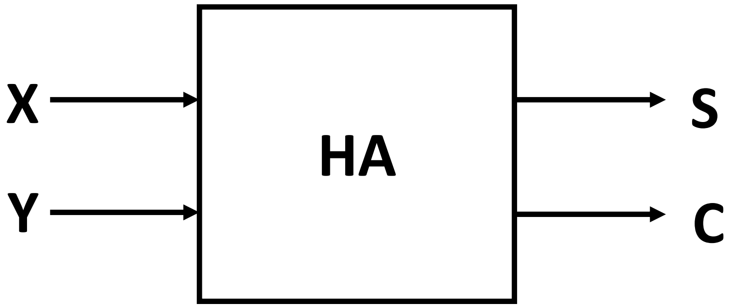
Adder & Subtractor

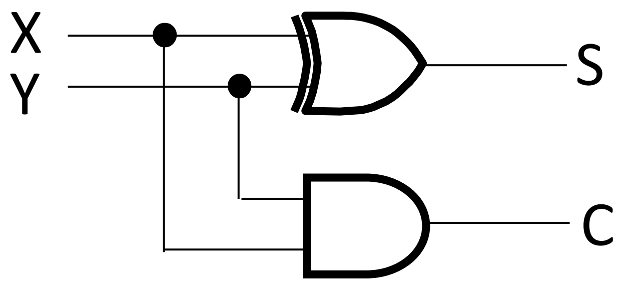
Nahin Ul Sadad
Lecturer
CSE, RUET

Adder Overview

Half Adder



X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

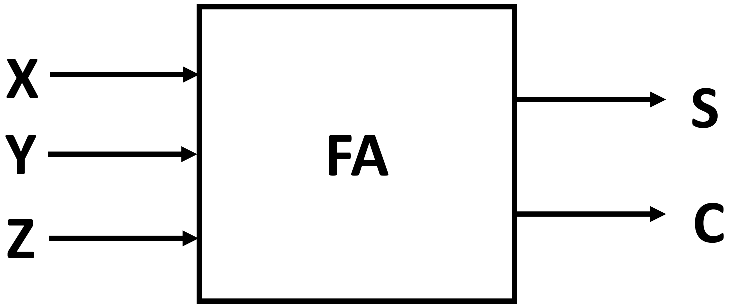


$$S = X \cdot \overline{Y} + \overline{X} \cdot Y = X \oplus Y$$

$$C = X \cdot Y$$

Adder Overview

Full Adder

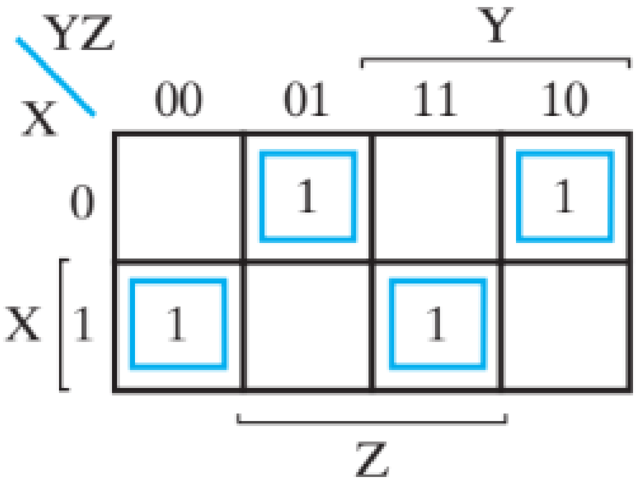


X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

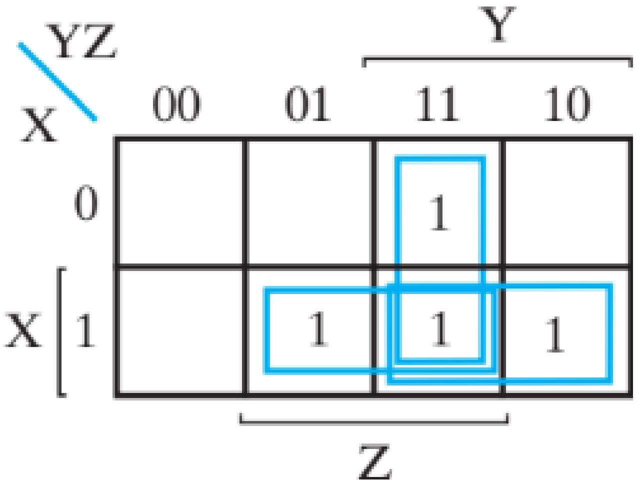
$$S = X\,\overline{Y}\,\overline{Z} + \overline{X}\,Y\,\overline{Z} + \overline{X}\,\overline{Y}\,Z + X\,Y\,Z$$
$$C = X\,Y + X\,Z + Y\,Z$$

Adder Overview

Adder circuit simplification



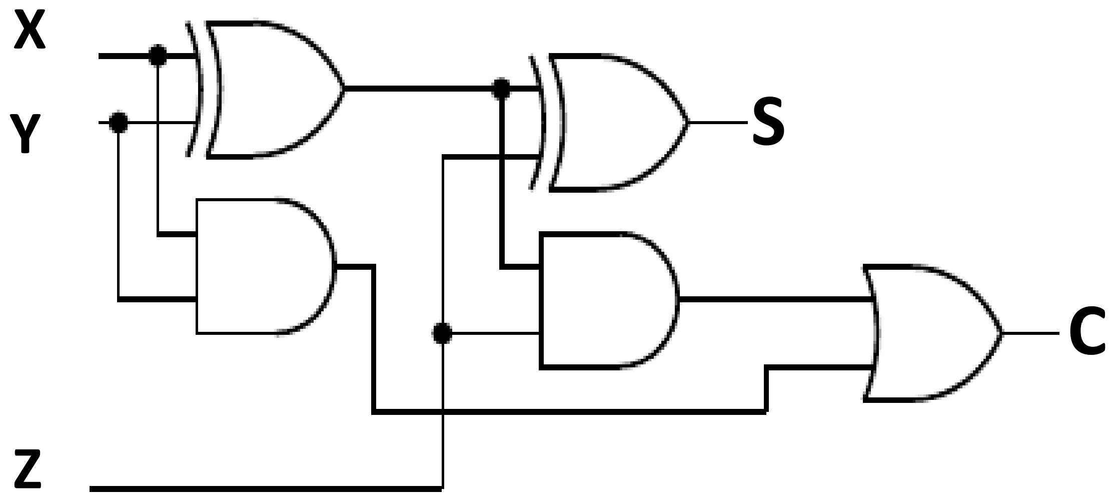
$$S = \bar{X}\bar{Y}Z + \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XYZ$$
$$= X \oplus Y \oplus Z$$



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

Adder Overview

Adder circuit simplification

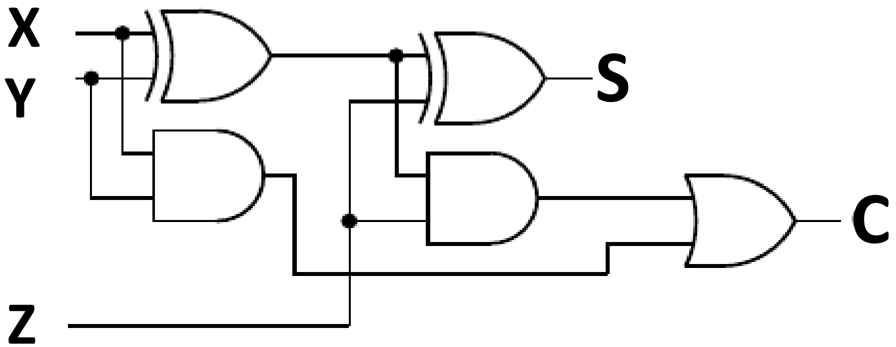


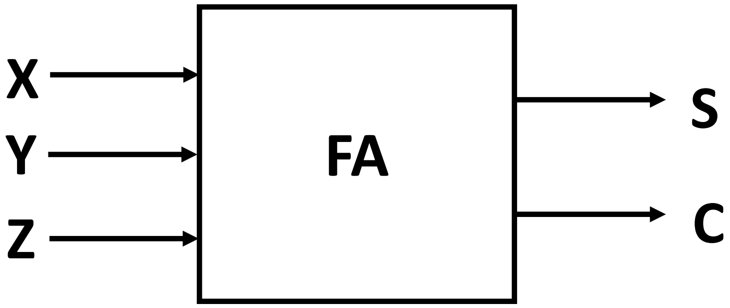
$$S = X \oplus Y \oplus Z$$

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

Adder Overview

Full Adder





X	Y	Z	C	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

$$S = X\,\overline{Y}\,\overline{Z} + \overline{X}\,Y\,\overline{Z} + \overline{X}\,\overline{Y}\,Z + X\,Y\,Z$$

