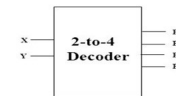


Logic Circuits: Combinational Circuits and Parts

H.R. Mahdiani
Computer Science and Engineering Department
Shahid Beheshti University
2020 (COVID19 Season!)

1

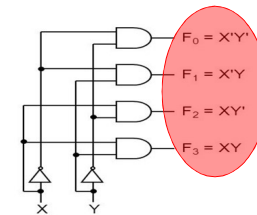
Binary Decoder



X	Y	F ₀	F ₁	F ₂	F 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

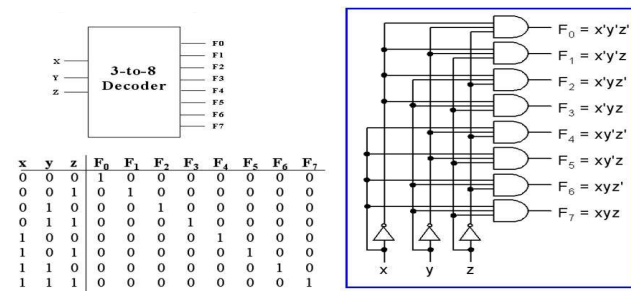


Each output is a Minterm



2

3 to 8 Binary Decoder

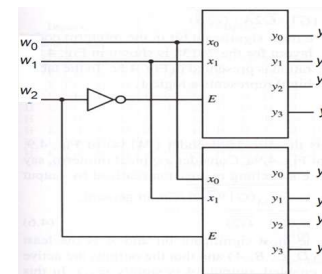


The inputs are treated as a binary number and the output selected is made active.

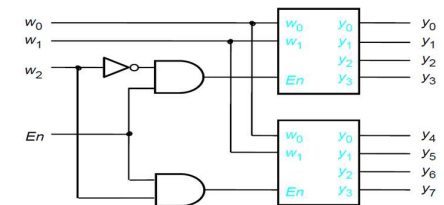
3

3 to 8 Decoder by Cascading 2 to 4 Decoders

Without Enable



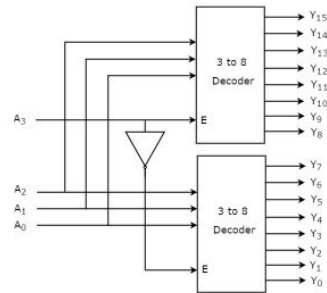
With Enable



4

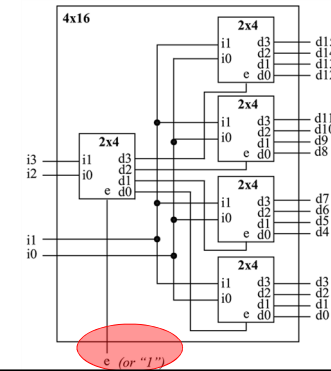
4 to 16 Decoder By Cascading 3 to 8 Decoders

- How to add enable?



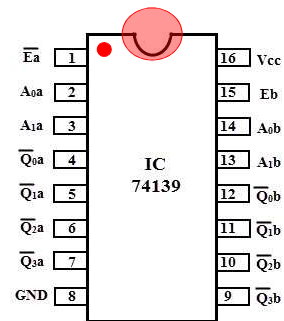
5

4 to 16 Decoder By Cascading 2 to 4 Decoders



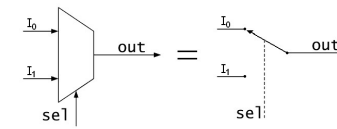
6

74139 Decoder Pin Description



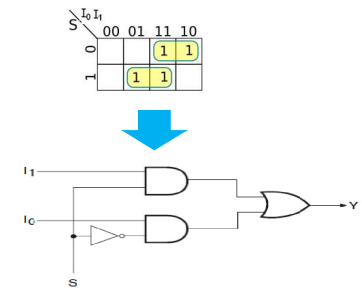
7

Multiplexer (MUX)



S	Y
0	I ₀
1	I ₁

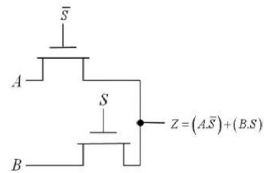
Inputs				Q
sel	I ₁	I ₀		
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1



