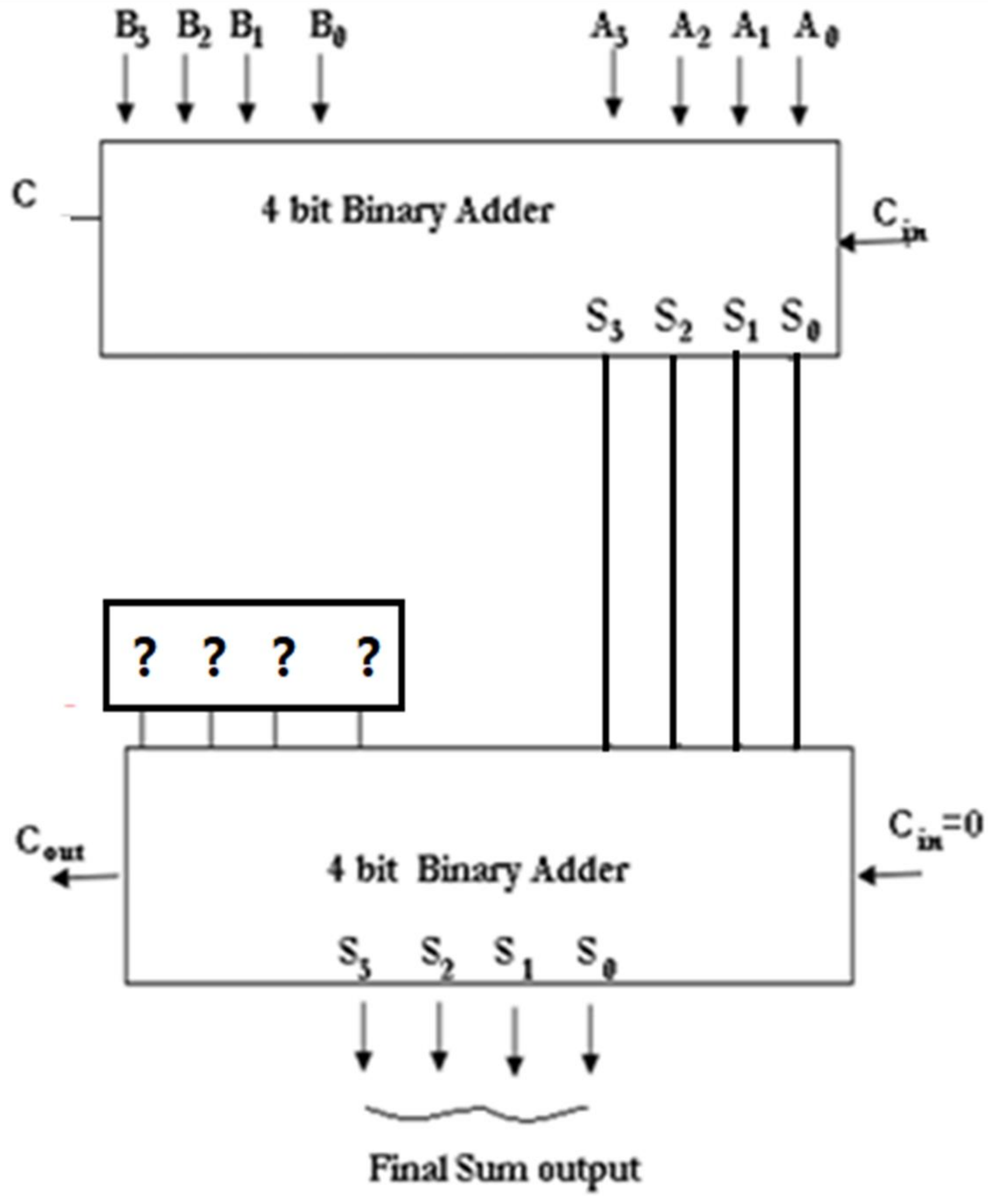

B3 B2 B1 B0
A3 A2 A1 A0
C
4 bit Binary Adder
Cin
S3 S2 S1 S0
Cout
4 bit Binary Adder
Cin=0
S3 S2 S1 S0
Final Sum output

4-bit Binary Adder

<u>Exercise 3:</u>

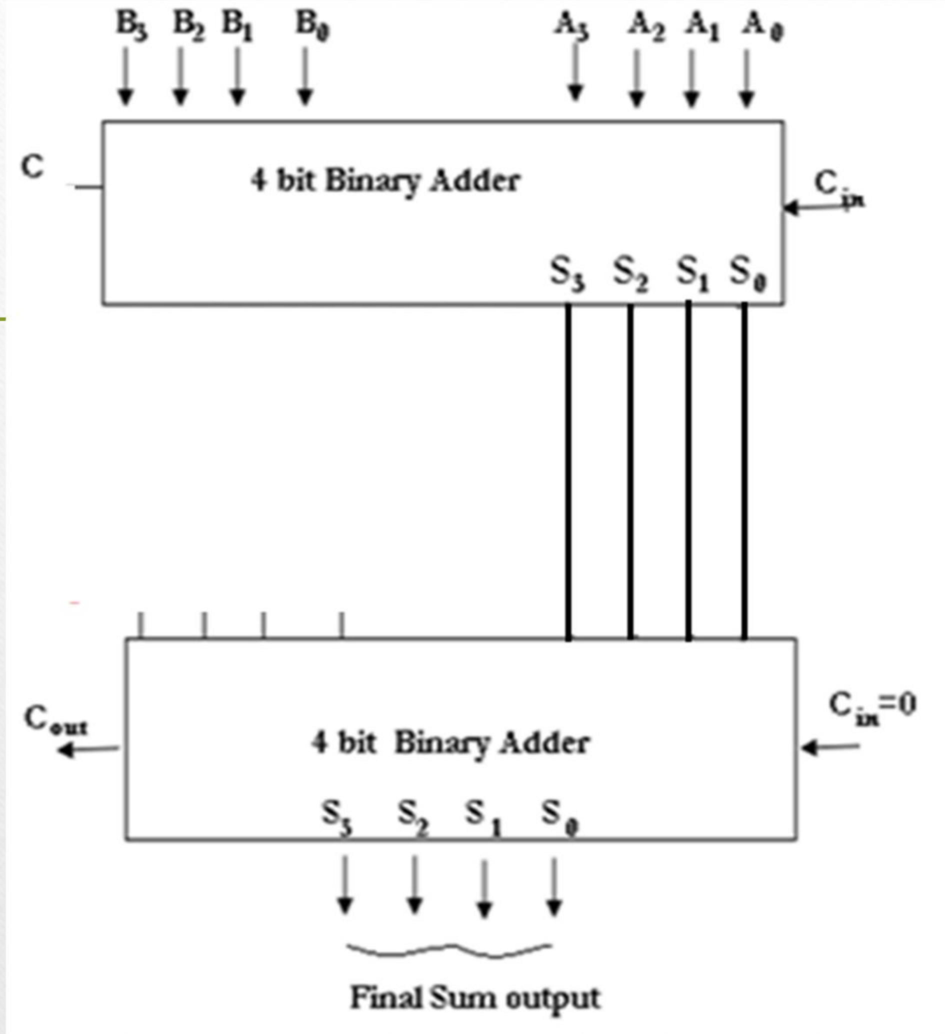
Using only two 4-bit adders and any simple gate you may need, design a circuit that add two 4-bit binary numbers, then do the following:

- If the result is between 0 and 15 included add 5.
- If the result is between 16 and 31 included subtract 5.



B3 B2 B1 B0
A3 A2 A1 A0
C
4 bit Binary Adder
Cin
S3 S2 S1 S0
? ? ? ?
Cout
4 bit Binary Adder
Cin=0
S3 S2 S1 S0
Final Sum output

	C	S3	S2	S1	S0	Z
0	0	0	0	0	0	
1	0	0	0	0	1	
2	0	0	0	1	0	
3	0	0	0	1	1	
4	0	0	1	0	0	
5	0	0	1	0	1	
6	0	0	1	1	0	
7	0	0	1	1	1	
8	0	1	0	0	0	
9	0	1	0	0	1	
10	0	1	0	1	0	
11	0	1	0	1	1	
12	0	1	1	0	0	
13	0	1	1	0	1	
14	0	1	1	1	0	
15	0	1	1	1	1	
16	1	0	0	0	0	
17	1	0	0	0	1	
18	1	0	0	1	0	
19	1	0	0	1	1	
20	1	0	1	0	0	
21	1	0	1	0	1	
22	1	0	1	1	0	
23	1	0	1	1	1	
24	1	1	0	0	0	
25	1	1	0	0	1	
26	1	1	0	1	0	
27	1	1	0	1	1	
28	1	1	1	0	0	
29	1	1	1	0	1	
30	1	1	1	1	0	
31	1	1	1	1	1	


B3 B2 B1 B0
A3 A2 A1 A0
C
4 bit Binary Adder
Cin
S3 S2 S1 S0
Cout
4 bit Binary Adder
Cin=0
S3 S2 S1 S0
Final Sum output

Ripple Binary Adder

This type of binary adder is called ripple adder.

As all combinational circuit, it cannot compute the output instantaneously, there is a delay between allocating inputs and producing the correct answer.

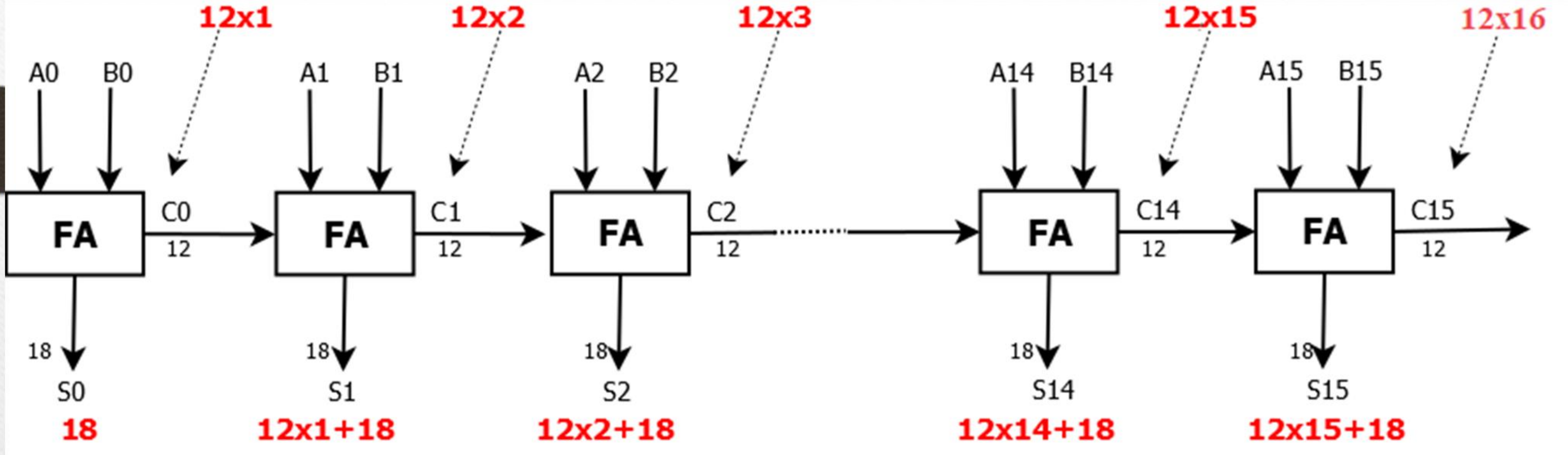
The problem with ripple (serial) adder is; for each stage to produce a correct answer, it has to wait for its previous stage carry, so the delay is accumulated.

Ripple Binary Adder

Example: Having a 16-bit ripple adder, if the carry (C) propagation delay of each full adder is 12 ns and the sum (S) propagation delay of each full adder is 18 ns. What is the propagation delay of this adder?

The propagation delay of this adder = Time after which output sum bit becomes available from the last full adder = Time taken for its carry in to become available + Sum propagation delay of full adder = {Carry (C) propagation delay of full adder X Total number of full adders before last full adder} + Sum (S) propagation delay of full adder = (12*15)+18=198ns

Ripple Binary Adder



Carry look ahead Binary Adder

A carry-look ahead adder (fast parallel adder) is a type of adders, that improves the speed by reducing the amount of time required to determine carry bits, so reducing the propagation delay.

A carry-look ahead adder generates the carry-in of each full adder simultaneously without causing any delay.

The carry-in of any stage full adder is independent of the carry bits generated during intermediate stages.

