Computer Architecture. Week 5

Muhammad Fahim, Alexander Tormasov

Innopolis University

m.fahim@innopolis.ru
a.tormasov@innopolis.ru

October 1, 2020

Topic of the lecture

- Hardware Building Blocks (Sequential Logic)

Topic of the tutorial

- Yuri Panchul will deliver the tutorial on FPGA Boards.

Topic of the lab

- Multiplexer / Demultiplexer / Selector
- Sequential Logic

Content of the class

- State Elements
- What are Clocks?
- Edge-Triggered Methodology
- Latches – SR Latches, D Latches
- Flip-Flops
- Memory Organization
- Memory Chips
- Field-Programmable Gate Arrays
- CPU Chip Control Pins

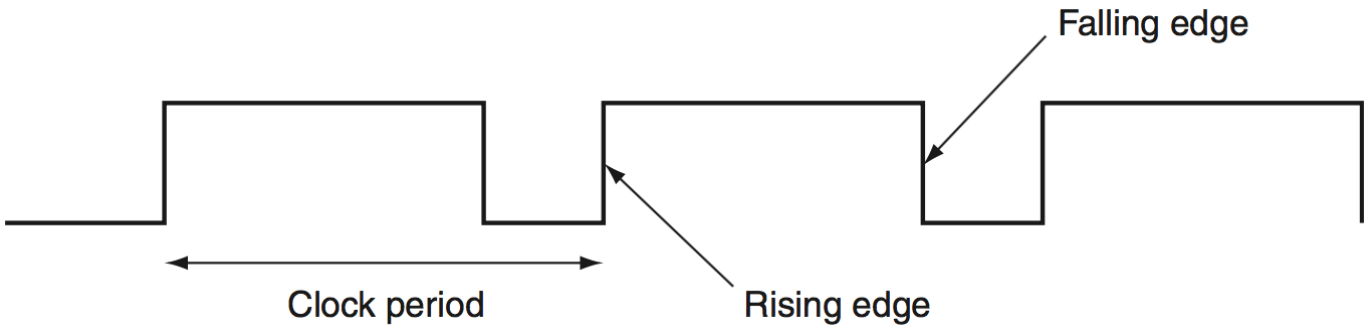
State Elements

- **State Element**
  A memory element that contains a value is the state of the element.

- Clocking is "used" to update state elements.

Clock

- A clock is simply a free-running signal with a fixed cycle time.
- A clock signal oscillates between high and low values.
- The clock period is the time for one full cycle.

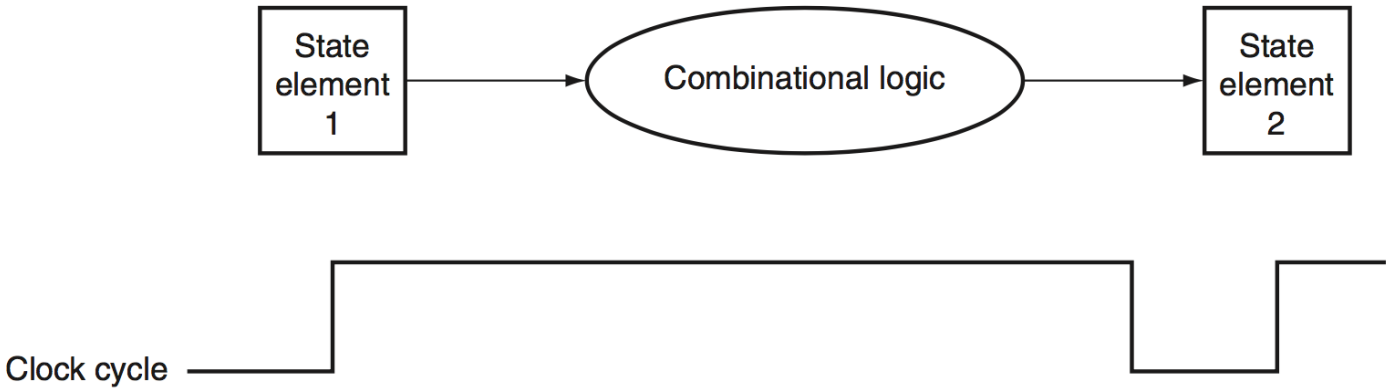


Clock Generator

- A clock signal is produced by a clock generator.
- A clock generator is a type of circuit that produces a continuous, synchronized electrical signal for timing purposes in a wide variety of devices.
- **Why we need clock?**
  - A circuit is required to produce a timing signal to synchronize circuit's operation.

Edge-Triggered Methodology

- Edge-triggered methodology is a clocking scheme in which all state changes occur on a clock edge.



- The inputs to a combinational logic block come from a state element, and the outputs are written into a state element.
- In an edge-triggered design, either the rising or falling edge of the clock is active and causes state to be changed.

