Introduction to Digital Design
Week 5: Design Process, More Gates

Overview
• Combinational design process
  – Translate from equation (or table) to circuit.
• More gates
  – NAND, NOR, XOR, XNOR.
• Muxes and decoders
  – Decoder: converts input binary number to one high output.
  – Multiplexor: routes one of its N data inputs to its one output, based on binary value of select inputs.
• Additional considerations
  – Circuit delay and critical path.
  – Active low Inputs.
  – Schematic capture and simulation.

Combinational Logic Design Process

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1: Capture the function</td>
<td></td>
</tr>
<tr>
<td>2A: Create equations</td>
<td></td>
</tr>
<tr>
<td>2B: Implement as a gate-based circuit</td>
<td></td>
</tr>
</tbody>
</table>

Example: Three 1s Pattern Detector

• Problem: Detect three consecutive 1s in 8-bit input: abcdegh

  - Step 1: Capture the function
    - Truth table or equation?
      - Truth table too big: $2^8=256$ rows
      - Equation: create terms for each possible case of three consecutive 1s
      - $y = abc + bcd + cde + def + efg + fgh$
    - Step 2a: Create equation
      - already done
    - Step 2b: Implement as a gate-based circuit

Example: Number of 1s Counter

• Problem: Output in binary on two outputs yz the # of 1s on three inputs
  - 010 $\rightarrow$ 01
  - 101 $\rightarrow$ 10
  - 000 $\rightarrow$ 00

  - Step 1: Capture the function
    - Truth table or equation?
      - Truth table is straightforward
    - Step 2a: Create equations
      - $y = abc + ab\bar{c} + ab\bar{c} + ab\bar{c}$
      - $z = \bar{a}bc + \bar{a}bc + \bar{a}bc + \bar{a}bc$
    - Optional: Let's simplify $y$:
      - $y = abc + ab\bar{c} + ab\bar{c} + ab\bar{c}$
      - $z = \bar{a}bc + \bar{a}bc + \bar{a}bc + \bar{a}bc$
  - Step 2b: Implement as a gate-based circuit
Simplifying Notations

- Used in previous circuit

List inputs multiple times

Less wiring in drawing

Draw inversion bubble rather than inverter. Or list input as complemented.

Example: Keypad Converter

- Keypad has 7 outputs
  - One per row
  - One per column
- Key press sets one row and one column output to 1
  - Press "5" \( r_2 = 1, c_2 = 1 \)
- Goal: Convert keypad outputs into 4-bit binary number
  - 0-9 \( \rightarrow \) 0000 to 1001
  - * \( \rightarrow \) 1010, # \( \rightarrow \) 1011
  - nothing pressed: 1111

Example: Keypad Converter

- Step 1: Capture behavior
  - Truth table too big (2^7 rows); equations not clear either
  - Informal table can help
  - NAND same as AND with power & ground switched
  - 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")

Example: Sprinkler Controller

- Microprocessor outputs which zone to water (e.g., \( cba=110 \) means zone 6) and enables watering (e=1)
- Decoder should set appropriate valve to 1

Example: Sprinkler Controller

- Step 2b: Implement
  - d0 = a'b'c'e
  - d1 = a'b'ce
  - d2 = a'bc'e
  - d3 = a'bce
  - d4 = ab'c'e
  - d5 = ab'ce
  - d6 = abc'e
  - d7 = abce
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

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
  - Thus, NAND is a universal gate
  - Can implement any circuit using just NAND gates
  - Likewise for NOR

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 binary numbers for each binary number
  - Use 6x64 decoder
  - 4 outputs unused

Number of Possible Boolean Functions

- How many possible functions of 2 variables?
  - 2^2 rows in truth table, 2 choices for each row
  - 2^2 = 2^2 = 16 possible functions
- N variables
  - 2^N rows
  - 2^2^N possible functions

Decoder Simulation

- Link

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

Decoder with enable e

- Inputs decoder: 2^N inputs
  - Enables decoder for that input
  - Outputs all 0 if e=0
  - Regular behavior if e=1
  - n-input decoder: 2^n outputs
### Video

### Decoder Questions

### 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 \( \rightarrow \) needs 2 select inputs to indicate which input to route through
  - 8 input mux \( \rightarrow \) 3 select inputs
  - N inputs \( \rightarrow \) \( \log_2(N) \) selects
  - Like a rail yard 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

### 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 on D using two inputs x and y
  - Pushing button sequences to the next item
- Use 8-bit 4x1 mux
Mux Questions

Video

Additional Considerations

Non-Ideal Gate Behavior -- Delay

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

Circuit Delay and Critical Path

- Wires also have delay
- Assume gates and wires have delays as shown
- Path delay – time for input to affect output
- Critical path – path with longest path delay
- Circuit delay – delay of critical path

Active Low Inputs

- Data inputs: flow through component (e.g., mux data input)
- Control input: influence component behavior
  - Normally active high – 1 causes input to carry out its purpose
  - Active low – Instead, 0 causes input to carry out its purpose
  - Example: 2x4 decoder with active low enable
    - 1 disables decoder, 0 enables
    - Drawn using inversion bubble

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

Summary

- Combinational design process
  - Translate from equation (or table) to circuit.
- More gates
  - NAND, NOR, XOR, XNOR.
- Muxes and decoders
  - Decoder: converts input binary number to one high output.
  - Multiplexer: routes one of its N data inputs to its one output, based on binary value of select inputs.
- Additional considerations
  - Circuit delay and critical path.
  - Active low Inputs.
  - Schematic capture and simulation.