

# Simple Neural Nets for Pattern Classification: McCulloch-Pitts *Threshold Logic* CS 5870

Jugal Kalita  
University of Colorado Colorado Springs  
Fall 2014

# Logic Gates and Boolean Algebra

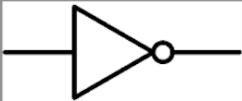
- Logic gates are used as building blocks of electronic circuits and to provide graphical representations to fundamental Boolean operators.
- Operators include: NOT, AND, OR, NAND, NOR, XOR, etc.
- One Basis set: {NOT, AND, OR} (all other functions can be constructed out of these three); NOR is a basis set by itself; NAND is a basis set by itself; {XOR, NOT} is a basis set.

# Standard Logic Gates

Standard Logic Gates

NOT

A



B

A

1

0

B


0

1

AND

A

B



C

A

1

1

0

0

B

1

0

1

0

C

1

0


0

0

NAND

A

B



C

A

1

1

0

0

B

1

0

1

0

C

0

1


1

1

OR

A

B



C

A

1

1

0

0

B

1

0

1

0

C

1

1


1

0

NOR

A

B



C

A

1

1

0

0

B

1

0

1

0

C

0

0


0

1

XOR

A

B



C

A

1

1

0

0

B

1

0

1

0

C

0

1


1

0

XNOR

A

B



C

A

1

1

0

0

B

1

0

1

0

C

1

0

0

1

From  
[http://  
 picokit.  
 com.au](http://picokit.com.au)

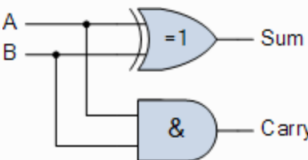
# Circuits from Logic Gates

- We can create complex circuits by networking a number of logic gates.
- For example, we can build an 4-bit (or more bit) binary adder circuit by building
  - 1-bit adder with carry-out (half adder)
  - Full-adder with carry-in
  - 4-bit binary adder and so on
  - Of course, we can build enormously complex circuits as well.

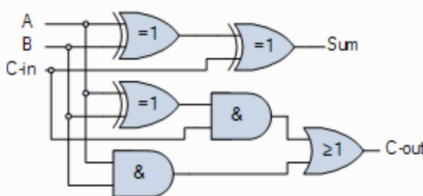
# Building a Full Adder

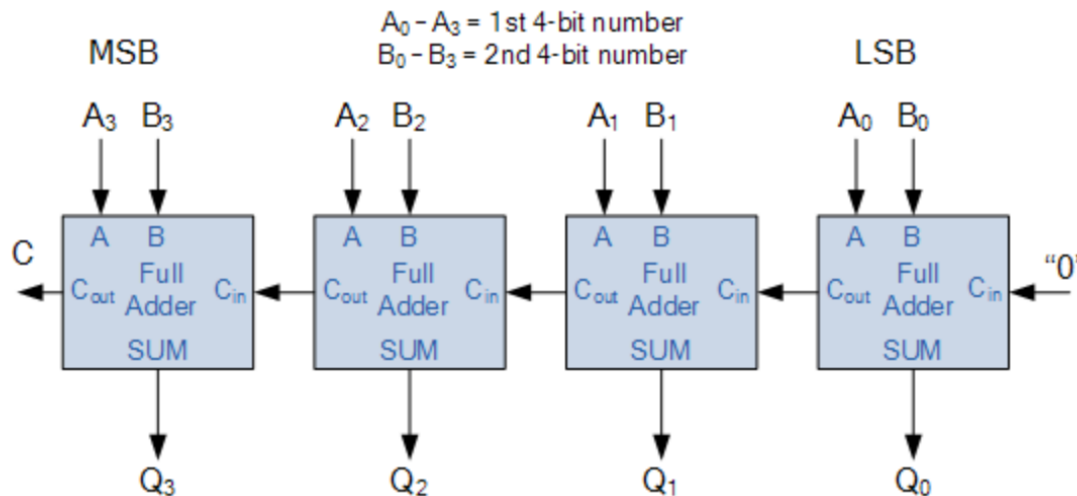
## The Half Adder Circuit

### 1-bit Adder with Carry-Out

Symbol	Truth Table			
	A	B	SUM	CARRY
	0	0	0	0
	0	1	1	0
	1	0	1	0
	1	1	0	1
Boolean Expression: $\text{Sum} = A \oplus B$ $\text{Carry} = A \cdot B$				

### Full Adder with Carry-In

Symbol	Truth Table<				
	A	B	C-in	Sum	C-out
	0	0	0	0	0
	0	1	0	1	0
	1	0	0	1	0
	1	1	0	0	1
	0	0	1	1	0
	0	1	1	0	1
	1	0	1	0	1
	1	1	1	1	1
Boolean Expression: $\text{Sum} = A \oplus B \oplus \text{C-in}$					

