EE2020
Digital Fundamentals

(L5 - Combinational logic)

Prof. Massimo Alioto

Dept of Electrical and Computer Engineering

Email: ***massimo.alioto@nus.edu.sg***

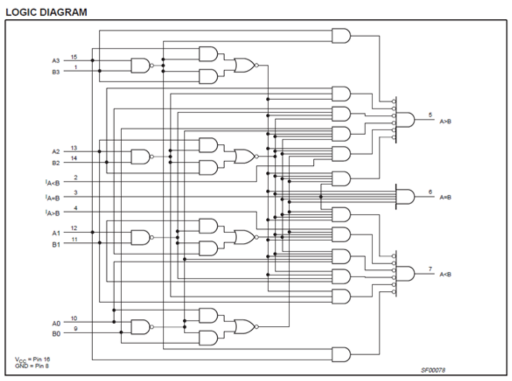
Outline

- Introduction
- Binary adders
  - Half adders, full adders, ripple adders.
- Magnitude comparators
- Decoders, BCD to 7-segment decoders
- Encoders, Multiplexers
- Demultiplexers
- Tri-state logic elements

Introduction

- There are two types of logic circuits
  - Combinational and sequential logic circuits
- Combinational logic
  - The output depends only on the current inputs
- Sequential logic
  - The output depends on both past and present inputs, which implies that there is a memory element in the sequential circuit
- Combinational logic circuits implement some commonly used logic functions in a single chip
- The scale of integration for combinational logic is considered medium (10- 1000gates), thus is called **MSI** (medium scale integration)
- The advantages of using MSI are low cost, small area, low power consumption and high reliability.

4-bit magnitude comparator – logic gate implementation vs. MSI

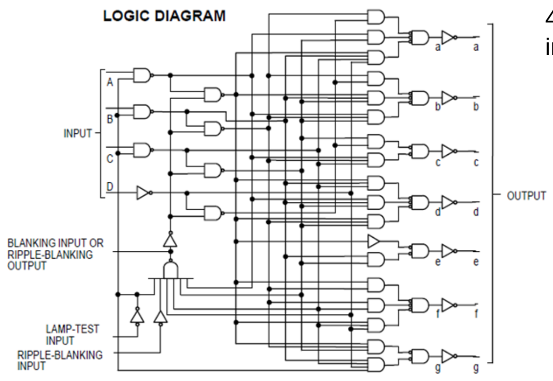


7485 4-bit magnitude comparator circuit diagram

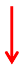
30 gate ICs are needed if it were implemented by individual gates

7485 (MSI)
(4-bit magnitude comparator)

BCD-to-7 segment decoder – Logic gate implementation vs. MSI



44 gate ICs are needed if it were implemented by individual gates



74’47 (MSI)
(BCD-to-7 segment decoder)

EE2020 Digital Fundamentals - XU YP 5

Approach to Learning MSI Devices

- Understand the function of the MSI devices
- Understand how the function of the device is described? (Voltage table or Boolean expression)
- What are the practical applications of the MSI devices?
- Design using MSI(s)

EE2020 Digital Fundamentals - XU YP 6

Binary Adders

- Perform addition of two binary numbers
- Half-adder
- Full adder
- MSI adder

Half Adders

- It's a one bit binary adder with two inputs of A_i and B_i

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

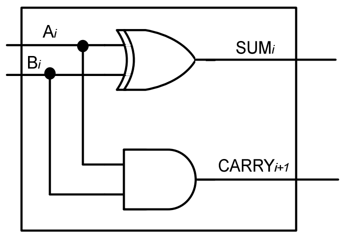
Carry → C_{i+1}

$A: A_n \ldots A_{i+1} A_i \ldots A_0$

$B: B_n \ldots B_{i+1} B_i \ldots B_0$

Sum → S_i

A_i	B_i	Sum_i	$Carry_{i+1}$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

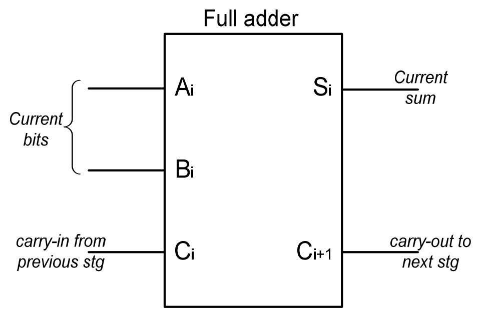


$$S_i = \bar{A}_i B_i + Ai\bar{B}_i$$
$$C_{i+1} = AiBi$$

Carry in from *i-1* bit cannot be added

Full Adders

- Full adders can use the carry bit from the previous stage of addition



A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adders (cont.)