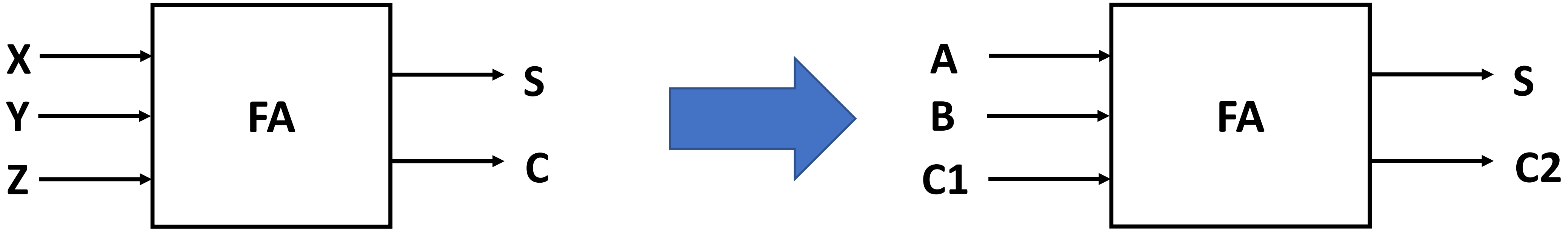
$$C = X\,Y + X\,Z + Y\,Z$$

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

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

Adder Overview

Full Adder



Remember this is only 1-bit Adder!

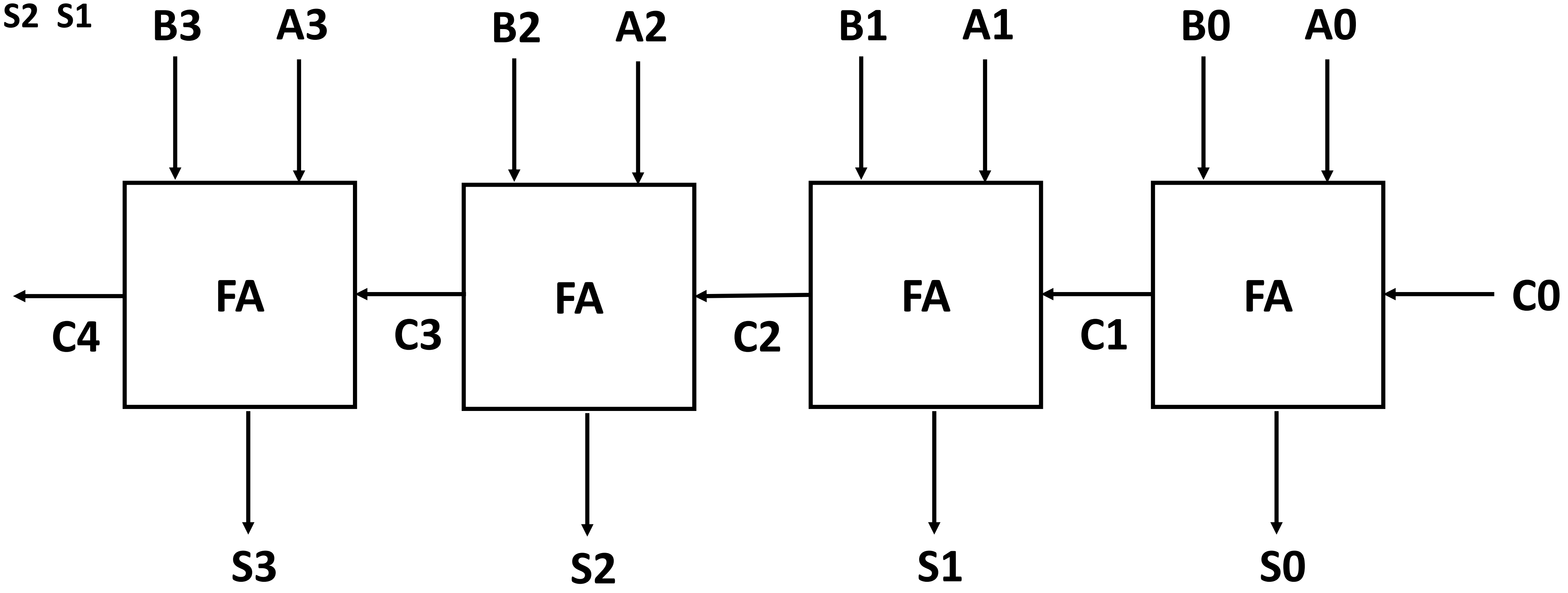
Q: How can we build 2-bit/4-bit/…32-bit adder?

A: n-bit parallel adder!

Adder Overview

4-bit Parallel Adder

	A4	A3	A2	A1
	B4	B3	B2	B1
C5	S4	S3	S2	S1



4-bit Parallel Adder!

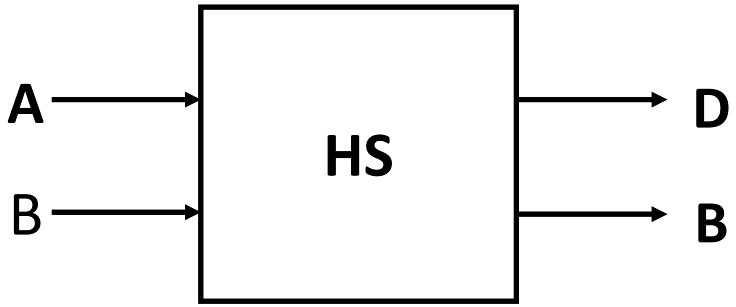
Unsigned Subtraction

Q: How can we build 2-bit/4-bit/…32-bit subtractor (unsigned)?

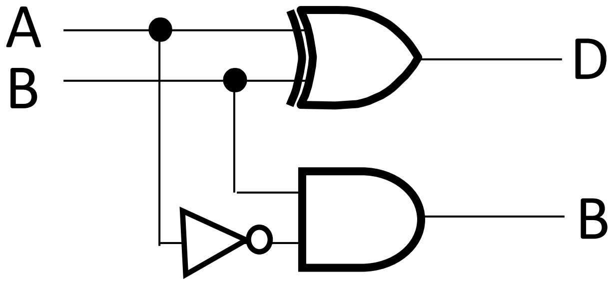
A: First, we have to build

1. Half-subtractor and
2. Full-subtractor

Half-subtractor



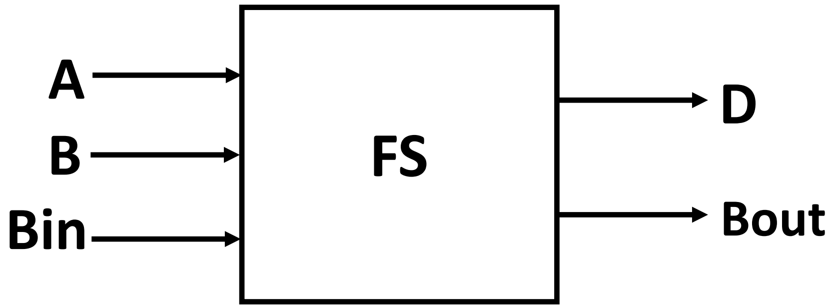
A	B	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



$$D = A \cdot \overline{B} + \overline{A} \cdot B = A \oplus B$$

$$C = \overline{A} \cdot B$$

Full Subtractor



A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$D = \overline{A}\,\overline{B}\,Bin + \overline{A}\,B\,\overline{Bin} + A\,\overline{B}\,\overline{Bin} + A\,B\,Bin$$

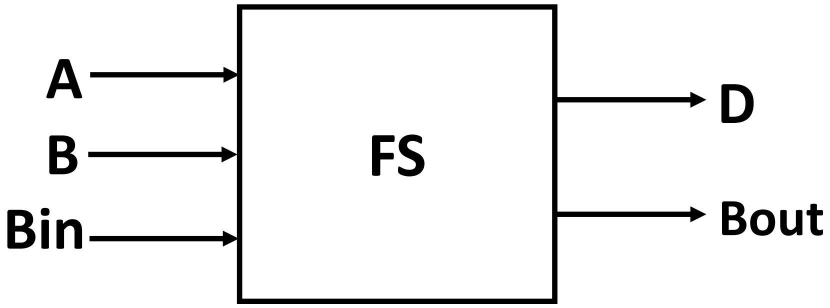
$$Bout = \overline{A}\,\overline{B}\,Bin + \overline{A}\,B\,\overline{Bin} + \overline{A}\,B\,Bin + A\,B\,Bin$$