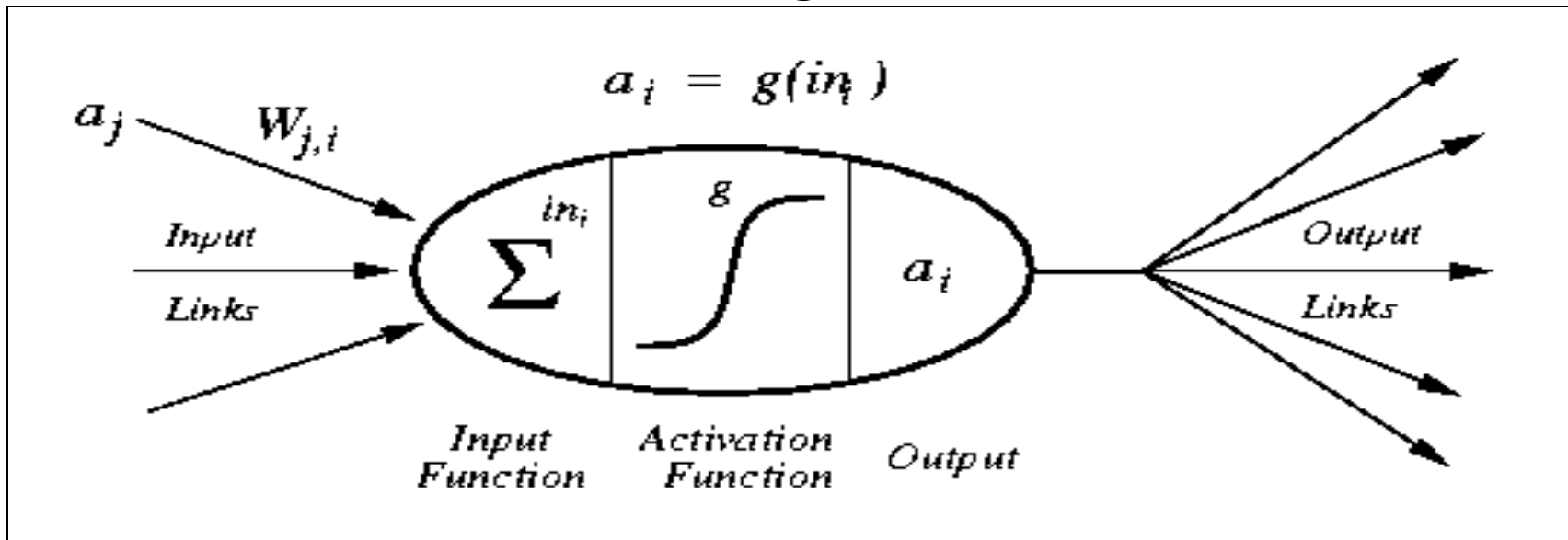


From <http://www.electronicstutorials.ws>

# Artificial Neural Networks

- McCulloch and Pitts (1943) tried to build something similar to the logic gates we just saw, but using threshold logic, using inspirations from actual neurons
- McCulloch & Pitts are generally recognized as the designers of the first artificial neural networks.
- Many of their ideas still used today, e.g.,
  - many simple units, “neurons” combine to give increased computational power
  - the idea of a threshold

# Modelling a Neuron



$$in_i = \sum_j W_{j,i} a_j$$

- $a_j$  : Activation value of unit j
- $w_{j,i}$  : Weight on link from unit j to unit i
- $in_i$  : Weighted sum of inputs to unit i
- $a_i$  : Activation value of unit i
- $g$  : Activation function

# Characteristics of McCulloch-Pitts ANN

- The activation is binary. A neuron fires when its activation is 1, otherwise, its activation is 0.
- Neurons are connected by directed, (weighted in Fausette's book, unweighted in Rojas' book) paths.
- A connection path is excitatory if the weight on the path is positive; otherwise, the path is inhibitory.
- All excitatory connections into a particular neuron have the same weights.

