11-Multiplexers, Decoders, Encoders and Read-Only Memory

Text: Unit 9

ECEGR/ISSC 201
Digital Operations and Computations
Winter 2011

Overview

- Mutliplexers
- Three-State Buffers
- Decoders/Encoders
- Read Only Memory

Introduction

- Basic building blocks have been discussed thus far: NOT, AND ,OR, NOR, NAND, etc gates
- Often, multiple gates are available in single integrated circuits
- We will next study some common types of these ICs

Introduction

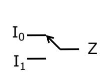
- Nomenclature
  - Small scale integrated (SSI): NAND, NOR, NOT, AND, OR and Flip-Flops
  - Medium scale integrated (MSI): adders, multiplexers, decoders, registers, etc
  - Large scale integration (LSI): 12-100 gates
  - Very Large Scale Integration (VLSI): 100-1000s of gates

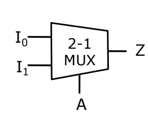
Multiplexers (MUX)

- A data selector
  - Data inputs
  - Control input(s)
- Acts as a multi-pole switch
- Example: source selection in a home stereo
- Widespread use in telecommunications

Multiplexers (MUX)

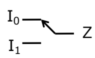
- Acts as a multi-pole switch
- Example 2-to-1 MUX

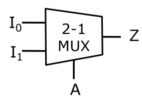




Multiplexers (MUX)

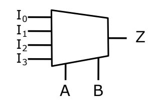
- If A = 0, then $Z = I_0$
- If A = 1, then $Z = I_1$
- $Z = A'I_0 + AI_1$





Multiplexers (MUX)

- In a 4-to-1 MUX, 2 control signals are needed, A and B
  - $Z = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$
- Other MUXs are possible
  - 8-to-1
  - 2^n-to-1



Example

- Write equation describing the operation of an 8-to-1 MUX
- Z =?

Example

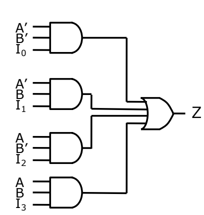
- Write equation describing the operation of an 8-to-1 MUX
- $Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$
- What is the output when A = B =0, C = 1?

Example

- Write equation describing the operation of an 8-to-1 MUX
- $Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$
- What is the output when A = B =0, C = 1?
  - $Z = I_1$

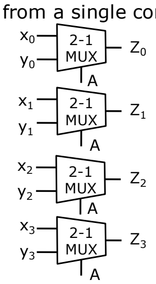
Multiplexers (MUX)

- Internal gates of a 4-to-1 MUX



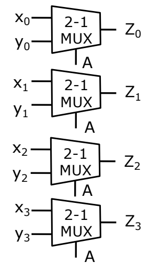
Multiplexers (MUX)

- MUXs are often used to select multi-bit words from a single control signal



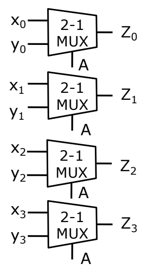
Multiplexers (MUX)

- If A = 0, then the output word is $x_0x_1x_2x_3$

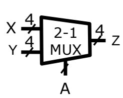


Multiplexers (MUX)

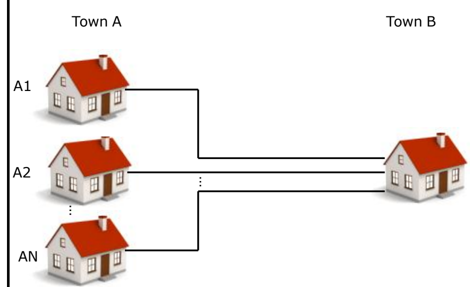
- Bus representation



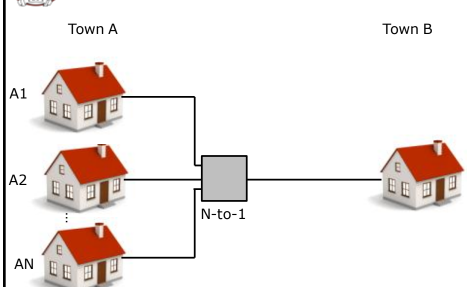
Often, a single thick line is used to indicate a bus



Multiplexers (MUX)

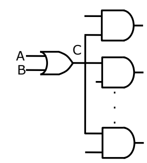


Multiplexers (MUX)

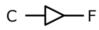


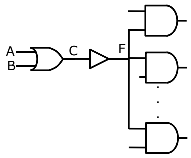
Buffers

- Complications can arise when the output of a gate is connected to several devices



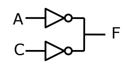
Buffers

- Solution: use a buffer (non-inverting)
- F = C 



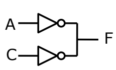
Three-State Buffers

- Consider the case in which the output of two gates are tied together
- Develop the truth table for F



Three-State Buffers

- Conflicts occur when A and C are not the same
- An unknown value (X) occurs
- How do we prevent this?