Full Subtractor

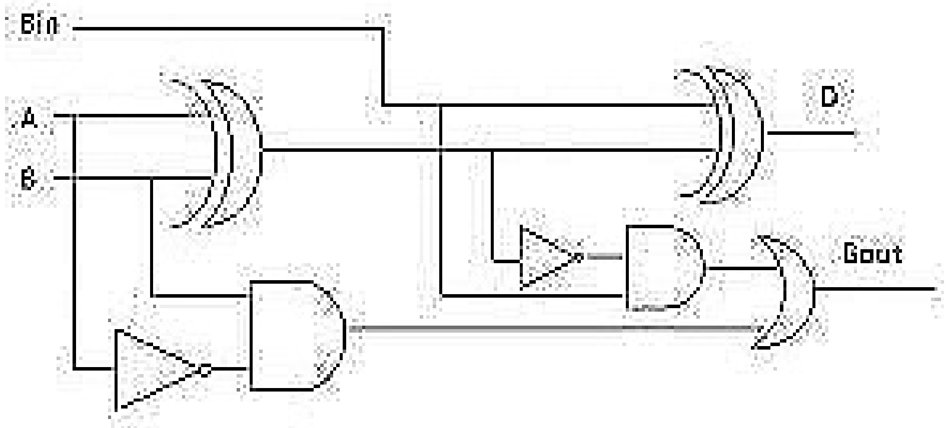
D = A'B'Bin + A'BBin' + AB'Bin' + ABBin
= Bin(A'B' + AB) + Bin'(AB' + A'B)
= Bin(A XNOR B) + Bin'(A XOR B)
= Bin (A XOR B)' + Bin'(A XOR B)
= Bin XOR (A XOR B)
= (A XOR B) XOR Bin

Bout = A'B'Bin + A'BBin' + A'BBin + ABBin
= Bin(AB + A'B') + A'B(Bin + Bin')
= Bin(A XNOR B) + A'B
= Bin (A XOR B)' + A'B

Full Subtractor



A	B	Bin	D	Bou
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



$$D = \bar{A}\,\bar{B}\,Bin + \bar{A}\,B\,\overline{Bin} + A\,\bar{B}\,\overline{Bin} + A\,B\,Bin$$

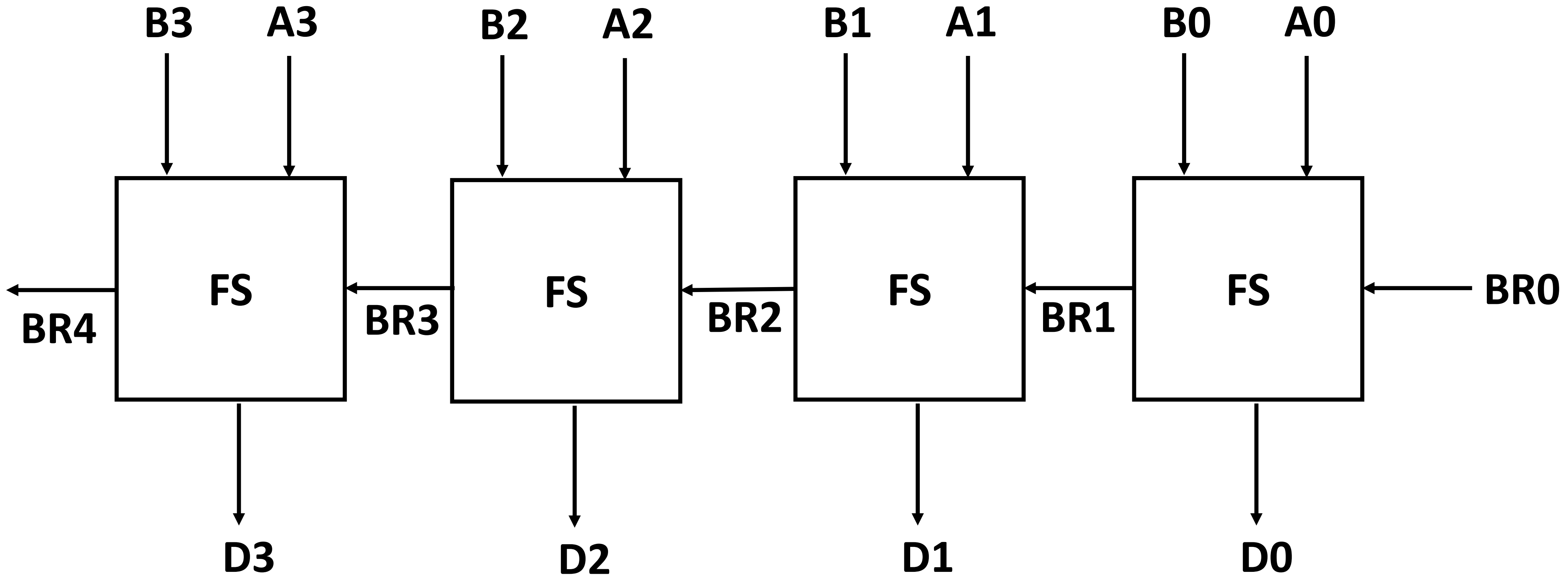
$$Bout = \bar{A}\,\bar{B}\,Bin + \bar{A}\,B\,\overline{Bin} + \bar{A}\,B\,Bin + A\,B\,Bin$$

$$D = A \oplus B \oplus Bin$$

$$Bout = Bin\,\overline{(A \oplus B)} + \bar{A}\,B$$

Q: How can we build 2-bit/4-bit/…32-bit unsigned subtractor?

A:



Signed Subtraction

Q: What are the limitations of Unsigned subtractor (built using half-subtractor and full-subtractor)?

A:

1. It cannot calculate signed subtraction like -4-2 etc. (For adders, there is no difference between signed and unsigned adders.)

2. It requires a separate circuit to perform this operations. It is possible to perform subtraction using Adder circuit.

Q: How can we perform signed subtraction?

A: It can be done using 2's complement!

Signed 2's compl (n=4)	decimal value
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	$7 = 2^{n-1} - 1$
1000	$-8 = -2^{n-1}$
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

When MSB = 0,
No of positive values = 2^{n-1}
Rage of positive values = 2^{n-1} to 0

When MSB = 1,
No of negative values = 2^{n-1}
Rage of negative values = -2^{n-1} to -1

Total number of values represented by n bits is $2^{n-1} + 2^{n-1} = 2^n$.

Q: How can to calculate negative number (For example : -4) using 2's complement?

A:

1. Calculate binary representation of its positive number:

$$(4)_{10} = (0100)_2$$

2. Calculate 1's complement: It means flipping/inverting its bits

$0100 \rightarrow 1011$ (Flipping its bits)

3. Calculate 2's complement: It means adding 1 to its 1's complement result

$$\begin{array}{r} 1011 \\ +1 \\ \hline 1100 \end{array}$$

Q: How can to calculate equivalent decimal value (For example : 1100) using 2's complement?

A: Just Perform 2's complement operation!

1. Calculate 1's complement: It means flipping/inverting its bits

$1100 \rightarrow 0011$ (Flipping its bits)

2. Calculate 2's complement: It means adding 1 to its 1's complement result

$$\begin{array}{r} 0011 \\ +1 \\ \hline 0100 \end{array}$$

3. Calculate its equivalent decimal:

$$(0100)_2 = (4)_{10}$$

Q: Calculate +4-3

A:

1. Binary representation of $+4$ is 0100
   Binary representation of -3 is 1101

2.

$$\begin{array}{r} 0100 \\ 1101 \\ \hline 0001 \end{array}$$

Carry = 1 (Ignore It)

3. Calculate its equivalent decimal:

$$(0001)_2 = (1)_{10}$$

Q: Calculate -4+3

A:

1. Binary representation of -4 is 1100
   Binary representation of $+3$ is 0011

2.

$$\begin{array}{r} 1100 \\ 0011 \\ \hline 1111 \end{array}$$

3. Calculate its equivalent decimal:

$$(1111)_2 = (-1)_{10}$$

Q: Calculate -4-5

A:

1. Binary representation of -4 is 1100
   Binary representation of -5 is 1011

2.

$$\begin{array}{r} 1100 \\ 1011 \\ \hline 0111 \end{array}$$

Carry = 1

3. Calculate its equivalent decimal:

$$(0111)_2 = (7)_{10}$$

But answer is Positive and it is 7 instead of -9!