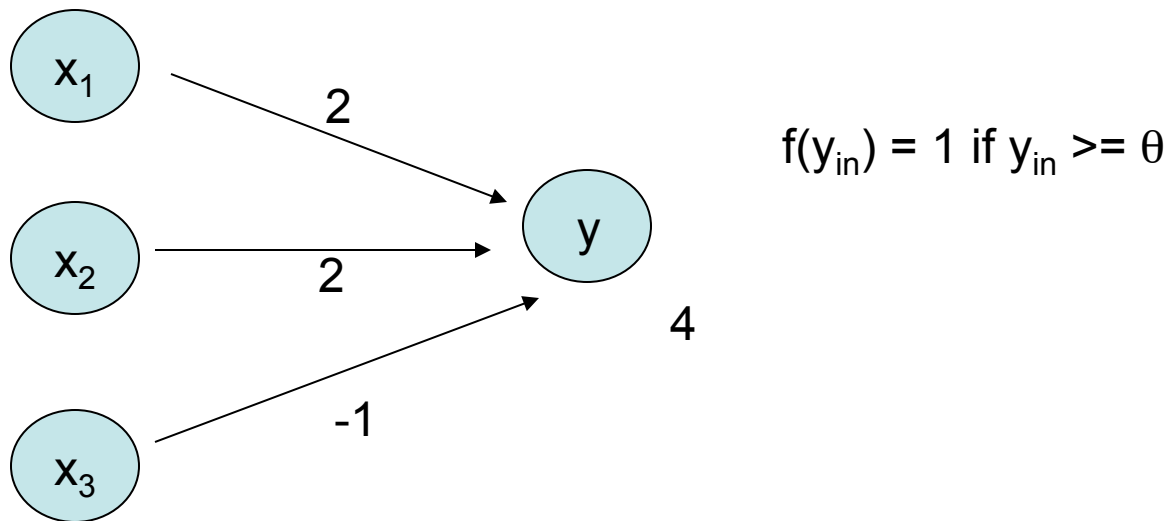


# Characteristics of McCulloch-Pitts ANN

- Each neuron has a fixed threshold. If the input to the neuron is greater than the threshold, the neuron fires.
- *Inhibition is absolute.* If there is even one inhibitory input, the neuron will not fire. The weights are set to make this happen.
- It takes one time step for a signal to pass over a connection link.

# McCulloch-Pitts Neuron

In this example, there are two excitatory links, one inhibitory link, the threshold for the unit is 4. All links are forward feeding. The values of the inputs determine the value of the output 1 time unit later.

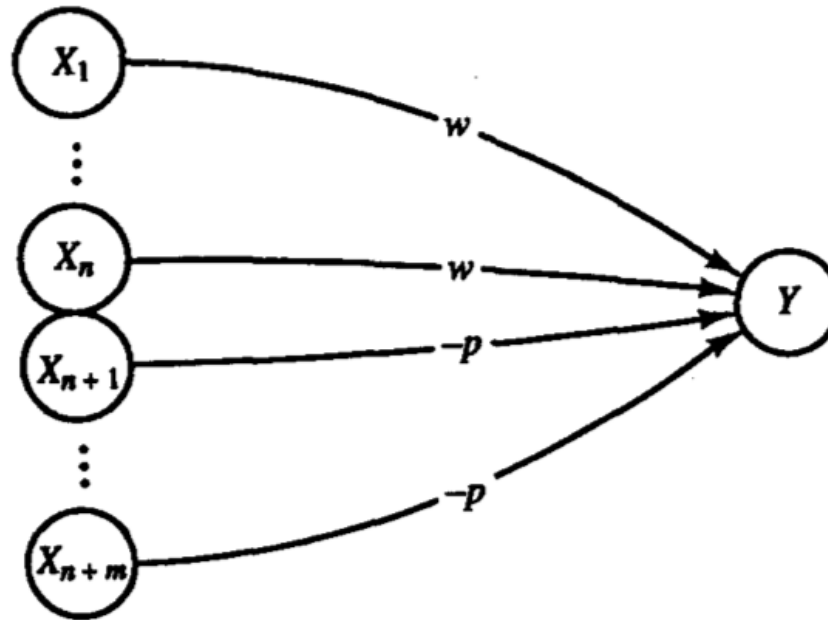


The weights are other than 1 and -1 in Fausette's discussions. They are 1 and -1 in Rojas' discussions.

# Learning: Rules

- *McCulloch-Pitts Neurons*: Obtain the weights by performing analysis of the problem

# McCulloch-Pitts Neuron: General Architecture



Although all excitatory weights into a particular unit are the same, the inputs coming to a unit do not have to be the same.

The condition that inhibition is absolute requires that the threshold  $\theta$  for the activation function satisfy the inequality

$$\theta > nw - p$$

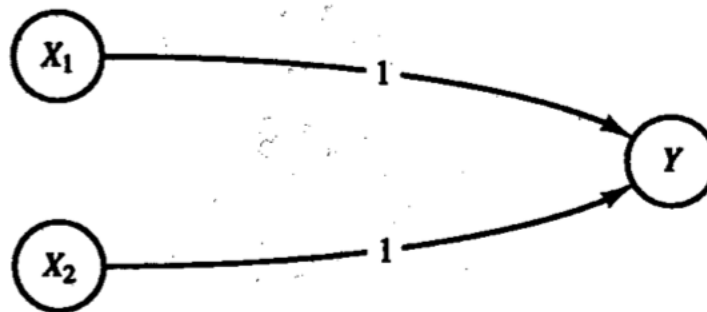
# Network & Weights

- The weights for McCulloch-Pitts neurons along with the threshold are set without learning, but by analysis of the problem at hand.
- We will discuss a few simple networks for solving logical problems.
- We will see that such simple networks can be put together to solve more complex problems.
- We will build McCulloch-Pitts networks for AND, AND NOT and XOR logical operations using Fausette's notation and then more complex ones using Roja's notation.

