Topic 9: Building Blocks for Computers

Readings for this topic:
P&H, Appendix B.3 thru B.6

Goal
• Summary of combinatorial and sequential components that are useful for computers.
• Techniques for combining them

Recall: Full Adder

Boo:lean Algebra

\[ \text{Sum} = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC \]
\[ \text{Cout} = AB + BC + AC \]

Bit Slice Adders

Problem: How to add 3 bit numbers?
• \(C2C1C0 = A2A1A0 + B2B1B0\)
• Redesigning circuit for 6 inputs would be messy and wouldn’t scale well.

Solution: Cascade 1-bit adders

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>Sum</th>
<th>Cout</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>1</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>1</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>

Adder bit slice

This is called a "full adder".
• A "half adder" adds two bits and produces sum and carry out
Bit-slice Adder

Problem: Time to compute carry grows with number of inputs
Solution: Carry look ahead adders.

- shortcut the carry from previous group of bits to following group of bits
- How: using local group of bits, determine:
  - generate: group will always generate carry into next group
  - kill: group will never generate carry into next group
  - propagate: group will propagate carry from previous group into next

Multiplexer

Given an n-bit number as input, select one of 2n inputs.

AND gate passes signal through if the control is 1

Shifters

Shifts 1 bit left/right, based on input: 1 => shift left, 0 => shift right

Full Shifter

Shift N-bit number N positions in one direction
Can build a shifter with multiplexors

Example: 4-bit right-shifter

Example: to make full 32-bit shifter, use 3 stages:
  - stage1: shift by 0, 8, 16, or 24
  - stage2: shift by 0, 2, 4, or 6
  - stage3: shift by 0, or 1
ALU

Summary: we can do shifters, adders, AND, OR, NOT, XOR, ...
  • Arithmetic operations generate a carry
  • Logic operations have no carry

An ALU computes a function of 2 inputs \( O = F(A, B) \),
where the function \( F \) is selected by other inputs \( (F_0, F_1) \).
  • Bit Slicing: Compute function for 1 bit using carry in and carry out.
    This is just a generalization of cascaded adders.

Decoders

How do we select an operation?
Decoder: given an n-bit number as input, enables one of 2n outputs

<table>
<thead>
<tr>
<th>F</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>A AND B</td>
</tr>
<tr>
<td>01</td>
<td>A OR B</td>
</tr>
<tr>
<td>10</td>
<td>NOT B</td>
</tr>
<tr>
<td>11</td>
<td>A + B</td>
</tr>
</tbody>
</table>

Circuit Symbol

ALU Bit Slice Schematic

An example ALU

Decoder: given an n-bit number as input, enables one of 2n outputs
ALU Schematic

A schematic diagram of an ALU (Arithmetic Logic Unit) showing the flow of data through a series of 1-bit ALUs, with inputs and outputs labeled.