Lecture 11:
Sequential Networks – Latches and Flip flops

CSE 140: Components and Design Techniques for Digital Systems

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
Flip-flop Components

SR latch (Set-Reset)

Inputs: S, R  
State: (Q, y)

SR=01,  (Q,y) = (0,1)
SR=10,  (Q,y) = (1,0)
SR=11,  (Q,y) = (0,0)
SR = 00 => if (Q,y) = (0,0) or (1,1), the output keeps toggling
<table>
<thead>
<tr>
<th>Id</th>
<th>Q(t) y(t) S R</th>
<th>Q(t1) y(t1) Q(t2)y(t2) Q(t3) y(t3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 0 0</td>
<td>1 1 0 0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0 0 0 1</td>
<td>0 1 0 1 0 1</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 0</td>
<td>1 0 1 0 1 0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 1 1</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>4</td>
<td>0 1 0 0</td>
<td>0 1 0 1 0 1</td>
</tr>
<tr>
<td>5</td>
<td>0 1 0 1</td>
<td>0 1 0 1 0 1</td>
</tr>
<tr>
<td>6</td>
<td>0 1 1 0</td>
<td>0 0 1 0 1 0</td>
</tr>
<tr>
<td>7</td>
<td>0 1 1 1</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>8</td>
<td>1 0 0 0</td>
<td>1 0 1 0 1 0</td>
</tr>
<tr>
<td>9</td>
<td>1 0 0 1</td>
<td>0 0 0 1 0 1</td>
</tr>
<tr>
<td>10</td>
<td>1 0 1 0</td>
<td>1 0 1 0 1 0</td>
</tr>
<tr>
<td>11</td>
<td>1 0 1 1</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>12</td>
<td>1 1 0 0</td>
<td>0 0 1 1 0 0</td>
</tr>
<tr>
<td>13</td>
<td>1 1 0 1</td>
<td>0 0 0 1 0 1</td>
</tr>
<tr>
<td>14</td>
<td>1 1 1 0</td>
<td>0 0 1 0 1 0</td>
</tr>
<tr>
<td>15</td>
<td>1 1 1 1</td>
<td>0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

State Diagram

State Transition

Q y

SR
Q: Which of the following is a good solution to avoid the output from toggling?

A) Avoid the input SR = (0,0)
B) Avoid the input SR = (1,1)
SR Latch Analysis

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

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

<table>
<thead>
<tr>
<th>PS</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q(t) 0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Q(t+1) NS (next state)

Characteristic Expression
Q(t+1) = S(t)+R’(t)Q(t)
SR Latch Symbol

• 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 ($S = R = 1$)
Clocks

- **Clock** -- Pulsing signal for enabling latches; ticks like a clock
- **Synchronous** circuit: sequential circuit with a clock

- **Clock period**: time between pulse starts
  - Above signal: period = 20 ns
- **Clock cycle**: one such time interval
  - Above signal shows 3.5 clock cycles
- **Clock duty cycle**: time clock is high
  - 50% in this case
- **Clock frequency**: 1/period
  - Above: freq = 1 / 20ns = 50MHz;

<table>
<thead>
<tr>
<th>Freq</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 GHz</td>
<td>0.01 ns</td>
</tr>
<tr>
<td>10 GHz</td>
<td>0.1 ns</td>
</tr>
<tr>
<td>1 GHz</td>
<td>1 ns</td>
</tr>
<tr>
<td>100 MHz</td>
<td>10 ns</td>
</tr>
<tr>
<td>10 MHz</td>
<td>100 ns</td>
</tr>
</tbody>
</table>

Sources: TSR, Katz, Boriello, Vahid, Rosing
D Latch

- Two inputs: \( CLK, D \)
  - \( CLK \): controls when the output changes
  - \( D \) (the data input): controls what the output changes to

- Function
  - When \( CLK = 1 \), \( D \) passes through to \( Q \) (the latch is transparent)
  - When \( CLK = 0 \), \( Q \) holds its previous value (the latch is opaque)

