Homework

• Reading
  – Tokheim, Chapter 3, 4, and 6.1 - 6.3
  – Logisim Website
• Machine Projects
  – Continue on mp3
• Labs
  – Continue in labs with your assigned section
Combining Basic Logic Gates

- Decoders
- Encoders
- Selectors - Multiplexers
- ALUs
- Control Units
- Buses
- Simple computers
Binary Decoder

- Logic with \( n \) input lines and \( 2^n \) output lines
- Only one output is a 1 for any given input

![Diagram of a binary decoder with \( n \) inputs and \( 2^n \) outputs.](image)
Building a Binary Decoder

• Start with a 2-bit decoder:
Then Add Two to Make Three...

Diagram:
- 2-bit Decoder
- NOT
- Inputs: $X_0$, $X_1$, $X_2$
- Outputs: $Y_0$, $Y_1$, $Y_2$, $Y_3$, $Y_4$, $Y_5$, $Y_6$, $Y_7$
Developing an Encoder

- If we can decode, then we need to encode
- Encode from 1 out of n into a binary weighted form
- A keyboard encoder does this
Next Comes a Selector

- Like a switch; also called a multiplexer or MUX

- Again, build it up from simple basic logic gates
A 1-bit Selector

Decoder

AND

S1

S2

a

b

AND

AND

AND

OR

y

c

d
A 4-bit Selector
The ALU Is Next

- Logical and arithmetic operations
- Variations in
  - Base
    - Binary
    - Decimal
    - BCD
  - Implementation
    - Serial
    - Parallel
    - Pipelined
Simple Example - Binary Adder

- Develop a half-adder (HA)
- Use two HA’s to build a full-adder (FA)
The Half-Adder

\[
\text{Sum} = (\overline{a} \cdot b) + (a \cdot \overline{b}) = (\overline{a} + \overline{b}) \cdot (a + b)
\]

\[
\text{Carry} = a \cdot b
\]
From HA to FA

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c_{in}</th>
<th>Sum</th>
<th>C_{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>

**Half Adder**

**Full Adder**
Using Full Adders for Addition

Note: The carry flag value after the addition represents the N+1 bit value in the result.
## Using Full Adders for Subtraction

Difference: \( a - b \)
- \( = a + (-b) \)
- \( = a + (~b + 1) \)
- \( = a + ~b + 1 \)

### Table

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b#</th>
<th>C&lt;sub&gt;0&lt;/sub&gt;</th>
<th>Sum</th>
<th>C&lt;sub&gt;1&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSB</td>
<td>0 0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>0 1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
<td>0 1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1 1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b#</th>
<th>C&lt;sub&gt;n-1&lt;/sub&gt;</th>
<th>C&lt;sub&gt;n&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSBs</td>
<td>0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: The carry flag value after the ~ and addition is the opposite of the subtract borrow condition.
Configurable Add/Subtract ALU

Subtract/Add# From Instruction Decoding Logic

\( a_0 \) \( b_0 \) \( \ldots \) \( a_{n-1} \) \( b_{n-1} \)

\( \text{XOR} \) \( \text{XOR} \)

\( \text{Full Adder} \) \( \text{Full Adder} \)

\( \text{Sum}_0 \) \( \text{Sum}_{n-1} \)

\( \text{C}_0 \) \( \text{C}_n \)

\( \text{NOR} \)

\( \text{EFlags Register} \)

Zero Flag
Sign Flag
Overflow Flag = \( \text{C}_{n-2} \times \text{C}_{n-1} \)

Carry Flag