# McCulloch-Pitts: Logical AND

Unit threshold =  $\theta = 2$

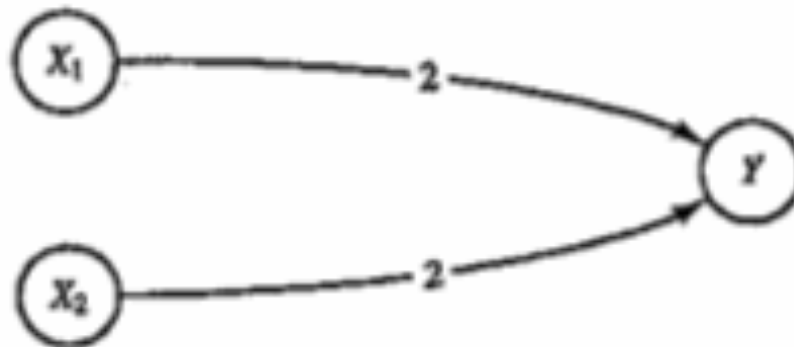
$x_1$	$x_2$	$y$
0	0	0
0	1	0
1	0	0
1	1	1



# McCulloch-Pitts: Logical OR

Unit threshold =  $\theta = 2$

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

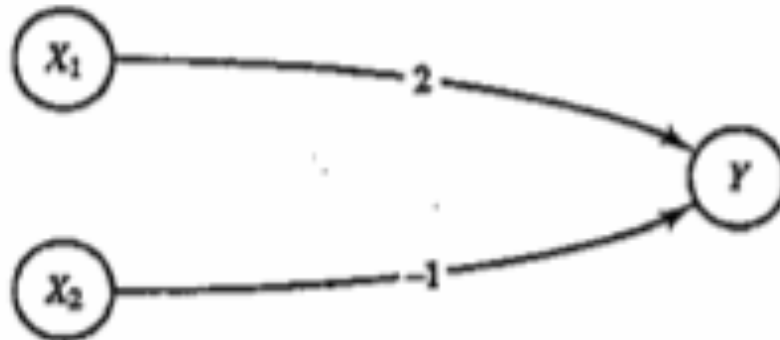


# McCulloch-Pitts: Logical AND NOT

Unit threshold =  $\theta = 2$

Unit is fired when  $X_1$  is true and  $X_2$  is false

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

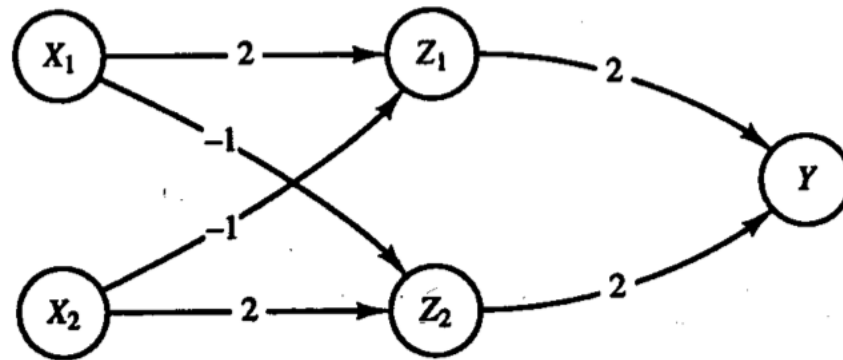




# McCulloch-Pitts: Logical XOR

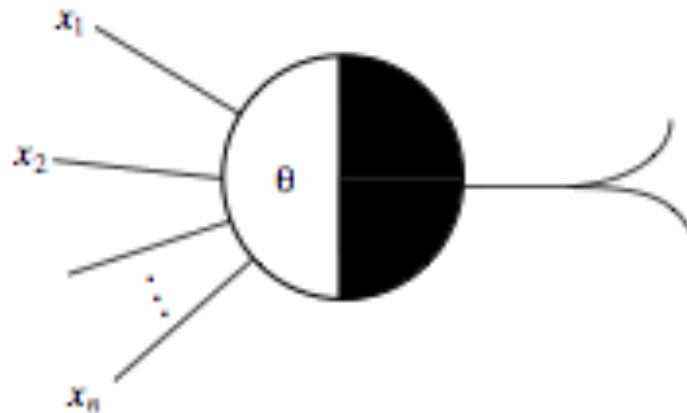
We need a two level network.

$$X_1 \text{ XOR } X_2 == (X_1 \text{ AND NOT } X_2) \text{ OR } (X_2 \text{ AND NOT } X_1)$$



