Fundamentals of Computer Systems
Sequential Logic

Stephen A. Edwards

Columbia University

Fall 2011
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 Diagram]
SR Latch

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

\[ Q \quad \text{Set} \]

\[ \bar{Q} \quad \_ \]
SR Latch

Hold, State 1
SR Latch

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

Hold, State 0
SR Latch

Huh?
SR Latch

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

Set

\( R \) \hfill \( S \) \hfill \( Q \) \hfill \( \overline{Q} \)
SR Latch

The diagram shows a SR Latch circuit with the following inputs and outputs:
- **R**: 0
- **S**: 0
- **Q**: 1
- **\(\overline{Q}\)**: 0

The waveform diagrams indicate that the latch is in a hold state, specifically State 1, where the inputs are stable and do not change, allowing the output to remain constant.
SR Latch

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

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

\[ R \quad \text{Huh?} \]
\[ S \]
\[ Q \]
\[ \bar{Q} \]
SR Latch

\[ R \quad X \quad Q \]

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

Diagram of SR Latch with waveforms for inputs and outputs:

- **R**
- **S**
- **Q**
- **\overline{Q}**

Undefined state highlighted on diagram.
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>

The table shows the truth table for a D Latch, with inputs $C$ and $D$, and outputs $Q$ and $\overline{Q}$. The diagram illustrates the circuit implementation of the D Latch.
A simple traffic light controller.
Want the lights to cycle green-yellow-red.

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

Master

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

Slave

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

\[ D \quad Q \]

\[ C \quad M \]

\[ C \quad S \]

\[ C \]

\[ D \]

\[ C_M \quad \text{transparent} \]

\[ D' \]

\[ C_S \quad \text{opaque} \]

\[ Q \]
Positive-Edge-Triggered D Flip-Flop

Master

Slave

\[ D \rightarrow D' \rightarrow Q \]

\[ C \]

\[ C_M \] transparent

\[ C_S \] opaque

\[ Q \]
Positive-Edge-Triggered D Flip-Flop

- **Master**
  - Input: \( D \)
  - Clock: \( C \)
  - Clock select: \( C_M \)
  - Output: \( Q \)

- **Slave**
  - Input: \( D' \)
  - Clock: \( C \)
  - Clock select: \( C_S \)
  - Output: \( Q \)

- **Logic Diagram**
  - The diagram shows the flow of signals through the flip-flop, with the master and slave stages connected by the clock and clock select signals.

- **Waveforms**
  - \( C \): Clock signal
  - \( D \): Data input
  - \( C_M \): Master clock select
  - \( D' \): Data input for slave
  - \( C_S \): Slave clock select
  - \( Q \): Output signal
Positive-Edge-Triggered D Flip-Flop
Positive-Edge-Triggered D Flip-Flop

Master

Slave

\[ D \rightarrow C \rightarrow Q \]

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

\[ C_M \rightarrow \text{transparent} \rightarrow \text{opaque} \rightarrow \text{transparent} \]

\[ C_S \rightarrow \text{opaque} \rightarrow \text{transparent} \rightarrow \text{opaque} \]
Positive-Edge-Triggered D Flip-Flop

Diagram:
- Master D Flip-Flop:
  - D input
  - C input
  - Q output
  - CM control

- Slave D Flip-Flop:
  - D' input
  - C' input
  - Q output
  - CS control

Waveforms:
- C
- D
- CM: transparent, opaque, transparent, opaque
- D': opaque, transparent, opaque
- CS: opaque, transparent, opaque, transparent
- Q
The Traffic Light Controller: A second try

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

CLK
R
Y
G

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
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 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

A  |  Q_0Q_1Q_2Q_3
---|-----------------|
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
Universal Shift Register

- **Operation**
  - $S_1$  $S_0$  Operation
  - 0  0  Shift right
  - 0  1  Load
  - 1  0  Hold
  - 1  1  Shift left

- **Truth Table**

<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>
Cool Sequential Circuits: Counters

Cycle through sequences of numbers, e.g.,

00 → 01 → 10 → 11
The 74LS163 Synchronous Binary Counter
Flip-Flop Timing

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

CLK

D

Q

$t_{su}$
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.
- **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?

$\text{Hold time constraint: how soon after the clock edge can D start changing? Min. FF delay + min. logic delay}$
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?

CLK1

CLK2

Q

D

$tp_{(\text{min,FF})}$

$tp_{(\text{min,CL})}$

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

Sufficient Setup Time?

CLK₁ arrives early: clock skew reduces setup time