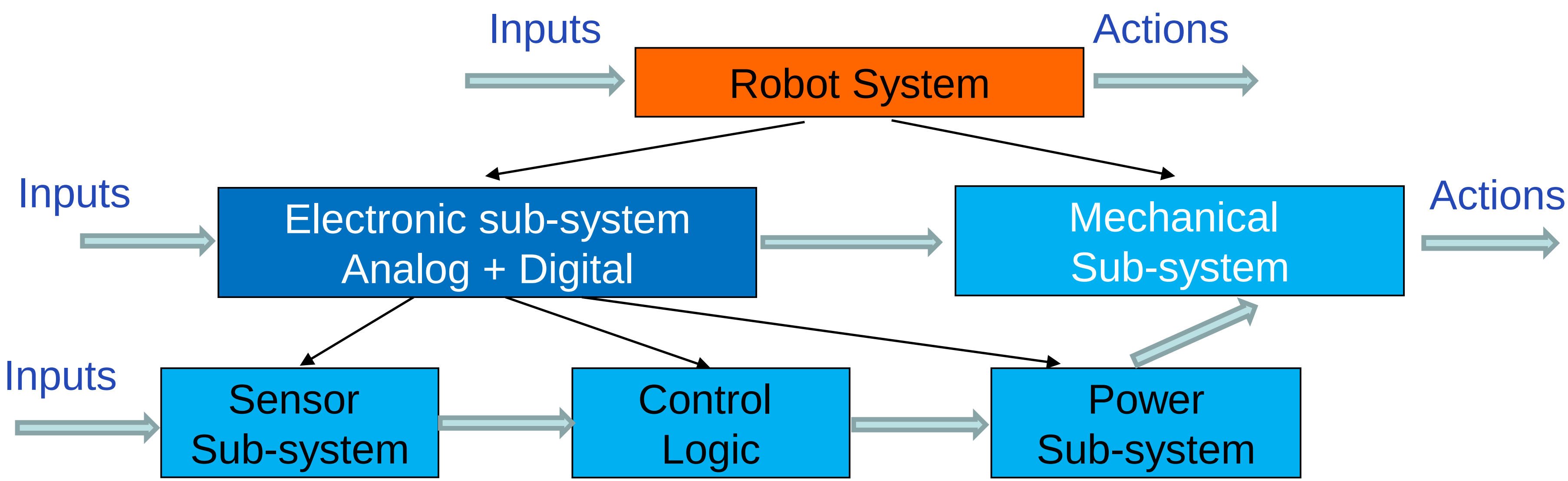
ELEC 1100: Introduction to Electro-Robot Design

Lecture 05: Sequential Logic + Embedded System + MCU

CHEN Wei, Dept. of ECE, HKUST

ELEC1100 ROADMAP



Sensor Basics:
Wk2: Sensor Basic

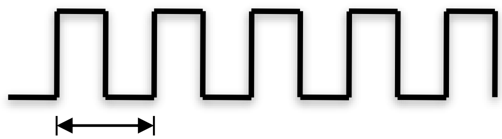
Combinational/Sequential Logic:
Wk3: Robot Brain: Logic Gate & Logic Operation
Wk4:Sequential Logic & MCU
Wk5:Programming Language

Basic electronics:
Wk1: Basic Electronics & DC Sources

Motor Power Supply:
Wk1: Pulse Signal & PWM Control
Wk2: Transistor and H-Bridge

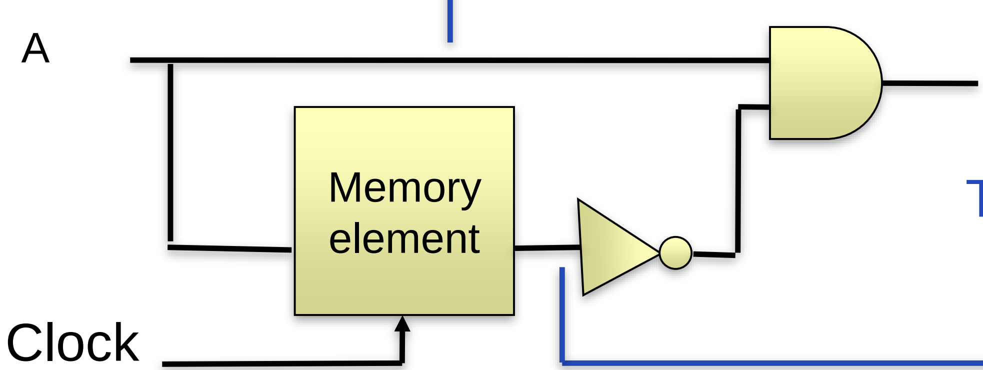
- Output of the logic not only depends on the current inputs but also on the history of the inputs
- Example: a counter's output depends on the previous number
- Needs two fundamentally new ingredients of sequential logic circuits:
  - A storage/memory element to keep the output at the current value when the input is removed
  - A regular clock that controls the updating of these storage elements with the results of new computations

Clock signal



T = Period

- Example of a sequential logic:
  - Take the AND of the input at the current time and the complement of the input at a previous time
  - Mathematically, we can express as



The nth input

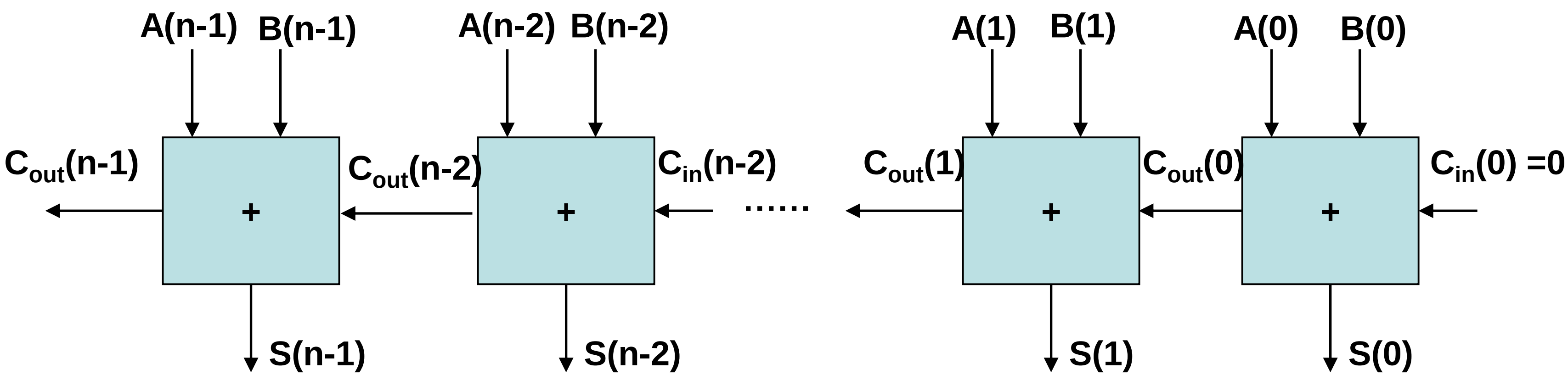
$$S_{out}(n) = A(n)\overline{A}(n-1)$$

The nth output

The (n-1)th input

ADDITION OF MULTI-BIT NUMBERS

- We have implemented one n-bit adder in previous lecture



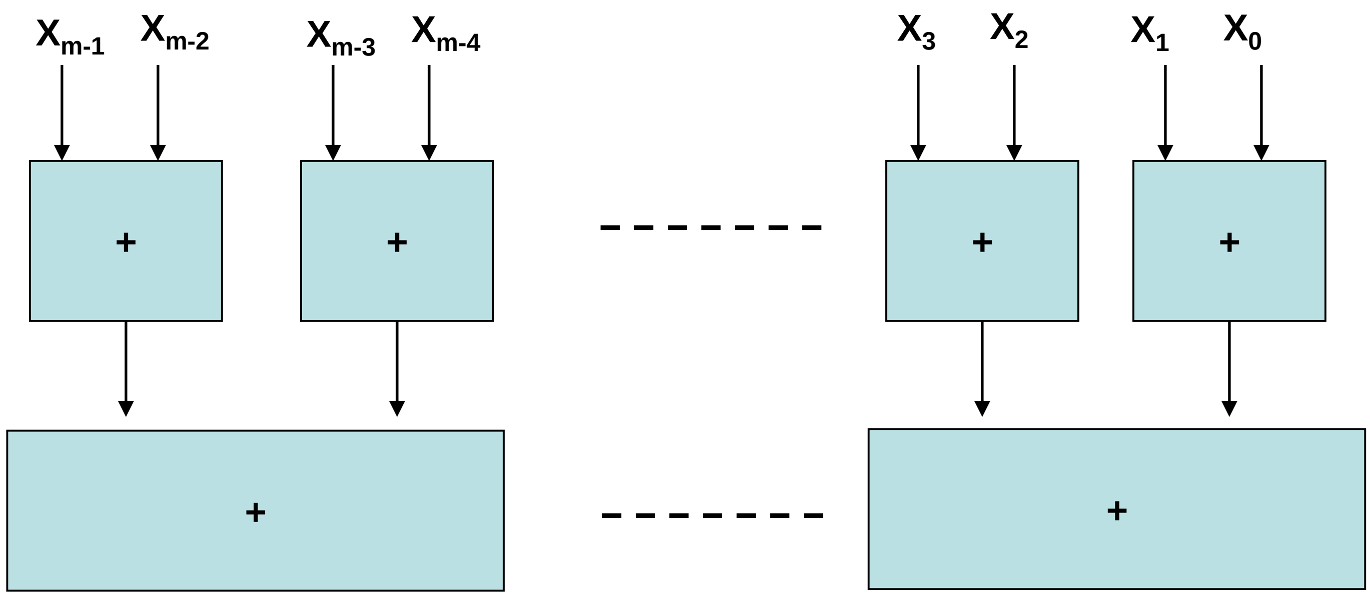
- Suppose we want to add *m* n-bit numbers (each X_i of the following equation is a N-bit number)

$$\text{Output} = X_{m-1} + X_{m-2} + \ldots + X_1 + X_0$$

- How would you find the output?