This is called **Signed Overflow problem**.

Q: Calculate +4+5

A:

1. Binary representation of $+4$ is 0100
   Binary representation of $+5$ is 0101

2.

$$
\begin{array}{r}
0100 \\
0101 \\
\hline
1001
\end{array}
$$

Carry = 1

3. Calculate its equivalent decimal:

$$(1001)_2 = (-7)_{10}$$

But answer is Negative and it is -7 instead of +9!
This is called **Signed Overflow problem**.

Signed Overflow occurs when

Same signed numbers are added but result is opposite sign.

For example: +4+5 = -7
-4-5 = +7

We can design a Flag Register to flag signed overflow problem!

Q: How can we perform subtraction using Adder circuit for 1-bit numbers (For example: 0-1)?

A:

Remember this is 1-bit Adder.

It means it can only add not subtraction!

We can write,

$$0 - 1 = 0 + (-1) = 0 + \text{2's complement of } 1!$$
$$= 0 + \text{1's complement of } 1 + 1$$
$$= 0 + \text{Inverting/Flipping of } 1 + 1$$
$$= 0 + {\sim}1 + 1$$
$$= 0 + 0 + 1$$

Binary representation of 0 is 0
Binary representation of 1 is 1

Inverting means using NOT gate!

Signed Subtraction using Full Adder

$$A - B = A + \sim B + 1 = A + NOT(B) + 1(C1)$$

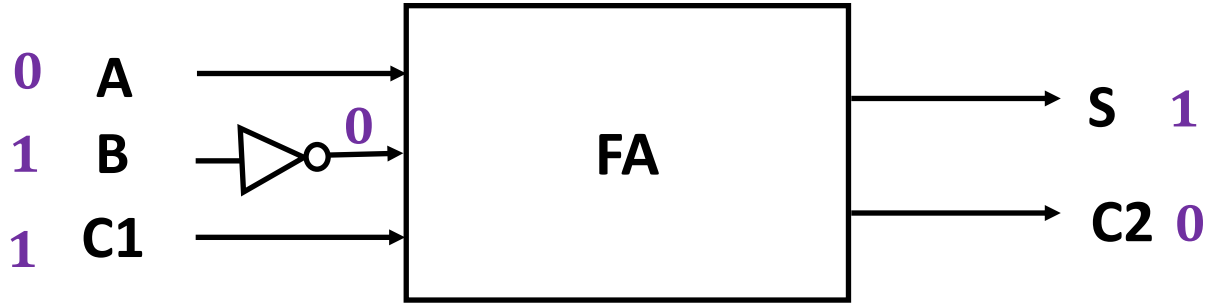
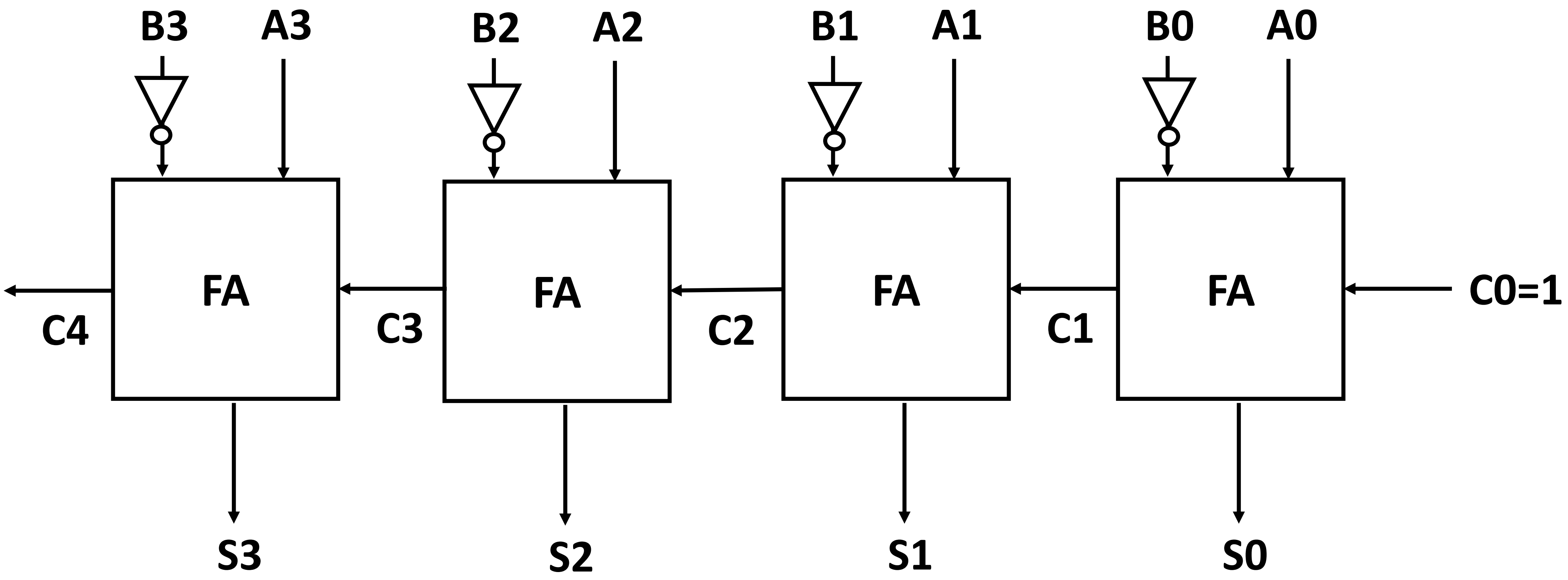


Figure: 1-bit Subtractor (Signed Subtraction)

Q: How can we build 2-bit/4-bit/…32-bit (signed) subtractor?

A: 4-bit Parallel Subtractor (Using Adder circuit)



Q: How can we combine addition and subtraction in same Adder circuit?

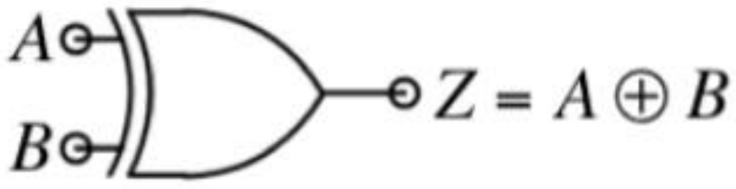
A: Using XOR gate!

XOR is also called Programmable Inverter!

