UNIT-V: LOGIC GATES AND ITS APPLICATIONS

Logic Gates: Basic gates AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR - Building of AND, OR and NOT Gate with diodes.

Applications: Half adder, Full adder, Half Subtractor, Full Subtractor and Binary parallel adder.

Logic Gates

Seven types of gates

- NOT
- AND
- OR
- XOR
- XNOR
- NAND
- NOR

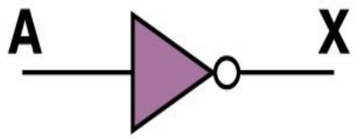
NOT Gate

- A NOT gate accepts one input signal (0 or 1) and returns the opposite signal as output

Boolean Expression

$$X = A'$$

Logic Diagram Symbol



Truth Table

A	X
0	1
1	0

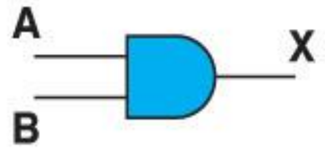
AND Gate

- An AND gate accepts two input signals
- If both are 1, the output is 1; otherwise, the output is 0

Boolean Expression

$X = A \cdot B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

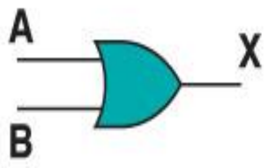
OR Gate

- An OR gate accepts two input signals
- If both are 0, the output is 0; otherwise, the output is 1

Boolean Expression

$X = A + B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

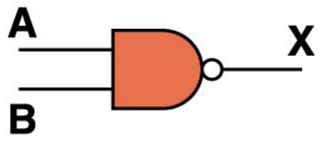
NAND Gate

- The NAND gate accepts two input signals
- If both are 1, the output is 0; otherwise, the output is 1

Boolean Expression

$X = (A \cdot B)'$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

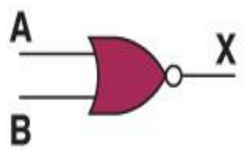
NOR Gate

- The NOR gate accepts two input signals
- If both are 0, the output is 1; otherwise, the output is 0

Boolean Expression

$X = (A + B)'$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

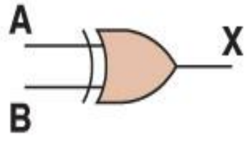
XOR Gate

- An XOR gate accepts two input signals
- If both are the same, the output is 0; otherwise,the output is 1

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol

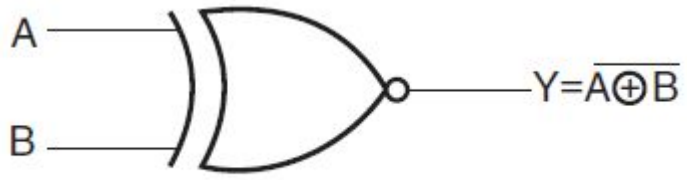


Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

XNOR Gate

- An XNOR gate accepts two input signals
- If both are the same, the output is 1; otherwise, the output is 0.