- Example: using divide and conquer

$$\text{Output} = (X_{m-1} + X_{m-2}) + \ldots + (X_1 + X_0)$$

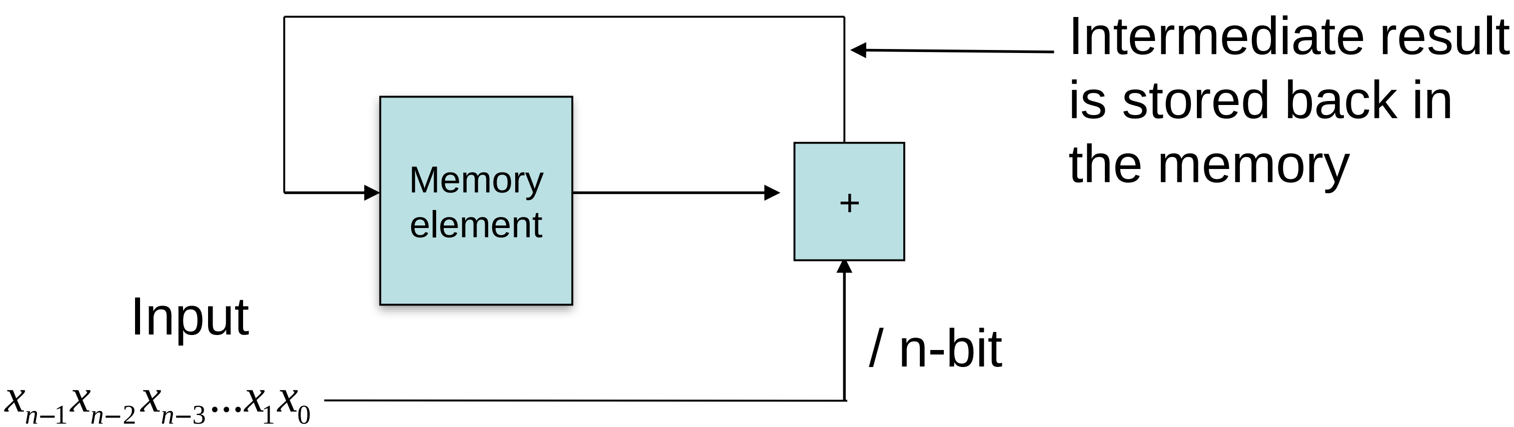


- How many adders do you need?

USING SEQUENTIAL LOGIC

- We can also use sequential logic to feedback the output of the intermediate sum to the input of the adder

$$\text{Output} = (..((X_0 + X_1) + X_2 + \ldots) + X_{m-1}) + X_{m-2})$$

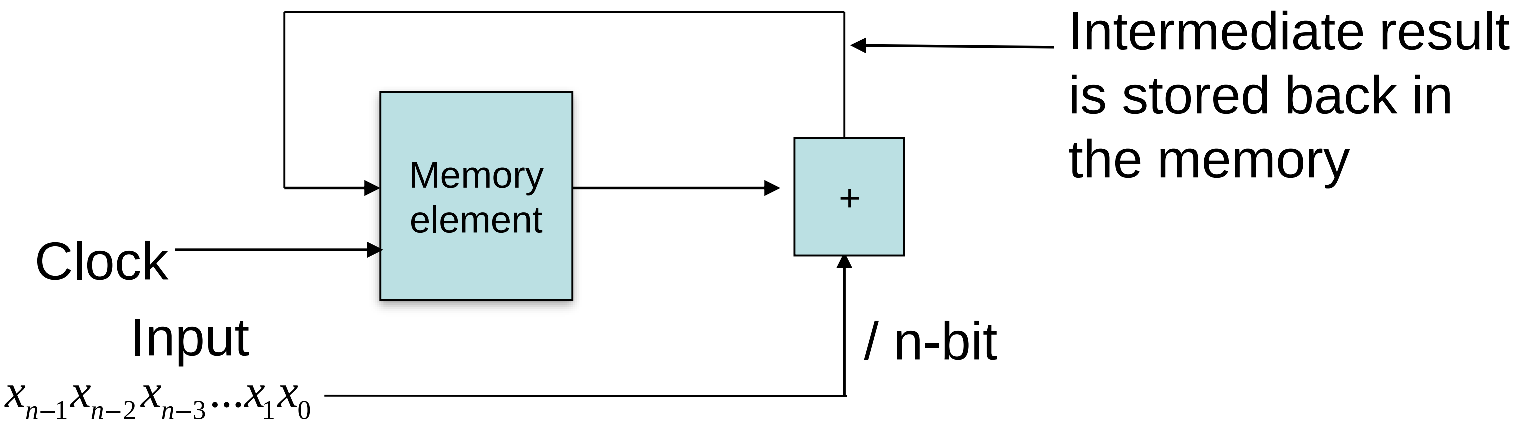


- What is missing in the scheme?
- When will we know it is ready to send the data in?

FINAL SEQUENTIAL LOGIC CIRCUIT

- We need a clock signal to synchronize the input and the memory output

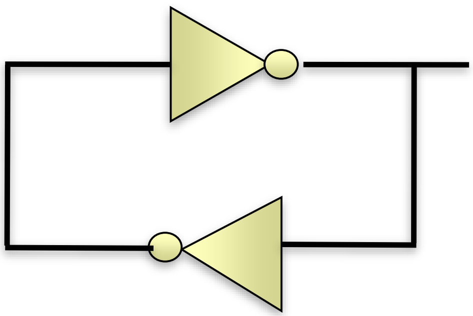
$$\text{Output} = (..((X_0 + X_1) + X_2 + \ldots) + X_{m-1}) + X_{m-2})$$



- What is the drawback compared with combinational logic?
- A key element, memory, is needed

FLIP-FLOP

- When we set the output of a flip-flop to 0, it should stay at 0, and if we set the output to 1, it should stay at 1
- It can be achieved by having two inverters cross connect to each other
- It is also called memory operation by feedback



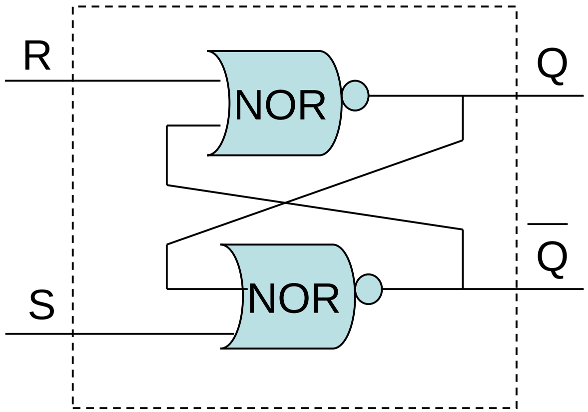
Output

Static memories are based on this cross-coupled feedback inverter pair

- How to set the value?

RS FLIP-FLOP USING NOR GATES

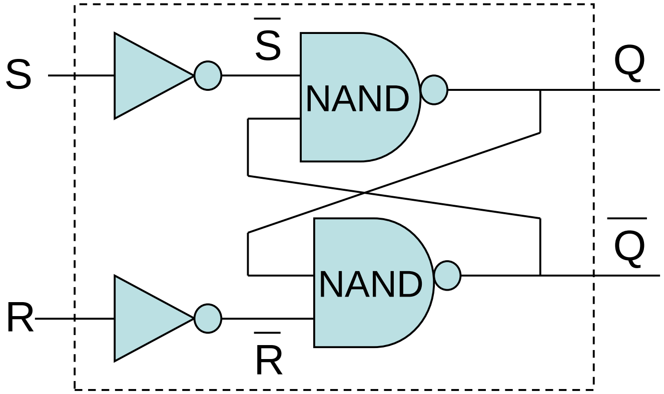
- It can be implemented using NOR gates instead of just inverters



R	S	Q (old)	Q(new)	$\overline{Q}$(new)	Function
0	0	0	0	1	(memory)
0	0	1	1	0	(memory)
0	1	0	1	0	(set)
0	1	1	1	0	(set)
1	0	0	0	1	(reset)
1	0	1	0	1	(reset)
1	1	0	0	0	(illegal)
1	1	1	0	0	(illegal)

RS FLIP-FLOP USING NAND GATES

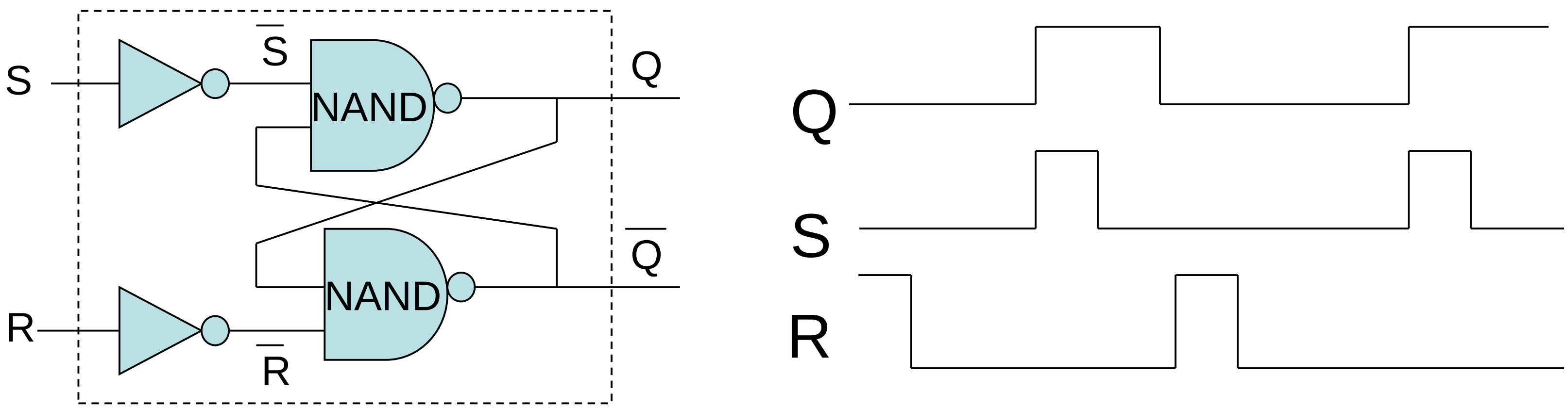
❖ It can be implemented using NAND gates, but two extra inverters are need to get the same truth table



$\overline{R}$	$\overline{S}$	Q (old)	Q(new)	$\overline{Q}$(new)	Function
1	1	0	0	1	(memory)
1	1	1	1	0	(memory)
1	0	0	1	0	(set)
1	0	1	1	0	(set)
0	1	0	0	1	(reset)
0	1	1	0	1	(reset)
0	0	0	0	0	(illegal)
0	0	1	0	0	(illegal)

TIMING DIAGRAM

- Actual operation of a flip-flop depends on the timing of the input signals
- For example:

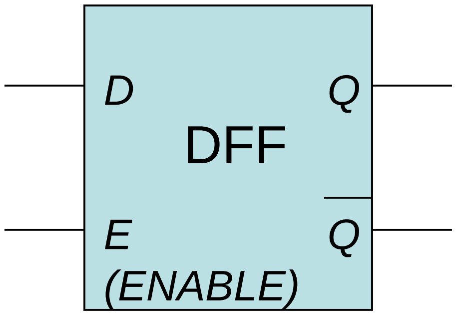


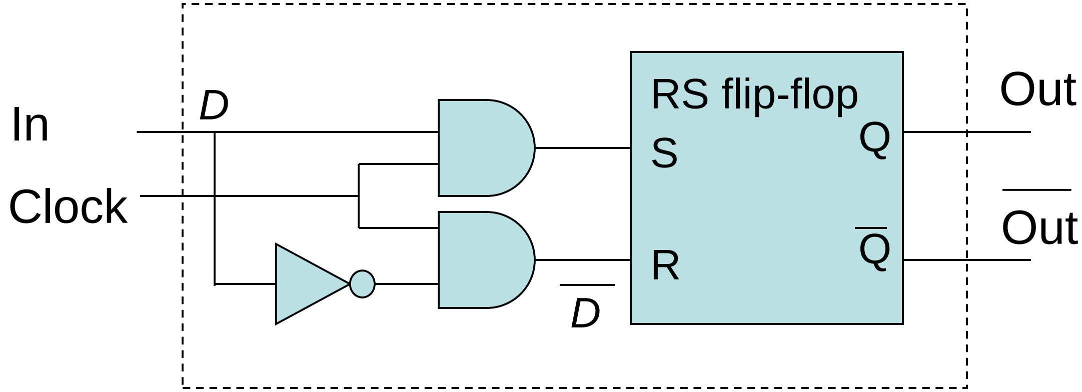
D FLIP-FLOP

- A transparent D flip-flop can be constructed from an RS flip-flip to allow clocking and synchronization (D is derived from “Data” and “Delay”) [important]
- When the clock is high, it acquires an input data
- When the clock is low, the memory remembers the data, no matter how the input changes, the output will not change
- Truth table of a D flip-flop