K-map for SUM

A_i / B_iC_i	0	1
00	0	1
01	1	0
11	0	1
10	1	0

Note: C_{i+1} is not a MSOP, but less overall hardware is reqd. if we use this expression. It allows sharing of A_i XOR B_i between SUM_i and C_{i+1}.

K-map for CARRY

A_i / B_iC_i	0	1
00	0	0
01	0	1
11	1	1
10	0	1

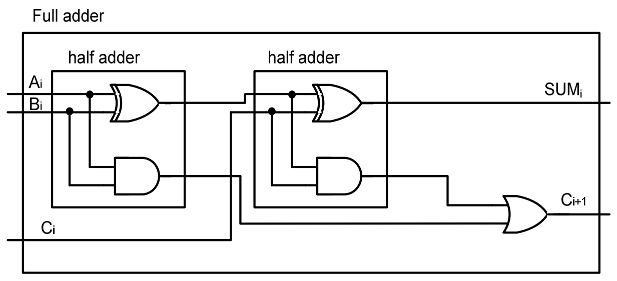
$$\begin{aligned} SUM &= \overline{A}_i\overline{B}_iC_i + \overline{A}_iB_i\overline{C}_i + A_i\overline{B}_i\overline{C}_i + A_iB_iC_i \\ &= \overline{A}_i(\overline{B}_iC_i + B_i\overline{C}_i) + A_i(\overline{B}_i\overline{C}_i + B_iC_i) \\ &= \overline{A}_i(B_i \oplus C_i) + A_i(\overline{B_i \oplus C_i}) \\ &= A_i \oplus B_i \oplus C_i \end{aligned}$$

$$\begin{aligned} C_{i+1} &= A_iB_i + A_i\overline{B}_iC_i + \overline{A}_iB_iC_i \\ &= A_iB_i + C_i(A_i\overline{B}_i + \overline{A}_iB_i) \\ &= A_iB_i + C_i(A_i \oplus B_i) \end{aligned}$$

Full Adder Circuit

$$SUM = (A_i \oplus B_i) \oplus C_i$$

$$C_{i+1} = A_i B_i + C_i (A_i \oplus B_i)$$

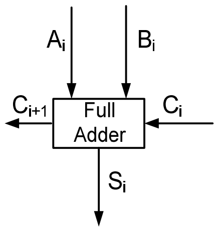


Voltage Table

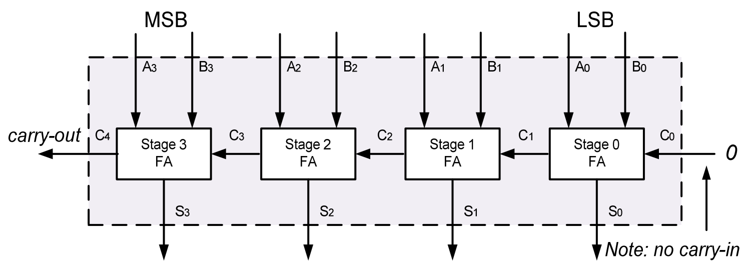
A_i	B_i	C_i	S_i	C_{i+1}
L	L	L	L	L
L	L	H	H	L
L	H	L	H	L
L	H	H	L	H
H	L	L	H	L
H	L	H	L	H
H	H	L	L	H
H	H	H	H	H

Note: A full adder adds 3 bits. Can also consider as first adding first two and then the result with the carry

Parallel Adders



C_4	C_3	C_2	C_1
	A_3	A_2	A_1 A_0
	B_3	B_2	B_1 B_0



4 full adders cascaded

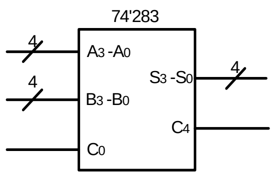
Parallel Adders (cont.)

