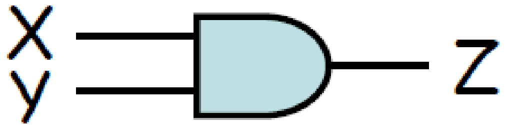
ESC201: Lecture 12

Dr. Imon Mondal

Assistant Professor,
Electrical Engineering, IIT Kanpur

Digital Processing and Other Gates

- Recall that we only have two values: 0 & 1
  - Map naturally to logic TRUE (T) and FALSE (F)
  - Can also represent numbers



AND gate

- We need to realize all functions using 0 and 1

Truth table

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Enumeration of all possible input combinations

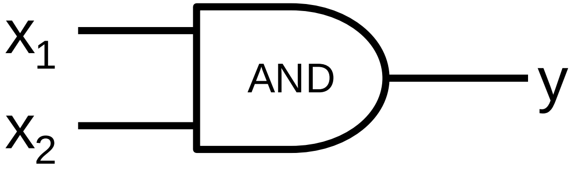
- Boolean logic (X, Y, Z are digital signals "0" and "1")
  - If X is true and Y is true then Z is true else Z is false
  - Z = X AND Y or Z = X. Y

AND & OR Gates

AND of two logic values

AND: $y = x_1 . x_2$

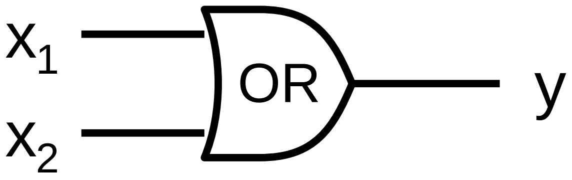
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1



OR of two logic values

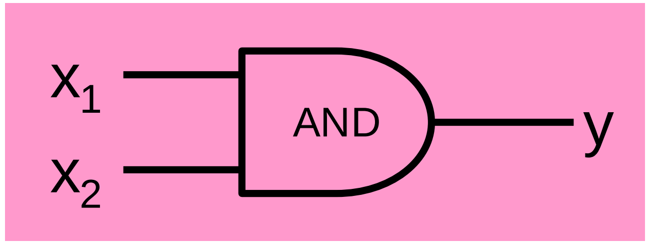
OR: $y = x_1 + x_2$

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

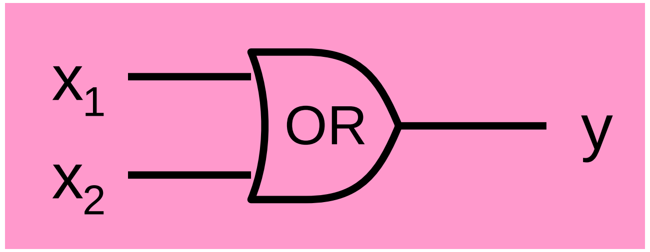


Logic Gates

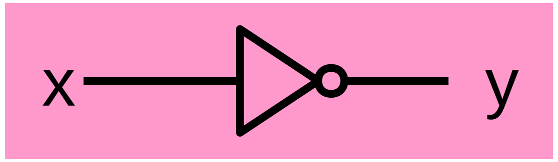
AND: $y = x_1 . x_2$



OR: $y = x_1 + x_2$



NOT: $y = \overline{x}$

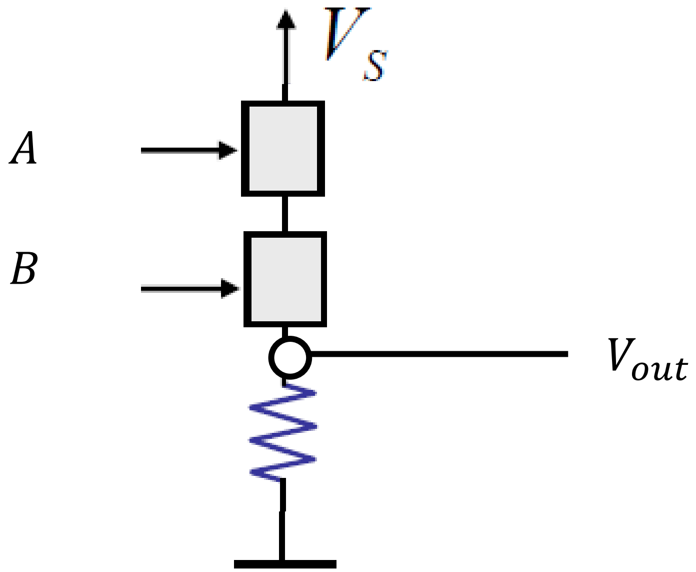


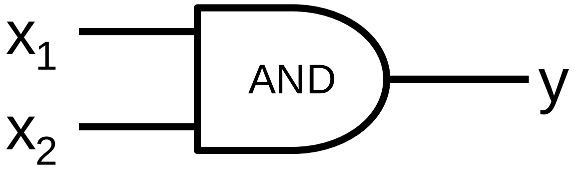
AND Gate?

- AND of two logic values

$$\text{AND:} \quad y = x_1 . x_2$$

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1





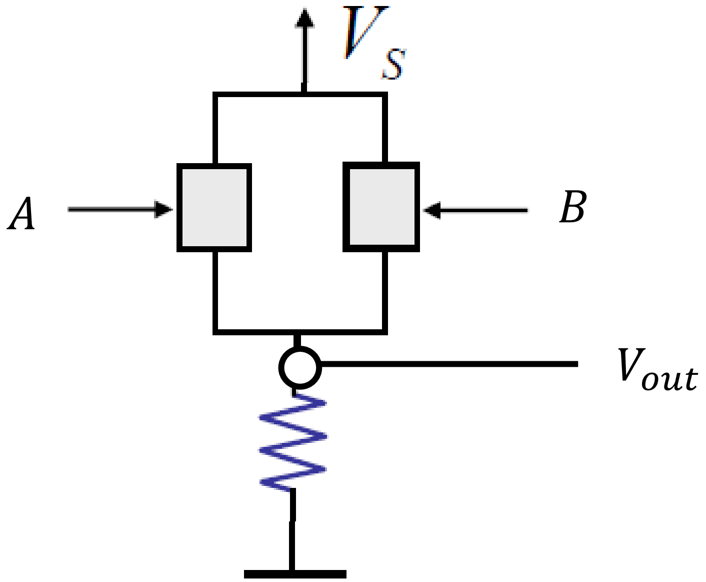
➢ No, the gain is always less than 1 => no noise immunity

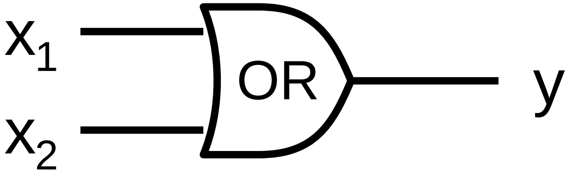
OR Gate?