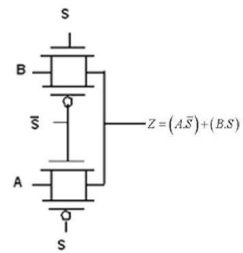
8

2:1 MUX in Pass Transistor Logic

Using NMOS Transistors



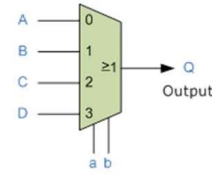
Using Transmission Gate



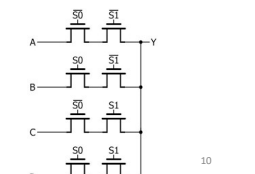
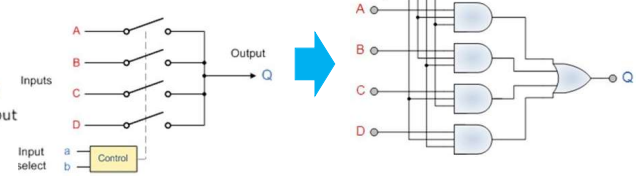
9

4:1 MUX

Multiplexer Symbol



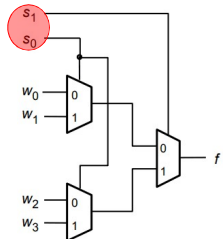
Addressing		Input Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D



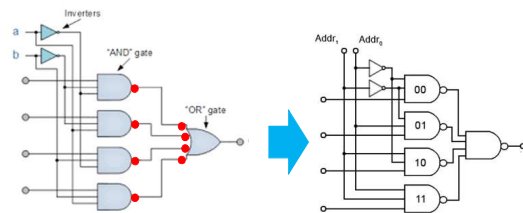
10

4:1 MUX By Cascading 2:1 MUX

Cascading 2:1 MUXes



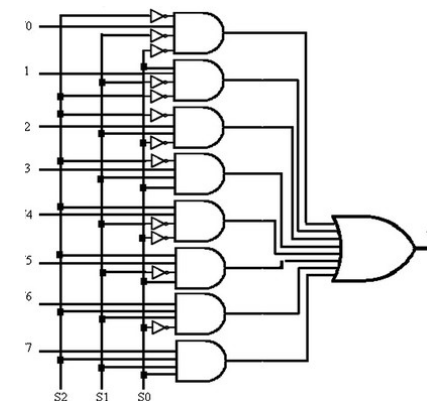
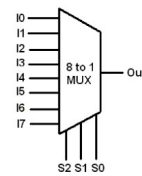
Using only NAND Gates