- In general, *n* full adders can be used to form a *n*-bit adder
- Carry ripple effect
  - Carry bits have to propagate from one stage to the next
  - Inherent propagation delays associated with this carry propagation
  - Output of each full adder is therefore not available until the carry-in from the previous stage is calculated
  - As the carries *ripple* through the chain → also known as *ripple* adders
- This relatively slow rippling effect is substantially minimized in commercial MSI chips by using *carry look ahead circuitry*
- Otherwise, addition may be unacceptably slow ~ 1.28μS to add 32 bit binary numbers

*Read about how carry look-ahead circuits work yourself

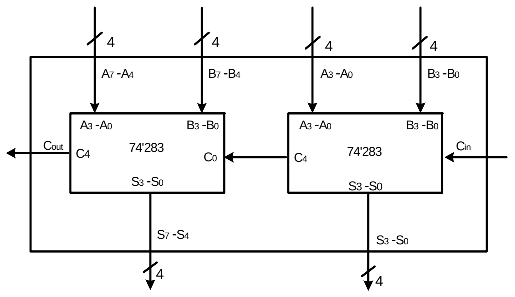
MSI Parallel Adders

4-bit Parallel Adder



- Commonly used chips are 74'83 and 74'283
- They are functionally the same, but have different pin configurations
- Both feature carry look ahead circuitry

An 8-bit adder constructed from two 74'283 adders

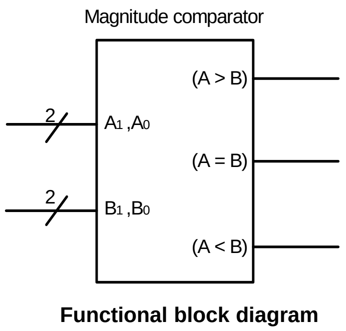


Binary Subtractors

- Binary adders can be used to perform subtraction if the two binary numbers are their 2’s complement
- Therefore, no separate subtractor MSI is available

Magnitude Comparator

- Outputs are functions of relative magnitudes of 2 input binary numbers, A and B



Functional block diagram

Magnitude Comparator truth table

Two-bit magnitude comparator

A_1	A_0	B_1	B_0	(A > B)	(A = B)	(A < B)
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

K-maps for A>B and A<B

A>B

B_1B_0 \ A_1A_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

A<B

B_1B_0 \ A_1A_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$$(A > B) = A_1\overline{B}_1 + A_0\overline{B}_1\overline{B}_0 + A_1A_0\overline{B}_0$$

$$(A < B) = \overline{A}_1B_1 + \overline{A}_1\overline{A}_0B_0 + \overline{A}_0B_1B_0$$

K-map for A=B

A=B

B_1B_0 \ A_1A_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

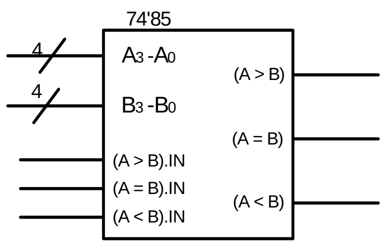
$$(A=B)=\bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0+\bar{A}_1A_0\bar{B}_1B_0$$
$$+A_1A_0B_1B_0+A_1\bar{A}_0B_1\bar{B}_0$$

This can be generated indirectly using (A<B) and (A>B)

$$(A=B)=\overline{(A<B)}\cdot\overline{(A>B)}$$

MSI Magnitude Comparator

- 74’85 magnitude comparator
  - Compares two 4-bit inputs
  - Supports cascading



Functional block diagram

Cascading 74’85

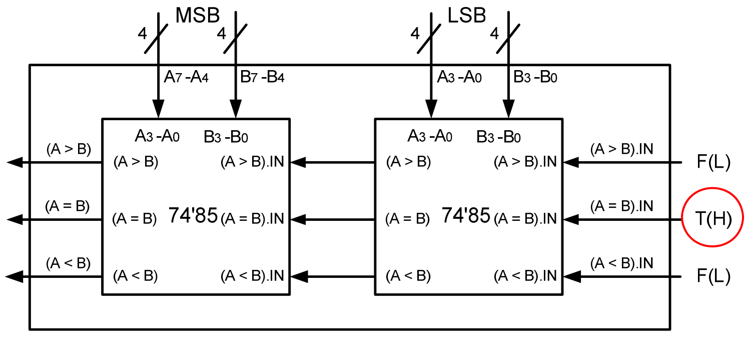
- Cascading inputs allow realization of a comparator of length 4N.
- *Note*:
  - if the 4 MSB’s of A are greater than the 4 MSB’s of B, then A>B is True regardless of cascading inputs from previous stage
  - if the 4 MSB’s of A are less than the 4 MSB’s of B, then A<B is True regardless of cascading inputs from previous stage.
  - if the 4 MSB’s of A are equal to the 4 MSB’s of B, then output depends on cascading inputs from previous stage

