University of British Columbia
Electrical and Computer Engineering
Digital Design and Microcomputers CPEN312

L03: Binary Logic and Gate Implementations.

Dr. Jesús Calviño-Fraga. P.Eng.
Department of Electrical and Computer Engineering, UBC
Office: KAIS 3024
E-mail: jesusc@ece.ubc.ca
Phone: (604)-827-5387

January 4/9, 2019

Objectives

- Truth tables
- OR, AND, NOT gates.
- NOR, NAND, XOR, XNOR gates.
- Boolean expressions.
- Voltages and bits.
- The electronics of logic gates.

Binary Logic and Operations

- Binary variables are based on two states:
  - On/Off, Yes/No, True/False, etc.
  - When used in electronics the two states are represented as voltages.
  - For convenience we call one state 1 and the other 0.
- The basic operations we can perform with binary variables are:
  - AND, represented with a dot (.)
  - OR, represented with a plus (+)
  - NOT, represented with a prime (‘) or a bar (¯)

Truth Table

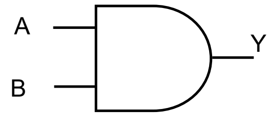
- A truth table enumerates all possible combinations of inputs and the output of a logic operation. For example, for a two input AND gate:

A, B are inputs. There could be more than 2!

Y is the output.

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

AND Gate



$$Y = A \cdot B$$

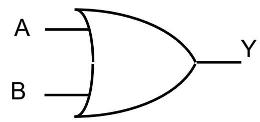
$$Y = AB$$

For convenience we can skip the dot!

A	B	Y=A.B
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

OR Gate

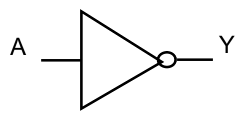


$$Y = A + B$$

A	B	Y=A+B
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

NOT Gate



A	Y=A’
0	1
1	0

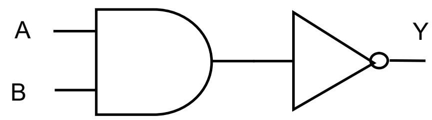
$$Y = \overline{A}$$

$$Y = A'$$

Logic Gates

- They can have more than one input, but they have only one output.
- The output of a gate can be the input to another gate.
- Two or more outputs can not be connected together.

AND and NOT



A	B	Y=(A.B)'
0	0	1
0	1	1
1	0	1
1	1	0

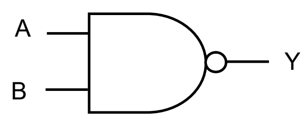
Truth Table

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

This operation is referred as a NOT AND, or NAND

NAND Gate

For convenience, we collapsed the not gate into a circle at the output!

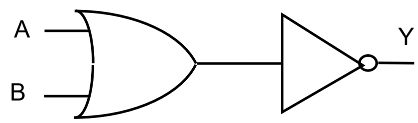


A	B	Y=(A.B)'
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

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

OR and NOT



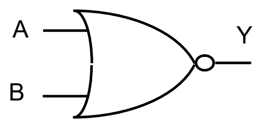
$$Y = \overline{A + B}$$

A	B	Y=(A+B)’
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table

This operation is referred as a NOT OR, or NOR

NOR Gate

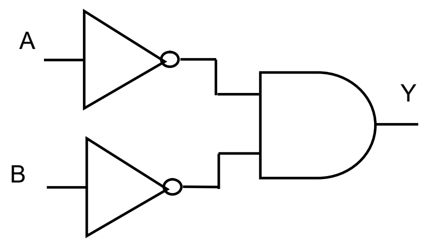


$$Y = \overline{A + B}$$

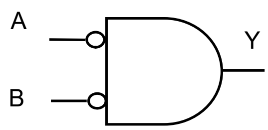
A	B	Y=(A+B)’
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table

Circles at the Inputs:



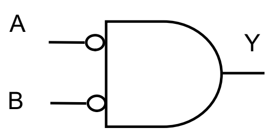
Can be redrawn as:



Y=A'.B'

Example 1

- Obtain the truth table of the gate below.