Introduction to Sequential Logic

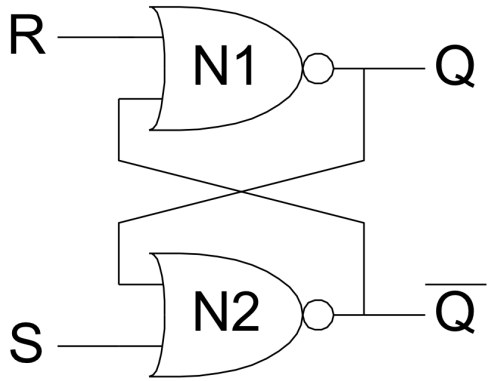
- So far, our circuits have been converting inputs to outputs without storing any data.
- To do more advanced things (e.g., to build computers) we need components that can store data.
- **State elements store state**
  - SR Latch
  - D Latch
  - D Flip-flop

Latch

- Latch is a circuit used to store information.
- It can retain an input signal until reset by another signal

SR (Set/Reset) Latch

- **SR Latch:** Cross-coupled NOR gates. It can set (S=1, R=0) or reset (R=1, S=0) the output



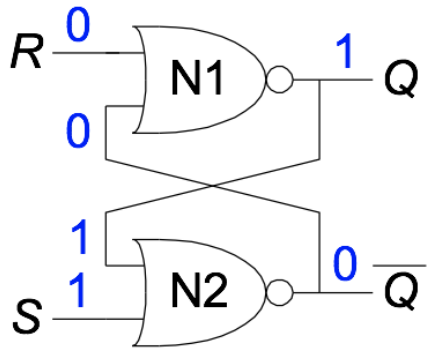
- Consider the four possible cases

$$S = 1, R = 0$$
$$S = 0, R = 1$$
$$S = 0, R = 0$$
$$S = 1, R = 1$$

SR Latch Analysis (1/2)

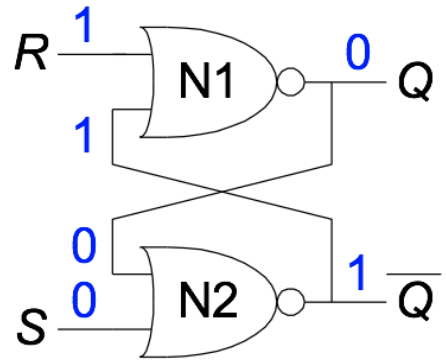
$$S = 1, R = 0 :$$

$$then\ Q = 1\ and\ \overline{Q} = 0$$



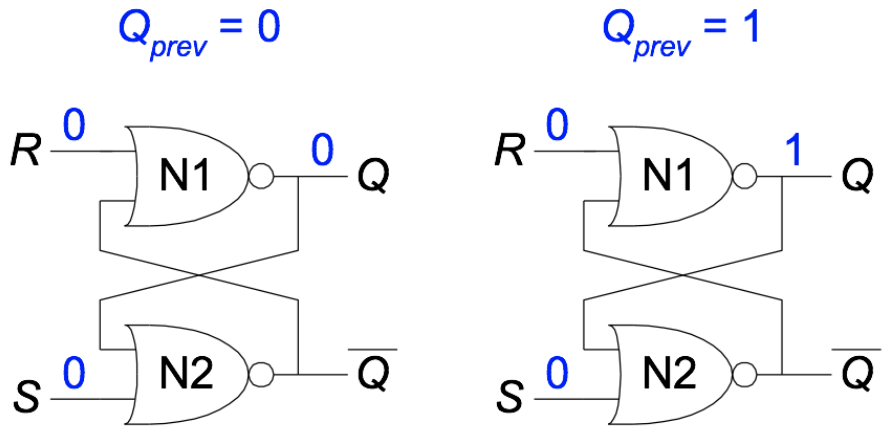
$$S = 0, R = 1 :$$

$$then\ \overline{Q} = 1\ and\ Q = 0$$

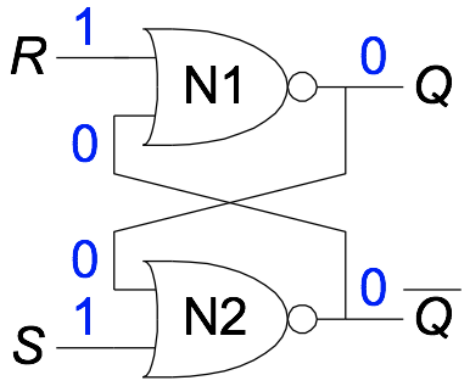


SR Latch Analysis (2/2)

$$S = 0, R = 0 :$$
$$then\ Q = Q_{prev}$$



$$S = 1, R = 1 :$$
$$then\ \overline{Q} = 0\ and\ Q = 0$$
Invalid State

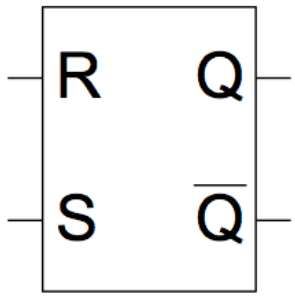


SR Latch - Summary

- If S and R are 0, Q remains unchanged
- If I want Q to be 1, then, I turn S to 1 leaving R 0
- If I want Q to be 0, then I turn R to 1 leaving S 0
- When S and R are both 1, the state is inconsistent