Voltage table of 74’85

Voltage Table

Comparing Inputs				Cascading Inputs			Outputs		
A_3, B_3	A_2, B_2	A_1, B_1	A_0, B_0	(A>B).IN	(A<B).IN	(A=B).IN	(A>B)	(A<B)	(A=B)
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L

8-bit Magnitude Comparator

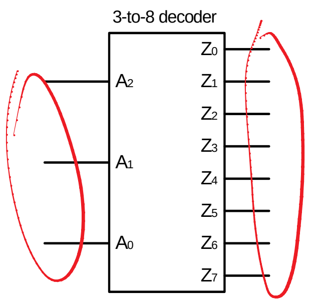


8-bit magnitude comparator designed using two 4-bit comparators

Decoder

- A decoder is a circuit element that will decode an N-bit code.
- It activates an appropriate output line as a function of the applied N-bit input code

Functional block diagram

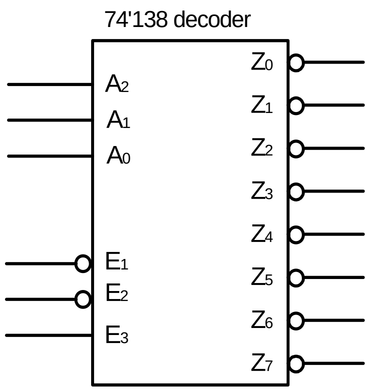


Truth Table

A_2	A_1	A_0	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

MSI Decoder 3-8

- A decoder can have up to 2^N output lines for N inputs
- MSI decoders are available as 2-4, 3-8, 4-10 decoders, etc
- *Example*: The 74'138 decoder 3-8 decoder



Functional block diagram

MSI Decoder 74'138

Voltage Table (Active low)

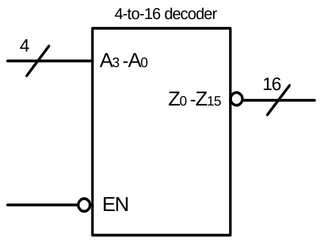
Inputs						Outputs							
E_1	E_2	E_3	A_2	A_1	A_0	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

MSI Decoder 74’138 – cont.

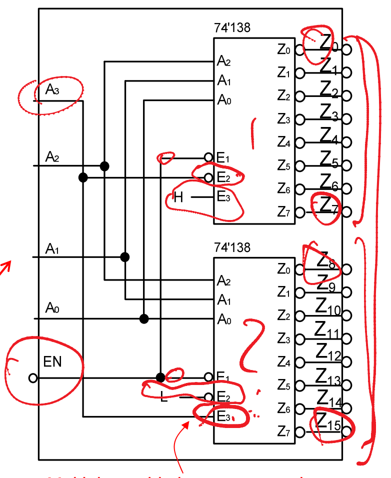
- A 74’138 functions as a decoder only if all 3 ENABLE inputs are asserted. Otherwise, all 8 active low outputs are de-asserted (H voltage).
- The ENABLE inputs can also be conveniently used for expanding the decoder
- In general, larger decoders can be constructed from smaller decoders using some additional circuitry

Cascading two 3-8 Decoders (74’138)

Realization of a 4-16 decoder



Functional block diagram

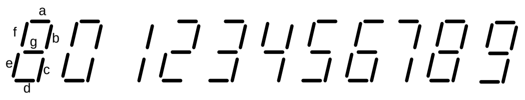


Top decoder is enabled only if A3 = 0.
Bottom decoder is enabled only if A3 = 1.
Hence, output is unique.

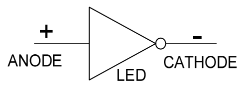
Multiple enable inputs ensure that no inverters are needed!

BCD-to-7 Segment Decoder

- Converts a BCD number into signals required to display that number on a 7-segment display

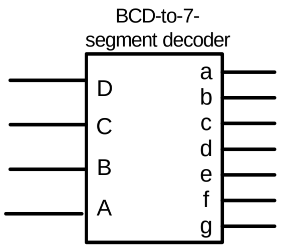


A 7-segment display. Each segment is an LED which will light when a logic T signal is applied to it

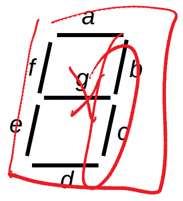


- 7-segment displays are of 2 types: *common anode* and *common cathode*
- Common anode display has all LED anodes connected and is active low, whereas the common cathode display is active high

BCD-to-7 Segment Decoder – cont.