If $A = 0, Z = B$

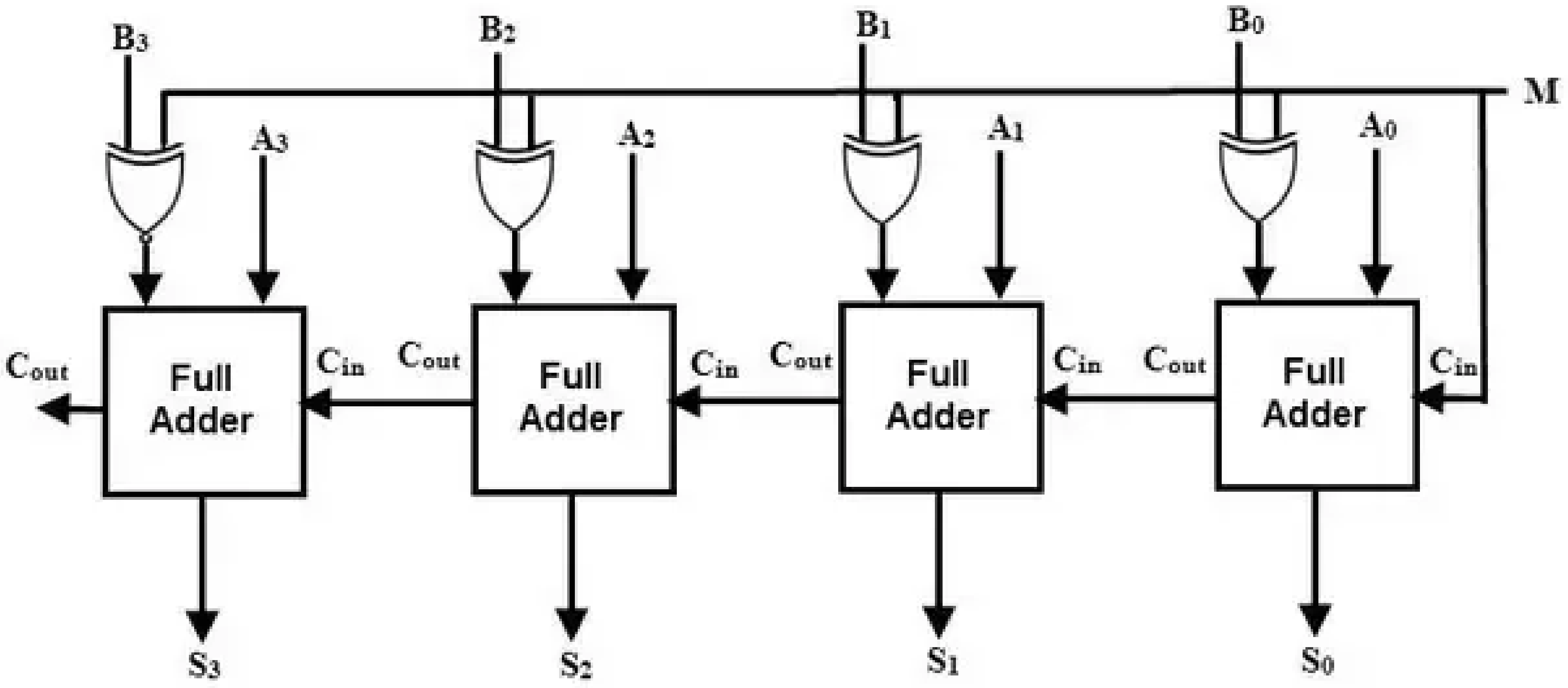
If $A = 1, Z = \sim B$

XOR (exclusive OR)

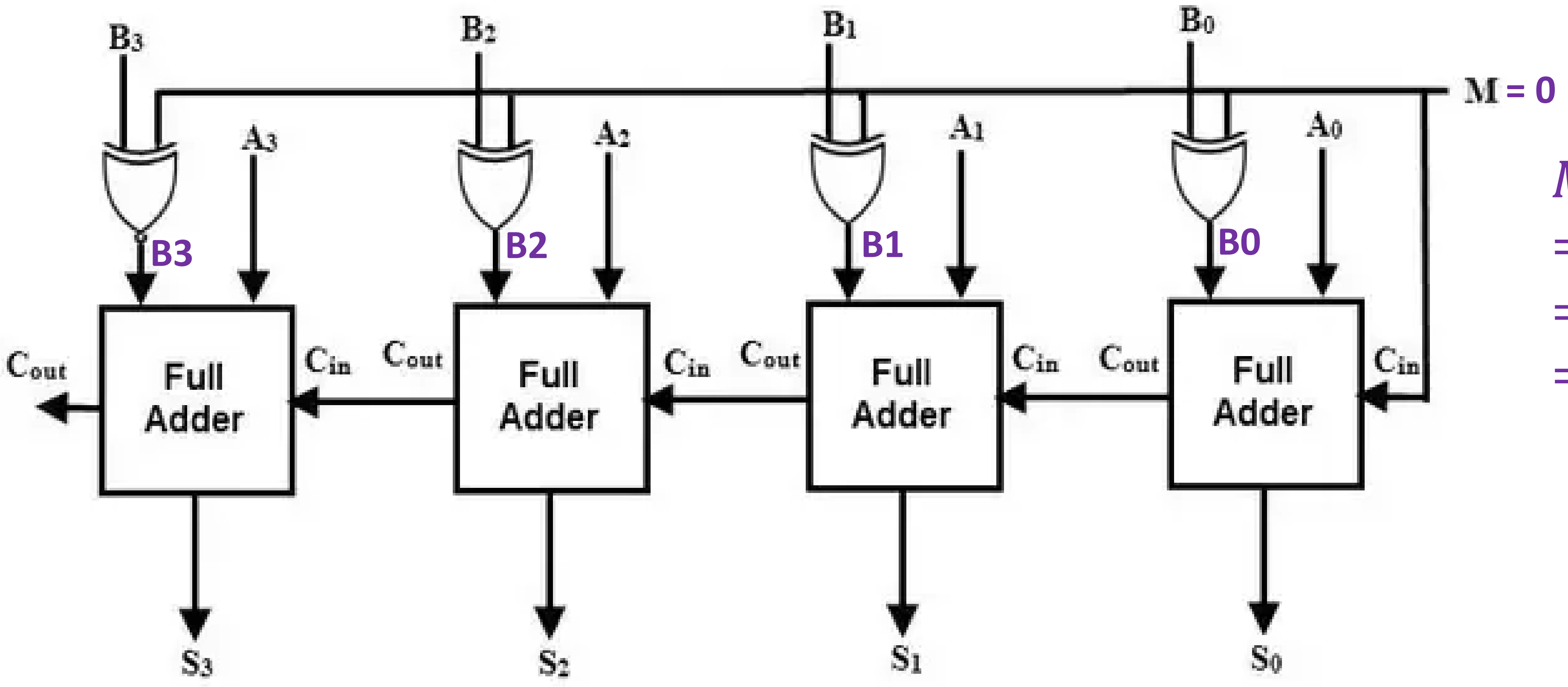


A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

4-bit Parallel Adder/Subtractor

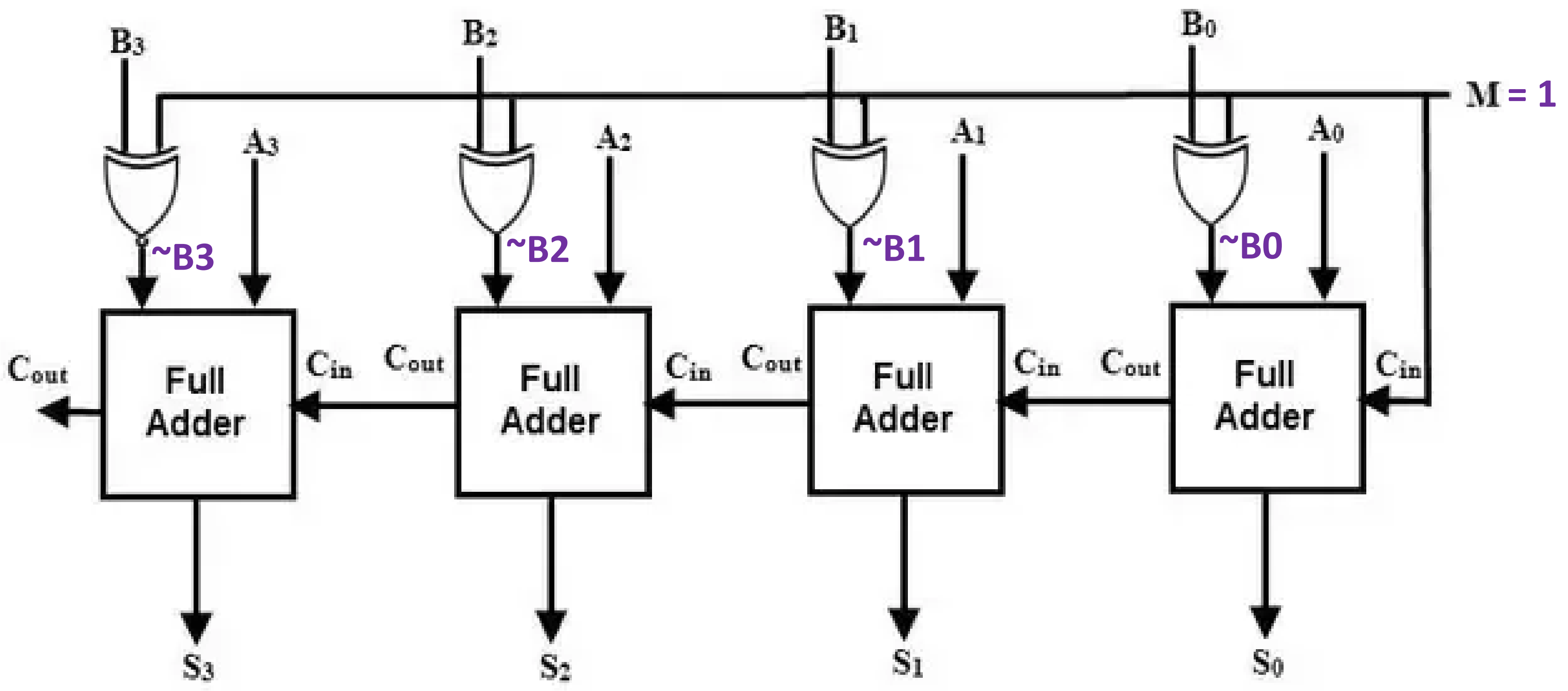


4-bit Parallel Adder/Subtractor



$$M \oplus B_n = M.\overline{B_n} + \overline{M}.B_n = 0.\overline{B_n} + \overline{0}B_n = B_n$$

