Sequential Logic

So far:
- "Combinational" logic
  - Composites of AND/OR/NOT or just NAND
  - Circuits are all DAGs — no loops or feedback
  - We assume output is independent of time — no state

Today:
- "Sequential" logic
  - Based on a feedback device, the data flip-flop or DFF
  - Output is clocked — we care about state at time t

Most of today's slides credit to www.nand2tetris.org

The Clock

- In our jargon, a clock cycle = tick-phase (low), followed by a tock-phase (high)
- In real hardware, the clock is implemented by an oscillator

Flip-Flop

- A fundamental state-keeping device
- For now, let us not worry about the DFF implementation
- Memory devices are made from numerous flip-flops, all regulated by the same master clock signal
- Notational convention:

1-Bit Register ("Bit")

Objective: build a storage unit that can:
(a) Change its state to a given input
(b) Maintain its state over time (until changed)

Implementation

Bit Register

Interface

Implementation
Multi-Bit Register

- If \( \text{load}(t-1) \) then \( \text{out}(t) = \text{in}(t-1) \)
- Else \( \text{out}(t) = \text{out}(t-1) \)

**Register's width:** a trivial parameter
- Read logic
- Write logic

Random Access Memory (RAM)

\[ \text{load} \]
\[ \text{in} \]
\[ \text{out} \]
\[ \text{address} \]
\[ \text{RAM} \]

- \( \text{load}(0 \text{ to } n-1) \)
- Direct Access Logic

RAM Interface

\[ \text{load} \]
\[ \text{in} \]
\[ \text{out} \]
\[ \text{address} \]

<table>
<thead>
<tr>
<th>Chip name</th>
<th>( n )</th>
<th>( W )</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM0</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>RAM64</td>
<td>64</td>
<td>6</td>
</tr>
<tr>
<td>RAM128</td>
<td>512</td>
<td>9</td>
</tr>
<tr>
<td>RAM256</td>
<td>1024</td>
<td>12</td>
</tr>
<tr>
<td>RAM4K</td>
<td>4096</td>
<td>16</td>
</tr>
<tr>
<td>RAM8K</td>
<td>8192</td>
<td>16</td>
</tr>
</tbody>
</table>

Building Up

Counter

Needed: a storage device that can:
(a) set its state to some base value
(b) increment the state in every clock cycle
(c) maintain its state (stop incrementing) over clock cycles
(d) reset its state

- Typical function: program counter
- Implementation: register chip + some combinational logic.

Recap: Sequential vs Combinational Logic

- **Combinational chip**
  - \( \text{in} \)
  - \( \text{out} \)
  - \( \text{comb. logic} \)
  - \( \text{out} = \text{some function of } \text{in} \)

- **Sequential chip**
  - \( \text{in} \)
  - \( \text{out} \)
  - \( \text{comb. logic} \)
  - \( \text{out} = \text{some function of } \text{in}, \text{load}(t-1), \text{reset}(t-1) \)