Functional block diagram



Truth Table

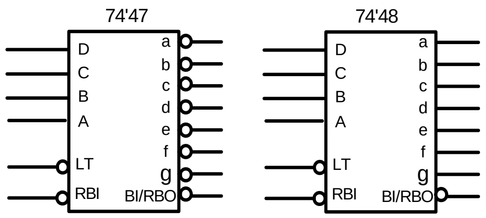
D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X

MSI BCD-to-7 Segment Decoder

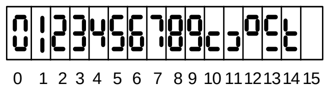
- Examples of commercial BCD–to-7 segment decoders are 74’47 (active low) and 74’48 (active high)

LT: Lamp test
RBI: Ripple blanking input

BI/RBO: Blanking input/ Ripple-blanking output



Functional block diagram for 74’47 and 74’48



Numerial designations – resultant displays

Voltage tables for 74’48

Inputs							I/O	Outputs						
Decimal or Function	**LT**	**RBI**	**D**	**C**	**B**	**A**	**BI/RBO**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	H	H	L	L	L	L	H	H	H	H	H	H	H	L
1	H	X	L	L	L	H	H	L	H	H	L	L	L	L
2	H	X	L	L	H	L	H	H	H	L	H	H	L	H
3	H	X	L	L	H	H	H	H	H	H	H	L	L	H
4	H	X	L	H	L	L	H	L	H	H	L	L	H	H
5	H	X	L	H	L	H	H	H	L	H	H	L	H	H
6	H	X	L	H	H	L	H	L	L	H	H	H	H	H
7	H	X	L	H	H	H	H	H	H	H	L	L	L	L
8	H	X	H	L	L	L	H	H	H	H	H	H	H	H
9	H	X	H	L	L	H	H	H	H	H	L	L	H	H
10	H	X	H	L	H	L	H	L	L	L	H	H	L	H
11	H	X	H	L	H	H	H	L	L	H	H	L	L	H
12	H	X	H	H	L	L	H	L	H	L	L	L	H	H
13	H	X	H	H	L	H	H	H	L	L	H	L	H	H
14	H	X	H	H	H	L	H	L	L	L	H	H	H	H
15	H	X	H	H	H	H	H	L	L	L	L	L	L	L
BI	X	X	X	X	X	X	L	L	L	L	L	L	L	L
RBI	H	L	L	L	L	L	L	L	L	L	L	L	L	L
LT	L	X	X	X	X	X	H	H	H	H	H	H	H	H

BCD-to-7 Segment Decoder Notes

- ALL SEGMENTS BLANKING: When BI.L is open or held high, normal operation is obtained. When BI.L is low, all segment outputs are low (blanked – chip is disabled)
- TEST MODE: if lamp-test input LT is low, and BI/RBO.L is open or held high, all segment outputs are high
- ZERO BLANKING: If ripple blanking input (RBI.L) is low, zero is blanked (to hide leading zeroes). If RBI.L is open or high, zeroes are displayed.
- When ripple-blanking input (RBI.L) and inputs A, B, C, and D are low and LT is high, all segment outputs go low and the ripple-blanking output (RBO.L) goes to a low level (response condition)

33

Controlling Zeroes Display

- A summary of how RBI/BI signal influences display of numbers. Note in the last one BI is an input

RBI	ABCD	BI/RBO	a, b, ---, f
L	0000	RBO=L	Blanked out
L	Non-Zero	RBO=H	As Appropriate
H	X	RBO=H	As Appropriate
X	**X**	**BI=L**	**Blanked out**

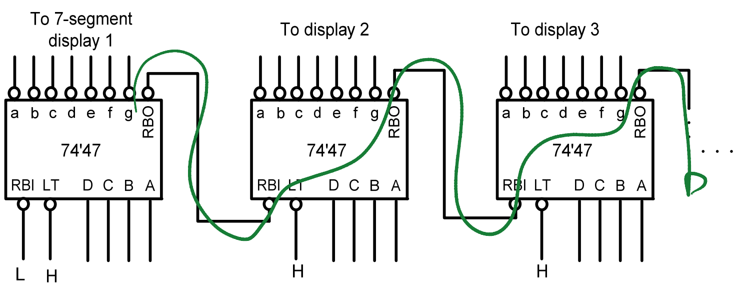
- A design choice is whether a display system should display leading 0's
- If it's OK to display leading zeros, all decoders are independent, and their RBI is set to H

EE2020 Digital Fundamentals - XU YP

34

Controlling Zeroes Display (cont.)