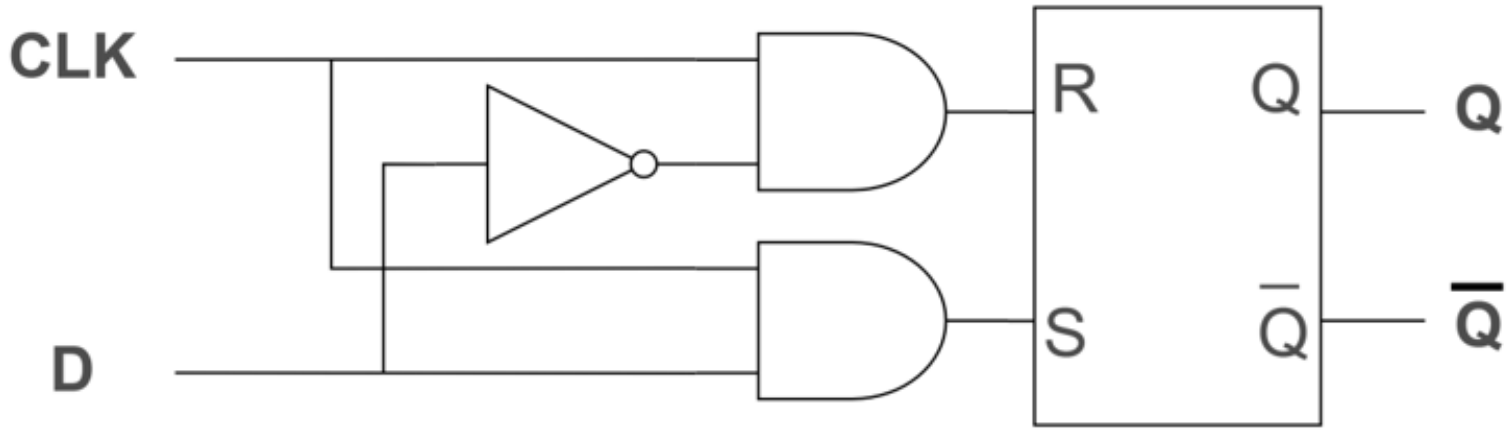
SR Latch Symbol

- Stores one bit of state (Q)
- Control what value is being stored with S, R inputs
- **Set:** Make the output 1
  $$(S = 1, R = 0, Q = 1)$$
- **Reset:** Make the output 0
  $$(S = 0, R = 1, Q = 0)$$



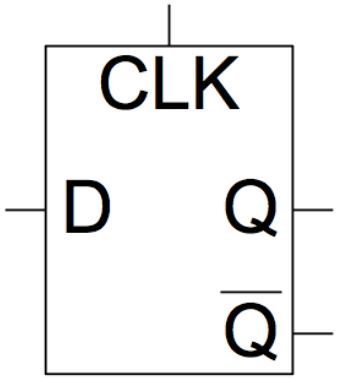
- Must do something to avoid invalid state (when $S = R = 1$)

D Latch



D Latch

- Two inputs: CLK, D
  - CLK: controls when the output changes
  - D: the data input
- Functions
  - When CLK = 1, D passes to Q (transparent)
  - When CLK = 0, Q holds its previous value (opaque)
- Note: It avoids invalid state case



Latches are Asynchronous

- **Latches are asynchronous**: the output changes very soon after the input changes.
- On the other hand, today, most computers are synchronous: the outputs of all sequential circuits change simultaneously to the rhythm of a global clock signal.

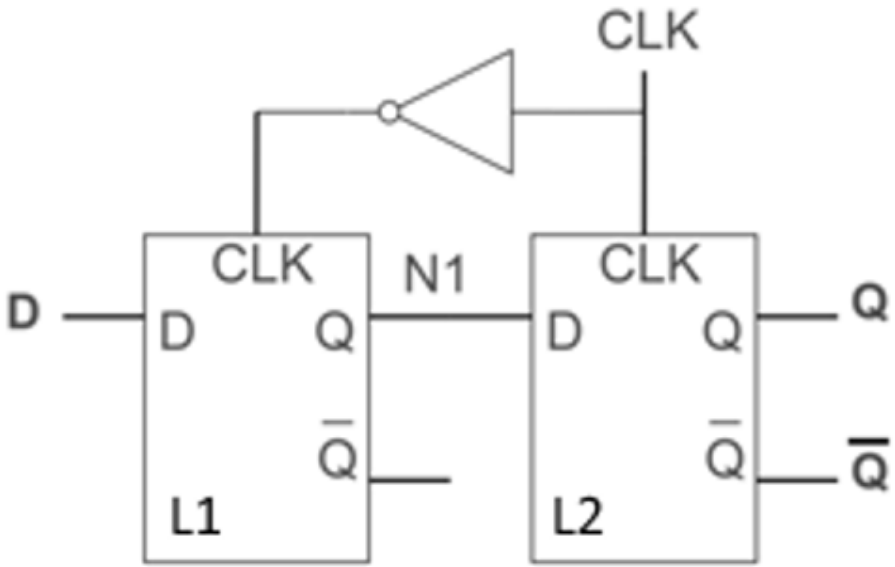
Flip Flops

- Flip flops are fundamental building blocks of digital electronics systems.
- A flip flop stores a single bit (binary digit) of data.
- One of its two states represents a “one” or “zero”.