Carry look ahead Binary Adder

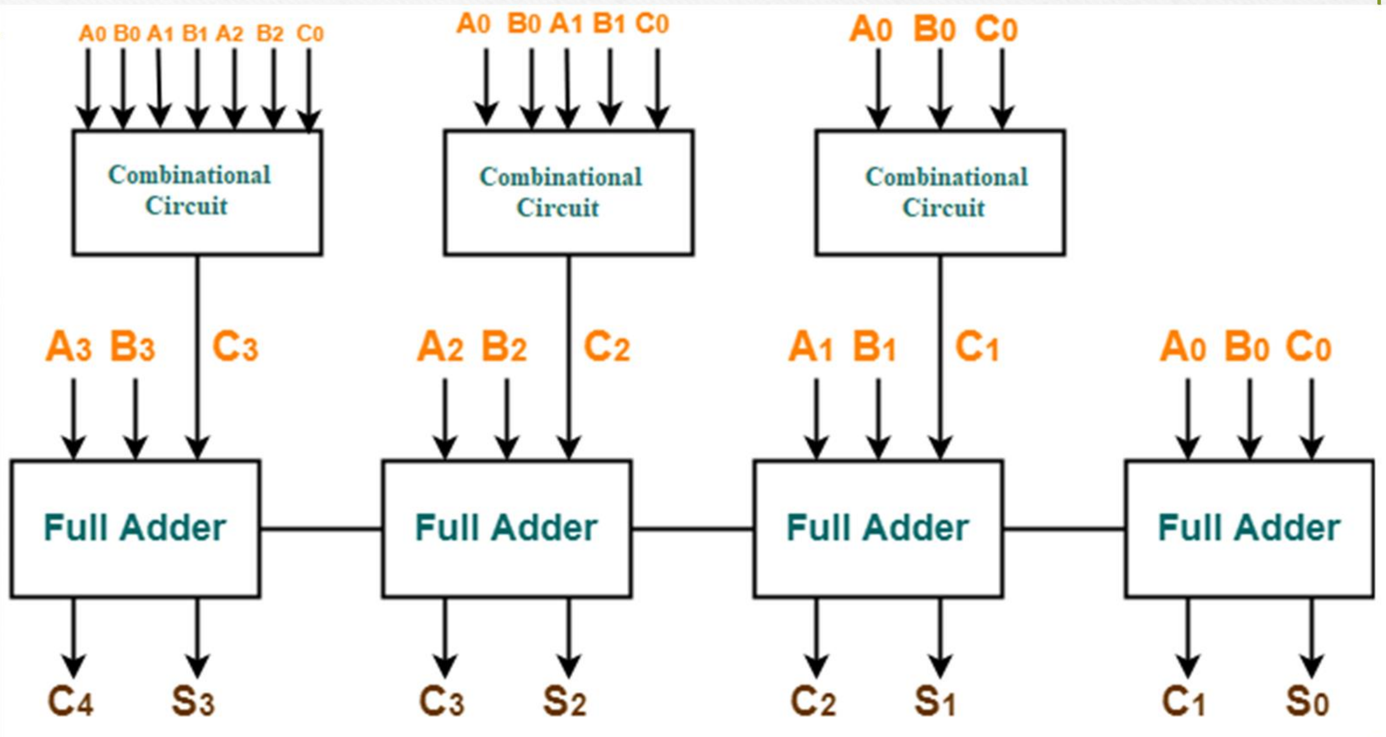
The carry-in of any stage full adder depends only on the following two parameters:

- Bits being added in the previous stages.
- Carry-in provided in the beginning.

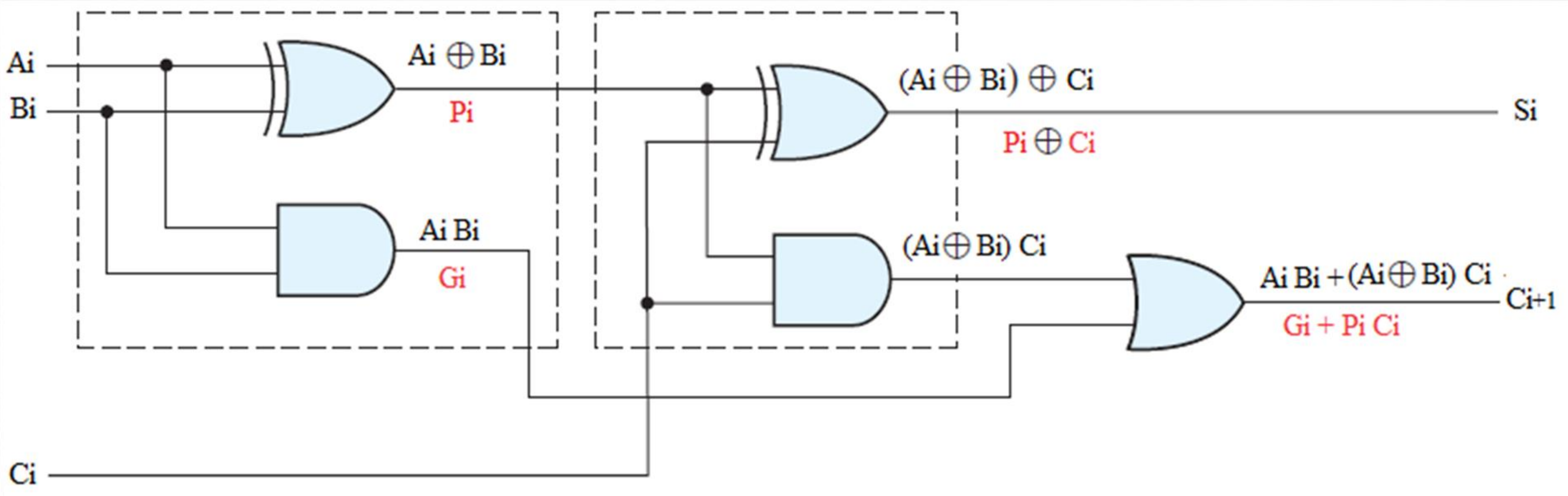
For the carry-look ahead adder, these two parameters are always known from the beginning. So, any full adder need not wait until its carry-in is generated by its previous stage full adder.

