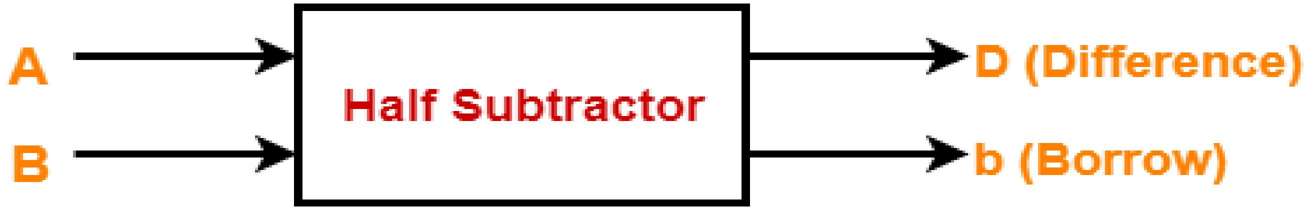
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with 'A' Grade by NAAC

Lecture session 2_ UNIT-3
Unit-3-COMBINATIONAL LOGIC SUBTRACTORS and PARALLEL ADDER

By
V.GEETHA
ASSISTANT PROFESSOR/EEE
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
CHENNAI-119

Half Subtractor-

- Half Subtractor is a combinational logic circuit.
- It is used for the purpose of subtracting two single bit numbers.
- It contains 2 inputs and 2 outputs (difference and borrow).



Step-01:

Identify the input and output variables-

- Input variables = A, B (either 0 or 1)
- Output variables = D, b where D = Difference and b = borrow

Step-02:

Draw the truth table-

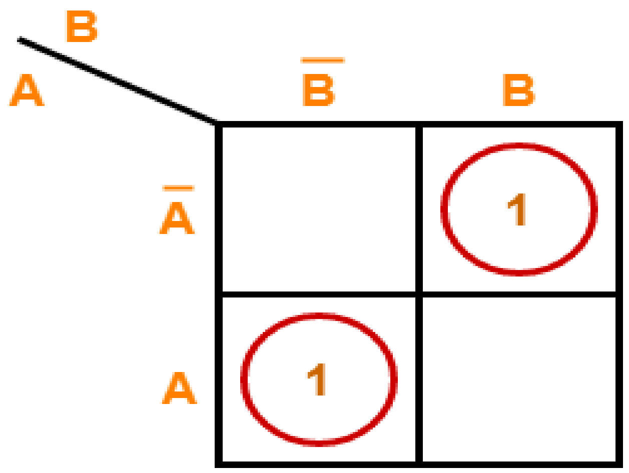
Inputs		Outputs	
A	B	D (Difference)	b (Borrow)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table

Step-03:

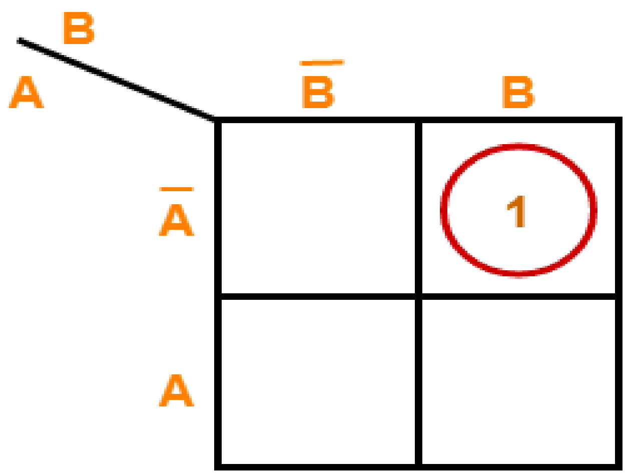
Draw K-maps using the above truth table and determine the simplified Boolean expressions-

For D:



$$\boxed{D = A \oplus B}$$

For b:



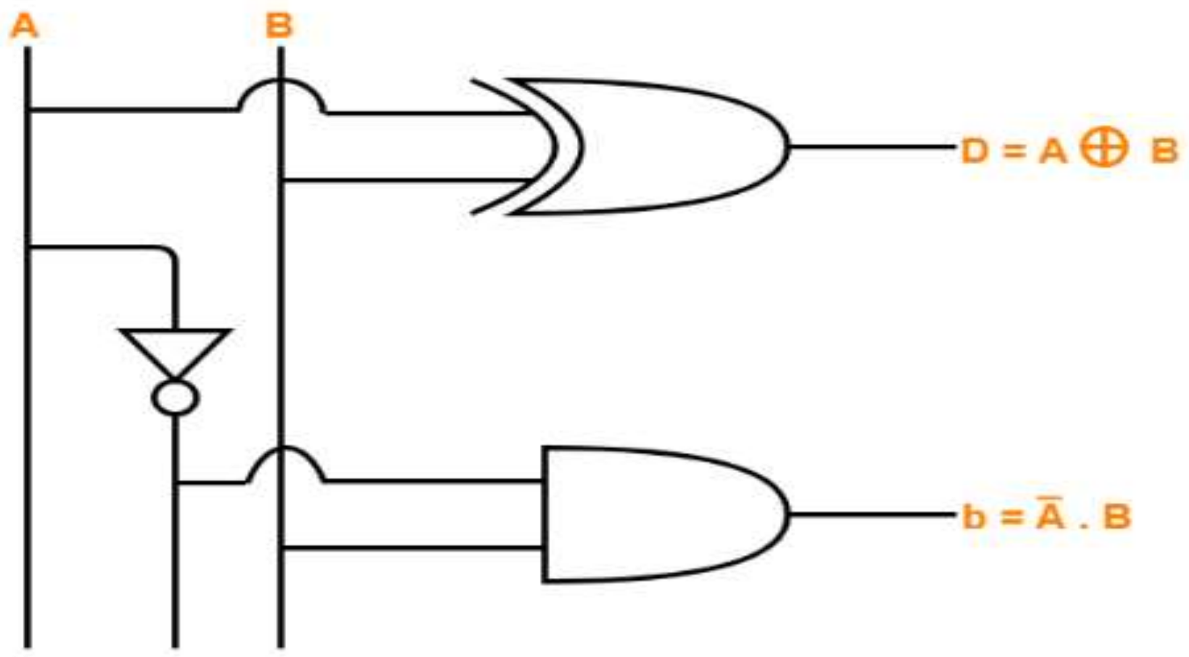
$$\boxed{b = \bar{A}.B}$$

K Maps

Step-04:

Draw the logic diagram.

