Fundamentals of Computer Systems
Sequential Logic

Stephen A. Edwards

Columbia University

Spring 2012
State-Holding Elements
Bistable Elements

Equivalent circuits; right is more traditional.

Two stable states:
A Bistable in the Wild

This “debounces” the coin switch.

SR Latch
SR Latch

\[ R \quad 0 \quad 1 \quad Q \]
\[ S \quad 1 \quad 0 \quad \overline{Q} \]

Set
SR Latch

\[
\begin{align*}
R & \quad 0 \\
S & \quad 0 \\
Q & \quad 0 \\
\overline{Q} &\quad 1
\end{align*}
\]

Hold, State 1
SR Latch

\[
\begin{align*}
R & \quad 1 \\
S & \quad 0 \\
Q & \quad 0 \\
\overline{Q} & \quad 1 \\
\end{align*}
\]

**Reset**
SR Latch

Hold, State 0
SR Latch

\[ \begin{align*}
R & \quad 1 \\
S & \quad 1 \\
Q & \quad 0 \\
\bar{Q} & \quad 0 \\
\end{align*} \]

\[ \begin{align*}
R & \quad \text{Huh?} \\
S & \\
Q & \\
\bar{Q} & \\
\end{align*} \]
SR Latch

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

Hold, State 1
SR Latch

The diagram shows a SR latch with inputs R and S. The truth table for an SR latch is:

- R = 0 and S = 0: Q = Q = 0
- R = 0 and S = 1: Q = 0, Q = 1
- R = 1 and S = 0: Q = 1, Q = 0
- R = 1 and S = 1: Q and Q are indeterminate

The waveforms for R, S, Q, and Q̅ are shown with expected responses. However, there is a question mark indicating confusion or an unexpected result.

Huh?
SR Latch

The diagram illustrates the SR latch with its truth table shown below. The states of the inputs (R, S) and outputs (Q, \overline{Q}) are tracked over time, with the outputs Q and \overline{Q} transitioning to undefined states at certain points. The SR latch is a type of memory element used in digital electronics to store binary information.
SR Latches in the Wild

Generates horizontal and vertical synchronization waveforms from counter bits.

D Latch

The diagram illustrates a D latch circuit with inputs D and C and outputs Q and Q̅. The truth table below shows the behavior of the D latch for different input combinations:

<table>
<thead>
<tr>
<th>inputs</th>
<th>outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>C  D</td>
<td>Q  Q̅</td>
</tr>
<tr>
<td>0    X</td>
<td>Q  Q̅</td>
</tr>
<tr>
<td>1    0</td>
<td>0  1</td>
</tr>
<tr>
<td>1    1</td>
<td>1  0</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

- Master
- Slave

\[ D \quad Q \quad C_M \quad C_S \quad D' \quad Q \]

- \( C \) (transparent)
- \( D \)
- \( C_M \) (transparent)
- \( D' \)
- \( C_S \) (opaque)
- \( Q \)
Positive-Edge-Triggered D Flip-Flop

Master

\[ D \rightarrow Q \]
\[ C \rightarrow C_M \]

Slave

\[ D' \rightarrow Q \]
\[ C \rightarrow C_S \]

\[ C \quad D \quad D' \quad Q \quad Q \]

\[ C_M \quad transparent \]
\[ C_S \quad opaque \]
Positive-Edge-Triggered D Flip-Flop
Positive-Edge-Triggered D Flip-Flop

Diagram of master and slave sections with inputs and outputs labeled.

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

Waveforms showing timing for inputs C, D, CM, D', CS, Q.
Positive-Edge-Triggered D Flip-Flop

Diagram showing the positive-edge-triggered D flip-flop with master and slave sections. The input signals are labeled as $D$, $C$, $C_M$, and $C_S$. The output signals are labeled as $Q$ and $Q'$. The timing diagrams for $C$, $D$, $C_M$, $D'$, $C_S$, and $Q$ are shown with different states of transparency and opacity.
Positive-Edge-Triggered D Flip-Flop

```
  Positive-Edge-Triggered D Flip-Flop
```

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

```
  C
  D
  C_M
  D'
  C_S
  Q
  Q
```

```
  transparent
  opaque
  transparent
  opaque
  opaque
  transparent
```

```
  D
  D'
  C_M
  C_S
  Q
```

```
  opaque
  transparent
  opaque
```

```
  D
  D'
  C_M
  C_S
  Q
```

```
  C
  D
  C_M
  D'
  C_S
  Q
```

```
  C
  D
  C_M
  D'
  C_S
  Q
```
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.
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?
Asynchronous Preset/Clear
The Traffic Light Controller w/ Async. Reset
The Synchronous Digital Logic 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

![Shift Register Diagram]

<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 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 X</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 1 0</td>
</tr>
<tr>
<td>0</td>
<td>1 0 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0 1 0 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><strong>Operation</strong></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>
Cycle through sequences of numbers, e.g.,

00 → 01 → 10 → 11
The 74LS163 Synchronous Binary Counter

Diagram of the 74LS163 Synchronous Binary Counter with inputs CLK, LOAD, CLR, DATA A, DATA B, DATA C, DATA D, and outputs QA, QB, QC, QD, RCO.
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
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?

CLK\textsubscript{1} \hspace{1cm} CLK\textsubscript{2} \hspace{1cm} Q \hspace{1cm} D

\begin{equation} t_{skew} \end{equation}

CLK\textsubscript{2} arrives late: clock skew reduces hold time
Clock Skew: What Really Happens

Sufficient Setup Time?

CLK₁ arrives early: clock skew reduces setup time