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.

Breakout, Atari 1976.
RS Latch

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

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\bar{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>

\(R\), \(S\), \(Q\), \(\bar{Q}\)
RS Latch

\[
\begin{array}{c|c|c|c}
R & S & Q & \overline{Q} \\
\hline
0 & 0 & Q & \overline{Q} \quad \text{Hold} \\
0 & 1 & 1 & 0 \quad \text{Set} \\
1 & 0 & \text{Hold, State 1} \\
1 & 1 & \\
\end{array}
\]
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></td>
<td></td>
</tr>
</tbody>
</table>

R: Set
S: Reset
Q: Output
̅Q: Complement of Q

Diagram of RS Latch with truth table.
RS Latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>( \bar{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>( \bar{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>

- **Hold, State 0**
RS Latch

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

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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>$\overline{Q}$</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>

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

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

Set

Set
RS Latch

\[
\begin{array}{c|c|c|c|c|}
R & S & Q & \overline{Q} \\
\hline
0 & 0 & Q & \overline{Q} & \text{Hold} \\
0 & 1 & 1 & 0 & \text{Set} \\
1 & 0 & 0 & 1 & \text{Reset} \\
1 & 1 & 0 & 0 & \text{Bad} \\
\end{array}
\]

\[
\begin{array}{c|c|c}
R & S & Q \\
\hline
R & 1 & 1 \\
S & 1 & 0 \\
Q & 0 & 1 \\
\overline{Q} & 0 & 1 \\
\end{array}
\]

Hold, State 1
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>1</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>$\overline{Q}$</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>

- **Hold**: $R = S = 0$
- **Set**: $R = 0, S = 1$
- **Reset**: $R = 1, S = 0$
- **Bad**: $R = 1, S = 1$

Undefined states:

- $Q$ and $\overline{Q}$ when $R = 1$ and $S = 1$
SR Latches in the Wild

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>$X$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</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

Diagram:

- **Main D Flip-Flop**: $D \rightarrow Q$
  - $C_M$ connected to $D$
- **Secondary D Flip-Flop**: $D' \rightarrow Q$
  - $C_S$ connected to $D'$

- **Clock Input**: $C$
  - **Transparent** connection
  - **Opaque** connection

Annotations:
- Transparent $D$ latches
- Pos. edge
- Clock input

Symbols:
- $D$
- $D'$
- $C$
- $C_M$
- $C_S$
- $Q$

Note: The diagram illustrates a positive-edge-triggered D flip-flop with transparent and opaque connections for the clock inputs. The main flip-flop is triggered by $D$, and the secondary flip-flop is triggered by $D'$. The clock inputs are connected in a way that allows for proper triggering at the positive edge.
Positive-Edge-Triggered D Flip-Flop

\[ \text{Main} \]
\[ \text{Secondary} \]

- **D**: Input data
- **C**: Clock
- **CM**: Clock enable (transparent)
- **CS**: Clock enable (opaque)
- **Q**: Output

Diagram shows the flow of signals through the flip-flop with different states for **D**, **C**, **CM**, **CS**, and **Q**.
Positive-Edge-Triggered D Flip-Flop

Diagram showing the positive-edge triggered D flip-flop with inputs D, C, C_M, and C_S, and outputs Q and Q'. The diagram illustrates the state transitions for different input conditions.
Positive-Edge-Triggered D Flip-Flop

```
D       0
C       
CM      

Main
D     Q
C

Secondary
D'    Q
C

D       Q
```

C

D

CM transparent opaque

D' opaque transparent

C' S

Q
Positive-Edge-Triggered D Flip-Flop
Positive-Edge-Triggered D Flip-Flop

The diagram illustrates the operation of a positive-edge-triggered D flip-flop. The input $D$ is applied to the primary flip-flop (Main) and then to the secondary flip-flop (Secondary) through control signals $C_M$ and $C_S$.

- **Main Flip-Flop**: 
  - $D$ input
  - $Q$ output

- **Secondary Flip-Flop**: 
  - $D'$ input
  - $Q'$ output

Control signals $C_M$ and $C_S$ are used to control the transparency and opacity of the flip-flops. The timing diagram shows the relationship between the input signals and the output $Q$.

- $C$: Control signal
- $D$: Data input
- $D'$: Data input for secondary flip-flop
- $Q$: Output
- $C_M$: Main control signal
- $C_S$: Secondary control signal
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.

![Diagram of traffic light controller with D flip-flops and timing diagram.]
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
D Flip-Flop with Enable

What's wrong with this solution?