11

8:1 MUX

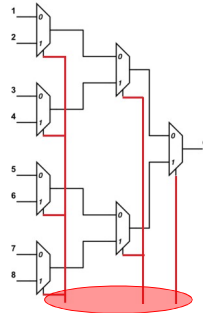
Symbol



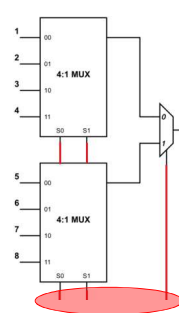
12

8:1 MUX

Cascading 2:1 MUX



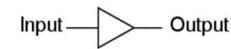
Cascading 4:1 and 2:1 MUX



13

Tri-state Buffer

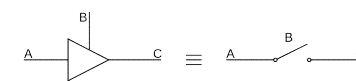
Buffer



Input	Output
0	0
1	1

Two states

Tri-state Buffer



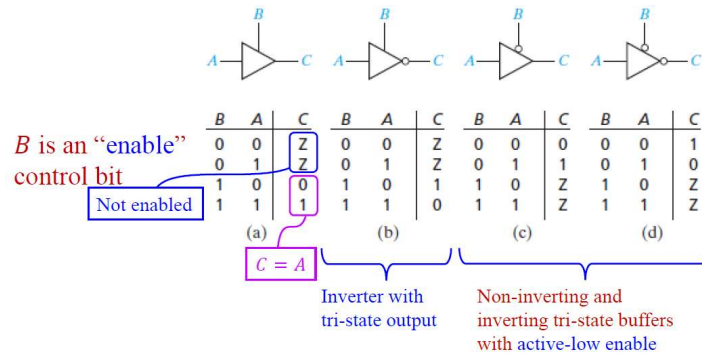
Truth table

A	B	Output
0	0	High-Z
0	1	0
1	0	High-Z
1	1	1

Three states

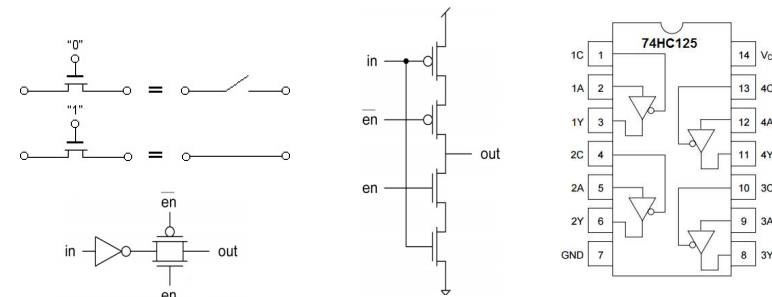
14

Different Types of Tristate Buffer



15

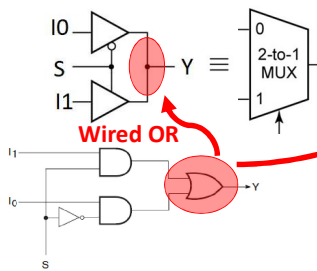
Tri-state Buffer Internal Circuit



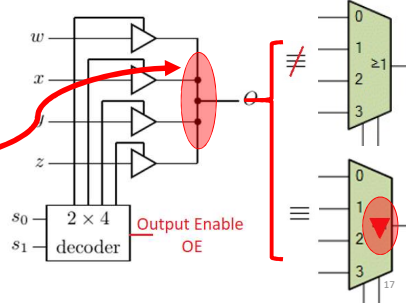
16

Building MUX Using Tristate Buffers

2:1 MUX (two state output)

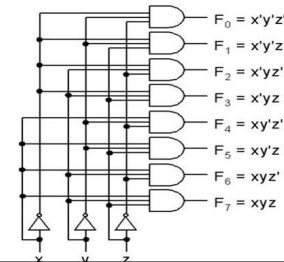


4:1 MUX (Tri-state Output)



Function Implementation Using Decoders

- A "n to 2ⁿ Decoder" implements all 2ⁿ Minterms
 - Decoder=Minterm Generator
- Any 3-input function can be implemented by ORing suitable Minterms



18

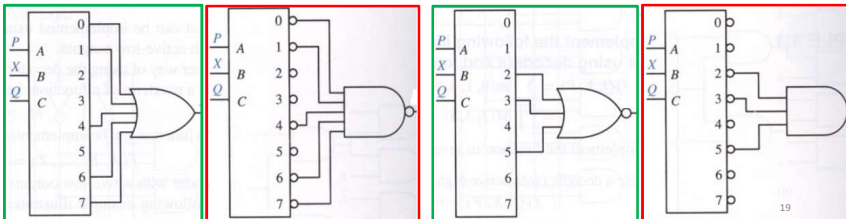
Function Implementation Using Decoders

- Implement function $f(Q, X, P) = \sum m(0, 1, 4, 6, 7)$ using a 3 to 8 decoder

Using Minterms

Using Maxterms

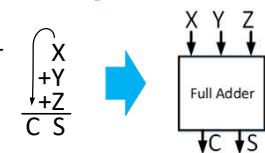
Active High Output Decoder Active Low Output Decoder Active High Output Decoder Active Low Output Decoder



19

Function Implementation Using Decoders

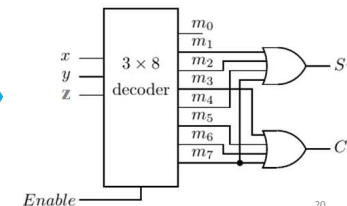
- Implement a 1-bit adder using a decoder
 - X, Y, Z, S, and C are 1-bit values