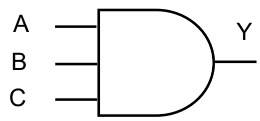


Y=A'.B'

A	**B**	Y=A'.B'
0	0	1
0	1	0
1	0	0
1	1	0

Example 2

- Obtain the truth table of the gate below.

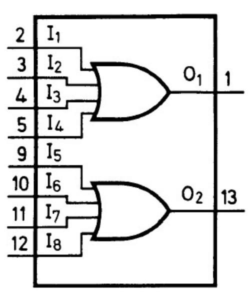


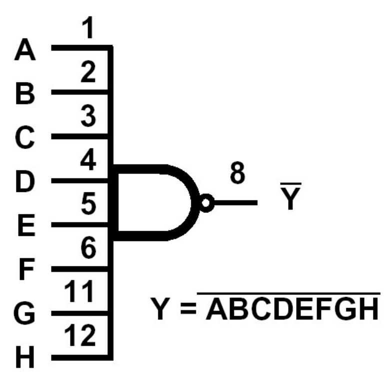
Y=A.B.C

Three input AND gate.

A	B	C	Y=A.B.C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

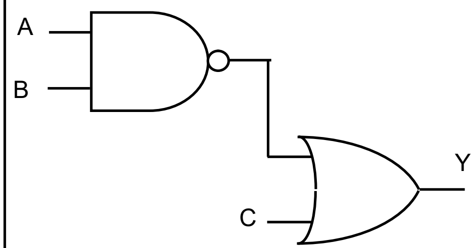
Gates with more than two inputs





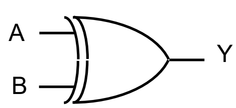
Example 3

- Obtain the truth table of the circuit below.



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

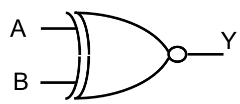
XOR GATE



$$Y = A \oplus B$$

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

XNOR GATE

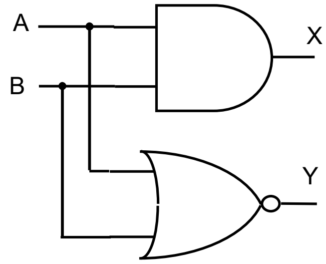


$$Y = \overline{A \oplus B}$$

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

Multiple Outputs

- If the logic circuit has more than one output the truth table can include all of them:

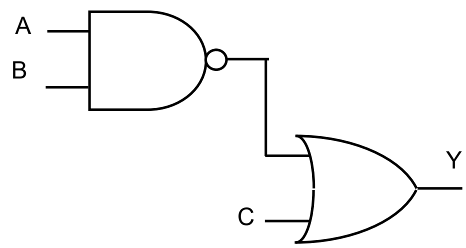


A	B	X	Y
0	0	0	1
0	1	0	0
1	0	0	0
1	1	1	0

Truth Table

Equations to/from Gates

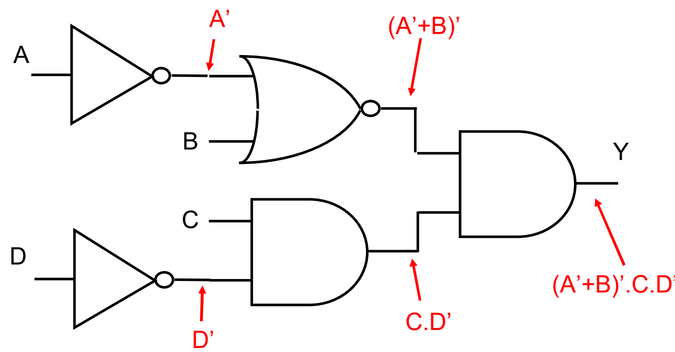
- Often we need to convert logic equations to gates and vice versa. For example:



Y=(A.B)'+C

Example 4

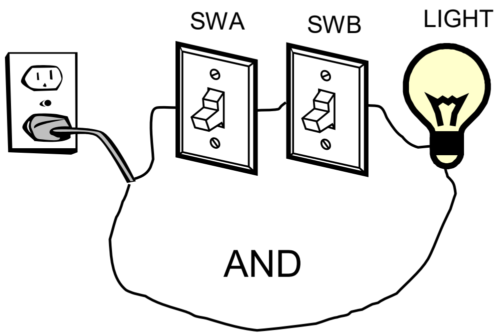
- Draw the gates circuit for the logic equation Y=(A'+B)'.C.D'. Use one or two input gates only.