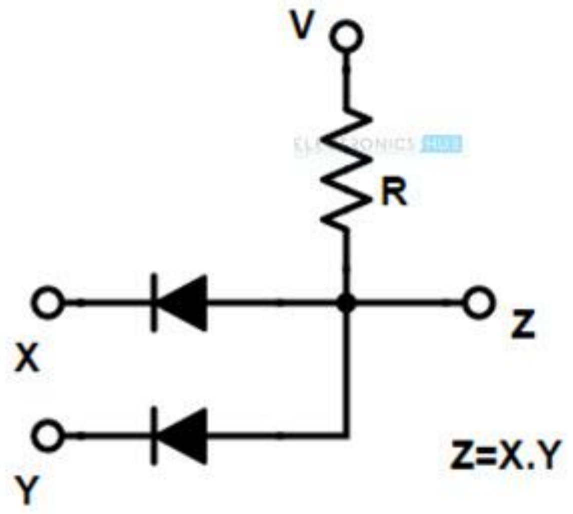


$$Y = (\overline{A \oplus B}) = (A.B + \overline{A}.\overline{B})$$

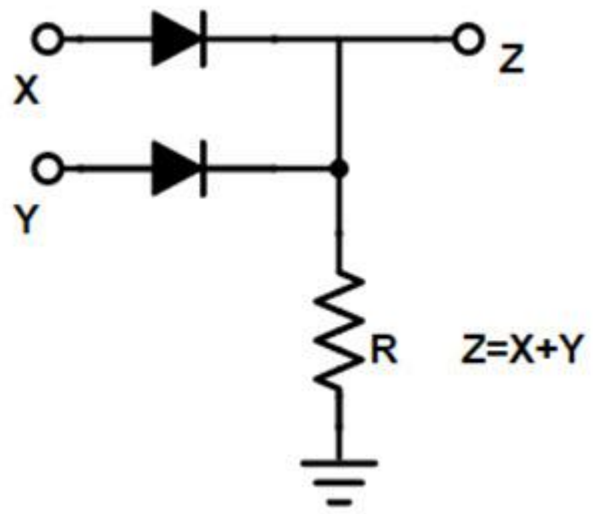
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Implementation of AND,OR and NOT logic using Diodes

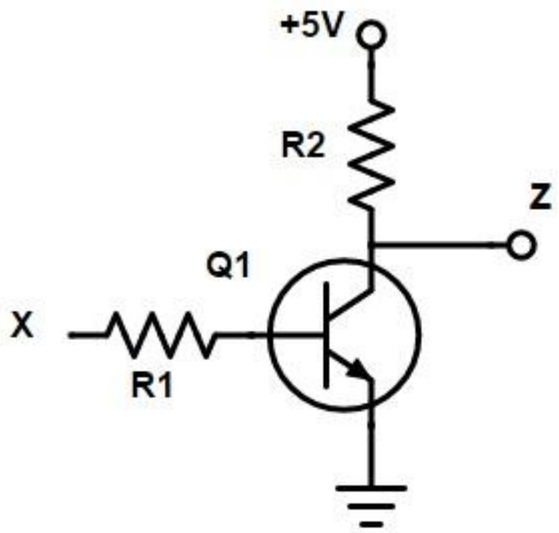
- **AND gate**:



- **OR gate**:

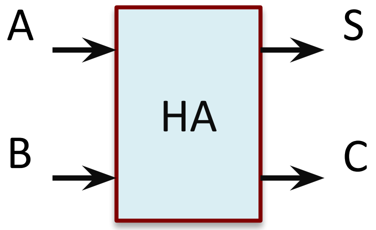


- **NOT Gate**:

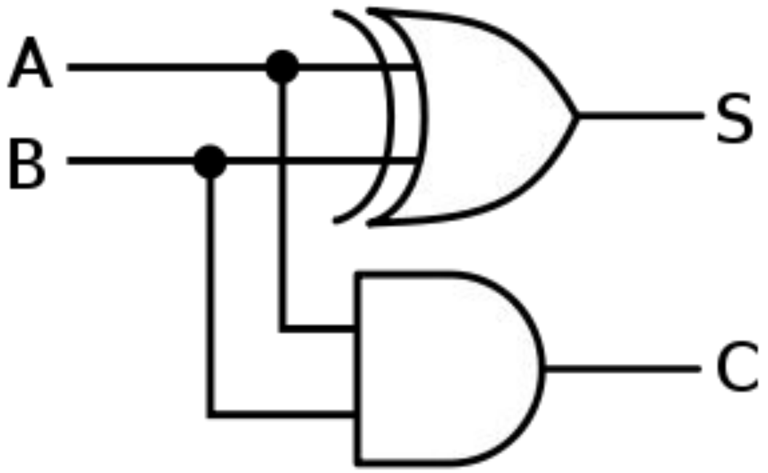


Applications

- **Half Adder**:



Logic Symbol



Circuit Diagram

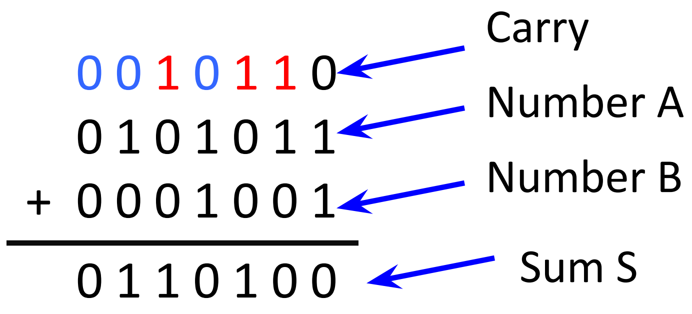
Truth Table

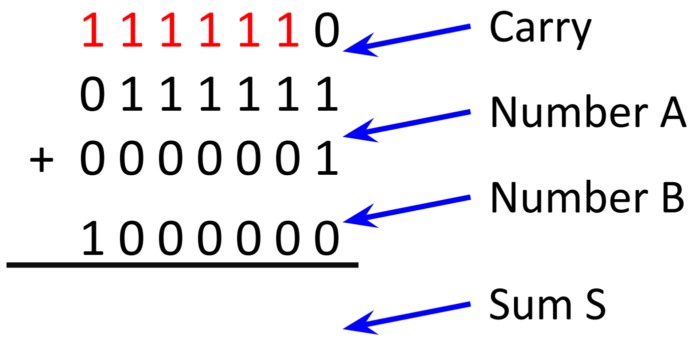
Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = A'.B + A.B' = A \oplus B$

$C = A.B$

Addition of Multi-bit Binary Numbers



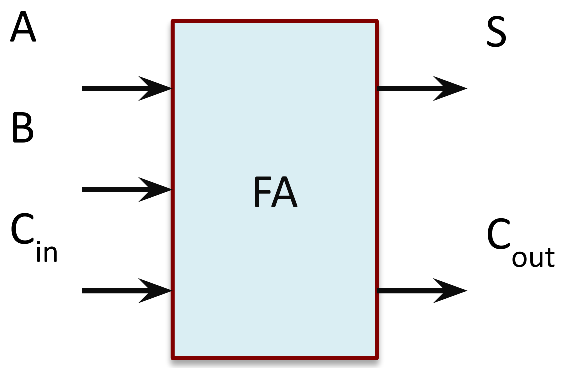


- At every bit position (stage), we require to add 3 bits:
  - 1 bit for number A
  - 1 bit for number B
  - 1 carry bit coming from the previous stage

WE NEED A FULL ADDER

Full Adder?

- A full adder has three inputs and two outputs:
  - Inputs: two input bits A and B, the the carry input C_{in}.
  - Outputs: the sum S, and the carry output C_{out}.



Full Adder

Inputs			Outputs	
A	B	C_{in}	S	C_{out}
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**

$$S = A'.B'.C_{in} + A'.B.C_{in}' + A.B'.C_{in}' + A.B.C_{in}$$
$$= A \oplus B \oplus C_{in}$$

$$C_{out} = A'.B.C_{in} + A.B'.C_{in} + A.B.C_{in}' + A.B.C_{in}$$
$$= A.B + B.C_{in} + A.C_{in}$$

- **SUM**:

$A'B'C_{in} + A'BC_{in}' + AB'C_{in}' + ABC_{in}$

$A'(B'C_{in} + BC_{in}') + A(B'C_{in}' + BC_{in})$

$A' (B \oplus C_{in}) + A (B \oplus C_{in})'$

$A \oplus B \oplus C_{in}$

- **C_{out}**:

$A'.B.\ C_{in} + A.B'.\ C_{in} + A.B.\ C_{in}' + A.B.\ C_{in}$

$A'.B.\ C_{in} + A.B'.\ C_{in} + A.B.(C_{in}' + C_{in}) \qquad (C_{in}' + C_{in} = 1)$