X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

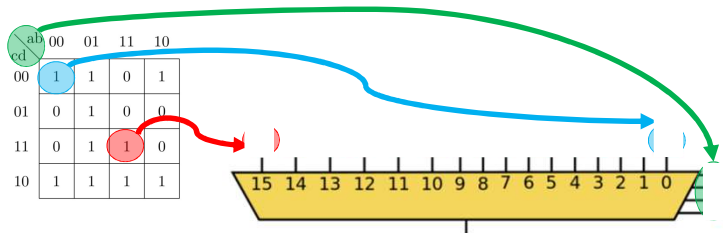
$$\text{Sum} = \sum m(1, 2, 4, 7)$$

$$\text{Carry} = \sum m(3, 5, 6, 7)$$



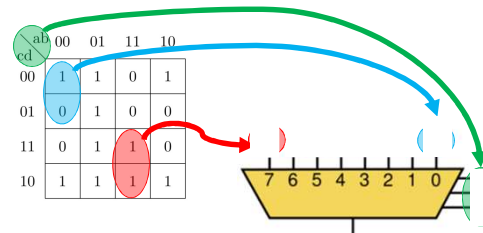
20

Function Implementation Using MUX



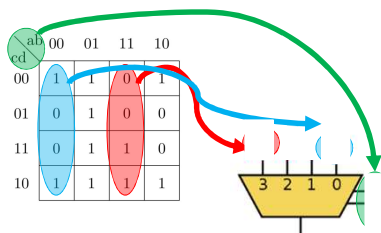
21

Function Implementation Using MUX



22

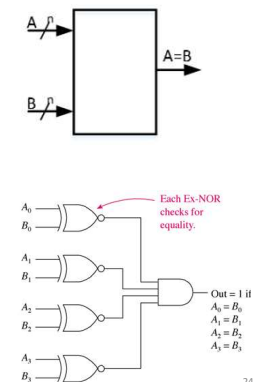
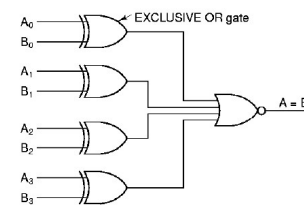
Function Implementation Using MUX



23

n-bit Comparator

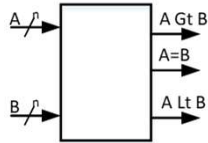
- Equality Comparator
 - Bitwise comparison
 - Works for both signed and unsigned numbers



24

n-bit Comparator

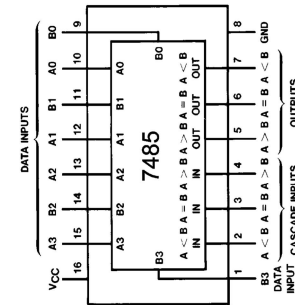
- Greater than comparator
 - Turns into Less than comparator by swapping its inputs
- Full Comparator
 - Each output might be constructed based on two others



25

7485 Unsigned 4-bit Comparator

- Special design for cascading

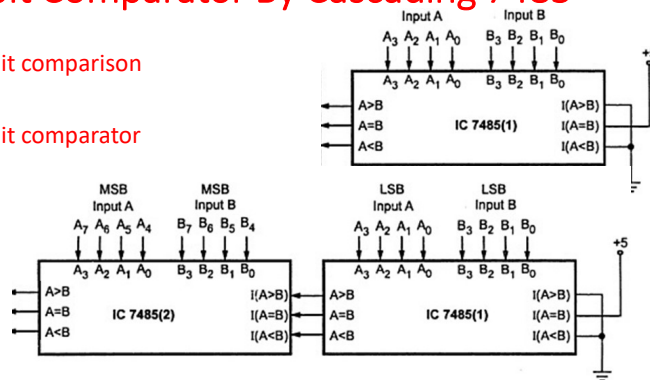


Comparing Inputs				Cascading Inputs			Outputs		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A = B	A < B	A > B	A = B	A < B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

