Previous Lecture: Combinational Circuits

Review: Combinational circuit

- Boolean logic: Operates on True (1) and False (0)
  - Operators: AND, OR, NOT
- Truth table: Enumerate output for all input combinations
  - K inputs \( \rightarrow \) \( 2^k \) rows needed for all input combinations?
- Can create a combinational circuit for any truth table
  - No feedback or cycles in circuit

Steps

- Combine transistors to implement logic gates
  - AND, OR, NOT
- Combine logic gates to build higher-level structures
  - Decoder, adder, register, memory ... (2 lectures)
- Combine structures with clock to build processor
Shorthand

Can draw AND and OR gates with more than two inputs

AND gate: Output is 1 if and only if all inputs are 1
OR gate: Output is 1 if one or more inputs are 1

Review: Sum of Products

Can implement any truth table with AND, OR, NOT

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</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>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</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>0</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>

1. AND combinations that yield a "1" in the truth table.
2. OR the results of the AND gates.
Mystery Circuit

What does this combinational circuit do?

Decoder Circuit

$n$ inputs, $2^n$ outputs
- exactly one output is 1 for each input pattern

Use large decoders when building memory circuits later...
Adder: Combinational circuit for binary addition?

\[
\begin{array}{c|c|c}
25 & 11001 & 16 + 8 + 1 = 25 \\
+29 & 11101 & 16 + 8 + 4 + 1 = 29 \\
\hline
54 & 110110 & 32 + 16 + 4 + 2 = 54 \\
\end{array}
\]

Goal: Design circuit to add any two \( N \)-bit integers

How can you use a truth table for \( N=64 \)?

Truth Table for \( N \)-bit addition?

Number of inputs (bits)?
- \( 2N \)

How many rows?
- \( 2^{2N} \)
  - Example: Common \( N = 32 \) bits \( \rightarrow \) Rows = \( 2^{64} \)

Implication:
- Need a fundamentally different approach

Modular Design
- Library of small number of basic components
- Combine together to achieve desired functionality
- Basic principle of modern industrial design
Modular Design for Addition

What algorithm do you follow for addition???

\[
\begin{align*}
&\text{Carry bits} \\
c_{N-1} & c_{N-2} \ldots c_1 c_0 \\
a_{N-1} & a_{N-2} \ldots a_1 a_0 \\
+ & b_{N-1} b_{N-2} \ldots b_1 b_0 \\
& s_N s_{N-1} s_{N-2} \ldots s_1 s_0
\end{align*}
\]

Use \( N \) 1-bit Full Adders

Full adder: Takes a carry bit (vs. Half adder)

Module: 1-bit Full Adder

Add two bits (A, B) and carry-in (\( C_{\text{in}} \)),
for one-bit sum (S) and carry-out (\( C_{\text{out}} \))

Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C_{\text{in}}</th>
<th>S</th>
<th>C_{\text{out}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<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>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Combinational Circuit
1-bit Full Adder

Represent this circuit:

With this diagram:

A_k  B_k

1-bit Full Adder

Carry bit for next adder

C_k

C_k+1

S_k

Four-bit Adder

Do you find anything strange or disturbing about this adder?
To compute sum of higher bits need results from lower bits
"Ripple-carry" adder
Timing Diagram

NOT gate

5V
X
0V

delay

output

5V
0V

delay

Time

Four-bit Adder

How many gate delays until output settles?
Each 1-bit adder requires 2 gate delays (AND + OR gates)
4 adders * 2 gate delays/adder = 8 gate delays
Adder: Example

\[
\begin{array}{c|c|c|c|c|c|c|c}
A & B & C_{in} & S & C_{out} \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
25 & 11001 \\
+29 & 11101 \\
54 & 110110 \\
\end{array}
\]

Combinational Circuits

Combinational Circuit

- Always gives same output for given set of inputs
  - ex: adder always generates sum and carry, regardless of previous inputs
- Cycles are not allowed
  - Cannot have feedback from output back to input

- Useful for many, but not all, aspects of computation
  - Arithmetic Logic Unit (ALU)
Matt likes Sue but he doesn’t like changing his mind

Represent with a circuit:
Matt will go to the party if Sue goes or if he already wanted to go

Sequential Circuits

Sequential Circuit (vs. Combinational)

• Stores information: state
• Output depends on state + input
  -- Given same input might produce different output, depending on stored information
• Example: ticket counter
  -- Advances when push button, output depends on previous state
• Cycles are allowed
  -- Can have feedback from output to input
• Useful for building memory and state machines
How can Matt change his Mind? Enter Rita

Matt will go to the party if Sue goes OR if the following holds:
(he already wanted to go AND Rita does not go)

How would you express?

\[ M' = S \text{ OR } (M \text{ AND NOT } R) \]

R, S: “control” inputs
What is S doing?
Setting state
What is R doing?
Resetting state (to 0)

R-S Flip-Flop
(Caution: Simplified !!)

If Set = 1 (and Reset = 0), M = 1
If Reset = 1 (and Set = 0), M = 0
If Set = 0, Reset = 0, M keeps old value!

Basic Building Block for remembering values or state
Build More Convenient 1-Bit Memory

“Data Flip-Flop” or “D flip flop” or “D latch”

Can be implemented using R-S flip flop

- If Write = 0, M just keeps its value. (It ignores D.)
- If Write = 1, then M becomes set to D

Register

Register stores N-bit value

- Collection of D-flip flops, all controlled by common Write Signal (WE)
- When WE=1, N-bit value D is written to register
What controls the “Write” signal?

- Often, the system clock!
- “clock” = device that sends out a fluctuating voltage signal that looks like this

\[ \text{Write} = 1 \]
\[ \text{Write} = 0 \]

“Computer speed” often refers to the clock frequency (e.g. 2.4GHz)

What limits the clock speed?

Synchronous Sequential Circuit

(a.k.a. Clocked Sequential Circuit)
Synchronous Sequential Circuit

Combination Circuit

CLK

Many N-bit Registers

This stands for "lots of wires"

Summary

Today’s Topics

- Combinational circuits: Output computed from inputs
  - Example: Decoder – One of \(2^N\) outputs high for \(N\) inputs
  - Example: Ripple-carry adder composed of \(N\) 1-bit full adders
- Sequential circuits:
  - Cycles; Remembers; Output computed from inputs+state

Reading: pp 152-183

Announcements

- Homework 6 Due Friday: Binary + Boolean Logic
- Project 2 Due Friday before Spring Break
  - Points-based game: Educational or Action
- Exam 2 after Spring Break