Checkers / Decoders

• Recall
  – AND gates output '1' for only a single combination
  – OR gates output '0' for only a single combination
  – Inputs (inverted or non-inverted) determine which combination is checked for
  – We say that gate is "checking for" or "decoding" a specific combination

\[
\begin{array}{ccc|c}
X & Y & Z & F \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

AND gate decoding (checking for) combination 010

\[
\begin{array}{ccc|c}
X & Y & Z & F \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

OR gate decoding (checking for) combination 110

Motivation

• Just like there are patterns and structures that occur commonly in nature, there are several common logic structures that occur over and over again in digital circuits
  – Decoders, Muxes, Adders, Registers
• In addition, we design hardware using a hierarchical approach
  – We design a small component using basic logic gates (e.g. a 1-bit mux)
  – We build a large component by interconnecting many copies of the small component + a few extra gates (e.g. a 32-bit mux)
  – We build chips by interconnecting many large components (e.g. a router)
  – Each components is truly made out of many gates but we the design process is faster and easier by using hierarchy
• Let's look at a few common components
  – We'll start by describing the behavior of the component and then determine what gates are inside

DECODERS
Decoders

A decoder is a building block that:
  - Takes an n-bit binary number as input
  - Decodes that binary number and activates the corresponding output
  - Individual outputs for each input combination

There are gates inside to implement each output

3-to-8 Decoder

3-bit binary number

1 output for each combination of the input number

Decoder Sizes

A decoder w/ an n-bit input has $2^n$ outputs
  - 1 output for every combination of the n-bit input

Only that numbered output is activated

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z (LSB)</th>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
<th>D6</th>
<th>D7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<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>0</td>
<td>0</td>
<td>0</td>
<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>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Only that numbered output is activated

2-to-4 Decoder

3-to-8 Decoder
Exercise

- Complete the design of a 2-to-4 decoder

<table>
<thead>
<tr>
<th></th>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Building Decoders

Vending Machine Example

If the keypad produces a 4-bit numeric output, add logic to produce the release signals for each of the 16 vending items.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

Enables

- In a normal decoder exactly one output is active at all times
- It may be undesirable to always have an active output
- We can add an extra input (called an enable) that can independently force all the outputs to their inactive values

- Consider any problems with this design.
### Enables

When $E=0$, inputs is ignored

\[
\begin{array}{c|c|c|c|c}
E & D_0 & D_1 & D_2 & D_3 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Since $E=0$, all outputs = 0

When $E=1$, inputs will cause the appropriate output to go active

\[
\begin{array}{c|c|c|c|c}
E & D_0 & D_1 & D_2 & D_3 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Since $E=1$, outputs will function normally

### Implementing Enables

- Original 2-to-4 decoder

\[
\begin{align*}
A' & \quad A \\
B' & \quad B \\
D_0 & \quad \quad \quad D_0 \\
D_1 & \quad \quad \quad D_1 \\
D_2 & \quad \quad \quad D_2 \\
D_3 & \quad \quad \quad D_3 \\
\end{align*}
\]

When $E=0$, force all outputs = 0

When $E=1$, outputs operate as they did originally

### Multiplexers

- Multiplexers are one of the most common digital circuits
- Anatomy: $n$ data inputs, $\log_2 n$ select bits, 1 output
- A multiplexer ("mux" for short) selects one data input and passes it to the output

\[
\begin{array}{c|c|c|c|c}
S_1 & S_0 & Y \\
0 & 0 & i0 \\
0 & 1 & i1 \\
1 & 0 & i2 \\
1 & 1 & i3 \\
\end{array}
\]

### 4-to-1 Mux

- Thus, input $2 = C$ is selected and passed to the output

\[
\begin{array}{c|c|c|c|c|c|c|c|c}
S_1 & S_0 & Y & i0 & i1 & i2 & i3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

As long as the select bits are $10_2 = 2$, whatever bit value appears on input 2 is copied to the output, same as if we had just wired input 2 directly to the output.
Multiplexers

4-to-1 Mux

Thus, input \( 0 = A \) is selected and passed to the output.

Select bits = 00, \( 0_2 = 0_{10} \)

<table>
<thead>
<tr>
<th>( S_1 )</th>
<th>( S_0 )</th>
<th>( Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>i0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>i1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>i2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>i3</td>
</tr>
</tbody>
</table>

Exercise: Build a 4-to-1 mux

- Complete the 4-to-1 mux to the right by drawing wires between the 2-to-4 decode and the AND gates.

Building a Mux

- To build a mux
  - Decode the select bits and include the corresponding data input.
  - Finally OR all the first level outputs together.