The implementation of half subtractor using 1 XOR gate, 1 NOT gate and 1 AND gate is as shown below-



Half Subtractor Logic Diagram

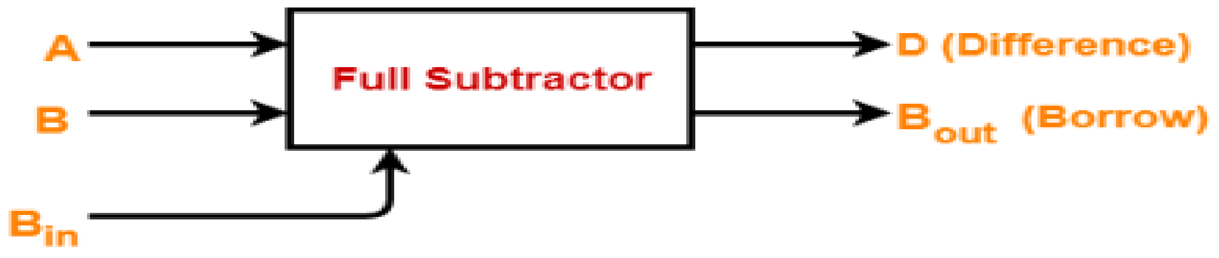
Limitation of Half Subtractor-

- Half subtractors do not take into account "Borrow-in" from the previous circuit.
- This is a major drawback of half subtractors.
- This is because real time scenarios involve subtracting the multiple number of bits which can not be accomplished using half subtractors.

To overcome this drawback, Full Subtractor comes into play.

Full Subtractor-

- Full Subtractor is a combinational logic circuit.
- It is used for the purpose of subtracting two single bit numbers.
- It also takes into consideration borrow of the lower significant stage.
- Thus, full subtractor has the ability to perform the subtraction of three bits.
- Full subtractor contains 3 inputs and 2 outputs (Difference and Borrow) as shown-



Step-01:

Identify the input and output variables-

- Input variables = A, B, B_{in} (either 0 or 1)
- Output variables = D, B_{out} where D = Difference and B_{out} = Borrow

Step-02:

Draw the truth table-

Inputs			Outputs	
A	**B**	**B_{in}**	**B_{out} (Borrow)**	**D (Difference)**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Truth Table

Step-03:

Draw K-maps using the above truth table and determine the simplified Boolean expressions-

For D:

$A \backslash BB_{in}$	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	BB_{in}	$B\bar{B}_{in}$
$\bar{A}$		1		1
A	1		1	

$$\boxed{D = A \oplus B \oplus B_{in}}$$

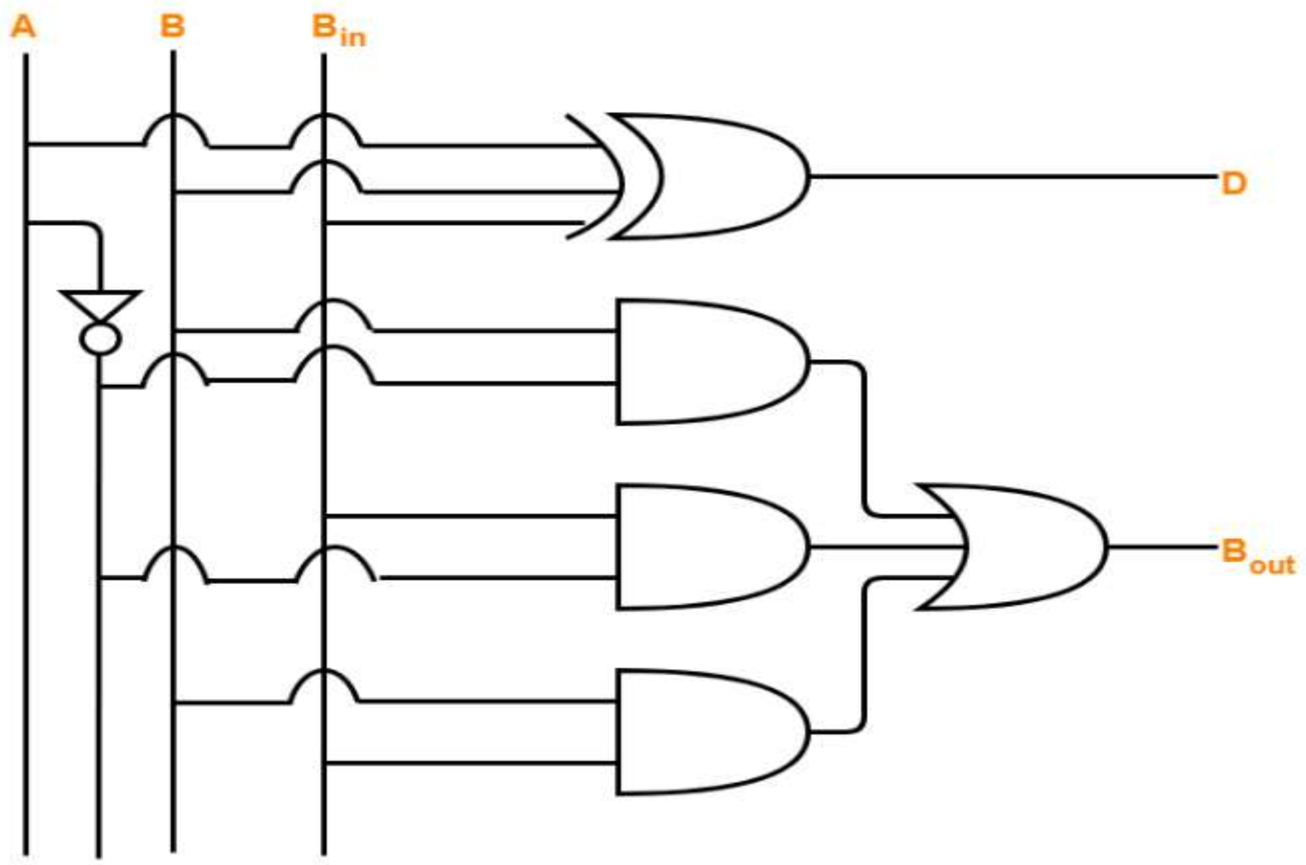
For B_{in}:

$A \backslash BB_{in}$	$\bar{B}\bar{B}_{in}$	$\bar{B}B_{in}$	BB_{in}	$B\bar{B}_{in}$
$\bar{A}$		1	1	1
A			1	

$$\boxed{B_{out} = \bar{A}B + (\bar{A} + B)B_{in}}$$

Step-04:

Draw the logic diagram.
The implementation of full adder using 1 XOR gate, 3 AND gates, 1 NOT gate and 1 OR gate is as shown below-



Full Subtractor Logic Diagram

Simplification of D:

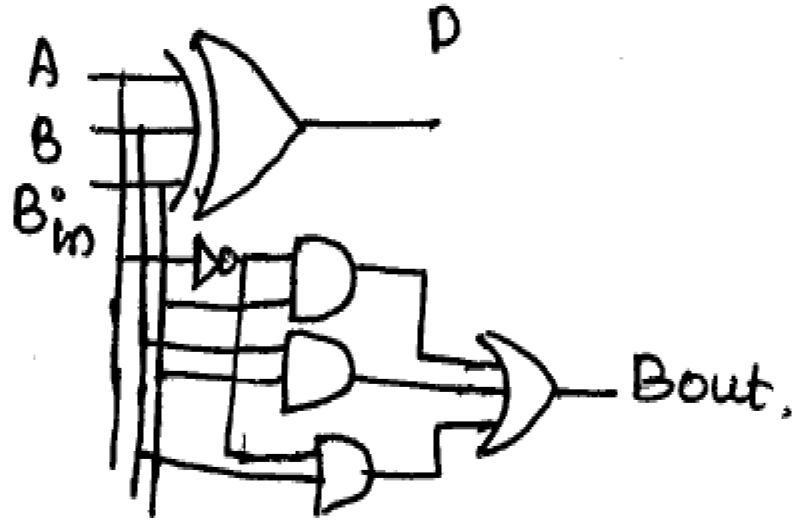
$= A\bar{B}\bar{B}_{in} + \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}B\bar{B}_{in}$

$= B_{in}(AB + \bar{A}\bar{B}) + \bar{B}_{in}(A\bar{B} + \bar{A}B)$

$= B_{in}(A \odot B) + \bar{B}_{in}(A \oplus B)$

$= B_{in}(\overline{A \oplus B}) + \bar{B}_{in}(A \oplus B)$

$= B_{in} \oplus A \oplus B.$



Full subtractor with two half subtractor,

$$B_{out} = \bar{A}B_{in} + \bar{A}B + BB_{in}$$

$$= \bar{A}B + \bar{A}B_{in}(B+\bar{B}) + BB_{in}(A+\bar{A})$$

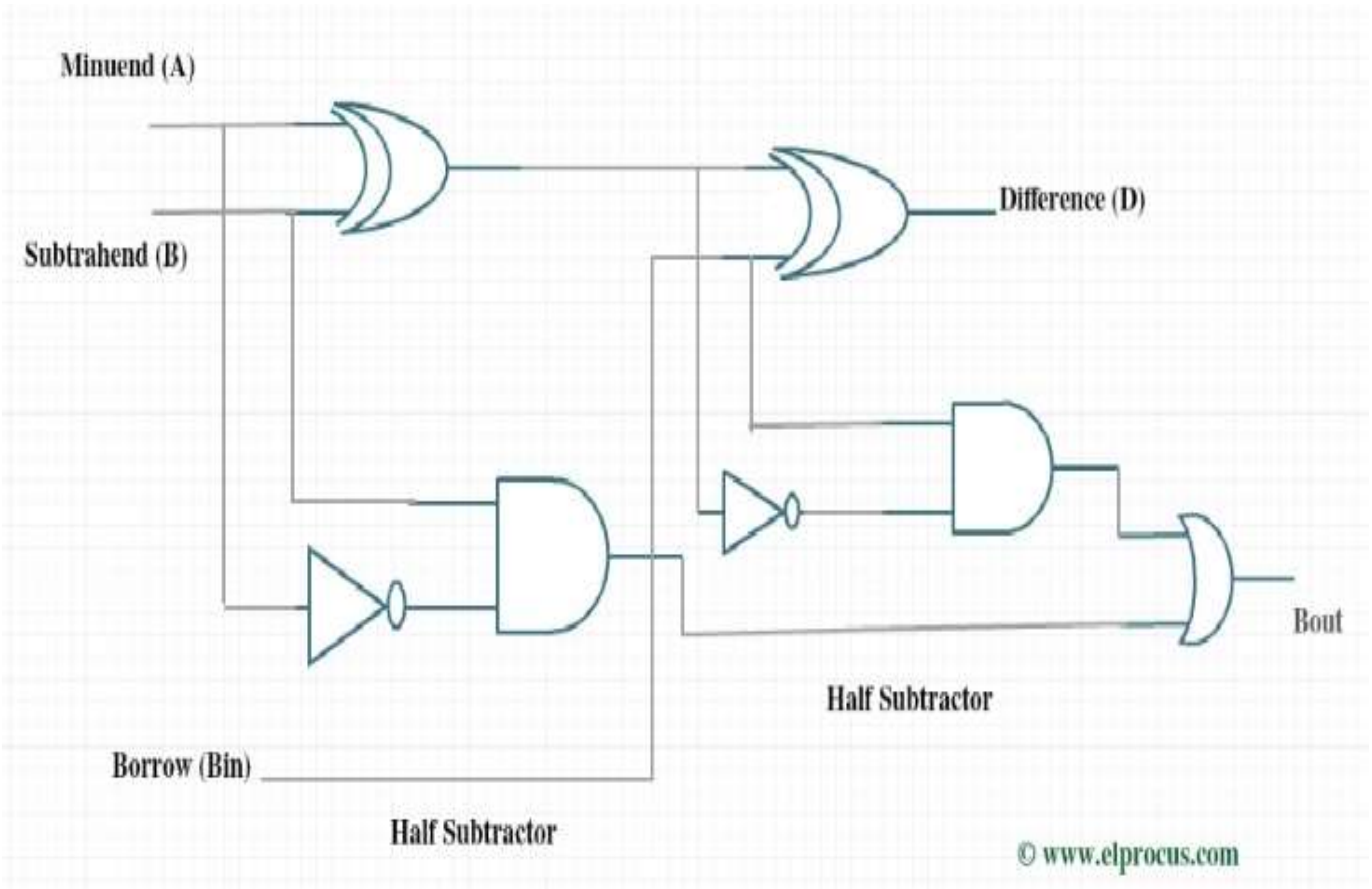
$$= \bar{A}B + \bar{A}B_{in}B + \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}BB_{in}$$

