Semi-conductor and Logic Gates

Lecture 21

Hyung-Sin Kim

SNU Graduate School of Data Science

Contents

Review

- **Bits, Data Types, and Operations**

Review

- Exact value vs. Presence/Absence
- Bit, multiple bits, and data types
- Logical variables, bit vectors, and operations
- Unsigned integers, signed integers, and operations
- Floating point, ASCII, and Hexagonal

Today's Goals

- We will explore computer hardware a little bit
- You will understand why semiconductor is the root of computers
- You will understand why computers only understands 0s and 1s
- You will understand how computer hardware perform logical operations, arithmetic operations, and store values
- You will understand that hardware can do everything software can do. It is just hard (fixed), instead of soft (flexible)

Semi-conductor

Computing Bootcamp

What is Semi-conductor

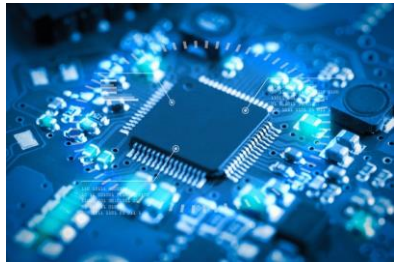
- Something that is necessary for Korean economy

- Something that Samsung Electronics and SK Hynix are doing well

SAMSUNG

SK hynix

- Some rectangular stuff that is necessary for making computers



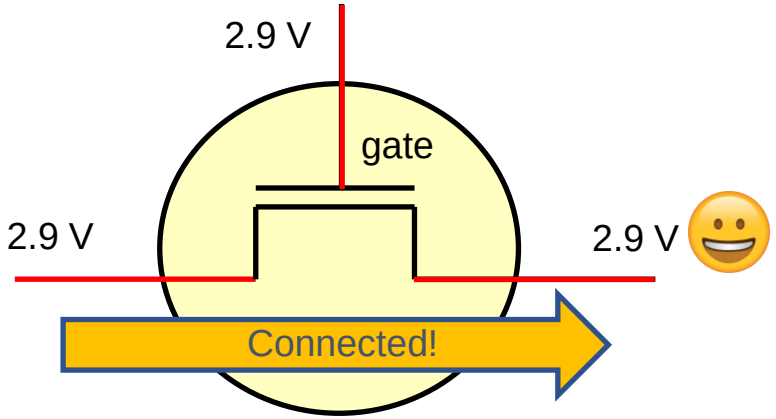
What is Semi-conductor

- Conductor: Any material that allows electric current to pass through it
  - Copper, Aluminum, Steel, etc…
- Insulator: Any material that does not allow electric current to pass through it
  - Plastic, Rubber, Glass, Wood, etc…
- **Semi-conductor:** A material that can act as a conductor or an insulator depending on conditions
- **Transistor**: A device built by using semi-conductor, which is a fundamental unit of computers

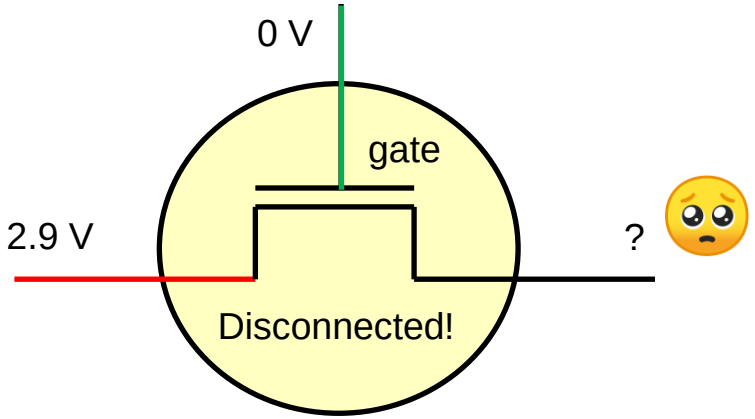
How does Transistor work?

- Left and right parts are connected or not depending on the gate status
  - n-type transistor

If gate is supplied with **2.9 V**,
left and right are connected
like a conductor (wire)



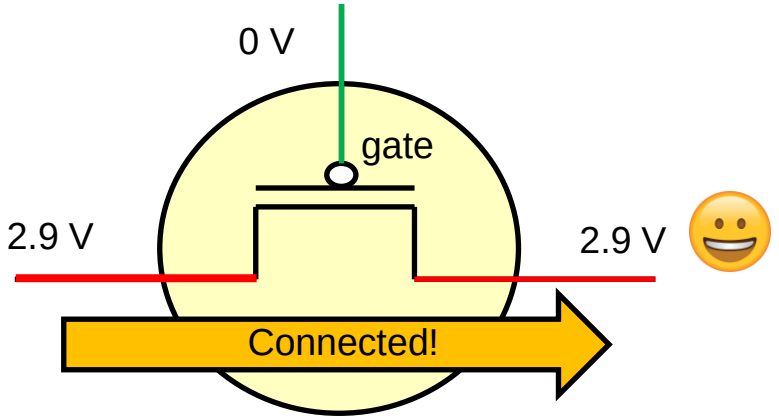
If gate is supplied with **0 V**,
left and right are disconnected
like an insulator (disconnected wire)



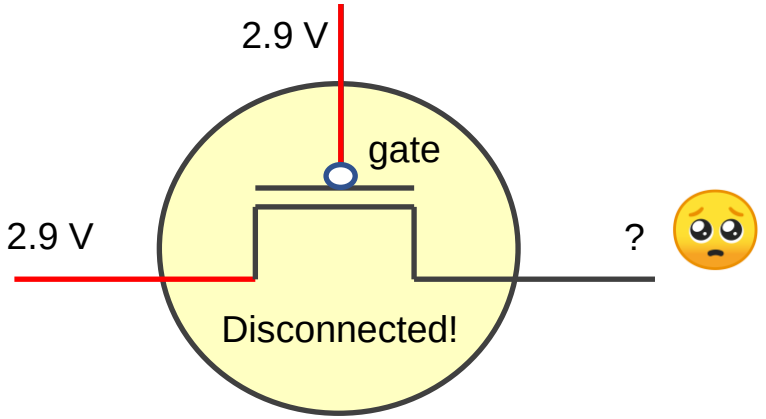
How does Transistor work?

- Left and right parts are connected or not depending on the gate status
  - p-type transistor

If gate is supplied with **0 V**,
left and right are connected
like a conductor (wire)



If gate is supplied with **2.9 V**,
left and right are disconnected
like an insulator (disconnected wire)



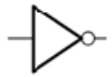
*We can do **Boolean operation** with binary numbers*
by arranging multiple transistors

Logic gates

Computing Bootcamp

Logic Gates

NOT



INPUT	OUTPUT
A	
0	1
1	0

AND

INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

OR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

NAND

INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR

INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

XNOR

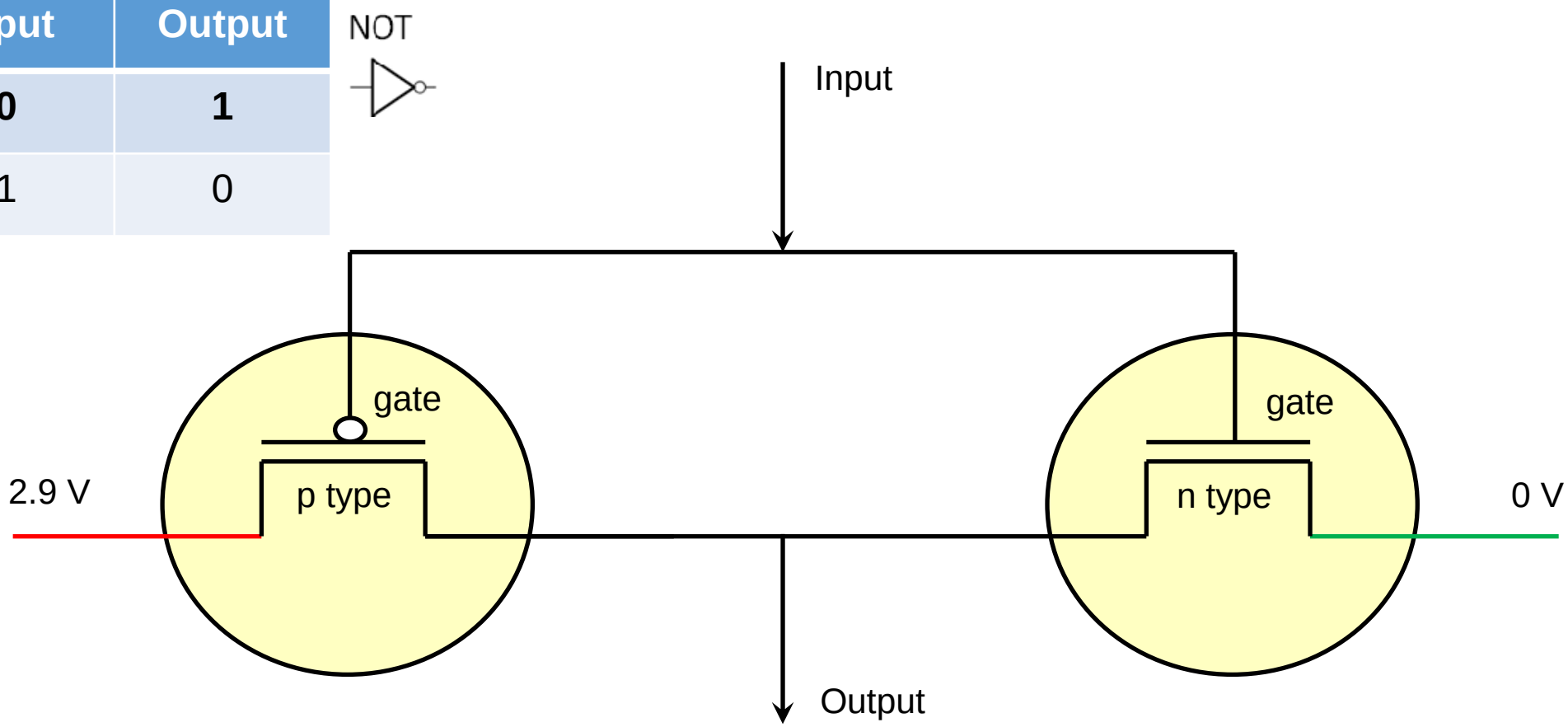
INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	1

NOT Gate (Inverter)

Input	Output
0	1
1	0

NOT Gate (Inverter) – For Input 0

Input	Output
0	**1**
1	0



NOT Gate (Inverter) – For Input 0

Input	Output
0	**1**
1	0

NOT Gate (Inverter) – For Input 0

Input	Output
0	**1**
1	0

NOT Gate (Inverter) – For Input 0

Input	Output
0	**1**
1	0

NOT Gate (Inverter) – For Input 1

Input	Output
0	1
1	**0**

NOT Gate (Inverter) – For Input 1

Input	Output
0	1
1	**0**

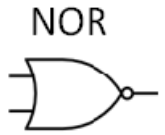
NOT Gate (Inverter) – For Input 1

Input	Output
0	1
1	**0**

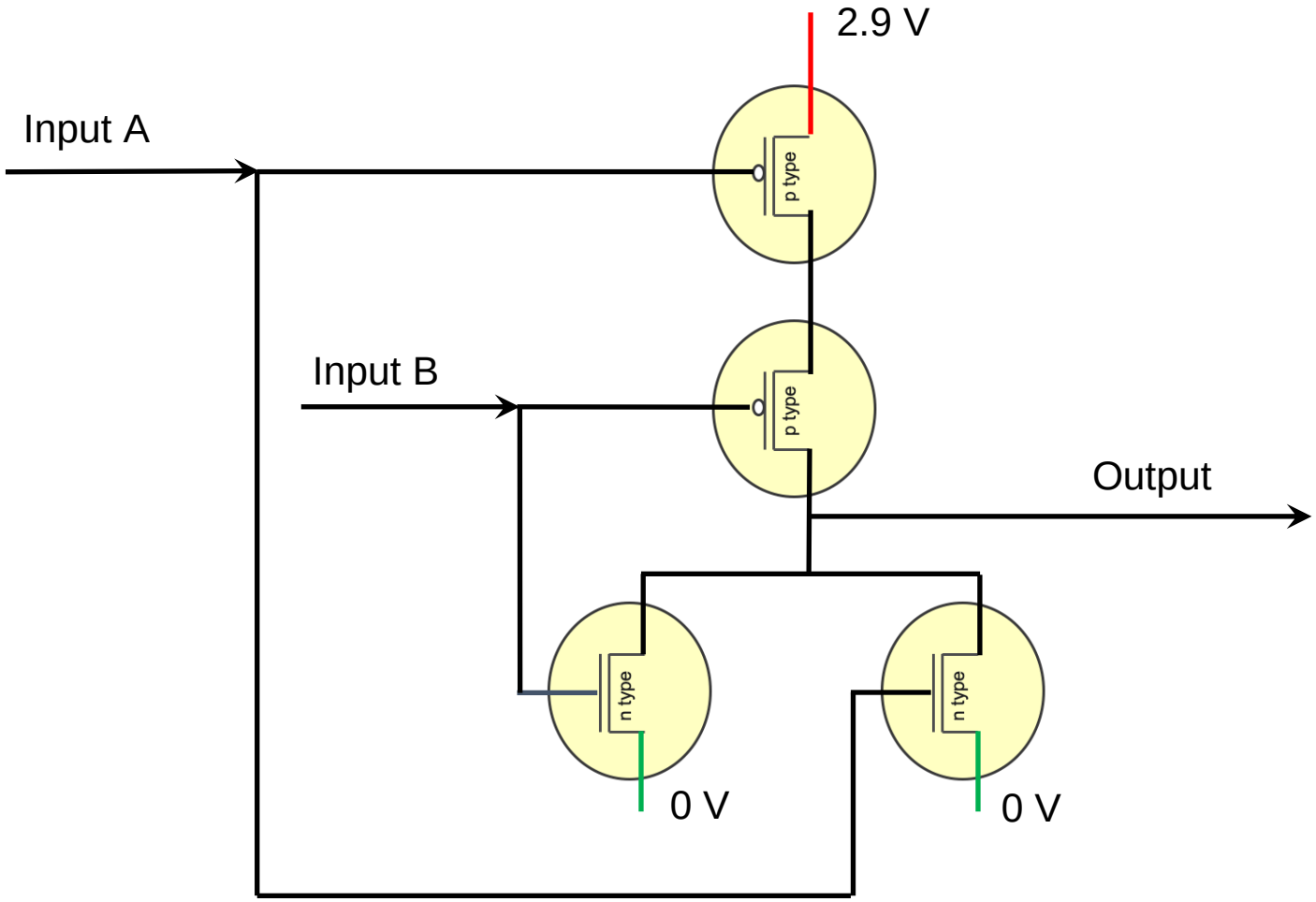
NOT Gate (Inverter) – For Input 1

Input	Output
0	1
1	**0**

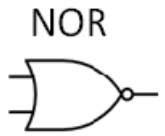
NOR Gate



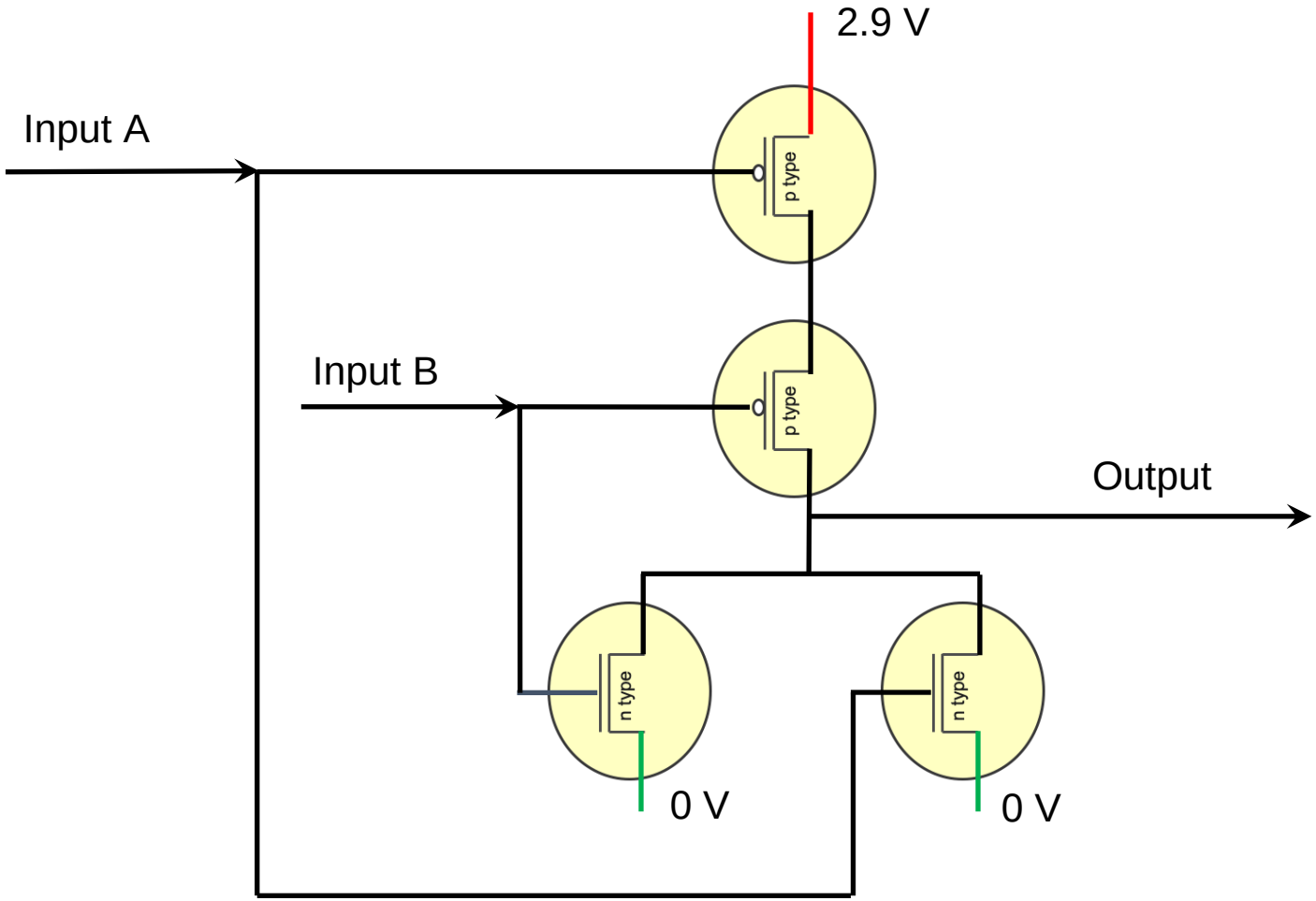
Input A	Input B	Output
0	0	1
1	0	0
0	1	0
1	1	0



