State-Holding Elements
  Bistable Elements
  RS Latch
  D Latch
  Positive-Edge-Triggered D Flip-Flop
  D Flip-Flop with Enable

Synchronous Digital Logic
  The Synchronous Paradigm
  Shift Registers
  Counters

Timing in Synchronous Circuits
  Flip-Flop Timing
  Timing in Synchronous Circuits
  Clock Skew
State-Holding Elements
Bistable Elements

Equivalent circuits; right is more traditional.

Two stable states:
A Bistable in the Wild

This “debounces” the coin switch.

RS Latch

<table>
<thead>
<tr>
<th></th>
<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></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
## RS Latch

![RS Latch Diagram]

<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

- $R$ ___
- $S$ ___
- $Q$ ___  Set
- $\overline{Q}$ ___
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></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( R \quad S \quad Q \quad \overline{Q} \) | Hold, State 1
**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>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>

- **Waveforms:**
  - $R$: 0
  - $S$: 0
  - $Q$: Reset
  - $\overline{Q}$: Reset

- **Diagram:**
  - RS latch with inputs $R$ and $S$,
  - Outputs $Q$ and $\overline{Q}$,
  - Logic gates illustrating the state transitions.

This RS Latch is designed to store binary information using two inputs ($R$ and $S$) and two outputs ($Q$ and $\overline{Q}$). The truth table and waveforms demonstrate the behavior under different input conditions, including hold, set, and reset states.
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></td>
<td></td>
</tr>
</tbody>
</table>

$R$  
$S$  
$Q$  
$\overline{Q}$  

Hold, State 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>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>

Huh?
RS Latch

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

R
S
Q
̅Q

Set
RS Latch

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>S</td>
<td>Q</td>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>Q</td>
<td>Hold</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Bad</td>
</tr>
</tbody>
</table>

Hold, State 1
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>

Huh?
RS Latch

<table>
<thead>
<tr>
<th></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>

\[ R \quad S \quad Q \quad \overline{Q} \quad \text{Undetermined} \]
Generates horizontal and vertical synchronization waveforms from counter bits.

D Latch

<table>
<thead>
<tr>
<th>inputs</th>
<th>outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>0</td>
<td>\textit{X}</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Diagram of D Latch with logic gates and truth table.
A Challenge

A simple 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

\[
\begin{array}{c}
\text{Master} \\
\begin{array}{c}
\text{D} \\
\text{Q}
\end{array} \\
\text{C} \\
\end{array} \\
\begin{array}{c}
\text{Slave} \\
\begin{array}{c}
\text{D} \\
\text{Q}
\end{array} \\
\text{C} \\
\end{array}
\]

\[
\begin{array}{c}
\text{D} \\
\text{C} \\
\text{M} \\
\text{S}
\end{array}
\]

\[
\begin{array}{c}
\text{D} \\
\text{Q}
\end{array}
\]

\[
\begin{array}{c}
\text{D}' \\
\text{C}_M \\
\text{transparent} \\
\text{D}' \\
\text{C}_S \\
\text{opaque} \\
\text{Q}
\end{array}
\]
Positive-Edge-Triggered D Flip-Flop

The diagram illustrates a positive-edge-triggered D flip-flop. It consists of a master and a slave section, with the master section controlled by the clock signal (C) and the slave section triggered by the clock edge (C'). The data input (D) is connected to both sections, and the output (Q) is generated by the slave section. The clock signals and data inputs are shown with waveforms indicating their timing relationships.
Positive-Edge-Triggered D Flip-Flop
Positive-Edge-Triggered D Flip-Flop

The diagram shows the logic diagram of a positive-edge-triggered D flip-flop. The inputs and outputs are labeled as follows:

- **D**: The input data.
- **C**: The clock input.
- **Q**: The output.
- **C_M**: The master clock input.
- **C_S**: The slave clock input.
- **D'**: The inverted input data.
- **Q'**: The inverted output.

The behavior of the flip-flop is as follows:

- When the clock (C) is high, the data (D) is transferred to the output (Q).
- When the clock (C) is low, the output (Q) remains unchanged.

The timing diagram illustrates the transitions of the inputs and output with respect to the clock signal. The diagram shows the states as follows:

- **C**: Clock input.
- **D**: Input data.
- **C_M**: Master clock input.
- **C_S**: Slave clock input.
- **Q**: Output.

The flip-flop operates in two phases:

1. **Master Phase**: When **C_M** is transparent, the data **D** is transferred to the master (M) flip-flop.
2. **Slave Phase**: When **C_S** is transparent, the data from the master flip-flop is transferred to the slave (S) flip-flop, resulting in the output **Q**.

The diagram also includes symbols for transparent and opaque states, indicating when the flip-flop is enabled or disabled for data transfer.
Positive-Edge-Triggered D Flip-Flop

Master

Slave

\( D \)
\( Q \)
\( C \)

\( D' \)
\( Q \)
\( C \)

\( C_M \)
\( C_S \)

\( D \)
\( D' \)

\( C \)

\( Q \)

\( C_M \) transparent, opaque, transparent, opaque

\( C_S \) opaque, transparent, opaque, transparent
The Traffic Light Controller: A second try

Let’s try this again with D flip-flops.

---

CLK: ___
R: ___
Y: ___
G: ___
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.
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

CLK

RESET

R

Y

G
The Traffic Light Controller with Reset

CLK

RESET

R

Y

G
The Traffic Light Controller with Reset
D Flip-Flop with Enable

What’s wrong with this solution?

<table>
<thead>
<tr>
<th>C</th>
<th>E</th>
<th>D</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>X</td>
<td>Q</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Q</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Q</td>
</tr>
</tbody>
</table>
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>Q0Q1Q2Q3</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</td>
</tr>
<tr>
<td>1</td>
<td>1 0 X X</td>
</tr>
<tr>
<td>0</td>
<td>1 1 0 X</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0</td>
</tr>
<tr>
<td>0</td>
<td>1 0 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>0</td>
<td>0 0 1 0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 1</td>
</tr>
<tr>
<td>0</td>
<td>1 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.,

\[ 00 \rightarrow 01 \rightarrow 10 \rightarrow 11 \]
The 74LS163 Synchronous Binary Counter
Flip-Flop Timing

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

CLK  
D  
Q
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

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

$t_c$: 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

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?

CLK2 arrives late: clock skew reduces hold time
Clock Skew: What Really Happens

CLK1 arrives early: clock skew reduces setup time