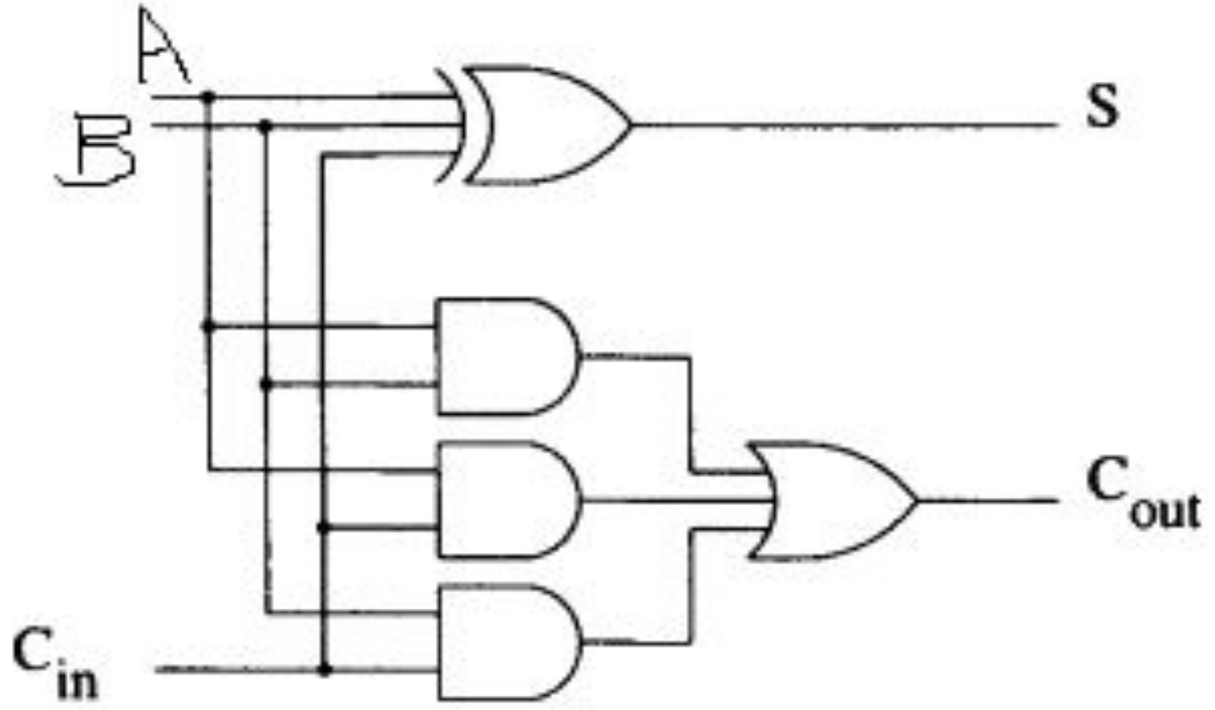
$A'.B.\ C_{in} + A(B'\ C_{in} + B) \quad (X+YZ=(X+Y)(X+Z))$

$A'.B.\ C_{in} + A((B'+B)(B'+C_{in})) = A'.B.\ C_{in} + A(B'+C_{in})$

$A'.B.\ C_{in} + AB' + AC_{in} = (A'.B+A)\ C_{in} + AB'$

$(A'+A)(B+A)\ C_{in} + AB' = (B+A)\ C_{in} + AB' = B.\ C_{in} + A.C_{in} + A.B'$

Full Adder



Half Subtractor

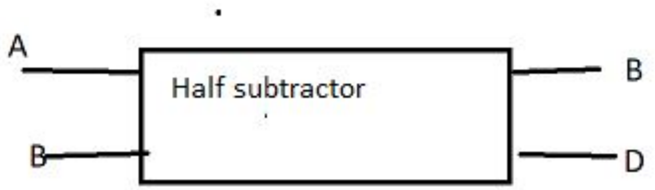
- It produces the difference between the two binary bits
- output (Borrow) to indicate if a 1 has been borrowed.
- In the subtraction (A-B), A is called as Minuend bit and B is called as Subtrahend bit.

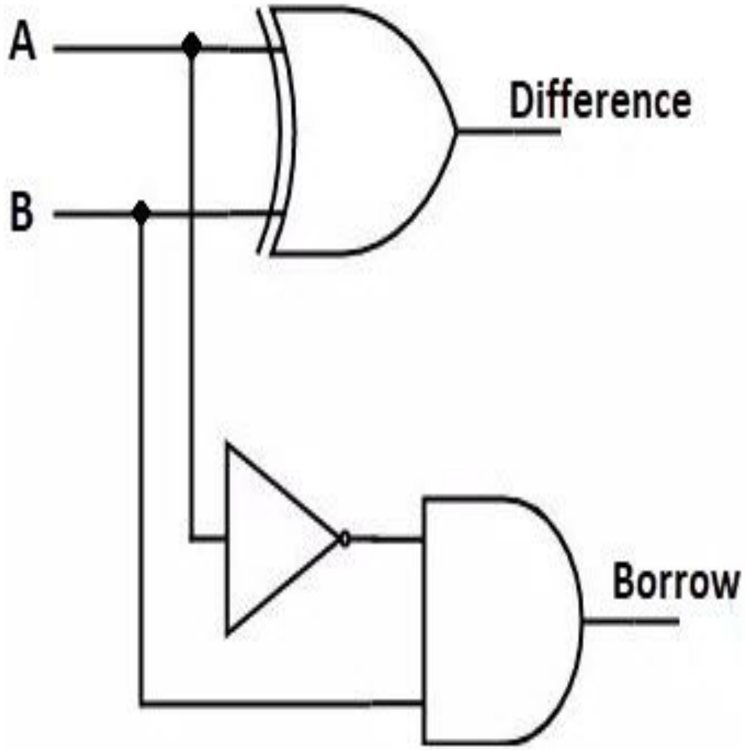
Half Subtractor

Inputs		Outputs	
A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

D=A’B+AB’