26

8-bit Comparator By Cascading 7485

- 4-bit comparison
- 8-bit comparator

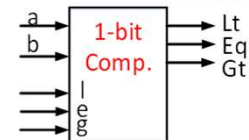


27

N-bit Comparator Iterative Design

- First implement a 1-bit comparator
 - It should be cascable
- Cascadable 1-bit comparator design

$$\begin{cases}
 a=1 \text{ \& } b=0: & Lt=0, Eq=0, Gt=1 \\
 a=b & : Lt=1, Eq=1, Gt=0 \\
 a=0 \text{ \& } b=1: & Lt=1, Eq=0, Gt=0
 \end{cases}$$

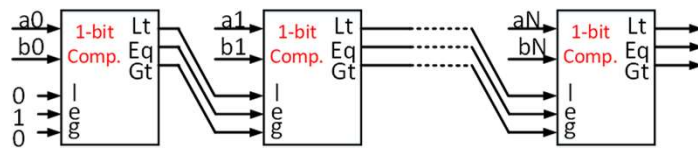


ab	0	1
00	0	1
01	1	1
11	0	1
10	0	0
	Lt	
ab	0	1
00	0	1
01	0	0
11	0	1
10	1	1
	Gt	
ab	0	1
00	0	1
01	0	0
11	0	1
10	0	0
	Eq	

28

N-bit Comparator Iterative Design

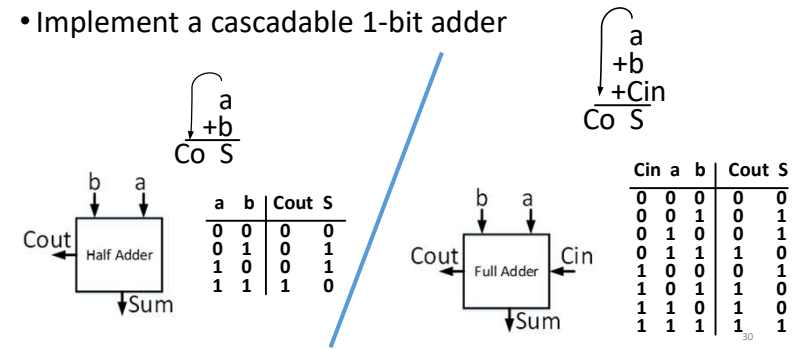
- Implement N-bit comparator by cascading N instances of 1-bit comparators



29

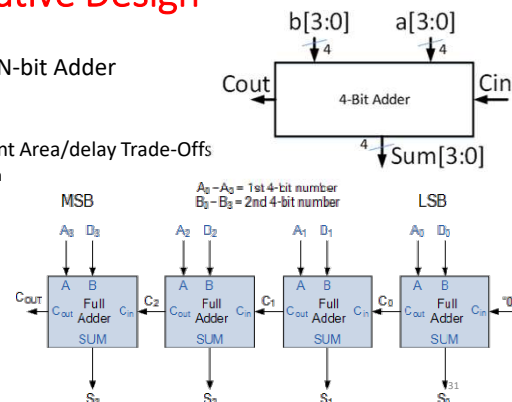
N-bit Adder Iterative Design

- Implement a cascadable 1-bit adder



N-bit Adder Iterative Design

- Cascade it to implement N-bit Adder
 - Ripple Carry Addition
 - Linear Area and Delay
 - Other adders with different Area/delay Trade-Offs
 - Carry Look Ahead Addition



74283 4-bit Adder

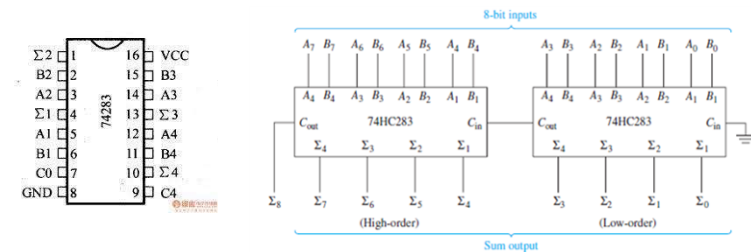


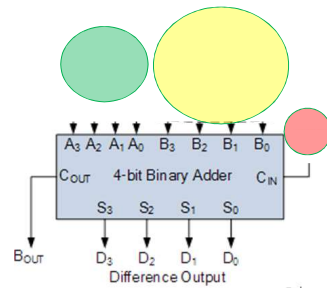
Figure 7-18 8-bit binary adder using two 74HC283 ICs.