- OR of two logic values

$$\text{OR:} \quad y = x_1 + x_2$$

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1



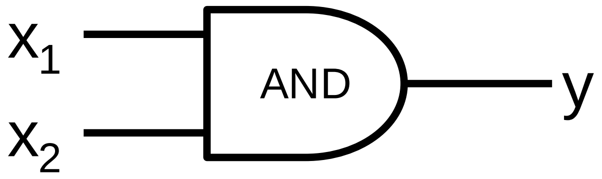


➢ No, the gain is always less than 1 => no noise immunity

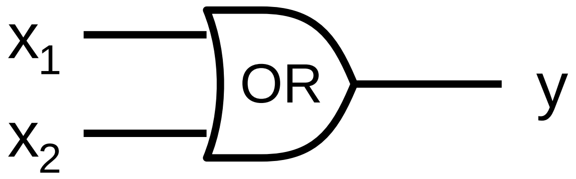
Elementary Logic Gates

Elementary Gates

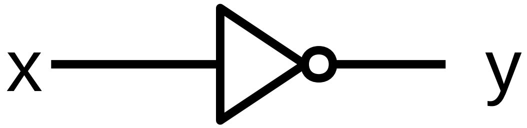
AND: $y = x_1 . x_2$



OR: $y = x_1 + x_2$

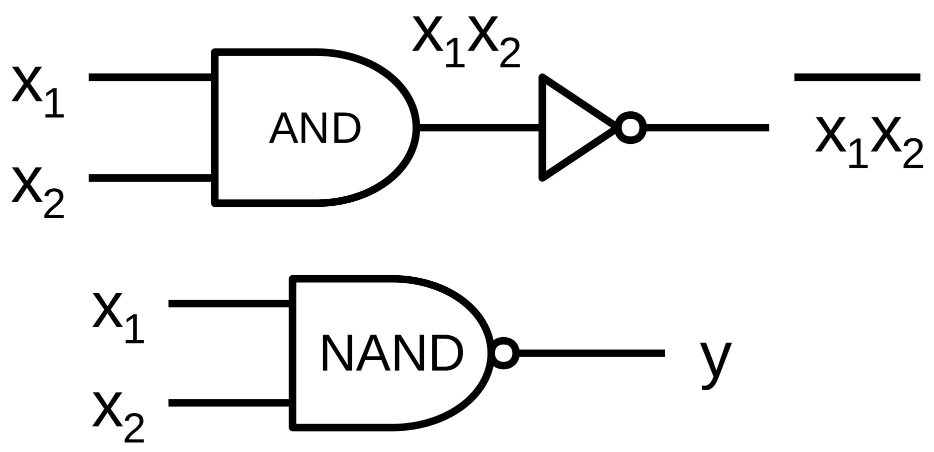


NOT: $y = \overline{x}$

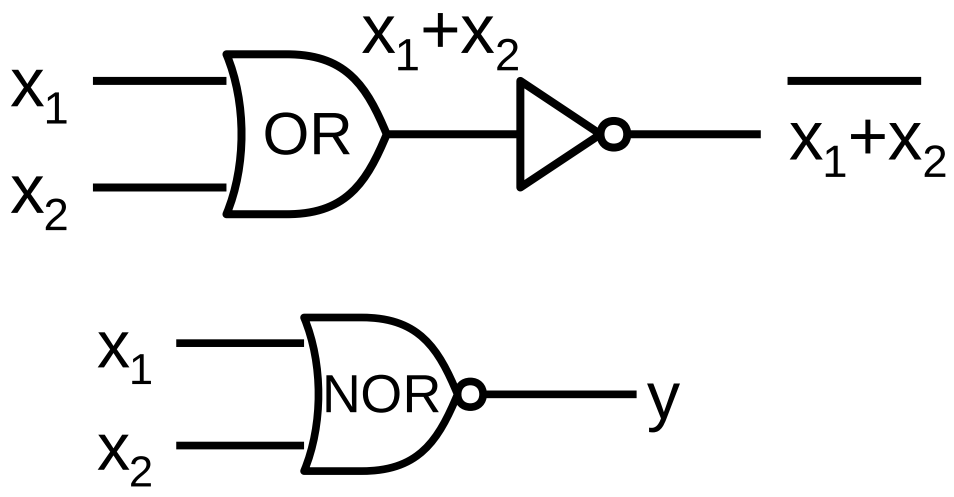


NAND & NOR Gates

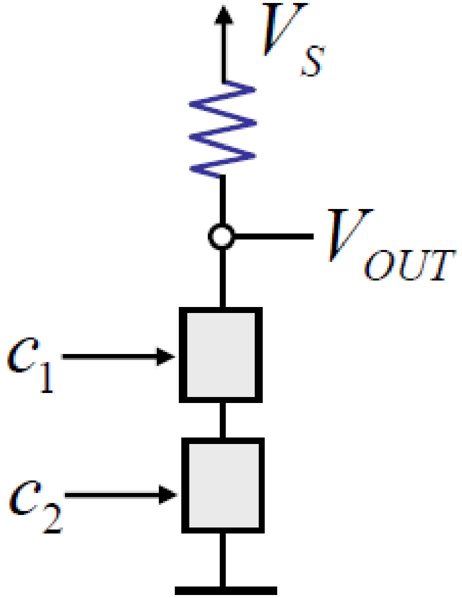
NAND: $y = \overline{x_1 . x_2}$

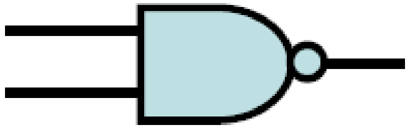


NOR: $y = \overline{x_1 + x_2}$



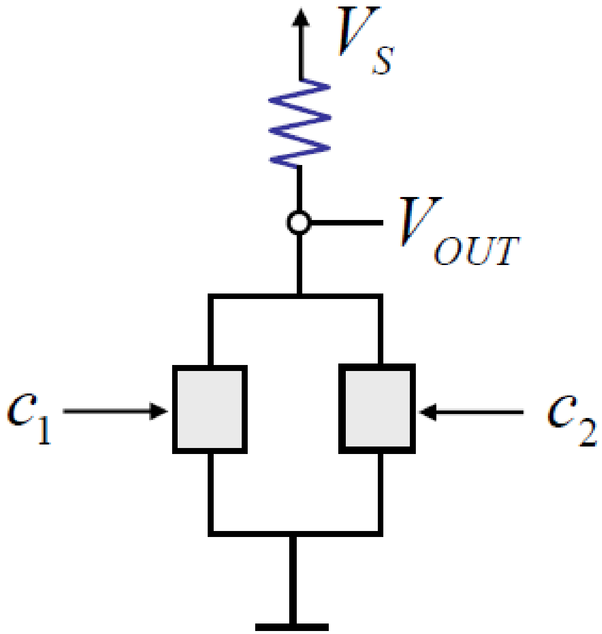
NAND & NOR Gates: Universal Gates

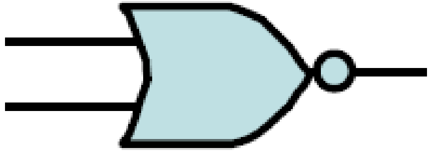




c_1	c_2	V_O
0	0	1
0	1	1
1	0	1
1	1	0

NAND gate



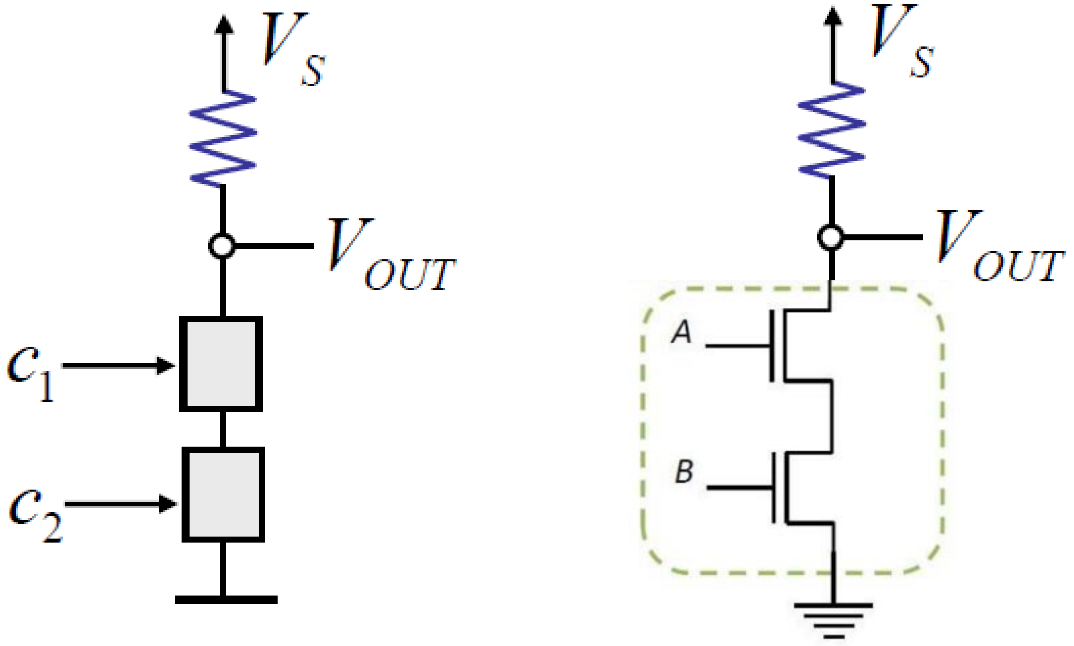


c_1	c_2	V_O
0	0	1
0	1	0
1	0	0
1	1	0

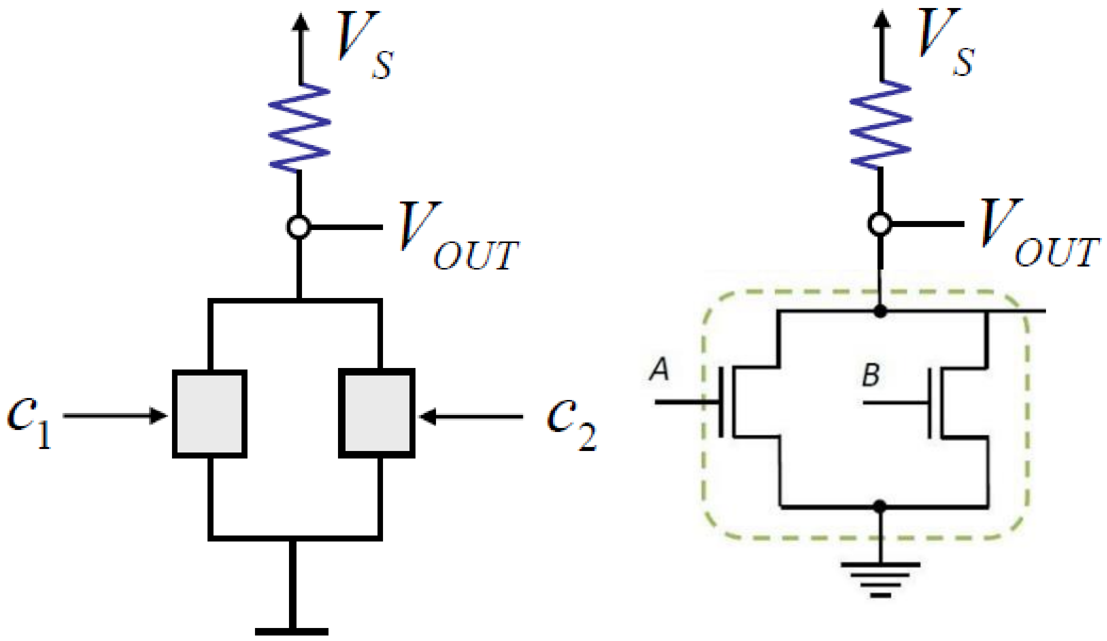
NOR gate

- ❑ NAND and NOR are universal gates.

