Sequential Systems Review

• Combinational Network
  – Output value only depends on input value

• Sequential Network
  – Output Value depends on input value and present state value
  – Sequential network must have some way of retaining state via memory devices.
  – can be synchronous or asynchronous

• Synchronous Sequential Network
  – Use a clock signal in a synchronous sequential system to control changes between states

Sequential System Diagram

• m outputs only depend on k PS bits - Moore Machine
  – REMEMBER: Moore is Less !!

• m outputs depend on k PS bits AND n inputs - Mealy Machine
Controller Block Diagrams

**MEALY MODEL**

- Controller Inputs
- Clock Input
- Combinational Logic (excitation)
- Memory Elements
- Combinational Logic (output)
- Controller Outputs

**MOORE MODEL**

- Controller Inputs
- Clock Input
- Combinational Logic (excitation)
- Memory Elements
- Combinational Logic (optional)
- Controller Outputs

---

**Clock Signal**

- Voltage (typical)
  - $f = \frac{1}{\tau}$
  - $D_c = \frac{P_w}{\tau} \times 100$

- Time
  - $\tau$ - period (in seconds)
  - $f$ - frequency (in Hertz)
  - $D_c$ - duty cycle - ratio of pulse width to period (in %)
  - $P_w$ - pulse width (in seconds)

- Time
  - Voltage
    - millisecond (ms) $10^{-3}$
    - microsecond ($\mu$s) $10^{-6}$
    - nanosecond (ns) $10^{-9}$
    - Kiloherz (kHz) $10^3$
    - Megahertz (MHz) $10^6$
    - Gigahertz (GHz) $10^9$
What is the pulse-width of a 4.77 MHz clock with a 30% duty cycle?

\[ \tau = \frac{1}{f} = (4.77 \times 10^6)^{-1} = 2.096 \times 10^{-7} = 210 \text{ ns} \]
Clock Signal Example

What is the pulse-width of a 4.77 MHz clock with a 30% duty cycle?

\[ \tau = \frac{1}{f} = (4.77 \times 10^6)^{-1} = 2.096 \times 10^{-7} = 210 \text{ ns} \]

\[ P_W = (\text{duty cycle}) \times \tau = (0.3) \times (210 \text{ ns}) = 63 \text{ ns} \]

Storage Elements

(a) level-sensitive storage element (D-latch)

always @(g or d)
begin
if \((g)\) \(q <= d\);
end

latch is transparent to changes on \(d\)

g

d
q

\(q\) follows \(d\) when \(g\) is high

(b) edge-triggered storage element (data flip-flop, or DFF)

always @(posedge \(clk\))
begin
\(q <= d\);
end

capture the \(d\) input

\( clk \)

d
q

\(q\) follows \(d\) on rising edge of \(clk\)
### Other State Elements

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q(t)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Q'(t)</td>
</tr>
</tbody>
</table>

JK can be useful for single bit flags with separate set(J), reset(K) control.

RARELY USED

<table>
<thead>
<tr>
<th>T</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Q(t)</td>
</tr>
<tr>
<td>1</td>
<td>Q'(t)</td>
</tr>
</tbody>
</table>

Can be useful for asynchronous counter design.

RARELY USED

### DFFs are most common

- Most FPGA families only have DFFs
- DFF is fastest, simplest (fewest transistors) of FFs
- Other FF types (T, JK) can be built from DFFs
- We will use DFFs almost exclusively in this class
  - Will always use edge-triggered state elements (FFs), not level sensitive elements (latches).
Synchronous vs Asynchronous Inputs

Synchronous input: Output will change after active clock edge
Asynchronous input: Output changes independent of clock

State elements often have async set, reset control.

D input is synchronous with respect to Clk

S, R are asynchronous. Q output affected by S, R independent of C. Async inputs are dominant over Clk.

D FF with async control

CLK

D input

Q (FF)

Reset

Set
D-Flip-Flop with Async. Set/Res

(a) DFF with asynchronous Set/Reset

<table>
<thead>
<tr>
<th>r</th>
<th>s</th>
<th>clk</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>x</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>↑</td>
<td>d</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1 or 0</td>
<td>q_{old}</td>
</tr>
</tbody>
</table>

always @(posedge clk or posedge r or negedge s)

begin
  if (r) q <= 1’b0;
  else if (!s) q <= 1’b1;
  else q <= d;
end

1. d input is captured on rising clock edge as r, s are negated; q becomes ‘1’.
2. r input is asserted; q becomes ‘0’.
3. s input is asserted; q becomes ‘1’.
4. assertion of s input (a) overrides clock input (b), so q output remains as ‘1’.
D-Flip-Flop with Sync. Set/Res

(b) DFF with synchronous preset/clear

\[ \text{DFF} \]

\[ \text{D} \quad \text{Q} \quad \text{q} \]

\text{pre} \text{ is low-true}

\text{clr} \text{ is high-true}

\text{clr} \text{ has priority over} \text{pre}

\begin{verbatim}
always @(posedge clk)
begin
  \text{q <= d; //lowest priority}
  if (!\text{pre}) \text{q <= 1'b1;}
  if (\text{clr}) \text{q <= 1'b0; //highest priority}
end
\end{verbatim}

---

D-Flip-Flop with Sync. Set/Res

1. \text{d input is captured on rising clock edge as \text{clr}, \text{pre}}
   \text{are negated; \text{q} becomes ‘1’}.  
2. \text{assertion of synchronous input has no effect if}
   \text{does not occur on the active clock edge}
3. \text{clr input is asserted on rising clock edge;}
   \text{q becomes ‘0’}.
4. \text{pre input is asserted on rising clock edge;}
   \text{q becomes ‘1’}.
FF Timing

• Propagation Delay
  – C2Q: Q will change some propagation delay after change in C. Value of Q is based on D input for DFF.
  – S2Q, R2Q: Q will change some propagation delay after change on S input, R input
  – Note that there is NO propagation delay D2Q for DFF!
  – D is a Synchronous INPUT, no prop delay value for synchronous inputs

Setup, Hold Times

• Synchronous inputs (e.g. D) have Setup, Hold time specification with respect to the CLOCK input
• Setup Time: the amount of time the synchronous input (D) must be stable before the active edge of clock
• Hold Time: the amount of time the synchronous input (D) must be stable after the active edge of clock.
Setup, Hold Time

If changes on D input violate either setup or hold time, then correct FF operation is not guaranteed.

Setup/Hold measured around active clock edge.