$$= \bar{A}B + \bar{A}B_{in}B + \bar{A}\bar{B}B_{in} + ABB_{in}$$

$$= \bar{A}B(1+B_{in}) + (\bar{A}\bar{B} + AB)B_{in}$$

$$= \bar{A}B + (A \odot B)B_{in}$$

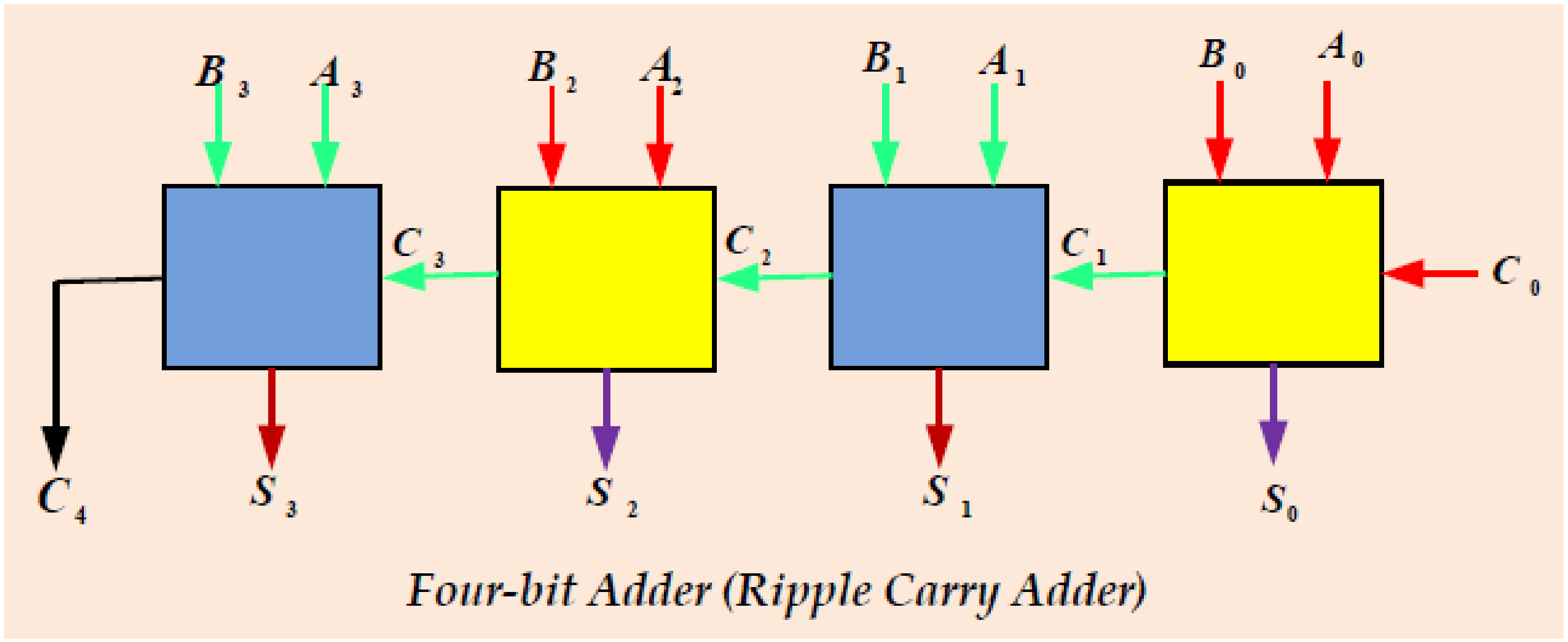
$$= \bar{A}B + (\overline{A \oplus B})B_{in}$$



Full Subtractor

3. Binary Adder (Asynchronous Ripple-Carry Adder)

- A binary adder is a digital circuit that produces the ***arithmetic sum of two binary numbers***.
- A binary adder can be constructed with ***full adders connected in cascade*** with the output carry form each full adder connected to the input carry of the next full adder in the chain.
- The ***four-bit adder*** is a typical example of a ***standard component*** .It can be used in many application involving arithmetic operations.