In	Clock	Operation
0	0	Q = Q (old)
1	0	Q = Q (old)
0	1	Q = 0
1	1	Q = 1

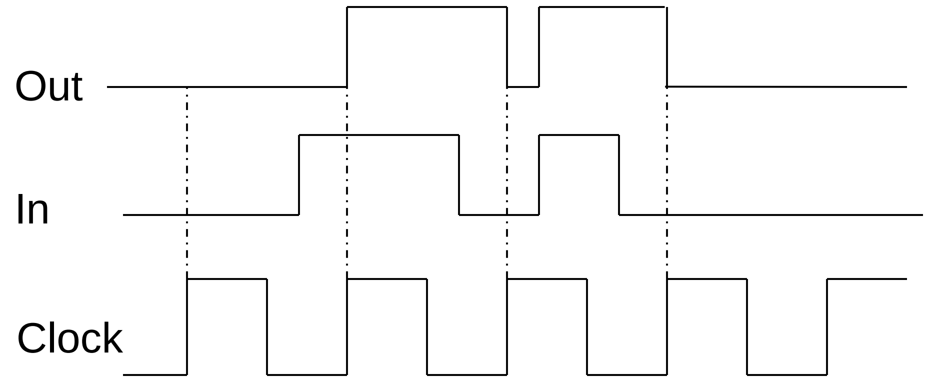
Schematic symbol





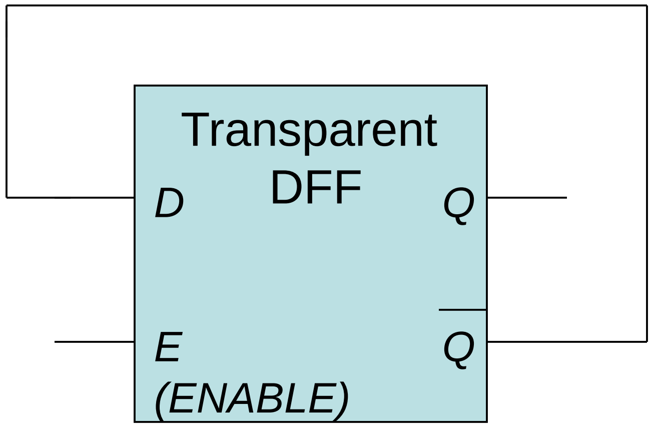
❖ Truth table and timing diagram

In	Clock	R	S	Operation
0	0	0	0	Q = Q (old)
1	0	0	0	Q = Q (old)
0	1	1	0	Q = 0
1	1	0	1	Q = 1



PROBLEM IN D FLIP-FLOP SEQUENTIAL LOGIC

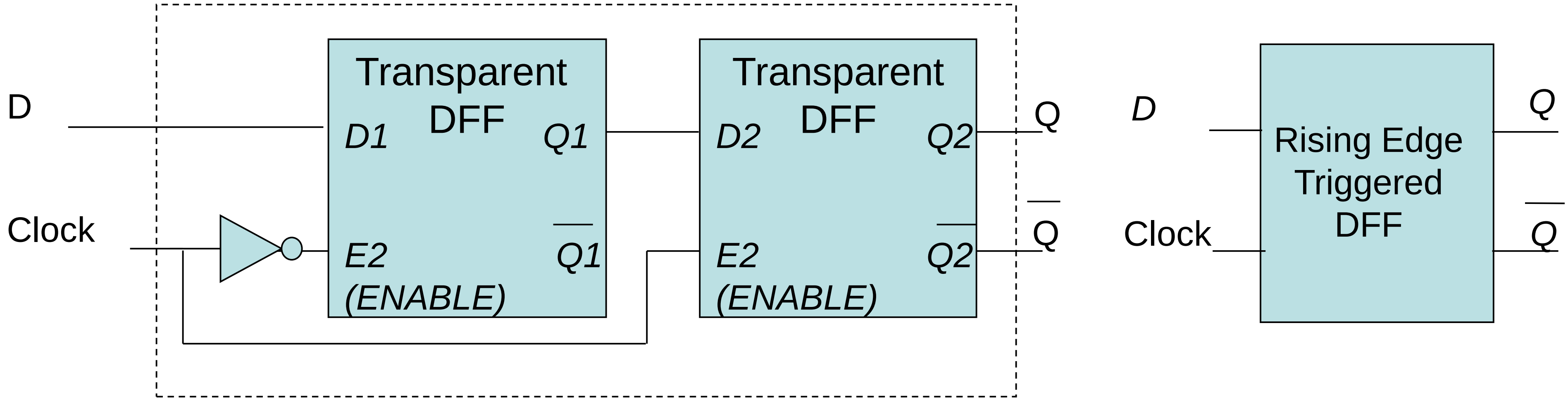
- Transparent D flip-flops require their D inputs all remain constant while clock (E) = 1
- This means that the preceding logic circuits cannot start computing the next operation while clock =1
- Example of problem usage – a D flip-flop Q’ output is feedback to the input



- Want $Q(n) = \overline{Q}(n-1)$
- But the output will oscillate between 0 and 1

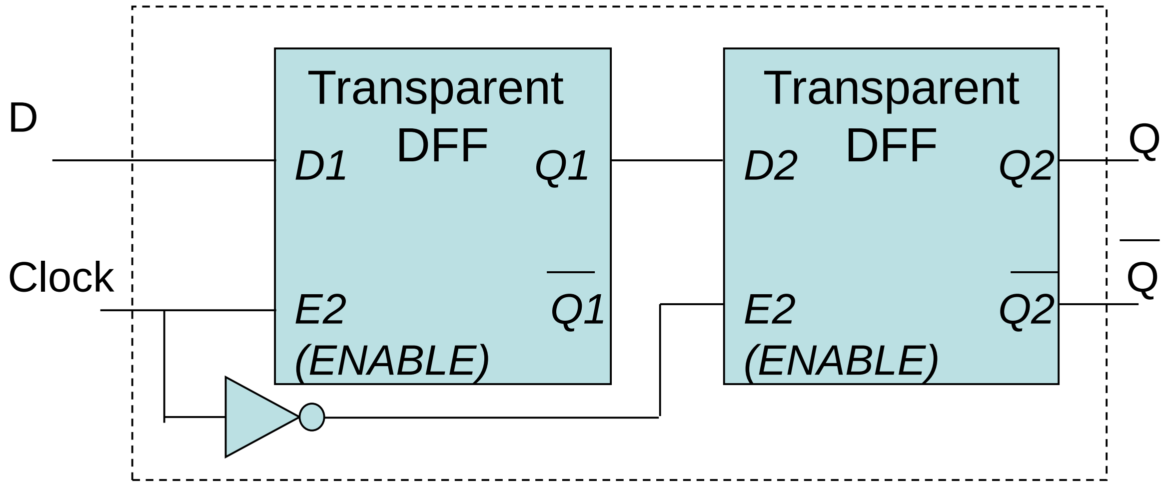
MASTER-SLAVE D FLIP-FLOP

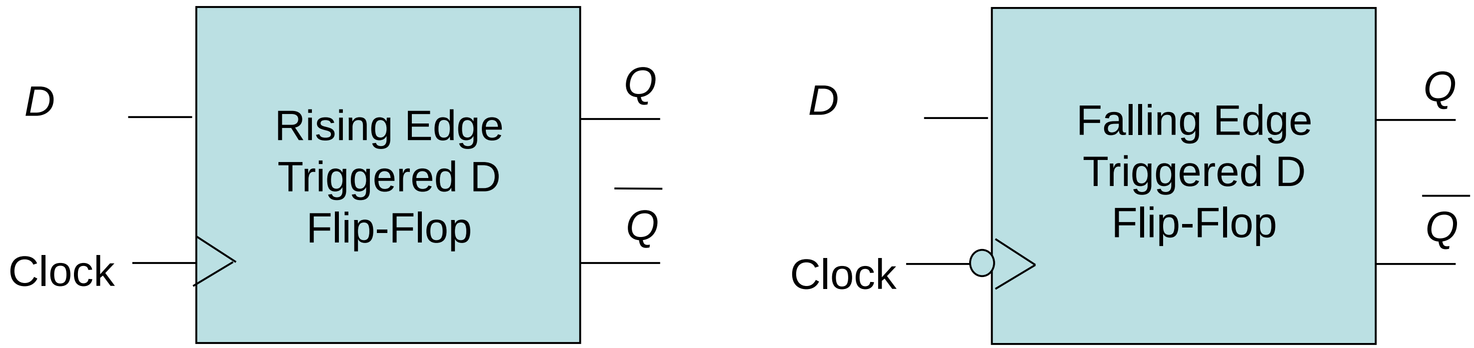
- We can fix this by using two separate transparent D flip-flops cascading together
- We call the first flip-flop a MASTER and the second one a SLAVE, and hence we call this new memory element a MASTER-SLAVE D flip-flop



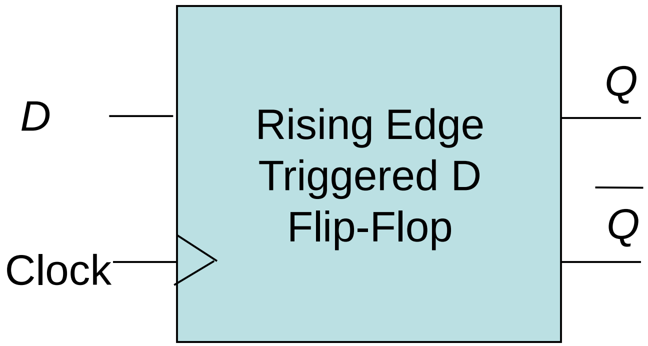
FALLING EDGE TRIGGER MASTER-SLAVE D FLIP-FLOP

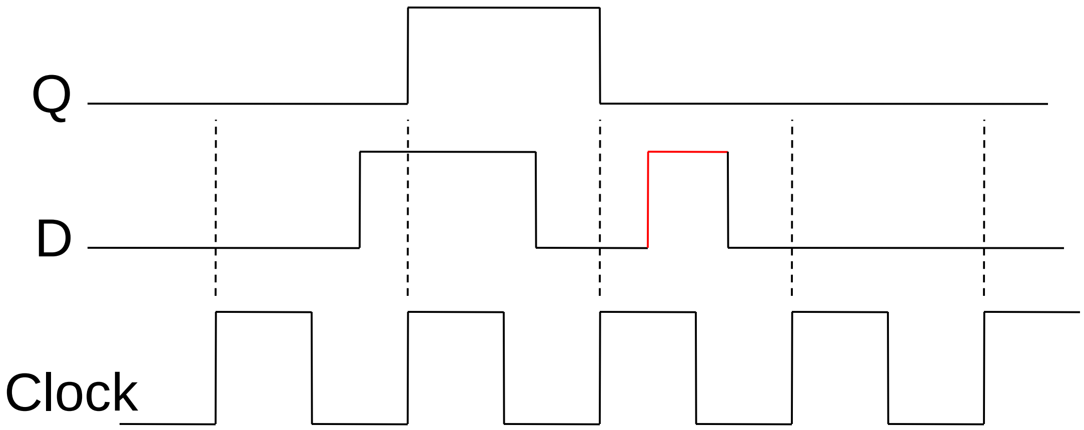
- The inverter can be connected to the slave flip-flop to provide a falling edge triggered Master-Slave D flip-flop

