Announcements
Homework 10 (LAST!!) due Wednesday by 5 pm
  • Watch TED talk; write essay answering questions
Final Project: Card Game
  • Due December 12 – In-class Demos
Intermediate Deadlines
  • Wed (11/30): Find project partner
    – Google Doc to find others (email to cs202-tas@cs.wisc.edu)
  • Fri (12/2): Project proposal
    – 1 sentence email to cs202-tas@cs.wisc.edu (cc partner)
  • Wed (12/7): Project draft to Learn@UW dropbox
    – Whatever you have completed
Instructor Office Hours: None today
  Tuesday and Thursday -- 1:30 – 4:30

Previous Lecture:
Boolean Logic
Boolean logic: Operates on True (1) and False (0)
  • Operators: AND, OR, NOT
Boolean expression, truth table, combinational circuit
  • All three are equivalent
Truth table: Give output for all input combinations
  • 1 input variable → how many rows?
  • 2 input variables?
  • 3 input variables?
  • K inputs variables?

How does a computer... do arithmetic?

Shorthand in Logic Gates
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.

**Approach #1: Sum of Products**

Inputs: Two binary numbers A and B
View each bit of number as an input; 2 bits each
- A = aₐa₀
- B = bₐb₀

What range of numbers can be added together?
- A can be 00, 01, 10, 11 (0, 1, 2, 3)
- B can be 00, 01, 10, 11 (0, 1, 2, 3)

Output is a three-bit binary number
- Sum = s₂s₁s₀

Are 3 bits enough to represent Sum?
- Largest Sum = 11 + 11 = 3 + 3 = 6 = 100

**Today’s Challenge**

Can we perform **addition** using only AND, OR, and NOT gates?

Can compute logic circuit for any function
"Sum = A plus B" is a function of only inputs
Therefore, can create circuit for addition!

**Approach #1: Sum of Products**

View each bit of number as an input; 2 bits each
- A = aₐa₀; B = bₐb₀

Output is a three-bit binary number
- Sum = s₂s₁s₀

Construct truth table of all input combinations
- How many rows?
- 4 bits of input
- 2⁴ = 16 rows of table

Use sum-of-products algorithm for three outputs
- s₂, s₁, and s₀
### How to construct addition circuit?

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</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>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>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. Set up Truth Table; Enumerate all 16 input combinations for \( a_1, a_0, b_1, b_0 \)

2. Determine decimal number corresponding to each input

3. Calculate decimal sum as output

4. Translate decimal number to binary

5. Create 3 circuits (Sum-of-products) for \( s_2, s_1, s_0 \)

---

### Approach #1: Sum of Products

What if 32-bit integers instead of 2-bits??

Number of inputs to circuit?
- \( 2 \times 32 = 64 \)

How many rows in truth table?
- \( 2^{64} \)

Implication:
- Need a fundamentally different approach for any real architecture!

---

### Approach #2: Modular Design

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

Requires some **insight** to design component

### What algorithm do you use for decimal addition?

\[
\begin{align*}
6925 & + 8729 \\
& \quad \downarrow \\
& 15654
\end{align*}
\]

**Informal:**
- Memorize facts for adding numbers 0.9 to 0.9 + 1 (carry)
- Apply facts to ones position; record units; carry tens
- Repeat for each position (tens, hundred, thousands) w/ carry
- Record final carry out
Algorithm for binary addition?

19 10011
+27 11011
46 101110

16 + 2 + 1 = 19
16 + 8 + 2 + 1 = 27
32 + 8 + 4 + 2 = 46

We know these facts:
0 + 0 + 0 = 00
1 + 0 + 0 = 01
1 + 1 + 0 = 10 (two)
1 + 1 + 1 = 11 (three)

Modular Design for Addition

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

Carry bits

Repeatedly (N times) do 1-bit full add:
Take cin, a, b as input
Compute cout, s as output

Module: 1-bit Full Adder

Implement using just AND, OR, NOT
• Add two bits (A, B) and carry-in (Cin), for one-bit sum (S) and carry-out (Cout)

Abstraction: 1-bit Full Adder

Represent this circuit:
With this diagram:
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

Adder: Example

25  11001
+29  11101

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C_in</th>
<th>F_out</th>
<th>S</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>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>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</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>

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
Influences clock speed of processor
Create Addition Circuit in Scratch!

Add two 4-bit numbers to produce 1 5-bit number
  • Only use AND, OR, and NOT!
  • 4 nearly identical Sprites

Scripts for 1 Full-bit Adder

Scripts identical across 4 Adder Sprites
Each has private variable for a, b, cin, cout, s

To connect adders, simply set cin to cout of “previous” Sprite

Summary

Today’s Topics
  • We can do addition with just AND, OR, and NOT!
Homework 10 (LAST!!) due Wednesday by 5 pm
  • Watch TED talk; write essay answering questions
Final Project : Card Game
  • Due December 12 – In-class Demos
Intermediate Deadlines
  • Wed (11/30): Find project partner
    – Google Doc to find others (email to cs202-tas@cs.wisc.edu)
  • Fri (12/2): Project proposal
    – 1 sentence email to cs202-tas@cs.wisc.edu (cc partner)
  • Wed (12/7): Project draft to Learn@UW dropbox
    – Whatever you have completed
Instructor Office Hours: None today
      Tuesday and Thursday -- 1:30 – 4:30