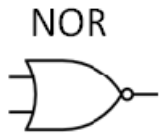
NOR Gate – For Input (0, 0)



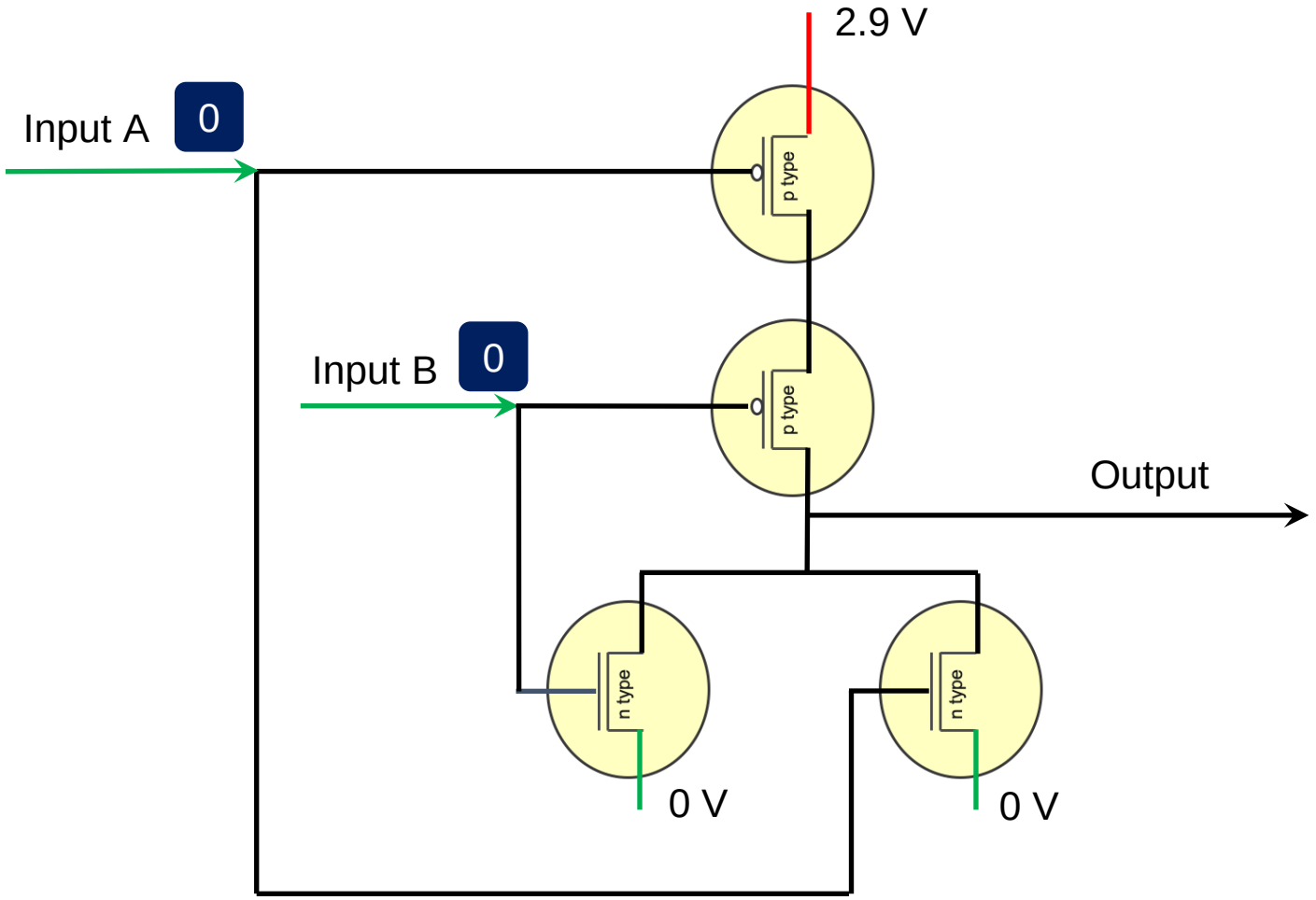
Input A	Input B	Output
0	**0**	**1**
1	0	0
0	1	0
1	1	0



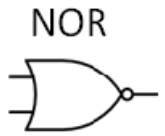
NOR Gate – For Input (0, 0)



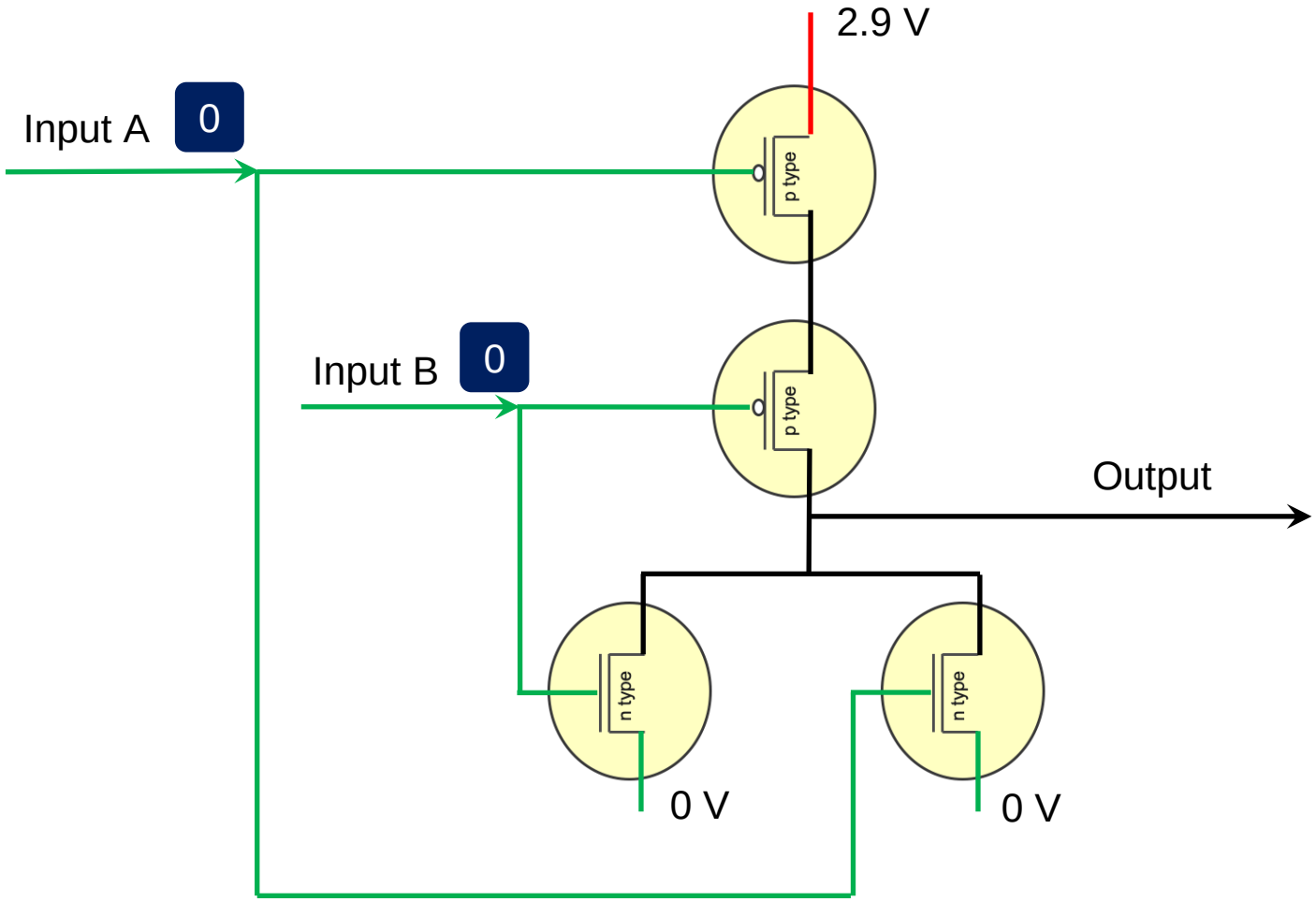
Input A	Input B	Output
0	**0**	**1**
1	0	0
0	1	0
1	1	0



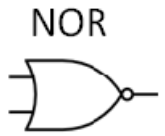
NOR Gate – For Input (0, 0)



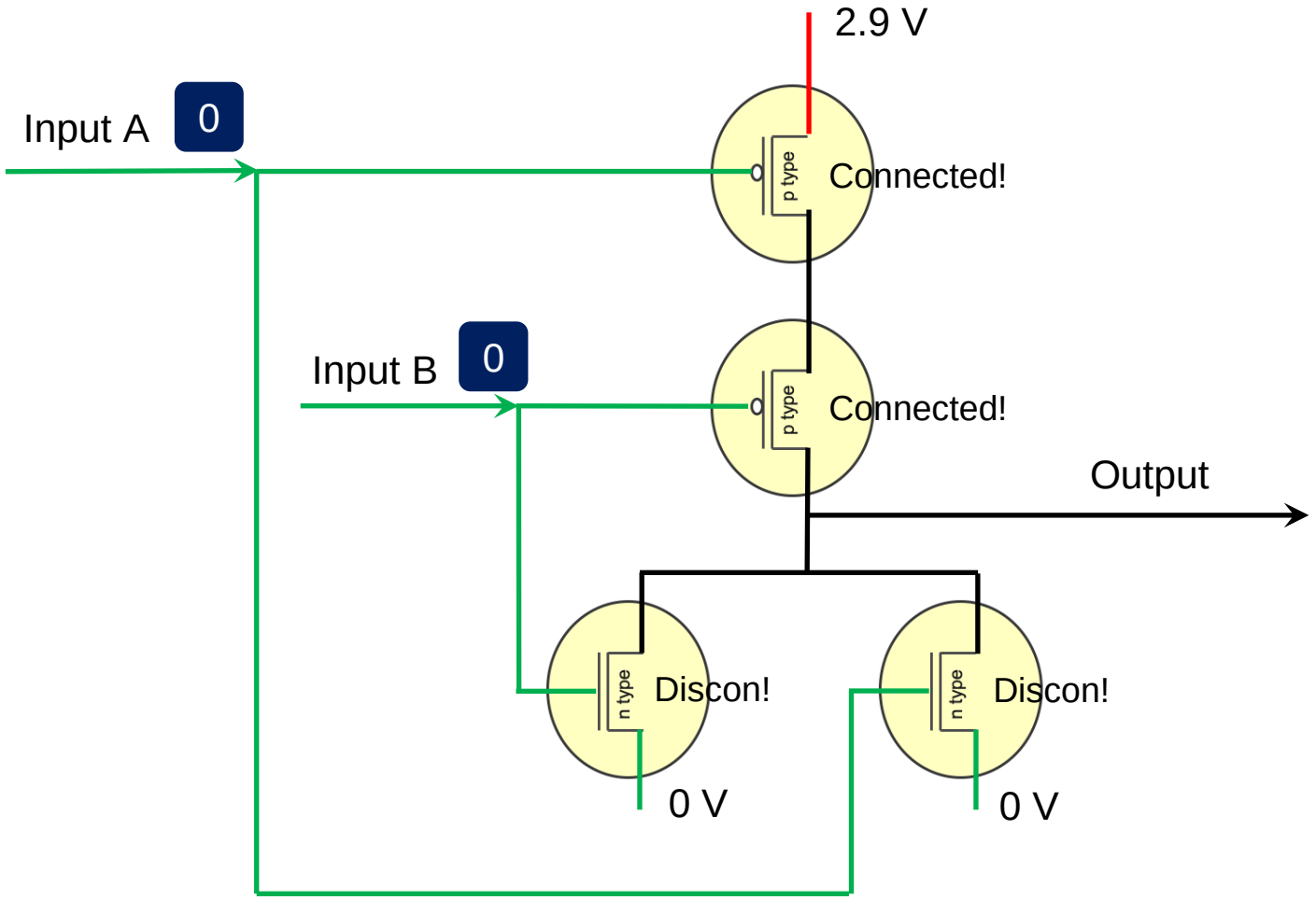
Input A	Input B	Output
0	**0**	**1**
1	0	0
0	1	0
1	1	0



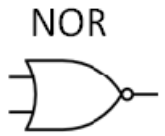
NOR Gate – For Input (0, 0)



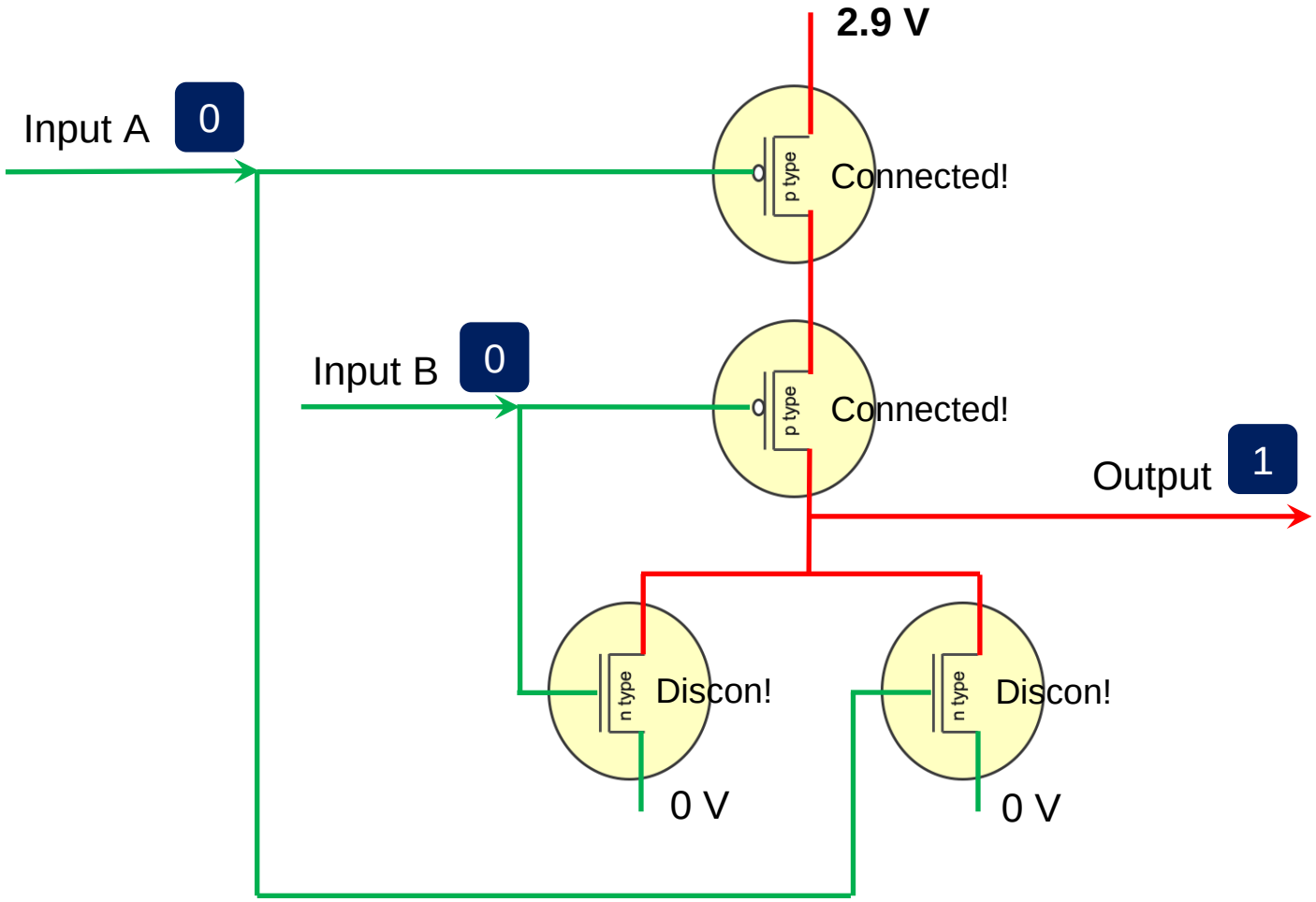
Input A	Input B	Output
0	**0**	**1**
1	0	0
0	1	0
1	1	0