One possible solution!

Implementation of Logic Gates

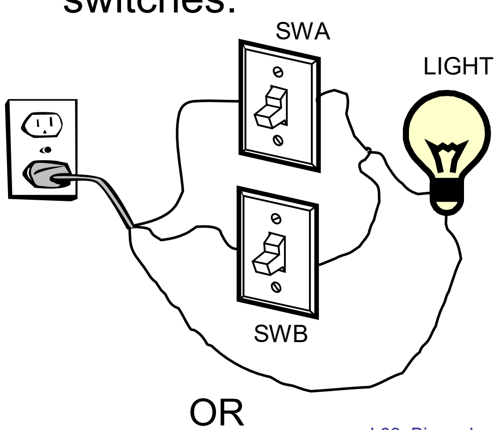
- AND & OR gates can be built using switches:



If SWA is on AND SWB is on, then LIGHT is on

Implementation of Logic Gates

- AND & OR gates can be built using switches:



If SWA is on OR SWB is on, then LIGHT is on

For the NOT gate we need a relay.

Some history

- Relay: invented in 1835 by Joseph Henry (1797–1878).
- Binary Logic: developed in 1847 by George Boole (1815–1864).
- Mechanical Computer: the Analytical Engine proposed in 1837 by Charles Babbage (1791–1871)
- It took humanity 90 years to put the above three developments together!

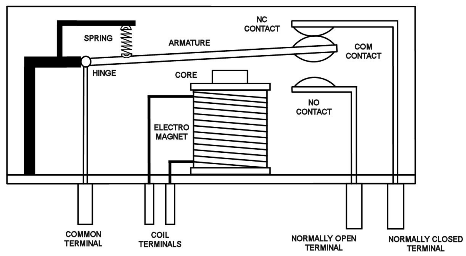
Claude Shannon

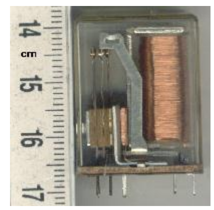
- Credited with founding both digital computer and digital circuit design theory in 1937.
- Master thesis "***A Symbolic Analysis of Relay and Switching Circuits"*** showed how to implement logic circuits with relays.

Claude Shannon (1916–2001)

Relay

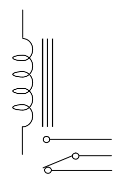
http://www.glolab.com/relays/NC%20RELAY.jpg





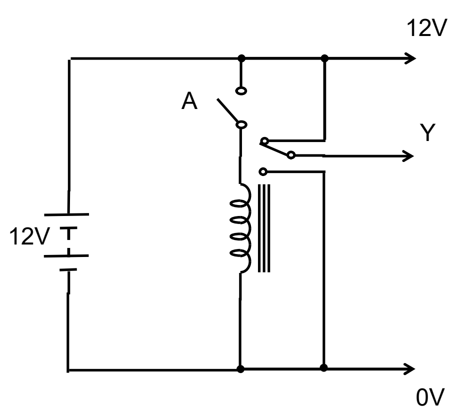
http://upload.wikimedia.org/wikipedia/commons/3/39/Relay2.jpg

Electro-magnet moves a switch contact



Electronic Symbol

Relay NOT Gate



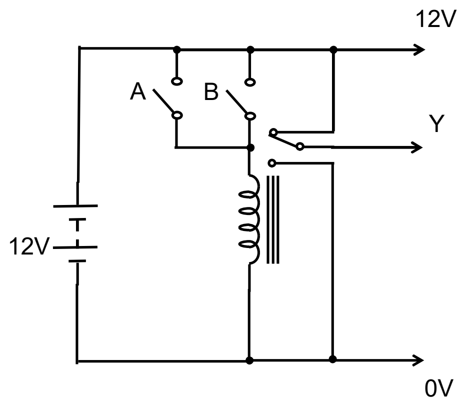
Logic ONE is 12V

Logic Zero is 0V

If switch A is open (logic zero), output Y is 12V (logic one)