Latches vs. Flip Flops

- The main differences are as follows:
- **Latches**
  - Latches are built from logic gates
  - Latch is level-sensitive. It changes the value as input change.
- **Flip Flops**
  - Flip flops are built from latches
  - Flip flops are edge-sensitive and change their states only when a control signal goes from high to low or low to high

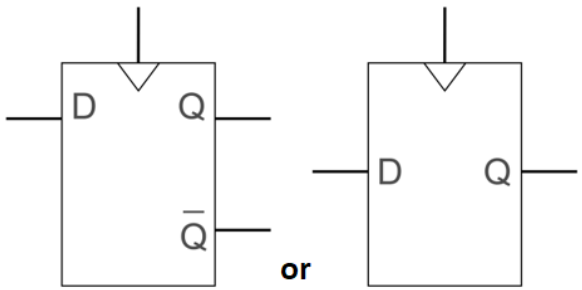
D Flip Flop



- It can be constructed from two cascaded D-latches, called the master and the slave.
- Both are controlled by one CLK signal.
- The latch labelled by L1 is called master, and the latch labelled by L2 is called slave.

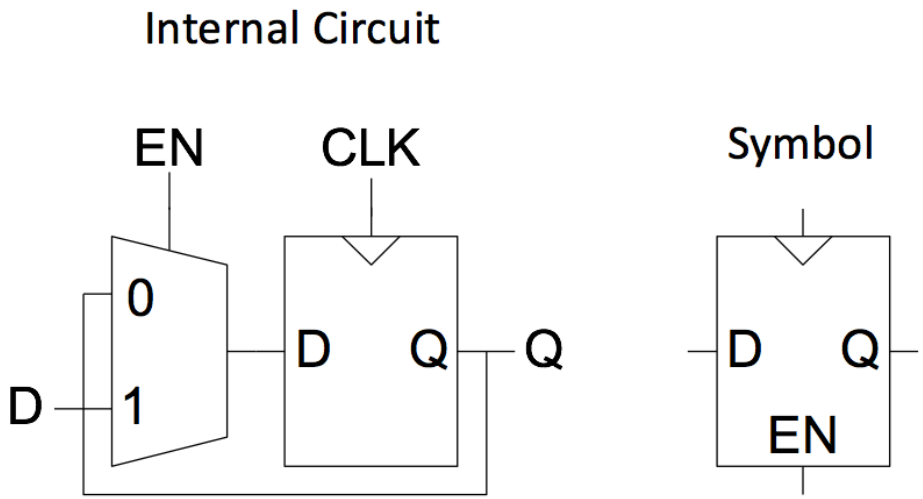
D Flip Flop

- **Inputs**: CLK, D
- **Functions**
  - Samples D on rising edge of CLK
    - When CLK rises from 0 to 1, it is stored at the internal state of D flip-flop and this state is propagated to the output Q
    - Otherwise, Q holds its previous value
  - Q changes only on rising edge of CLK
- Called edge-triggered
- Activated on the clock edge



Enabled Flip-Flop

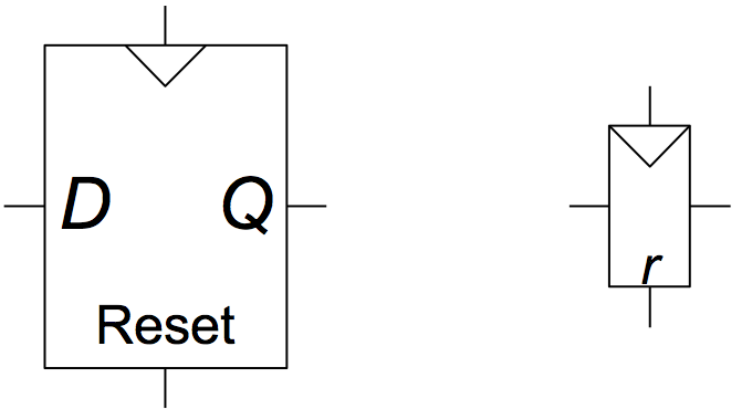
- **Inputs**: CLK, D, EN
  - The enable input (EN) controls when new data (D) is stored
- **Function**
  - EN = 1: D passes to Q on the clock edge
  - EN = 0: the flip-flop retains its previous state