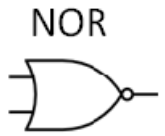
NOR Gate – For Input (0, 0)



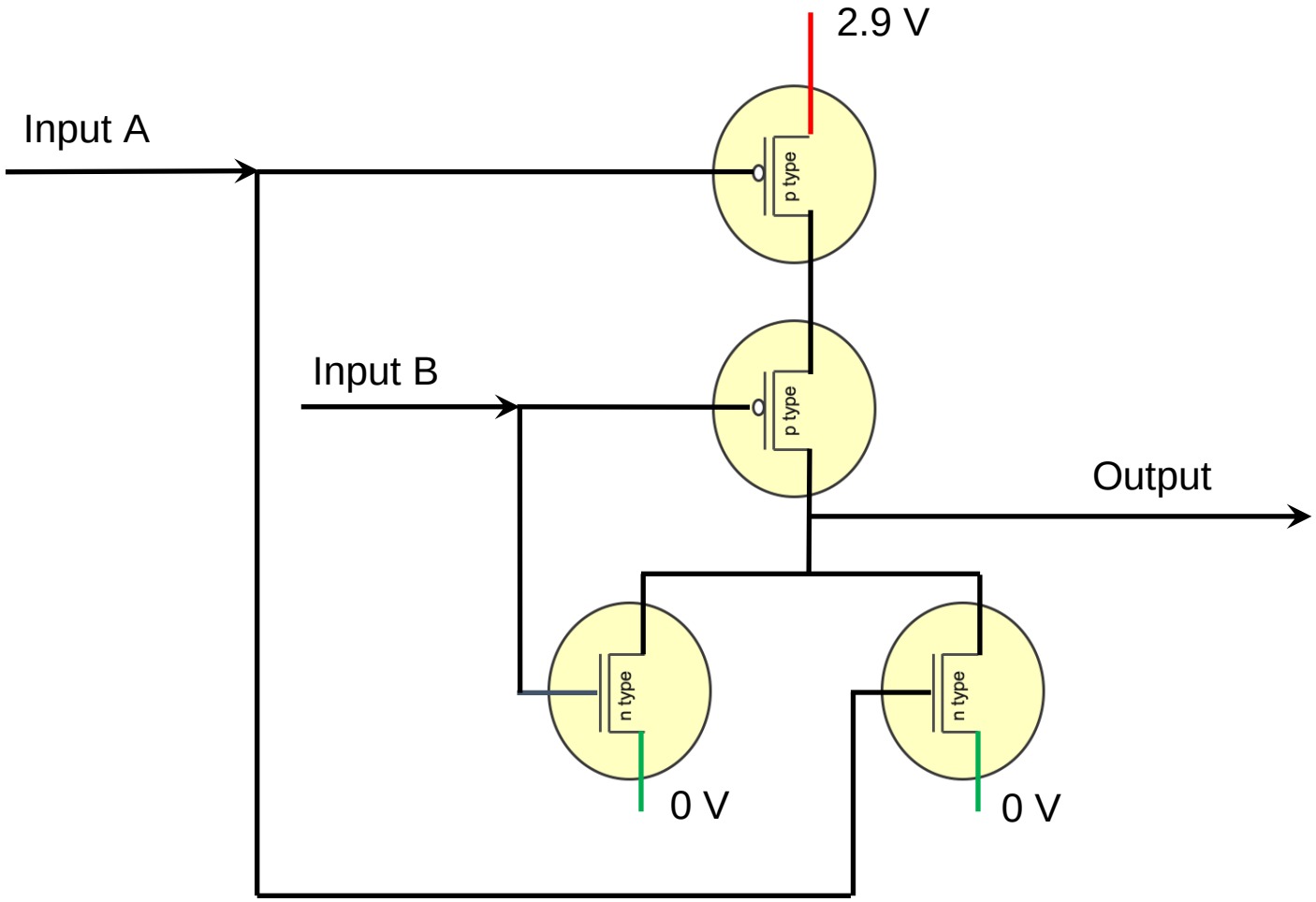
Input A	Input B	Output
0	**0**	**1**
1	0	0
0	1	0
1	1	0



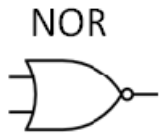
NOR Gate – For Input (1, 0)



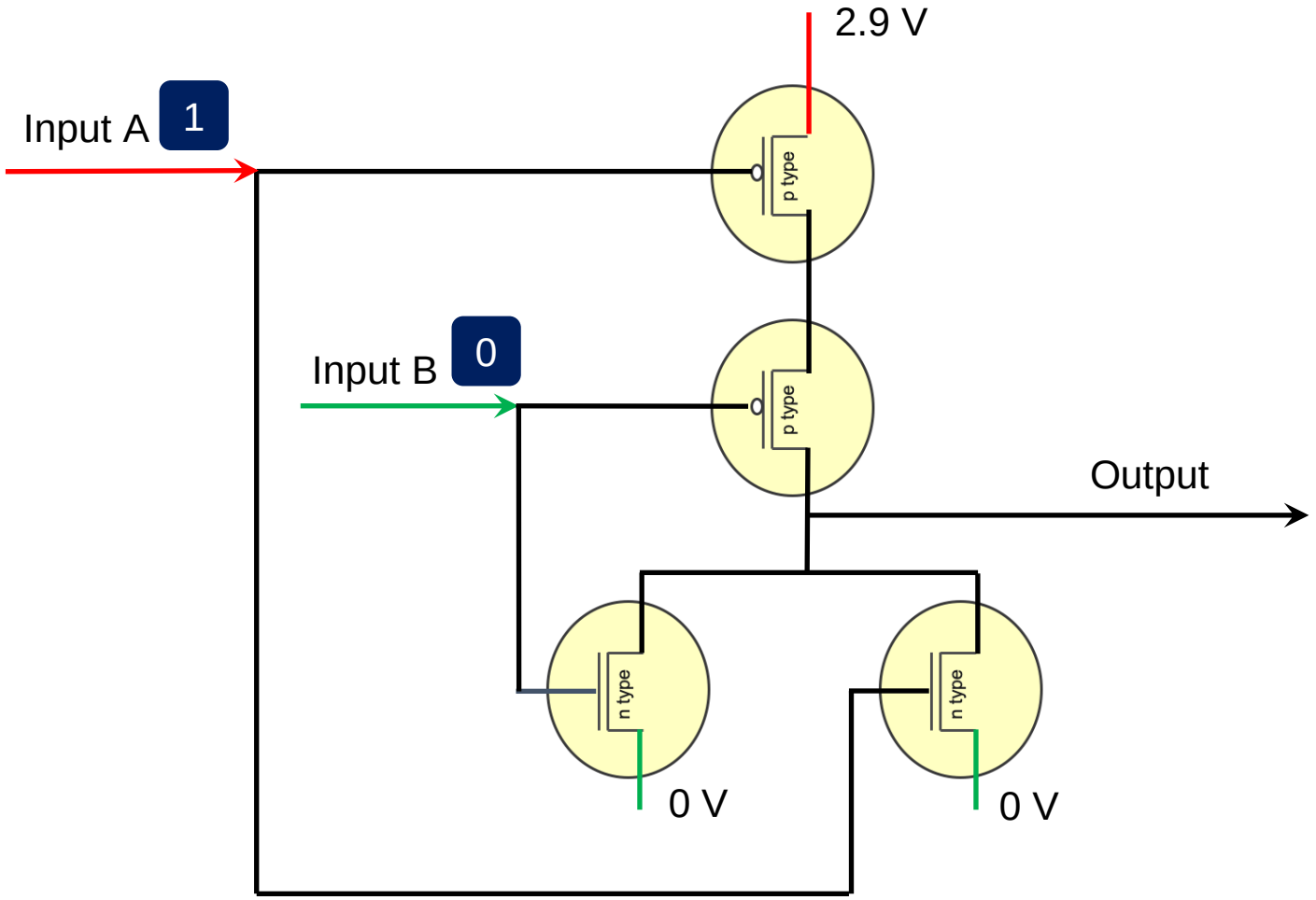
Input A	Input B	Output
0	0	1
1	**0**	**0**
0	1	0
1	1	0



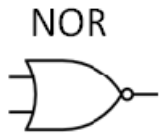
NOR Gate – For Input (1, 0)



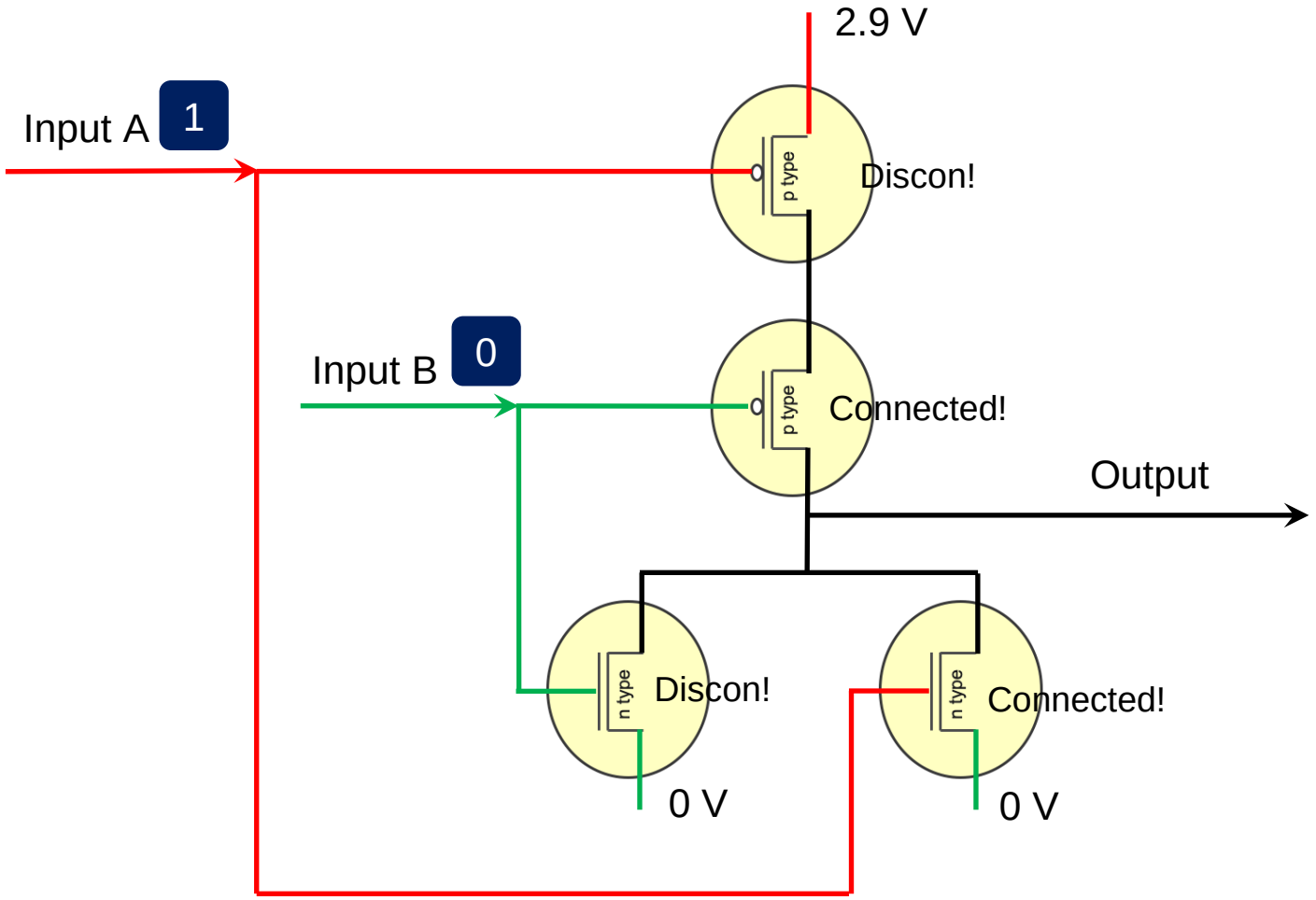
Input A	Input B	Output
0	0	1
1	**0**	**0**
0	1	0
1	1	0



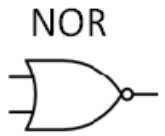
NOR Gate – For Input (1, 0)



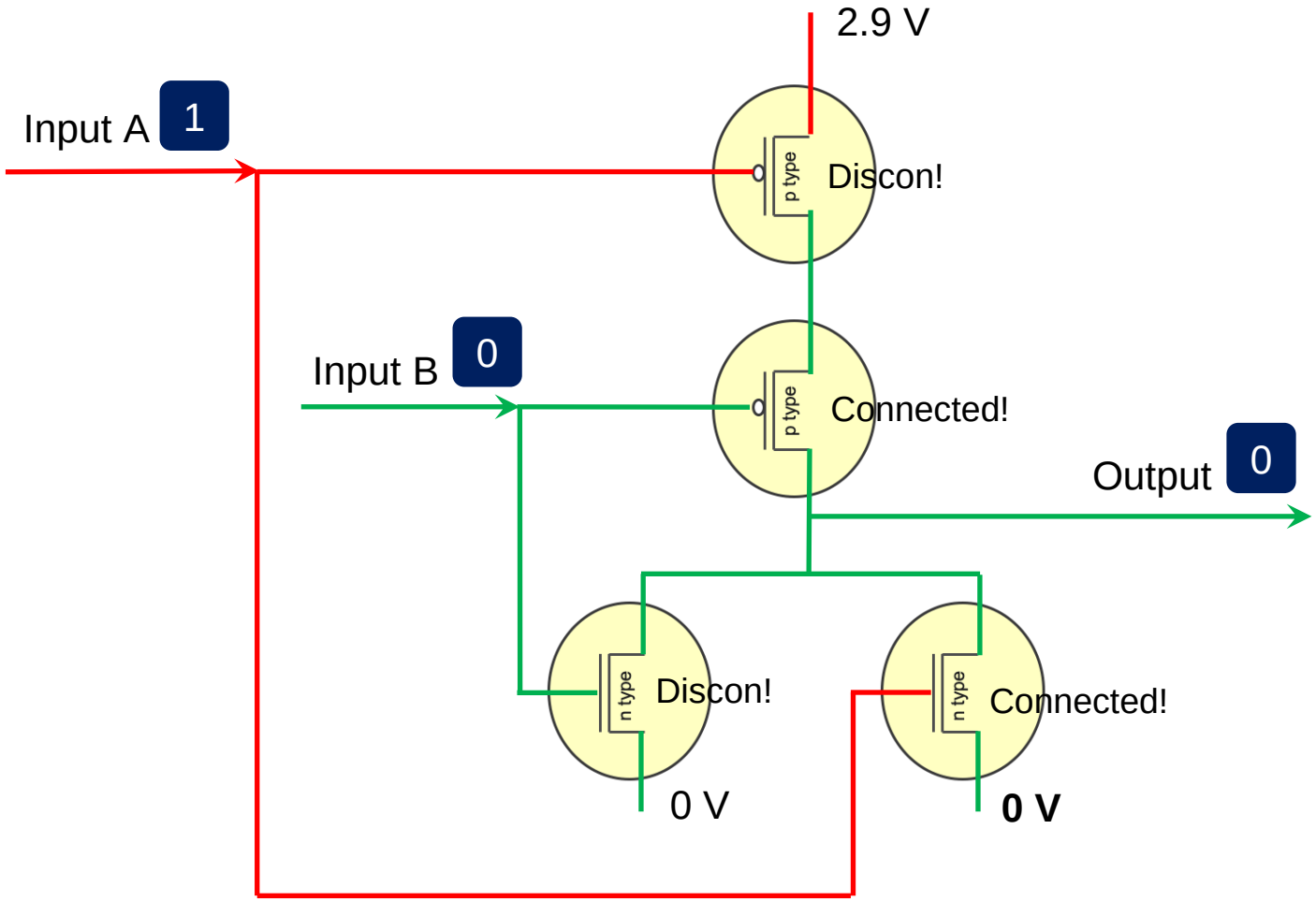
Input A	Input B	Output
0	0	1
1	**0**	**0**
0	1	0
1	1	0



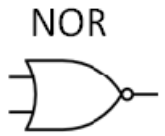
NOR Gate – For Input (1, 0)