If switch A is closed (logic one), output Y is 0V (logic zero)

Relay NOR Gate

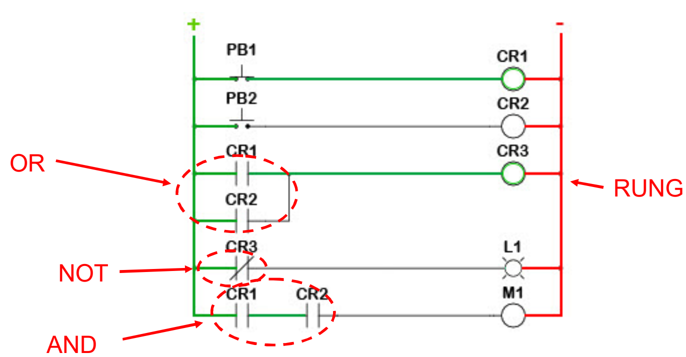


A	B	Y=(A+B)’
0V	0V	12V
0V	12V	0V
12V	0V	0V
12V	12V	0V

Truth Table

Believe it or not, computers had been built with relay gates!

Logic representation using relays: Ladder Logic



Ladder Logic is extensively used in industry to “program” Programmable Logic Controllers (PLCs).

Ladder Logic in Shannon's work

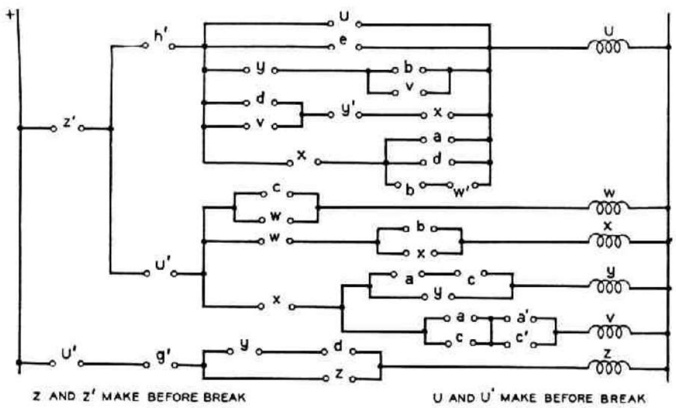
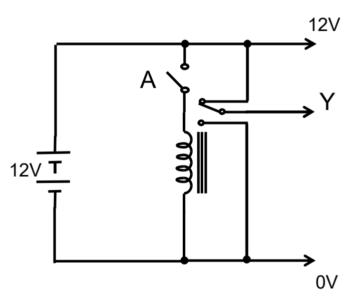


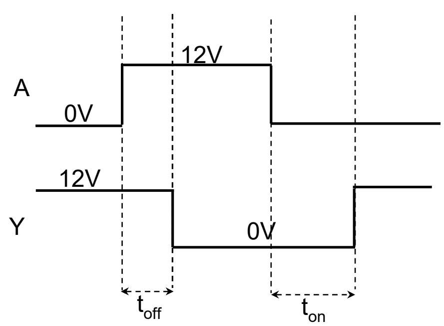
Figure 34. Combination-lock circuit

Make-before-break: In a switching device, a configuration in which the new connection path is established before the previous contacts are opened. This prevents the switched path from ever seeing an open circuit.

Propagation Delay

- Logic circuits do not change their outputs immediately after the inputs change. There will be always a propagation delay!