Building a Mux

- To build a mux
  - Decode the select bits and include the corresponding data input.
  - Finally OR all the first level outputs together.
Recall Using $T1/T2$

- 1st Level of AND gates act as barriers only passing 1 channel
- OR gates combine 3 streams of 0's with the 1 channel that got passed (i.e. ICH1)
- 2nd Level of AND gates passes the channel to only the selected output

---

2-to-1 Multiplexers

2-to-1 Mux

Thus, input 1 = B is selected and passed to the output

1. Select bits = $1_2 = 10_2$.

---

Building a 2-to-1 Mux

- To build a mux
  - Decode the select bits and include the corresponding data input.
  - Finally OR all the first level outputs together.

---

Building Large Muxes

- Similar to a tournament of sports teams
  - Many teams enter and are narrowed down to 1 winner
  - In each round winners play ________

---

Railroad Switch Station
Design an 8-to-1 mux with 2-to-1 Muxes

Cascading Muxes
- Use several small muxes to build large ones
- Rules
  1. Arrange the muxes in stages (based on necessary number of inputs in 1st stage)
  2. Outputs of one stage feed to inputs of the next until only 1 final output
  3. All muxes in a stage connect to the same group of select bits
     - Usually, LSB connects to first stage
     - MSB connect to last stage

Building a 4-to-1 Mux
- 4-to-1 mux built w/ 2-to-1 muxes
- Rule 1: Outputs from stage 1 connect to inputs of stage 2
- Rule 2: LSB $S_0$ connect to all muxes in first stage. MSB $S_1$ connects to all muxes in second stage

Walk through an example:
$S_1S_0 = 01$
Building a 4-to-1 Mux

Stage 1 Stage 2

S_1 S_0 Y
0 0 D_0
0 1 D_1
1 0 D_2
1 1 D_3

S_0 = 1 narrows our choices down to D_1 and D_3

Walk through an example:
S_1S_0 = 01

S_1 = 1 selects our final choice, D_1

Device vs. System Labels

- When using hierarchy (i.e., building blocks) to design a circuit be sure to show both device and system labels
  - Device Labels: Signal names used inside the block
    - Name the designer of the block uses to indicate which input/output is which to the outside user (Names may change; read the manual)
  - System labels: Signal names used outside the block
    - Actual signals from the circuit being built
    - Can have the same name as the device label if such a signal name exists at the outside level

Exercise

- Sketch how you could build a 16-to-1 mux with 4-to-1 muxes? 8-to-1 and 2-to1 muxes?
Exercise

- Create a 3-to-1 mux using 2-to-1 muxes
  - Inputs: I0, I1, I2 and select bits S1, S0
  - Output: Y

<table>
<thead>
<tr>
<th>S1</th>
<th>S0</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>D0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>D1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>D2</td>
</tr>
</tbody>
</table>

Select-bit Ordering

- If we connect the select bits as shown to build an 8-to-1 mux, show how to label the inputs (i0-i7) so that the correct input is passed based on the binary value of S2:S0

<table>
<thead>
<tr>
<th>Select</th>
<th>S2</th>
<th>S1</th>
<th>S0</th>
<th>OUT</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<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>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</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>
<td>1</td>
</tr>
</tbody>
</table>

Typical Logic Gate

- Gates can output two values: 0 & 1
  - Logic '1' (Vdd = 3V or 5V), or Logic '0' (GND)
  - But they are ALWAYS outputting something!!!
- Analogy: a sink faucet
  - 2 possibilities: Hot (‘1’) or Cold (‘0’)
- In a real circuit, inputs cause EITHER a pathway from output to VDD OR VSS
Output Connections

- Can we connect the output of two logic gates together?
- ____! Possible ______________ (static, low-resistance pathway from Vdd to GND)
- We call this situation “__________”

Tri-State Buffers

- Normal digital gates can output two values: 0 & 1
  1. Logic 0 = 0 volts
  2. Logic 1 = 5 volts
- Tristate buffers can output a third value:
  3. ____ = __________________ = “Floating” (no connection to any voltage source...infinite resistance)
- Analogy: a sink faucet
  - 3 possibilities:
    1.) Hot water,
    2.) Cold water,
    3.) ____ water

- We use tri-state buffers to __________ one output amongst several sources
- Rule: Only _________________________ at a time
Tri-State Buffers

- We use tri-state buffers to share one output amongst several sources.
- Rule: Only 1 buffer enabled at a time.
- When 1 buffer enabled, its output overpowers the Z's (no connection) from the other gates.

Enable Polarity

- Side note: Some tri-states are designed to pass the input (be enabled) when the enable is 0 (rather than 1).
  - A inversion bubble is shown at the enable input to indicate the "______" polarity needed to enable the tristate.

Communication Connections

- Multiple entities need to communicate.
- We could use:
  - Point-to-point connections
  - A ______________________________

Bidirectional Bus

- 1 transmitter (otherwise bus contention).
- N receivers.
- Each device can send (though 1 at a time) or receive.
**Tri-State Gates**

- Advantage: don’t have to know in advance how many devices will be connected together
  - Tri-State gates give us the option of connecting together the outputs of many devices without requiring a circuit to multiplex many signals into one
- Just have to make sure only one is enabled (output active) at any one time.