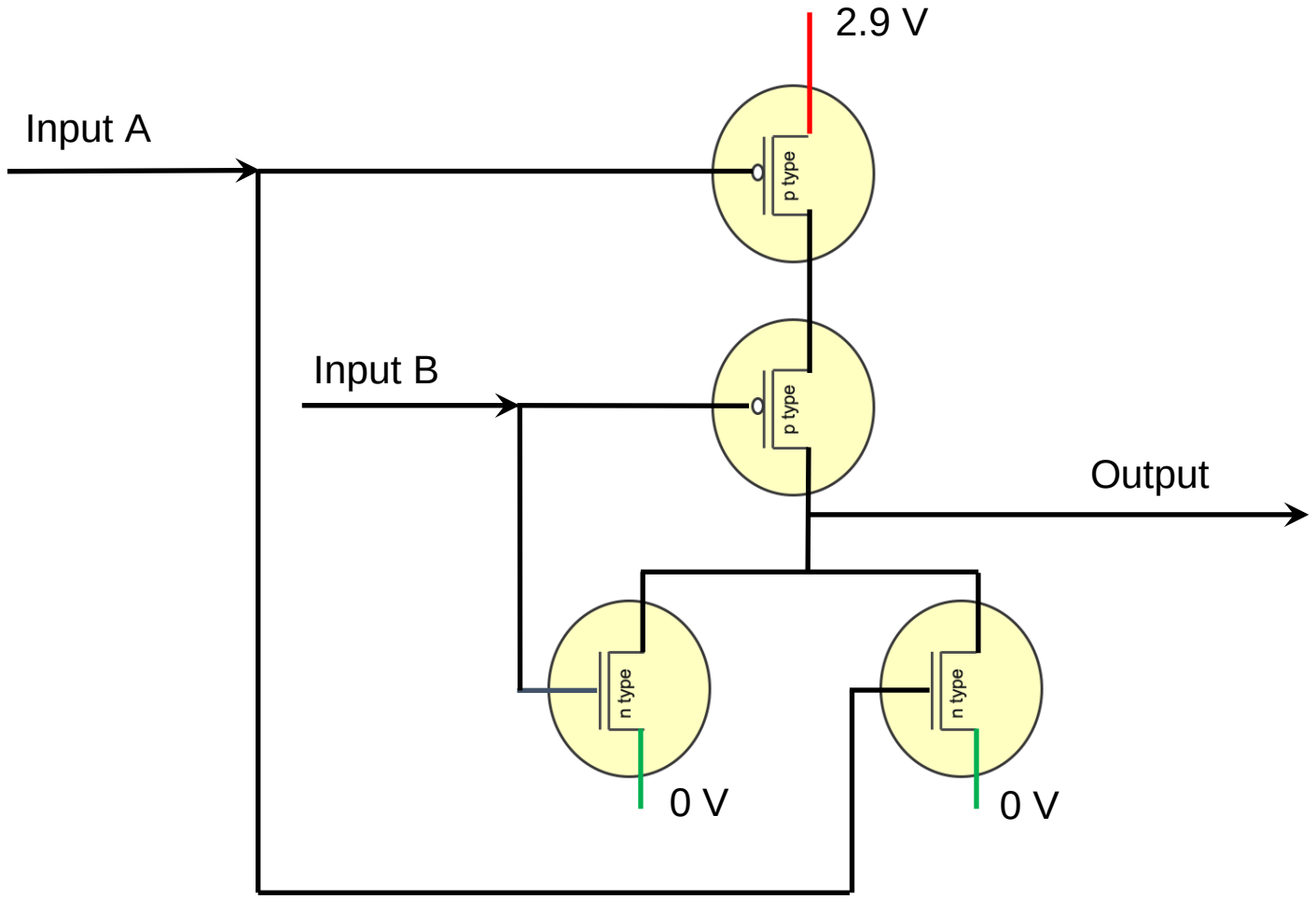
Input A	Input B	Output
0	0	1
1	**0**	**0**
0	1	0
1	1	0



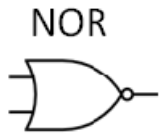
NOR Gate – For Input (0, 1)



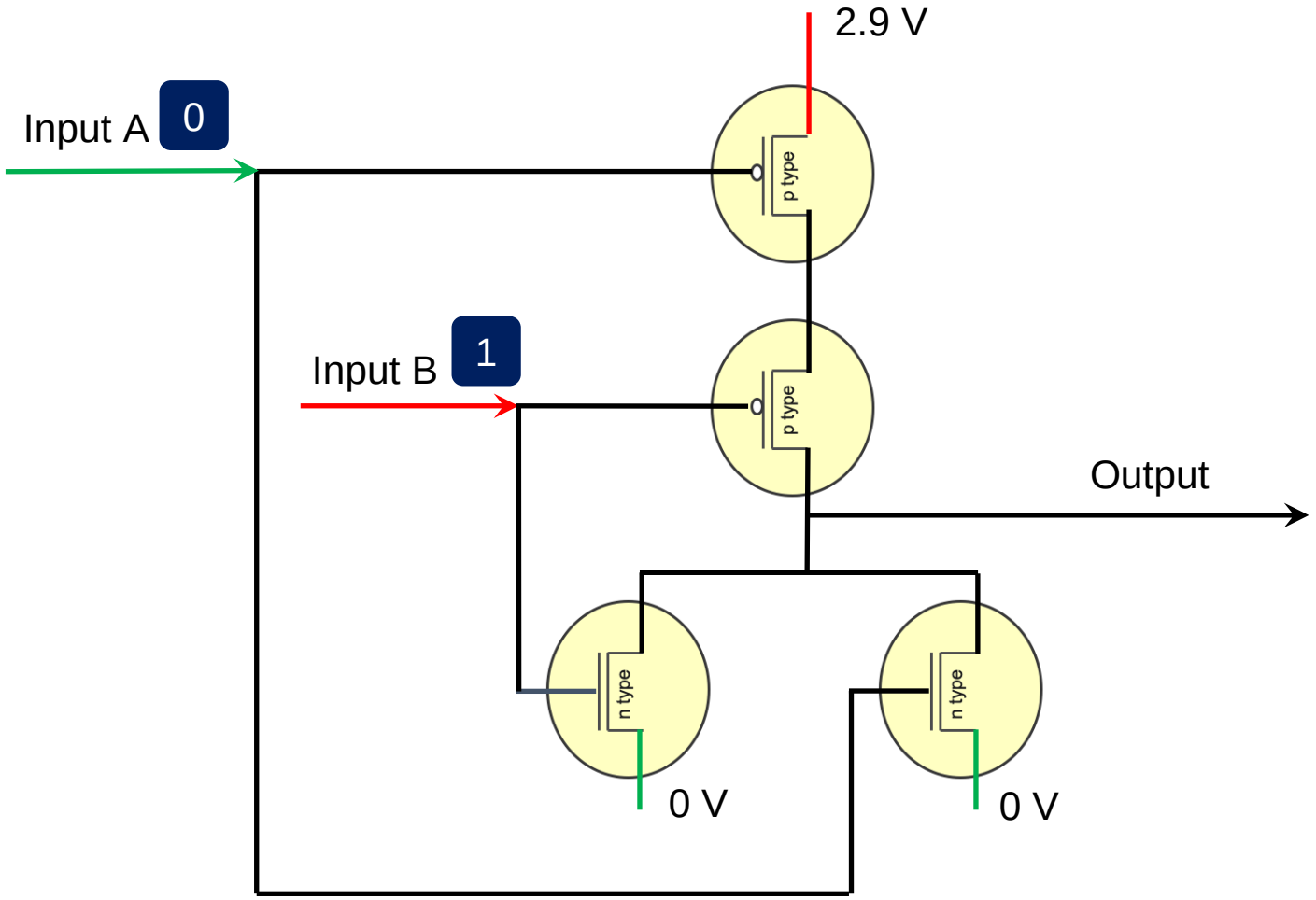
Input A	Input B	Output
0	0	1
1	0	0
0	**1**	**0**
1	1	0



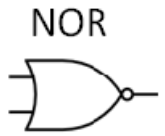
NOR Gate – For Input (0, 1)



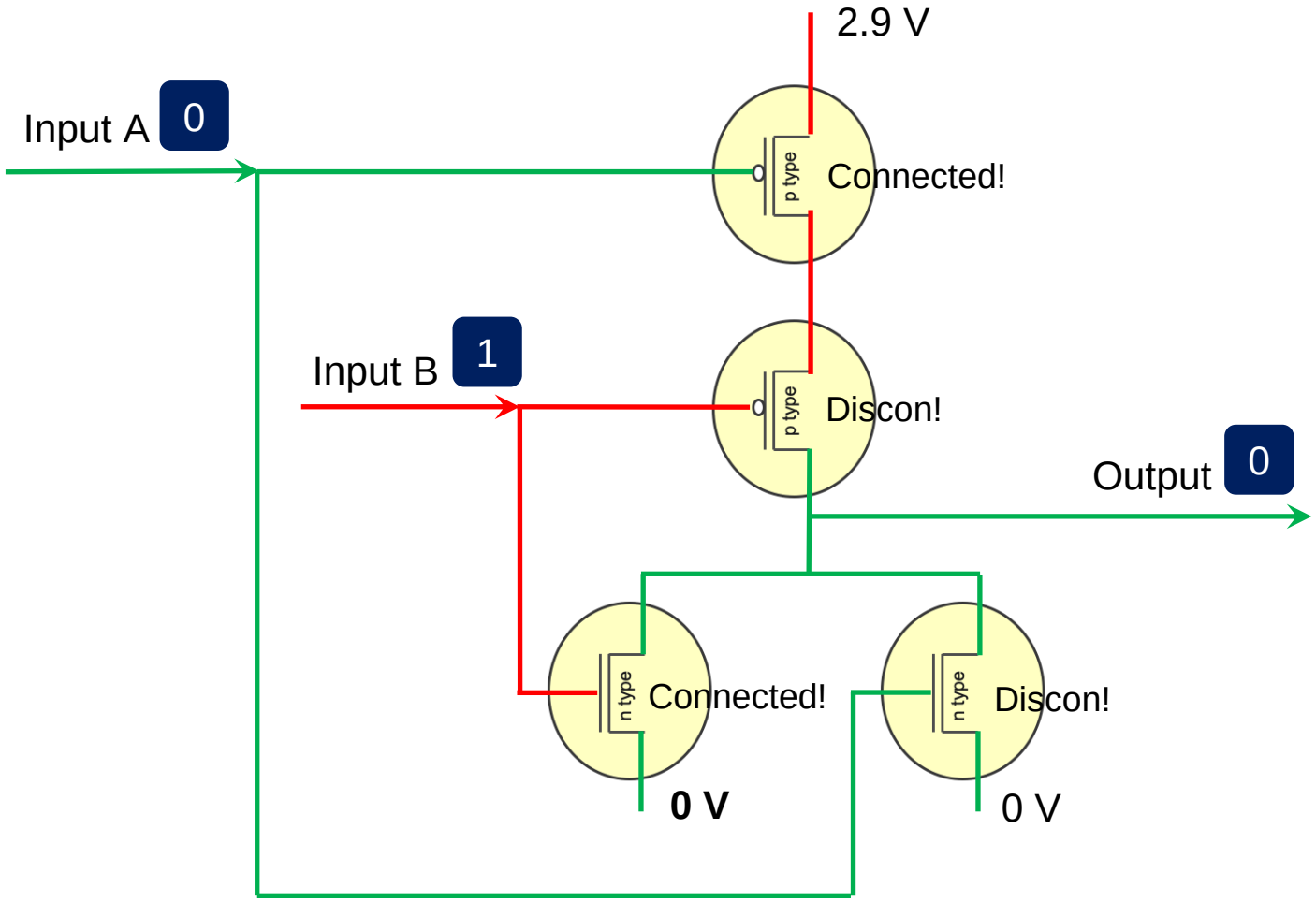
Input A	Input B	Output
0	0	1
1	0	0
0	**1**	**0**
1	1	0



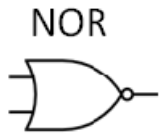
NOR Gate – For Input (0, 1)



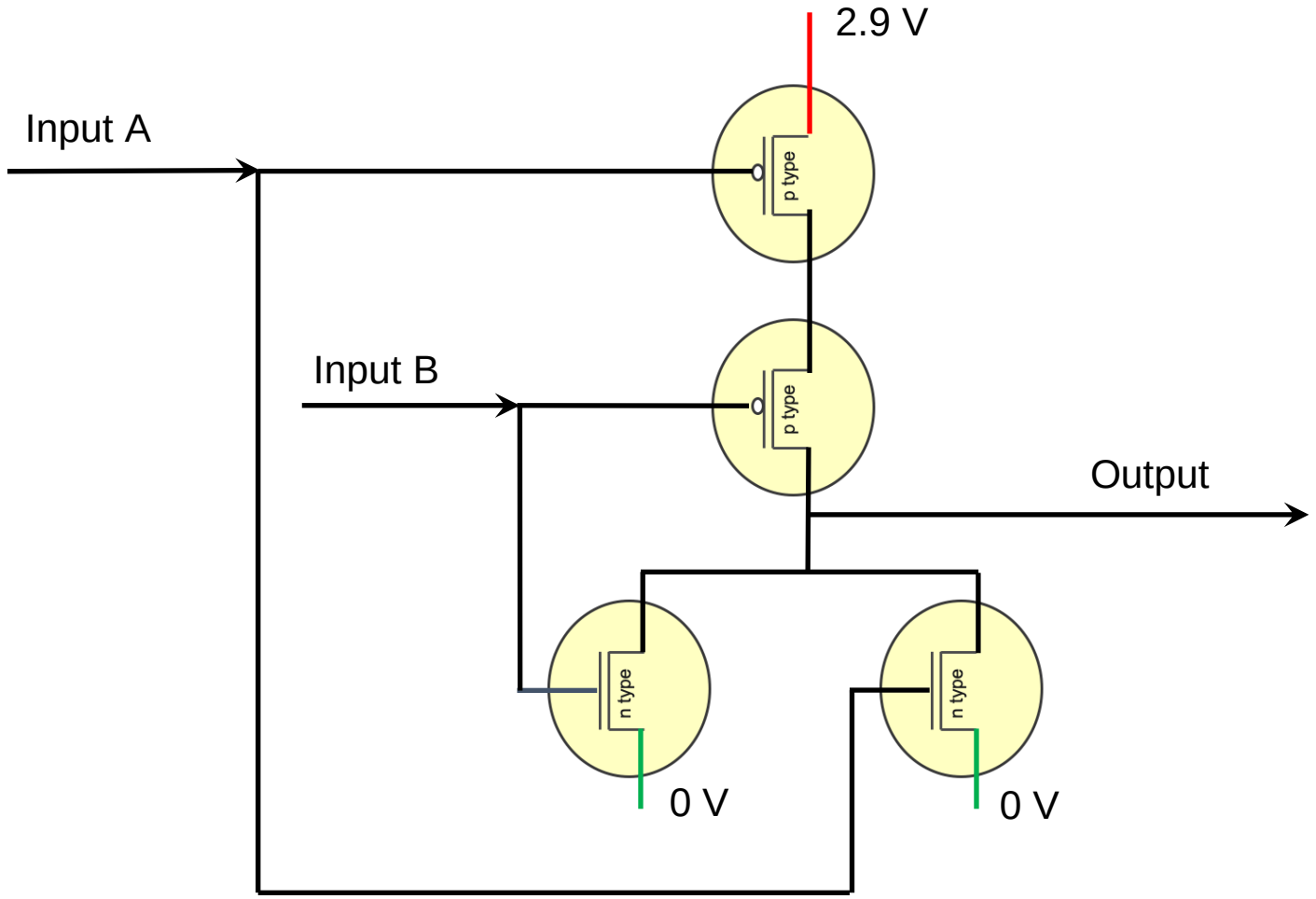
Input A	Input B	Output
0	0	1
1	0	0
0	**1**	**0**
1	1	0



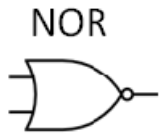
NOR Gate – For Input (1, 1)



