SEQUENTIAL LOGIC

Digital Logic Design (CC131)

BSCS 2nd Semester Spring-2026

Lecture: Week-9

By

Syed Zulfiqar Ali Shah,

Associate Professor (CS) / HoD

Govt. College of Management Sciences, Kohat

www.linkedin.com/in/syedzulfiqaralishah

Sequential Logic

- Sequential logic is a type of digital logic in which the **output depends on both the current inputs and the previous history (stored state)** of the system.

- It includes **memory elements** such as flip-flops and latches to store past information.

Sequential Logic

- Sequential logic circuits are generally termed as two state or Bistable devices.
- They can have their output or outputs set in one of two basic states, a logic level “1” or a logic level “0” .
- They remain “latched” (hence the name latch) indefinitely in this current state or condition until some other input trigger pulse or signal is applied.

Sequential Logic

- A sequential circuit is a digital circuit composed of:
  - **Combinational logic**, and
  - **Memory elements (flip-flops/latches)**
- These circuits **store and process data over time**, and their behavior is described in terms of **states and state transitions**, often controlled by a clock signal.

Sequential Logic

Sequential circuits are fundamental in building core components of computer systems, including:

- **Registers** Store data temporarily inside the CPU
- **Counters** Used for instruction counting, timing, and control
- **Memory Units** RAM and cache rely on sequential logic for data storage
- **Control Unit** Manages execution of instructions step-by-step using states
- **Finite State Machines (FSMs)** Used in instruction decoding and control flow
- **Pipelines and Timing Control** Coordinate different stages of instruction execution

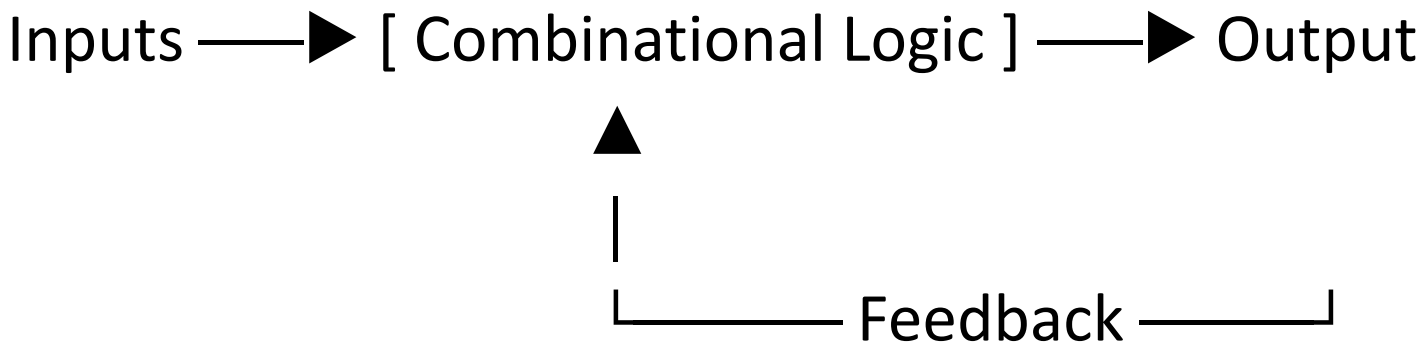
Combinational Logic VS Sequential Logic

Combinational Logic

- the output is produced and that’s it.
- the circuit cannot remember anything.

Combinational Logic VS Sequential Logic

Sequential Logic



- The **output is fed back into the circuit as input**.
- This means the next output depends on what the circuit produced *before*.
- Previous state is remembered.

Combinational Logic VS Sequential Logic

EXAMPLE:

Calculator	VS	Stopwatch
Do not remember		Remembers

Combinational Logic VS Sequential Logic

EXAMPLE:

Function

```
int add(int a, int b)
{    return a + b;   }
```

Do not remember

VS

Variable

```
int counter = 0;
counter = counter + 1;
```

Remembers

Types of Sequential Logic Devices

Sequential logic devices are mainly classified based on **how they store and update information (state)**.