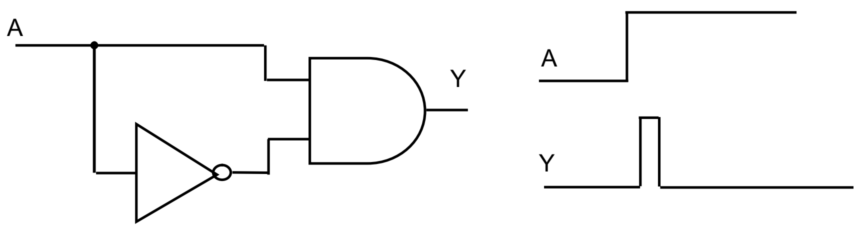




For relays t_{on}/t_{off} is in the tens of mili-seconds

Propagation Delay

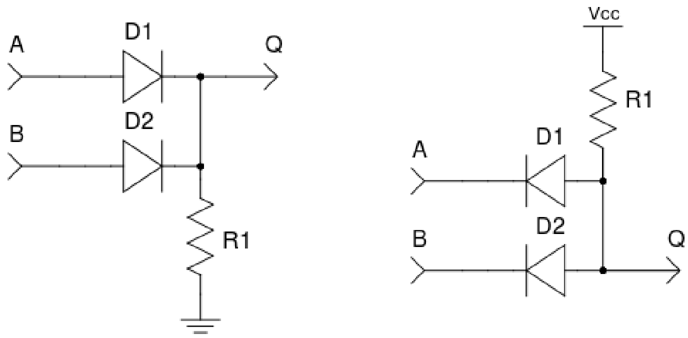
- Propagation delays can affect the output of a logic circuit in unexpected ways:



Diode Logic

- Diodes behave somehow like switches. They let DC current flow in one direction only.
- Similarly to switch logic, only AND & OR gates can be implemented.
- To implement NOT, NAND, or NOR gates a transistor is needed.

Diode Logic



OR Gate

AND Gate

http://en.wikipedia.org/wiki/Diode_logic

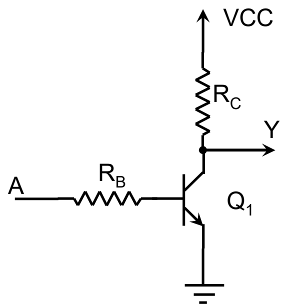
Transistor Logic

- With transistors, which behave similarly to relays, we can implement a NOT gate as well as any other gate we want!
- There are two types of transistor we can use:
  - Bipolar Junction Transistors or BJTs.
  - Metal Oxide Semiconductor Field Effect Transistors or MOSFETs.

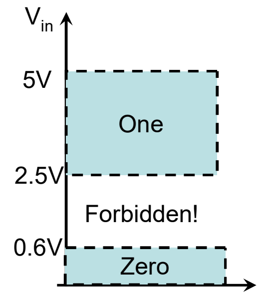
BJT NOT Gate

- The basic NOT gate with BJTs looks like this:

Say VCC=5V, Logic 1: V_{in}>2.5V, Logic 0: V_{in}<0.6V, 0.6>V_{in}<2.5 is forbidden!

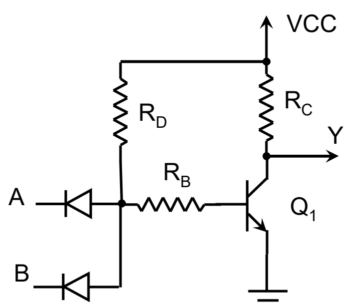


A	Q_1	Y
0	off	1
1	on	0



Diode-Transistor-Logic (DTL) NAND Gate

- The basic NAND gate with a BJT and diodes:

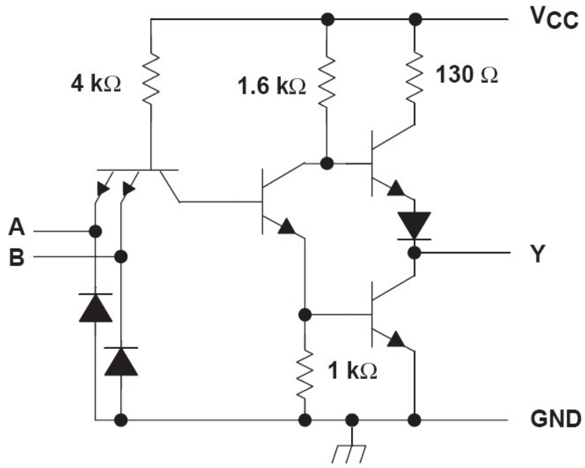


This is called Diode-Transistor-Logic or DTL. It was the technology used in the Apollo spacecraft. Look for "Apollo guidance computer".