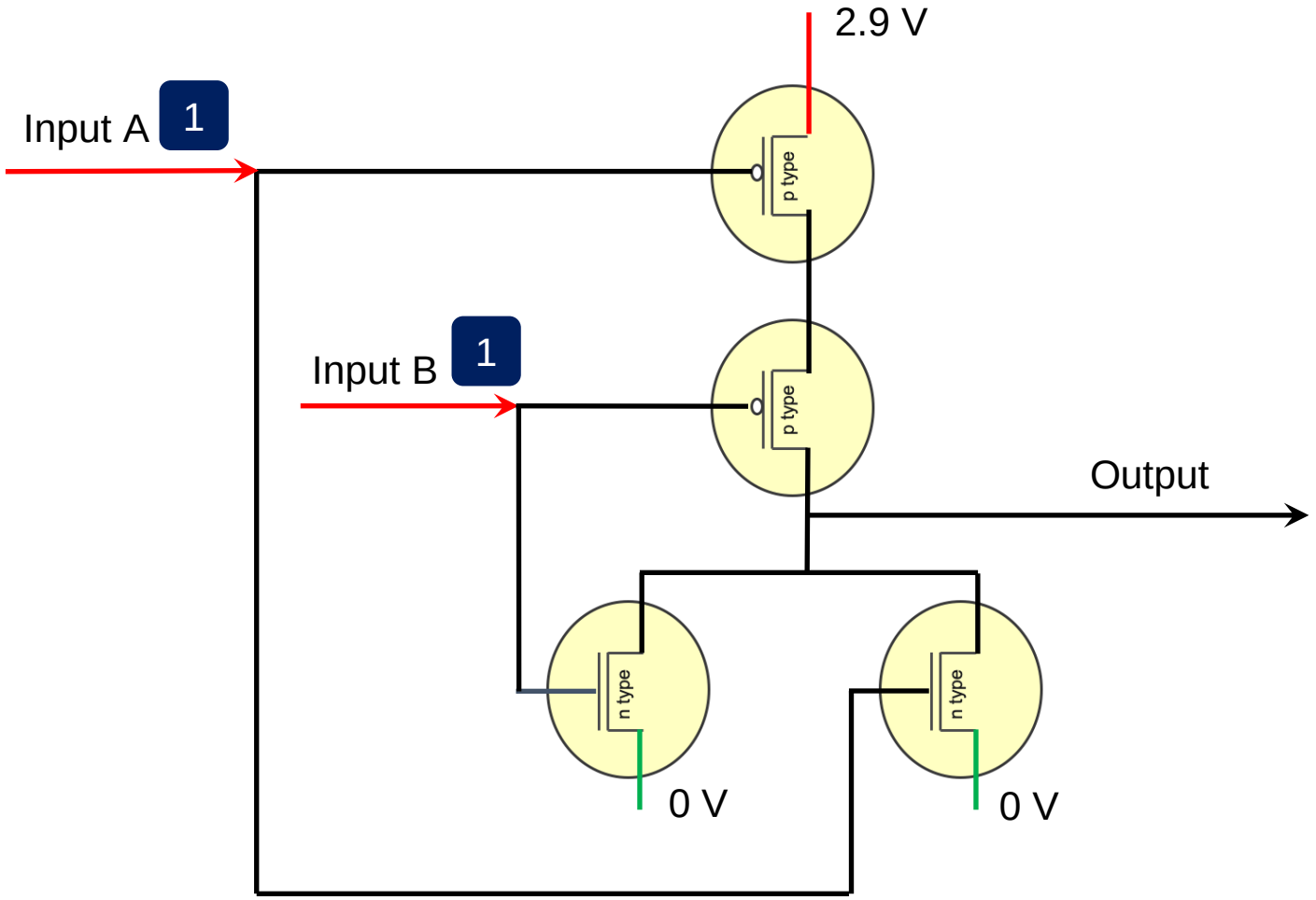
Input A	Input B	Output
0	0	1
1	0	0
0	1	0
1	**1**	**0**



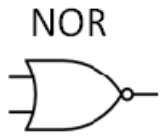
NOR Gate – For Input (1, 1)



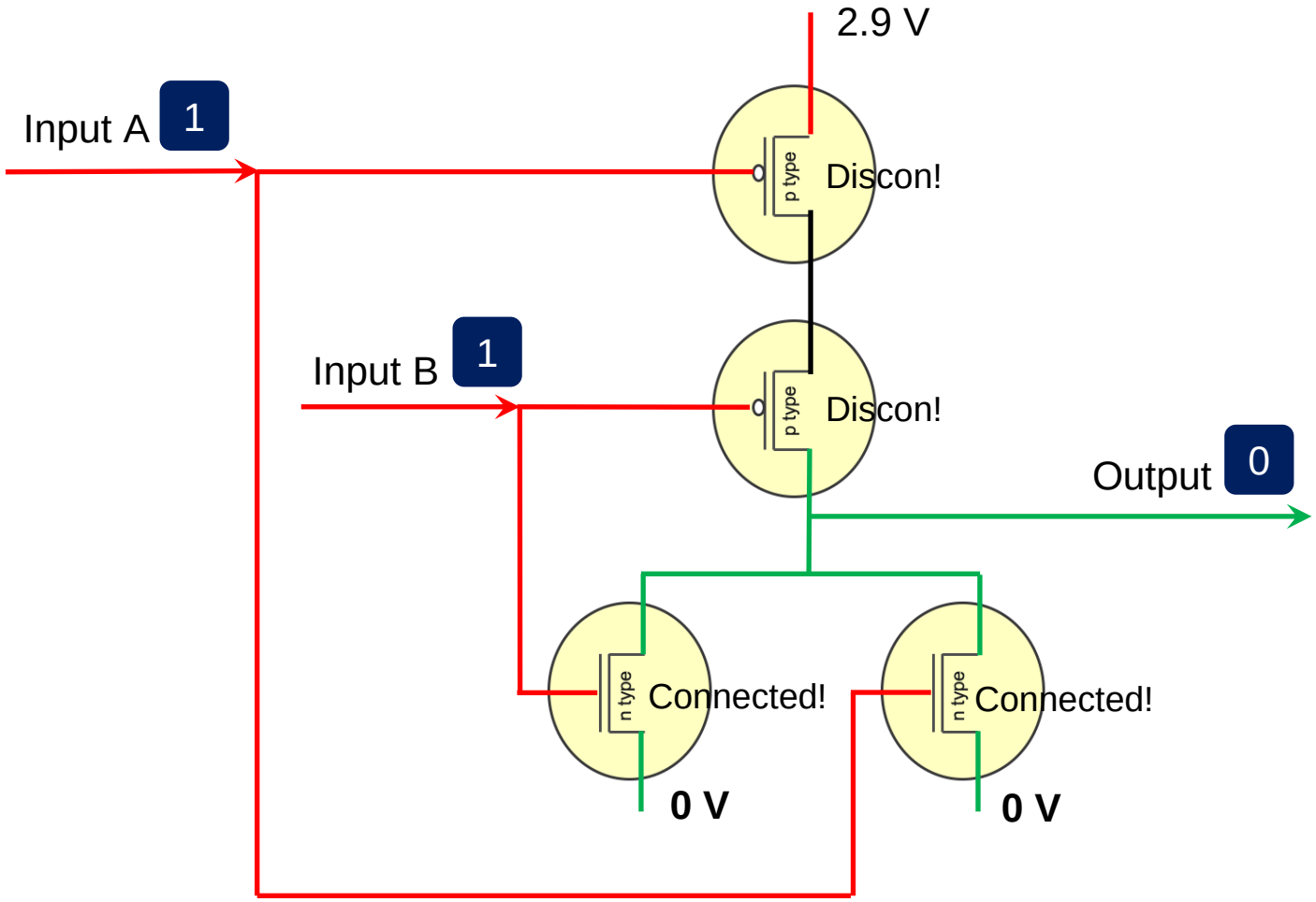
Input A	Input B	Output
0	0	1
1	0	0
0	1	0
1	**1**	**0**



NOR Gate – For Input (1, 1)



Input A	Input B	Output
0	0	1
1	0	0
0	1	0
1	**1**	**0**



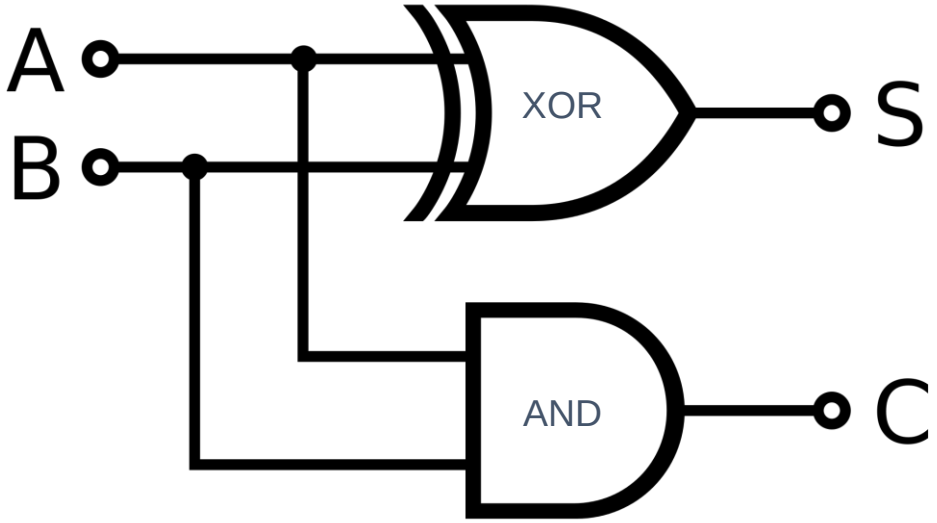
Combining logic gates

By combining the basic logic gates,
We can do more general ***binary computation****!*

1-bit Binary (Half) Adder

- Adding two unit binary digits (no carry as an input)

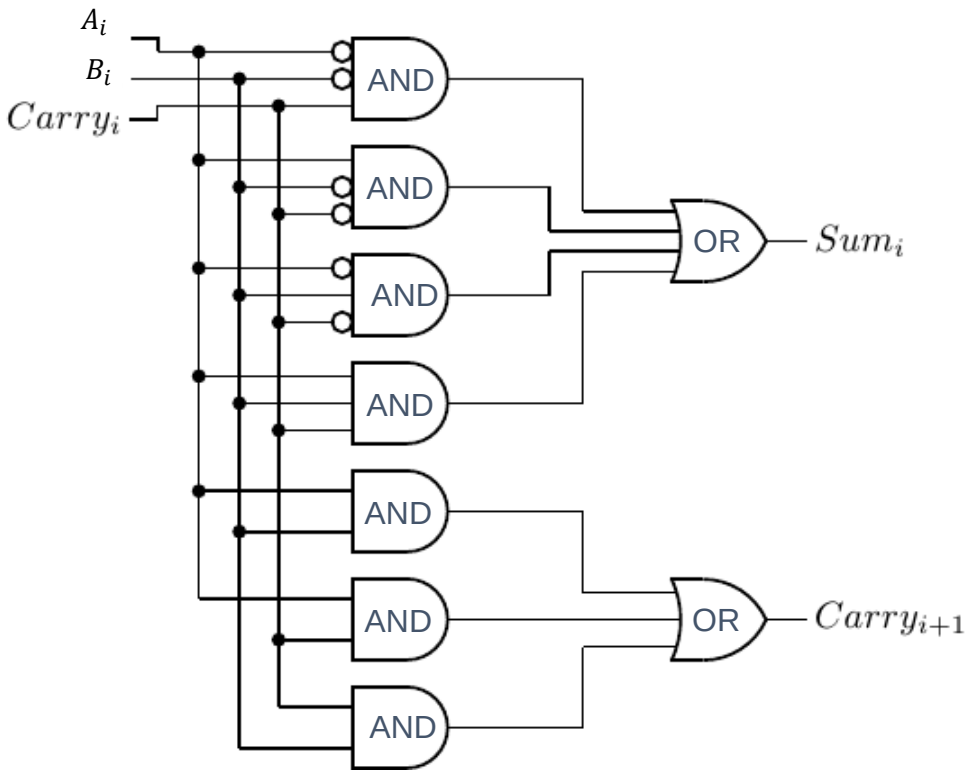
A	B	Sum	Carry
0	0	**0**	**0**
1	0	**1**	**0**
0	1	**1**	**0**
1	1	**0**	**1**



1-bit Binary (Full) Adder

- Adding two i-th binary digits (with carry input)

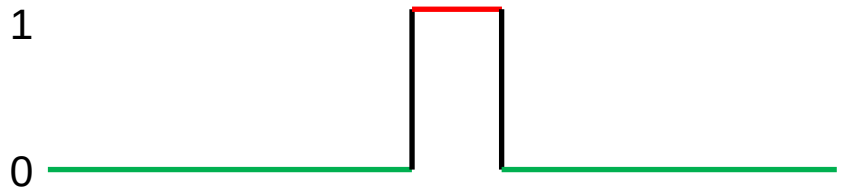
A_i	B_i	$Carry_i$	Sum_i	$Carry_{i+1}$
0	0	0	**0**	**0**
0	1	0	**1**	**0**
1	0	0	**1**	**0**
1	1	0	**0**	**1**
0	0	1	**1**	**0**
0	1	1	**0**	**1**
1	0	1	**0**	**1**
1	1	1	**1**	**1**



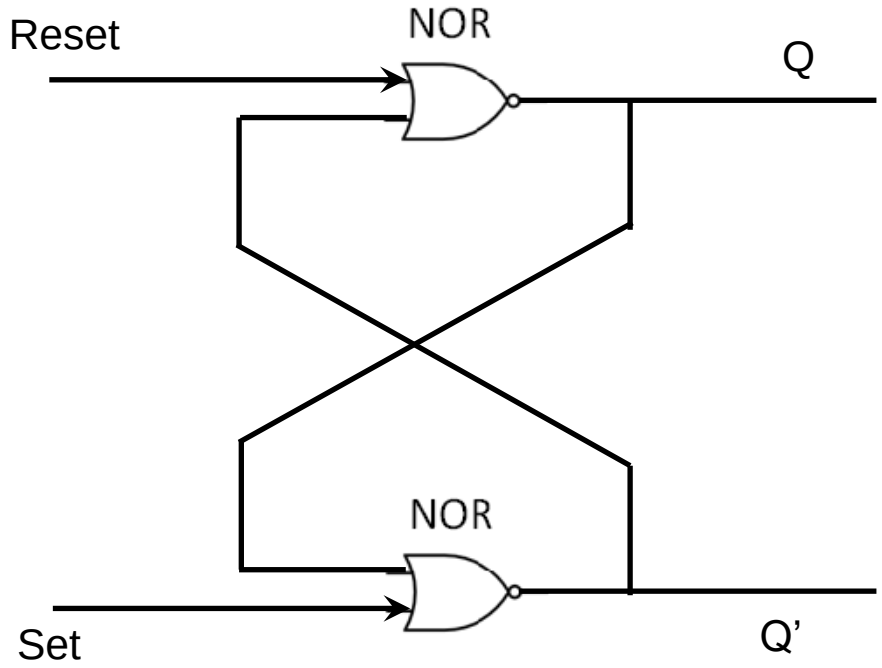
*We can even **store** binary information by arranging multiple transistors (i.e., memory)*

R-S Latch – Overview

- Q is the binary value that R-S Latch stores
  - Q' is the complement of Q
- Set and Reset are 0 most of the time
  - Each can be 1 for a short time (pulse)

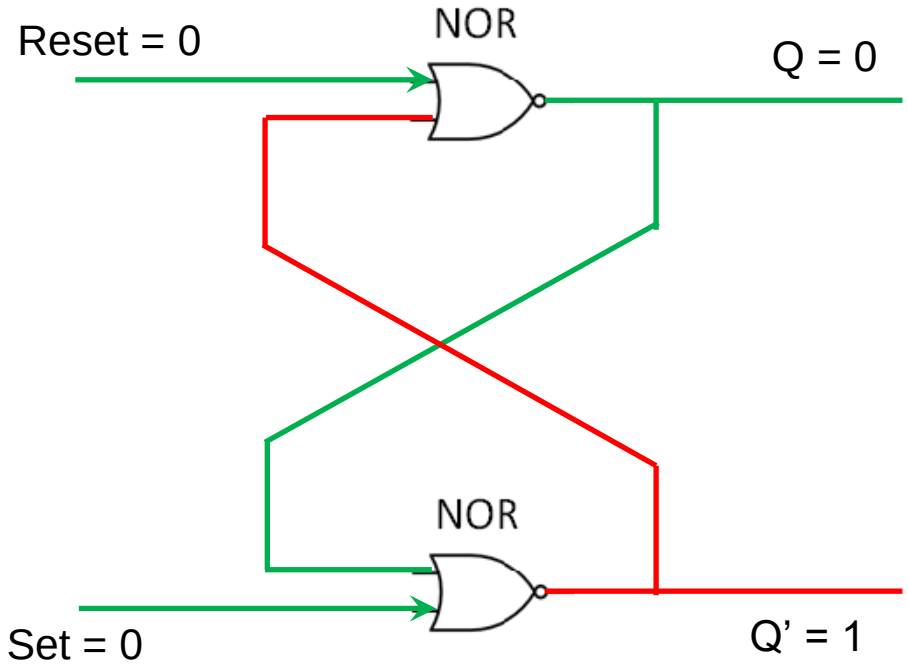


- When Set=1, 1 starts to be stored in Q
- When Reset=1, 0 starts to be stored in Q