NAND & NOR Gates: NMOS Implementation



NAND gate

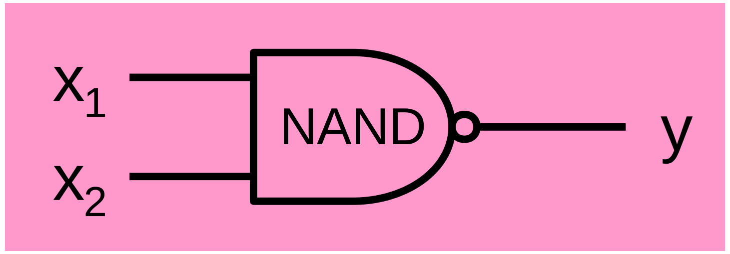


NOR gate

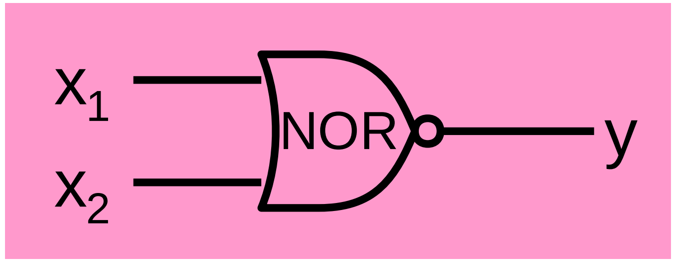
- ❑ NAND and NOR are universal gates.

Logic Gates

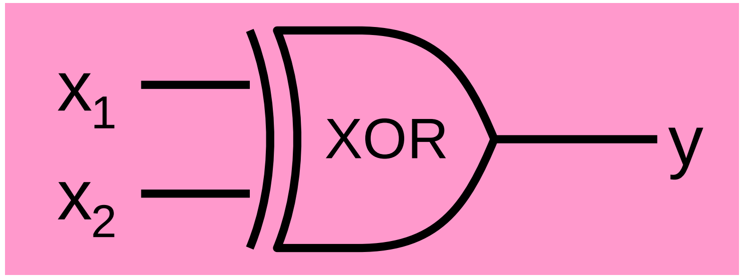
NAND: $y = \overline{x_1 . x_2}$



NOR: $y = \overline{x_1 + x_2}$



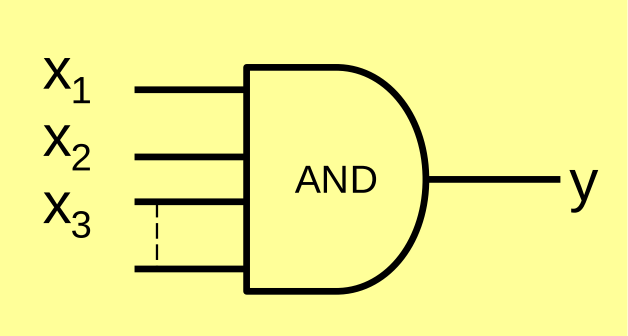
XOR: $y = x_1 \oplus x_2 = x_1 . \overline{x_2} + \overline{x_1} . x_2$



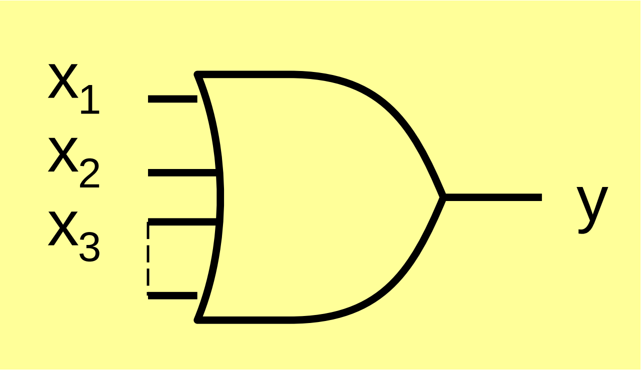
A typical intel core i7 processor has about 500 million (5×10^8) logic gates

Gates with more than 2 inputs

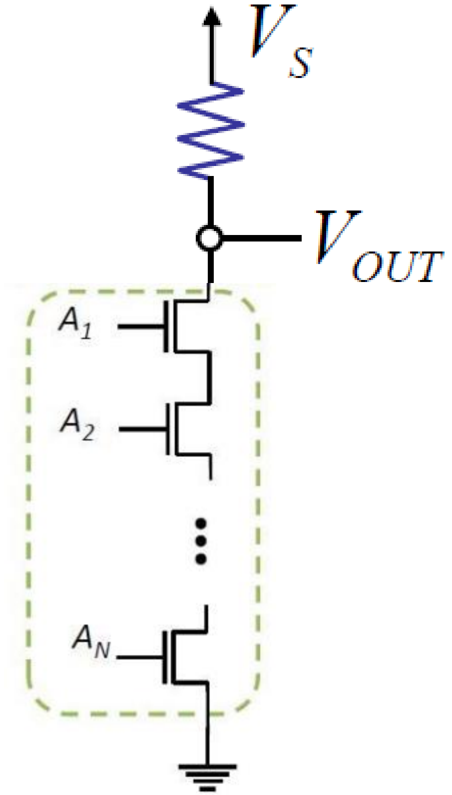
AND: $y = x_1 . x_2 . x_3 \ldots$



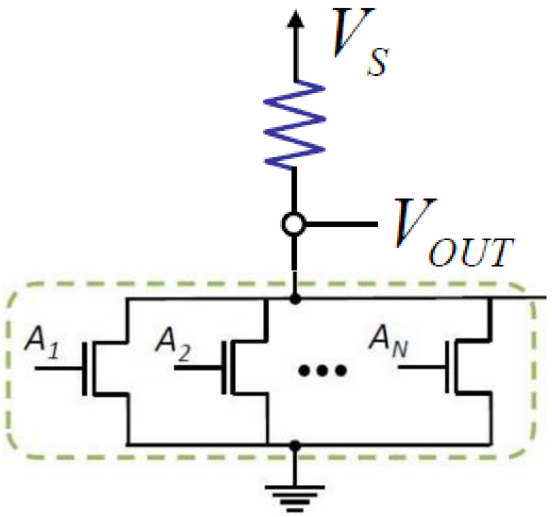
OR: $y = x_1 + x_2 + x_3 + \ldots.$



Multiple-input NAND & NOR Gates Implementation



n-input NAND gate



n-input NOR gate

Boolean Operators

- Basic operators

AND: $y = x_1 . x_2$ OR: $y = x_1 + x_2$ NOT: $y = \overline{x}$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

x	y
0	1
1	0

Truth Tables shows all possible values of an output for all possible combinations of inputs

How to show equivalency of two expressions?

- Compare the truth table on both side
- Show $x + x = x$

x	x	$x + x$
0	0	0
1	1	1

How to show equivalency of two expressions?

- Compare the truth table on both side
- Show $\overline{x\,.\,y} = (\bar{x} + \bar{y})$

x	y	$\overline{x\,.\,y}$	$(\bar{x} + \bar{y})$
0	0	1	1
1	0	1	1
0	1	1	1
1	1	0	0

Basic Postulates and Theorems

- Every algebra has basic theorems/postulates regarding operators
- OR & AND are not exactly like Addition & Multiplication operators
- Commutative

$$x + y = y + x$$

$$x.y = y.x$$

- Distributive

$$x.(y + z) = x.y + y.z$$

$$x + (y.z) = (x + y).(x + z)$$

Basic Postulates and Theorems

P1.a: $x + 0 = x$	P1.b: $x \cdot 1 = x$	Identity element
P2.a: $x + y = y + x$	P2.b: $x \cdot y = y \cdot x$	Commutative
P3.a: $x.(y+z) = x.y+x.z$	P3.b: $x+y.z = (x+y).(x+z)$	Distributive
P4.a: $x + \overline{x} = 1$	P4.b: $x \cdot \overline{x} = 0$	Complement

T1.a: $x + x = x$	T1.b: $x \cdot x = x$	
T2.a: $x + 1 = 1$	T2.b: $x \cdot 0 = 0$	
T3.a: $\overline{(\overline{x})} = x$		
T4.a: $x + (y+z) = (x+y)+z$	T4.b: $x \cdot (y.z) = (x.y).z$	
T5.a: $\overline{(x+y)} = \overline{x} \cdot \overline{y}$	T5.b: $\overline{(x.y)} = \overline{x} + \overline{y}$	DeMorgan's Theorem
T6.a: $x + x.y = x$	T6.b: $x.(x+y) = x$	

How to verify an expression?

