Sequential Circuits
Circuits

- **Circuits** are formed by linking gates (or other circuits) together
  - Inputs and outputs
    - Link output of one gate to input of another
    - Some circuits have multiple inputs and/or outputs
  - **Combinational** circuits: outputs are a boolean function of inputs
    - Not time-dependent
    - Used for computation
  - **Sequential** circuits: output is dependent on previous outputs
    - Time-dependent
    - Used for memory
Question: How do we make a circuit “remember” something?

- Answer: Create a feedback loop!
- Creates a “storage” circuit, often called a latch
- Truth table must include previous state
- Alternatively, draw a timing diagram
  - Shows how input/output signals change with respect to time
  - Given input signals in diagram, we can determine output signals
SR AND-OR latch

S = “set”   R = “reset”

Event a (S on)

Event b (S off)

Event c (R on)

Event d (R off)
SR AND-OR latch

**S = “set”  R = “reset”**

<table>
<thead>
<tr>
<th>S</th>
<th>R</th>
<th>Q (old)</th>
<th>Q (new)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</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>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

← “reset”

← “set”

← the R “overrides” the S in this circuit
SR NOR latch

Works similarly to AND-OR, but requires one fewer gate (and it is a universal gate!)

Question: What happens if we turn both R and S on at the same time?

Disallow S=1, R=1 because Q’ ≠ !Q
Aside: oscillation

The circuit will be unstable; it begins to oscillate back and forth as quickly as possible, generating heat and eventually melting the connection and destroying the circuit.

Question: What happens if we turn both $R$ and $S$ off at the same time (from the position previously disallowed)?
D latch

From “Code” book: $S = “Save that bit!”$

- As long as $S$ is on, $Q$ reflects the value of $D$.
- When $S$ turns off, $Q$ is "frozen" and retains its previous value.
- $D$ can change while $S$ is off with no change in $Q$. 
Clocks

- Provide oscillating signal
- Often used as “set” signal for latches
- Keeps computation and memory in sync
- Clocked latches are called **flip-flops**
- The clock period is the inverse of the frequency (measured in **hertz**)
- The length of a clock period determines the minimum time an instruction takes to execute

\[
\text{Clock period} = \frac{1}{f}
\]
Flip-flop types

- **SR**: “set-reset”
- **D**: “data” bit + clock
- **T**: “toggle”
- **JK**: like SR + T (toggle when S=1, R=1)
  - J is S, K is R
- Any of these can be used to build the others
- Also can be built from basic logic gates in multiple ways
Registers

- **Registers**: arrays of flip-flops with a single set/clock input
- Connected by **buses** (groups of wires) to other components
- **Edge triggering** allows computation to stabilize before results are saved
Register files

- **Register files**: multiple registers w/ read/write ports
  - Use multiplexors and decoders to differentiate
Register files

- Register files: multiple registers w/ read/write ports
  - Use multiplexors and decoders to differentiate
Memory

- Memory: multiple flip-flops with address input
  - Random access memory (RAM) - can access any address at any time
  - Use decoder (translates n-bit number to $2^n$ “set” signals) to write data
  - Use selector (multiplexor) to read data

Single 8-element RAM array (3-bit addresses)

Abstraction of multiple RAM arrays
CPUs

- Combine ALU with registers and memory to make CPUs

(next time!)