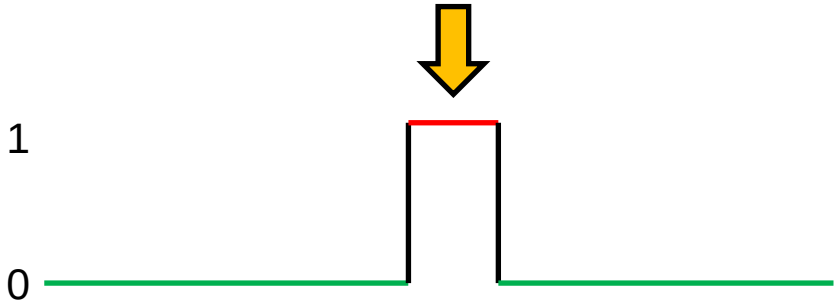
R-S Latch – Steady State

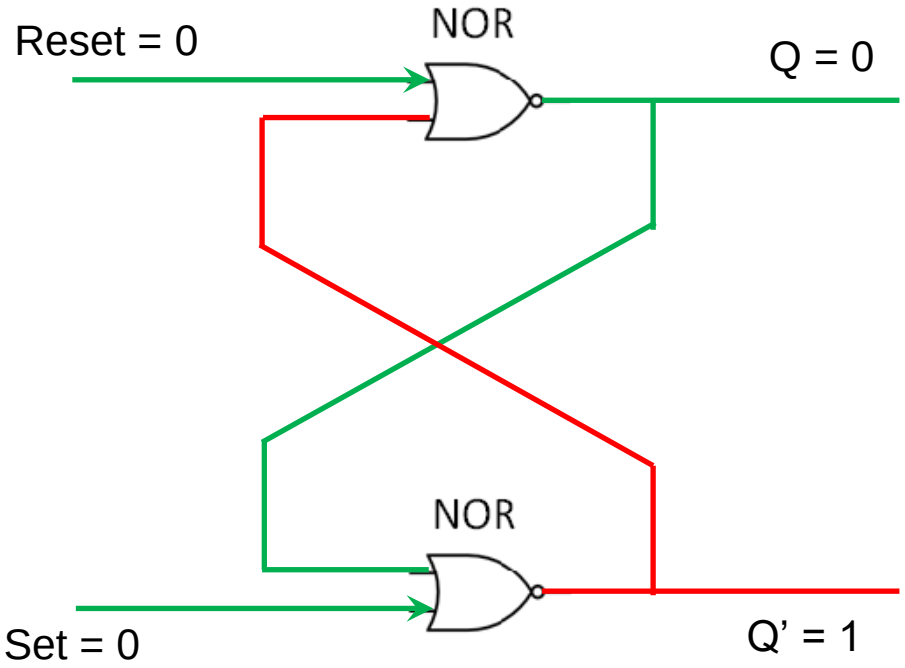
- Q = 0 is stored



R-S Latch – Store binary value 1

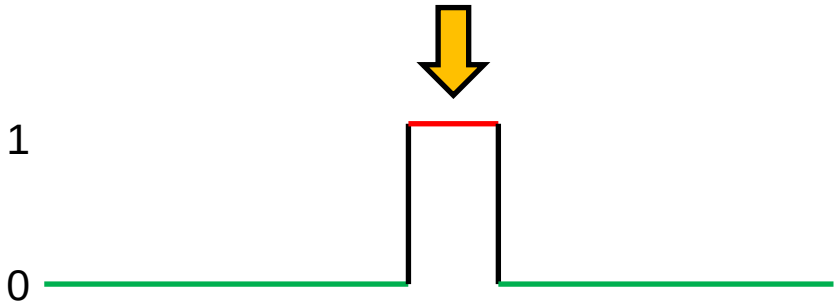
- Input a binary pulse to “Set”
  - Set becomes 1

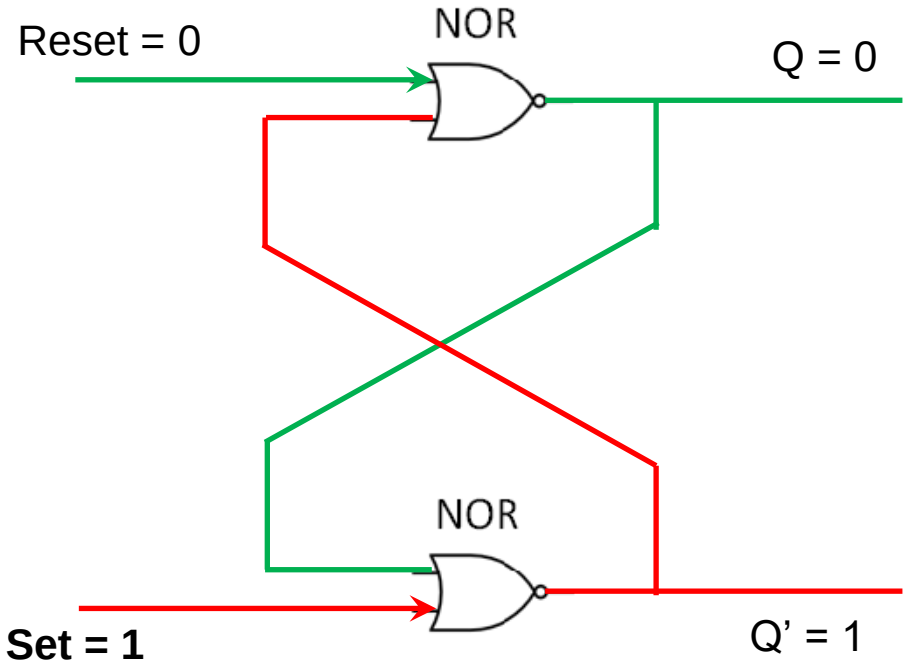




R-S Latch – Store binary value 1

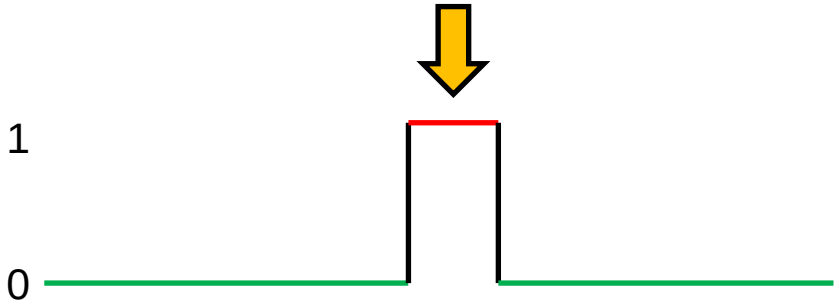
- Input a binary pulse to “Set”
  - Set becomes 1

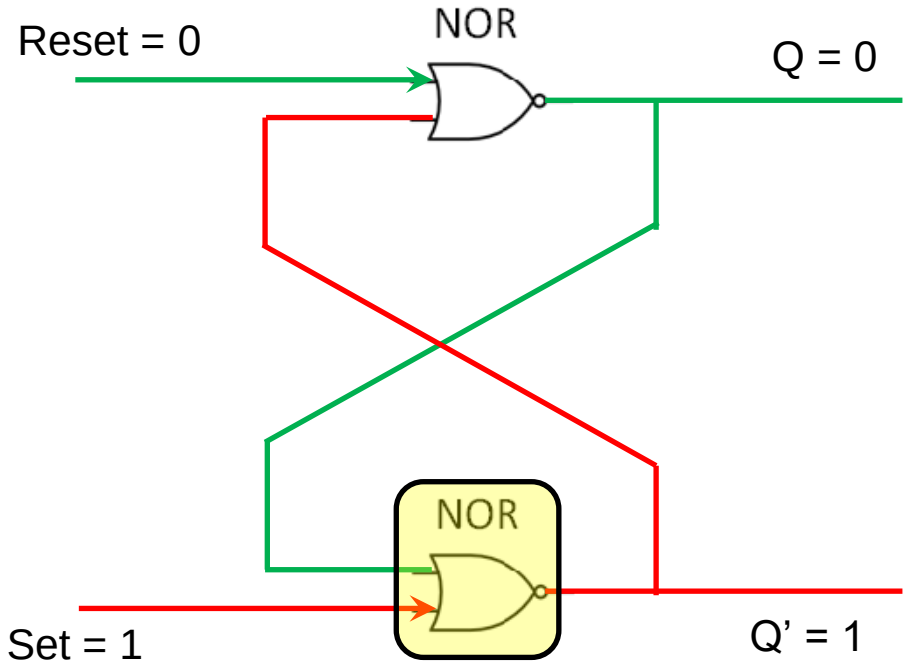




R-S Latch – Store binary value 1

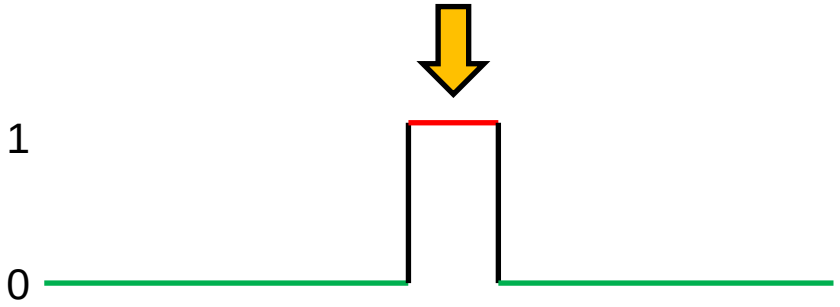
- Input a binary pulse to “Set”
  - Set becomes 1
  - Lower NOR gate makes Q’ become 0

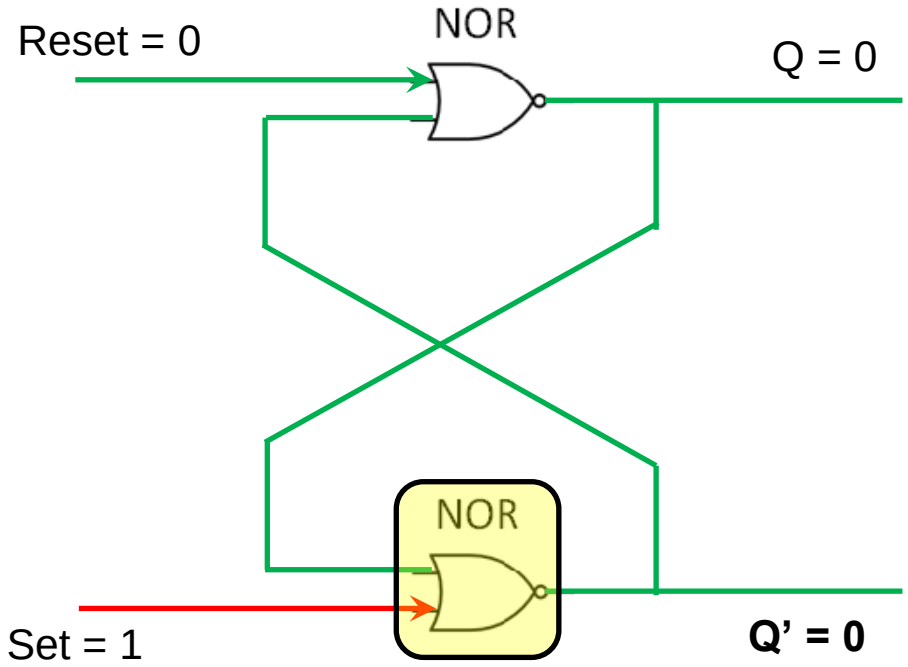




R-S Latch – Store binary value 1

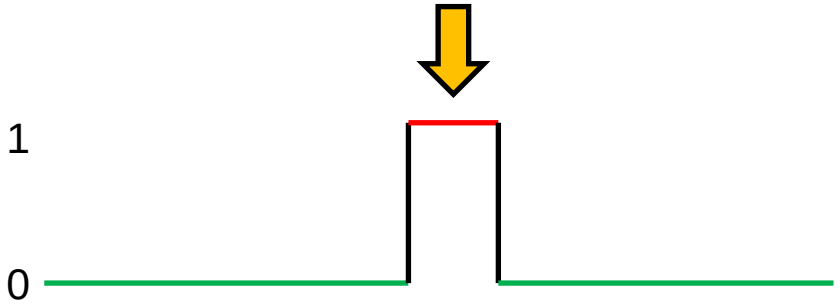
- Input a binary pulse to “Set”
  - Set becomes 1
  - Lower NOR gate makes Q’ become 0

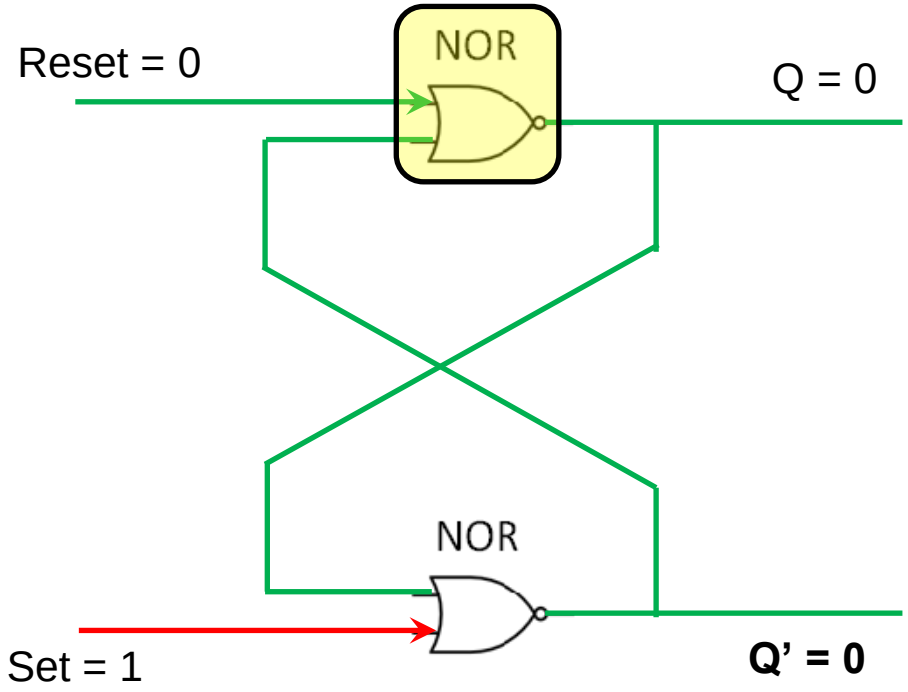




R-S Latch – Store binary value 1

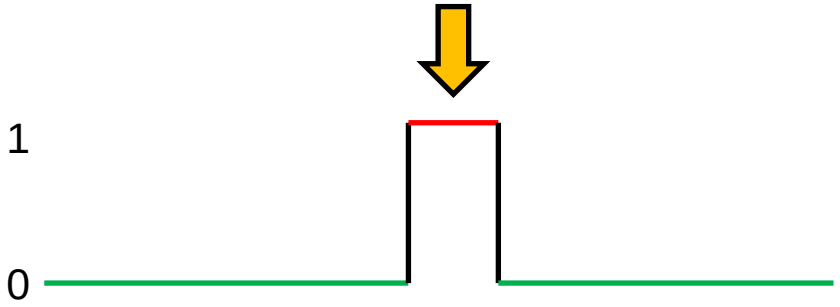
- Input a binary pulse to “Set”
  - Set becomes 1
  - Lower NOR gate makes Q’ become 0
  - Upper NOR gate makes Q become 1

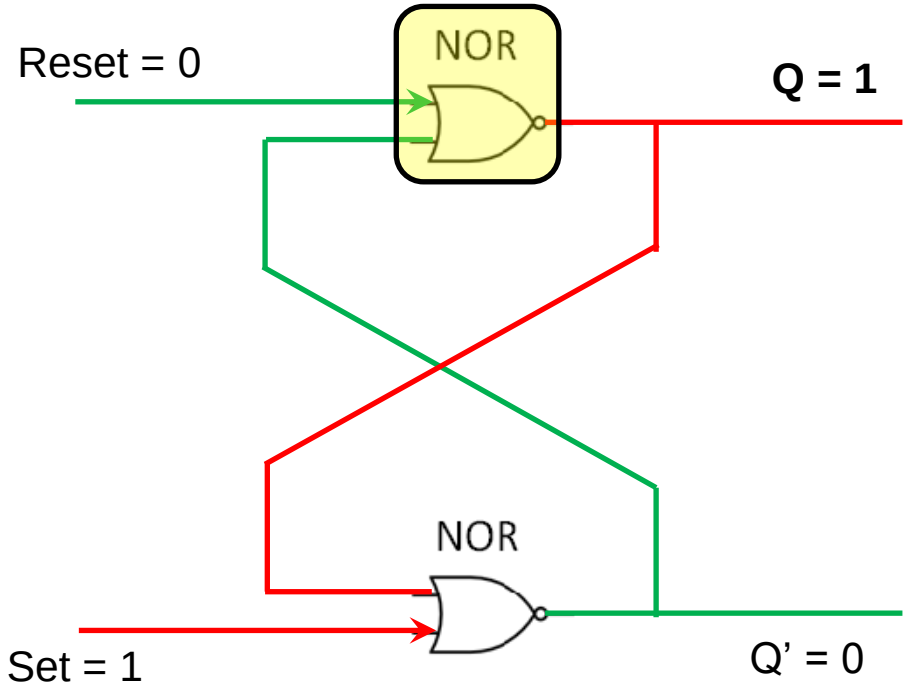




R-S Latch – Store binary value 1

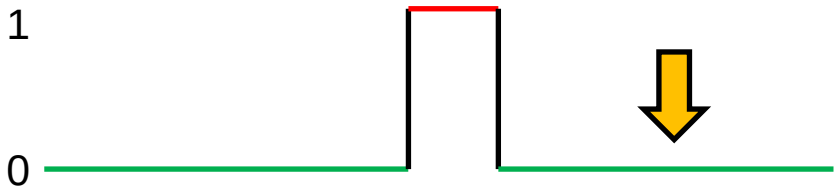
- Input a binary pulse to “Set”
  - Set becomes 1
  - Lower NOR gate makes Q’ become 0
  - Upper NOR gate makes Q become 1

