Overview

- Last lecture
  - Introduction to sequential logic and systems
    - The basic concepts
    - A simple example

- Today
  - Latches
  - Flip-flops
    - Edge-triggered D
    - Master-slave
  - Timing diagrams
  - T flip-flops and SR latches

The D latch

- Output depends on clock
  - Clock high: Input passes to output
  - Clock low: Latch holds its output

- Latch are level sensitive and transparent

![D latch diagram](image-url)
The D flip-flop

- Input sampled at clock edge
  - Rising edge: Input passes to output
  - Otherwise: Flip-flop holds its output

- Flip-flops are rising-edge triggered, falling-edge triggered, or master-slave

Terminology & notation

Rising-edge triggered D flip-flop

Falling-edge triggered D flip-flop

Positive D latch

Negative D latch
Latches versus flip-flops

behavior is the same unless input changes while the clock is high

The master-slave D

Class example: Draw the timing diagram
Flip-flop timing

- **Setup time** $t_{su}$: Amount of time the input must be stable before the clock transitions high (or low for negative-edge triggered FF).
- **Hold time** $t_h$: Amount of time the input must be stable after the clock transitions high (or low for negative-edge triggered FF).

There is a timing "window" around the clock edge during which the input must remain stable.

---

Flip-flop timing (cont’d)

- **Timing constraints**
  - Must meet setup and hold times
  - Must meet minimum clock width
  - Will have propagation delays (low to high & high to low)
Latches versus flip-flops

The master-slave D

Class example: Draw the timing diagram
How do we make a latch?

- Two inverters hold a bit
  - As long as power is applied

```
1
\rightarrow
0
\rightarrow
"stored bit"
```

- Storing a new memory
  - Temporarily break the feedback path

```
"remember"
\rightarrow
"data"
\rightarrow
"load"
\rightarrow
"stored bit"
```

How do we make a D F/F?

- Edge triggering is difficult
  - Label the internal nodes
  - Draw a timing diagram
  - Start with Clk=1

```
Clk
\rightarrow
W
\rightarrow
X
\rightarrow
Q
\rightarrow
Q'
\rightarrow
Y
\rightarrow
Z
\rightarrow
D
```

T flip-flop

- Full name: Toggle flip-flop
- Output toggles when input is asserted
  - If T=1, then Q → Q' when CLK ↑
  - If T=0, then Q → Q when CLK ↑

```
<table>
<thead>
<tr>
<th>Input(t)</th>
<th>Q(t)</th>
<th>Q(t + Δt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

The SR latch

- Cross-coupled NOR gates
  - Can set (S=1, R=0) or reset (R=1, S=0) the output

- Cross-coupled NAND gates
  - Can set (S=1, R=0) or reset (R=1, S=0) the output
SR latch behavior

- Truth table and timing

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>disallow</td>
</tr>
</tbody>
</table>

SR latch is glitch sensitive

- Static 0 hazards can set/reset latch
  - Glitch on S input sets latch
  - Glitch on R input resets latch
Clear and preset in flip-flops

- **Clear** and **Preset** set flip-flop to a known state
  - Used at startup, reset

- **Clear or Reset** to a logic 0
  - Synchronous: \(Q=0\) when next clock edge arrives
  - Asynchronous: \(Q=0\) when reset is asserted
    - Doesn't wait for clock
    - Quick but dangerous

- **Preset or Set** the state to logic 1
  - Synchronous: \(Q=1\) when next clock edge arrives
  - Asynchronous: \(Q=1\) when reset is asserted
    - Doesn't wait for clock
    - Quick but dangerous