1. Latches (Level-Sensitive Devices)

- Store **1 bit** of information
- Output changes **as long as input is active** (no clock required)
- Controlled by an **enable signal**

Common Types:

- **SR Latch (Set-Reset)**
- **D Latch (Data Latch)**

"Latches respond immediately to input changes"

2. Flip-Flops (Edge-Triggered Devices)

- Also store **1 bit**
- Controlled by a **clock signal**
- Output changes only at specific moments (clock edges)

Common Types:

- **SR Flip-Flop**
- **D Flip-Flop**
- **JK Flip-Flop**
- **T Flip-Flop**

“Flip-flops change state only on clock edge → more stable and reliable”

3.Registers

- Group of flip-flops
- Store **multiple bits (e.g., 4-bit, 8-bit, 16-bit)**

Types:

- Shift Registers
- Parallel Registers

Use:

- Data storage
- Data transfer

4. Counters

- Sequential circuits that **count pulses (clock signals)**

Types:

- Asynchronous (Ripple Counter)
- Synchronous Counter

Use:

- Digital clocks
- Event counting

5. Finite State Machines (FSMs)

- Advanced sequential systems
- Move between **states based on input + current state**

Types:

- Moore Machine
- Mealy Machine

Use:

- Control units
- Traffic light systems
- Sequence detectors

Classification Summary

Category	Basic Unit	Function
Latches	1-bit	Simple storage (no clock)
Flip-Flops	1-bit	Controlled storage (clocked)
Registers	Multi-bit	Store data
Counters	Multi-bit	Count events
FSMs	Multi-state	Control logic

All sequential devices are built from **flip-flops**, and they differ in how they **store and use state information**.

LATCHES

A latch is a basic sequential logic device that stores one bit of data and whose output changes continuously based on input as long as it is enabled.

Key Characteristics:

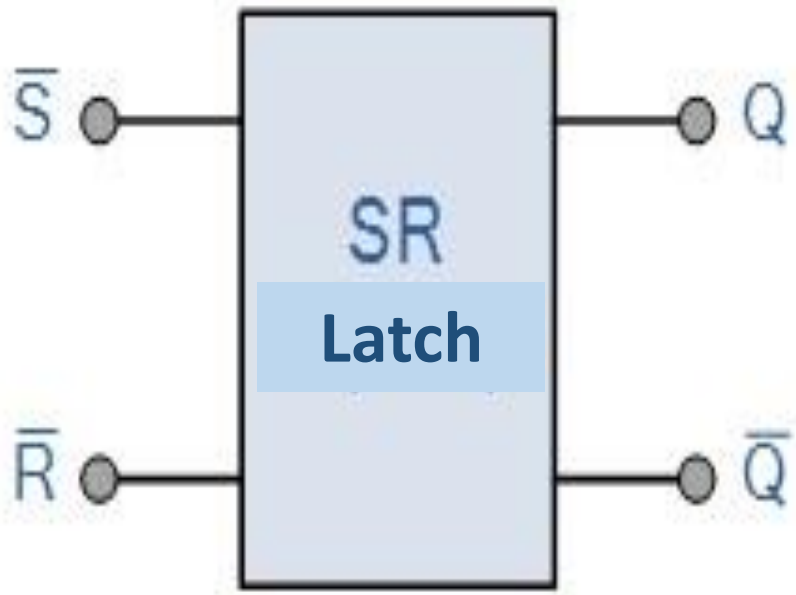
- Stores 1 bit (0 or 1)
- No clock required
- Controlled by an enable signal
- Output can change immediately when inputs change