- Using truth tables
- Showing using postulates

$$\overline{(\overline{x_1}.x_2 + \overline{x_2}.x_3)} = x_1.\,x_2 + \overline{x_2}.x_2 + x_1\,.\,\overline{x_3} + \overline{x_2}.\,\overline{x_3}$$

Boolean Algebra

$$\overline{(\overline{x}_1 . x_2 + \overline{x}_2 . x_3)} = (\overline{\overline{x}_1 . x_2}) . (\overline{\overline{x}_2 . x_3})$$

$$= (\overline{\overline{x}}_1 + \overline{x}_2) . (\overline{\overline{x}}_2 + \overline{x}_3)$$

$$= (x_1 + \overline{x}_2) . (x_2 + \overline{x}_3)$$

$$= x_1 . x_2 + \overline{x}_2 . x_2 + x_1 . \overline{x}_3 + \overline{x}_2 . \overline{x}_3$$

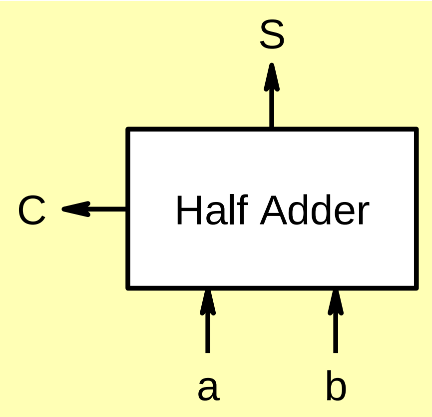
$$= x_1 . x_2 + x_1 . \overline{x}_3 + \overline{x}_2 . \overline{x}_3$$

Calculation using Binary System

- Binary signals represent logic states
  - can implement any logic
  - logic consists of AND, OR & NOT condition
  - Boolean Algebra
- Binary signals can represent any numbers
  - We can do all arithmetic over it
  - Enables any calculations
  - Addition, Subtraction, Multiplication, Division, etc.
- Computers can do calculations
  - evaluate logic states and make decision over that

How to do such calculations?

Addition



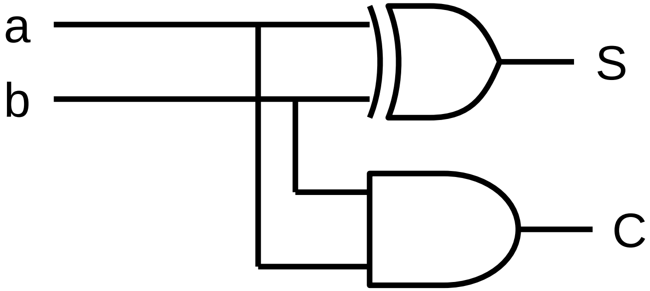
a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth Table

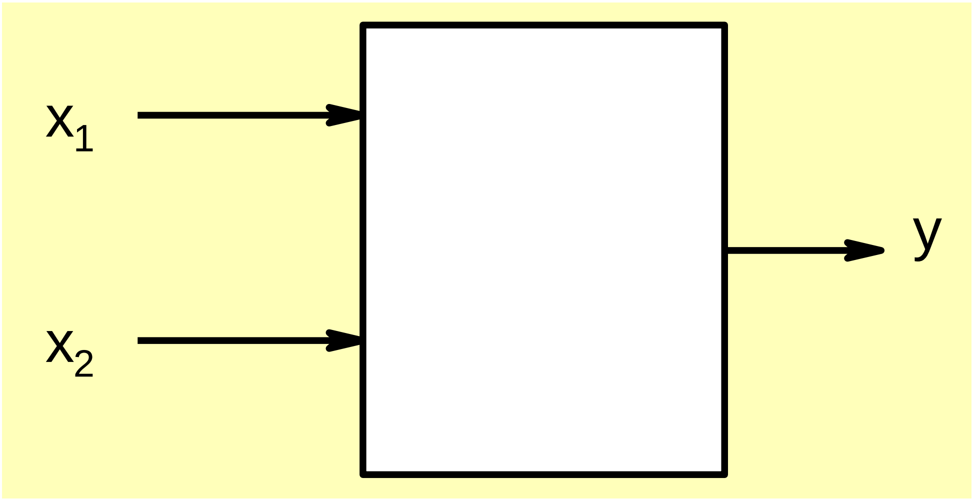
$$S = \bar{a}.b + a.\bar{b}; C = a.b$$

How to get this expression?

How to get this gate implementation?



How to get an expression from truth table?



x_1	x_2	y
0	0	0
0	1	1
1	0	0
1	1	0

$y = 1$ when x_1 is 0 and x_2 is 1

Boolean expression $$y = \overline{x_1} \,.\, x_2$$ (NOT x_1) AND x_2

How to get an expression from truth table?

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	0

$$y = \overline{x_1} \, . \, \overline{x_2}$$

x_1	x_2	y
0	0	0
0	1	0
1	0	1
1	1	0

$$y = x_1 \, . \, \overline{x_2}$$

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

y_1 $\overline{x_1} \, . \, \overline{x_2}$ y_2

y_1	y_2
1	0
0	0
0	0
0	1

$x_1 \, . \, x_2$

$y = y_1 + y_2$

$$y = \overline{x_1} \, . \, \overline{x_2} + x_1 . x_2$$

(NOT x_1) AND (NOT x_2) OR x_1 AND x_2

Boolean Expressions & Truth Tables

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

$\overline{x_1} \cdot \overline{x_2}$

$x_1 \cdot x_2$

$$y = \overline{x_1} \cdot \overline{x_2} + x_1 \cdot x_2$$

Sum of Products (SOP) form

Boolean Expressions & Truth Tables

Instead of writing expressions as sum of terms that make y equal to 1, we can also write expressions using terms that make y equal to 0

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

$$y = \overline{x_1} . \overline{x_2} + \overline{x_1} . x_2 + x_1 . \overline{x_2}$$

Here we are telling when y will be true

$$y = \overline{x_1 . x_2}$$

FALSE when both are true

Here we are telling when y will be false

$$y = \overline{x_1} + \overline{x_2}$$

Recall

$\overline{x_1} + \overline{x_2} = \overline{x_1 . x_2}$

Boolean Expressions & Truth Tables

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

$$y = \overline{x_1} . x_2 + x_1 . \overline{x_2}$$

Sum of Products (SOP) form

Boolean Expressions & Truth Tables

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

$$x_1 + x_2$$

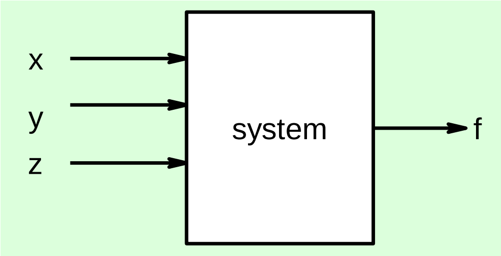
$$\overline{x_1} + \overline{x_2}$$

$$y = (x_1 + x_2).(\overline{x_1} + \overline{x_2})$$

Product of Sum (POS) form

Digital Design

System Description



Truth Table

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Boolean Expression

$$\mathrm{f} = \bar{x}.\bar{y}.z + \bar{x}.y.z + x.\bar{y}.z + x.y.z$$

Minimized Boolean Expression

$$\Rightarrow \ \mathrm{f} = z$$

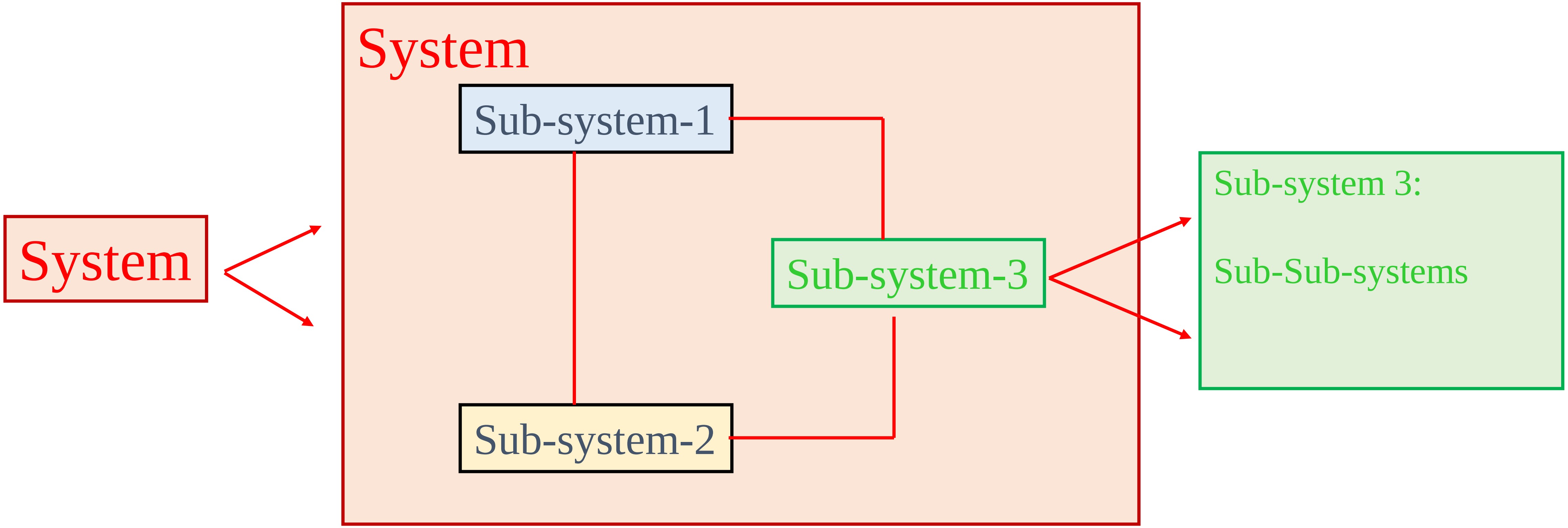
Gate Netlist

Calculation using Digital System

- Binary signals represent logic states
  - can implement any logic
  - logic consists of AND, OR & NOT condition
  - Boolean Algebra
- Binary signals can represent any numbers
  - We can do all arithmetic over it
  - Enables any calculations
  - Addition, Subtraction, Multiplication, Division, etc.
- Computers can do calculations
  - evaluate logic states and make decision over that

How to do such calculations?