Carry look ahead Binary Adder

To design a 4-bit carry-look ahead adder



Carry look ahead Binary Adder



The internal outputs are:

$$P_i = A_i \oplus B_i$$
$$G_i = A_i\, B_i$$

The final outputs are:

$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i\, C_i$$

Carry look ahead Binary Adder

As, $C_{i+1} = G_i + P_i\, C_i$, So the outputs carry for the 4 stages are:

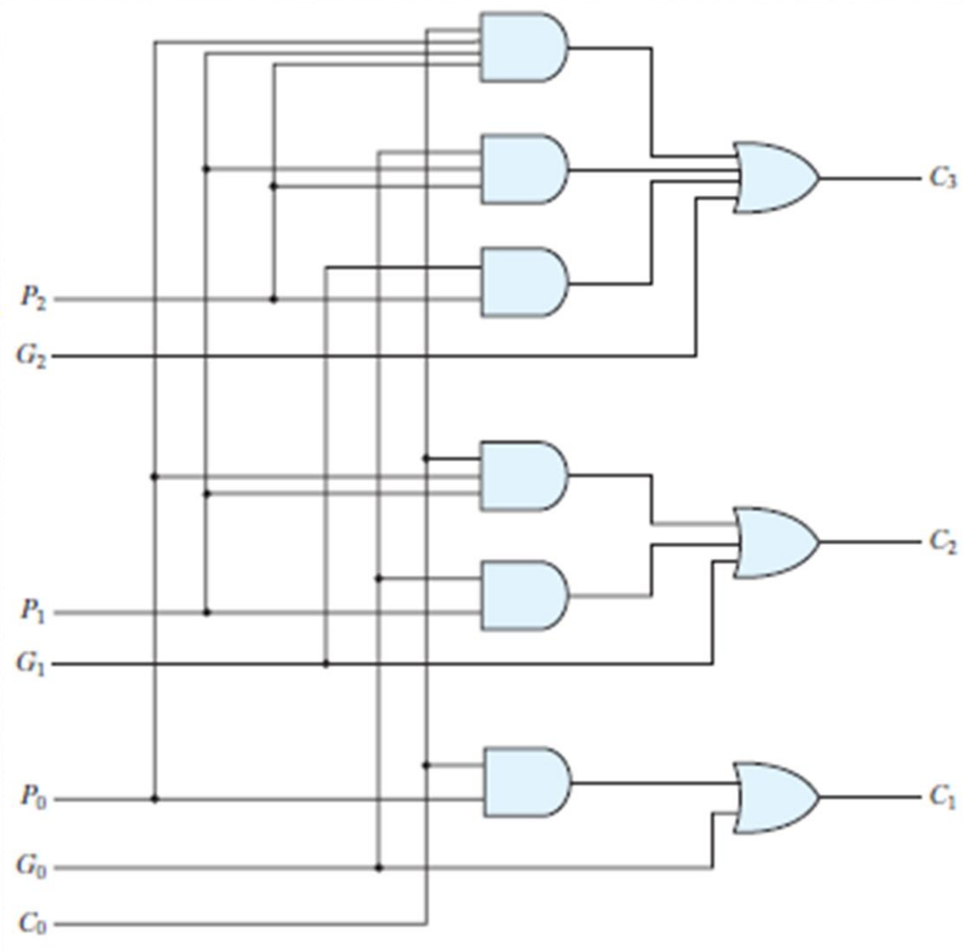
$$C_1 = G_0 + P_0\, C_0$$

$$C_2 = G_1 + P_1\, C_1 = G_1 + P_1\,(G_0 + P_0\, C_0) = G_1 + P_1\, G_0 + P_1\, P_0\, C_0$$

$$\begin{aligned} C_3 &= G_2 + P_2\, C_2 = G_2 + P_2\,(G_1 + P_1\, G_0 + P_1\, P_0\, C_0) \\ &= G_2 + P_2\, G_1 + P_2\, P_1 G_0 + P_2\, P_1\, P_0\, C_0 \end{aligned}$$

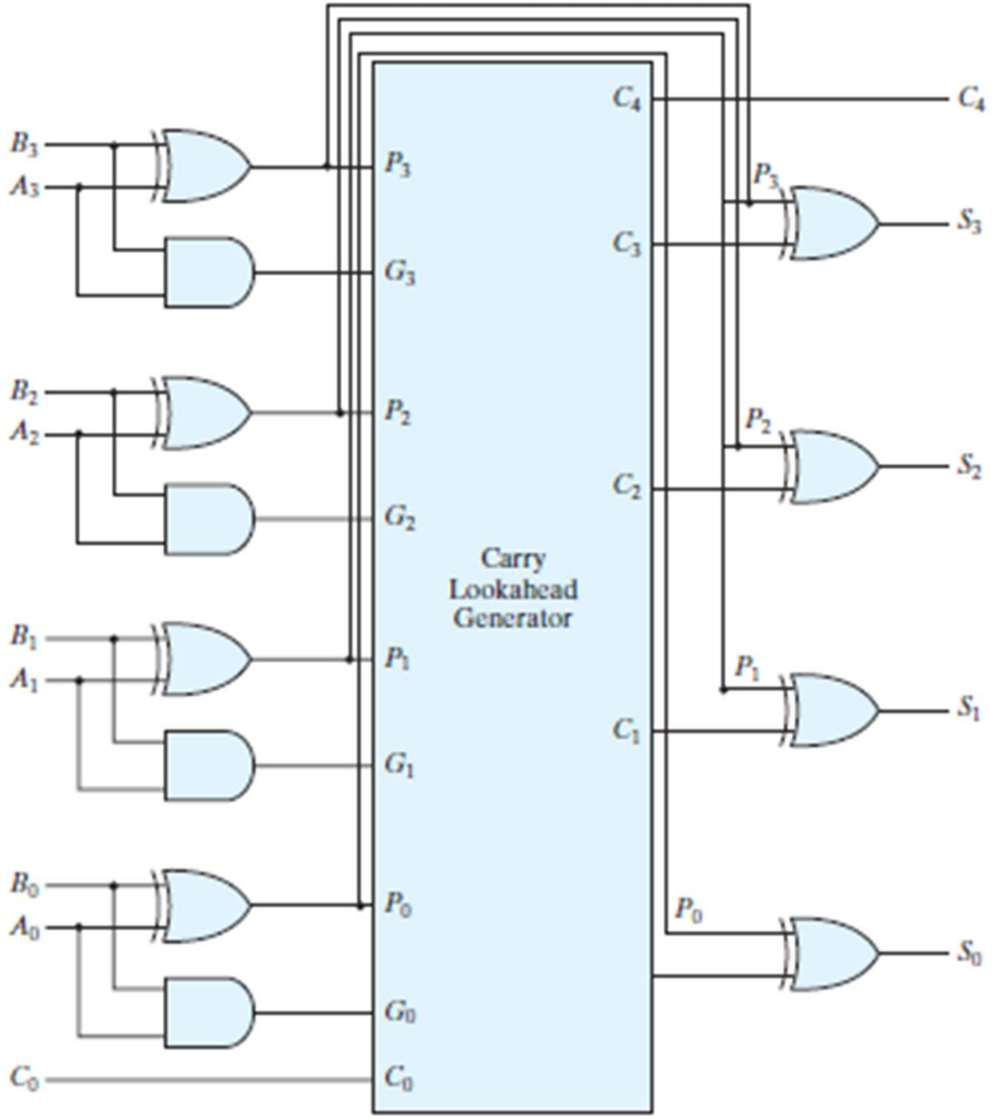
$$\begin{aligned} C_4 &= G_3 + P_3\, C_3 = G_3 + P_3\,(G_2 + P_2\, G_1 + P_2\, P_1 G_0 + P_2\, P_1\, P_0\, C_0) \\ &= G_3 + P_3\, G_2 + P_3\, P_2\, G_1 + P_3\, P_2\, P_1 G_0 + P_3\, P_2\, P_1\, P_0\, C_0 \end{aligned}$$