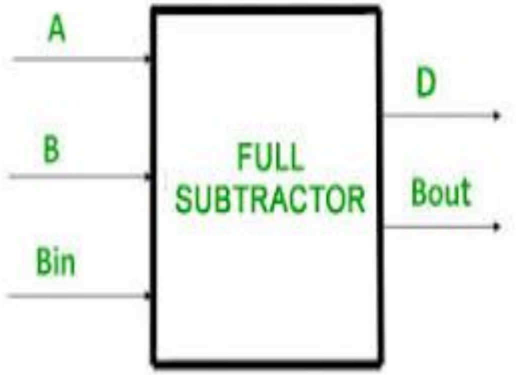
B=A’B





Full Subtractor

Inputs			Outputs	
A	B	B_{in}	D	B_{out}
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



- **Difference**:

 $A'B'C_{in} + A'BC_{in}' + AB'C_{in}' + ABC_{in}$

 $A'(B'C_{in} + BC_{in}') + A(B'C_{in}' + BC_{in})$

 $A' (B \oplus C_{in)} + A (B \oplus C_{in})'$

 $A \oplus B \oplus C_{in}$

- **B$_{out}$**:

$A'.B'.\ B_{in} + A'.B.\ B_{in}' + A'.B.\ B_{in} + A.B.\ B_{in}$

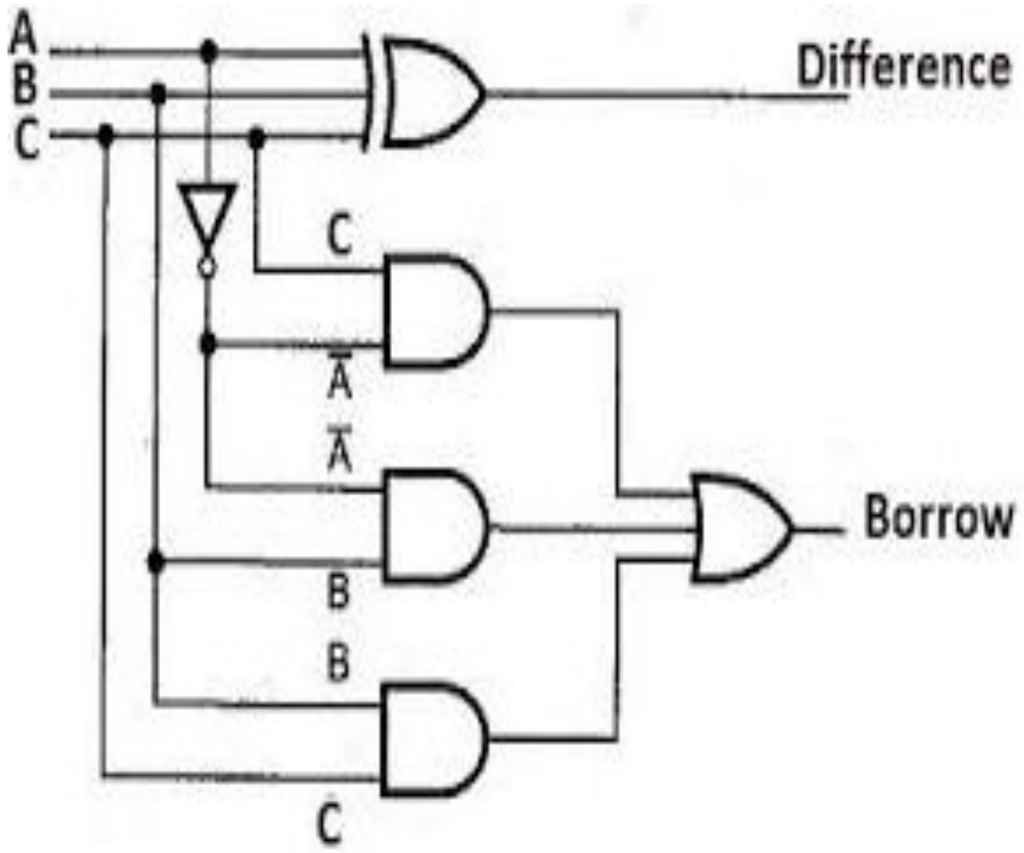
$A'.B'.\ B_{in} + A'.B.\ B_{in}' + B.\ B_{in}(A' + A)$ $(A' + A = 1)$