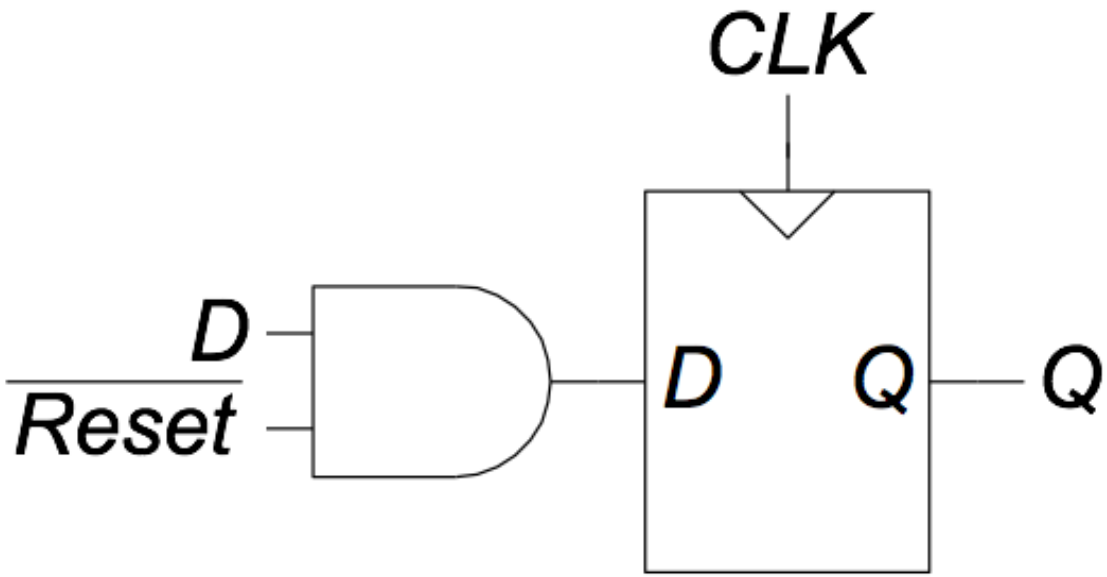
Why we need it? It enables signal to change the state of a flip-flop only at desired times and limits the power consumption.

Resettable Flip-Flop (1/2)

- Inputs: CLK, D, Reset
- Function:
  - Reset = 1: Q is forced to 0
  - Reset = 0: flip-flop behaves as ordinary D flip-flop

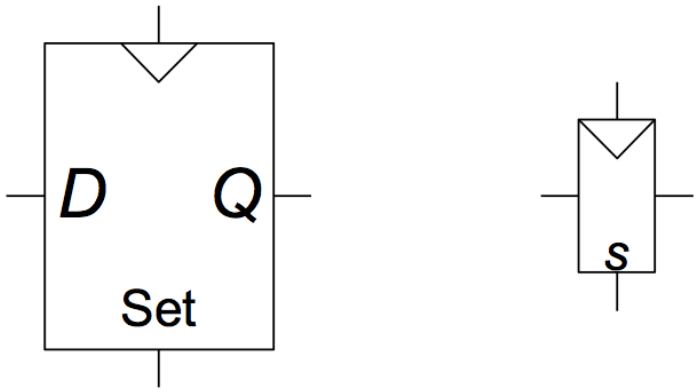


Resettable Flip-Flop (2/2)



Settable Flip-Flop

- Inputs: CLK, D, Set
- Function:
  - Set = 1: Q is set to 1
  - Set = 0: the flip-flop behaves as ordinary D flip-flop

