Chapter 3 :: Sequential Logic Design

_Digital Design and Computer Architecture_

David Money Harris and Sarah L. Harris
Bistable Circuit

• Fundamental building block of other state elements
• Two outputs: $Q$, $\overline{Q}$
• No inputs
Bistable Circuit Analysis

- Consider the two possible cases:
  - \( Q = 0 \): then \( \overline{Q} = 1 \) and \( Q = 0 \) (consistent)
  - \( Q = 1 \): then \( \overline{Q} = 0 \) and \( Q = 1 \) (consistent)

- Bistable circuit stores 1 bit of state in the state variable, \( Q \) (or \( \overline{Q} \))
- But there are no inputs to control the state
SR (Set/Reset) Latch

• SR Latch

Consider the four possible cases:
- \( S = 1, R = 0 \)
- \( S = 0, R = 1 \)
- \( S = 0, R = 0 \)
- \( S = 1, R = 1 \)
SR Latch Analysis

- $S = 1, R = 0$: then $Q = 1$ and $\overline{Q} = 0$

- $S = 0, R = 1$: then $Q = 0$ and $\overline{Q} = 1$
SR Latch Analysis

- $S = 1, R = 0$: then $Q = 1$ and $\bar{Q} = 0$

- $S = 0, R = 1$: then $Q = 0$ and $\bar{Q} = 1$
SR Latch Analysis

- \( S = 0, R = 0 \): then \( Q = Q_{\text{prev}} \)
  \[
  Q_{\text{prev}} = 0
  \]

- \( S = 1, R = 1 \): then \( Q = 0 \) and \( \bar{Q} = 0 \)

- \( S = 0, R = 0 \): then \( Q = Q_{\text{prev}} \)
  \[
  Q_{\text{prev}} = 1
  \]

- \( S = 1, R = 1 \): then \( Q = 0 \) and \( \bar{Q} = 0 \)
SR Latch Analysis

- \( S = 0, \ R = 0 \): then \( Q = Q_{prev} \) and \( \overline{Q} = \overline{Q_{prev}} \) (memory!)
  \( Q_{prev} = 0 \)

- \( S = 1, \ R = 1 \): then \( Q = 0 \) and \( \overline{Q} = 0 \) (invalid state: \( \overline{Q} \neq \text{NOT} \ Q \))

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

\[\begin{array}{c}
R & 0 & 1 & \\hline
S & \begin{array}{c} 0 \\hline 1 \end{array} & \begin{array}{c} 0 \\hline 1 \end{array} & \begin{array}{c} 1 \\hline 0 \end{array} \\
\hline
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1
\end{array}\]
SR stands for Set/Reset Latch
- Stores one bit of state ($Q$)
- Control what value is being stored with $S$, $R$ inputs
  - Set: Make the output 1 ($S = 1$, $R = 0$, $Q = 1$)
  - Reset: Make the output 0 ($S = 0$, $R = 1$, $Q = 0$)

Must do something to avoid invalid state (when $S = R = 1$)
D Latch Internal Circuit

CLK \rightarrow D

\begin{array}{c|c|c|c|c|}
CLK & D & \overline{D} & S & R & Q & \overline{Q} \\
0 & X & & & & & \\
1 & 0 & & & & & \\
1 & 1 & & & & & \\
\end{array}
D Latch Internal Circuit

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>$\overline{D}$</th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>$Q_{prev}$</td>
<td>$\overline{Q}_{prev}$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
D Flip-Flop

- Two inputs: \( CLK, D \)
- **Function**
  - The flip-flop “samples” \( D \) on the rising edge of \( CLK \)
    - When \( CLK \) rises from 0 to 1, \( D \) passes through to \( Q \)
    - Otherwise, \( Q \) holds its previous value
  - \( Q \) changes only on the rising edge of \( CLK \)
- A flip-flop is called an *edge-triggered* device because it is activated on the clock edge

D Flip-Flop

Symbols

\[ \text{D Flip-Flop} \]
D Flip-Flop Internal Circuit

- Two back-to-back latches (L1 and L2) controlled by complementary clocks
- When $CLK = 0$
  - L1 is transparent
  - L2 is opaque
  - $D$ passes through to N1
- When $CLK = 1$
  - L2 is transparent
  - L1 is opaque
  - N1 passes through to $Q$
- Thus, on the edge of the clock (when $CLK$ rises from 0→1)
  - $D$ passes through to $Q$
Timing