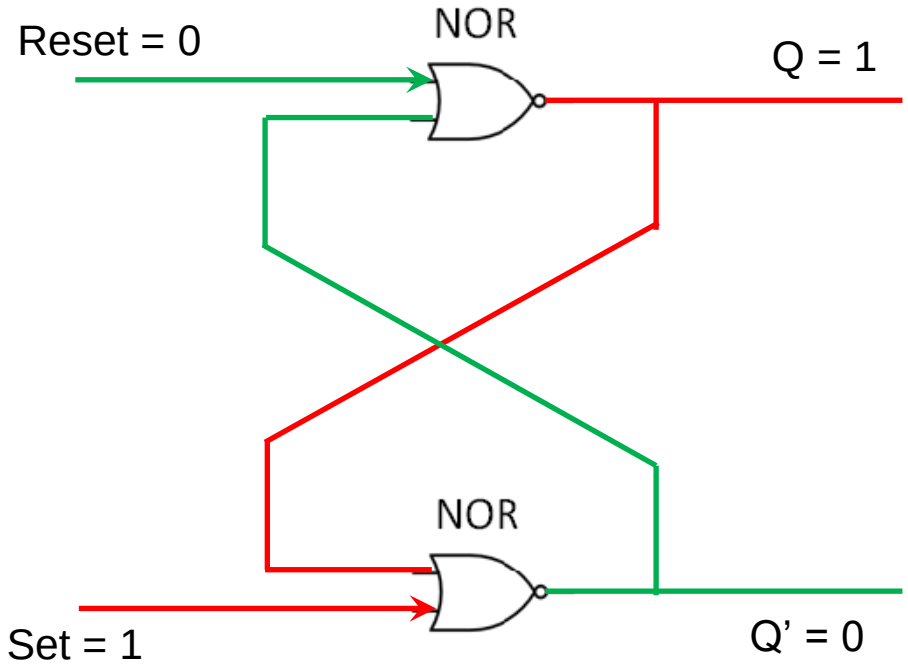




R-S Latch – Store binary value 1

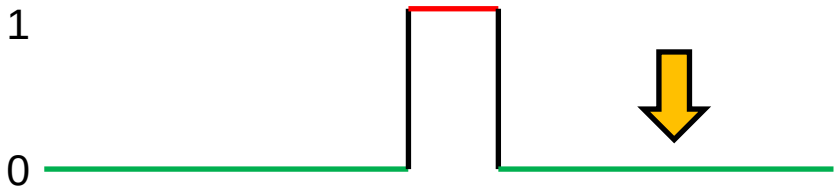
- Input a binary pulse to “Set”
  - Set becomes 1
  - Lower NOR gate makes Q’ become 0
  - Upper NOR gate makes Q become 1
  - Set goes back to 0

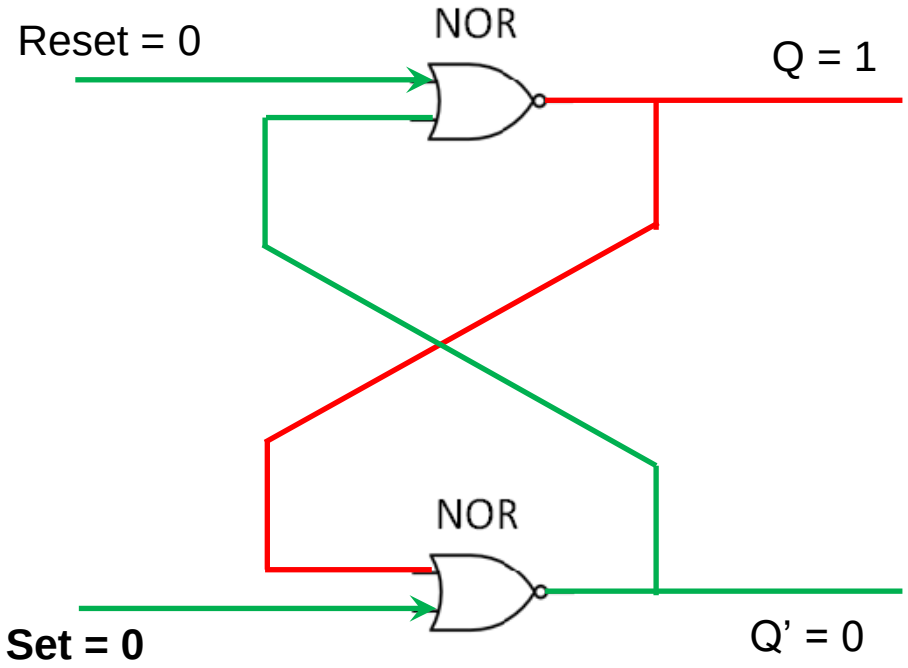




R-S Latch – Store binary value 1

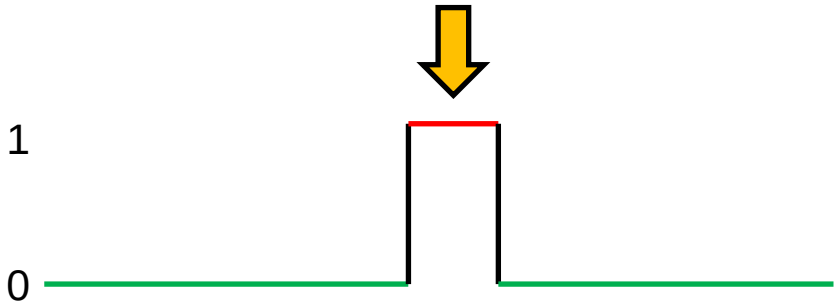
- Input a binary pulse to “Set”
  - Set becomes 1
  - Lower NOR gate makes Q’ become 0
  - Upper NOR gate makes Q become 1
  - Set goes back to 0
  - Now the latch goes back to the steady state
  - But stores binary value 1

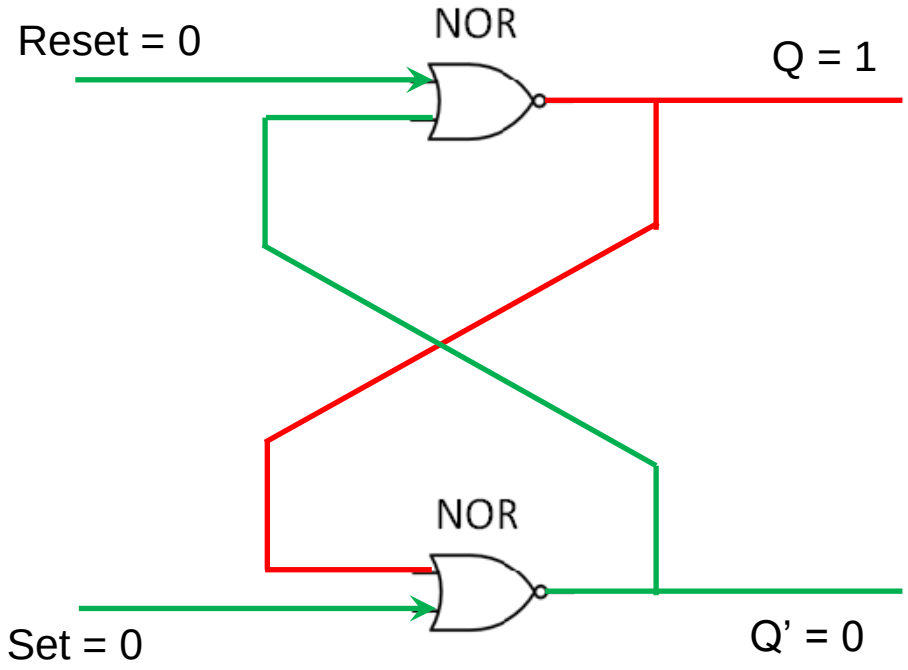




R-S Latch – Store binary value 0

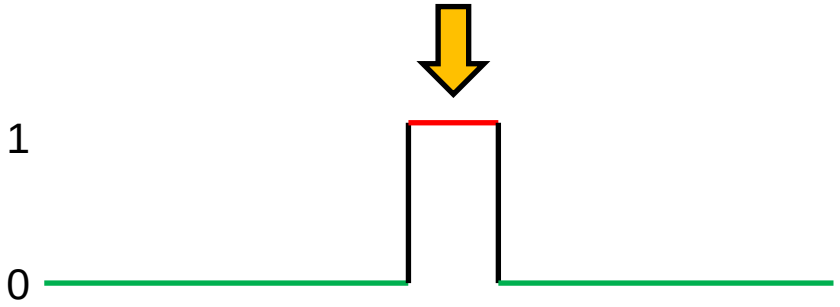
- Input a binary pulse to “Reset”
  - Reset becomes 1

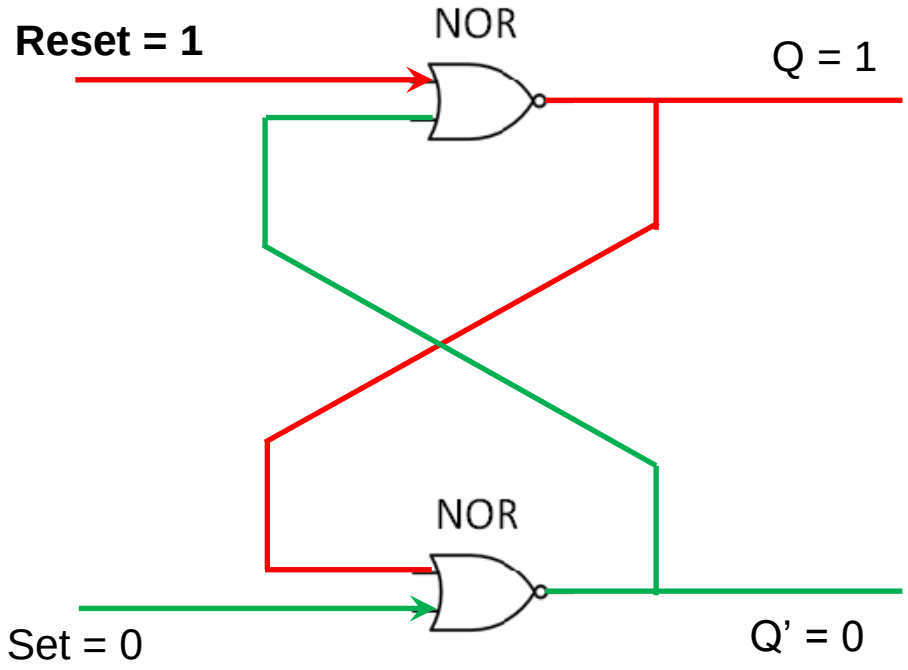




R-S Latch – Store binary value 0

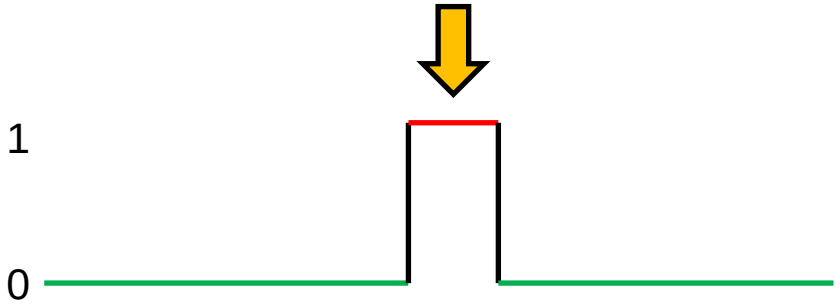
- Input a binary pulse to “Reset”
  - Reset becomes 1

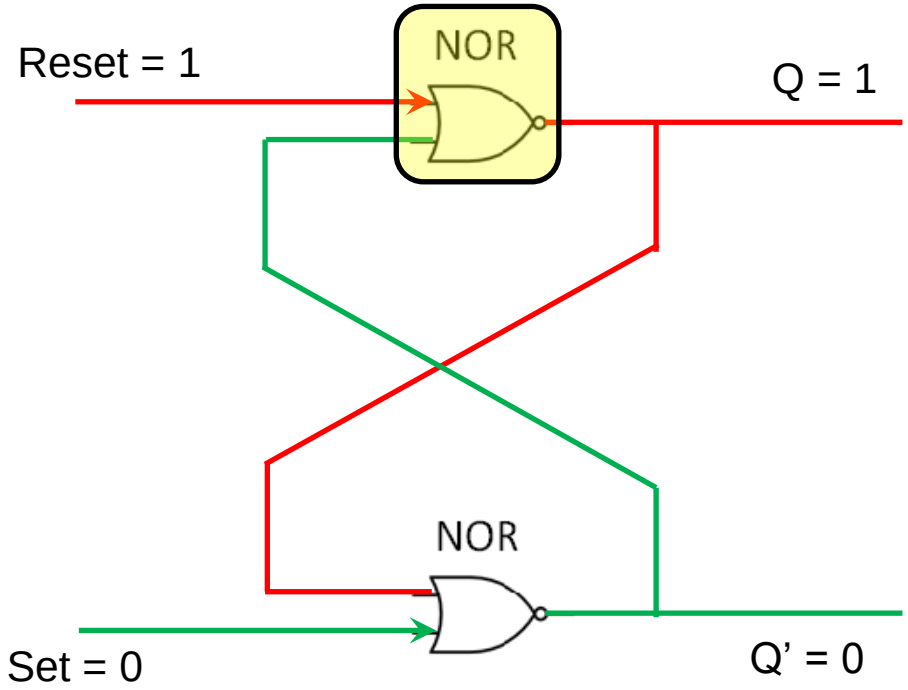




R-S Latch – Store binary value 0

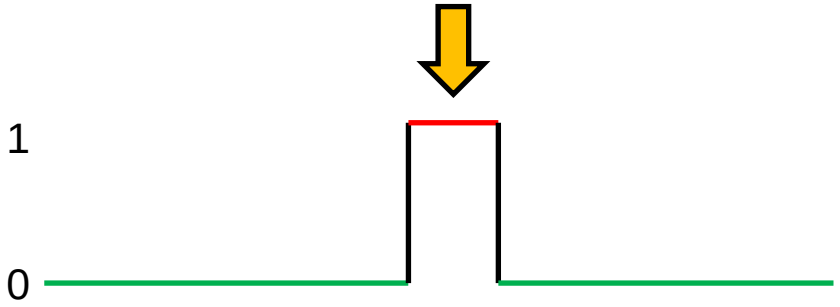
- Input a binary pulse to “Reset”
  - Reset becomes 1
  - Upper NOR gate makes Q become 0

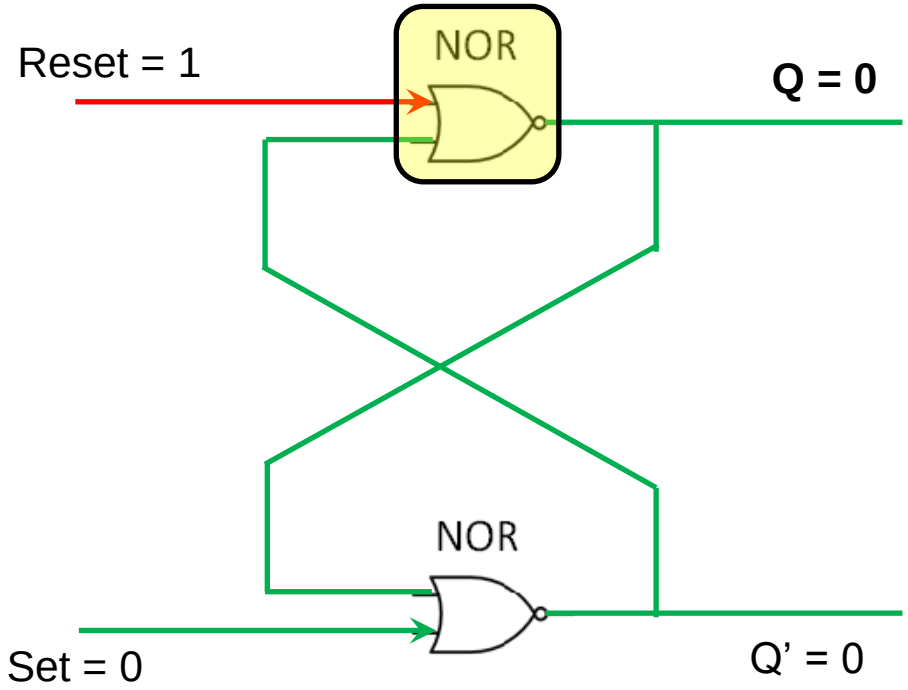




R-S Latch – Store binary value 0

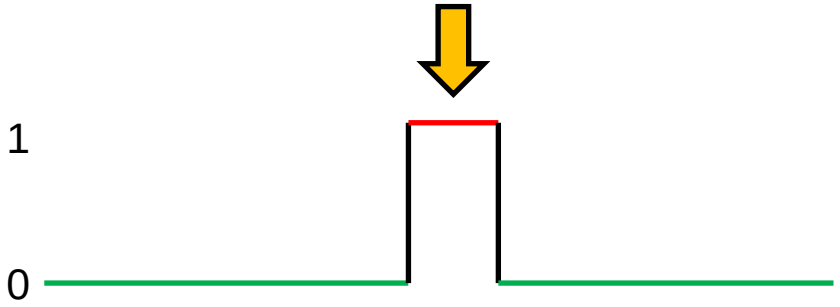
- Input a binary pulse to “Reset”
  - Reset becomes 1
  - Upper NOR gate makes Q become 0