Modular Approach



There are certain sub-systems or blocks that are used quite often such as :

1. Adder/Subtractors, Multipliers
2. Decoders, Encoders
3. Multiplexers, Demultiplexers
4. Comparators
5. Parity Generators

Binary Addition

```
 0
 0
---
 0
---
```

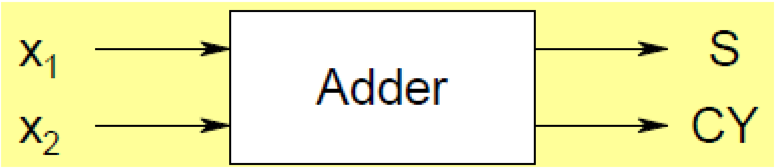
```
 1      0
 0      1
---    ---
 1      1
---    ---
```

```
  1
  1
----
 1 0
----
```

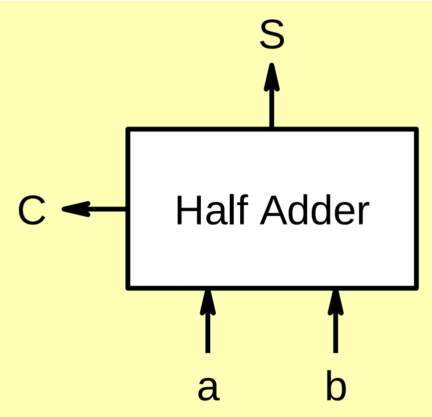
```
  1
  1
  1
----
 1 1
----
```

```
   1 0 1
   1 1 0
--------
 1 0 1 1
--------
```

```
    1 1 0 1
 +  1 1 1 0
-----------
  1 1 0 1 1
-----------
```



1 bit Addition: Half Adder

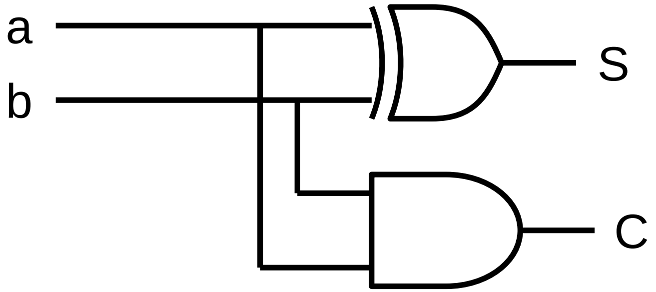


a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Truth Table

$$S = \bar{a}.b + a.\bar{b} ; C = a.b$$

Boolean Expression



Gate level implementation

Why a Modular Approach?

- Let us make a 2 bit adder circuit which can add two 2-bit numbers

$$\begin{array}{cccc} & & x_1 & x_0 \\ + & & y_1 & y_0 \\ \hline & z_2 & z_1 & z_0 \end{array}$$

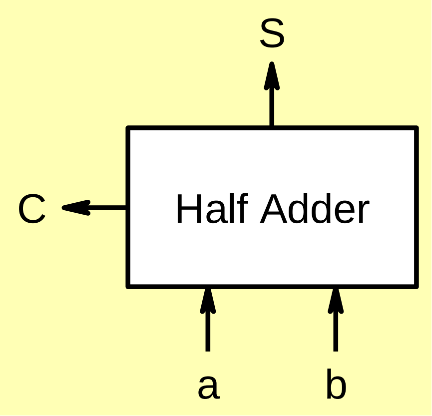
- There are 4 inputs and 3 outputs
- Let us write down all possible combinations!
  - $2^4 = 16$ rows in the truth table

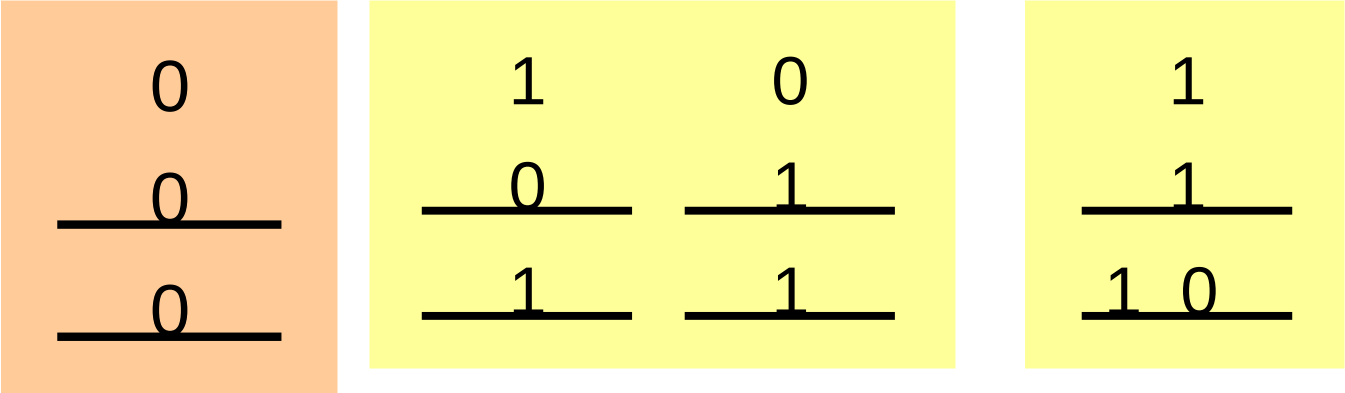
➢ Write down Boolean expressions and design implementation?

➢ What about 3 bits?

❑ Let us take the modular approach

Adder: First bit

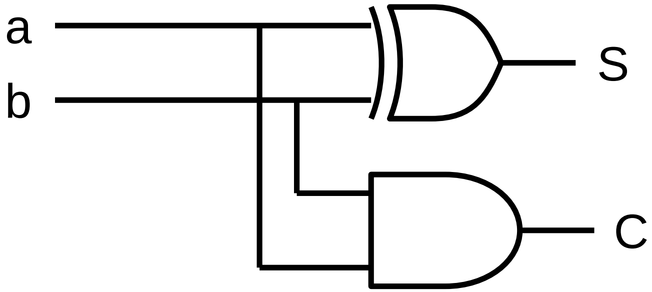




Truth Table

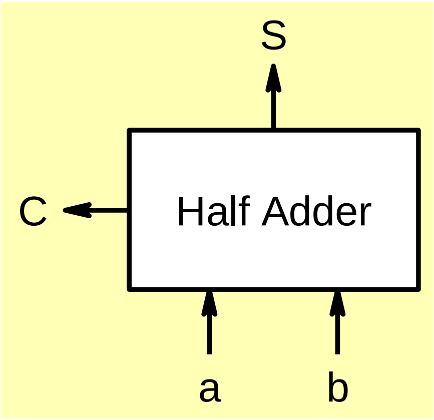
a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = \bar{a}.b + a.\bar{b}; C = a.b$$



Implementation

Adder: Second bit

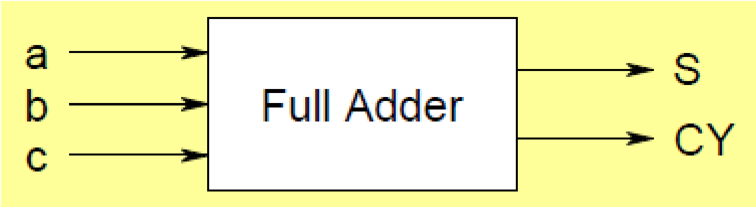


$$\begin{array}{r} 0 \\ \underline{0} \\ \underline{0} \end{array} \qquad \begin{array}{r} 1 \\ \underline{0} \\ \underline{1} \end{array} \qquad \begin{array}{r} 0 \\ \underline{1} \\ \underline{1} \end{array} \qquad \begin{array}{r} 1 \\ \underline{1} \\ \underline{1\ 0} \end{array}$$

But there can be carry from previous bits.