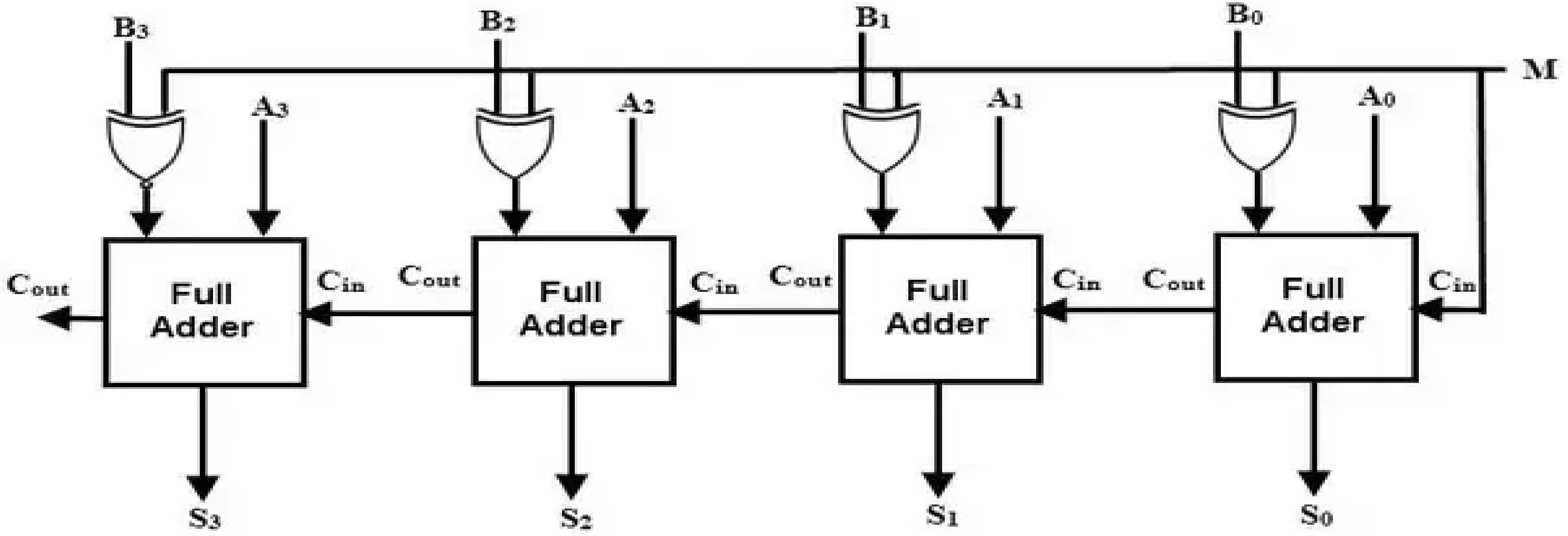
4-bit Parallel Adder/Subtractor



$$M \oplus B_n$$
$$= M.\overline{B_n} + \overline{M}.B_n$$
$$= 1.\overline{B_n} + \overline{1}B_n$$
$$= \overline{B_n}$$

Q: What are the limitations of parallel adder/subtractor?

A: Parallel adder/subtractor is very slow. Because:



4th FA has to wait for
3rd FA to get value of C3
Which is waiting for
2nd FA to get value of C2
Which is waiting for
3rd FA to get value of C1

3rd FA has to wait for
2nd FA to get value of C2
Which is waiting for
1st FA to get value of C1

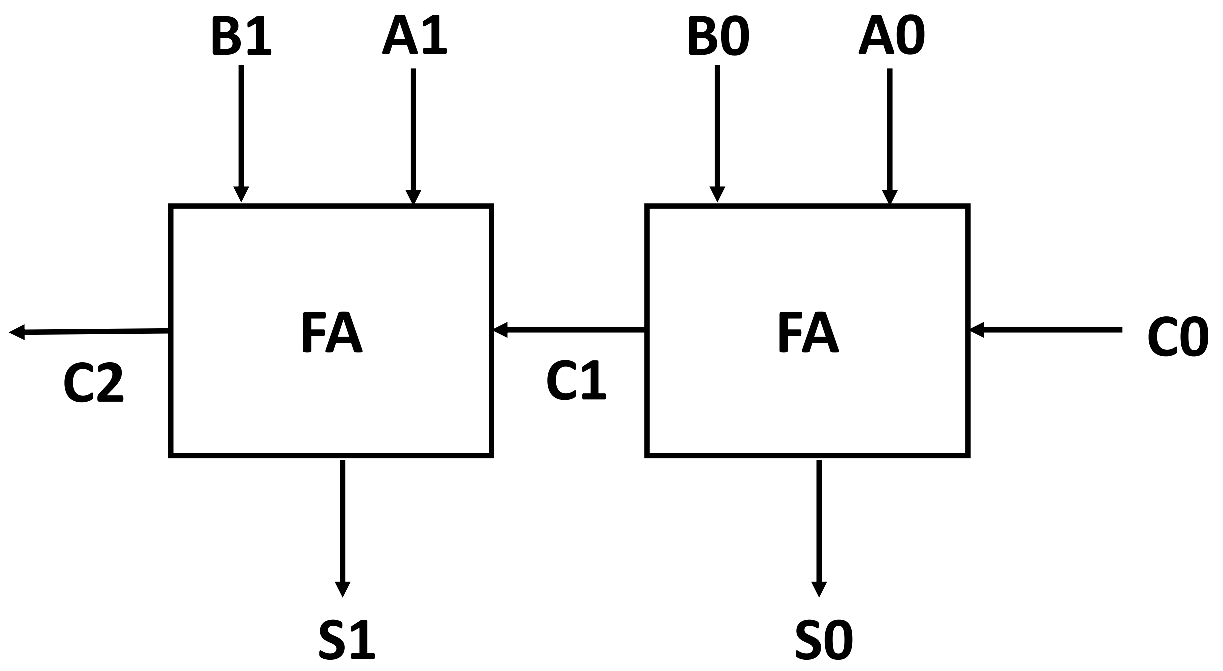
2nd FA has to wait for
1st FA to get value of C1

So, Parallel Adder/Subtractor is slower!

Q: How can we solve this problem?

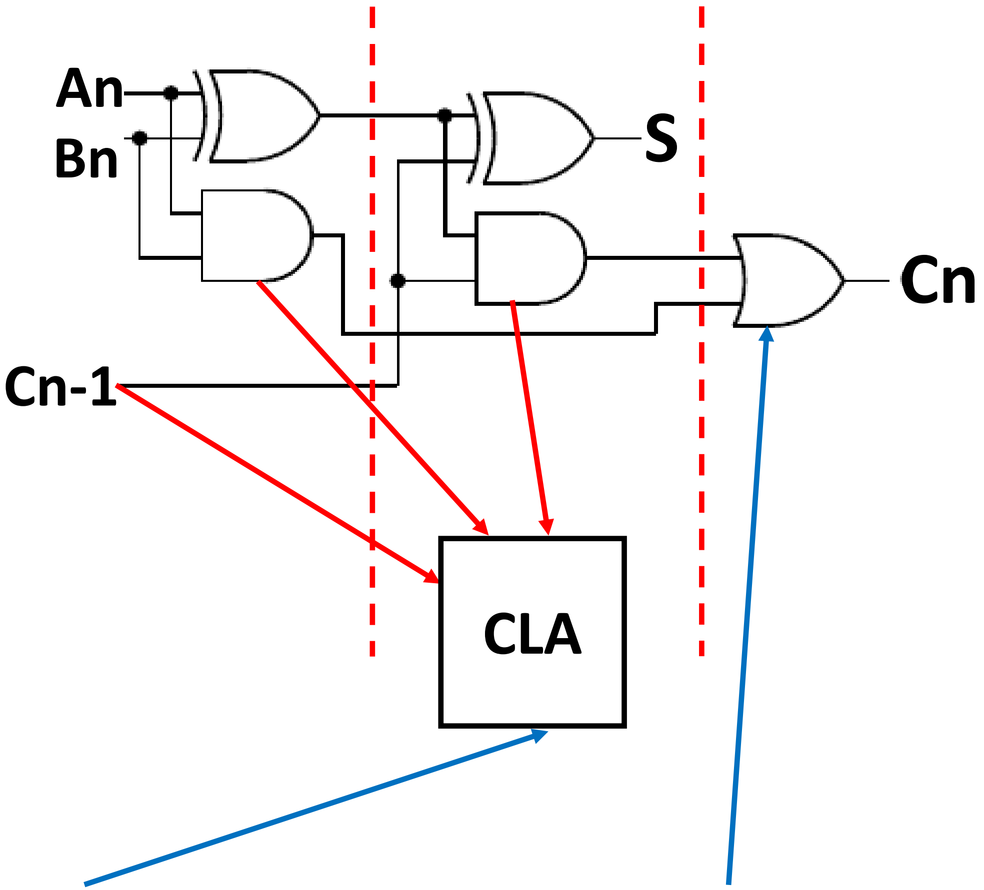
A: We can use another variant of Parallel adder which is called
Carry Look Ahead Adder!