- Flip-flop samples $D$ at clock edge
- $D$ must be stable when it is sampled
- Similar to a photograph, $D$ must be stable around the clock edge
- If $D$ is changing when it is sampled, metastability can occur
Input Timing Constraints

- Setup time: $t_{\text{setup}} =$ time before the clock edge that data must be stable (i.e. not changing)
- Hold time: $t_{\text{hold}} =$ time after the clock edge that data must be stable
- Aperture time: $t_a =$ time around clock edge that data must be stable ($t_a = t_{\text{setup}} + t_{\text{hold}}$)
Output Timing Constraints

• Propagation delay: $t_{pcq} =$ time after clock edge that the output $Q$ is guaranteed to be stable (i.e., to stop changing)
• Contamination delay: $t_{ccq} =$ time after clock edge that $Q$ might be unstable (i.e., start changing)
Dynamic Discipline

- The input to a synchronous sequential circuit must be stable during the aperture (setup and hold) time around the clock edge.
- Specifically, the input must be stable
  - at least $t_{\text{setup}}$ before the clock edge
  - at least until $t_{\text{hold}}$ after the clock edge
Dynamic Discipline

- The delay between registers has a **minimum** and **maximum** delay, dependent on the delays of the circuit elements.
Setup Time Constraint

- The setup time constraint depends on the maximum delay from register R1 through the combinational logic.
- The input to register R2 must be stable at least $t_{\text{setup}}$ before the clock edge.

$$T_c \geq$$
Setup Time Constraint

- The setup time constraint depends on the maximum delay from register R1 through the combinational logic.
- The input to register R2 must be stable at least $t_{\text{setup}}$ before the clock edge.

\[ T_c \geq t_{\text{pcq}} + t_{\text{pd}} + t_{\text{setup}} \]

\[ t_{\text{pd}} \leq \]
Setup Time Constraint

- The setup time constraint depends on the maximum delay from register R1 through the combinational logic.
- The input to register R2 must be stable at least $t_{\text{setup}}$ before the clock edge.

$$T_c \geq t_{\text{pcq}} + t_{\text{pd}} + t_{\text{setup}}$$

$$t_{\text{pd}} \leq T_c - (t_{\text{pcq}} + t_{\text{setup}})$$
Hold Time Constraint

- The hold time constraint depends on the **minimum** delay from register R1 through the combinational logic.
- The input to register R2 must be stable for at least $t_{\text{hold}}$ after the clock edge.
Hold Time Constraint

- The hold time constraint depends on the minimum delay from register R1 through the combinational logic.
- The input to register R2 must be stable for at least $t_{\text{hold}}$ after the clock edge.

\[ t_{\text{hold}} < t_{\text{ccq}} + t_{\text{cd}} \]

\[ t_{\text{cd}} > \]
Hold Time Constraint

- The hold time constraint depends on the **minimum** delay from register R1 through the combinational logic.
- The input to register R2 must be stable for at least $t_{\text{hold}}$ after the clock edge.

$$t_{\text{hold}} < t_{\text{ccq}} + t_{\text{cd}}$$

$$t_{\text{cd}} > t_{\text{hold}} - t_{\text{ccq}}$$
Timing Analysis

Timing Characteristics

- $t_{ccq} = 30\ \text{ps}$
- $t_{pcq} = 50\ \text{ps}$
- $t_{setup} = 60\ \text{ps}$
- $t_{hold} = 70\ \text{ps}$
- $t_{pd} = 35\ \text{ps}$
- $t_{cd} = 25\ \text{ps}$

Setup time constraint:

$$T_c \geq \frac{1}{f_c}$$

Hold time constraint:

$$t_{ccq} + t_{pd} > t_{hold}?$$

$\text{per gate}$

- $t_{pd} = 35\ \text{ps}$
- $t_{cd} = 25\ \text{ps}$
Timing Analysis

Timing Characteristics

- $t_{ccq} = 30$ ps
- $t_{pcq} = 50$ ps
- $t_{setup} = 60$ ps
- $t_{hold} = 70$ ps
- $t_{pd} = 35$ ps
- $t_{cd} = 25$ ps