$$\begin{array}{cccc} & {\color{blue}1} & & \\ & x_1 & 1 \\ + & y_1 & 1 \\ \hline z_2 & z_1 & 0 \end{array}$$

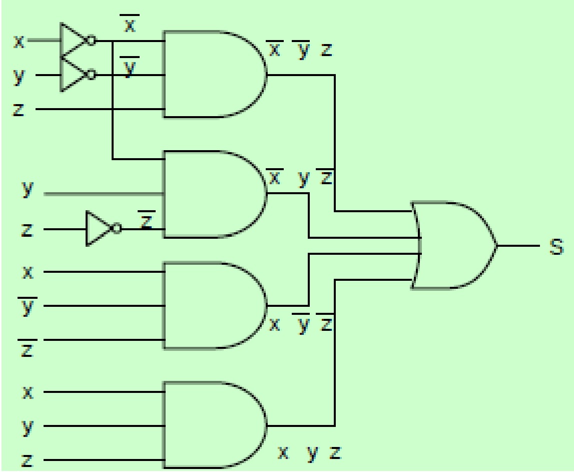
Single Bit Full Adder

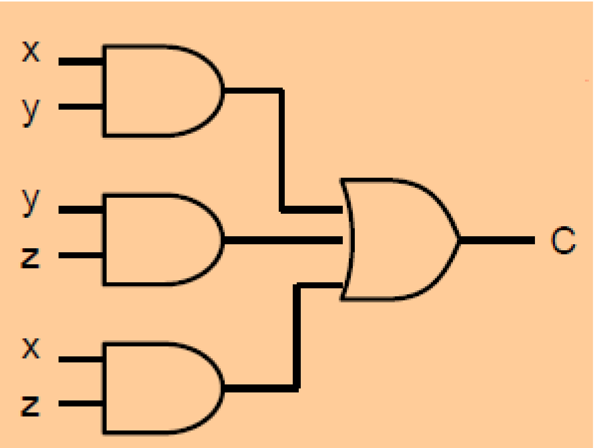


$$S = \overline{x}.\overline{y}.z + \overline{x}.y.\overline{z} + x.\overline{y}.\overline{z} + x.y.z$$

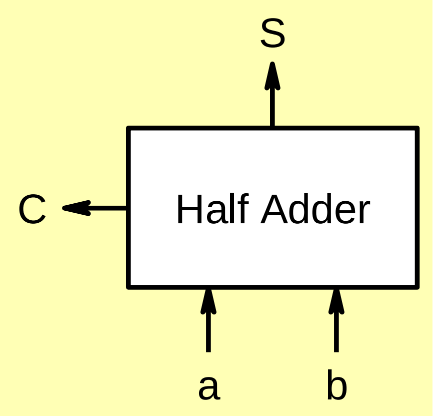
$$C = x.y + x.z + y.z$$

a	b	c	S	CY
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

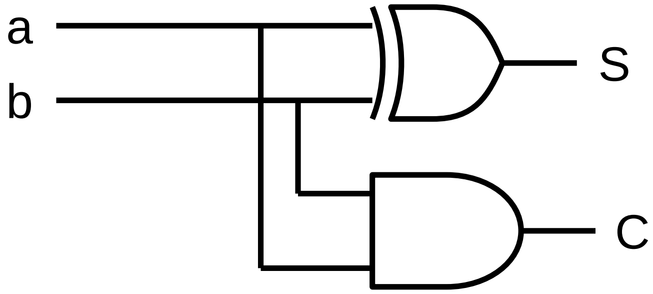




Adder: Half Adder vs Full adder

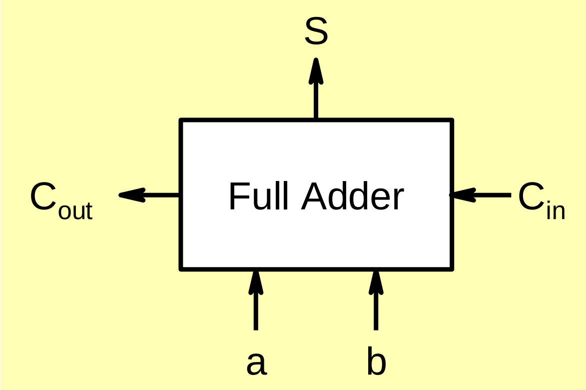


a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$$S = \bar{a}.b + a.\bar{b}\,; C = a.b$$

1
1 1 1
1 1 0
----
1 1 0 1



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

$$S = \bar{a}.\bar{b}.c_{in} + \bar{a}.b.\overline{c_{in}} + a.\bar{b}.\overline{c_{in}} + a.b.c_{in}\,;$$

$$C_{out} = \bar{a}.b.c_{in} + a.\bar{b}.c_{in} + a.b.\overline{c_{in}} + a.b.c_{in}$$

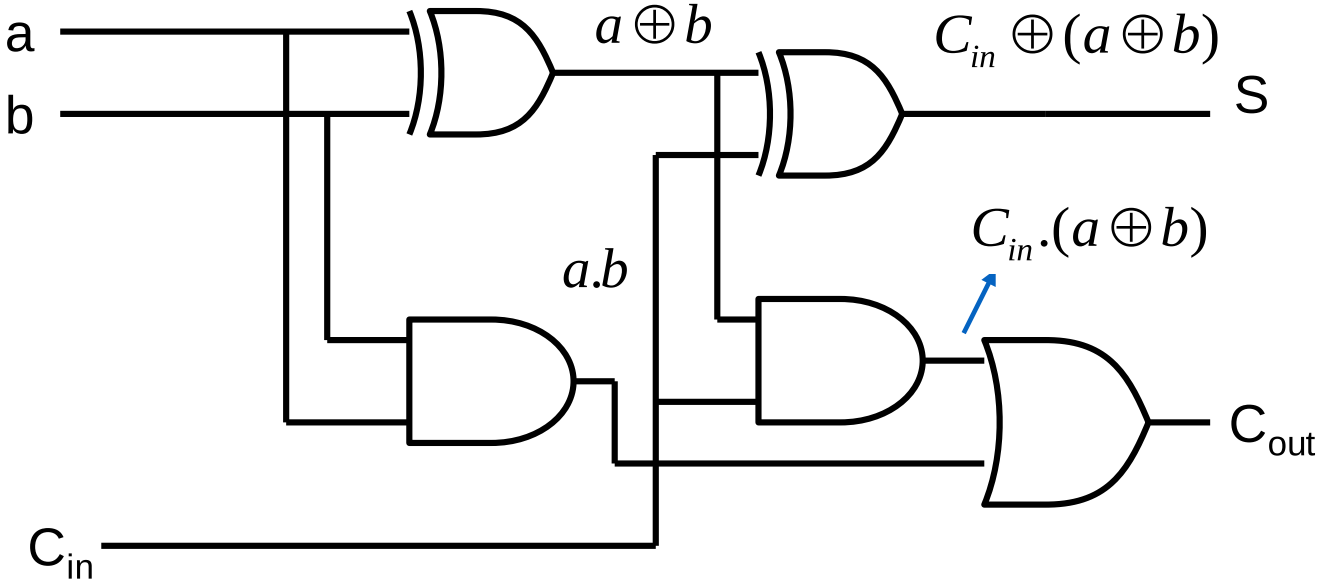
Full Adder Circuit using Half Adders

$$S = \overline{a}.\overline{b}.c_{in} + \overline{a}.b.\overline{c_{in}} + a.\overline{b}.\overline{c_{in}} + a.b.c_{in}$$