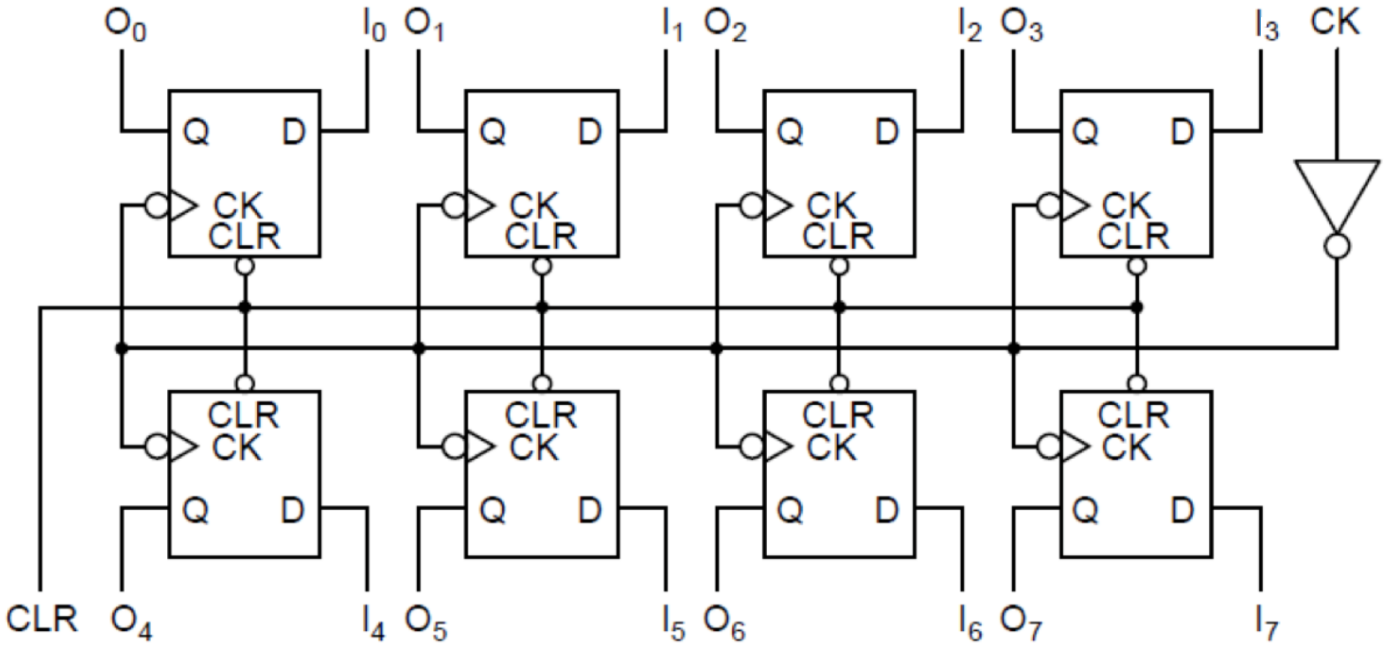


Observations on Resettable/settable Flip flop

- Resettable and settable flip flops are the same thing.
- They do not have any fundamental difference.
- The only distinguishable characteristic: depends on the design.

Registers

- An 8-bit register can be constructed from single-bit flip-flops.



Register Files (1/2)

- Register files are same as arrays in program.
- It is a place to store a piece of information.
- A register file consists of a set of registers that can be read and write by supplying a register number.
- A register file can be implemented with a decoder for each read or write port and an array of registers built from D flip flops.

Register Files (2/2)

- **Reading a register in a register file:** It does not change any state, we need only supply a register number as an input, and the only output will be the data contained in that register.

- **Writing a register in a register file:** we will need three inputs:
  1. a register number,
  2. the data to write, and
  3. a clock that controls the writing into the register.

Memory Elements: SRAMs and DRAMs

- Registers and register files provide the basic building blocks for small memories.
- The larger amounts of memory are built using either:
  - **SRAMs** (Static random access memories) or
  - **DRAMs** (Dynamic random access memories)

SRAM: Static Random Access Memories

- SRAM is built of transistors and flip-flops. It does not have the leakage problem.
- SRAM has a fixed access time to any datum, though the read and write access characteristics often differ.
- SRAM is faster than DRAM and is mainly used for cache.

Code boot vulnerability: It is a buffer overflow vulnerability in GRUB2 (GRand Unified Bootloader version 2), a piece of software that loads an Operating System (OS) into memory when a system boots up. [more details later]

DRAM: Dynamic Random Access Memories

- DRAM is comprised of a transistor and a capacitor
- It uses capacitors that loose charge over time due to leakage, even if the supply voltage is maintained.
- Since the charge on a capacitor decays when a voltage is removed, DRAM must be supplied with a voltage to retain memory (and is thus volatile).
- Capacitors can lose their charge a bit even when supplied with voltage if they have devices nearby (like transistors) that draw a little current even if they are in an "off" state; this is called capacitor leakage.
- Due to capacitor leakage, DRAM needs to be refreshed often.

Memory Chips

- A comparison of various memory types.

Type	Category	Erasure	Byte Alterable	Volatile	Typical Use
SRAM	Read/Write	Electrical	Yes	Yes	Level 2 cache
DRAM	Read/Write	Electrical	Yes	Yes	Main memory (old)
SDRAM	Read/Write	Electrical	Yes	Yes	Main memory (new)
ROM	Read only	Not possible	No	No	Large volume appliance
PROM	Read only	Not possible	No	No	Small volume appliance
EPROM	Read mostly	UV light	No	No	Device prototyping
EEPROM	Read mostly	Electrical	Yes	No	Device prototyping
Flash	Read/Write	Electrical	No	No	Digital cameras etc.

- NOTE
  - Volatile means that everything in volatile memory is lost when power is removed.
  - Byte Alterable means that stored information can be switched from one to zero, or zero to one, without a separate erase step. Flash memory technology requires a separate erase step in order to change information.