# McCulloch-Pitts: Using Rojas' notation and diagram

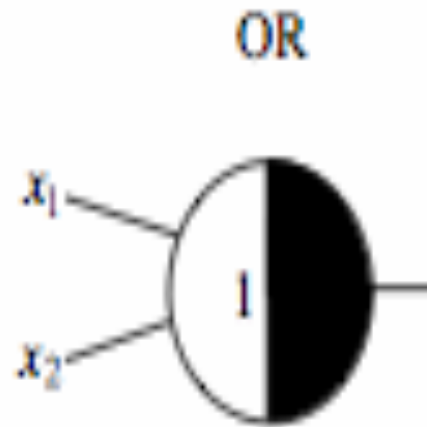
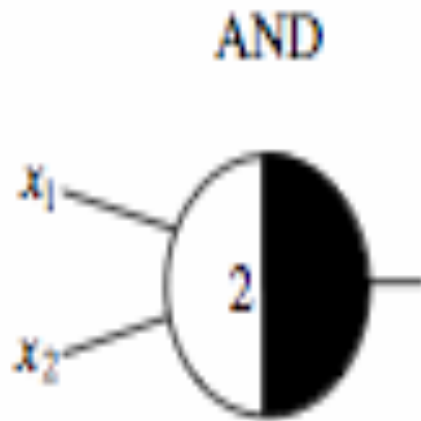
Rojas uses notation by Minsky, the nodes are divided into a white half and a black half. The threshold is given in the node in the white side. **The links are all weighted one, plus or minus.** Negation is indicated by a small circle against the unit. Outgoing edges can fan out any number of times.



# McCulloch-Pitts: AND and OR

Using Rojas notation/diagram

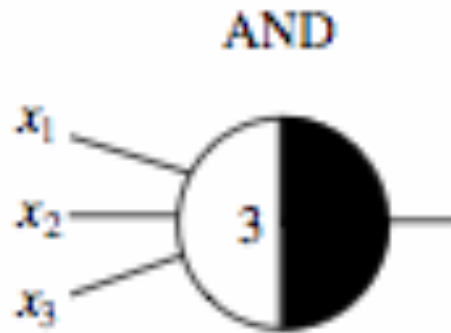
Assume each edge weight is 1



# McCulloch-Pitts: General AND and OR

Using Rojas notation/diagram

We can increase the number of inputs to 3, 4, ... All we need to do is choose the threshold appropriately.

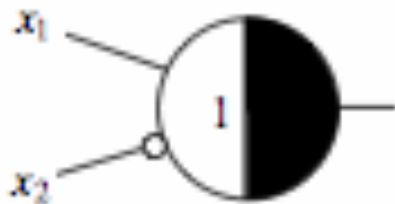


# McCulloch-Pitts: AND NOT, NOR and NOT

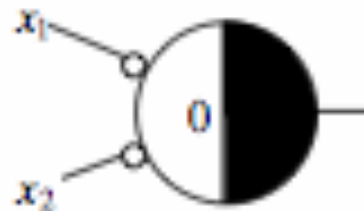
Using Rojas notation/diagram

The small circle at the end means that the weight is -1.  
Otherwise, the weight is 1.

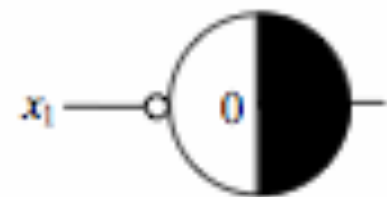
$x_1$  AND  $\neg x_2$



NOR



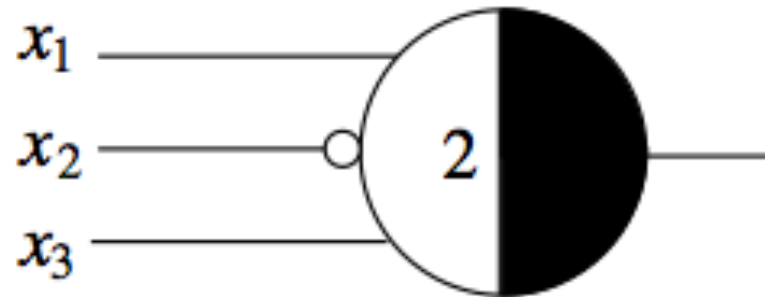
NOT



# McCulloch-Pitts: General Logical Function

Using Rojas notation/diagram

$Y = x_1 \text{ and } (\text{not } x_2) \text{ and } x_3$



# McCulloch-Pitts: General Logical function

Using Rojas notation/diagram

Consider a function of three inputs  $x_1, x_2, x_3$ .

Create a McCulloch-Pitts ANN for this function.

We proceed just like Boolean circuits.

