ECE 301 – Digital Electronics

Tristate Buffers,
Read-Only Memories
and
Programmable Logic Devices

(Lecture #17)

The slides included herein were taken from the materials accompanying Fundamentals of Logic Design, 6th Edition, by Roth and Kinney, and were used with permission from Cengage Learning.
Tristate Buffers
Tristate Buffer

- A tristate buffer can output 3 different values:
  - Logic 1 (high)
  - Logic 0 (low)
  - High-Impedance
Tristate Buffers

Enable

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Z</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>

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Z</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Z</td>
</tr>
</tbody>
</table>
Building a Mux with Tristate Buffers

Outputs can be “shorted” together

Only one buffer is enabled at a time
IC Bi-directional I/O Pin

- Integrated Logic Circuit
- Output
- Input
- EN
- Bi-Directional Input-Output Pin
Read-Only Memories
A read-only memory (ROM) consists of an array of semiconductor devices that are interconnected to store a set of binary data.

Once binary data is stored in the ROM, it can be read out whenever desired, but the data that is stored cannot be changed under normal operating conditions.

- Data is written to the ROM once, and read from the ROM many times.
ROM

address

A  B  C

ROM
8 Words
x 4 Bits

F₀  F₁  F₂  F₃

<table>
<thead>
<tr>
<th>A B C</th>
<th>F₀</th>
<th>F₁</th>
<th>F₂</th>
<th>F₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>1 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>1 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0 0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Typical Data
Stored in
ROM
(2³ words of
4 bits each)

(b) Truth table for ROM
ROM – Basic Structure

n Input Lines

Decoder

ROM

Memory Array

2^n Words x m Bits

m Output Lines

address

data
Building Logic Functions using ROM

\[ F_0 = \Sigma m(0, 1, 4, 6) \]
\[ F_1 = \Sigma m(2, 3, 4, 6, 7) \]

What functions are realized by the ROM for \( F_2 \) and \( F_3 \)?
Building Logic Functions using ROM

<table>
<thead>
<tr>
<th>Input W X Y Z</th>
<th>Hex Digit</th>
<th>ASCII Code for Hex Digit A6 A5 A4 A3 A2 A1 A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0</td>
<td>0 1 1 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>1</td>
<td>0 1 1 0 0 0 1</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>2</td>
<td>0 1 1 0 0 1 0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>3</td>
<td>0 1 1 0 0 1 1</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>4</td>
<td>0 1 1 0 1 0 0</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>5</td>
<td>0 1 1 0 1 0 1</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>6</td>
<td>0 1 1 0 1 1 0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>7</td>
<td>0 1 1 0 1 1 1</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>8</td>
<td>0 1 1 0 0 0 0</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>9</td>
<td>0 1 1 0 0 0 1</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>A</td>
<td>1 0 0 0 0 0 1</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>B</td>
<td>1 0 0 0 0 0 1</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>C</td>
<td>1 0 0 0 0 0 1</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>D</td>
<td>1 0 0 0 0 1 1</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>E</td>
<td>1 0 0 0 1 0 1</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>F</td>
<td>1 0 0 0 1 1 0</td>
</tr>
</tbody>
</table>
Building Logic Functions using ROM

\[ A_5 = A_4 \]
\[ A_6 = A_5' \]
Programmable Logic Devices
Programmable Logic Device

A programmable logic device (or PLD) is a general name for a digital integrated circuit capable of being programmed to provide a variety of different logic functions.

When a digital system is designed using a PLD, changes in the design can easily be made by changing the programming of the PLD without having to change the wiring in the system.

- Programmable Logic Devices (PLDs) include:
  - Programmable Logic Arrays (PLAs)
  - Programmable Array Logic (PALs)
  - Complex Programmable Logic Devices (CPLDs)
  - Field Programmable Gate Arrays (FPGAs)
Programmable Logic Array

- A Programmable Logic Array (PLA) performs the same basic function as the ROM.
- A PLA with \(n\) inputs and \(m\) outputs can realize
  - \(m\) functions
  - of \(n\) variables
- A PLA consists of
  - An AND array to realize product terms
  - An OR array to realize the output functions
- Thus, a PLA implements SOP expressions.
PLA – Basic Structure

AND Array

OR Array

n Input Lines

k Word Lines

m Output Lines

PLA
Building Logic Functions with PLA

F_0 = \Sigma m(0, 1, 4, 6)
F_1 = \Sigma m(2, 3, 4, 6, 7)

What functions are realized by the PLA for F_2 and F_3?
Building Logic Functions using PLA

Wired-AND

Wired-OR

same functions as previous slide
Building Logic Functions using PLA

Inputs

a b c d

AND

a'bd
ab'd
a'b'c'
b'c
b c
bc

OR

F1 F2 F3

Word Lines

Outputs
Programmable Array Logic

- The **Programmable Array Logic** (PAL) is a special case of the PLA
  - AND array is programmable
  - OR array is fixed
- A PAL is less expensive than the more general PLA.
- A PAL is easier to program.
Building Logic Functions using PAL

(a) Unprogrammed

(b) Programmed

\[ l_1 \cdot l_2' + l_1' \cdot l_2 \]
Building Logic Functions using PAL
Complex Programmable Logic Device

- A Complex Programmable Logic Device integrates many PLAs (or PALs) onto a single chip.
- In addition to the individual PLAs (or PALs) being programmable, the interconnection between these components is also programmable.
- A small digital system can be realized using
  - A single CPLD
  - Necessary memory elements (i.e. flip-flops)
Architecture of the Xilinx XCR3064XL CPLD

(Figure based on figures and text owned by Xilinx, Inc., Courtesy of Xilinx, Inc. © Xilinx, Inc. 1999-2003. All rights reserved.)
CPLD

CPLD Function Block and Macrocell
(A Simplified Version of XCR3064XL)
Field Programmable Gate Array

- A **Field Programmable Gate Array** consists of:
  - An array of identical logic cells
  - An interconnection network between logic cells
- The logic cells, aka Configurable Logic Blocks (CLBs), are programmable.
- The interconnection between CLBs is also programmable.
- The CLBs are surrounded by a ring of I/O blocks
  - which connect the CLBs to the I/O pins.
Layout of a Typical FPGA

Configurable Logic Block

I/O Block

Interconnect Area
Simplified Configurable Logic Block

\[ \ast = \text{Programmable MUX} \]
Implementation of a Lookup Table

<table>
<thead>
<tr>
<th>abcd</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>1111</td>
<td>1</td>
</tr>
</tbody>
</table>

- A 4-input Lookup Table (LUT) is, essentially, a reprogrammable ROM with 16 1-bit words.
Questions?