- Avoids invalid case when \( Q \neq \text{NOT } Q \)
D Latch Internal Circuit

SR Latch

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
D Latch Internal Circuit

CLK
---

D
---

S
---

R
---

Q
---

Q
---

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>R</th>
<th>S</th>
<th>D</th>
<th>Q</th>
<th>Q prev</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>Q</td>
<td>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 an *edge-triggered* device because it is activated on the clock edge (when $CLK$ rises from 0 → 1)
D Flip-Flop Internal Circuit

- When $CLK = 0$
  - $L_1$ is transparent, $L_2$ is opaque
  - $D$ passes through to $N_1$
- When $CLK = 1$
  - $L_2$ is transparent, $L_1$ is opaque
  - $N_1$ passes through to $Q$
Latch and Flip-flop (two latches)

A latch can be considered as a door

CLK = 0, door is shut
CLK = 1, door is unlocked

A flip-flop is a two door entrance

CLK = 1
CLK = 0
CLK = 1
D Flip-Flop vs. D Latch

CLK
D Q
 Q

D Q
 Q

Q (latch)
Q (flop)
D Flip-Flop vs. D Latch

CLK

D

Q

Q

D

Q

Q

CLK

D

Q (latch)

Q (flop)
D Flip-Flop (Delay)

- **State table**
  - \( PS \times D \):
    - 0 0 1
    - 1 0 1

- **Characteristic Expression**
  - \( Q(t+1) = D(t) \)

<table>
<thead>
<tr>
<th>Id</th>
<th>D Q(t)</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0 1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1 0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1 1</td>
<td>1</td>
</tr>
</tbody>
</table>

What does the equation mean?
Can D flip-flop serve as a memory component?
A. Yes
B. No
JK F-F

State table

<table>
<thead>
<tr>
<th>JK</th>
<th>PS 00 01 11 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1 1 0 1</td>
</tr>
</tbody>
</table>

Q(t+1)

Characteristic Expression

\[ Q(t+1) = Q(t)K'(t)+Q'(t)J(t) \]

Sounds a lot like a latch I know…
Characteristic Expression
\[ Q(t+1) = Q(t)K'(t) + Q'(t)J(t) \]

State table

<table>
<thead>
<tr>
<th>PS</th>
<th>JK</th>
<th>( Q(t+1) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00</td>
<td>0 0 1 1 1</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>1 1 0 0 1</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Diagram of JK F-F flip-flop with input signals and state table.
T Flip-Flop (Toggle)

Characteristic Expression
\[ Q(t+1) = Q'(t)T(t) + Q(t)T'(t) \]

State table

\[
\begin{array}{c|cc}
PS & T & 0 & 1 \\
\hline
0 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]
Using a JK F-F to implement a D and T F-F

The above circuit behaves as which of the following flip flops?
A. D F-F
B. T F-F
C. None of the above
Using a JK F-F to implement a D and T F-F

T flip flop
Enabled Flip-Flops

• **Inputs:** $CLK$, $D$, $EN$
  - The enable input ($EN$) controls when new data ($D$) is stored

• **Function**
  - $EN = 1$: $D$ passes through to $Q$ on the clock edge
  - $EN = 0$: the flip-flop retains its previous state

Sources: TSR, Katz, Boriello & Vahid
Resettable Flip-Flops

- **Inputs:** $CLK$, $D$, $Reset$
- **Function:**
  - $Reset = 1$: $Q$ is forced to 0
  - $Reset = 0$: flip-flop behaves as ordinary D flip-flop
- **Two types:**
  - **Synchronous:** resets at the clock edge only
  - **Asynchronous:** resets immediately when $Reset = 1$
- Asynchronously resettable flip-flop requires changing the internal circuitry of the flip-flop
- Synchronously resettable flip-flop circuit:
- There are also synch/asynch settable FFs

Sources: TSR, Katz, Boriello & Vahid