Sequential Logic:

Clocks

Latches

Flip-Flops
Announcements

• HW 1 is due tomorrow

• Lab 2 will be posted today
  – Includes a new tutorial on Verilog
Encoders

• **Opposite of decoders**

• **Binary encoders:** $2^n$ inputs and $n$ outputs

![Diagram of Decoder and Encoder]
4-to-2 Encoder

Exactly one input is asserted at any given time

<table>
<thead>
<tr>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>Y1</th>
<th>Y0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
4-to-2 Encoder

Exactly one input is asserted at any given time

<table>
<thead>
<tr>
<th>I3</th>
<th>I2</th>
<th>I1</th>
<th>I0</th>
<th>Y1</th>
<th>Y0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Combinational vs. Sequential Circuits

**Combinational Circuit**

- **Combinational Logic**
  - Output depends only on current inputs

**Sequential Circuit**

- **Combinational Logic**
- **State**
  - Output depends on current inputs plus past history
  - Includes memory
Sequential Circuits

• Outputs depend on inputs and state variables

• The state variables embody the past history of the circuit
  – Storage elements hold the state variables

• A clock periodically advances the circuit
Basic Storage Element
Bistable Element

• Basic storage element
• Inverters with outputs connected to inputs

• What does the circuit do?
Bistable = Two Stable States

- Two inverters with outputs connected to inputs
- Bistable element stores a “given” value indefinitely (as long as powered)

How to change the stored value? (needs extra inputs)
Revisit 2-input NOR and NAND

- **NOR / NAND can implement an inverter**

  ![NOR Inverter Diagram]

- **NOR / NAND has a null element**

  ![NAND Null Element Diagram]
S-R Latch (Set-Reset Latch)

Set (S) and Reset (R) inputs allow (re)setting stored value

Hold: both NORs act as inverters, forming a bi-stable element

Reset/Set: the null element of NOR ("1") is in effect

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>(Q_{\text{next}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

\(Q_{\text{next}}\) is new state of \(Q\) when R or S changes
S-R Latch (Set-Reset Latch)

Boolean expression for $Q_{\text{next}}$ in terms of $R$, $S$, $Q$:

$$Q_{\text{next}} = (R + QN)'$$
$$Q_{\text{next}} = (R + (S + Q)')'$$
$$Q_{\text{next}} = R'\cdot(S + Q)$$
$$Q_{\text{next}} = R'\cdot S + R'\cdot Q$$

When $SR=00$, it holds (latches) the previous state.
Avoid $SR = 11$
Exercise: S-R Latch

- Suppose the S-R latch is initialized with $Q=0$, write down the next state given the following input sequence:
  
  $S=1, R=0 \rightarrow Q = ?$
  
  $S=0, R=0 \rightarrow Q = ?$
  
  $S=0, R=1 \rightarrow Q = ?$
Instability (Avoid SR=11)

• SR=11 may cause instability of the internal state
  1. When SR=11 $\rightarrow$ Q=QN=0
  2. If inputs subsequently change to SR=00, output may be undefined
D Latch

• **D latch**: builds on S-R latch where S and R cannot be both 1
  – Output “follows” input

• **D latch captures input data (what to set)** when certain condition holds (when to set)

• **Operates in two modes**
  – Open (or enabled): input flows through to output
  – Closed (or disabled): output does not change
D Latch

- When C is enabled, Q output follows D input
- When C is disabled, Q output retains last state

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>R</th>
<th>S</th>
<th>Q_{next}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td>(Last) Q</td>
</tr>
</tbody>
</table>

Q_{next} is new state of Q when D or C changes
D Latch

- When C is enabled, Q output follows D input
- When C is disabled, Q output retains last state

\[
\begin{align*}
\text{R (RESET)} & = C \cdot D' \\
\text{S (SET)} & = C \cdot D
\end{align*}
\]

Table:

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>R</th>
<th>S</th>
<th>Q_{next}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>(Last) Q</td>
</tr>
</tbody>
</table>

R = C \cdot D'
S = C \cdot D
Multi-Bit Latch

- Simultaneously latches multiple bits
- “Latch” may refer to 1 bit latch or multi-bit one
Clock

• An input to a sequential circuit that changes output and state values at a predetermined rate

Positive (rising) edge

Negative (falling) edge

• **Triggering edge**: Transition of the clock that captures input data
  – By default, we use **positive rising edge** (L→H) as the triggering edge in this course
  – **Clock tick**: Occurrence of a triggering edge
Clock Period and Frequency

- **Clock Period (cycle time):** Time between successive transitions in the same direction (e.g., L→H)
  - e.g., 1ms, 2ns, 500ps
- **Clock Frequency:** \(1/\text{period}\)
  - e.g., 1kHz, 500MHz, 2GHz

![Diagram showing clock period and frequency](https://via.placeholder.com/150)

**Diagram:**
- \(t_H\) and \(t_L\) are the transition times.
- \(t_{\text{per}}\) is the period.
- Frequency is given by \(1/t_{\text{per}}\).
• **D latch is level sensitive**
  – Captures input D when the latch is enabled (open)
  – Holds the previous state otherwise (closed)
Flip-Flop

• An edge-sensitive storage element

• D flip-flop (FF): Two D latches back-to-back
  – Inputs: D (Data) and Clock
  – Output: Q

• Captures input (copy D to Q) on triggering edge of clock
  – Otherwise (when clock steady at 0 or 1) it retains its value
D Flip-Flop

\[ \text{D} \rightarrow \text{Q} \]

<table>
<thead>
<tr>
<th>(D)</th>
<th>(\text{CLK})</th>
<th>(Q_{\text{next}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>↑</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>↑</td>
<td>1</td>
</tr>
<tr>
<td>(\times)</td>
<td>0</td>
<td>(Last) (Q)</td>
</tr>
<tr>
<td>(\times)</td>
<td>1</td>
<td>(Last) (Q)</td>
</tr>
</tbody>
</table>

\(Q_{\text{next}}\) is new state of \(Q\) after a triggering clock edge occurs (rising edge in this case)
D Flip-Flop Timing

Input D copied to Q on the rising edge of the clock
Register

- Collection of FFs operating off common clock
- A single D flip-flop is a 1-bit register
Before Next Class

• H&H 4.1-4.5 (skip VHDL parts), 5.4

Next Time

More Sequential Logic
Verilog