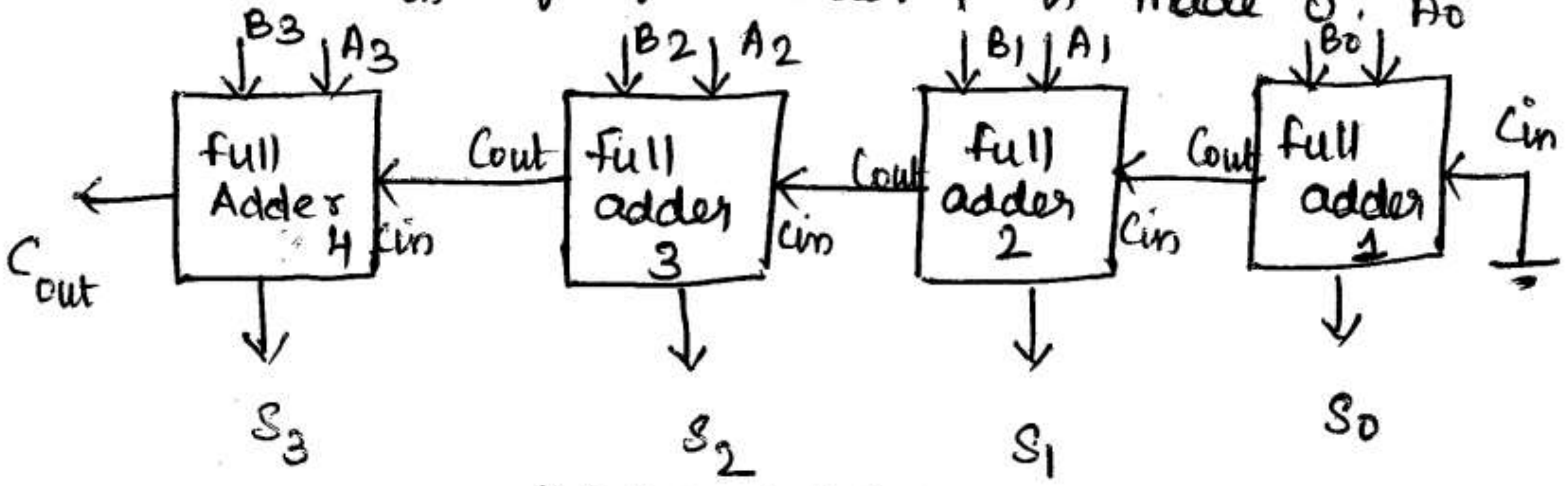


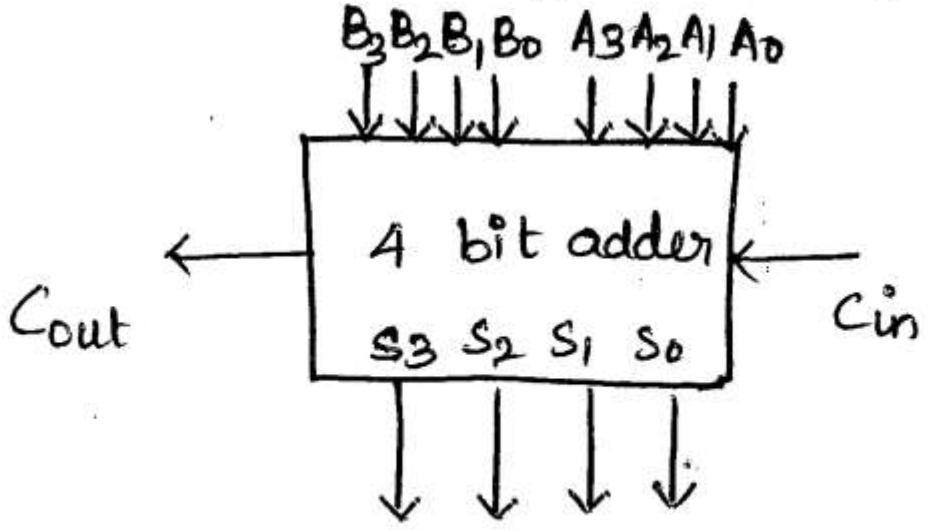
Four-bit Adder (Ripple Carry Adder)

1. Design a 4 bit parallel adder using full adders.

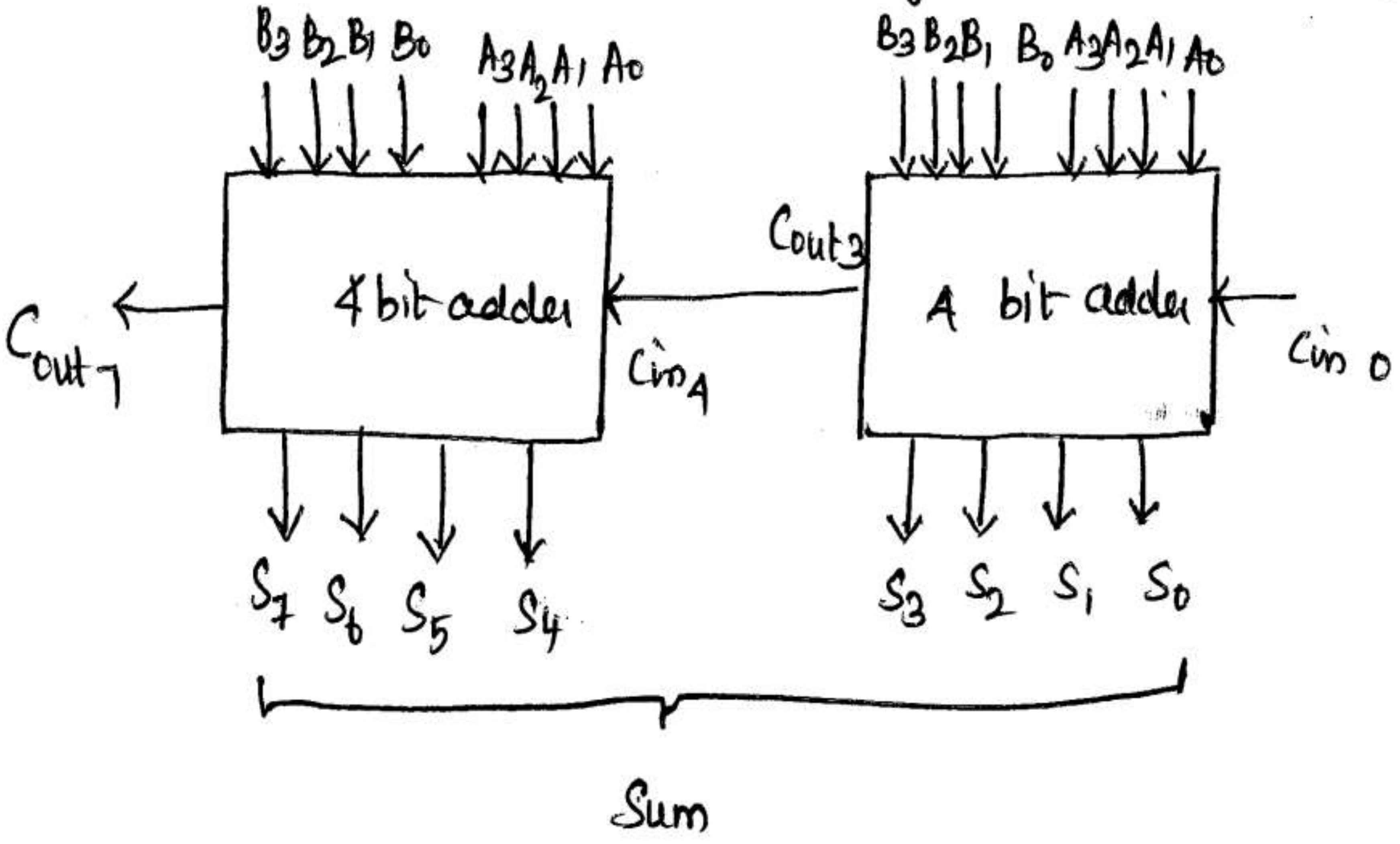
* 4 Full adders.