Carry look ahead Binary Adder



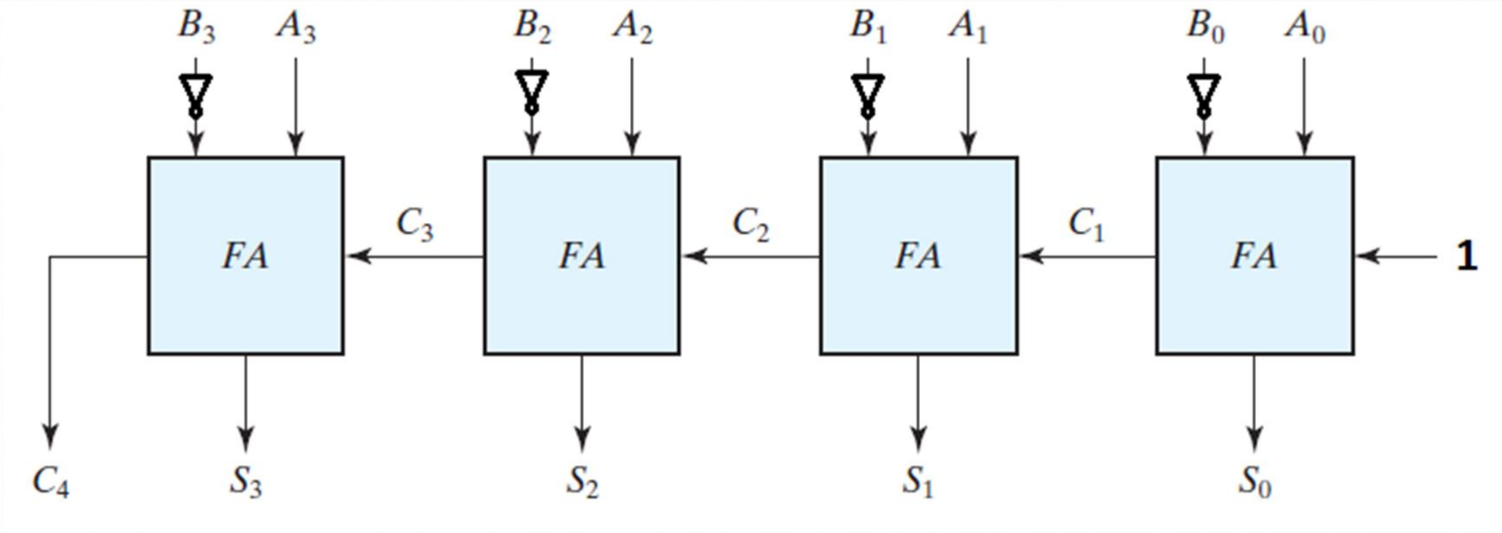
Complete the figure by drawing the circuit of C_4

Carry look ahead Binary Adder



Binary Subtractor

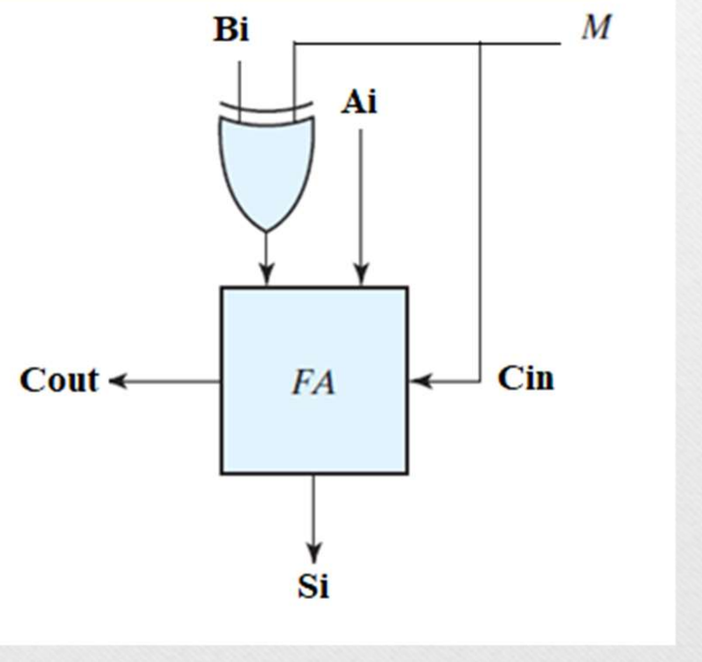
$$\mathbf{A - B = A + \overline{B} + 1}$$



Binary Adder - Subtractor

This circuit acts as an adder or subtractor according to the value of M.