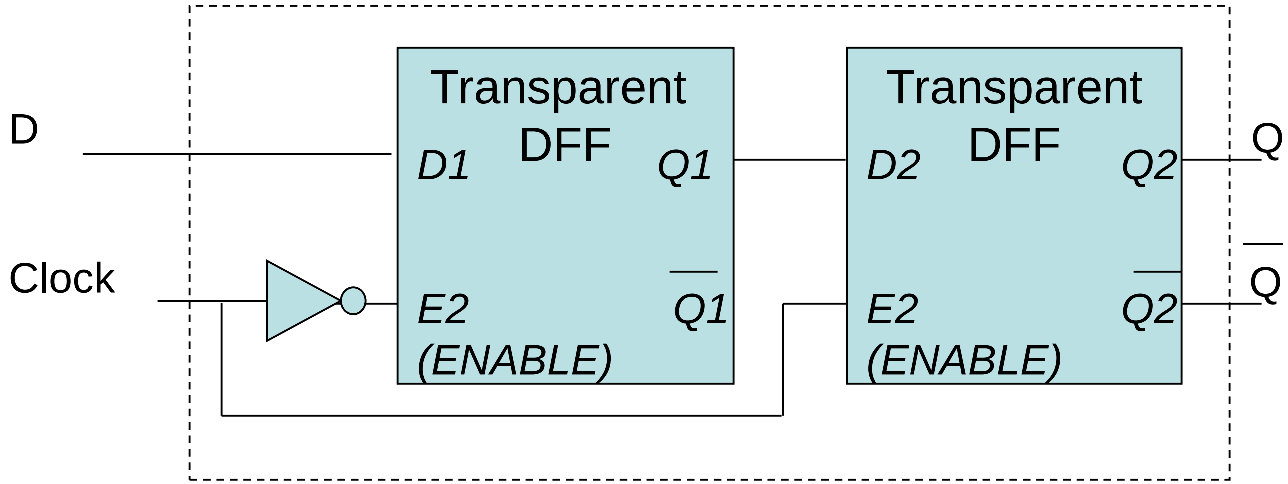




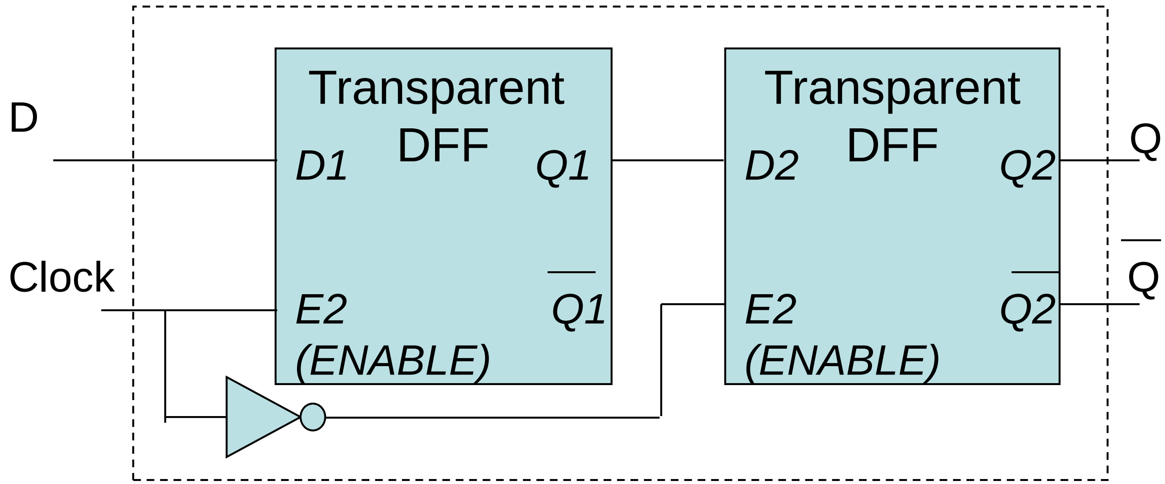
TIMING DIAGRAM OF MASTER-SLAVE D FLIP-FLOP

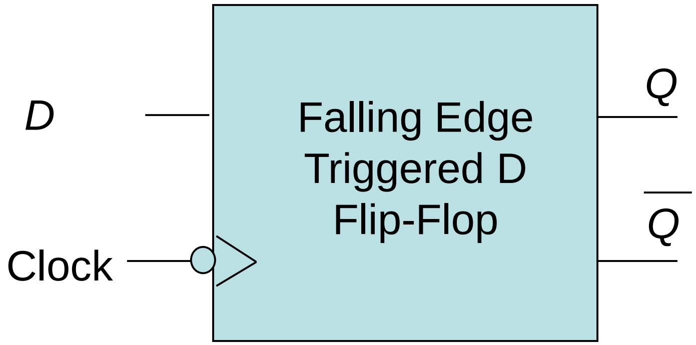


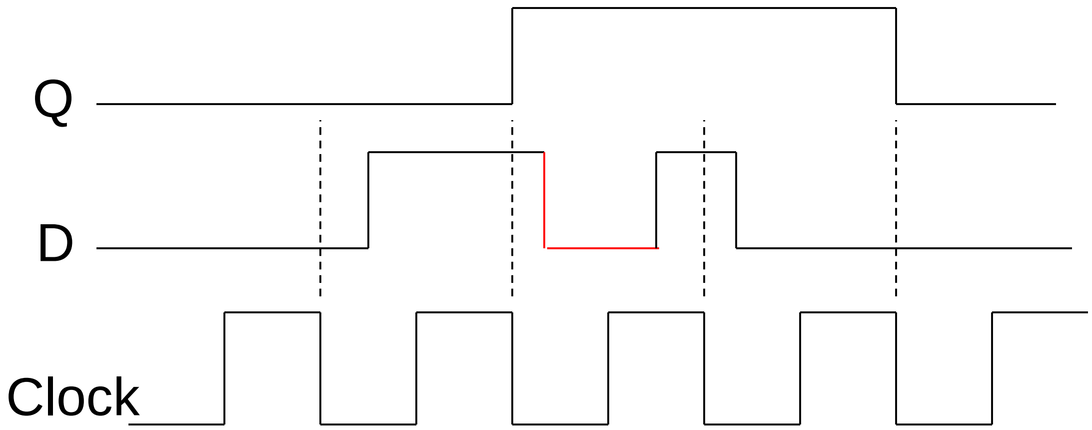




TIMING DIAGRAM OF MASTER-SLAVE D FLIP-FLOP

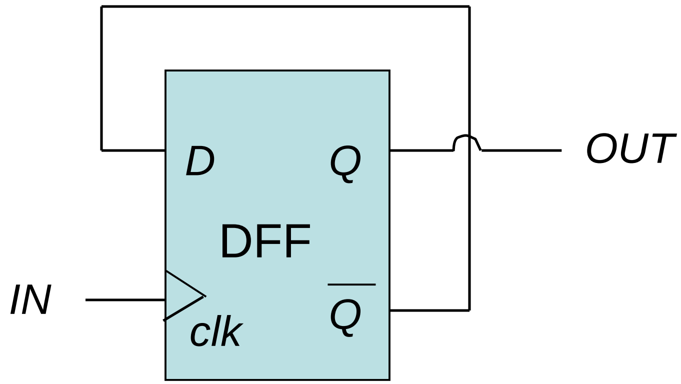


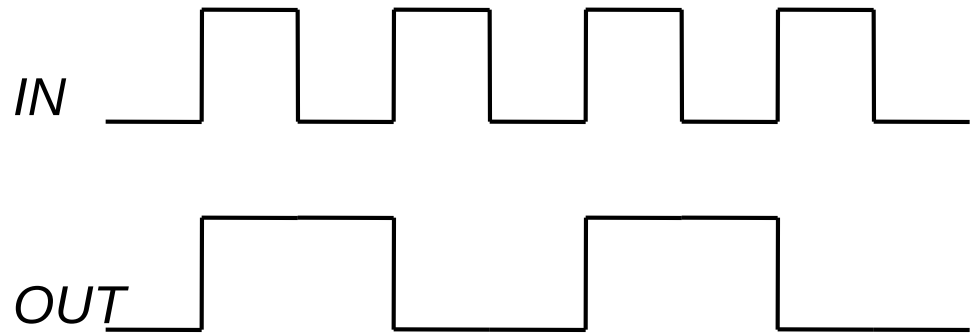




Department of Electronic and Computer Engineering, The Hong Kong University of Science & Technology

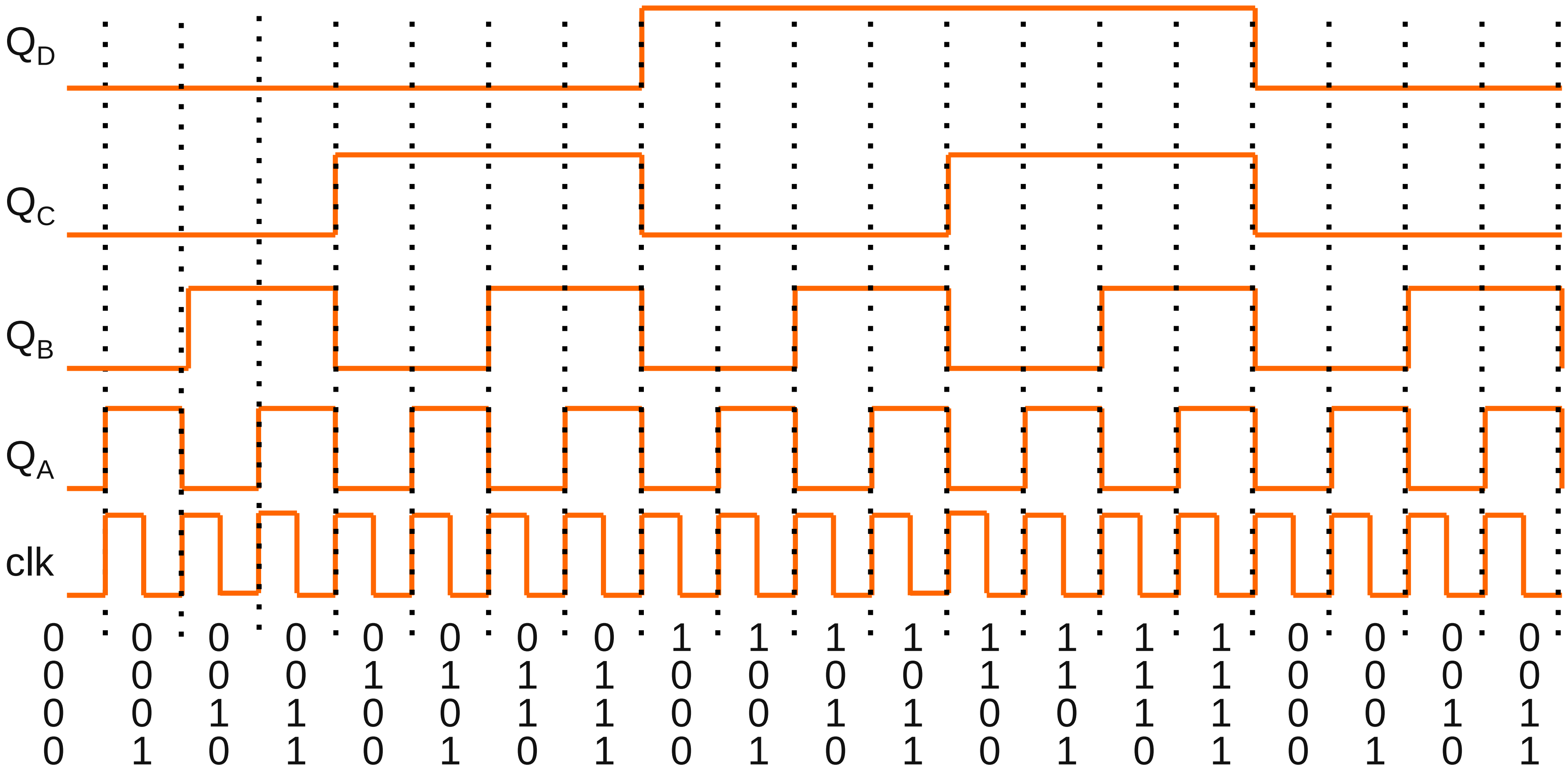
- D Flip-Flop can be used to form a frequency divider