* C_{in} of full adder 1 is made 0.





2. Design a 8 bit adder using two four bit adder.



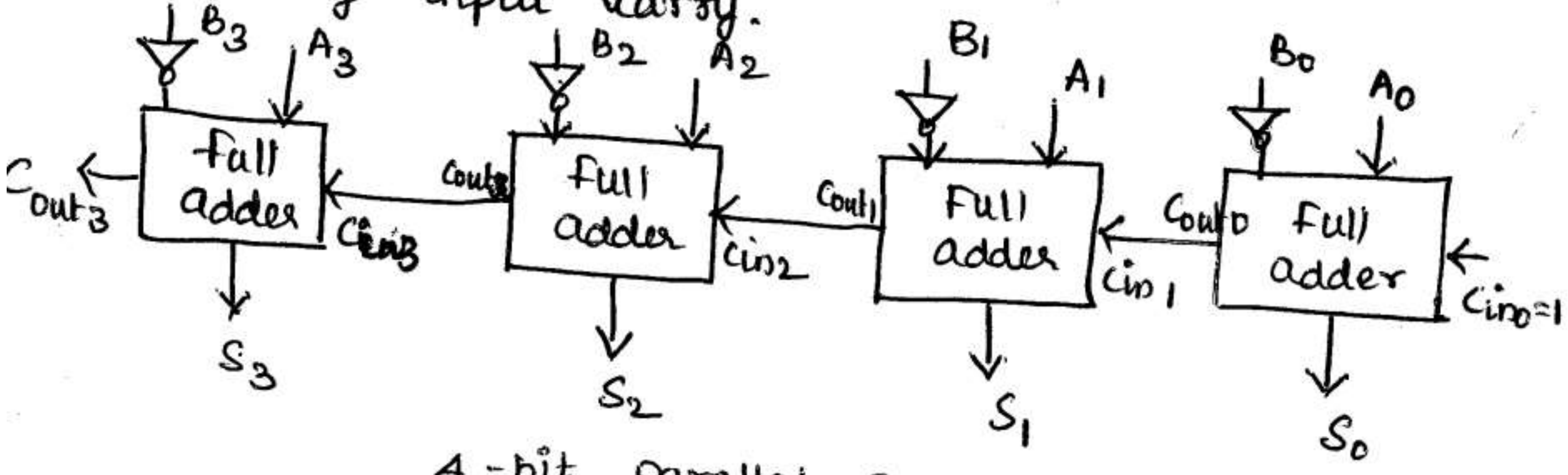
Parallel subtractor

Subtraction of binary numbers can be done most Conveniently by means of Complements.

Subtraction of A and B can be done by taking 2's complement of B and adding it with A.

2's Complement is done by taking 1's Complement and adding 1 to 1's Complemented number