A	B	D_1	D_2	Q_1	Y
0	0	on	on	off	1
0	1	on	off	off	1
1	0	off	on	off	1
1	1	off	off	on	0

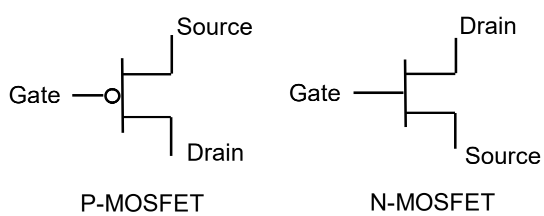
Truth Table

Transistor-Transistor Logic (TTL)

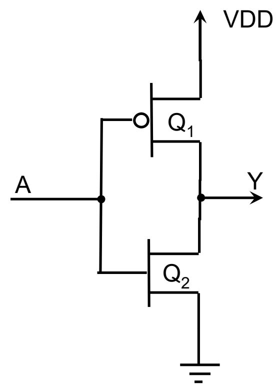


MOSFET Logic

- MOSFET logic, in particular Complementary MOSFETs or CMOS, is the most widely used kind of logic. It is small, fast, cheap, and reliable.
- CMOS uses two types of MOSFETs. The N-MOSFET is turned on (closes) with logic one at the gate pin; the P-MOSFET is turned on (closes) with logic zero at the gate pin.



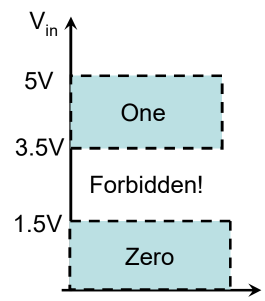
CMOS NOT Gate



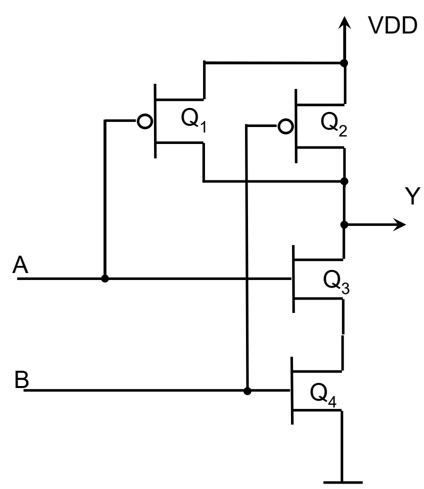
A	Q_1	Q_2	Y
0	on	off	1
1	off	on	0

Truth Table

Say VDD=5V, Logic 1: V_{in}>3.5V, Logic 0: V_{in}<1.5V, 1.5>V_{in}<3.5 is forbidden!



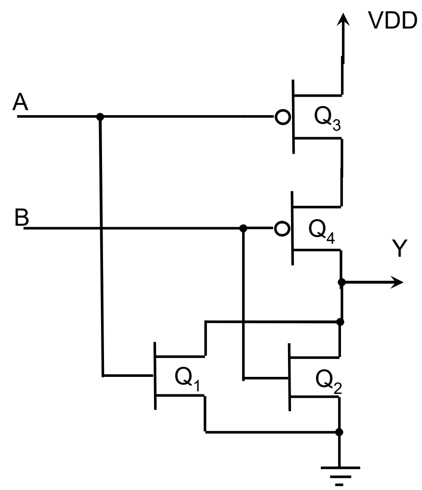
CMOS NAND Gate



A	B	Q_1	Q_2	Q_3	Q_4	Y
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

Truth Table

CMOS NOR Gate



A	B	Q_1	Q_2	Q_3	Q_4	Y
0	0	off	off	on	on	1
0	1	off	on	on	off	0
1	0	on	off	off	on	0
1	1	on	on	off	off	0

Truth Table

Exercises

1. Obtain the truth table and draw the digital circuit for the equation Y=A'.B'+A.B. Have you seen that truth table before?
2. Design a three input NAND gate using a BJT, diodes, and resistors.
3. Design a three input NOR gate using MOSFETS.
4. Design a 2 input OR gate using MOSFETS.
5. Design a 2 input AND gate using MOSFETS.
6. Draw the digital circuit for the 2-input, 4-output truth table in the next slide.

Exercises

Inputs		Outputs			
A	B	Y_1	Y_2	Y_3	Y_4
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Truth Table