Lecture Topics

- Today: Sequential Circuits
  (H&H 3.1-3.5)
- Next: continued

Announcements

- Self-study module #3 (this week)
- Project #1 scores sent via email later this week
- Project #2 (due no later than 9/14)
- Project #3 (due no later than 9/21)
Sequential Circuits

- Combinational circuits have no "memory": the output is determined by the current inputs.

- Sequential circuits "remember" the current state: the output is determined by the current inputs and the current state.

Example: vending machine

- A vending machine must remember how many coins and what kinds of coins have been inserted. The behavior is based not only on the current coin inserted (current input), but also on how many and what kinds of coins have been inserted previously (current state).

- These are referred to as finite state machines (finite state automata).
Example FSM

Transition Table

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

Finite State Machine has combinational logic and DFFs (data flip-flops) in a feedback loop.

The DFFs maintain state information.
D Flip-flop

Positive edge-triggered D flip-flop: the value of D is copied into the flip-flop (and becomes the value of Q) when C changes from 0 to 1.

Clock Waveform

- Clock pulse "edges"
  - Rising edge (leading edge, positive edge)
  - Falling edge (trailing edge, negative edge)
- A clock cycle of 25 ns equals a clock rate of 40 MHz
S-R Latch

- The S input sets the latch (forces it to contain a 1)
- The R input resets the latch (forces it to contain a 0)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>$Q^+$</td>
</tr>
<tr>
<td>$R$</td>
<td>$\bar{Q}^+$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$S$</th>
<th>$R$</th>
<th>$Q^+$</th>
<th>$\bar{Q}^+$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0*</td>
<td>0*</td>
</tr>
</tbody>
</table>

*Unpredictable behavior will result if inputs return to 0 simultaneously

NAND and NOR Gates

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$F = \overline{A \cdot B}$

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$F = A + B$
Analysis: SR Latch

- Four cases:
  - $S = 1, R = 0$
  - $S = 0, R = 1$
  - $S = 0, R = 0$
  - $S = 1, R = 1$

Analysis: SR Latch

- if $S = 1, R = 0$ then $Q = 1, Q' = 0$ (Set the latch)
- if $S = 0, R = 1$ then $Q = 0, Q' = 1$ (Reset the latch)
Analysis: SR Latch

- if $S = 0$, $R = 0$ then
  no change to $Q$, $Q'$
  (Keep previous $Q$)

  \[
  Q_{\text{prev}} = 0
  \]

- if $S = 1$, $R = 1$ then
  $Q = 0$, $Q' = 0$
  (Invalid!)

- Cannot set and reset the latch simultaneously

  - Note: $Q \neq \text{NOT } Q'$
S-R Latch

Using NOR gates (active high)

Using NAND gates (active low)

Gated S-R Latch

The Enable signal (usually a clock signal) controls the NAND gates (which control the S and R inputs)
Gated D Latch

Only D signal (instead of S and R signals)

Potential problem: If D changes while the Enable control signal is high, the output will also change.

D Latch Timing Diagram

- Output Q changes only when Clk = 1
  - Q tracks D when Clk = 1
- This latch is level-sensitive since the output is sensitive to the level of the clock
Master-Slave D Flip-Flop

Want changes in Q only on the transition of the Clk signal from 1 \rightarrow 0 \text{ (or from 0 \rightarrow 1)}

When Clock = 1, master latch tracks D; slave latch remains unchanged (Q remains fixed)

When Clock = 0, master latch is unchanged; slave latch tracks \(Q_m\)

Timing of Master-Slave D Flip-Flop

Changes to Q occur only on the negative edge of the Clock
Negative Edge-Triggered DFF

• When CLK is high, the two input latches contain 0, so the Main latch remains in its previous state, regardless of any changes to D.

• On the falling edge of CLK, values in the two input latches will affect the state of the Main latch.

• While CLK is low, D cannot affect the Main latch.

Positive Edge-Triggered DFF
DFF with Clear and Preset

Asynchronous Clear and Preset (active low)

Commercial D Flip-flop

Asynchronous controls (active low):
PRE' – force to 1 (preset)
CLR' – force to 0 (clear)
Commercial D Flip-flop

<table>
<thead>
<tr>
<th>PRE'</th>
<th>CLR'</th>
<th>D</th>
<th>CLK</th>
<th>Q</th>
<th>Q'</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>Not allowed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>Clear</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>Q_{n-1}</td>
<td>Q'_{n-1}</td>
<td>Hold</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>1</td>
<td>Q_{n-1}</td>
<td>Q'_{n-1}</td>
<td>Hold</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>Clocked operation</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>Clocked operation</td>
</tr>
</tbody>
</table>

Uses of Data Flip-flops

- **State info in FSM**
  - Counter
  - Sequencer

- **Data storage**
  - Register with parallel load
  - Register file (collection of registers)
Counters

- Standard Up Counter
  - Sequence starts at 0, counts up by 1
  - Starts over after reaching max

- Example: 2-bit Up Counter
  0, 1, 2, 3, 0, 1, 2, 3, 0, ...

- Variations: count down, count by 2, etc.

3-bit Up Counter

Output: 000, 001, 010, 011, 100, 101, 110, 111, 000, ...
3-bit Up Counter

Output: 000, 001, 010, 011, 100, 101, 110, 111, 000, ...

Three DFFs used to store current state
Combinational logic used to determine next state

Reset: initialize DFFs to state 000
Clock: copy next state into DFFs

Design of 3-bit Up Counter

<table>
<thead>
<tr>
<th>$C_2$</th>
<th>$C_1$</th>
<th>$C_0$</th>
<th>$N_2$</th>
<th>$N_1$</th>
<th>$N_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Three separate functions:

$N_2(C_2, C_1, C_0) = \ldots$

$N_1(C_2, C_1, C_0) = \ldots$

$N_0(C_2, C_1, C_0) = \ldots$
Karnaugh Map for $N_2$

$N_2(C_2, C_1, C_0) = C_2 C_1' + C_2 C_0' + C_2' C_1 C_0$

Karnaugh Map for $N_1$

$N_1(C_2, C_1, C_0) = C_1' C_0 + C_1 C_0'$
Karnaugh Map for $N_0$

$N_0(C_2, C_1, C_0) = C_0'$

Sequencers

- Generate signals for $N$ steps in a cycle
  - $N$ bits of output
  - Exactly 1 bit asserted at each step

Example: 5-bit Sequencer

10000, 01000, 00100, 00010, 00001, 10000, …
3-bit Sequencer

Output: 100, 010, 001, 100, 010, 001, 100, ...

3-bit Sequencer

Output: 100, 010, 001, 100, 010, 001, 100, ...