Set up time constraint:

$T_c \geq (50 + 105 + 60) \text{ ps} = 215 \text{ ps}$

Frequency:

$f_c = 1/T_c = 4.65 \text{ GHz}$

Hold time constraint:

$t_{ccq} + t_{pd} > t_{hold} \? \\
(30 + 25) \text{ ps} > 70 \text{ ps} \? \text{ No!}$
Fixing Hold Time Violation

Add buffers to the short paths:

Timing Characteristics

\[ t_{ccq} = 30 \text{ ps} \]
\[ t_{pcq} = 50 \text{ ps} \]
\[ t_{	ext{setup}} = 60 \text{ ps} \]
\[ t_{	ext{hold}} = 70 \text{ ps} \]

\[ t_{pd} = 35 \text{ ps} \]
\[ t_{cd} = 25 \text{ ps} \]

Setup time constraint:
\[ T_c \geq \]
\[ f_c = \]

Hold time constraint:
\[ t_{ccq} + t_{pd} > t_{hold} \]
Fixing Hold Time Violation

Add buffers to the short paths:

Timing Characteristics

\[ t_{ccq} = 30 \text{ ps} \]
\[ t_{pcq} = 50 \text{ ps} \]
\[ t_{setup} = 60 \text{ ps} \]
\[ t_{hold} = 70 \text{ ps} \]
\[ t_{pd} = 35 \text{ ps} \]
\[ t_{cd} = 25 \text{ ps} \]

Setup time constraint:
\[ T_c \geq (50 + 105 + 60) \text{ ps} = 215 \text{ ps} \]
\[ f_c = 1/T_c = 4.65 \text{ GHz} \]

Hold time constraint:
\[ t_{ccq} + t_{pd} > t_{hold} ? \]
\[ (30 + 50) \text{ ps} > 70 \text{ ps} ? \ Yes! \]
Clock Skew

- The clock doesn’t arrive at all registers at the same time
- Skew is the difference between two clock edges
- Examine the worst case to guarantee that the dynamic discipline is not violated for any register – many registers in a system!
Setup Time Constraint with Clock Skew

- In the worst case, the CLK2 is earlier than CLK1

\( T_c \geq \)
Setup Time Constraint with Clock Skew

- In the worst case, the CLK2 is earlier than CLK1

\[ T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew} \]

\[ t_{pd} \leq \]
Setup Time Constraint with Clock Skew

• In the worst case, the CLK2 is earlier than CLK1

\[ T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}} + t_{skew} \]

\[ t_{pd} \leq T_c - (t_{pcq} + t_{\text{setup}} + t_{skew}) \]
Hold Time Constraint with Clock Skew

- In the worst case, CLK2 is later than CLK1

\[
t_{ccq} + t_{cd} > t_{cd}
\]
Hold Time Constraint with Clock Skew

- In the worst case, CLK2 is later than CLK1

\[ t_{ccq} + t_{cd} > t_{hold} + t_{skew} \]

\[ t_{cd} > \]
Hold Time Constraint with Clock Skew

- In the worst case, CLK2 is later than CLK1

Mathematically:

\[ t_{ccq} + t_{cd} > t_{hold} + t_{skew} \]

\[ t_{cd} > t_{hold} + t_{skew} - t_{ccq} \]
Asynchronous (for example, user) inputs might violate the dynamic discipline.

Diagram:
- CLK input
- D input
- Q output

Cases:
1. Case I: D transitions before clock edge, Q holds.
2. Case II: D and Q transition simultaneously.
3. Case III: D transitions after clock edge, Q latches but is in an unknown state.
Metastability

- Any bistable device has two stable states and a metastable state between them.
- A flip-flop has two stable states (1 and 0) and one metastable state.
- If a flip-flop lands in the metastable state, it could stay there for an undetermined amount of time.
Flip-flop Internals

• Because the flip-flop has feedback, if $Q$ is somewhere between 1 and 0, the cross-coupled gates will eventually drive the output to either rail (1 or 0, depending on which one it is closer to).