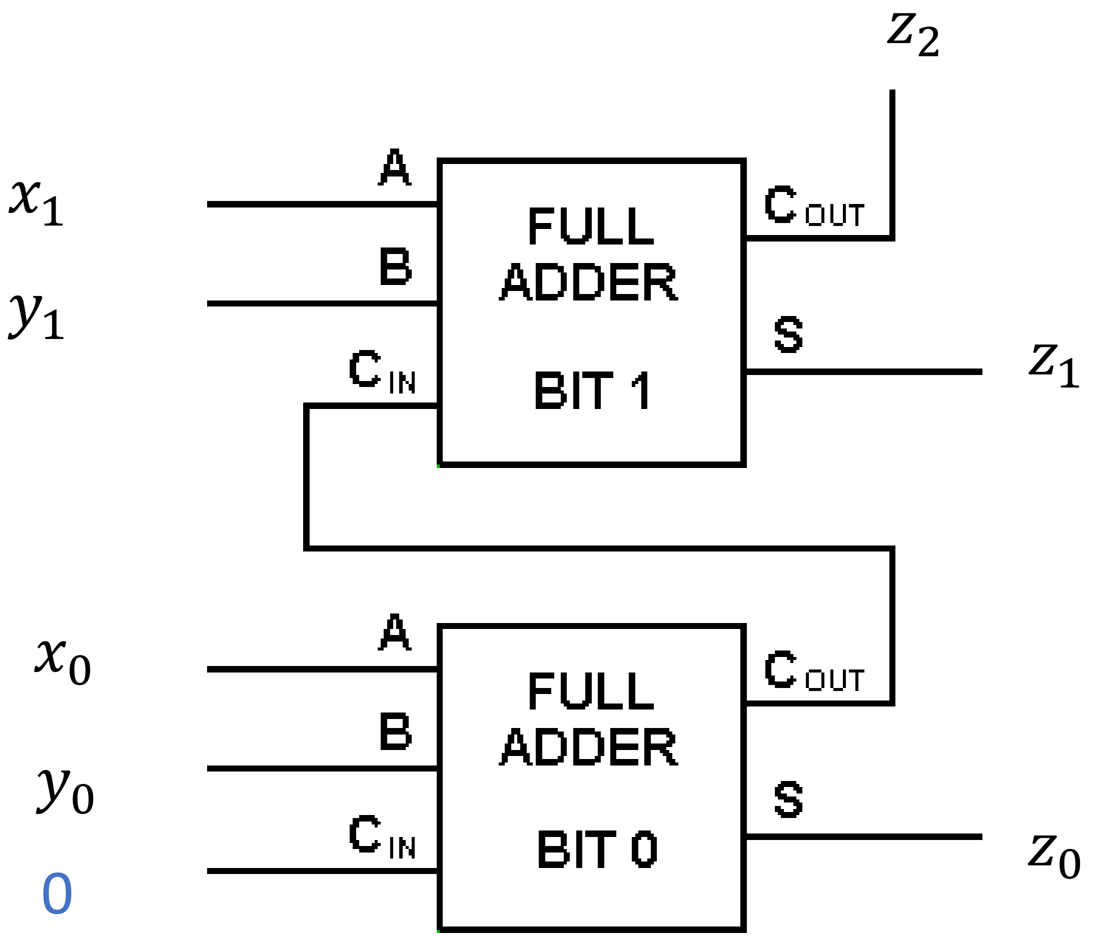
$$S = C_{in} \oplus (a \oplus b)$$

$$C_{out} = \overline{a}.b.C_{in} + a.\overline{b}.C_{in} + a.b.\overline{C_{in}} + a.b.C_{in}$$

$$C_{out} = C_{in}(a.\overline{b} + \overline{a}.b) + a.b = C_{in}.(a \oplus b) + a.b$$



Multi-bit Adder

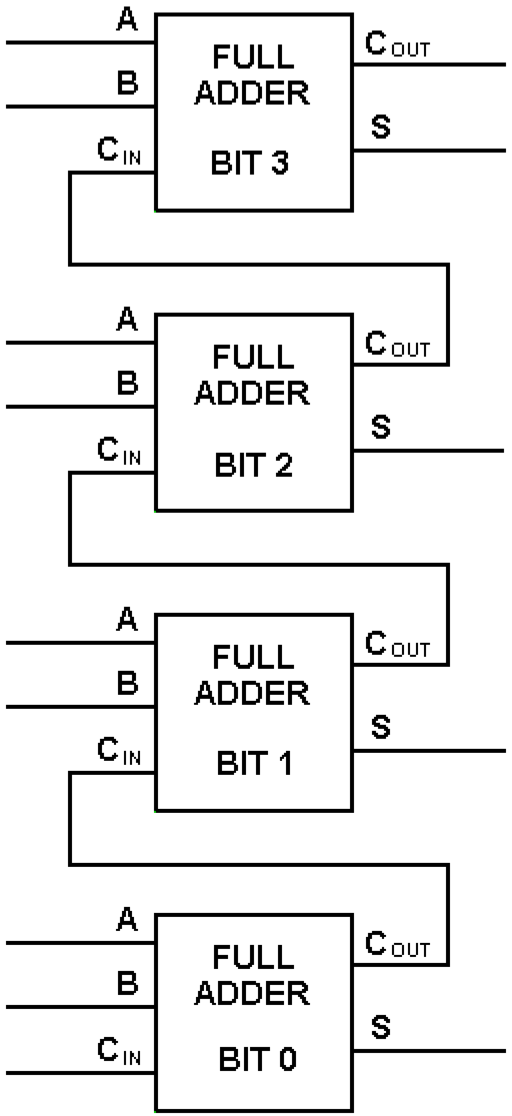


$$
\begin{array}{rcc}
 & x_1 & x_0 \\
+ & y_1 & y_0 \\
\hline
z_2 & z_1 & z_0
\end{array}
$$

Multi-bit Adder

- How to add two 4-bit numbers?
- Truth table would have 2^8=256 entries
- Instead, use already designed logic circuits as subsystems

$$\begin{array}{r} 1\,1\,0\,1 \\ +\;1\,1\,1\,0 \\ \hline 1\,1\,0\,1\,1 \end{array}$$



Addition/Subtraction Computation

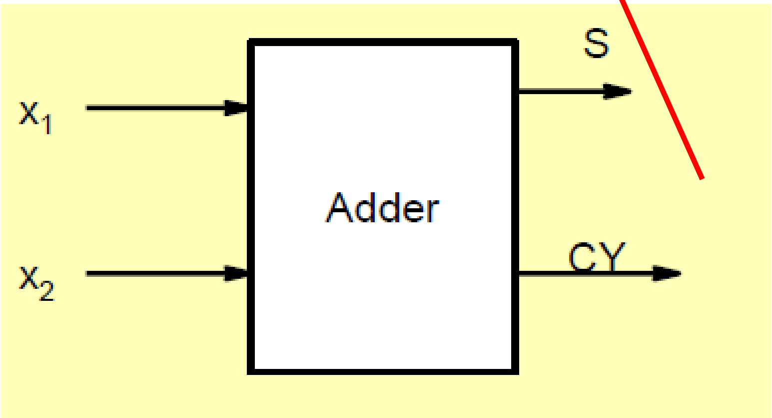
$$\begin{array}{r} +5 \\ +2 \\ \hline +7 \end{array} \qquad \begin{array}{r} 0101 \\ +\,0010 \\ \hline 0111 \end{array} \qquad\qquad \begin{array}{r} +5 \\ -2 \\ \hline +3 \end{array} \qquad \begin{array}{r} 0101 \\ +\,1110 \\ \hline 0011 \end{array}$$

$$\begin{array}{r} -5 \\ +2 \\ \hline -3 \end{array} \qquad \begin{array}{r} 1011 \\ +\,0010 \\ \hline 1101 \end{array} \qquad\qquad \begin{array}{r} -5 \\ -2 \\ \hline -7 \end{array} \qquad \begin{array}{r} 1011 \\ +\,1110 \\ \hline 1001 \end{array}$$

2's complement is 0011 = 3

2's complement is 0111 = 7

Answer is in 2's complement form



x_1,x_2: N bit numbers in 2's complement

Overflow

- Take care to detect overflow when adding

+ 5	0 0 1 0 1
+ 13	0 1 1 0 1
+ 18	0 1 0 0 1 0

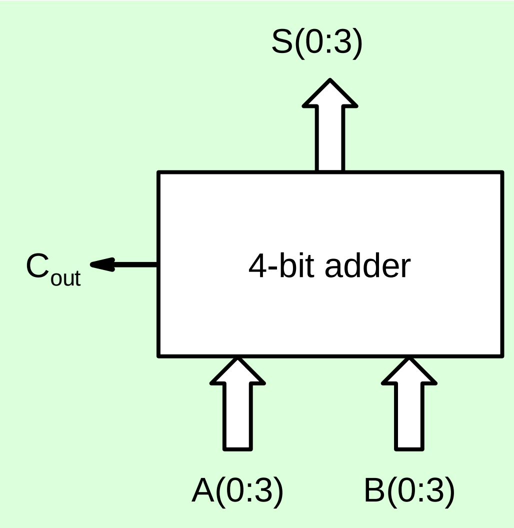
After discarding the final carry 0,
2's complement of 10010 is 01110 = $(14)_{10}$
We get a wrong answer!

- Sum of positive numbers = negative
- Sum of negative numbers = positive

overflow

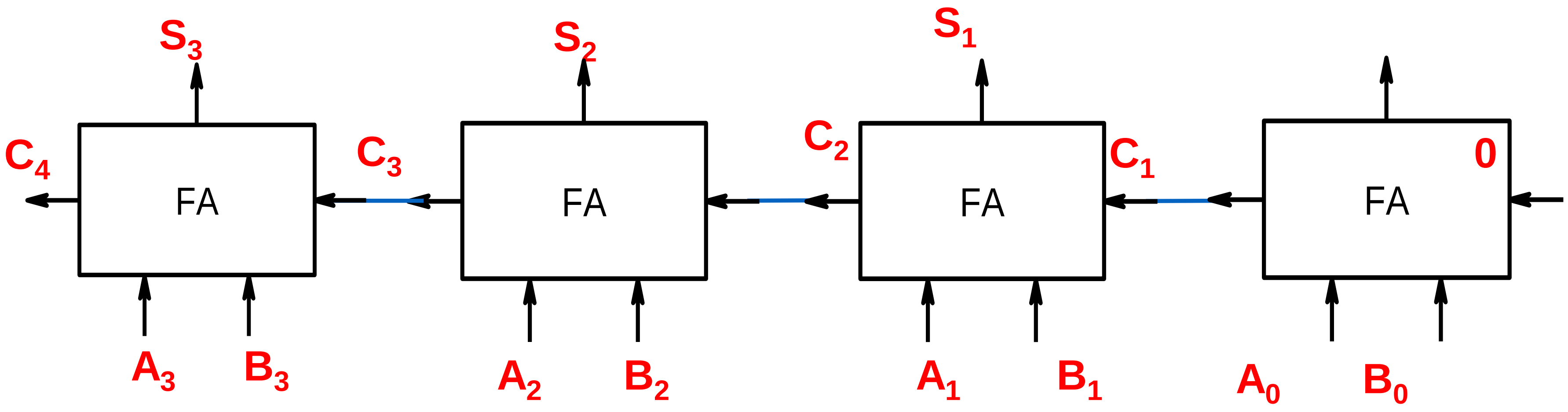
August 2016: Casino machine at Resorts World Casino printed a prize ticket of $42,949,672.76 as a result of an overflow bug. The Casino refused to pay this amount calling it a malfunction. The Iowa Supreme Court ruled in favor of the Casino.