A: First consider simple 2-bit Parallel Adder



Because waiting for carry is bottleneck of this design.
What we can do is calculate carry as early as possible!

A: First consider simple 2-bit Parallel Adder



This design is faster than that design.

$$S = An \oplus Bn \oplus Cn\text{-}1$$

$$Cn = An\,Bn + (An \oplus Bn)\,Cn\text{-}1$$

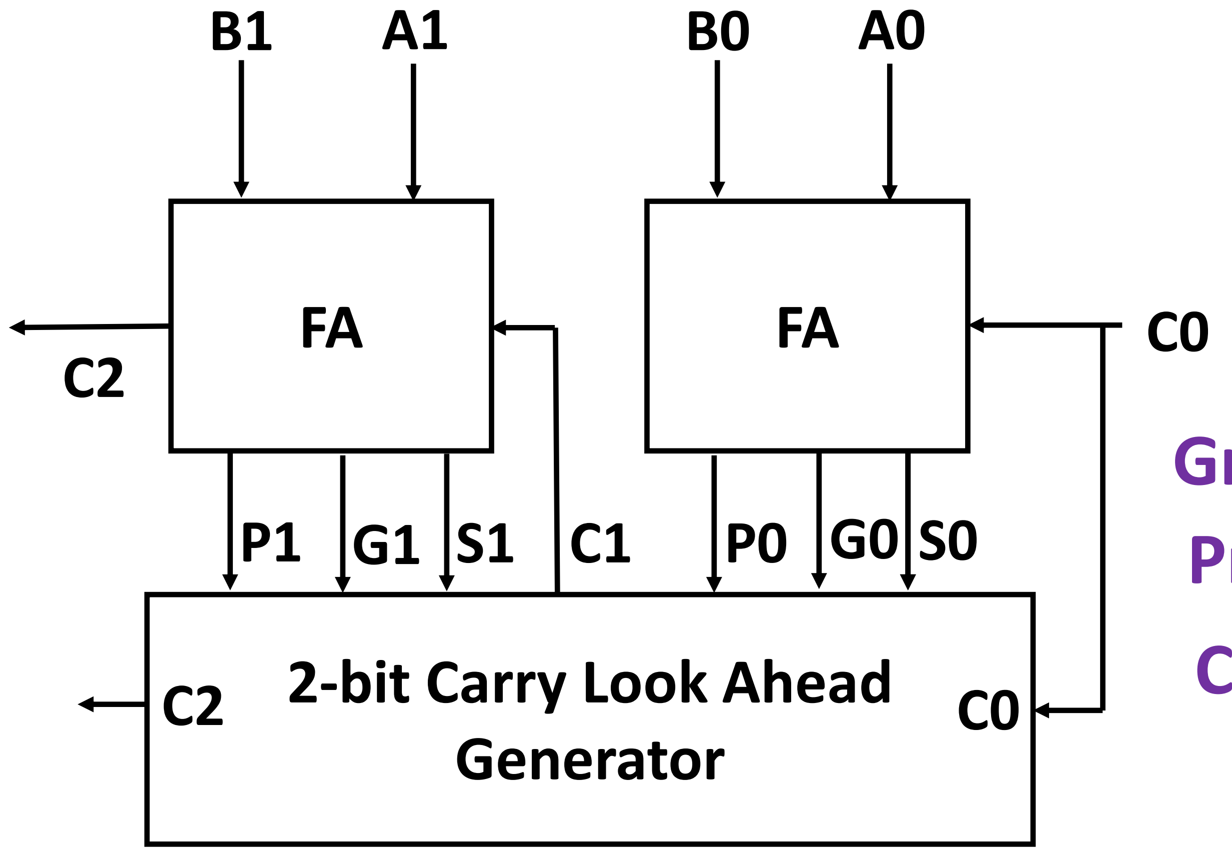
Let,

$$Gn = An\,Bn$$

$$Pn = An \oplus Bn$$

$$Cn = Gn + Pn\,Cn\text{-}1$$

A: simple 2-bit Parallel Adder

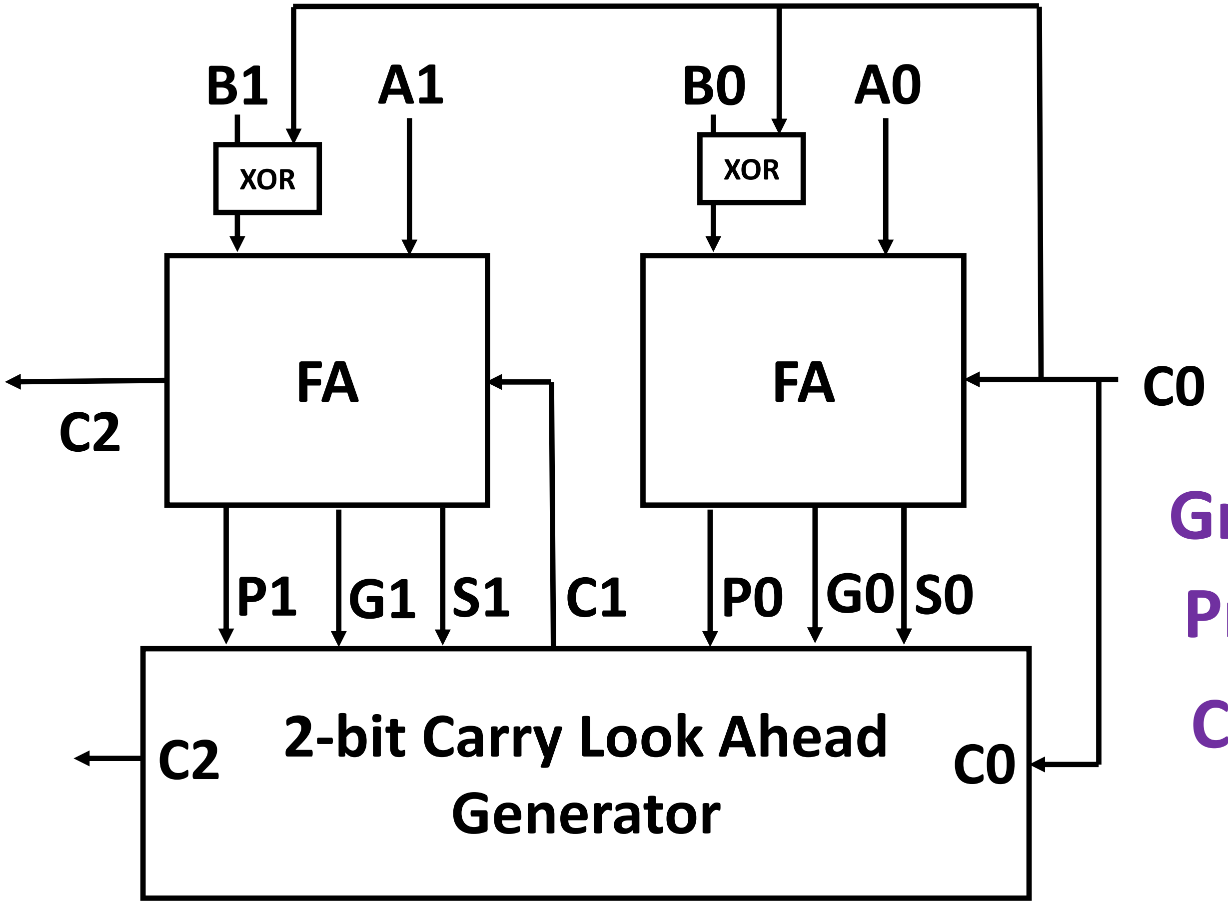


$$G_n = A_n B_n$$

$$P_n = A_n \oplus B_n$$

$$C_n = G_n + P_n C_{n-1}$$

A: Similarly 2-bit Parallel Adder/Subtractor



$$G_n = A_n B_n$$

$$P_n = A_n \oplus B_n$$

$$C_n = G_n + P_n C_{n-1}$$

Similarly, Create 4-bit Carry Look Ahead Adder/Subtractor!

