1. Introduction: What is a sequential circuit?

“A circuit whose output depends on current inputs and past outputs”

“A circuit with memory”

Memory: a key parameter is Time

$y_i = f_i(s_i, x)$

$s_{i+1} = g_i(s_i, x)$
1. Introduction: Sequential Network Key features

Main Theme: Timing

**Present time** = \( t \) and **next time** = \( t+1 \)

Timing constraints to separate the present and next times.

- \( y_l = f(S^l, X) \)
- \( s_{l+1} = g(S^l, X) \)

1. Introduction: Different Types of Memory/Storage

- Typical Computer Memory Hierarchy
- Tradeoff between speed (latency) and size
  - Storage density (area/bit)
  - Power (power/bit)
- Size relates to (bus) 6-7 hrs
  - Register File (Static Memory - SRAM)
  - Cache Memory (Static Memory - SRAM)
  - Main Memory (Dynamic Memory - DRAM)
  - NVM Main Memory (Non-Volatile Memory - e.g. Flash)

1. Introduction: Fundamental Memory Mechanism

**Capacitive Load**

- Fundamental building block of sequential circuits
- Two outputs: \( \overline{Q}, Q \)
- There is a feedback loop!
  - In a typical combinational logic, there is no feedback loop.
- No inputs
1. Introduction: Capacitive Loads

- Consider the two possible cases:
  - \( Q = 0 \) then \( Q' = 1 \) and \( Q = 0 \) (consistent)
  - \( Q = 1 \) then \( Q' = 0 \) and \( Q = 1 \) (consistent)
- Bistable circuit stores 1 bit of state in the state variable, \( Q \) (or \( Q' \))
- Hold the value due to capacitive charges and feedback loop strengthening
- But there are no inputs to control the state

Example

Q. Given a memory component made out of a loop of inverters, the number of inverters in the loop has to be
A. Even
B. Odd
C. No constraints

UC San Diego

1. Introduction: Memory Storage Mechanism

- Word line (WL) to access cell
- Read – measure voltage difference between B+ and B-
- Write – force values of B+ and B- which may flip the cell
- Static Random Access Memory - SRAM

2. Basic Building Blocks

- Latches (Level Sensitive)
  - SR Latches, D Latches
- Flip-Flops (Edge Triggered)
  - D FFs, (JK FFs, T FFs)
- Examples of Memory Modules
  - Registers, Shift Registers, Pattern Recognizers, Counters, FIFOs
2. Basic Building Blocks: Flight attendant call button

- Flight attendant call button:
  - Press call: light turns on
    - Stays on after button released
  - Press cancel: light turns off
  - Logic gate circuit to implement this?

- SR latch implementation
  - Call=1: sets Q to 1 and keeps it at 1
  - Cancel=1: resets Q to 0

2. Basic Building Blocks: SR (Set/Reset) Latch

- SR Latch

- Consider the four possible cases:
  - $S = 1, R = 0$
  - $S = 0, R = 1$
  - $S = 0, R = 0$
  - $S = 1, R = 1$

2. Basic Building Blocks: SR Latch Analysis

- $S = 1, R = 0$: then $Q = 1$ and $\overline{Q} = 0$

- $S = 0, R = 1$: then $Q = 0$ and $\overline{Q} = 1$

- $S = 1, R = 1$: then $Q = 0$ and $\overline{Q} = 0$

- $S = 0, R = 0$: then $Q = Q_{prev}$
  - $Q_{prev} = 0$
  - $Q_{prev} = 1$
2. Basic Building Blocks: SR Latch

\[ y = (S+Q)^* \]

\[ Q = (R+y)^* \]

Truth table of SR latch with incremental steps in time

<table>
<thead>
<tr>
<th>State</th>
<th>Present State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>00</td>
</tr>
<tr>
<td>5</td>
<td>01</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>00</td>
</tr>
<tr>
<td>9</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>00</td>
</tr>
<tr>
<td>13</td>
<td>01</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
</tr>
</tbody>
</table>

Inputs: S, R
State: (Q, y)

<table>
<thead>
<tr>
<th>id</th>
<th>S</th>
<th>R</th>
<th>Q</th>
<th>y</th>
<th>Q</th>
<th>y</th>
<th>Q</th>
<th>y</th>
<th>Q</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<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>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

"State Table" of SR latch

<table>
<thead>
<tr>
<th>( SR )</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>11</td>
<td>01</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>01</td>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>01</td>
<td>10</td>
<td>00</td>
</tr>
</tbody>
</table>

Cases:
- SR=00: (Q,y) = (0,1)
- SR=01: (Q,y) = (0,1)
- SR=10: (Q,y) = (1,0)
- SR=11: (Q,y) = (0,0)

\( SR=00: (Q,y) \) does not change if \( (Q,y)=(1,0) \) or \( (0,1) \)

However, when \( (Q,y) = (0,0) \) or \( (1,1) \), the output keeps changing

Remark: To verify the design, we need to enumerate all combinations.
State Table and State Diagram

State Table

<table>
<thead>
<tr>
<th>SR</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qy</td>
<td>11</td>
<td>01</td>
<td>10</td>
<td>00</td>
</tr>
</tbody>
</table>

CASES:
SR=01: (Q,y) = (0,1)
SR=10: (Q,y) = (1,0)
SR=11: (Q,y) = (0,0)
SR= 00: (Q,y) does not change if (Q,y)=(1,0) or (0,1)
However, when (Q,y) = (0,0) or (1,1), the output keeps changing

Q. Suppose that we can set the initial state (Q,y)=(0,1). To avoid the SR latch output from toggling or behaving in an undefined way which input combinations should be avoided:
A. (S, R) = (0, 0)
B. (S, R) = (1, 1)
C. None of the above

SR Latch

We set the initial state (Q,y)=(0,1) or (1,0). To avoid the state (Q,y)=(0,0) or (1,1), we block the input SR=11.
Thus, without input SR=11, the state can only be (Q,y)=(0,1) or (1,0).

The only way to reach state (Q,y)=(0,0) or (1,1) is via edge labeled SR=11.