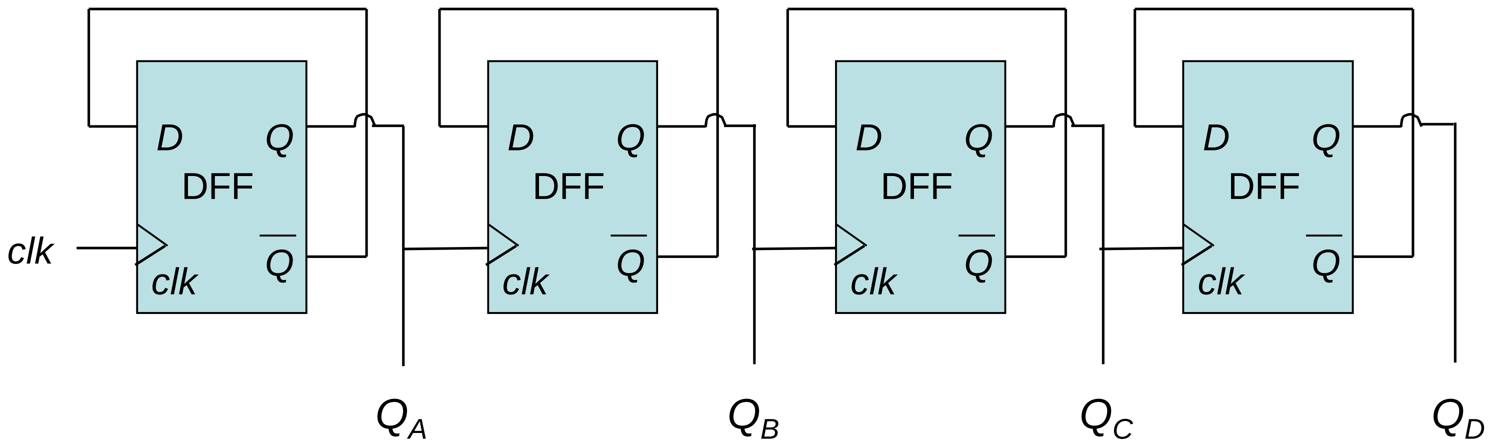
- The frequency at the output of the circuit becomes half of that of the input

❖ Reminder: the timing diagram of a binary counter 74HC163



CONSTRUCTING BINARY COUNTER

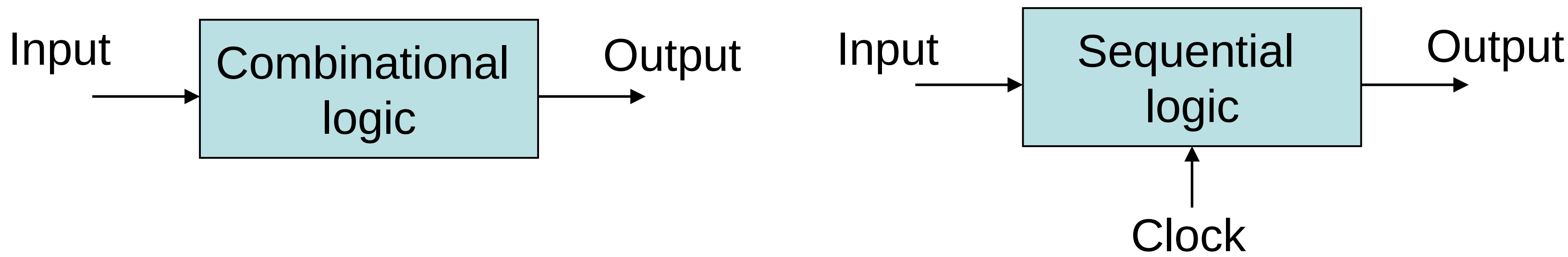
- Binary counters can be constructed using Master-Slave DFFs



- Q: what's wrong with the above configuration?
- A: Last three clocks should be connected to $\overline{Q}$ instead of Q
- Each flip-flop divides the clock frequency of the input from the previous flip-flop/ clock

COMBINATIONAL AND SEQUENTIAL LOGIC

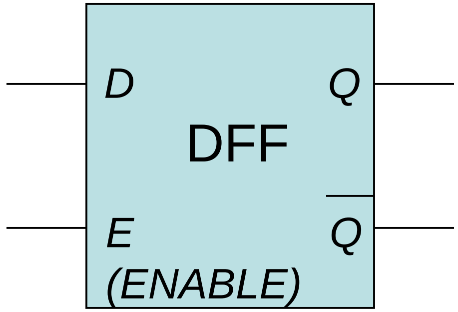
- Assume all logics do not have delay
- Combinational logic: Output changes as soon as inputs change



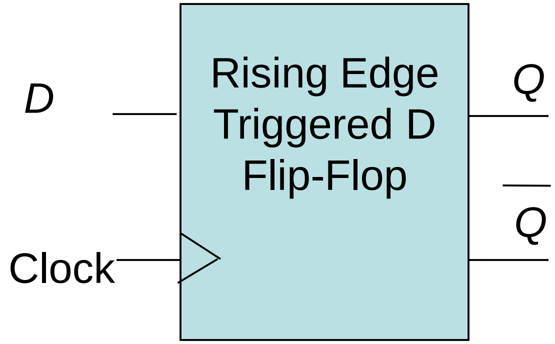
- Sequential logic: Output may change only at a specific time, depending on the clock and the type of flip-flop
  - Edge-triggered flip-flop – only changes at the clock edge
  - Transparent flip-flop – changes with the input during the enable clock phase

SUMMARY

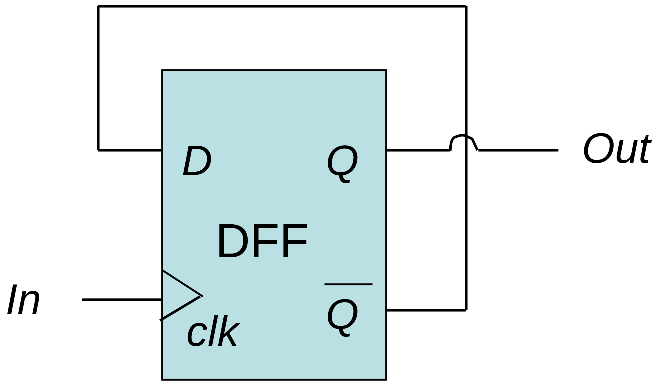
❖ Transparent D Flip-Flop



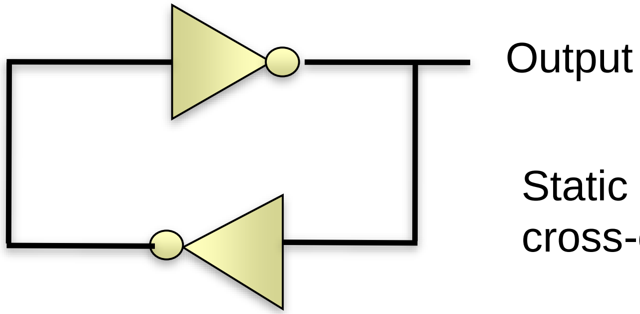
❖ Edge Triggered D Flip-Flop



❖ Frequency Divider and Counter

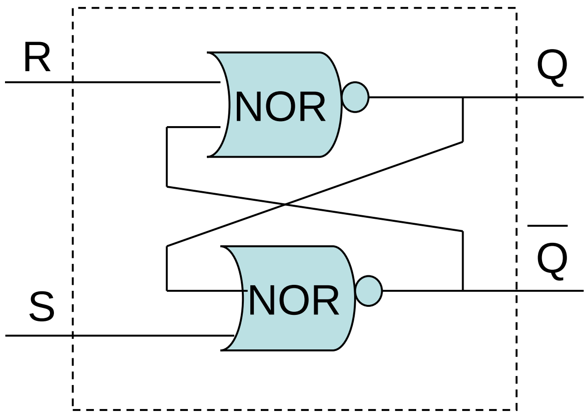


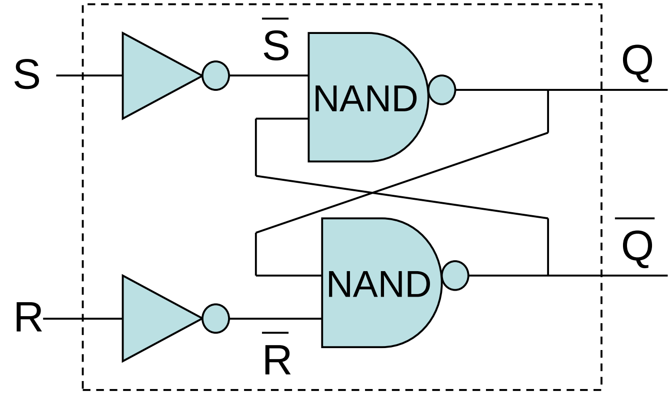
❖ Basic structure of a flip-flop



Static memories are based on this cross-coupled feedback inverter pair

❖ Logic gate implementation

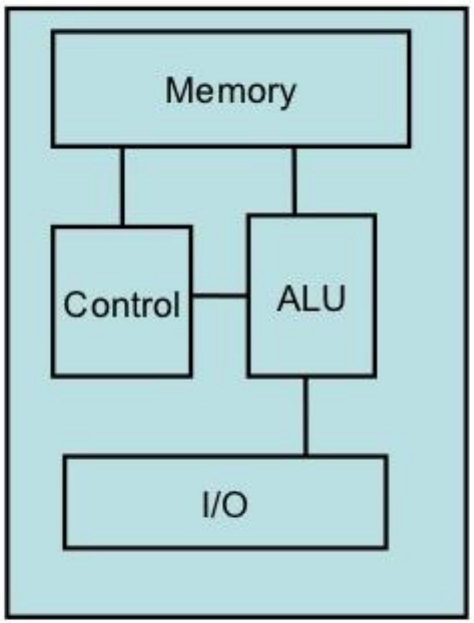




COMBINATIONAL AND SEQUENTIAL LOGIC

- We have learned how to use logic gates to build a circuit to complete any given functions where the outputs only depend on the inputs, i.e., combinational logic.
- We also know how to build memory units.
- By combining combinational logic and memory units, we can create sequential logic, which handles problems where the outputs not only depend on the current input but the history.

