Lecture 22: Sequential Circuits

- Latches:
  - SR
  - D
  - JK
- Flip-Flops
- Memory

Sequential Circuits

- In sequential circuits, new state depends on not only the input, but also on the previous state.
- Devices like registers require components that will maintain their current value until an external stimulus causes it to change.

S-R Latch (Clocked)

\[
\begin{array}{ccc}
S & Ck & R \\
& Q & \\
\end{array}
\]

- S = Set
- R = Reset
- Ck = Clock (enable)
- Q = state of the latch

2 inputs (+ clock pulse)
2 outputs which are complements of each other.

The state of the latch is allowed to change with time (allowed to change when clock pulse goes to one)

Characteristic Table, SR Latch

- Characteristic table specifies next state when inputs and present state are known:

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q(t+)</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>not allowed</td>
</tr>
</tbody>
</table>

\( t = \) time clock pulse is 1
\( Q(t+) \) – state of the latch just after the clock pulse has changed to 1
\( Q(t-) \) – state of the latch just prior to the change
Excitation Table, SR Latch

- Excitation table: lists required input combinations for a given change of state.
  a) create column for each possible state change
  b) use characteristic table to figure out inputs.

<table>
<thead>
<tr>
<th>Q(t-)</th>
<th>Q(t+)</th>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>x</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</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>x</td>
<td>0</td>
</tr>
</tbody>
</table>

SR - Summary

- When the clock pulse changes from 0 to 1, AND
  - R=S=0, the state does not change. I.e., Q and notQ are whatever they were before the clock changed, i.e. the latch remembers the previous state (1 bit memory)
  - R=0, S=1, Q becomes 1 (latch is set)
  - R=1, S=0, Q becomes 0 (latch is reset)
- So, set latch in a certain state by passing inputs 01 or 10. Once in a state, keep it there by sending 00.
- Inputs (S&R) get passed to circuit only when the clock pulse = 1.

SR Latch Implementation

- Clock – either circuitry or driven by quartz crystal oscillator.
  - goal is to provide a voltage signal that goes high or low at some regular rate. This causes all latches in the system to change at defined intervals and in unison (synchronously).
- Goal – produce a circuit that can remember.

S-R Timing Diagram

- A convenient way to visualize sequential circuits is with a timing diagram:
**Problem with S-R**

- What happens when $S=R=1$?
  - Circuit becomes non-deterministic when both inputs are one.
- When $S=R=1$, $Q = \text{not}Q = 0$
- So what happens if it drops back to $S=R=0$?
  - A state with both outputs $= 0$ is inconsistent: it forces both gates to have 2 zeros as input, which if true would produce 1 as an output, not zero.
  - Results in a race condition – if either input drops back to zero before the other, the remaining one wins (it forces the state)
  - If both drop to zero at the same time (unlikely), the latch jumps to a stable state at random.
- We need a circuit where $S=R=1$ is not possible!

**Clocked D Latch**

- Only one input (besides clock); the input goes into one AND gate and it’s complement into the other AND gate.
- When clock=1, the current value of $D$ is sampled and stored in the latch.
- When clock=0, the latch “remembers” (holds its last value)
- This is how most memories work (this is a one-bit memory)

**Characteristic Table, D Latch**

- Characteristic table specifies next state when inputs and present state are known:

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

$t =$ time clock pulse is 1

$Q(t+)$ – state of the latch just after the clock pulse has changed to 1

**D Latch as Memory**

- To “remember” the last output (read memory) either:
  - disable the clock input or,
  - feed output back into the input so that clock pulses keep the state of the latch unchanged.
- To write memory, set or clear the D input to the value and the next clock pulse will load the current value of D into memory.
**J-K Latch**

- Like an S-R latch, except 1-1 is also a valid input.

<table>
<thead>
<tr>
<th>J</th>
<th>K</th>
<th>Q(+)</th>
<th>Q(+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Q(t-)</td>
<td>Q(t-)</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>Q(t-)</td>
<td>Q(t+)</td>
</tr>
</tbody>
</table>

- 1-1 is the toggle function. It outputs the complement of the state before the clock pulse.
- Versatile and useful. Used to implement counters.

**Latches vs. Flip-Flops**

- A Latch constantly computes its result (in the case of D, stores its input) while the CK input is 1. I.e., it acts as a combinational circuit while CK=1.
- A Flip-Flop is an edge triggered device: the value stored in the flip flop is only changed when CK changes from 0 to 1 (rising edge triggered) or 1 to 0 (falling edge triggered)
- If the input changes while CK=1, a latch and a flip-flop produce different outputs.

**Pulse Generator**

- Figure 3-25 from Tannenbaum

**D Latches and Flip-Flops**

- Figure 3-27, Tannenbaum
- Figure 3-38, Tannenbaum
Memory

• Figure 3-29 from Tannenbaum

Random Access Memory

• RAM (Random Access Memory)
  – static – contents are retained as long as power is kept on. Constructed using circuits like our D flip-flop.
  – dynamic (DRAM) – array of cells containing one transistor and a tiny capacitor. Capacitor holds a charge – charged = 1, discharged = 0. Charge leaks out over time so DRAM must be refreshed.
• Why use DRAM? higher capacity
• DRAM is slower though.
• Typical configuration:
  – DRAM main memory
  – Static RAM cache

Read Only Memory

• Hold programs and data that are never changed and that must remain stored even if the power is turned off.
• ROM – data is inserted during manufacture. Cheaper than RAM (in large volume)
• PROM – Programmable ROM. This can be programmed once.
• EPROM – Erasable PROM – can be erased using UV light and then reprogrammed.
• EEPROM – erased using pulses, can be reprogrammed in place. Slower, more expensive.
• Flash Memory – used as film in digital cameras. Very fast.
Memory Type Comparison

- Figure 3-32, Tannenbaum