- If leading 0's need to be suppressed, then the decoders must be serially connected as below

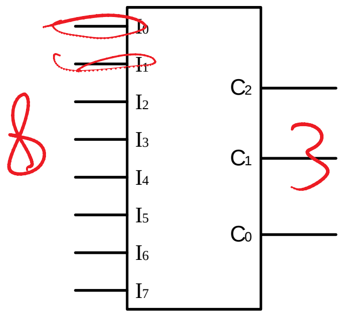


Use of RBI and RBO in a cascade of 7-segment displays

Encoder

- Perform the inverse of the decoding function
- For N different inputs, an encoder is a circuit element that generates a binary code that uniquely identifies the input

Functional block diagram



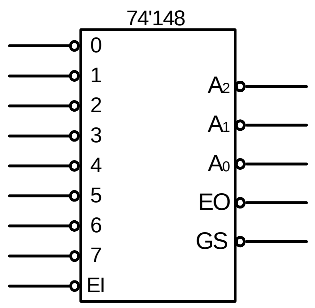
Truth Table (an 8-3 encoder)

I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	C_2	C_1	C_0
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

MSI Priority Encoder

- These encoders allow several inputs to be active simultaneously (as opposed to only one), or no input at all e.g. commercial **74’148** 8-3 priority encoder

Functional block diagram



All input/output and control elements are active low (indicated by the bubbles)

EO: Enable output (for cascading)

GS: Is asserted when at least one input is asserted (see row2, row10 of table)

MSI Priority Encoder (cont.)

- EI and EO are used for cascading 74’148

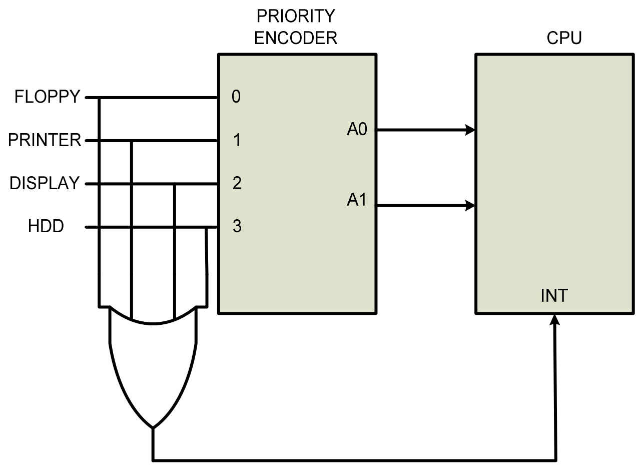
Voltage table

Inputs									Outputs				
EI	0	1	2	3	4	5	6	7	A_2	A_1	A_0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

When more than one input is activated, 74’148 encodes input with highest priority:

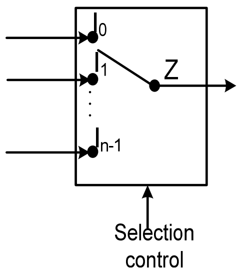
7 has priority over 6
6 has priority over 5,
etc.

Priority Encoder at Work!

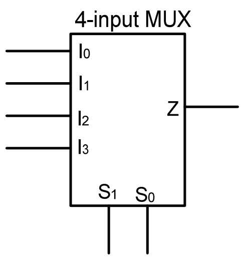


Multiplexer

- A multiplexer (MUX) is a combinational circuit element that selects data from one of many inputs and directs it to a single output



Functional block diagram



Actual truth table would have 2^6 rows corresponding to I_0, I_1, I_2, I_3, S_0 and S_1

Condensed truth table

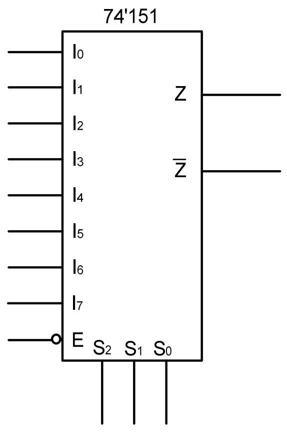
S_1	S_0	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Selection control inputs allow one of the inputs to pass through to the output