A signal is considered metastable if it hasn’t resolved to 1 or 0.

• If a flip-flop input changes at a random time, the probability that the output $Q$ is metastable after waiting some time, $t$, is:

$$P(t_{\text{res}} > t) = \left( \frac{T_0}{T_c} \right) e^{-t/\tau}$$

$t_{\text{res}}$ : time to resolve to 1 or 0

$T_0$, $\tau$ : properties of the circuit
Metastability

• Intuitively:
  – $T_0/T_c$ describes the probability that the input changes at a bad time, i.e., during the aperture time
    $$ P(t_{res} > t) = \left(\frac{T_0}{T_c}\right) e^{-t/\tau} $$
  – $\tau$ is a time constant indicating how fast the flip-flop moves away from the metastable state; it is related to the delay through the cross-coupled gates in the flip-flop
    $$ P(t_{res} > t) = \left(\frac{T_0}{T_c}\right) e^{-t/\tau} $$

• In short, if a flip-flop samples a metastable input, if you wait long enough ($t$), the output will have resolved to 1 or 0 with high probability.
Synchronizers

- Asynchronous inputs ($D$) are inevitable (user interfaces, systems with different clocks interacting, etc.).
- The goal of a synchronizer is to make the probability of failure (the output $Q$ still being metastable) low.
- A synchronizer cannot make the probability of failure 0.
Synchronizer Internals

- A synchronizer can be built with two back-to-back flip-flops.
- Suppose the input D is transitioning when it is sampled by flip-flop 1, F1.
- The amount of time the internal signal D2 can resolve to a 1 or 0 is \((T_c - t_{\text{setup}})\).

![Diagram of synchronizer with flip-flops and timing notations](attachment:diagram.png)
For each sample, the probability of failure of this synchronizer is:

\[
P(\text{failure}) = \left( \frac{T_0}{T_c} \right) e^{-\left( \frac{T_c - t_{\text{setup}}}{\tau} \right)}
\]
Synchronizer Mean Time Before Failure

• If the asynchronous input changes once per second, the probability of failure per second of the synchronizer is simply $P$(failure).
• In general, if the input changes $N$ times per second, the probability of failure per second of the synchronizer is:

$$
P(\text{failure})/\text{second} = \left(\frac{NT_0/T_c}{e^{(T_c - t_{\text{setup}})/\tau}}\right)
$$

• Thus, the synchronizer fails, on average, $1/[P(\text{failure})/\text{second}]$
• This is called the mean time between failures, MTBF:

$$
MTBF = 1/[P(\text{failure})/\text{second}] = \left(\frac{T_c}{NT_0}\right) e^{(T_c - t_{\text{setup}})/\tau}
$$
Example Synchronizer

• Suppose:  \( T_c = \frac{1}{500 \text{ MHz}} = 2 \text{ ns} \quad \tau = 200 \text{ ps} \)
  \( T_0 = 150 \text{ ps} \quad t_{\text{setup}} = 100 \text{ ps} \)
  \( N = 10 \text{ events per second} \)

• What is the probability of failure? MTBF?
  \[ P(\text{failure}) = \]

  \[ P(\text{failure})/\text{second} = \]

  \[ \text{MTBF} = \]
Example Synchronizer

- Suppose: $T_c = 1/500 \text{ MHz} = 2 \text{ ns}$ \hspace{1cm} $\tau = 200 \text{ ps}$
  
  $T_0 = 150 \text{ ps}$ \hspace{1cm} $t_{\text{setup}} = 100 \text{ ps}$

- $N = 10$ events per second

- What is the probability of failure? MTBF?

\[
P(\text{failure}) = \frac{(150 \text{ ps})}{2 \text{ ns}} e^{-\frac{1.9 \text{ ns}}{200 \text{ ps}}} = 5.6 \times 10^{-6}
\]

\[
P(\text{failure})/\text{second} = 10 \times (5.6 \times 10^{-6}) = 5.6 \times 10^{-5} / \text{ second}
\]

**MTBF** \hspace{1cm} $= 1/[P(\text{failure})/\text{second}] \approx 5 \text{ hours}$