State-Holding Elements
  Latches
  D Flip-Flop
  Resets (Sync and Async)

The Synchronous Digital Logic Paradigm

Sequential Circuits
  Shift Registers
  Counters

Timing in Synchronous Circuits
Bistable Elements

Equivalent circuits; right is more traditional.

Two stable states:
RS Latch

\[ R S Q \]

\[ R Q \quad S \bar{Q} \]

\[
\begin{array}{cccc}
 R & S & Q & \bar{Q} \\
 0 & 0 & 0 & \\
 0 & 1 & 1 & \\
 1 & 0 & 0 & \\
 1 & 1 & 1 & \\
\end{array}
\]
### RS Latch

- **Truth Table**:

<table>
<thead>
<tr>
<th>$R$</th>
<th>$S$</th>
<th>$Q$</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Set ($Q = 1$)**: (when $S = 1$ and $R = 0$)
RS Latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>( \overline{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \overline{Q} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RS Latch

\[(0 + \overline{Q}) = Q\]

\[(0 + Q) = \overline{Q}\]

<table>
<thead>
<tr>
<th>$R$</th>
<th>$S$</th>
<th>$Q$</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>$\overline{Q}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**RS Latch**

<table>
<thead>
<tr>
<th>$R$</th>
<th>$S$</th>
<th>$Q$</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$Q$</td>
<td>$\overline{Q}$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**RS Latch**

<table>
<thead>
<tr>
<th>$R$</th>
<th>$S$</th>
<th>$Q$</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$Q$</td>
<td>$\overline{Q}$</td>
</tr>
</tbody>
</table>
D Latch

\[
\begin{array}{cccc}
C & D & Q & \overline{Q} \\
0 & X & Q & \overline{Q} \\
1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 \\
\end{array}
\]
A Challenge: Build a traffic light controller

Want the lights to cycle green-yellow-red.

Does this work?
Positive-Edge-Triggered D Flip-Flop
Positive-Edge-Triggered D Flip-Flop
Positive-Edge-Triggered D Flip-Flop

![Diagram of Positive-Edge-Triggered D Flip-Flop]

- Master
  - D
  - Q
  - C
  - CM
- Slave
  - D'
  - Q
  - C
  - CS
- C
- D
- CM: transparent, opaque
- D': opaque, transparent
- CS: opaque, transparent
- Q

Diagram showing the connection between master and slave with control signals C, CM, CS, D, D', and Q.
Positive-Edge-Triggered D Flip-Flop

The diagram illustrates the positive-edge triggered D flip-flop. It consists of a master and a slave section. The master section has inputs $D$, $C$, and $C_M$, and outputs $Q$ and $Q'$. The slave section has inputs $D'$, $C_S$, and outputs $Q$. The clock signal $C$ triggers the flip-flop, and the data input $D$ is transferred to the output $Q$ when the clock signal changes from low to high.

- $C$: Clock signal, transparent when high, opaque when low.
- $D$: Data input, transparent when high, opaque when low.
- $C_M$: Master clock, transparent when high, opaque when low.
- $D'$: Data input to the slave, transparent when high, opaque when low.
- $C_S$: Slave clock, transparent when high, opaque when low.
- $Q$: Output, transparent when high, opaque when low.
- $Q'$: Complementary output, transparent when low, opaque when high.

The diagram shows the waveforms for the clock signal $C$, data input $D$, master clock $C_M$, slave clock $C_S$, and output $Q$. Each waveform indicates the state of the signal during the high and low phases of the clock cycle.
Positive-Edge-Triggered D Flip-Flop

![Diagram of Positive-Edge-Triggered D Flip-Flop]

- **Master**
  - Input: D
  - Clock: C
  - Output: Q

- **Slave**
  - Input: D'
  - Clock: C
  - Output: Q

- **Control Signals**
  - **C_M**: Transparent, Opaque, Transparent
  - **C_S**: Opaque, Transparent, Opaque
Positive-Edge-Triggered D Flip-Flop
Let’s try this again with D flip-flops.
The Traffic Light Controller: A second try