MSI Multiplexer

- MUX’s are commercially available with 2, 4, 8, and 16 inputs, and active high/low I/O
- 74’151 is an 8-input multiplexer shown here

$$Z = \overline{E}(\overline{S}_2\overline{S}_1\overline{S}_0 I_0 + \overline{S}_2\overline{S}_1 S_0 I_1 + \overline{S}_2 S_1\overline{S}_0 I_2 + \overline{S}_2 S_1 S_0 I_3 + S_2\overline{S}_1\overline{S}_0 I_4 + S_2\overline{S}_1 S_0 I_5$$
$$+ S_2 S_1\overline{S}_0 I_6 + S_2 S_1 S_0 I_7)$$



Functional block diagram

MSI Multiplexer 74’151

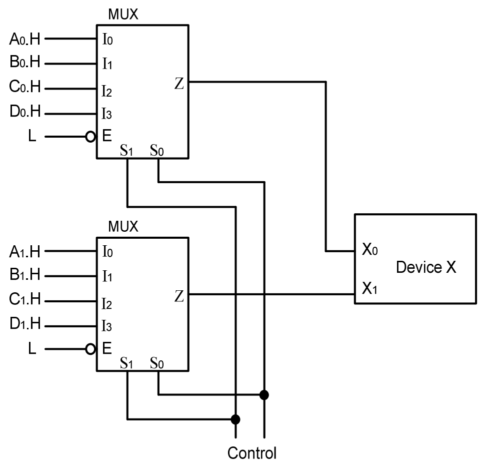
Voltage table for 74’151

E	S_2	S_1	S_0	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Z’	Z
H	X	X	X	X	X	X	X	X	X	X	X	H	L
L	L	L	L	L	X	X	X	X	X	X	X	H	L
L	L	L	L	H	X	X	X	X	X	X	X	L	H
L	L	L	H	X	L	X	X	X	X	X	X	H	L
L	L	L	H	X	H	X	X	X	X	X	X	L	H
L	L	H	L	X	X	L	X	X	X	X	X	H	L
L	L	H	L	X	X	H	X	X	X	X	X	L	H
L	L	H	H	X	X	X	L	X	X	X	X	H	L
L	L	H	H	X	X	X	H	X	X	X	X	L	H
L	H	L	L	X	X	X	X	L	X	X	X	H	L
L	H	L	L	X	X	X	X	H	X	X	X	L	H
L	H	L	H	X	X	X	X	X	L	X	X	H	L
L	H	L	H	X	X	X	X	X	H	X	X	L	H
L	H	H	L	X	X	X	X	X	X	L	X	H	L
L	H	H	L	X	X	X	X	X	X	H	X	L	H
L	H	H	H	X	X	X	X	X	X	X	L	H	L
L	H	H	H	X	X	X	X	X	X	X	H	L	H

*Inputs (I_0 – I_7) can be either H or L

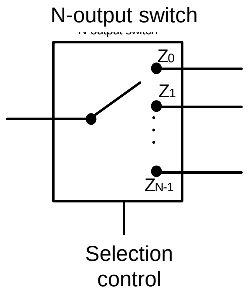
MSI Multiplexer Example Usage

- Use 2 four-input MUX's to select one of the four 2-bit data inputs to be processed by device X
- For S1 S0 inputs of 00, 01, 10, or 11, either the 2-bit data A, B, C, or D, respectively, is connected to input of device X

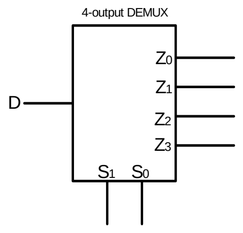


Demultiplexer

- Connects a signal to any one of a number of output lines based on selection control