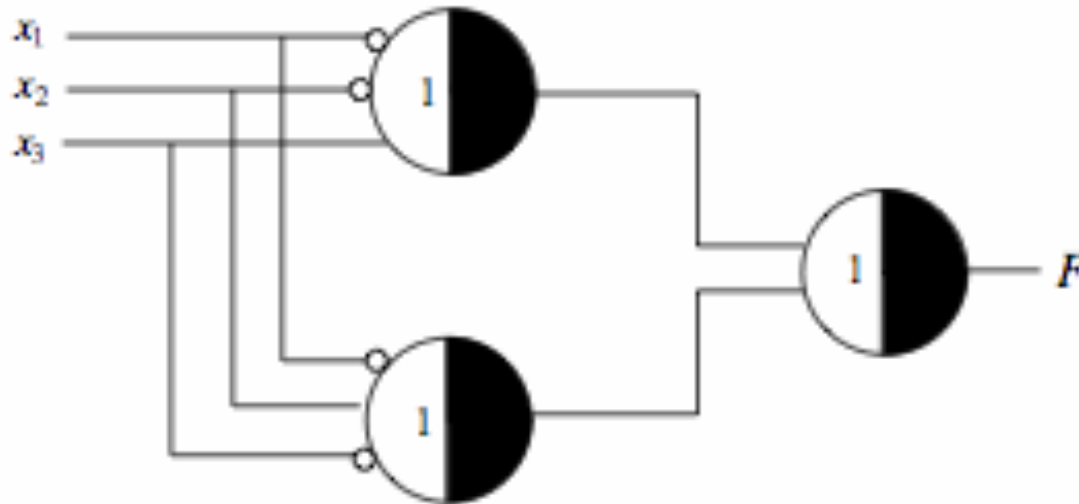
Consider only the rows where  $F$  is +ve.

$$F = (\text{not } x_1 \text{ and not } x_2 \text{ and } x_3) \text{ or } (\text{not } x_1 \text{ and } x_2 \text{ and not } x_3)$$

$x_1$	$x_2$	$x_3$	$F$
0	0	1	1
0	1	0	1
All other	combinations		0

# McCulloch-Pitts: General Logical function

Using Rojas notation/diagram





# General Logical Function

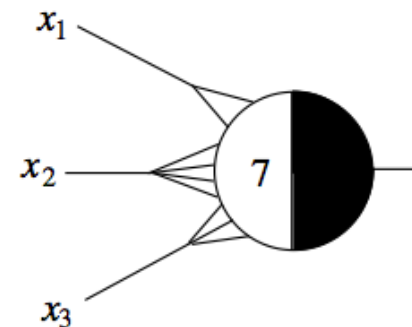
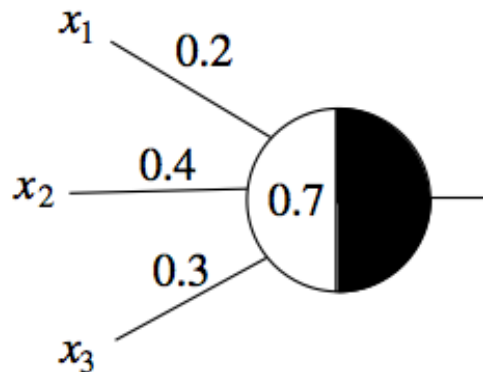
- Proposition: Any logical function  $F: \{0, 1\}^n \rightarrow \{0, 1\}$  can be computed using a McCulloch-Pitts network of two layers.

# Equivalent Networks

- One can ask the question: Is there an advantage to using weights in McCulloch-Pitts networks or no?
- Answer: It doesn't make a difference as an example will show.

# Weighted and Unweighted Networks

- Consider the network with weights on left.
- This unit computes
$$0.2x_1 + 0.4x_2 + 0.3x_3 \geq 0.7$$
This is equivalent to
$$2x_1 + 4x_2 + 3x_3 \geq 7$$
- Therefore, the unit on the right is equivalent.
- The network on right has multiple fan-ins.