Example: Adder

Question: Design a 4 bit signed adder (normal) and show output of each circuit in when X = 1001 and Y= 1111.

Answer:

X	Y	Z	C	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

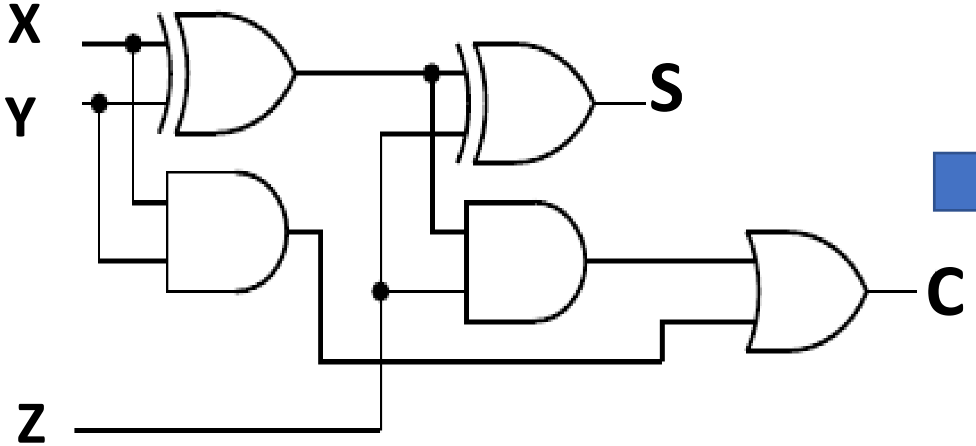


Figure: 1-bit Full Adder Circuit

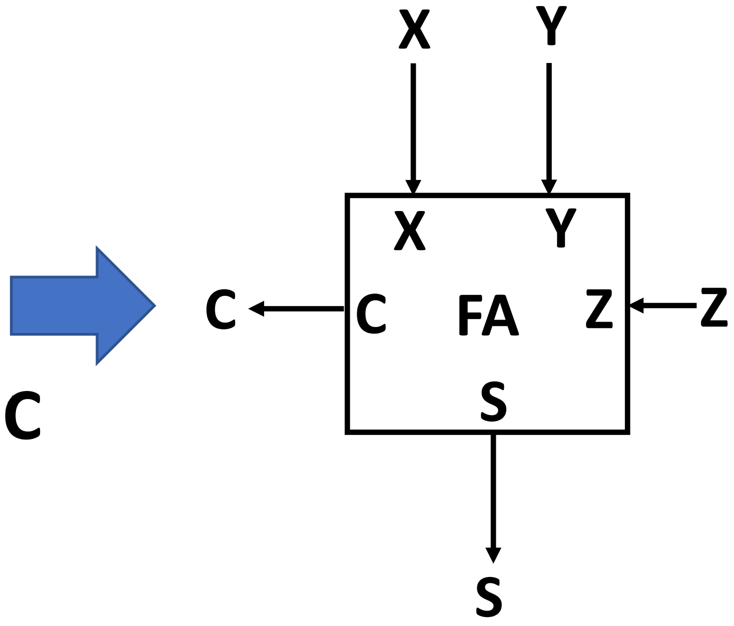


Figure: 1-bit Full Adder Chip

$$S = X\overline{Y}\,\overline{Z} + \overline{X}Y\overline{Z} + \overline{X}\,\overline{Y}Z + XYZ$$
$$C = XY + XZ + YZ$$

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

Example: Adder

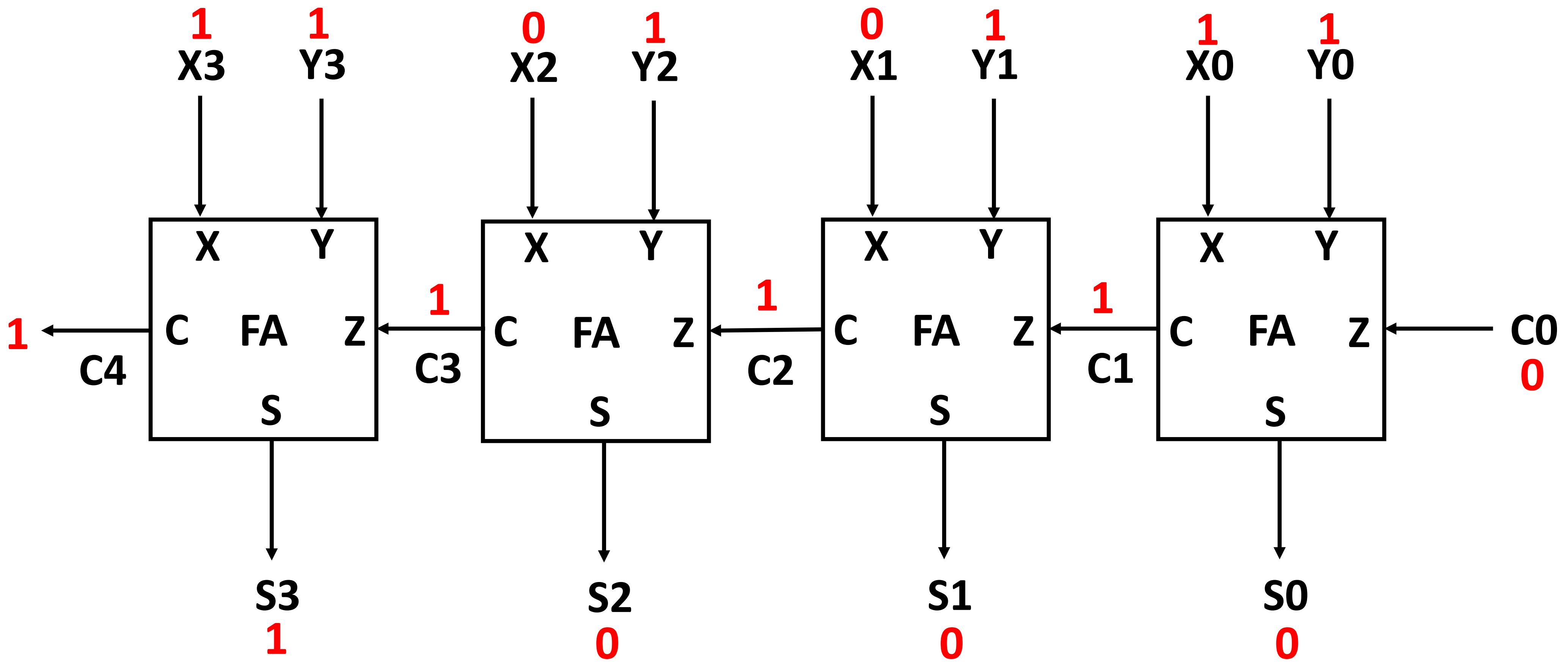


Figure: 4-bit Full Adder Circuit with output when X=1001 and Y=1111

Exercises

1. Consider

$$X = 11011/1111/110/11 \text{ (5-bit/4-bit/3-bit/2-bit)}$$
$$Y = 11011/1111/110/11 \text{ (5-bit/4-bit/3-bit/2-bit)}$$

   i. Calculate X+Y (Unsigned/Signed)
   ii. Calculate X-Y (Unsigned/Signed)

2. Calculate 1010-0100/1010+0100 (Signed/Unsigned) and design a circuit which can calculate this.

3. How does your computer do subtract in program statement,

$$Z = X - Y \text{ or } Z = 1010 - 0100 \text{ (both are Unsigned).}$$

   Design a circuit and show how it calculates the result in each component.

4. Design a 2/3/4 bit unsigned/signed adder (normal/carry look ahead) and show output of each circuit in when X = 10 or 111 or 1001 and Y=11 or 100 or 1111.

Thank You ☺