$$S=(A+(B\oplus M))+M$$

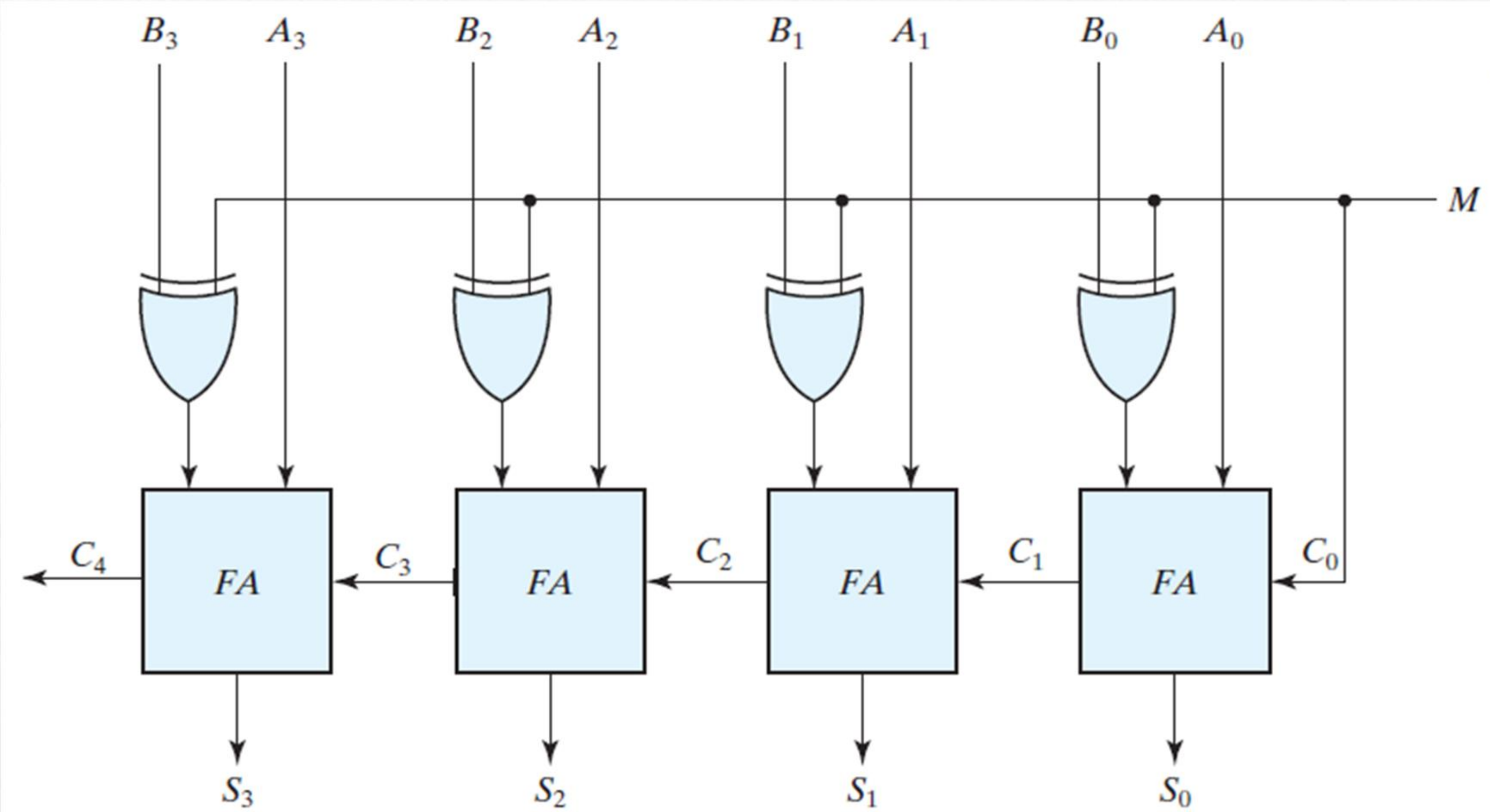


Binary Adder - Subtractor

$$S=(A+(B\oplus M))+M$$

M=0	M=1
$S= (A+(B\oplus 0))+0$	$S= (A+(B\oplus 1))+1$
$= A+B$	$= A+\overline{B}+1$
	$= A-B$
Addition	Subtraction

Binary Adder - Subtractor



Binary Multiplier

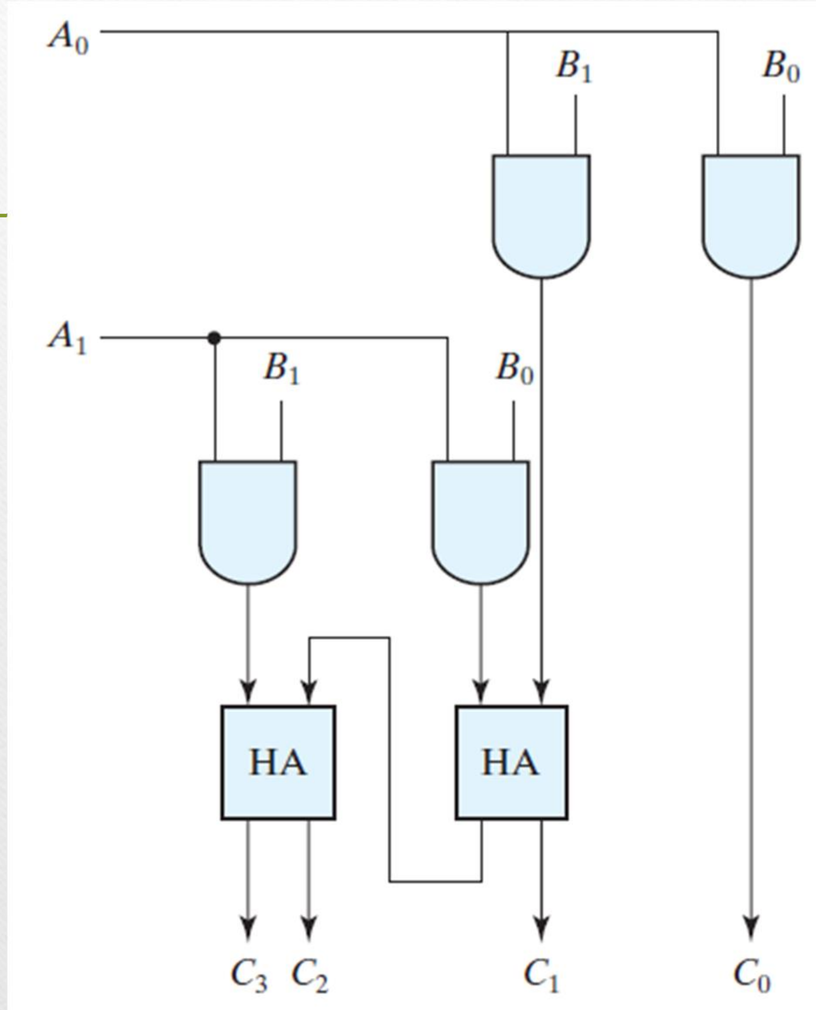
A binary multiplier is digital circuit, to multiply two binary numbers. It is built using binary adders.