4-bit Adder



$A_3A_2A_1A_0$	$B_3B_2B_1B_0$	$S_3S_2S_1S_0$	C_{out}
0000	0000	0000	0
0000	0001	0001	0
0001	0000	0001	0
⋮	⋮	⋮	⋮

$$
\begin{array}{ccccc}
 & C_3 & C_2 & C_1 & \\
 & A_3 & A_2 & A_1 & A_0 \\
 & B_3 & B_2 & B_1 & B_0 \\
\hline
C_4 & S_3 & S_2 & S_1 & S_0 \\
\hline
\end{array}
$$

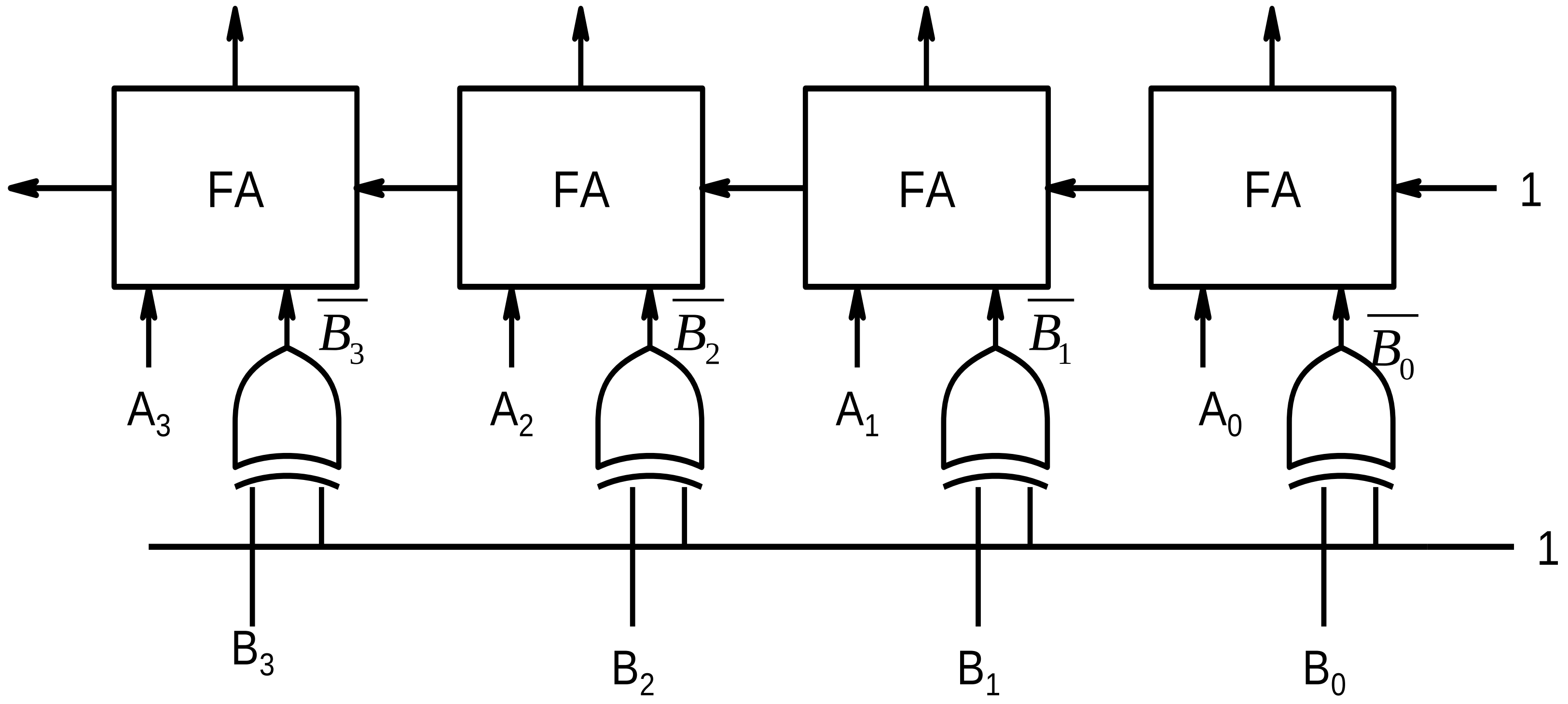


4-bit Subtractor

A – B = A + 2's complement of B

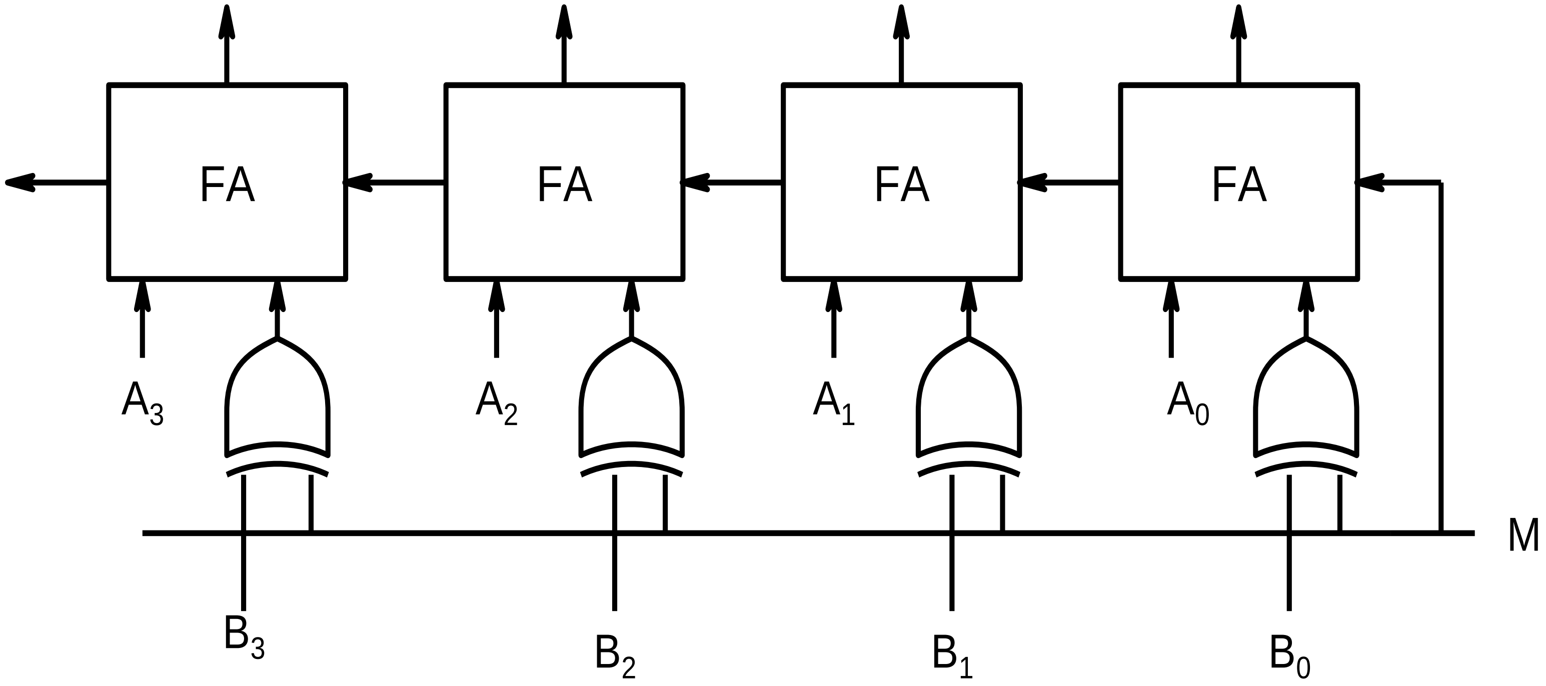
A – B = A + 1's complement of B + 1

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



$$B_0 \oplus 1 = B_0.\bar{1} + \overline{B_0}.1 = \overline{B_0}$$

4-bit Adder and Subtractor



$$B_0 \oplus 0 = B_0.\bar{0} + \overline{B_0}.0 = B_0$$

$$B_0 \oplus 1 = B_0.\bar{1} + \overline{B_0}.1 = \overline{B_0}$$

$M = 0$ for Adder

M=1 for Subtractor