Overview

- Circuits require memory to store intermediate data
- Sequential circuits use a periodic signal to determine when to store values.
  - A clock signal can determine storage times
  - Clock signals are periodic
- Single bit storage element is a flip flop
- A basic type of flip flop is a latch
- Latches are made from logic gates
  - NAND, NOR, AND, OR, Inverter
The story so far ...

- Logical operations which respond to combinations of inputs to produce an output.
  - Call these combinational logic circuits.

- For example, can add two numbers. But:
  - No way of adding two numbers, then adding a third (a sequential operation);
  - No way of remembering or storing information after inputs have been removed.

- To handle this, we need sequential logic capable of storing intermediate (and final) results.
Sequential Circuits

Inputs → Combinational circuit → Outputs

Clock
a periodic external event (input)
synchronizes when current state changes happen
keeps system well-behaved
makes it easier to design and build large systems
Cross-coupled Inverters

- A stable value can be stored at inverter outputs

State 1

State 2
S-R Latch with NORs

- S-R latch made from **cross-coupled** NORs
- If $Q = 1$, set state
- If $Q = 0$, reset state
- Usually $S=0$ and $R=0$
- $S=1$ and $R=1$ generates unpredictable results

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
S-R Latch with NANDs

- Latch made from **cross-coupled** NANDs
- Sometimes called S'-R' latch
- Usually S=1 and R=1
- S=0 and R=0 generates unpredictable results

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</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>0</td>
<td>1</td>
</tr>
</tbody>
</table>
S-R Latches

(a) Logic diagram

(b) Function table

---

ENGIN112 L19: Sequential Circuits: Latches
Occasionally, desirable to avoid latch changes

- **C = 0** disables all latch state changes
- Control signal enables data change when **C = 1**
- Right side of circuit same as ordinary S-R latch.
Latch operation enabled by C

Input sampling enabled by gates

Outputs change when C is low: RESET and SET
Otherwise: HOLD

Latch is **level-sensitive**, in regards to C

Only stores data if C' = 0

**NOR S-R Latch with Control Input**
D Latch

- $Q_0$ indicates the previous state (the previously stored value)

<table>
<thead>
<tr>
<th>D</th>
<th>C</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>$Q_0$</td>
<td>$Q_0'$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>C</th>
<th>Q</th>
<th>Q'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$Q_0$</td>
<td>$Q_0'$ Store</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1  Reset</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0  Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1  Disallowed</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>0</td>
<td>$Q_0$</td>
<td>$Q_0'$ Store</td>
</tr>
</tbody>
</table>
D Latch

Input value D is passed to output Q when C is high
Input value D is ignored when C is low
D Latch

- Z only changes when E is high
- If E is high, Z will follow X
D Latch

The D latch stores data indefinitely, regardless of input D values, if C = 0

Forms basic storage element in computers
Symbols for Latches

- SR latch is based on NOR gates
- S’R’ latch based on NAND gates
- D latch can be based on either.
- D latch sometimes called transparent latch
Latches are based on combinational gates (e.g. NAND, NOR)

Latches store data even after data input has been removed

S-R latches operate like cross-coupled inverters with control inputs (S = set, R = reset)

With additional gates, an S-R latch can be converted to a D latch (D stands for data)

D latch is simple to understand conceptually
  • When C = 1, data input D stored in latch and output as Q
  • When C = 0, data input D ignored and previous latch value output at Q

Next time: more storage elements!