Binary multiplication is similar to decimal multiplication.

Note that multiplication of two bits is equivalent to ANDed the two bits.

2-bit Binary Multiplier

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



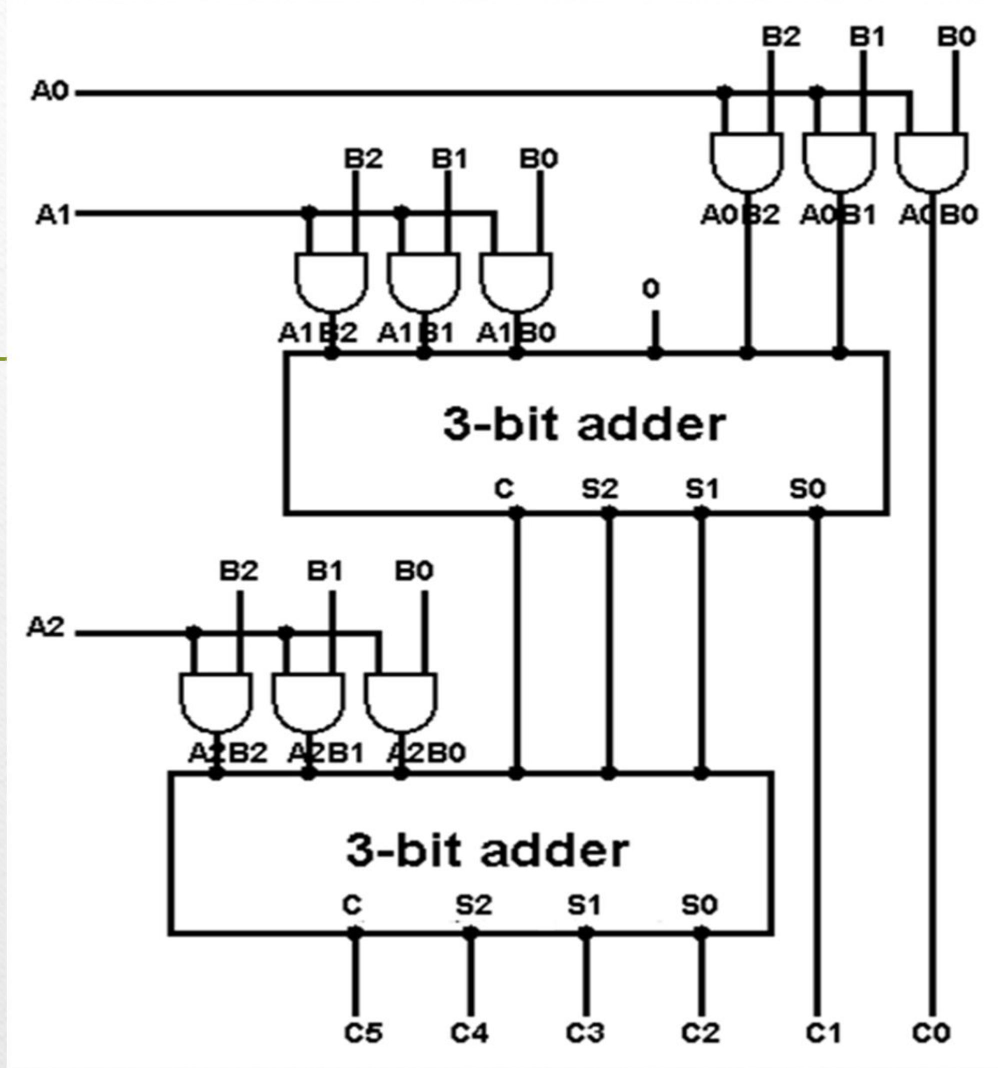
3-bit Binary Multiplier

Using two 3-bit adder and simple gates you may need to design a 3-bit Binary Multiplier.

				B_2	B_1	B_0
			x	A_2	A_1	A_0
				A_0B_2	A_0B_1	A_0B_0
	+		A_1B_2	A_1B_1	A_1B_0	
		A_2B_2	A_2B_1	A_2B_0		
C_5		C_4	C_3	C_2	C_1	C_0