Field Programmable Devices (FPD)

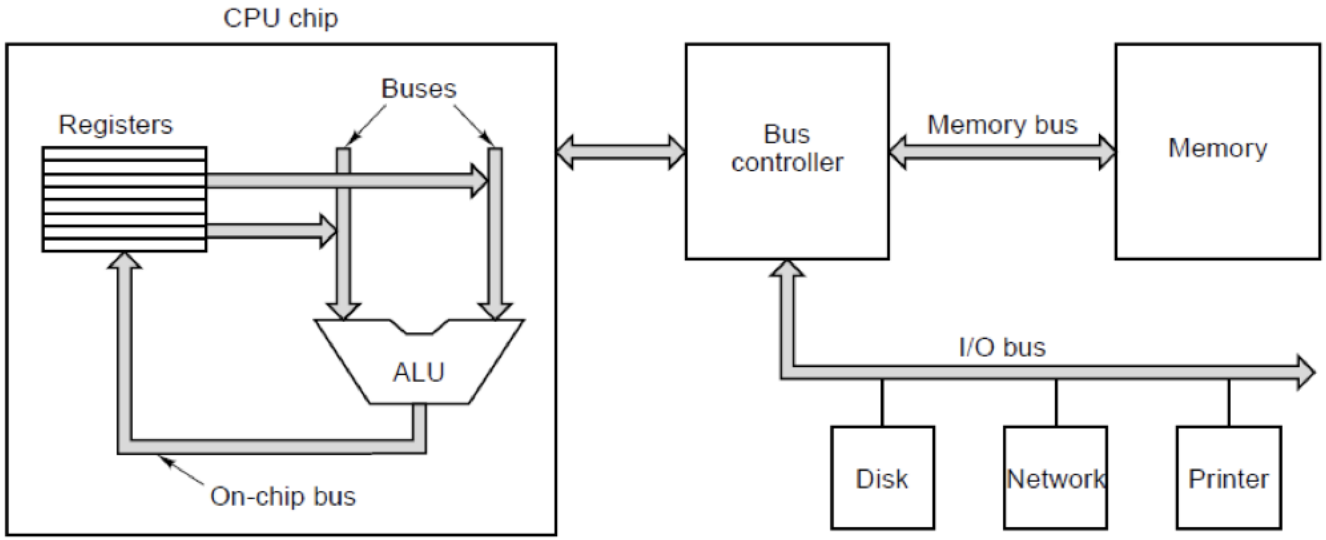
- An integrated circuit containing combinational logic, and possibly memory devices that are configurable by the end user.
- FPDs generally fall into two categories:
- **Programmable logic devices (PLDs):** which are purely combinational.
- **Field programmable gate arrays (FPGAs):** which provide both combinational logic and flip-flops

Computer Buses (1/3)

- **Definition:** Common electrical pathway between multiple devices
- **Usage:** A bus is used between components or devices which are connected to a computer.
- Following topics are discussed to understand the computer buses
  - Bus Protocol
  - Parallel and Serial Buses
  - Bus Address Lines
  - Master and Slave Buses
  - Bus Synchronization

Computer Buses (2/3)

- A computer system with multiple buses.



Computer Buses – Example (3/3)

- On-chip Buses
  - **Advanced eXtensible Interface (AXI) Bus:** Used to connect CPU core with GPU and L2 cache
- Off-chip Buses
  - **Peripheral Component Interconnect Express (PCI-Express) Bus:** Used to connect to peripherals outside CPU
  - **Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2c), Universal asynchronous receiver-transmitter (UART) Buses:** Used to connect devices like sensors and serial terminals

Bus Protocol

- The bus protocol is a well-defined set of rules governing how the bus works
- All attached devices must obey these rules
- Makes it possible for boards designed by third parties to be attached

Parallel and Serial Buses

- **Parallel buses:** It transfers several data bits at the same time.
  - It is desirable to have wide buses because large chunks of data can be transferred quickly when multiple lines can be used.
  - Parallel buses typically have 8, 16, 32 or 64 data lines.
  - Examples: ISA, EISA and PCI

- **Serial buses:** It uses the same line to transfer different data bits of the same byte/word.
  - Typically they have only one data line and the bits are sent one after the other, as a packet.
  - Examples: The Universal Standard Bus (USB) and IEEE 1394 bus architecture.

Note: Serial buses are less expensive than parallel buses, however, parallel buses have higher throughput.

Buses – Masters and Slaves

- Active devices attached to the bus that can initiate bus transfers are called masters
- Passive devices that wait for requests are called slaves
- Some devices may act as slaves at sometimes and masters at others
- Memory can never be a master device.

