Part III - Standard Combinational Modules

Signal Transport
- Decoder: Decode address
- Encoder: Encode address
- Multiplexer (Mux): Select data by address
- Demultiplexer (DeMux): Direct data by address
- Shifter: Shift bit location

Data Operator
- Adder: Add two binary numbers
- Multiplier: Multiply two binary numbers

Part III - Standard Combinational Modules

• Introduction
• Decoder
  - Behavior, Logic, Usage
• Encoder
• Multiplexer (Mux)
  - Behavior, Logic, Usage
• Demultiplexer (DeMux)

Interconnect: Decoder, Encoder, Mux, DeMux
1. Decoder: Definition

A. A device that decodes
B. An electronic device that converts signals from one form to another
C. A machine that converts a coded text into ordinary language
D. A device or program that translates encoded data into its original format
E. All of the above

Decoder Definition: A digital module that converts a binary address to the assertion of the addressed device.

- N inputs, $2^N$ outputs
- One-hot outputs: only one output HIGH at most
1. Decoder: Definition

Example: A 3-input decoder has how many outputs?
A. 2 outputs
B. 4 outputs
C. 8 outputs
D. 10 outputs

Decoder Definition

Example: For a 3-input decoder, suppose $(E, I_2, I_1, I_0) = (1, 0, 0, 0)$, then $(y_7, y_6, ..., y_0)$ is equal to:
A. (00000000)
B. (00000001)
C. (00000010)
D. (01000000)
E. (10000000)

Decoder: Logic Diagram (Inside a decoder)

$y_0 = 1$ if $(A_1, A_0) = (0,0)$ & $E = 1$

$y_1 = m_1 E$

2:4 Decoder

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_0$</th>
<th>$Y_3$</th>
<th>$Y_2$</th>
<th>$Y_1$</th>
<th>$Y_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$y_3 = A_1 A_0 E$

1. Decoder: Definition

PI Q: What is the output $Y_{3:0}$ of the 2:4 decoder for $(A_1, A_0) = (1,0)$?
A. (1, 1, 0, 0) 
B. (1, 0, 1, 1) 
C. (0, 0, 1, 0) 
D. (0, 1, 0, 0)
Decoder Application: universal set \{Decoder, OR\}

Example:
Implement the following functions with a 3-input decoder and OR gates.
i) \( f_1(a,b,c) = \Sigma m(1,2,4) \)
ii) \( f_2(a,b,c) = \Sigma m(2,3) \)
iii) \( f_3(a,b,c) = \Sigma m(0,5,6) \)

Decoder produces minterms when \( E=1 \).
We can use an OR gate to collect the minterms to cover the On-set.
For the Don’t Care-Set, we can just ignore the terms.

Decoder Application: universal set \{Decoder, OR\}

Example: Implement functions
i) \( f_1(a,b,c) = \Sigma m(1,2,4) + \Sigma d(0,5) \),
ii) \( f_2(a,b,c) = \Sigma m(2,3) + \Sigma d(1,4) \),
iii) \( f_3(a,b,c) = \Sigma m(0,5,6) \)

with a 3-input decoder and OR gates.

Decoders

- OR minterms
Tree of Decoders: Scale up the size of the decoders using a tree structure.
Implement a 4-2^4 decoder with 3-2^3 decoders.

Tree of Decoders

Implement a 6-2^6 decoder with 3-2^3 decoders.

Construction: A four variable switching function \( f(a, b, c, d) \) can be implemented using which of the following?

A. 1:2 decoders and OR gates
B. 2:4 decoders and OR gates
C. 3:8 decoders and OR gates
D. All of the above
E. None of the above

2. Encoder

- Definition
- Logic Diagram
- Priority Encoder
Definition of Encoder

A. Any program, circuit or algorithm which encodes
B. In digital audio technology, an encoder is a program that converts an audio WAV file into an MP3 file
C. A device that convert a message from plain text into code
D. A circuit that is used to convert between digital video and analog video
E. All of the above

Encoder Definition: A digital module that converts the assertion of a device to the binary address of the device.

Encoder Description:
At most one $I_i = 1$.
$(y_{n-1}, \ldots, y_0) = i$ if $I_i = 1$ & $E = 1$
$(y_{n-1}, \ldots, y_0) = 0$ otherwise.
$A = 1$ if $E = 1$ and one $i$ s.t. $I_i = 1$
$A = 0$ otherwise.

Encoder: Logic Diagram

Priority Encoder:
Priority Encoder: Definition

Description: Input \((I_2^{n-1}, \ldots, I_0)\), Output \((y_{n-1}, \ldots, y_0)\)

\((y_{n-1}, \ldots, y_0) = i\) if \(I_i = 1\) and \(E = 1\) and \(I_k = 0\)

for all \(k > i\) (high bit priority) or

for all \(k < i\) (low bit priority).

\(E_o = 1\) if \(E = 1\) and \(I_i = 0\) for all \(i_\ast\)

\(G_s = 1\) if \(E = 1\) and \(\exists i\) s.t. \(I_i = 1\).