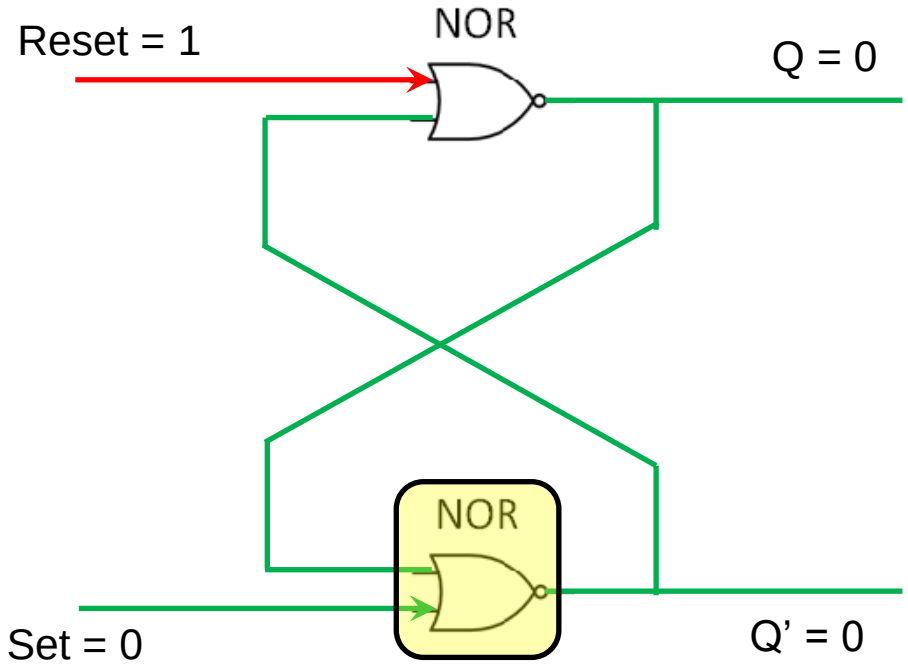




R-S Latch – Store binary value 0

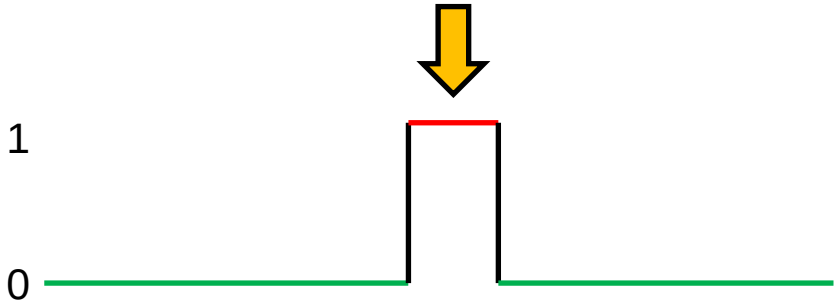
- Input a binary pulse to “Reset”
  - Reset becomes 1
  - Upper NOR gate makes Q become 0
  - Lower NOR gate makes Q’ become 1

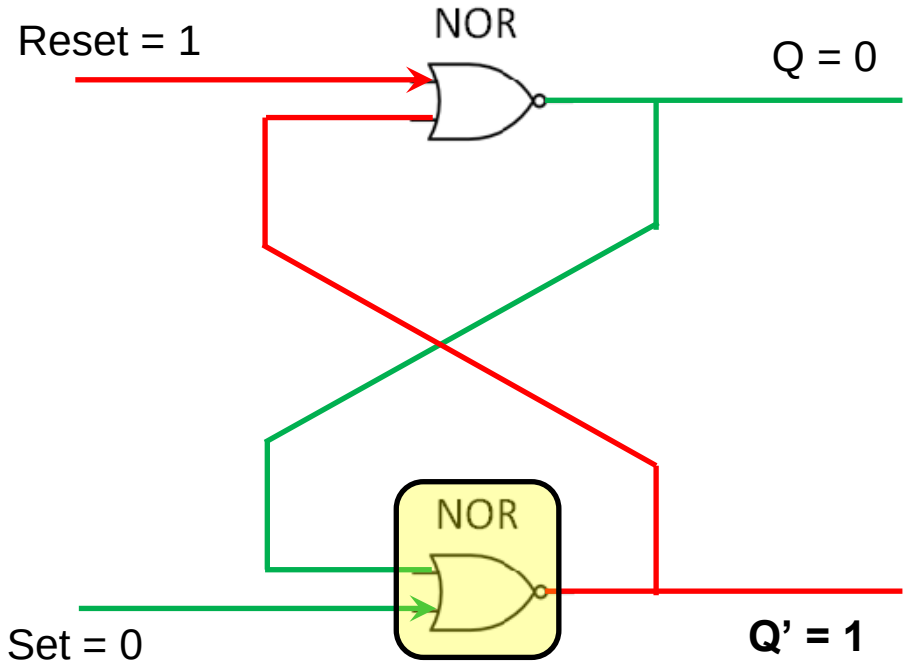




R-S Latch – Store binary value 0

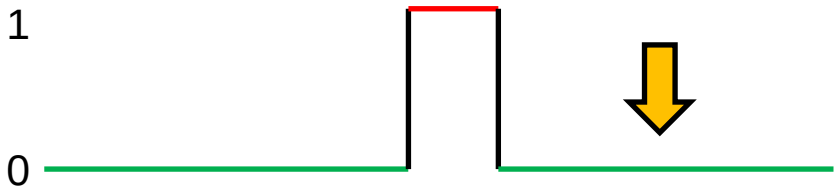
- Input a binary pulse to “Reset”
  - Reset becomes 1
  - Upper NOR gate makes Q become 0
  - Lower NOR gate makes Q’ become 1

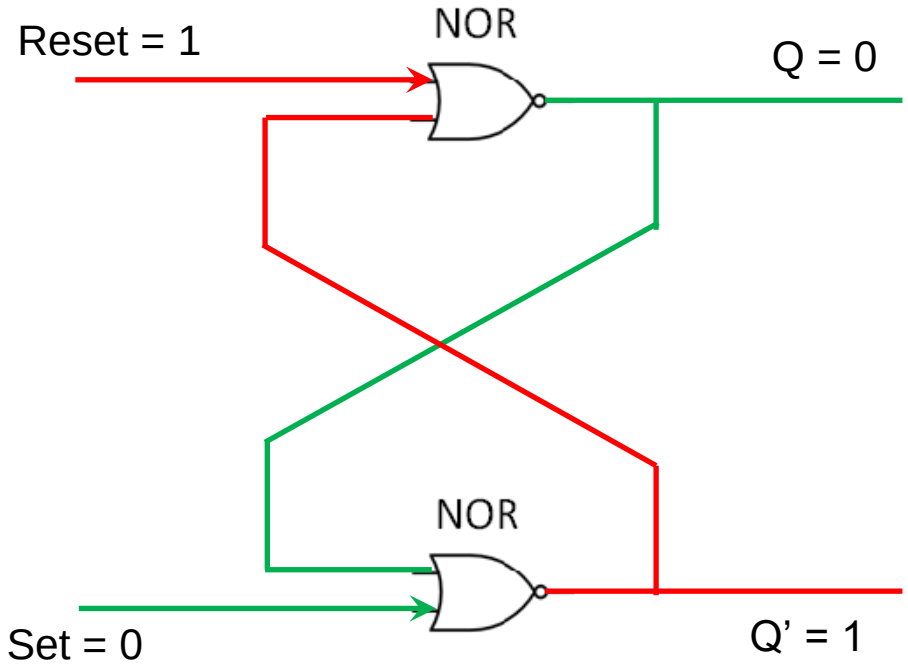




R-S Latch – Store binary value 0

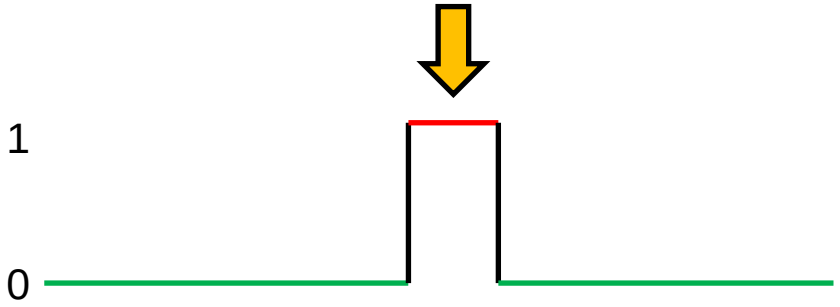
- Input a binary pulse to “Reset”
  - Reset becomes 1
  - Upper NOR gate makes Q become 0
  - Lower NOR gate makes Q’ become 1
  - Reset goes back to 0

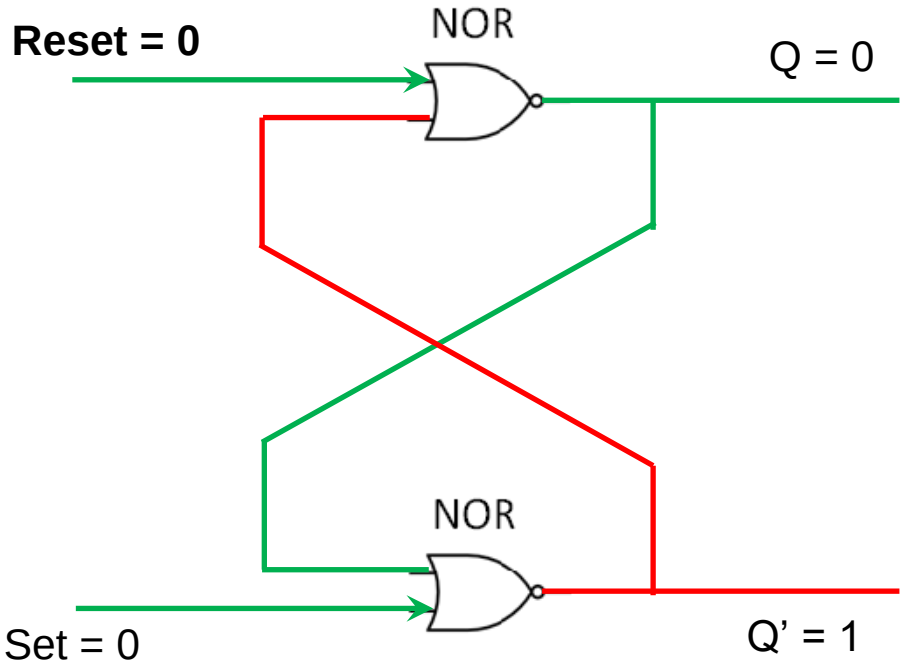




R-S Latch – Store binary value 0

- Input a binary pulse to “Reset”
  - Reset becomes 1
  - Upper NOR gate makes Q become 0
  - Lower NOR gate makes Q’ become 1
  - Reset goes back to 0
  - Now the latch goes back to the steady state
  - But stores binary value 0





Memory

- Memory has a number of uniquely identifiable memory locations
  - Each location has its own **address**
  - Each location stores a fixed number of bits, **one byte** (called **addressability**)
- A location's address is represented as a bit sequence
- You can write (or read) an information to (or from) the memory location by using the location's address (circuits for the other locations are closed!)

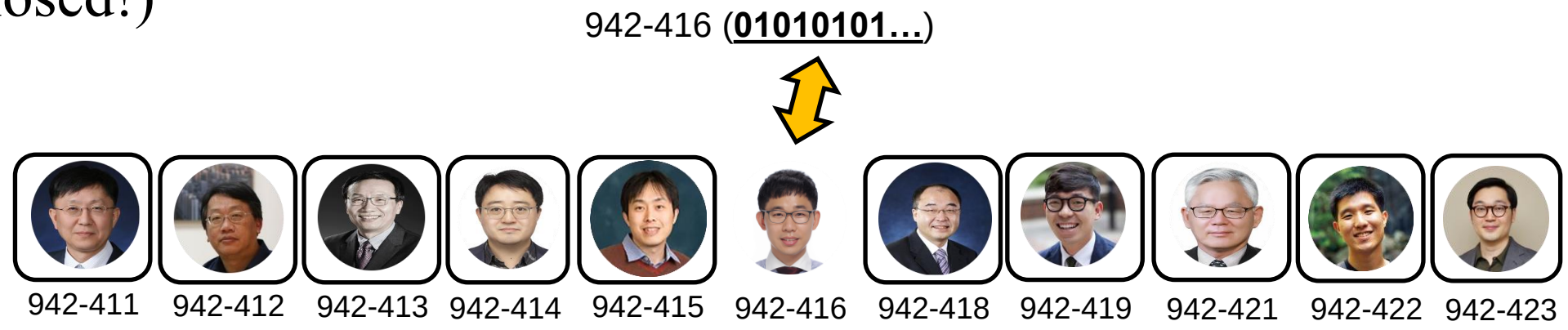
Memory

- Memory has a number of uniquely identifiable memory locations
  - Each location has its own **address**
  - Each location stores a fixed number of bits, **one byte** (called **addressability**)
- A location's address is represented as a bit sequence
- You can write (or read) an information to (or from) the memory location by using the location's address (circuits for the other locations are closed!)

942-416 (**01010101…**)

Memory

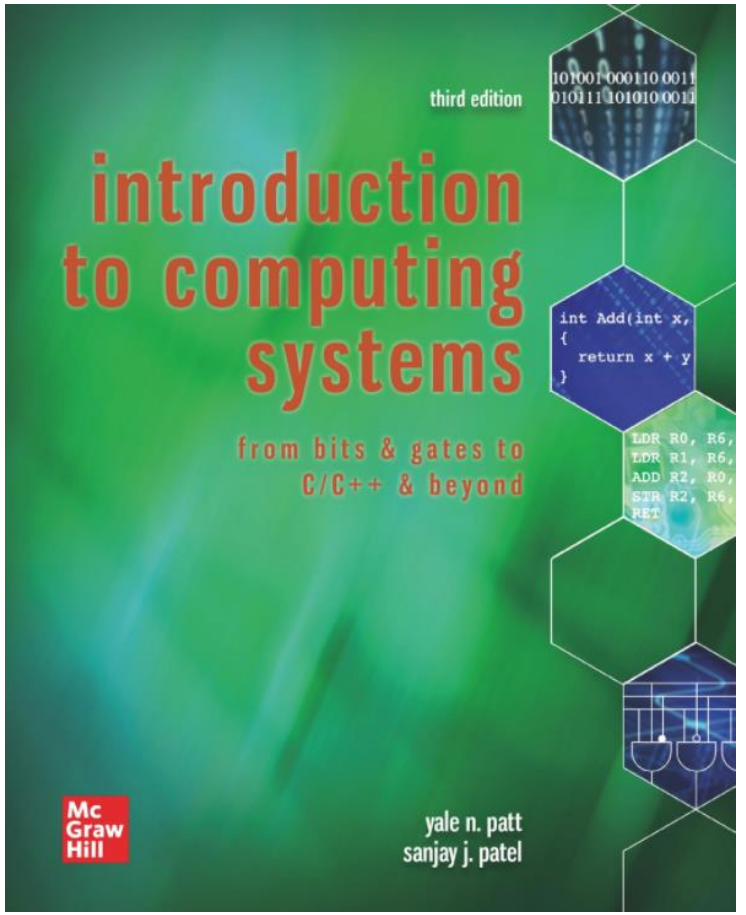
- Memory has a number of uniquely identifiable memory locations
  - Each location has its own **address**
  - Each location stores a fixed number of bits, **one byte** (called **addressability**)
- A location's address is represented as a bit sequence
- You can write (or read) an information to (or from) the memory location by using the location's address (circuits for the other locations are closed!)



For More Details…

- Textbook - Chapter 3

- Logic gate courses provided by ECE and CS
  - You will have a change to play with FPGA



Summary

Summary

- Thanks to semi-conductor, we can convert a continuous voltage value to a binary value (analog to digital conversion)
- By combining transistors, we can build hardware that can **compute** and **store** binary values
- Today's CPU has about 3 billion transistors to perform complex tasks fast !

Software vs. Hardware

- If we can build software that can do a task, we can also build hardware that can do the same task

- Once you build the hardware for a task,
  - The hardware will do the task **super fast**
  - But it can do only the specific task… to do another task, you need to build another "computer"

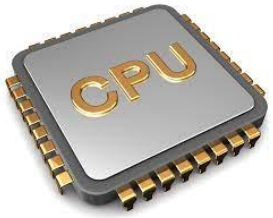
- Once you build software for a task,
  - The software will do the task relatively slower than hardware
  - But you can **easily build another** software (i.e., programming) without building another computer

Software ❤ Hardware

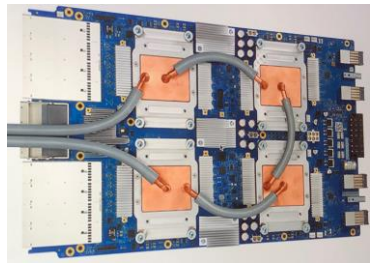
- There are various tasks that people want to do by using computers
- It is important to **synergistically** use hardware and software to do these tasks
- If a task is well defined and needs to be done very frequently and fast, we can make it "**hard**"
  - It would be good to build a hardware component specialized for the specific task
- If not, we need flexibility to do the task, treating it "**soft**"ly

CPU ❤ Accelerators

- CPU is a processing unit that can execute general software instructions
  - Wide applications but slow performance



- Accelerator is a processing unit that is optimized for specific types of operations
  - Limited applications but fast performance (GPU, mobile GPU, TPU, edge TPU…)



Q&A

Any questions?

Thanks!