Functional block diagram of 4-output demux

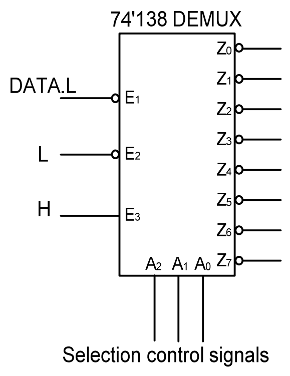


Truth table

D	S_1	S_0	Z_0	Z_1	Z_2	Z_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

MSI Demultiplexer

- 74’138 – same as the decoder chip
  - $A_0A_1A_2$ are now control signals

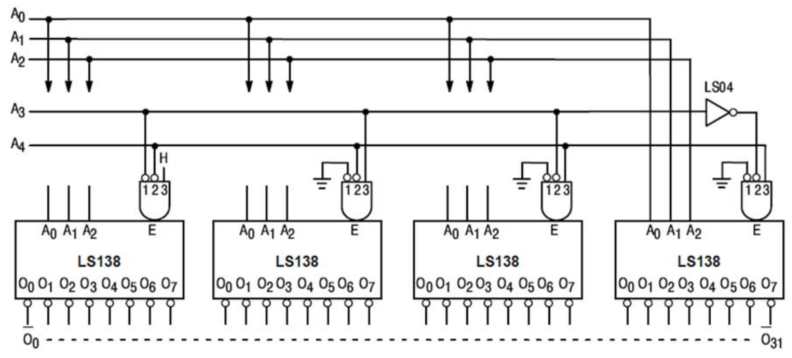


Voltage table

Enable		Data	Selection			Outputs							
E_2	E_3	E_1	A_2	A_1	A_0	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	L	X	X	X	X	H	H	H	H	H	H	H	H
X	X	H	X	X	X	H	H	H	H	H	H	H	H
L	H	L	L	L	L	L	H	H	H	H	H	H	H
L	H	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	H	L	H	H	L	H	H	H	H	H
L	H	L	L	H	H	H	H	H	L	H	H	H	H
L	H	L	H	L	L	H	H	H	H	L	H	H	H
L	H	L	H	L	H	H	H	H	H	H	L	H	H
L	H	L	H	H	L	H	H	H	H	H	H	L	H
L	H	L	H	H	H	H	H	H	H	H	H	H	L

Cascading demultiplexer

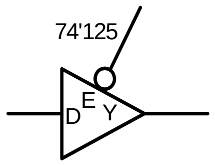
1-to-32 Demultiplexer (using 4 1-to-8 demux)



Tri-State Logic Elements

- Ordinarily, a digital device has 2 states.
  - A tri-state device has a 3rd state called the *high impedance state.* Very useful in memory buses.
- MSI example: 74'125

Functional block diagram



Voltage table

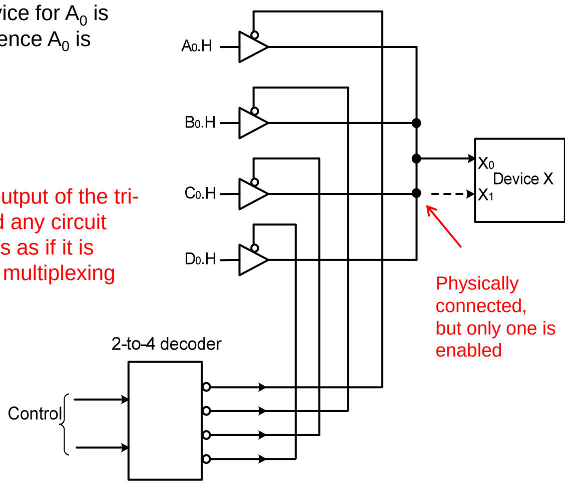
E	D	Y
L	L	L
L	H	H
H	X	(Z)

← (Z) = high impedance

Multiplexer Using Tri-State Element

When Control = 00, tri-state device for A_0 is enabled, others are disabled. Hence A_0 is connected to X_0, etc

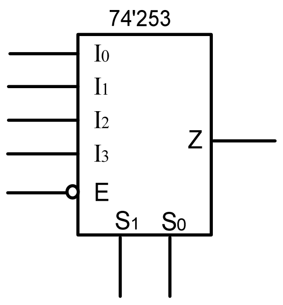
In the high impedance state, the output of the tri-state device does not drive or load any circuit connected to it, i.e. output behaves as if it is electrically disconnected – makes multiplexing possible



MSI Tri-State Logic Elements

- 74’253 is a 4-input tri-state MUX
- Note the high impedance state instead of H/L

Functional block diagram



Voltage table

Select inputs		Data inputs				Enable	Output
S_1	S_0	I_0	I_1	I_2	I_3	E	Z
X	X	X	X	X	X	H	(Z)
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

Summary

- Introduction to MSI elements
- Binary adders
  - Half adders, full adders, ripple adders.
- Magnitude comparators
  - Cascading two magnitude comparator chips
- Decoders, BCD-to-7-segment decoders
  - Cascading two decoder chips
- Encoders, Priority encoders
- Multiplexers
- Demultiplexers
- Tri-state logic elements