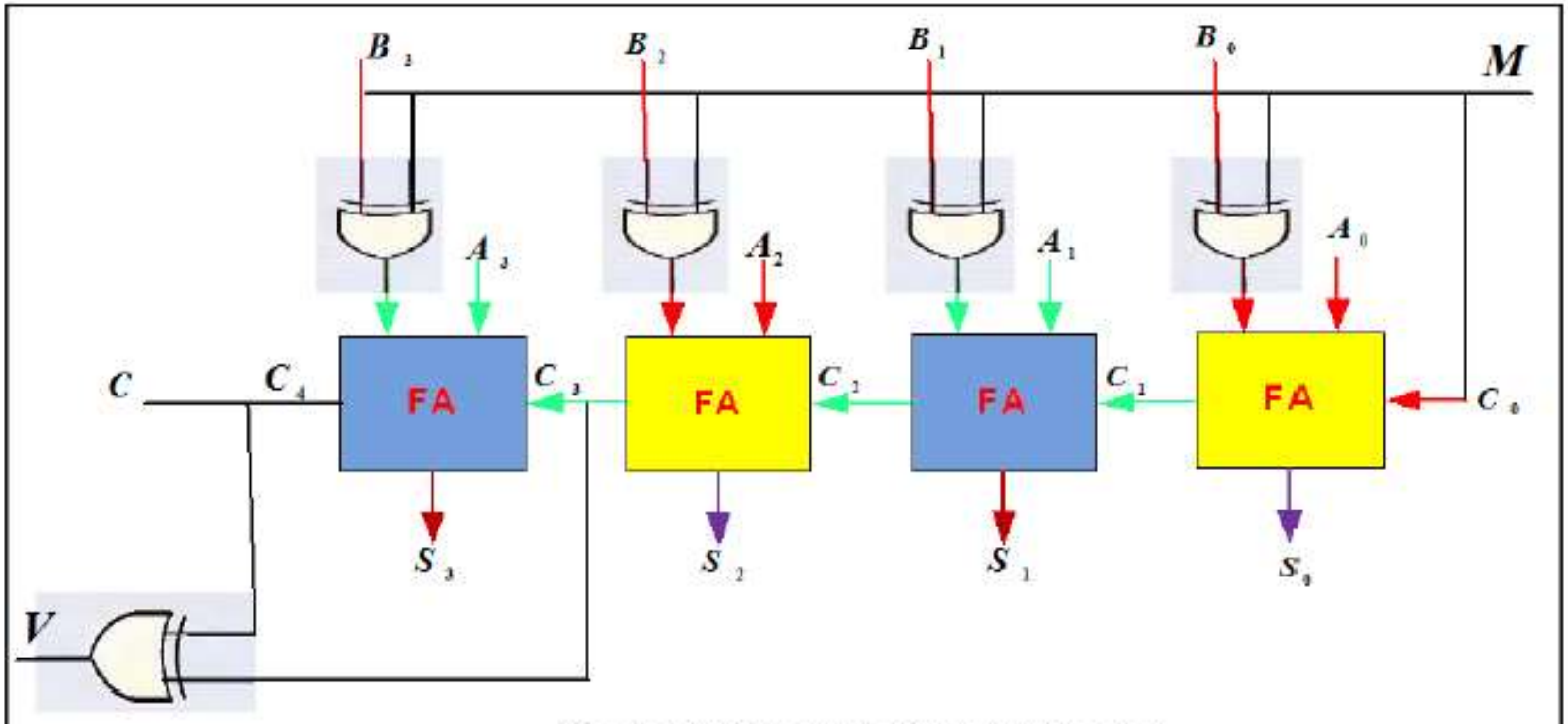
1's complement is done by taking inversion of the number. Here 1 is added to the complemented number by input carry.



4-bit parallel subtractor.

5. Binary Adder–Subtractor

- The addition and subtraction operations can be combined into one circuit with one common binary adder by including an *exclusive-OR* gate with each full-adder.



Four-bit Binary Adder–Subtractor

The mode input ***M*** controls the operation as the following:

- ($M = 0 \rightarrow$adder.
- $M = 1 \rightarrow$ subtractor.

➢ Each *XOR* gate receives M signal and B
  - When $M = 0$ then $B \oplus 0 = B$ and the carry $= 0$, then the circuit performs the operation $A + B$.
  - When $M = 1$ then $B \oplus 1 = \overline{B}$ and the carry $= 1$, then the circuit performs the operation $A - B$.

➢ The *exclusive-OR* with output V is for detecting an overflow.

CARRY PROPAGATION DELAY

$A + B$ $(A = 1011)$ and $(B = 0011)$

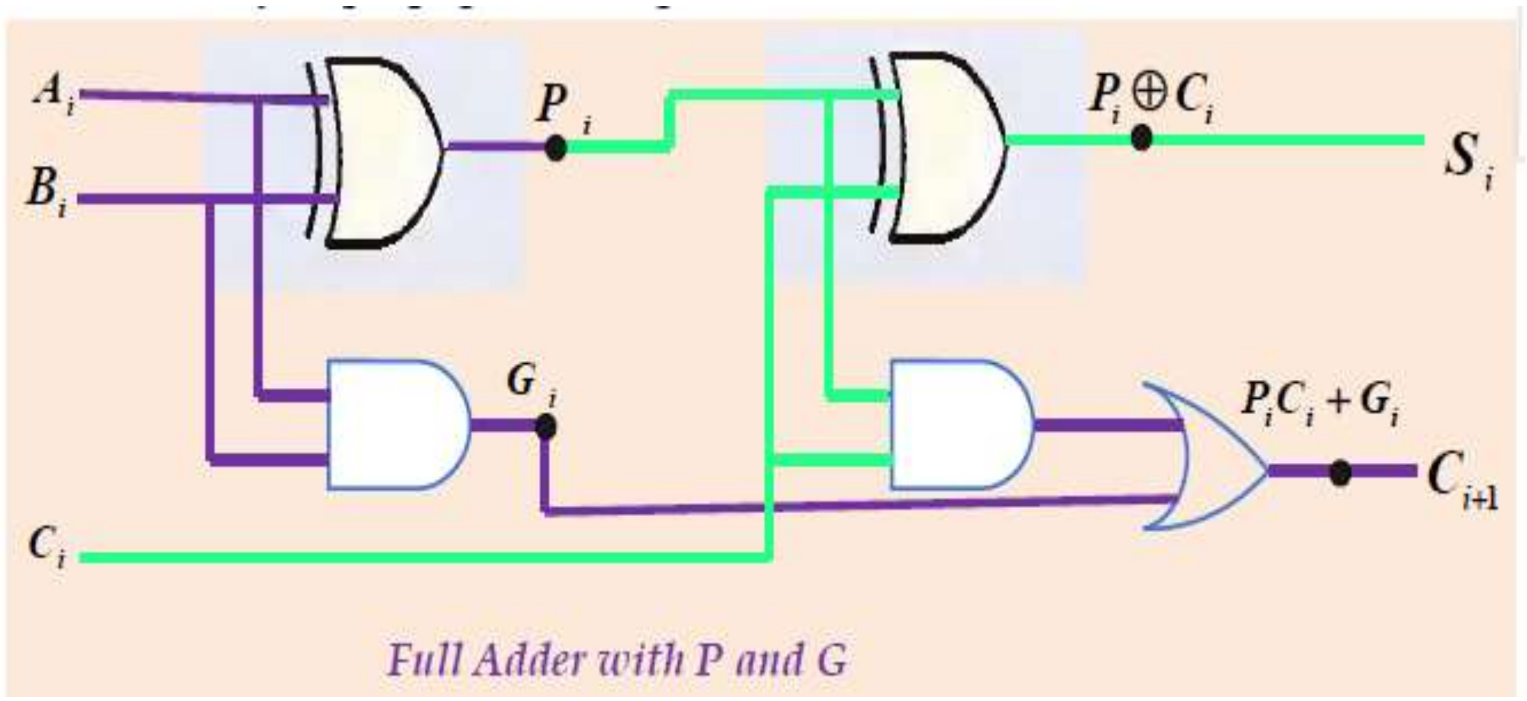
Subscript i	3	2	1	0		
Input Carry	0	1	1	0	C_i	
A	1	0	1	1	A_i	$C_0 = 0$
+						
B	0	0	1	1	B_i	
Sum	1	1	1	0	S_i	
Output Carry	0	0	1	1	C_{i+1}	

The carry propagation time is an important attribute of the adder because it limits the speed with which two numbers are added.