\((G_s\) is like \(A\), and \(E_o\) passes on enable).\n
\(E = 1\) but \(I_i = 0\) \(\forall i_\ast\)

Priority Encoder: Implement a 32-input priority encoder with 8 input priority encoders (high bit priority).

- \(I_{31:24} \to E_0 \to y_{32}, y_{31}, y_{30}\)
- \(I_{23:16} \to E_0 \to y_{22}, y_{21}, y_{20}\)
- \(I_{15:8} \to E_0 \to y_{12}, y_{11}, y_{10}\)
- \(I_{7:0} \to E_0 \to y_{02}, y_{01}, y_{00}\)
CSE 140 Lecture 12
Standard Combinational Modules

CK Cheng
CSE Dept.
UC San Diego

Part III. Standard Modules

Interconnect Modules:
1. Decoder, 2. Encoder
3. Multiplexer, 4. Demultiplexer

Multiplexer

- Definition
- Logic Diagram
- Application

Interconnect: Decoder, Encoder, Mux, DeMux

Decoder: Decode the address to assert the addressed device
Mux: Select the inputs according to the index addressed by the control signals
Multiplexer Definition
A. A device that interleaves two or more activities
B. A communications device that combines several signals for transmission over a single medium
C. A logic circuit that sends one of several inputs out over a single output channel.
D. The circuit that uses a common communications channel for sending two or more messages or signals.
E. All of the above

3. Mux (Multiplexer) Definition: A digital module that selects one of data inputs according to the binary address of the selector.

\[ y = D_i \text{ where } i = (s_{n-1}, \ldots, s_0) \]

Description
If \( E = 1 \) then \( y = D_i \)
Else \( y = 0 \)

(Selector or Address)

Multiplexer (Mux): Definition
- Selects between one of \( N \) inputs to connect to the output.
- \( \log_2 N \)-bit select input – control input

\[ \begin{array}{c|c|c}
   S_0 & S_1 & \text{Output} \\
   \hline
   0 & 0 & D_0 \\
   0 & 1 & D_1 \\
   1 & 0 & D_1 \\
   1 & 1 & D_0 \\
\end{array} \]

S: Selector or Address

Multiplexer (Mux): Definition
- Selects between one of \( N \) inputs to connect to the output.
- \( \log_2 N \)-bit select input – control input
- Example:

2:1 Mux

\[ \begin{array}{c|c|c|c|c|c|c}
   S & D_0 & D_1 & y & S & y \\
   \hline
   0 & 0 & 0 & 0 & 0 & D_0 \\
   0 & 0 & 1 & 1 & 1 & D_1 \\
   0 & 1 & 0 & 0 & 0 & D_0 \\
   0 & 1 & 1 & 0 & 1 & D_1 \\
   1 & 0 & 0 & 1 & 0 & D_0 \\
   1 & 0 & 1 & 1 & 0 & D_1 \\
   1 & 1 & 0 & 0 & 1 & D_0 \\
   1 & 1 & 1 & 1 & 1 & D_1 \\
\end{array} \]
What is the output of the following MUX?
A. 0
B. 1
C. Can't say

Multiplexer: Logic Diagram

- Logic gates
  - Sum-of-products

- Tristates
  - For an N-input mux, use N tristates
  - Turn on exactly one to select the appropriate input

Multiplexer Definition: 4-input mux

<table>
<thead>
<tr>
<th>S</th>
<th>D_0</th>
<th>D_1</th>
<th>D_2</th>
<th>D_3</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>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>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</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>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ Y = D_0 \bar{S} + D_1 S \]

Multiplexer Application: universal set \{Mux\}

We use selector to decompose the function into smaller functions (less number of variables), which follows Shannon's expansion.

We simplify the decomposed functions using K-map, which follows consensus theorem.
Multiplexer Application

- Mux for a Boolean function with truth table as input
- Building blocks of FPGA (Field Programmable Gate Array).

For the logic diagram on left, output Y is

A. AB
B. (AB)'
C. A+B
D. (A+B)'
E. None of the above

Example 1: Given \( f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2) \), implement with an 8-input Mux.

<table>
<thead>
<tr>
<th>id</th>
<th>abc</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>1</td>
</tr>
</tbody>
</table>

Example 2: Given \( f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2) \), implement with 4-input Muxes.

<table>
<thead>
<tr>
<th>ab</th>
<th>c=0</th>
<th>c=1</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td></td>
<td></td>
<td>( D_0 )</td>
</tr>
<tr>
<td>01</td>
<td></td>
<td></td>
<td>( D_1 )</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td>( D_2 )</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>( D_3 )</td>
</tr>
</tbody>
</table>

Example 3: Given \( f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2) \), implement with 2-input Muxes.

<table>
<thead>
<tr>
<th>a\bc</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
<th>D(b,c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>( D_0 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( D_1 )</td>
</tr>
</tbody>
</table>