If clock is high, a glitch on E if E goes high could change Q

The problem: If clock is high, a glitch on E.
Asynchronous Preset/Clear

Active low

CLK
D
PRE
CLR
Q

Not how you use it
The Traffic Light Controller w/ Async. Reset

![Traffic Light Diagram]
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

Discretizing Time
Cool Sequential Circuits: Shift Registers

\[
\begin{array}{c|ccccc}
A & Q_0 & Q_1 & Q_2 & Q_3 \\
\hline
0 & X & X & X & X & X \\
1 & 0 & X & X & X \\
1 & 1 & 0 & X & X \\
0 & 1 & 1 & 0 & X \\
1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 \\
\end{array}
\]
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
Implementing a 4-bit Binary Counter

<table>
<thead>
<tr>
<th>(Q_{8421})</th>
<th>(D_{8421})</th>
<th>(D_{1})</th>
<th>(Q_{1})</th>
<th>(D_{2})</th>
<th>(D_{4})</th>
<th>(D_{8})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0001</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>0010</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>0011</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0011</td>
<td>0100</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0100</td>
<td>0101</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0101</td>
<td>0110</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0110</td>
<td>0111</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0111</td>
<td>1000</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1000</td>
<td>1001</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1001</td>
<td>1010</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1010</td>
<td>1011</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1011</td>
<td>1100</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1100</td>
<td>1101</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1101</td>
<td>1110</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1110</td>
<td>1111</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1111</td>
<td>0000</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\(D_{1} = \overline{Q_{1}}\)
\(D_{2} = Q_{1}\overline{Q_{2}} + \overline{Q_{1}}Q_{2}\)
\(D_{4} = Q_{1}Q_{2}\overline{Q_{4}} + \overline{Q_{1}}Q_{4} + \overline{Q_{2}}Q_{4}\)
\(D_{8} = Q_{1}Q_{2}Q_{4}\overline{Q_{8}} + \overline{Q_{2}}Q_{8} + \overline{Q_{1}}Q_{8} + \overline{Q_{4}}Q_{8}\)
Implementing a 4-bit Binary Counter

<table>
<thead>
<tr>
<th>$Q_{8421}$</th>
<th>$D_{8421}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0001</td>
</tr>
<tr>
<td>0001</td>
<td>0010</td>
</tr>
<tr>
<td>0010</td>
<td>0011</td>
</tr>
<tr>
<td>0011</td>
<td>0100</td>
</tr>
<tr>
<td>0100</td>
<td>0101</td>
</tr>
<tr>
<td>0101</td>
<td>0110</td>
</tr>
<tr>
<td>0110</td>
<td>0111</td>
</tr>
<tr>
<td>0111</td>
<td>1000</td>
</tr>
<tr>
<td>1000</td>
<td>1001</td>
</tr>
<tr>
<td>1001</td>
<td>1010</td>
</tr>
<tr>
<td>1010</td>
<td>1011</td>
</tr>
<tr>
<td>1011</td>
<td>1100</td>
</tr>
<tr>
<td>1100</td>
<td>1101</td>
</tr>
<tr>
<td>1101</td>
<td>1110</td>
</tr>
<tr>
<td>1110</td>
<td>1111</td>
</tr>
<tr>
<td>1111</td>
<td>0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$D_1$</th>
<th>$Q_1$</th>
<th>$D_2$</th>
<th>$D_4$</th>
<th>$D_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Trick: XOR each $D$ with its $Q$

- $D_1 \oplus Q_1 = 1$
- $D_2 \oplus Q_2 = Q_1$
- $D_4 \oplus Q_4 = Q_1 Q_2$
- $D_8 \oplus Q_8 = Q_1 Q_2 Q_4$

"Count up" at 1111
Implementing a 4-bit Binary Counter

Theorem: \( X \oplus 1 \oplus 1 = X \)

If \( X \oplus Y = Z \) then \( X = Y \oplus Z \)

Proof:
\( X \oplus Y = Z \) (assumed)
\( Y \oplus X \oplus Y = Y \oplus Z \) (apply \( Y \oplus \))
\( Y \oplus Y \oplus X = Y \oplus Z \) (\( \oplus \) commutes)
\( X = Y \oplus Z \) (\( y \oplus y \oplus x = x \))

So if
\( D_1 \oplus Q_1 = 1 \)
\( D_2 \oplus Q_2 = Q_1 \)
\( D_4 \oplus Q_4 = Q_1 Q_2 \)
\( D_8 \oplus Q_8 = Q_1 Q_2 Q_4 \),

\( D_1 = Q_1 \oplus 1 \)
\( D_2 = Q_2 \oplus Q_1 \)
\( D_4 = Q_4 \oplus Q_1 Q_2 \)
\( D_8 = Q_8 \oplus Q_1 Q_2 Q_4 \)
Flip-Flop Timing

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

$t_{su}$

$0$ (ps)

stable (no change)

may be 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

$\text{CLK} \quad \underline{\text{D}} \quad \underline{\text{Q}}$

$t_{su} \quad t_h$

may be negative!
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 late, CLK2 arrives early: clock skew reduces setup time

Sufficient Setup Time?

CLK2 arrives early: clock skew reduces setup time