To reduce the carry propagation delay time:

1) Employ faster gates with reduced delays.

2) Employ the principle of Carry Look a head Logic.



Full Adder with P and G

***Proof**: (using carry lookahead logic)*

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

The output sum and carry are:

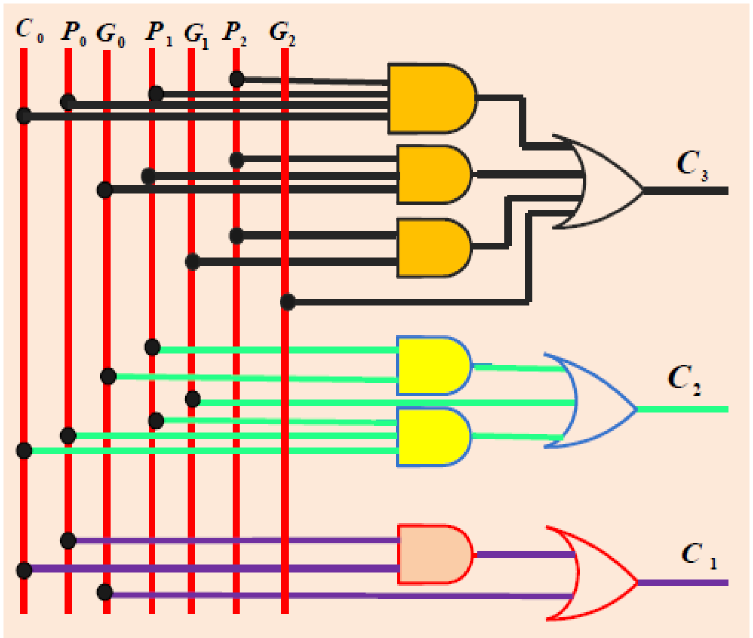
$$S_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + P_i C_i$$

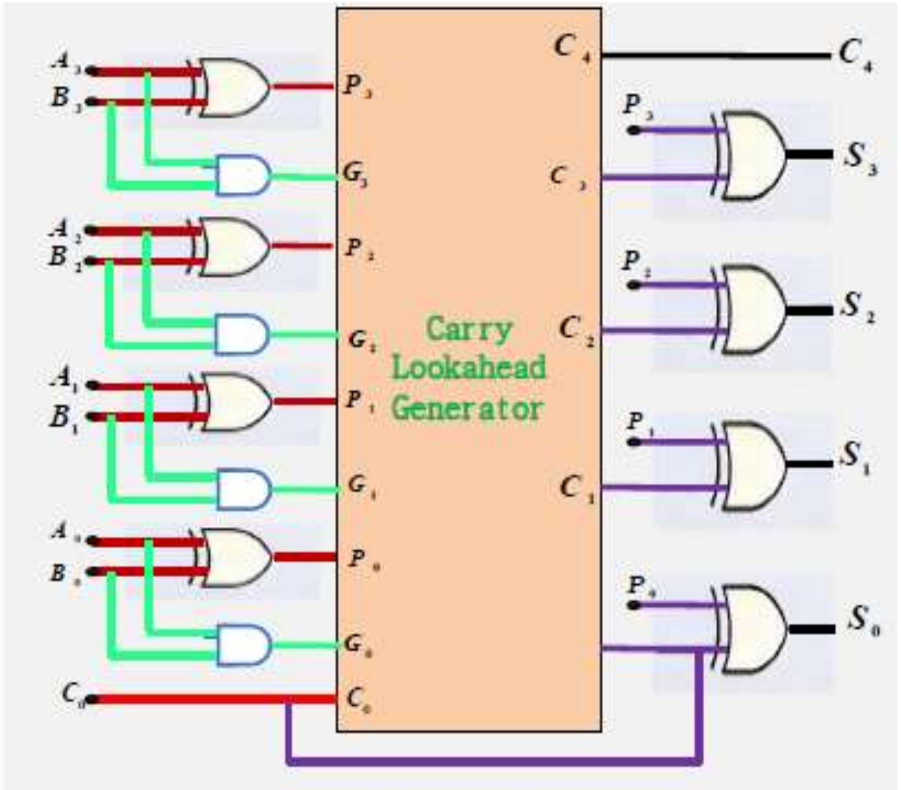
- ✓ G_i-called a ***carry generate***, and it produces a carry of ***1*** when both A_i and B_i are ***1***.
- ✓ P_i-called a ***carry propagate***, it determines whether a carry into stage i will propagate into stage $i+1$.
- ✓ The ***Boolean function*** for the carry outputs of each stage and substitute the value of each C_i form the previous equations:

$$\left\{\begin{aligned} C_0 &= \textit{input carry} \\ C_1 &= G_0 + P_0 C_0 \\ C_2 &= G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) \\ &= G_1 + P_1 G_0 + P_1 P_0 C_0 \\ C_3 &= G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 C_0) \\ &= G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0 \end{aligned}\right\}$$

- ➢ The three Boolean functions C_1, C_2 and C_3 are implemented in the ***carry lookahead generator.***



Logic Diagram for Carry Lookahead Generator



four-bit adder with a carry lookahead scheme

The two level-circuit for the output carry C_4 is not shown, it can be easily derived by the equation.

- C_3 does not have to wait for C_2 and C_1 to propagate, in fact C_3 is propagated at the same time as C_1 and C_2.

Thank you