# Recurrent Networks

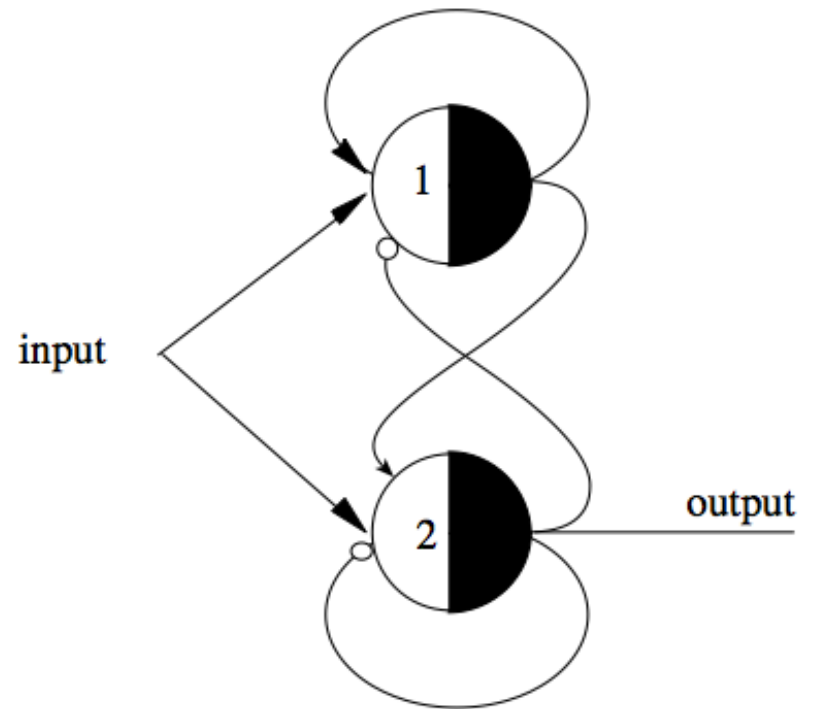
- A feed-forward McCulloch-Pitts network can be used to implement arbitrary logical functions.
- In such a case, the dimensions of the input and output data are known beforehand.
- However, when we want to perform computations with an input of variable length, such a feedforward network will not work.
- An example: Adding two binary numbers fed bit by bit into a network, which in turn produces bits of the result one after another.
- We need recurrent network, in which partial computations are recycled back to the network.

# Recurrent Network

- McCulloch-Pitts networks can be used in recurrent networks by introducing a temporal factor to the computation.
- We assume that the computation of the activation of each unit consumes a time unit.
- If the input arrives at time  $t$ , the output is produced at time  $t+1$ .

# Recurrent Network

- Here is an example of a recurrent network that can be built from McCulloch-Pitts units.



# Applications of Threshold Logic

- Threshold units can be used in an application where we want to reduce the execution time of a logic operation to possibly just two layers of computational delay without using a huge number of units.
- It has been shown that parity and majority functions, for example, are very difficult functions to implement in logic, without using exponentially growing number of conventional logic gates.
- The majority  $k$  out of  $n$  can be implemented with a single threshold McCulloch-Pitts unit.

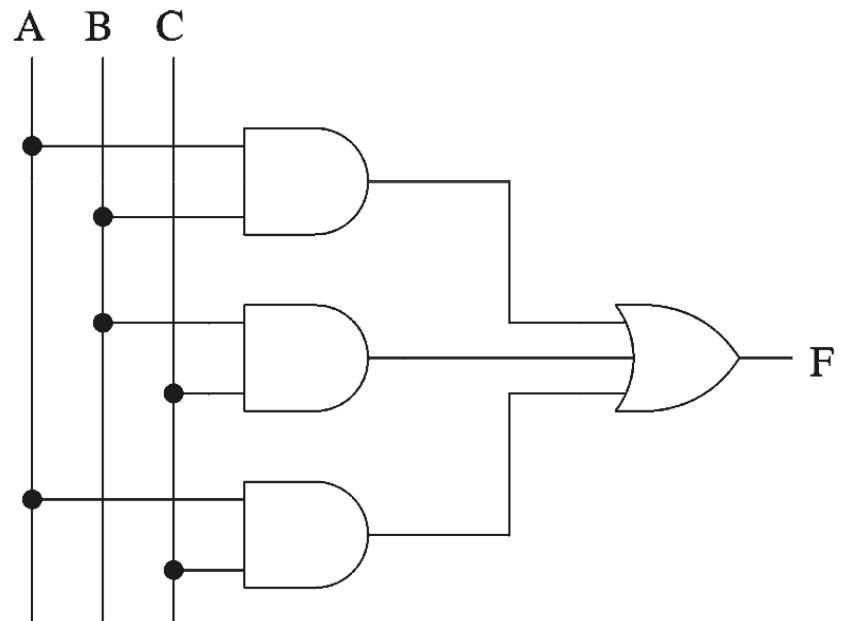
# 3-input Majority Function in Boolean Logic

3-input majority function

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

■ Logical expression form

$$F = A B + B C + A C$$



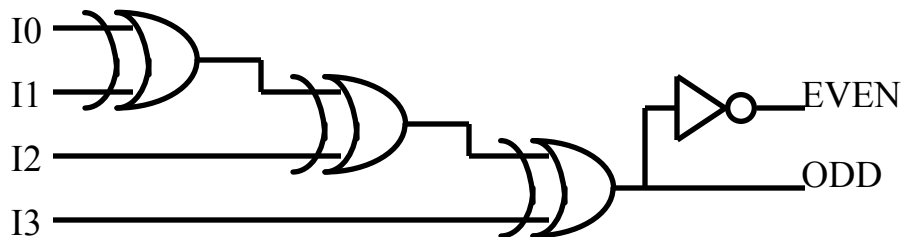


# Parity Function/Circuit in Boolean Logic

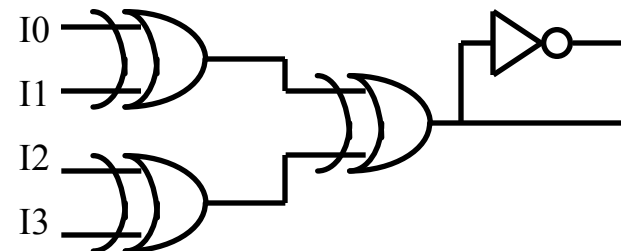
Odd Parity Circuit : The output is 1 if odd number of inputs are 1