- Using logic gates, we can design different functions modules, including ALU (Arithmetic Logic Unit) and memory units etc.
- Putting them together, we can have a simple processor unit.

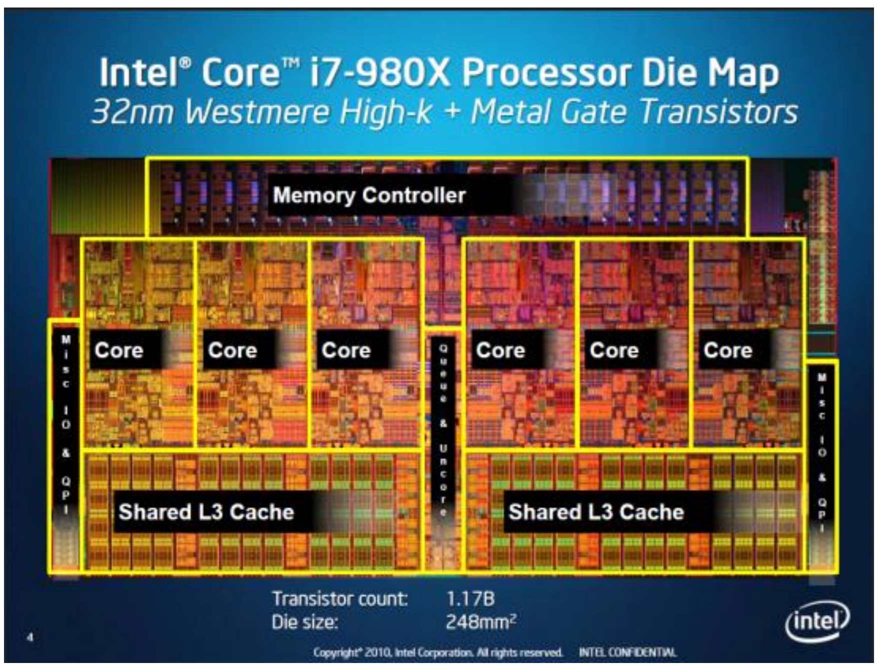


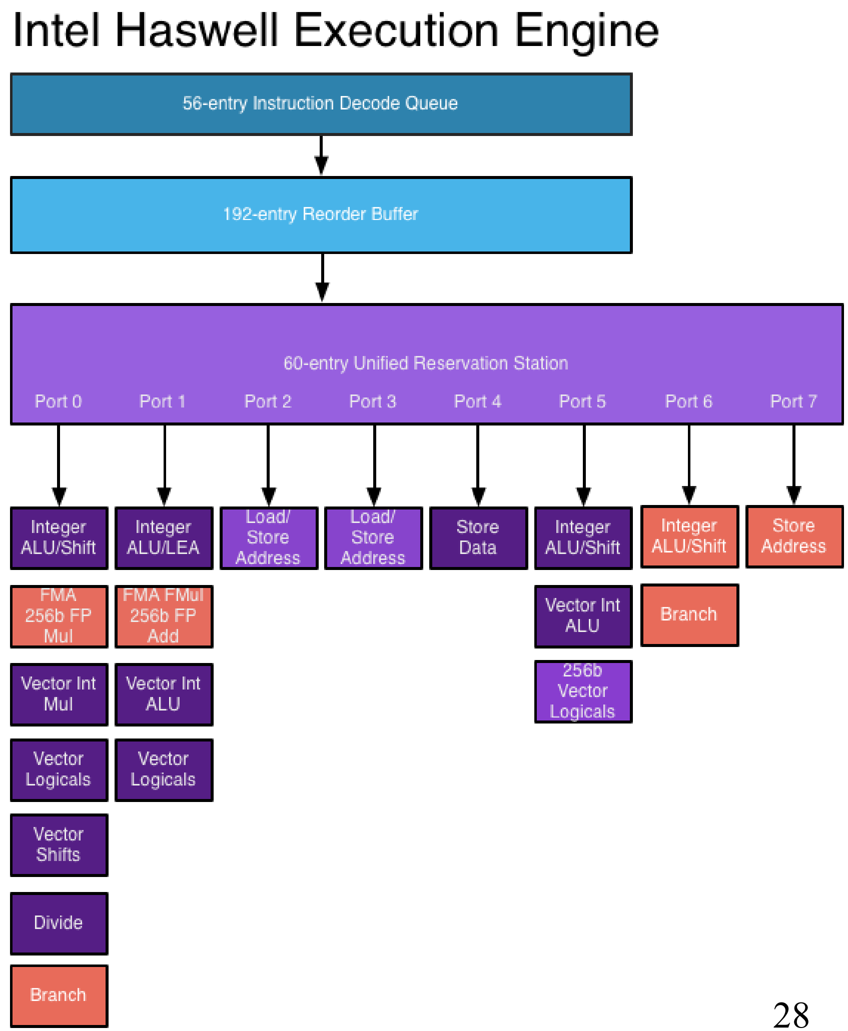
Von Neumann architecture.

Data and instructions are stored in memory, the Control Unit takes instructions and controls data manipulation in the Arithmetic Logic Unit.

Input/Output is needed to make the machine a practicality

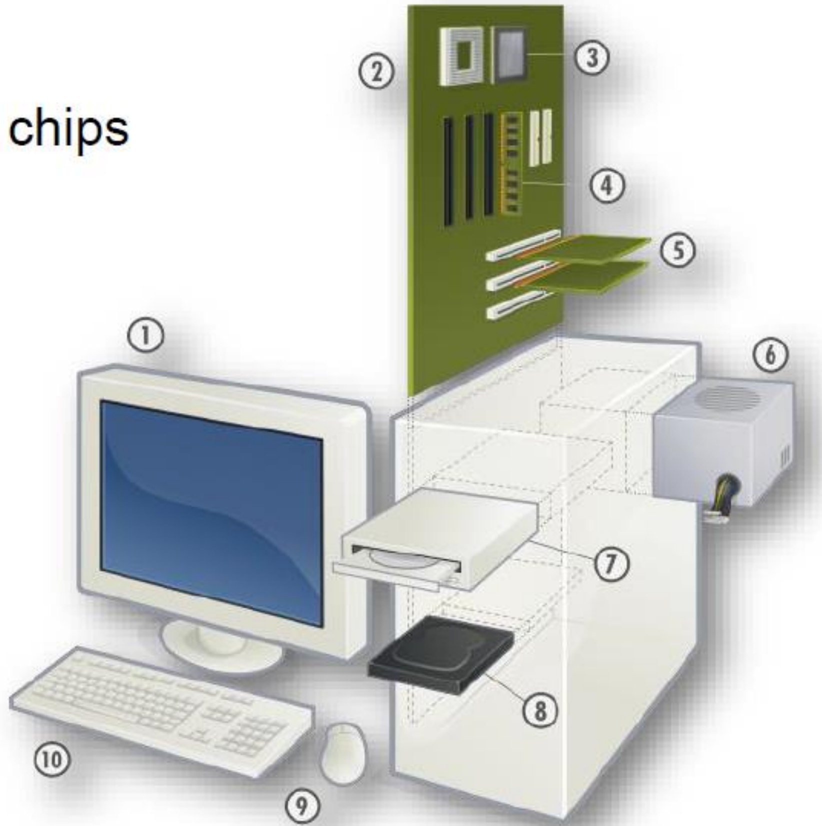
DESKTOP PROCESSOR



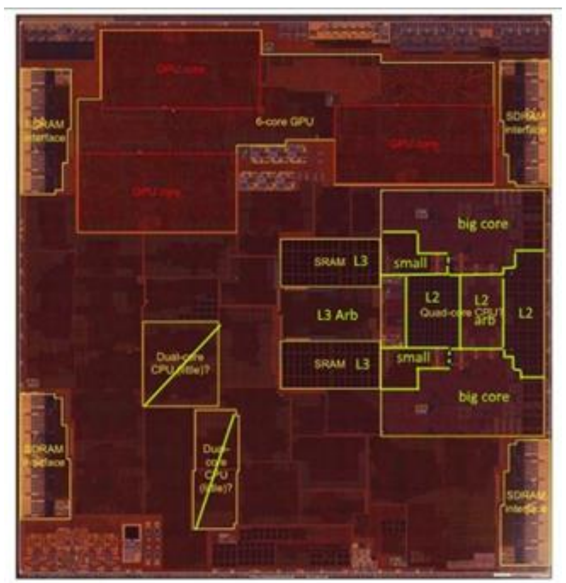


DESKTOP COMPUTERS

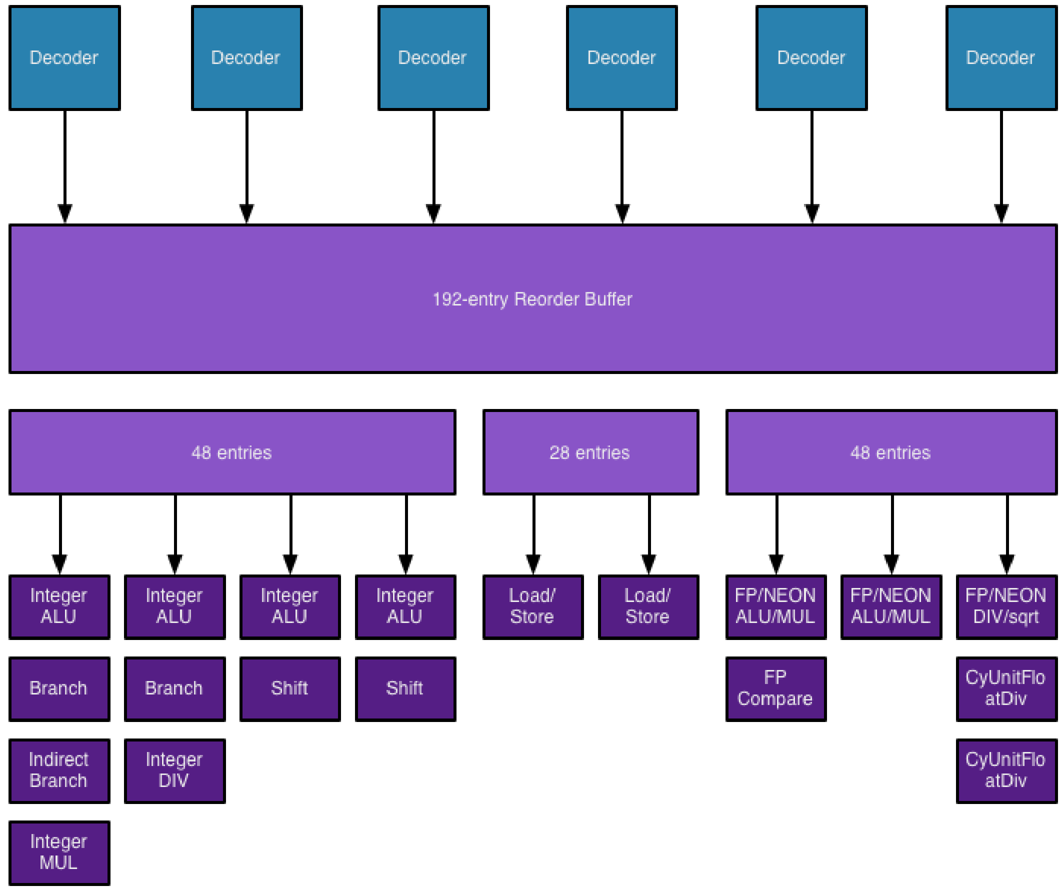
- Components
  - (1) Monitor
  - (2) PCB (printed circuit board) and chips
  - (3) CPU (central processing unit)
  - (4) Memory
  - (5) Sound/network/video cards
  - (6) Power supply
  - (7) CD/DVD drive
  - (8) Hard drive
  - (9) Mouse
  - (10) Keyboard
- Component types
  - Processor (3)
  - Memory (4)
  - Interconnection and input/output (I/O) device (1),(2),(5),(7),(8),(9),(10)



