Lecture 6: 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
Flight attendant call button

- Flight attendant call button
  - Press call: light turns on
    - *Stays on* after button released
  - Press cancel: light turns off
  - Logic gate circuit to implement this?

- SR latch implementation
  - Call=1 : sets Q to 1 and keeps it at 1
  - Cancel=1 : resets Q to 0
SR (Set/Reset) Latch

• SR Latch

• Consider the four possible cases:
  - $S = 1, R = 0$: set output to ‘1’
  - $S = 0, R = 1$: (reset) output to ‘0’
  - $S = 0, R = 0$: store – output should be unchanged
  - $S = 1, R = 1$: Trouble!

(S+Q)’
SR Latch Analysis

- $S = 1, R = 0$:
  \[ Q = 1, \quad \overline{Q} = 0 \]

- $S = 0, R = 1$:
  \[ Q = 0, \quad \overline{Q} = 1 \]
SR Latch Analysis

- **S = 0, R = 0:**

  Assume \(Q_{\text{prev}} = 0, \overline{Q}_{\text{prev}} = 1\)  
  (Memory) \(Q = 0, \overline{Q} = 1\)

```latex
\begin{align*}
\text{Assume} & \quad Q_{\text{prev}} = 0, \overline{Q}_{\text{prev}} = 1 \\
\text{(Memory)} & \quad Q = 0, \overline{Q} = 1
\end{align*}
```
SR Latch Analysis

- $S = 0$, $R = 0$:

What happens if $Q_{\text{prev}} = 0$ and $Q'_{\text{prev}} = 0$?

A. The output $Q$ toggles

B. The output $Q$ remains 0 and $Q'$ changes to 1

C. The output $Q$ becomes 1 and $Q'$ remains 0
SR Latch Analysis

- $S = 1, R = 1$:

$q_2 = 0, \overline{q}_2 = 0$: This is problematic if it is followed by $s_2 = 0, r_2 = 0$.
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
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 $\overline{Q} = \overline{Q}_{prev}$ (memory!)

  \[
  Q_{prev} = 0
  \]

  \[
  Q_{prev} = 1
  \]

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

  \[
  Q_{prev} = 0
  \]

  \[
  Q_{prev} = 1
  \]
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>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;

Sources: TSR, Katz, Boriello, Vahid, Rosing
Clock question

The clock shown in the waveform below has:

A. Clock period of 4ns with 250MHz frequency
B. Clock duty cycle 75%
C. Clock period of 1ns with 1GHz frequency
D. A. & B.
E. None of the above

A. Clock period of 4ns with 250MHz frequency

\[ f = \frac{1}{4\text{ns}} = 250\text{MHz} \]

\[ f = \frac{1}{4 \times 10^{-9}} = 250 \times 10^6 \text{Hz} = 250 \text{MHz} \]
D Latch

- Two inputs: \textit{CLK}, \textit{D}
  - \textit{CLK}: controls \textit{when} the output changes
  - \textit{D} (the data input): controls \textit{what} the output changes to
- Function
  - When \( \text{CLK} = 1 \), \( \text{D} \) passes through to \( \text{Q} \) (the latch is \textit{transparent})
  - When \( \text{CLK} = 0 \), \( \text{Q} \) holds its previous value (the latch is \textit{opaque})
- Avoids invalid case when \( \text{Q} \neq \text{NOT Q} \)
D Latch Internal Circuit

SR Latch

CLK
D
S
R
Q(t+1)
Q'(t+1)

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>S</th>
<th>R</th>
<th>Q(t+1)</th>
<th>Q'(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Q(+)</td>
<td>Q̅(+)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Q(+)</td>
<td>Q̅(+)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Q̅</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Q̅</td>
<td>0</td>
</tr>
</tbody>
</table>
D Latch Internal Circuit

CLK
D

S R Q
Q

CLK D Q
Q

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>D</th>
<th>S</th>
<th>R</th>
<th>Q(t+1)</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></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 to 1)
D Flip-Flop Internal Circuit

- 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 \( Q \)
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 (latch)

CLK  D  Q  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

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

Characteristic Expression

\[ Q(t+1) = D(t) \]

What does the equation mean?
How long does a D-flip flop store a bit before its output can potentially change?

A. Half a clock cycle
B. One clock cycle
C. Two clock cycles
D. There is no minimum time