Let’s try this again with D flip-flops.
The Traffic Light Controller: A second try

Let’s try this again with D flip-flops.
The Traffic Light Controller: A second try

Let’s try this again with D flip-flops.
The Traffic Light Controller: A second try

Let’s try this again with D flip-flops.
The Traffic Light Controller with Reset

CLK
RESET
R
Y
G
The Traffic Light Controller with Reset
The Traffic Light Controller with Reset
The Traffic Light Controller with Reset
The Traffic Light Controller with Reset
The Traffic Light Controller with Reset

CLK

RESET

R

Y

G
D Flip-Flop with Enable

What's wrong with this solution?
Asynchronous Preset/Clear
The Traffic Light Controller w/ Async. Reset
The Synchronous Digital Logic Paradigm

Gates and D flip-flops only

Each flip-flop driven by the same clock

Every cyclic path contains at least one flip-flop
Cool Sequential Circuits: Shift Registers

<table>
<thead>
<tr>
<th>$A$</th>
<th>$Q_0Q_1Q_2Q_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X X X X X</td>
</tr>
<tr>
<td>1</td>
<td>0 X X X X</td>
</tr>
<tr>
<td>1</td>
<td>1 0 X X X</td>
</tr>
<tr>
<td>0</td>
<td>1 1 0 X X</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1 0 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1 0 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0 0 1 0 1</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0 1 0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 0 1</td>
</tr>
<tr>
<td>0</td>
<td>1 0 0 0 0</td>
</tr>
</tbody>
</table>
Universal Shift Register

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_0$</th>
<th>$Q_3$</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$R$</td>
<td>$Q_3$</td>
<td>$Q_2$</td>
<td>$Q_1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$D_3$</td>
<td>$D_2$</td>
<td>$D_1$</td>
<td>$D_0$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$Q_3$</td>
<td>$Q_2$</td>
<td>$Q_1$</td>
<td>$Q_0$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$Q_2$</td>
<td>$Q_1$</td>
<td>$Q_0$</td>
<td>$L$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_0$</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Shift right</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Load</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Hold</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Shift left</td>
</tr>
</tbody>
</table>
Cool Sequential Circuits: Counters

Cycle through sequences of numbers, e.g.,

00 → 01 → 10 → 11
Flip-Flop Timing

Setup Time: Time before the clock edge after which the data may not change

\[ t_{su} \]
Flip-Flop Timing

Setup Time: Time before the clock edge after which the data may not change

Hold Time: Time after the clock edge after which the data may change
Flip-Flop Timing

Setup Time: Time before the clock edge after which the data may not change

Hold Time: Time after the clock edge after which the data may change

Minimum Propagation Delay: Time from clock edge to when Q might start changing
Flip-Flop Timing

Setup Time: Time before the clock edge after which the data may not change

Hold Time: Time after the clock edge after which the data may change

CLK

D

Q

Minimum Propagation Delay: Time from clock edge to when Q might start changing

Maximum Propagation Delay: Time from clock edge to when Q guaranteed stable
Timing in Synchronous Circuits

\[ t_c : \text{Clock period. E.g., 10 ns for a 100 MHz clock} \]
Timing in Synchronous Circuits

Sufficient Hold Time?

Hold time constraint: how soon after the clock edge can D start changing? Min. FF delay + min. logic delay
Timing in Synchronous Circuits

Sufficient Setup Time?

$$t_{p}(\text{max, FF})$$

$$t_{p}(\text{max, CL})$$

Setup time constraint: when before the clock edge is D guaranteed stable? Max. FF delay + max. logic delay
Clock Skew: What Really Happens

- Sufficient Hold Time?
- $t_{skew}$
- $t_{p(min,FF)}$ and $t_{p(min,CL)}$
- CLK$_2$ arrives late: clock skew reduces hold time
Clock Skew: What Really Happens

Sufficient Setup Time?

CLK₂ arrives early: clock skew reduces setup time