SPECIAL PURPOSE PROCESSOR: EMBEDDED



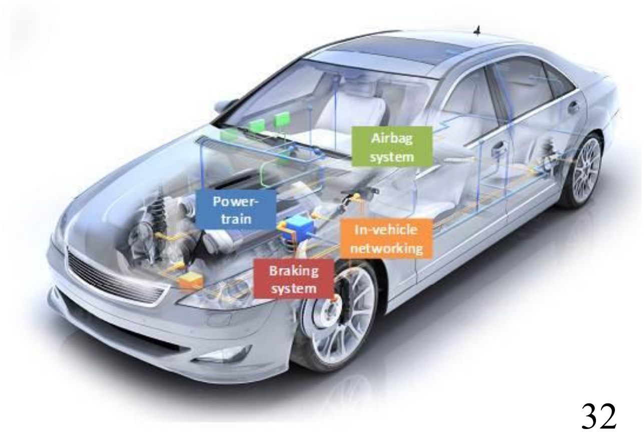
Apple Cyclone



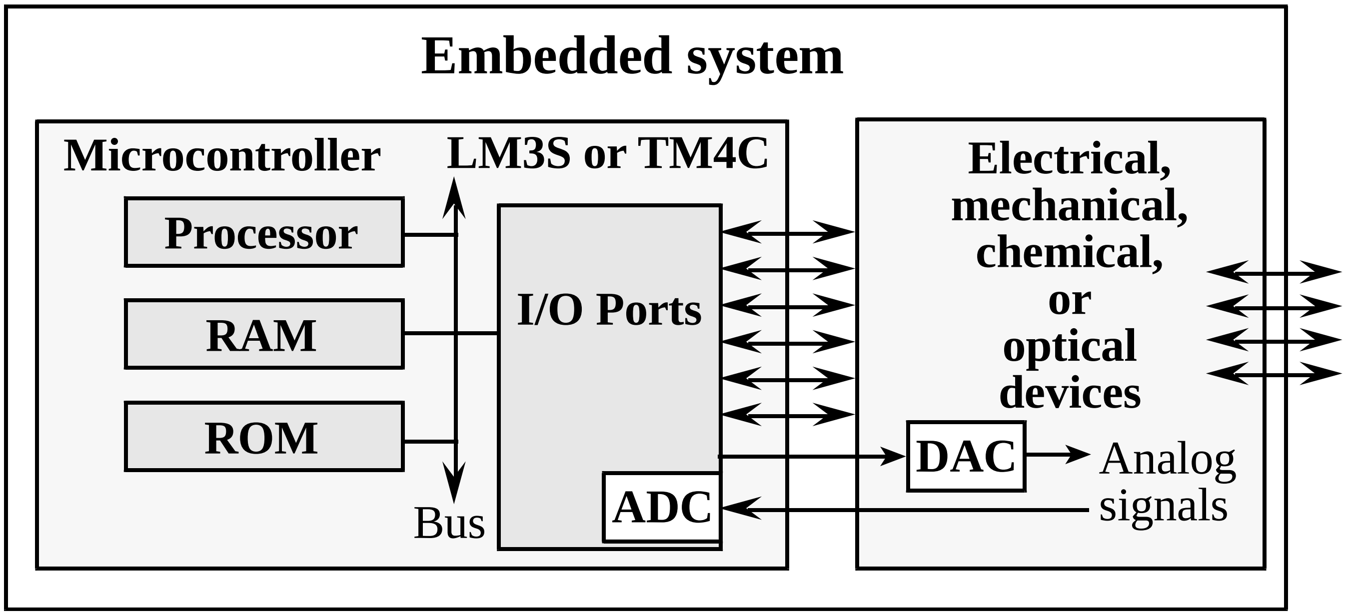
EMBEDDED COMPUTERS

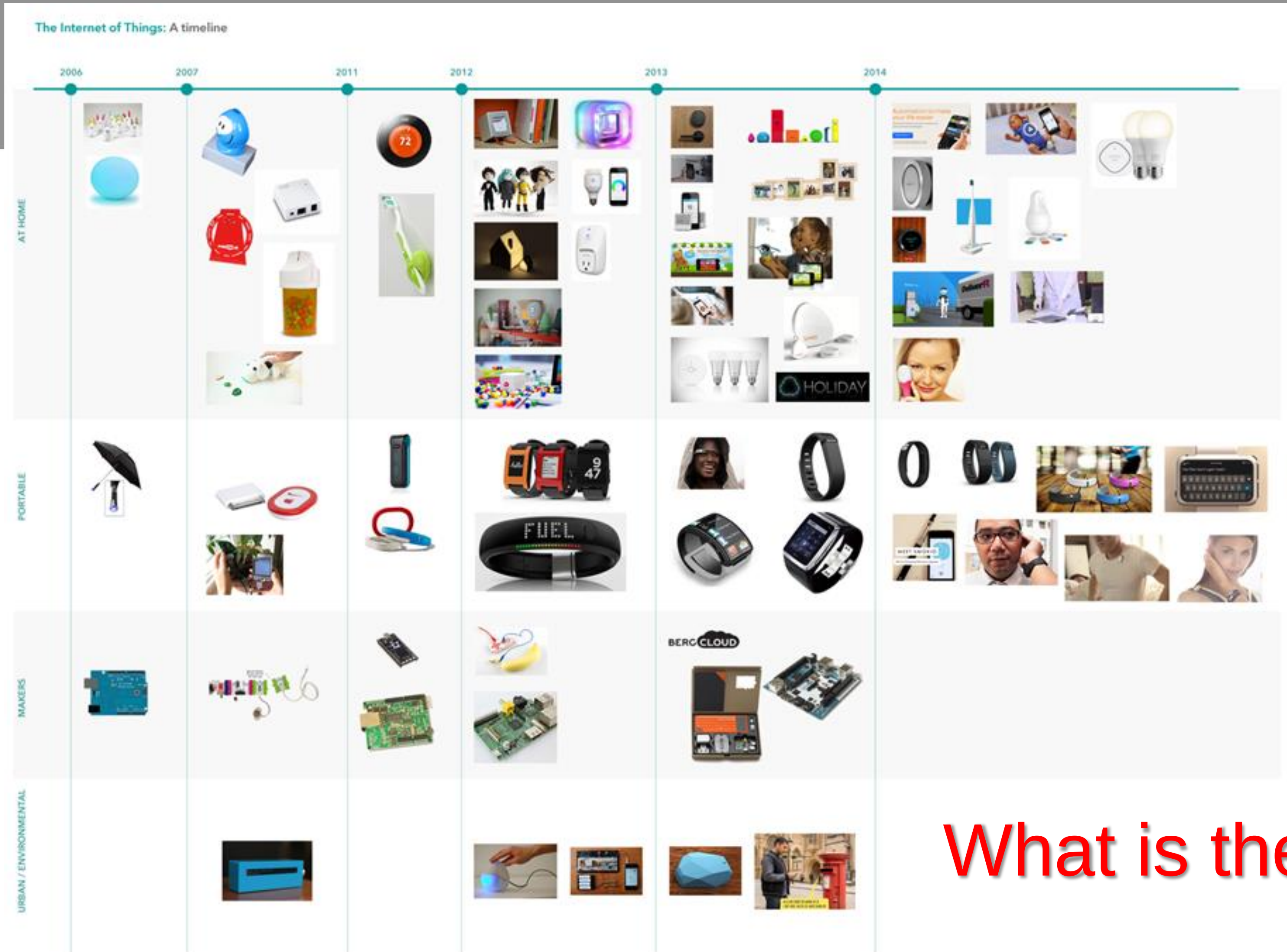
What is *Embedded System?*

- An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints.
- Any sort of device which includes a programmable computer but itself is not intended to be a general-purpose computer - Marilyn Wolf.
- Like a regular computer system, it includes
  - Hardware components
  - Software components



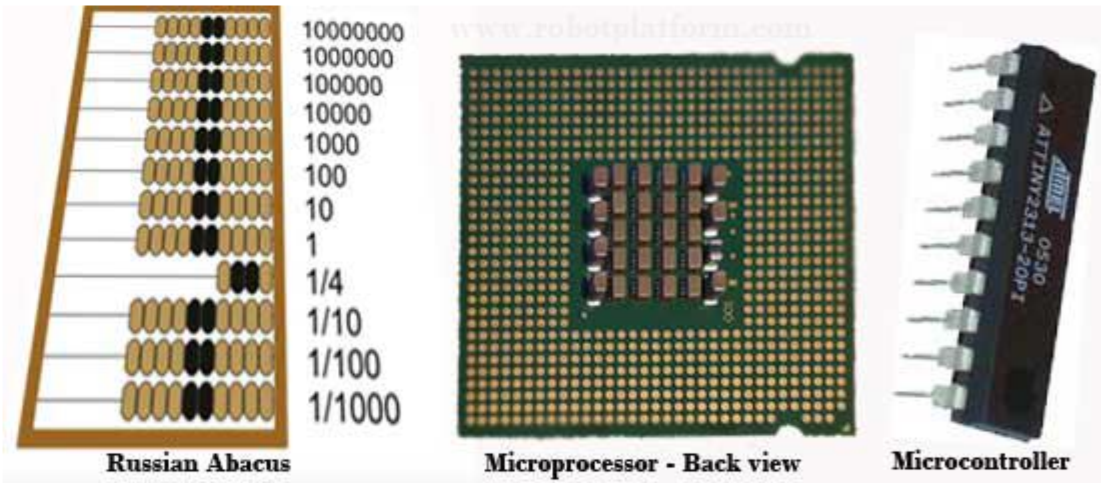
EMBEDDED SYSTEM DIAGRAM





What is the future?