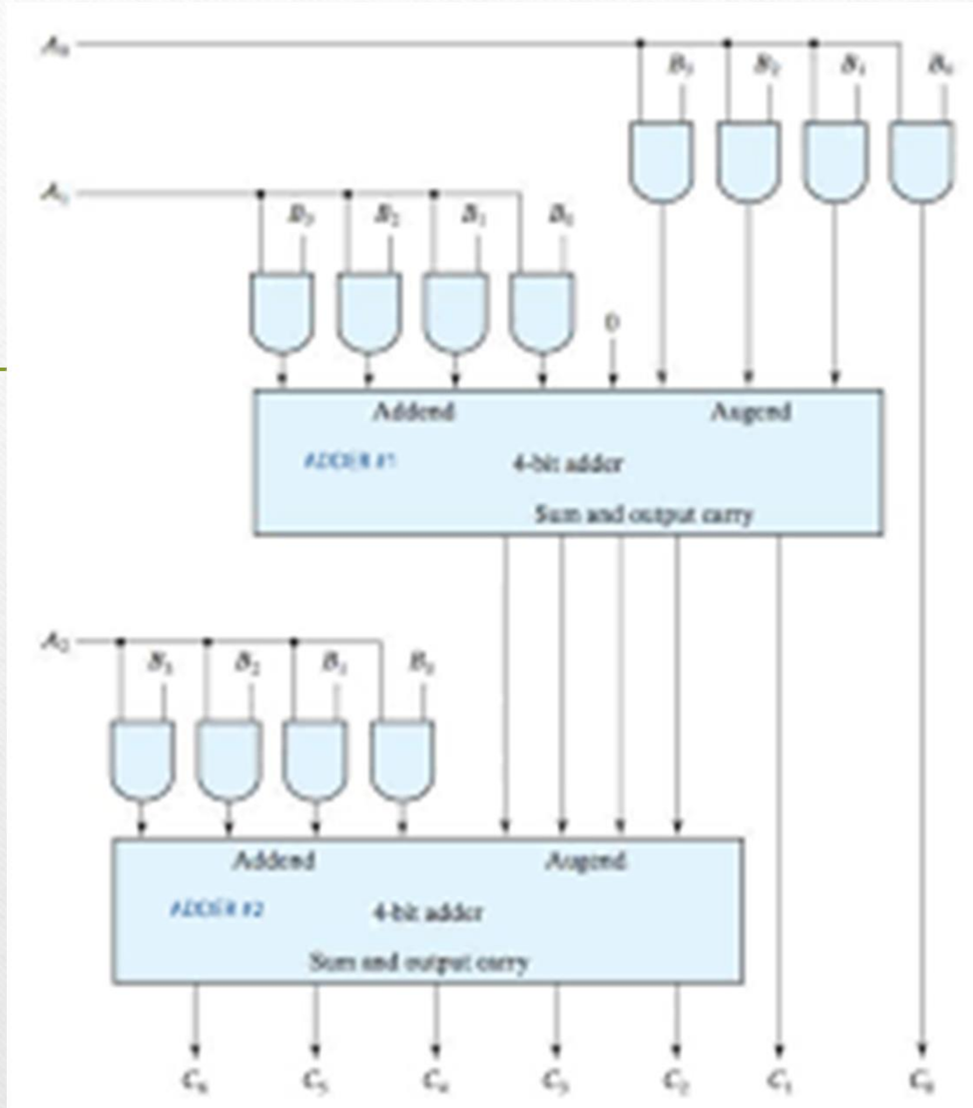
3-bit Binary Multiplier

			B_2	B_1	B_0
		x	A_2	A_1	A_0
			A_0B_2	A_0B_1	A_0B_0
+		A_1B_2	A_1B_1	A_1B_0	
	A_2B_2	A_2B_1	A_2B_0		
C_5	C_4	C_3	C_2	C_1	C_0



4x3 Bit Binary Multiplier

	B_3	B_2	B_1	B_0
x		A_2	A_1	A_0



Multipliers

In general

m-bit x n-bit multiplier needs:

- **n-1 (m-bit adders)**
- **mxn 2-input ANDs.**

Building of a Simple Equation

<u>Note:</u>

$X)_2$*2 is done by just shift left X, i.e. Adding 0 to the right

Ex: 100 * 2 = 1000

100*4=100*2*2=1000

Added Zeros is according to the power with 2

In General $A_3A_2A_1A_0$ *2 = $A_3A_2A_1A_00$ $(2=2^1)$

$A_3A_2A_1A_0$ *4 = $A_3A_2A_1A_000$ $(4=2^2)$

$A_3A_2A_1A_0$ *8 = $A_3A_2A_1A_0000$ $(8=2^3)$

…. etc

Building of a Simple Equation

<u>Example 3</u>

Using one 4-bit adder and one half adder to design a circuit that implements the following function Y=25X, where X is 4-bit binary number.

Note that doubling a binary number leads to left shifting this number and adding 0 to the left significant bit.

i.e. if $X= X_3\,X_2\,X_1\,X_0$, $2X= X_3\,X_2\,X_1\,X_0\,0$

Example if X= 1011 then 2X=10110

Building of a Simple Equation

(Example 3)

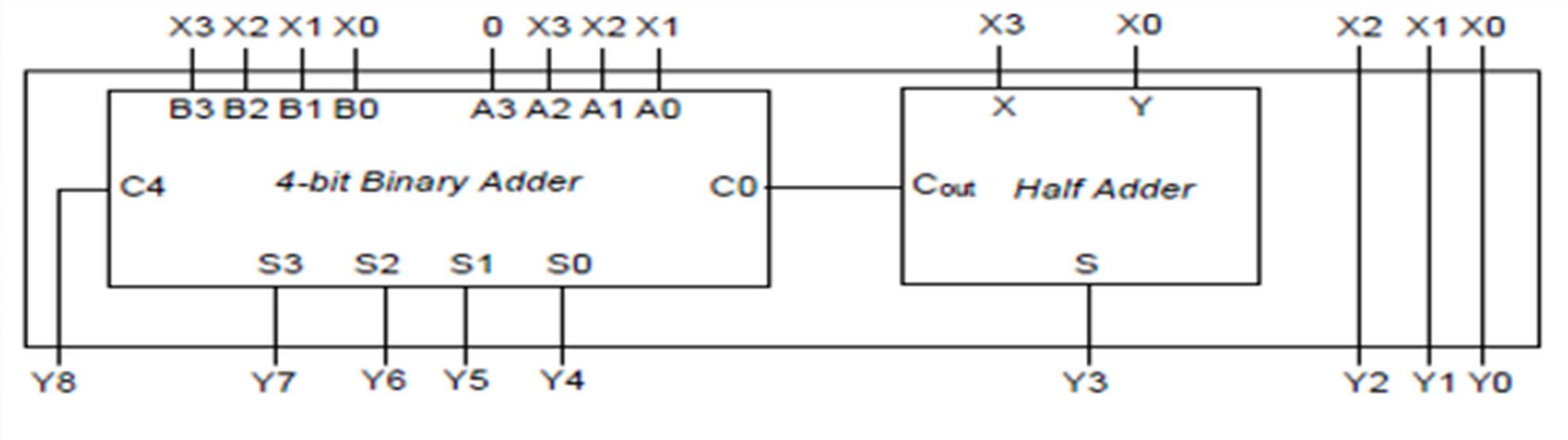
$Y=25X=16X+8X+X$

$Y=X_3\,X_2\,X_1\,X_0\,0000 + X3\,X_2\,X_1\,X_0\,000 + X3\,X_2\,X_1\,X_0$

C4	C3	C2	C1	C0				
	X_3	X_2	X_1	X_0	0	0	0	0
	0	X_3	X_2	X_1	X_0	0	0	0
					X_3	X_2	X_1	X_0
Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0

Building of a Simple Equation

(Example 3)



Building of a Simple Equation

<u>Exercise 4:</u>

Using one 4-bit adder and one full adder to design a circuit that implements the following function Y=25X+8, where X is 4-bit binary number.