Even Parity Circuit : The output is 1 if even number of inputs are 1

Example : 4-bit Parity Circuit



Daisy-Chain Structure



Tree structure

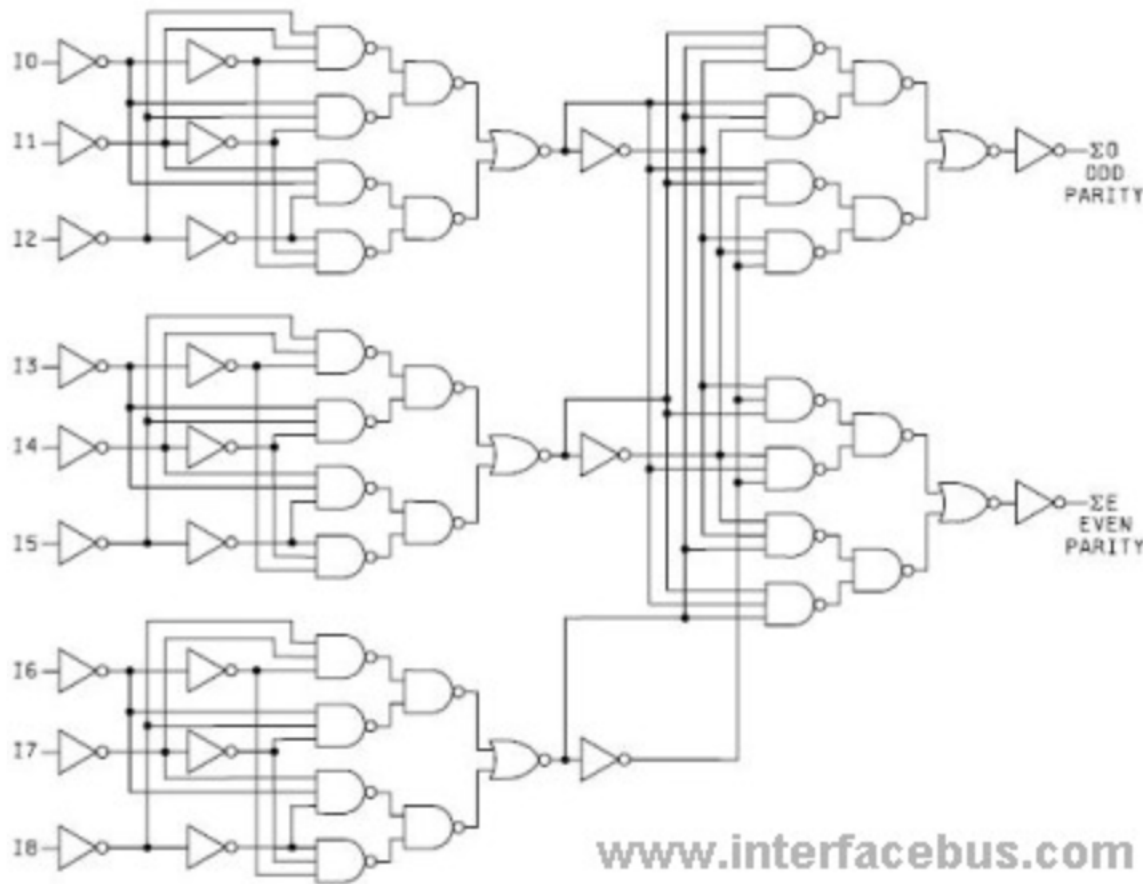
Input : 1101

Odd Parity output : 1

Even Parity output : 0

The gates used are XOR gates:  $X \oplus Y = X' \cdot Y + X \cdot Y'$

# 9-bit Parity Function/Circuit in Boolean Logic



**Parity Checker Chip**

It's not straightforward to draw parity circuits with McCulloch-Pitts units, but it's not exponential.

Try to come up with 2-bit or 3-bit parity circuits using McCulloch-Pitts units.

# Threshold Logics and Fault-tolerance

Threshold logic have been used to build fault-tolerant circuits.

Below, we see a unit, with three inputs  $x_1$ ,  $x_2$  and  $x_3$ .

Each input is transmitted twice by input lines that are independent of each other.

This gate will produce the correct result even if one of the lines is transmitted with an error.