32

N-bit Subtractor

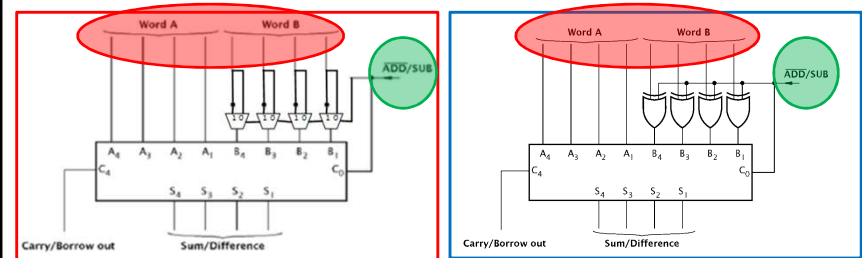
- The subtractor can be implemented using an adder

- $A - B = A + (-B)$
- $-B = B' + 1$



33

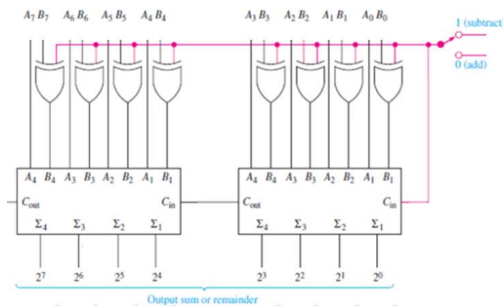
N-bit Adder/Subtractor



34

Adder/Subtractor Cascading

- 8-bit adder/subtractor by cascading 4-bit adders



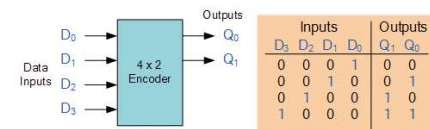
35

Encoders

- Simple Encoder

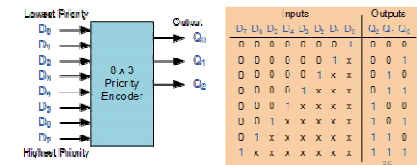
- Challenges

- Multiple input activation
- No input activation



- Priority Encoder

- Multiple input activation solution

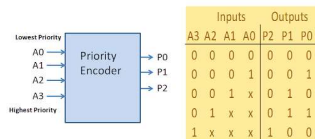


A. K. K. K.

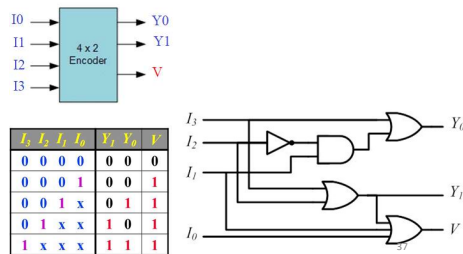
36

Encoders

- Extra output
 - No input activation solution
- Type 1

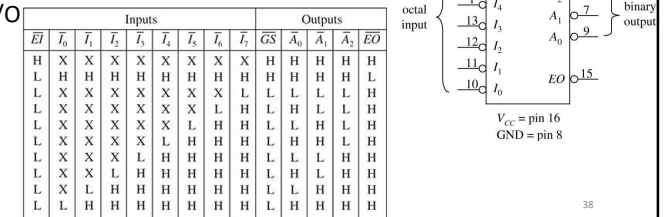


Type 2



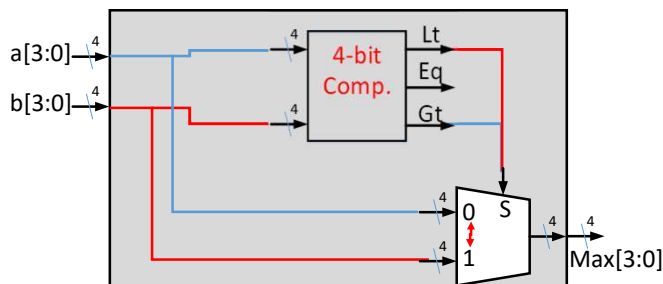
74148 8-to-3 Priority Encoder

- GS distinguishes between “no active input” and “I0” activation when EI is active
- EI and EO cascade signals
 - EO is active when EI is active and no input is active
- Active low I/O