Examples of Bus Masters and Slaves

Master	Slave	Example
CPU	Memory	Fetching instructions and data
CPU	I/O device	Initiating data transfer
CPU	Coprocessor	CPU handling instructions of to coprocessor
I/O Device	Memory	DMA (Direct Memory Access)
Coprocessor	CPU	Coprocessor fetching operands from CPU

Bus Synchronization

Buses can be divided up into two categories, depending on their synchronization:

- **Synchronous bus**
  - It has a line driven by a crystal oscillator (Master Clock)
  - The signal on this line consists of a square wave with a frequency of 5 - 100 MHz
  - All bus activities take an integral number of cycles, called bus cycles
- **Asynchronous bus**
  - No master clock
  - Bus cycles can be of any length required and need not all be the same

Synchronous Bus

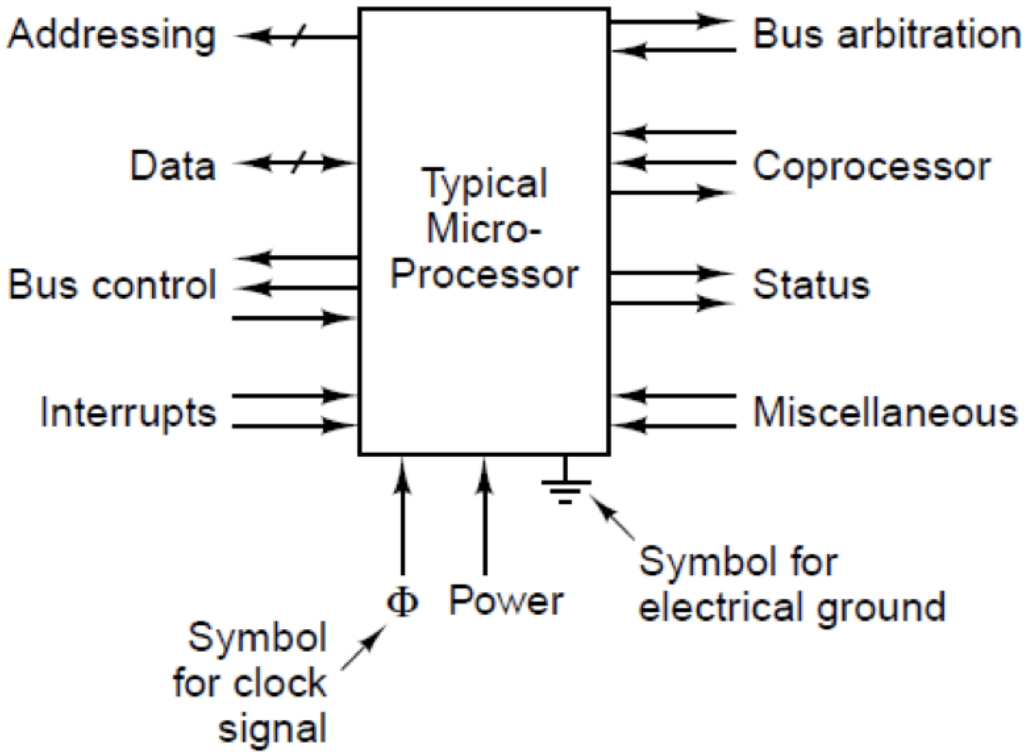
- Although synchronous buses are easy to work with due to their discrete time intervals, they also have some problems.
- Everything works in multiples of the bus cycle.
- If the CPU and memory can complete a transfer in 3.1 cycles they have to stretch it to 4.0 because fractional cycles are impossible.
- Once a bus cycle has been chosen, and memory and I/O cards have been built for it, it is difficult to take advantage of future improvements in technology. The bus has to be geared to the slowest ("legacy") devices on the bus.
- The system goes at the speed of the slowest device.

Bus Synchronization

- Despite the advantages of asynchronous buses, most buses are synchronous since it is easier to build a synchronous system.
- The CPU just asserts its signals, and the memory just reacts.
  - There is no feedback (cause and effect), but if the components have been chosen properly, everything will work without handshaking.

CPU Chips

- The logical pinout of a generic CPU. The arrows indicate input signals and output signals. The short diagonal lines indicate that multiple pins are used.



CPU Chips

CPU Chip Control Pins (1/2)

These are the different types of pins that will be found on a CPU

- **Address and Data pins:** Sometimes an address line and a data line are multiplexed into 1 line. Logically they are separate, but physically they share the same pins on the chip. There may be pins to indicate the size of the data to write: 8, 16 or 32 bits.
- **Interrupts:** There can be different levels of interrupts. The Intel chip has two levels of interrupts, the Motorola chip has eight.
- **Bus arbitration:** Intel chip has Hold (Bus Request) and Hold Acknowledge (Bus Grant). Motorola has Bus Request, Bus Grant, and Bus Acknowledge. There may be separate pins for a coprocessor.

CPU Chip Control Pins (2/2)

- **Bus Control:** Pins to indicate if it is a memory or I/O access. It is also possible to distinguish between code and data. There may be a pin to indicate if the device has completed the operation.
- **Cache Control:** Pins to allow the CPU to communicate with a cache chip.
- **Status:** Pins for indicating the current status of the CPU.
- **Miscellaneous:** The RESET pin is an example of a miscellaneous pin.

Summary

- State Elements and Edge-Triggered Methodology
- Latches and Flip-flops
- Memory Organization
- Computer Buses
- CPU chip control pins

Acknowledgements

- This lecture was created and maintained by Muhammad Fahim, Giancarlo Succi and Alexander Tormasov