Chapter #6: Sequential Logic Design

Contemporary Logic Design

Cross-Coupled NOR Gates

Just like cascaded inverters, with capability to force output to 0 (reset) or 1 (set)

![Diagram of Cross-Coupled NOR Gates](image)
**Observed R-S Latch Behavior**

Very difficult to observe R-S Latch in the 1-1 state
Ambiguously returns to state 0-1 or 1-0
A so-called "race condition"

**Definition of Terms**

**Clock:**
Periodic Event, causes state of memory element to change
- rising edge, falling edge, high level, low level

**Setup Time (T_{su})**
Minimum time before the clocking event by which the input must be stable

**Hold Time (T_{th})**
Minimum time after the clocking event during which the input must remain stable

There is a timing "window" around the clocking event during which the input must remain stable and unchanged in order to be recognized
**Edge triggered device** sample inputs on the event edge

**Transparent latches** sample inputs as long as the clock is asserted

**Timing Diagram:**

- Behavior the same unless input changes while the clock is high

---

**Typical Timing Specifications: Flipflops vs. Latches**

**74LS74 Positive Edge Triggered D Flipflop**

- Setup time
- Hold time
- Minimum clock width
- Propagation delays (low to high, high to low, max and typical)

All measurements are made from the clocking event that is, the **rising edge** of the clock
Typical Timing Specifications: Flipflops vs. Latches

74LS76
Transparent Latch

- Setup time
- Hold time
- Minimum Clock Width
- Propagation Delays:
  - high to low, low to high,
  - maximum, typical
  - data to output
clock to output

Measurements from falling clock edge
or rising or falling data edge

J-K Flipflop

How to eliminate the forbidden state?

Idea: use output feedback to guarantee that R and S are never both one

J, K both one yields toggle

Characteristic Equation:

\[ Q_{+} = Q\bar{K} + QJ \]
**J-K Latch: Race Condition**

Race Condition

**Toggle Correctness:** Single State change per clocking event

**Solution:** Master/Slave Flipflop

---

**Master/Slave J-K Flipflop**

Master Stage

Slave Stage

Sample inputs while clock high

Sample inputs while clock low

Uses time to break feedback path from outputs to inputs!

**Correct Toggle Operation**

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Cik</th>
<th>P</th>
<th>Q</th>
<th>\Q</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Set  | Reset  | T's  | Toggle  | 100 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Master outputs

Slave outputs
Edge-Triggered Flipflops

1's Catching: a 0-1-0 glitch on the J or K inputs leads to a state change! forces designer to use hazard-free logic

Solution: edge-triggered logic

Negative Edge-Triggered D flipflop

4-5 gate delays
setup, hold times necessary to successfully latch the input

Characteristic Equation:
\[ Q^+ = D \]

Negative edge-triggered FF when clock is high

The Problem of Clock Skew

Correct behavior assumes next state of all storage elements determined by all storage elements at the same time

Not possible in real systems!
- logical clock driven from more than one physical circuit with timing behavior
- different wire delay to different points in the circuit

Effect of Skew on Cascaded Flipflops:

Original State: \( Q_0 = 1, Q_1 = 1, \text{In} = 0 \)

Because of skew, next state becomes: \( Q_0 = 0, Q_1 = 0, \) not \( Q_0 = 0, Q_1 = 1 \)
Realizing Circuits with Different Kinds of Flipflops

R-S: \[ Q^+ = S + \overline{R} \bar{Q} \]
D: \[ Q^+ = D \]
J-K: \[ Q^+ = J \overline{Q} + \overline{K} Q \]
T: \[ Q^+ = \overline{T} Q + \bar{T} \bar{Q} \]

Implementing One FF in Terms of Another

Implementing D FF with a J-K FF:
1) Start with K-map of \( Q^+ = f(D, Q) \)
2) Create K-maps for J and K with same inputs (D, Q)
3) Fill in K-maps with appropriate values for J and K to cause the same state changes as in the original K-map

E.g., \( D = Q = 0, Q^+ = 0 \) then \( J = 0, K = \bar{X} \)

Excitation Tables: What are the necessary inputs to cause a particular kind of change in state?

<table>
<thead>
<tr>
<th>Q</th>
<th>Q^+</th>
<th>R</th>
<th>S</th>
<th>J</th>
<th>K</th>
<th>T</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Implementing D FF with a J-K FF:
1) Start with K-map of \( Q^+ = f(D, Q) \)
2) Create K-maps for J and K with same inputs (D, Q)
3) Fill in K-maps with appropriate values for J and K to cause the same state changes as in the original K-map
Implementing J-K FF with a D FF:

1) K-Map of $Q^+ = F(J, K, Q)$

2,3) Revised K-map using D’s excitation table
   its the same! that is why design procedure with D FF is simple!

\[ Q^+ = D = JQ + KQ \]

Resulting equation is the combinational logic input to D to cause same behavior as J-K FF. Of course it is identical to the characteristic equation for a J-K FF.