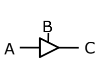
A	C	F
0	0	1
0	1	(X)
1	0	(X)
1	1	0

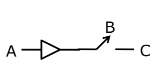
Three-State Buffers

- Complications can arise when the outputs of two or more logic gates are connected together
- Voltage levels can vary resulting in indeterminate values (not 0, not 1)
- Solution: use three-state buffers

Three-State Buffers

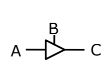
- Three-state buffer
  - When B = 1; C = A
  - When B = 0; C acts as an open circuit (Hi Z)
- Open circuit: no flow of current



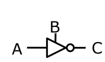


Three-State Buffers

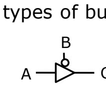
- Truth tables for various types of buffers



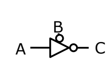
B	A	C
0	0	Z
0	1	Z
1	0	0
1	1	1



B	A	C
0	0	Z
0	1	Z
1	0	1
1	1	0



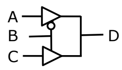
B	A	C
0	0	0
0	1	1
1	0	Z
1	1	Z



B	A	C
0	0	1
0	1	0
1	0	Z
1	1	Z

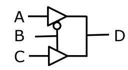
Three-State Buffers

- Analyze the following circuit
- Two, three-state buffers controlled by B
- Coupled output D



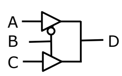
Three-State Buffers

- When B =0, what is D?
  - Top behaves as a non-inverting buffer, so D =A
  - Bottom state is Hi-Z, so C is not fed through



Three-State Buffers

- When B = 1, what is D?
  - Top state is Hi-Z, so A is not fed through to D
  - Bottom behaves as a non-inverting buffer, so D =C

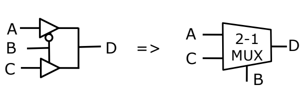


Three-State Buffers

- When B =0, D =A
- When B = 1, D =C
- What does this type of operation remind you of?

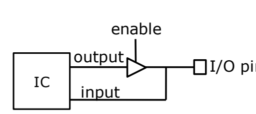
Three-State Buffers

- A 2-to-1 MUX!



Three-State Buffers

- Three-state buffers are also used in ICs with bi-directional pins (same pin can be used for input or output)

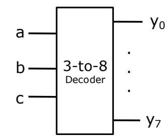


Decoders and Encoders

- We have used decoders and encoders in the labs
- What do they do?

Decoders

- Decoders are a commonly used type of IC
- Decodes n-bit input code into m-bit output code



Decoders

- There may be unused input combinations
- We discuss "Line Decoders"
- Outputs minterms based on inputs

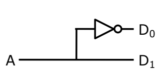
Decoders

- A simple 1-to-2 decoder
  - 1 input: A
  - 2 outputs: D_0 and D_1
- Assume we want the outputs to behave according to the following truth table
- How can we build this?

A	D_0	D_1
0	1	0
1	0	1

Decoders

- A simple 1-to-2 decoder
  - 1 input: A
  - 2 outputs: D_0 and D_1



A	D_0	D_1
0	1	0
1	0	1

Decoders

- Assume that we have a system that accepts input binary numbers between 0 and 3
- Want to light four different colored bulbs depending on the input value
  - If input is 0, red bulb lights
  - If input is 1, yellow bulb lights
  - If input is 2, blue bulb lights
  - If input is 3, green bulb lights
- Design the decoder

Decoder

- Assign
  - If input is 0, red bulb =>D_0
  - If input is 1, yellow bulb =>D_1
  - If input is 2, blue bulb =>D_2
  - If input is 3, green bulb =>D_3

A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Decoders

- 3-to-8 decoder

a	b	c	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
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							1	
1	1	1								1

The remaining entries are 0

Decoders

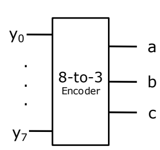
- A general n-to-2^n decoder generates all 2^n minterms (or maxterms) of the n input variables
  - $y_i = m_i$ for i = 0 to $2^n - 1$ (noninverted outputs)
  - $y_i = m_i' = M_i$ for i= 0 to $2^n - 1$ (inverted outputs)

Encoders

- Encoders perform the opposite function as a decoder
- Outputs generate binary code corresponding to input

Encoders

- Octal-to-Binary Encoder



Encoders

- Octal-to-Binary Encoder
- Write the equations for a, b and c

Encoders

- Octal-to-Binary Encoder
- Write the equations for a, b and c
  - $a = y_4 + y_5 + y_6 + y_7$
  - $b = y_2 + y_3 + y_6 + y_7$
  - $c = y_1 + y_3 + y_5 + y_7$