Basic SR Latch

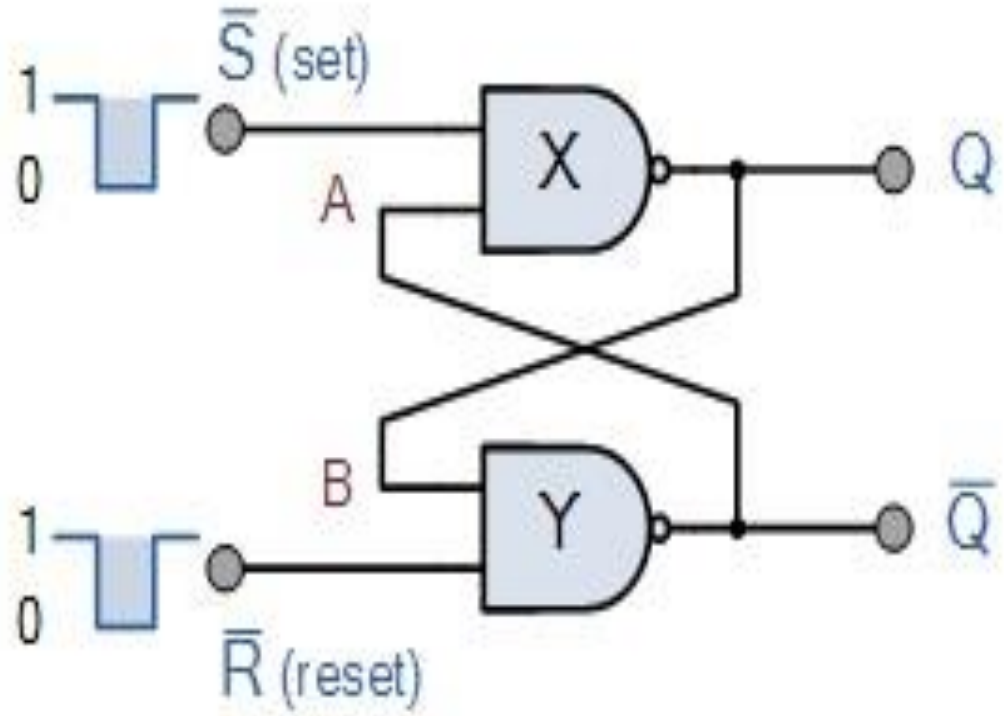
- The simplest way to make any basic single bit set-reset SR Latch is to connect together a pair of cross-coupled 2-input NAND gates, to form a Set-Reset Bistable also known as an active LOW SR NAND Gate Latch, so that there is feedback from each output to one of the other NAND gate inputs.

- This device consists of two inputs, one called the *Set*, S and the other called the *Reset*, R with two corresponding outputs Q and its inverse or complement Q (not-Q).

Basic SR Latch



Symbol



Circuit

Basic SR Latch

The Set State

Consider the circuit shown above. If the input R is at logic level “0” (R = 0) and input S is at logic level “1” (S = 1), the NAND gate *Y* has at least one of its inputs at logic “0” therefore, its output Q’ must be at a logic level “1” (NAND Gate principles). Output Q’ is also fed back to input “A” and so both inputs to NAND gate *X* are at logic level “1”, and therefore its output Q must be at logic level “0”.

Again NAND gate principals. If the reset input R changes state, and goes HIGH to logic “1” with S remaining HIGH also at logic level “1”, NAND gate *Y* inputs are now R = “1” and B = “0”. Since one of its inputs is still at logic level “0” the output at Q’ still remains HIGH at logic level “1” and there is no change of state. Therefore, the Latch circuit is said to be “Set” with Q’ = “1” and Q = “0”.

Basic SR Latch

The Reset State

In this second stable state, Q is at logic level “0”, (not Q = “0”) its inverse output at Q’ is at logic level “1”, (Q’ = “1”), and is given by R = “1” and S = “0”.

As gate X has one of its inputs at logic “0” its output Q must equal logic level “1” (again NAND gate principles). Output Q is fed back to input “B”, so both inputs to NAND gate Y are at logic “1”, therefore, Q’ = “0”.

If the set input, S now changes state to logic “1” with input R remaining at logic “1”, output Q still remains LOW at logic level “0” and there is no change of state. Therefore, the flip-flop circuits “Reset” state has also been latched and we can define this “set/reset” action in the following truth table.

Truth Table for this Set-Reset Function

State	S	R	Q	$\overline{Q}$	Description
Set	1	0	0	1	Set $\overline{Q}$ » 1
	1	1	0	1	no change
Reset	0	1	1	0	Reset $\overline{Q}$ » 0
	1	1	1	0	no change
Invalid	0	0	1	1	Invalid Condition

Thank You