CSE 140 Lecture 12
Combinational Standard 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
iClicker: 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.

Description:

If $E = 1$

$$y = D_i \text{ where } i = (S_{n-1}, \ldots, S_0)$$

Else

$$y = 0$$

(Data input) $D_{2^{n-1}}-D_0$

(Selector or Address) $S_{n-1,0}$
Multiplexer (Mux): Definition

- Selects between one of $N$ inputs to connect to the output.
- $\log_2 N$-bit select input – control input

![Diagram of a multiplexer]

- E: Enable
- y: Output
- D₀, D₁: Data inputs
- S: Selector or Address
PI Q: What is the output of the following MUX?
A. 0
B. 1
C. Can’t say
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

<table>
<thead>
<tr>
<th>( S )</th>
<th>( D_1 )</th>
<th>( D_0 )</th>
<th>( Y )</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>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>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>1</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>1</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c}
S \downarrow \\
\hline
0 & D_0 \\
0 & D_1 \\
1 & 0 \\
1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c}
S & Y \\
\hline
0 & D_0 \\
1 & D_1 \\
\end{array}
\]
Multiplexer Definition: Example

<table>
<thead>
<tr>
<th>D0</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>( S_1 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( S_1 )</th>
<th>( S_0 )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Multiplexer Definition: Example

E=1:
If $D_0 = 0$ and $S_1 S_0 = 00 \Rightarrow y = 0$
If $D_0 = 1$ and $S_1 S_0 = 00 \Rightarrow y = 1$
Multiplexer: Logic Diagram

• Logic gates
  – Sum-of-products form

<table>
<thead>
<tr>
<th></th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ Y = D_0S + D_1S \]

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

![Logic gates diagram]

![Tristates diagram]
Multiplexer Application

- Mux for a Boolean function with truth table as input

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$Y = AB$
Multiplexer: Application

\[ Y = AB \] 

\[
\begin{array}{c|c|c}
A & B & Y \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

\[ A \quad B \quad Y \]

\[
\begin{array}{c|c}
A & Y \\
0 & 0 \\
1 & 1 \\
\end{array}
\]

\[ A \quad B \quad Y \]

\[ 0 \quad 1 \]

\[ 0 \quad 1 \]

\[ 0 \quad 0 \]

\[ 0 \quad 0 \]

\[ 0 \quad 0 \]

\[ 1 \quad 1 \]

\[ 1 \quad 1 \]

\[ 1 \quad B \]

\[ A \quad Y \]

\[ 0 \quad B \]

\[ 1 \quad 1 \]
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: universal set \{Mux\}

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>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c = 0$</th>
<th>$c = 1$</th>
<th>$D(c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>$D_0(c) =$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td>$D_1(c) =$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>$D_2(c) =$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>$D_3(c) =$</td>
</tr>
</tbody>
</table>

- $D_0(c) =$
- $D_1(c) =$
- $D_2(c) =$
- $D_3(c) =$
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</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(b,c)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D_1(b,c)</td>
</tr>
</tbody>
</table>

![Mux Diagram]

- E
- 0
- 1
- a
- y
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</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(b,c)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>D_1(b,c)</td>
</tr>
</tbody>
</table>

- $D_0(b,c) = b'$
- $D_1(b,c) = bc$

![Diagram of the Mux implementation](image)
Example 3: Given $f(a,b,c) = \Sigma m(0,1,7) + \Sigma d(2)$, implement with 2-input Muxes.

<table>
<thead>
<tr>
<th>$b$</th>
<th>$c = 0$</th>
<th>$c = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

$l_1(0) = 0$

$l_1(c) = c$

![Diagram of 2-input Mux]
Example 3: Given $f(a, b, c) = \Sigma m (0, 1, 7) + \Sigma d(2)$, implement with 2-input Muxes.

<table>
<thead>
<tr>
<th>$D_1(b, c)$</th>
<th>$b$</th>
<th>$c = 0$</th>
<th>$c = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$l_1(0) = 0$</td>
</tr>
<tr>
<td>$1$</td>
<td>$0$</td>
<td>$1$</td>
<td>$l_1(c) = c$</td>
</tr>
</tbody>
</table>

![Diagram of 2-input Muxes implementation]
Example 4: Given \( f(a,b,c) = \Sigma m(0,2,4,7) + \Sigma d(3,5) \), implement with 2-input Muxes.

\[
\begin{array}{c|cccc}
  f & \mathbf{00} & \mathbf{01} & \mathbf{10} & \mathbf{11} \\
  \hline
  a \\
  0 & 1 & 0 & 1 & - & D_0(b,c) \\
  1 & 1 & - & 0 & 1 & D_1(b,c)
\end{array}
\]
4. Demultiplexers
4. Demultiplexers

\[ y_i = x \text{ if } i = (S_{n-1}, \ldots, S_0) \& E=1 \]
\[ y_i = 0 \text{ otherwise} \]
Shifters

• **Logical shifter**: shifts value to left or right and fills empty spaces with 0’s
  – Ex: \(11001 >> 2 = 00110\)
  – Ex: \(11001 << 2 = 00100\)

• **Arithmetic shifter**: same as logical shifter, but on right shift, fills empty spaces with the old most significant bit (msb).
  – Ex: \(11001 >>> 2 = 111\)
  – Ex: \(11001 <<< 2 = 00100\)

• **Rotator**: rotates bits in a circle, such that bits shifted off one end are shifted into the other end
  – Ex: \(11001 \text{ ROR } 2 = 01110\)
  – Ex: \(11001 \text{ ROL } 2 = 00111\)
Shifter

Can be implemented with a mux

\[ y_i = x_{i-1} \text{ if } E = 1, s = 1, \text{ and } d = L \]
\[ = x_{i+1} \text{ if } E = 1, s = 1, \text{ and } d = R \]
\[ = x_i \text{ if } E = 1, s = 0 \]
\[ = 0 \text{ if } E = 0 \]
Shifter Design

A_{3:0} \quad >> \quad Y_{3:0}

\text{shamt}_{1:0}

\begin{array}{cccc}
A_3 & A_2 & A_1 & A_0 \\
\hline
Y_3 & Y_2 & Y_1 & Y_0 \\
\hline
00 & 01 & 10 & 11 \\
00 & 01 & 10 & 11 \\
00 & 01 & 10 & 11 \\
00 & 01 & 10 & 11 \\
\end{array}
Barrel Shifter

- **O or 1 shift**
- **O or 2 shift**
- **O or 4 shift**

Inputs: $s_0$, $s_1$, $s_2$

Outputs: $x$, $y$

Example input: 01010101

Shift operations:
- 01010101
- 01010101
- 01010101

Shifted output: 01010101010101
Shifters as Multipliers and Dividers

• A left shift by $N$ bits multiplies a number by $2^N$
  – Ex: $00001 \ll 2 = 00100$ ($1 \times 2^2 = 4$)
  – Ex: $11101 \ll 2 = 10100$ ($-3 \times 2^2 = -12$)

• The arithmetic right shift by $N$ divides a number by $2^N$
  – Ex: $01000 \ggg 2 = 00010$ ($8 \div 2^2 = 2$)
  – Ex: $10000 \ggg 2 = 11100$ ($-16 \div 2^2 = -4$)