Some Design Examples: Max Circuit

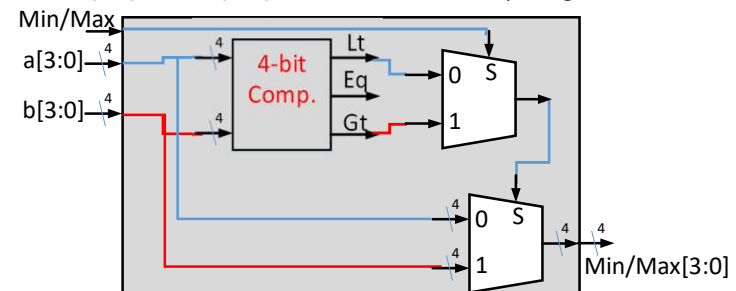
- Design a circuit with two 4-bit inputs a and b, whose 4-bit output is **MAX(a, b)**



39

Some Design Examples: Min/Max Circuit

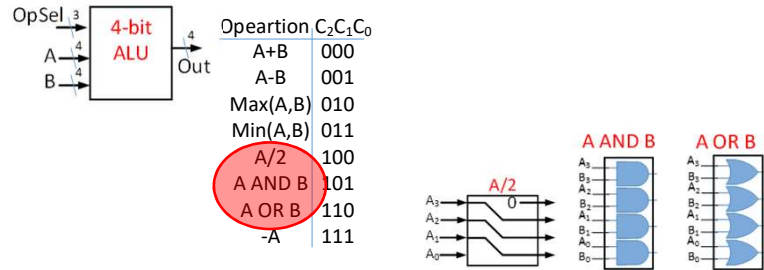
- Design a circuit with two 4-bit inputs a and b, that outputs **either MIN(a,b) or MAX(a, b)** based on MIN/MAX input signal



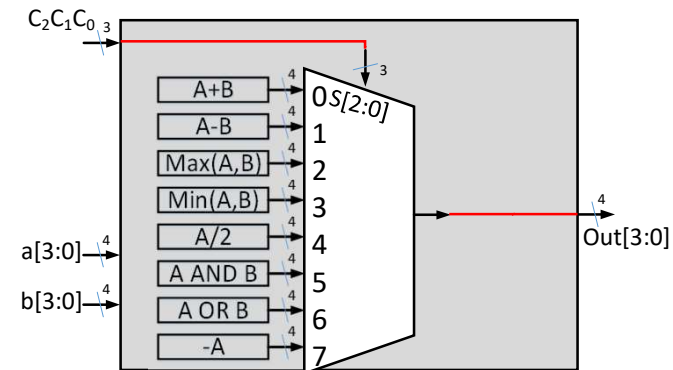
40

Some Design Examples: 8-Operation 4-Bit ALU

- Implement a 4-bit ALU to perform eight operations
 - A+B, A-B, Max(A,B), Min(A,B), A/2, A AND B, A OR B, -A

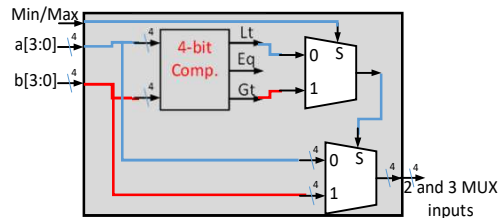


Some Design Examples: 8-Operation 4-Bit ALU



Some Design Examples: 8-Operation 4-Bit ALU

- Max(A,B) and Min(A,B) have very similar structures
- Their results are not concurrently necessary
- A Max/Min block might be exploited instead of both



Some Design Examples: 8-Operation 4-Bit ALU

- A+B, A-B, and -A blocks need a 4-bit adder
- The results of these blocks are not concurrently necessary
- They can be merged in a single block with three operations

