Combinational Logic Design Process

Step 1: Capture the function
Create a truth table or equations, whichever is most natural for the given problem, to describe the desired behavior of the combinational logic.

Step 2: Convert to equations
This step is only necessary if you captured the function using a truth table instead of equations. Create an equation for each output by ORing all the minterms for that output. Simplify the equations if desired.

Step 3: Implement as a gate-based circuit
For each output, create a circuit corresponding to the output's equation. (Sharing gates among multiple outputs is OK optionally.)

Example: Three 1s Detector
• Problem: Detect three consecutive 1s in 8-bit input: abcdefgh

Example: Number of 1s Count
• Problem: Output in binary on two outputs yz the number of 1s on three inputs

Example: Prime Number Detector
• Problem: Create a 4-bit prime number detector. The circuit has four inputs, N3, N2, N1, and N0 that correspond to a 4-bit number (N3 is the most significant bit) and one output labeled P that outputs a 1 when the input is a prime number, 0 otherwise.
Example: Squared Number Detector

Problem: Create a 4-bit squared number detector. The circuit has four inputs, N3, N2, N1, and N0 that correspond to a 4-bit number (N3 is the most significant bit) and one output labeled S that outputs a 1 when the input is the square of a positive integer, 0 otherwise.

Number of Possible Boolean Functions

- How many possible functions of 2 variables?
  - $2^2 = 4$ possible functions
- N variables
  - $2^N$ possible functions

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

More Gates: Example Uses

- Aircraft lavatory sign example
  - $S = (abc)'$
- Detecting all 0s
  - Use NOR
- Detecting equality
  - Use XNOR
- Detecting odd # of 1s
  - Use XOR
  - Useful for generating “parity” bit common for detecting errors

Completeness of NAND

- Any Boolean function can be implemented using just NAND gates. Why?
  - Need AND, OR, and NOT
  - NOT: 1-input NAND (or 2-input NAND with inputs tied together)
  - AND: NAND followed by NOT
  - OR: NAND preceded by NOTs
- Likewise for NOR

Decoders and Muxes

- Decoder: Popular combinational logic building block, in addition to logic gates
  - Converts input binary number to one high output
  - 2-input decoder: four possible input binary numbers
    - So has four outputs, one for each possible input binary number
    - Internal design
      - AND gate for each output to detect input combination
      - Decoder with enable e
        - Outputs all 0 if e=0
        - Regular behavior if e=1
  - n-input decoder: 2^n outputs

More Gates

- NAND: Opposite of AND (“NOT AND”)
- NOR: Opposite of OR (“NOT OR”)
- XOR: Exactly 1 input is 1, for 2-input XOR. (For more inputs -- odd number of 1s)
- XNOR: Opposite of XOR (“NOT XOR”)
**Decoder Example**

- **New Year’s Eve Countdown Display**
  - Microprocessor counts from 59 down to 0 in binary on 6-bit output
  - Want illuminate one of 60 lights for each binary number
  - Use 6x64 decoder
  - 4 outputs unused

**Multiplexor (Mux)**

- **Mux** Another popular combinational building block
  - Routes one of its N data inputs to its one output, based on binary value of select inputs
  - 4 input mux ➔ needs 2 select inputs to indicate which input to route through
  - 8 input mux ➔ 3 select inputs
  - N inputs ➔ \( \log_2(N) \) selects
  - Like a railyard switch

**Mux Internal Design**

- **2x1 mux**
- **4x1 mux**

**Mux Example**

- City mayor can set four switches up or down, representing his/her vote on each of four proposals, numbered 0, 1, 2, 3
- City manager can display any such vote on large green/red LED (light) by setting two switches to represent binary 0, 1, 2, or 3
- Use 4x1 mux

**Muxes Commonly Together -- N-bit Mux**

- Ex: Two 4-bit inputs, A (a3 a2 a1 a0), and B (b3 b2 b1 b0)
  - 4-bit 2x1 mux (just four 2x1 muxes sharing a select line) can select between A or B

**N-bit Mux Example**

- Four possible display items
  - Temperature (T), Average miles-per-gallon (A), Instantaneous mpg (I), and Miles remaining (M) — each is 8-bits wide
  - Choose which to display using two inputs x and y
  - Use 8-bit 4x1 mux
Additional Considerations

Schematic Capture and Simulation

- Schematic capture
  - Computer tool for user to capture logic circuit graphically
- Simulator
  - Computer tool to show what circuit outputs would be for given inputs
    - Outputs commonly displayed as waveform

Additional Considerations

Non-Ideal Gate Behavior -- Delay

- Real gates have some delay
  - Outputs don't change immediately after inputs change

Additional Considerations

Encoders

- Encoder: Combinational logic building block with opposite functionality of decoder
  - Outputs binary encoding for input signal that is 1
  - 4x2 encoder would have four inputs and 2 outputs

- What if two inputs are 1?
  - Can use a priority encoder
    - Gives priority to the highest input that is 1, and outputs binary encoding for that input
    - Example: if d3=1 and d1=1, will output e0=1 and e1=0 because d3 has priority

Design Challenge

- Design Challenge
  - Design a 4x2 encoder using AND, OR, and NOT gates.

Due Next Lecture (as announced in class)
1 point extra credit (Homework)