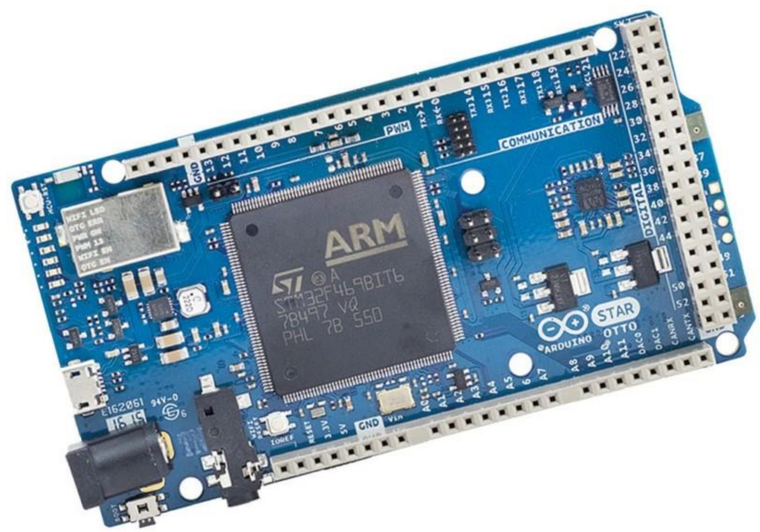
MICRO-CONTROLLER UNIT (MCU)



MICRO-PROCESSOR

- To control the processing and execution of instructions inside a computer there rests a processor or a central processing unit (CPU) which contains an arithmetic logic unit (ALU) and a Control Unit (CU).
- A microprocessor is one such general purpose CPU.
- Additional components like RAM, ROM, internal circuitry, data bus, and other peripheral devices are added to make it a computer.

- A highly integrated chip in which most or all components needed for a controller like a CPU, RAM, ROM, timers, Input and Output pins, registers, clock circuit etc. are built within.
- They are small, powerful with limited speed and memory which can be used in embedded applications and for specific tasks.

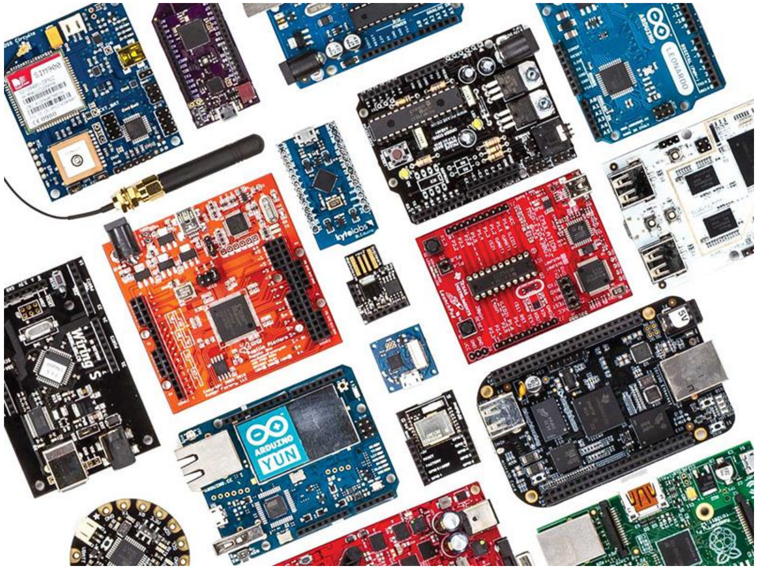


MICRO-CONTROLLER

- Microcontrollers are normally embedded within other devices so that they can control the actions of those devices. Typically a microcontroller is used for three basic purposes:
  - Receive input (via sensors, human intervention etc.,)
  - Store and Process this input into a set of actions
  - Apply the processed data for some other actions which goes as an output.

- Arduino is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects.

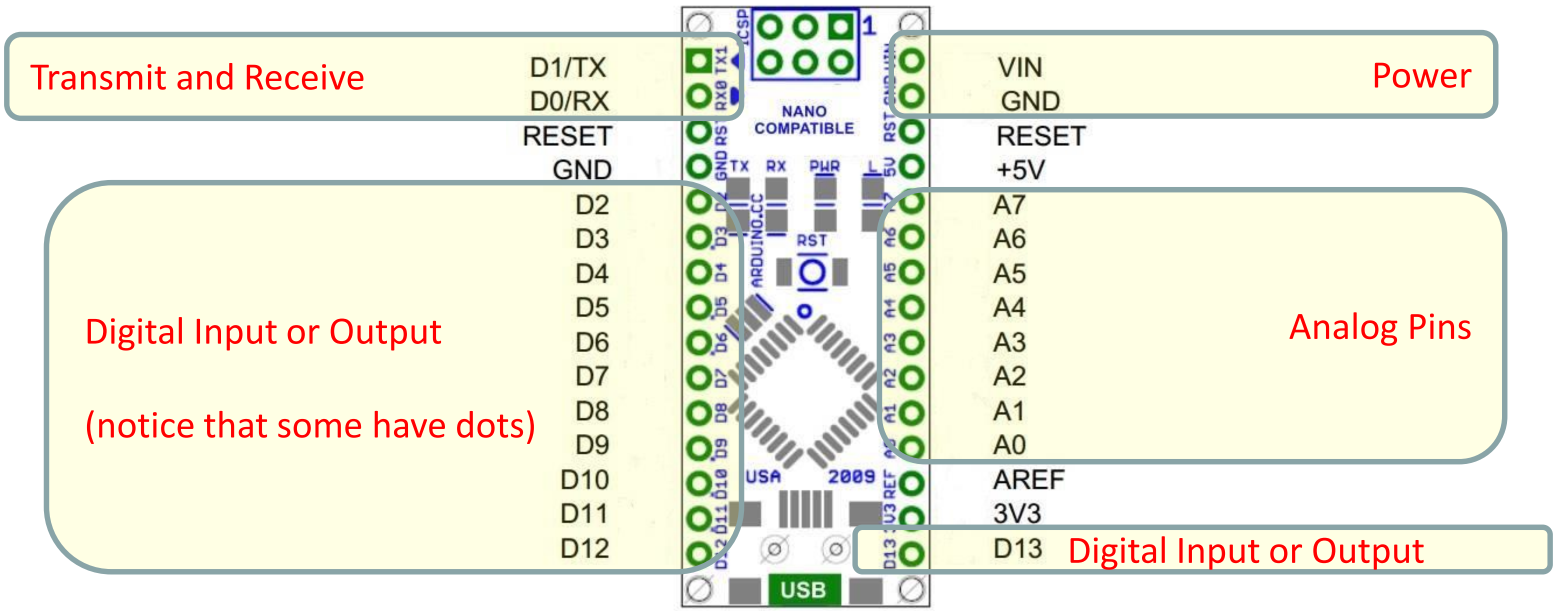
- The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone.



ARDUINO NANO

Microcontroller	ATmega328
Architecture	AVR
Operating Voltage	5 V
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
Clock Speed	16 MHz
Analog IN Pins	8
EEPROM	1 KB
DC Current per I/O Pins	40 mA (I/O Pins)
Input Voltage	7-12 V
Digital I/O Pins	22 (6 of which are PWM)
PWM Output	6
Power Consumption	19 mA
PCB Size	18 x 45 mm
Weight	7 g
Product Code	A000005

Nano Board


Transmit and Receive
D1/TX
D0/RX
RESET
GND
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
Digital Input or Output
(notice that some have dots)
VIN
GND
RESET
+5V
A7
A6
A5
A4
A3
A2
A1
A0
AREF
3V3
D13
Power
Analog Pins
Digital Input or Output
NANO COMPATIBLE
USB

Power

- The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply, or 5V regulated external power supply.
- The power source is automatically selected to the highest voltage source.

Memory

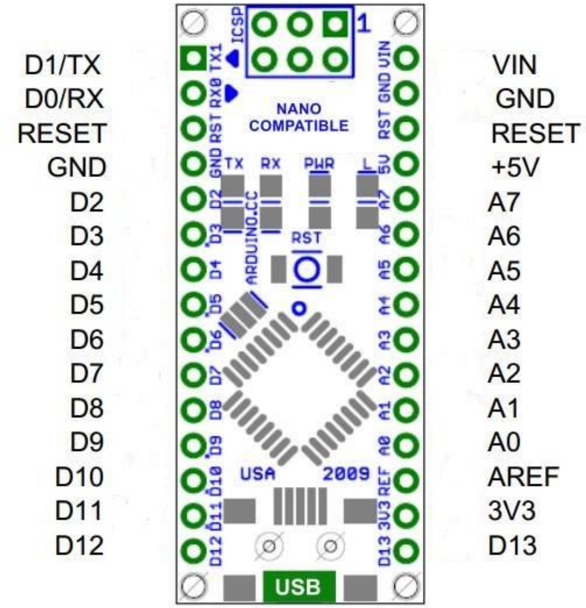
- The ATmega168 has 16 KB of flash memory for storing code (of which 2 KB is used for the bootloader); The ATmega328 has 32 KB, (also with 2 KB used for the bootloader).
- The ATmega168 has 1 KB of SRAM and 512 bytes of EEPROM (which can be read and written with the EEPROM library); the ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

ARDUINO NANO: INPUT AND OUTPUT

- Each of the 14 digital pins (D0-D13) on the Nano can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions.

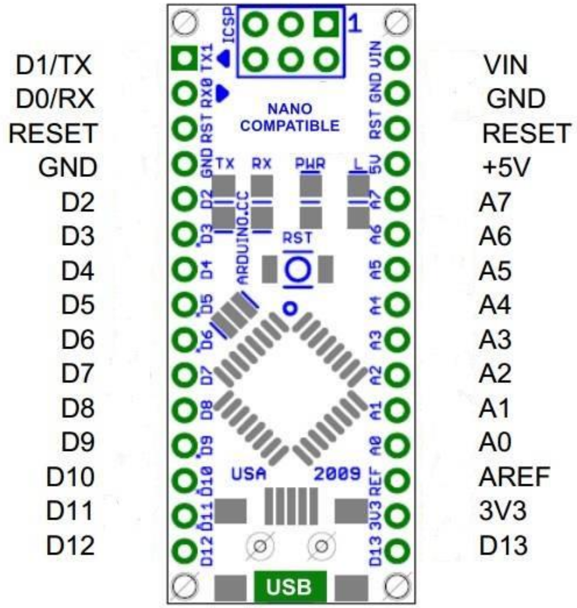
- They operate at 5 volts.

- Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms.



❖ In addition, some pins have specialized functions

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.



A Sample Program

```
// System setup: Run when we start or reset the Ardunio board:
void setup()
{
        pinMode(13, OUTPUT);          // initialize the digital pin as an output.
}

//  Control the Ardunio board
void loop()                                    // the loop routine runs over and over again forever:
{
        digitalWrite(13, HIGH);       // turn the LED on (HIGH/LOW are the voltage levels)

        delay(1000);                  // wait for a second

        digitalWrite(13, LOW);        // turn the LED off by making the voltage LOW

        delay(1000);                   // wait for a second
}
```

NEXT LECTURE

- Arduino hardware & software
- Programming Language

Questions ?!