$A'.B'.\ B_{in} + A'.B.\ B_{in}' + B.\ B_{in}$ $X+YZ=(X+Y)(X+Z))$

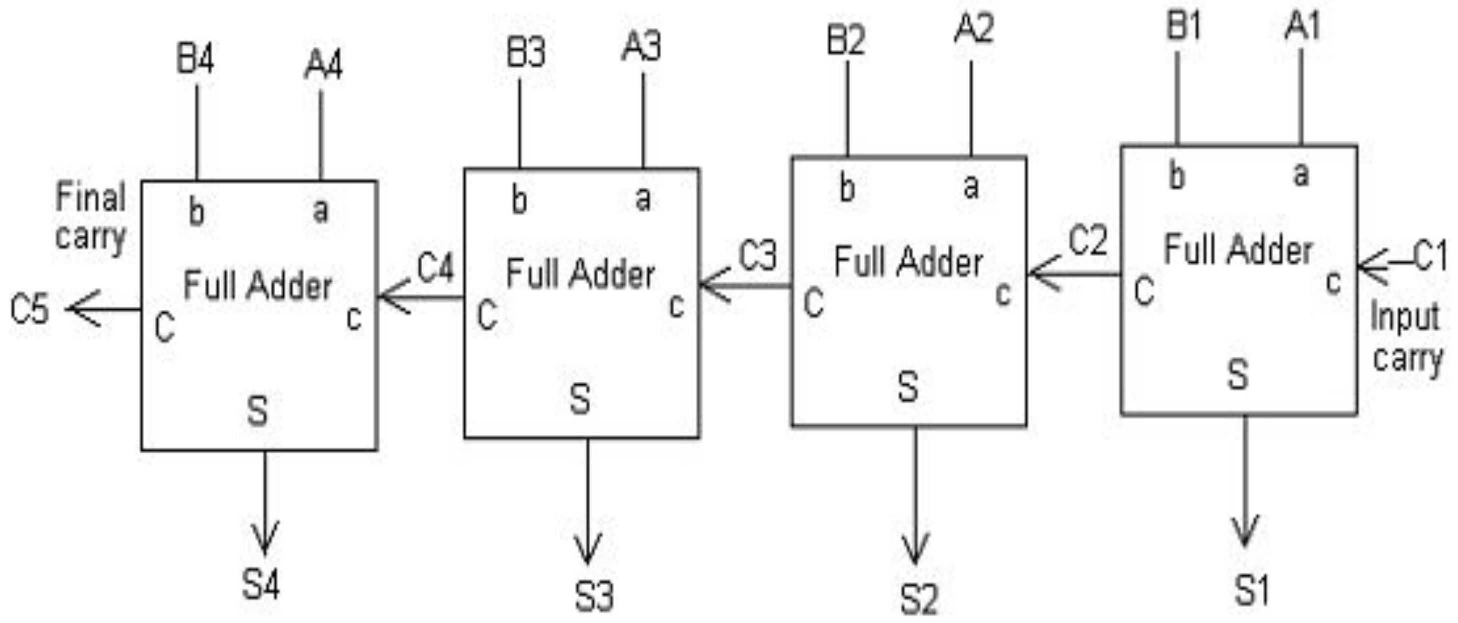
$A'.B'.\ B_{in} + B(A'.B_{in}' + B_{in}) = A'.B'.\ B_{in} + B(A' + B_{in}) =$

$A'.B'.\ B_{in} + A'.B + B.B_{in} = A'(B'.\ B_{in} + B) + B.B_{in} = A'B_{in} + A'B + B.B_{in}$

Full subtractor



- **Binary parallel Adder**:


B4
A4
B3
A3
B2
A2
B1
A1
Final carry
C5
b
a
Full Adder
C
c
S
C4
C3
C2
C1
Input carry
S4
S3
S2
S1