CSE 140 Lecture 8
Sequential Networks

Professor CK Cheng
CSE Dept.
UC San Diego
Part II. Sequential Networks (Ch. 3)

Memory: Flip flops
Specification: Finite State Machines
Implementation: Excitation Tables

\[ y_i = f_i(S^t, X) \]
\[ s_{i}^{t+1} = g_i(S^t, X) \]
iClicker

Sequential Network vs. Combinational Logic:

• A. Combinational logic can replace any sequential network to realize the same function.

• B. Sequential network uses the same set of logic gates as combinational logic.

• C. Sequential network can implement a CPU.

• D. Sequential network requires a precise clock for its timing.

• E. Most of the above.
Memory Devices

• Memory Storage
• Latches
• Flip-Flops
  – SR, D, T, JK
  – State Tables
  – Characteristic Expressions
Memory Storage: Capacitive Loads

- Fundamental building block of state elements
- Two outputs: $\bar{Q}$, $Q$
- No inputs
 Capacitive Loads

• Consider the two possible cases:
  – $Q = 0$: then $Q' = 1$ and $Q = 0$ (consistent)
  – $Q = 1$: then $Q' = 0$ and $Q = 1$ (consistent)

  – Bistable circuit stores 1 bit of state in the state variable, $Q$ (or $Q'$)

• But there are no inputs to control the state
Given a memory component out of a loop of inverters, the number of inverters has to be

- A. Even
- B. Odd
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 $\overline{Q} = 0$

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

- $S = 0, \ R = 0$: then $Q = Q_{prev}$

  $Q_{prev} = 0$

- $S = 1, \ R = 1$: then $Q = 0$ and $\overline{Q} = 0$
\[ y = (S+Q)' \]
\[ Q = (R+y)' \]
Flip-flop Components

**SR F-F (Set-Reset)**

Inputs: S, R  State: (Q, y)
<table>
<thead>
<tr>
<th>Id</th>
<th>Q(t) y(t) S R</th>
<th>Q(t^1) y(t^1) Q(t^2)y(t^2) Q(t^3) y(t^3)</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 Diagram**
  - **State**: 00, 01, 10, 11
  - **Transition**: 00, 01, 10, 11

- **SR**
  - **Transition**: 00, 01, 11

- **Q y**
  - **Transition**: 10, 11, 10
CASES:
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 changing
Solutions: 1) SR $\neq (0,0)$, or
    2) SR $\neq (1,1)$.

iClicker
Should we choose 1 or 2?
A.1
B.2
CASES:
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 changing
Solutions: 1) SR \(\neq (0,0)\), or
2) SR \(\neq (1,1)\).

State table

\[
\begin{array}{c|ccccc}
\text{inputs} & \text{SR} \\
\hline
\text{PS} & 00 & 01 & 10 & 11 \\
\hline
\text{Q(t)} & 0 & 0 & 0 & 1 & - \\
& 1 & 1 & 0 & 1 & - \\
\end{array}
\]

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

Q(t+1) NS (next state)
SR Latch Analysis

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

\[ Q_{prev} = 0 \]

\[ Q_{prev} = 1 \]

– $S = 1$, $R = 1$: then $Q = 0$ and $\overline{Q} = 0$ (invalid state: $Q \neq \text{NOT} \ Q$)
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 (when $S = R = 1$)
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 $Q \neq \text{NOT } \overline{Q}$
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></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>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# D Latch Internal Circuit

- **CLK**: Clock input
- **D**: Data input
- **R**: Reset input
- **S**: Set input
- **Q**: Output
- **Q̅**: Complement of output

### Timing Table

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>D̅</th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q̅</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>Q0prev</td>
<td>Q̅prev</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>

- When **CLK = 0**, the state of **D** and **CLK** is ignored. The output **Q** is set to **Q0prev**.
- When **CLK = 1**, the state of **D** is captured and passed to the output **Q**.
- The output **Q̅** is the complement of **Q**.
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 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 $Q$
D Flip-Flop vs. D Latch

CLK

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)
Latch and Flip-flops (two latches)

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 (Delay)

State table

<table>
<thead>
<tr>
<th>PS</th>
<th>D</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Characteristic Expression

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

Characteristic Expression

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

State table

<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>

NS = Q(t+1)
Can D flip-flop serve as a memory component?
A. Yes
B. No
JK F-F

State table

\[
\begin{array}{c|cccc}
\text{PS} & \text{JK} & 00 & 01 & 10 & 11 \\
0 & 0 & 0 & 0 & 1 & ? \\
1 & 1 & 1 & 0 & 1 & ? \\
\end{array}
\]

Q(t+1)
JK F-F

Characteristic Expression
$Q(t+1) = Q(t)K'(t) + Q'(t)J(t)$
T Flip-Flop (Toggle)

Characteristic Expression

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

State table:

<table>
<thead>
<tr>
<th>PS</th>
<th>T</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Q(t+1)
Using a JK F-F to implement a D and T F-F

What is the function of the above circuit?

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