Encoders

- Priority Encoder: if more than one input is 1, the most significant 1 takes precedent
- V: valid output indicator, needed to distinguish all 0 input from all 0 input except D_0

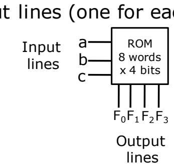
D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

Read-Only Memories

- Read-Only Memory (ROM): array of semiconductor devices that are interconnected to store an array of binary data
- Data can be read, but it cannot be changed (easily)
- Non-volatile

Read-Only Memories

- Consider ROM that holds 8, 4-bit words
- Each of the 8 words has a unique address (example: 000, 110, 111, etc.)
- Need three input lines to select an address (a, b, c)
- Four output lines (one for each word bit)



Read-Only Memories

- Truth Table

A	B	C	F_0	F_1	F_2	F_3
0	0	0				
0	0	1				
0	1	0				
0	1	1		DATA		
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Example

- What is the output word if A = B =1, C =0?

A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

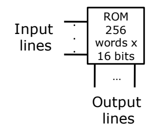
Example

- What is the output word if A = B =1, C =0?
- 1111

A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

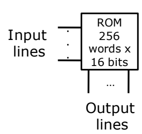
Example

- Design (determine the number of inputs and outputs) for a 256-word, 16 bit ROM



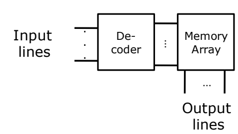
Example

- Design (determine the number of inputs and outputs) for a 256-word, 16 bit ROM
- Inputs $2^x = 256 \Rightarrow x = 8$
- Outputs 16

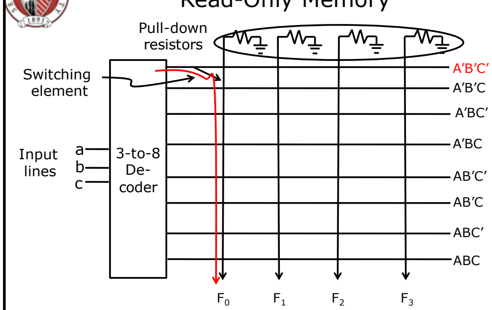


Read-Only Memory

- Basic components of a ROM are a decoder and memory array



Read-Only Memory

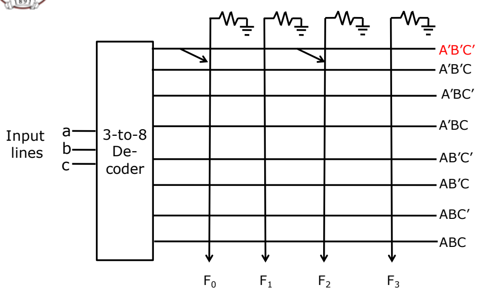


Example

- Draw the switching elements on the ROM to realize the following truth table

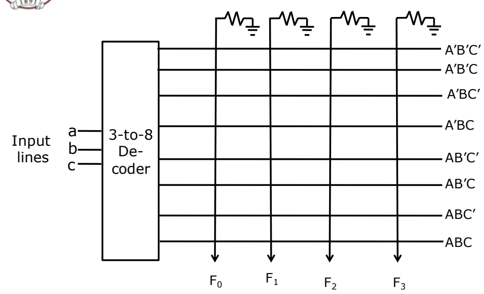
A	B	C	F_0	F_1	F_2	F_3
0	0	0	1	0	1	0
0	0	1	1	0	1	0
0	1	0	0	1	1	1
0	1	1	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	1
1	1	0	1	1	1	1
1	1	1	0	1	0	1

Read-Only Memory



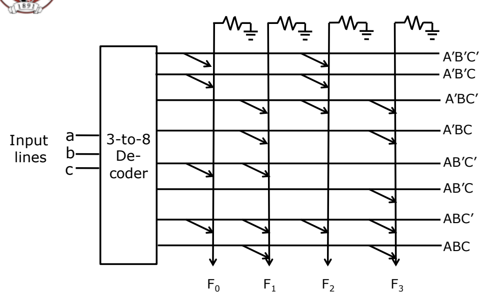
Dr. Louie 55

Example



Dr. Louie 56

Example



Dr. Louie 57

Read-Only Memory

- Common Types of ROM
  - Mask-Programable ROMs
  - Programmable ROMs (PROMs)
  - Electrically erasable programmable ROMs (EEPROMs)

Dr. Louie 58

Read-Only Memory

- Mask-Programable ROMs
  - Data is permanently stored at time of manufacture
  - A special mask is required, which is expensive
  - Usually selected if several thousand duplicates are used

Dr. Louie 59

Read-Only Memory

- Electrically erasable programmable ROMs (EEPROMs)
  - Allows for the ROM data to be erased and new data stored
  - Used in developmental stages of a digital system
  - Similar to flash memory, but there is a different underlying mechanism

Dr. Louie 60