DIGITAL TECHNICS I

Dr. Bálint Pődör

Óbuda University, Microelectronics and Technology Institute

11. LECTURE: FUNCTIONAL BUILDING BLOCKS II

1st year BSc course 1st (Autumn) term 2018/2019

1

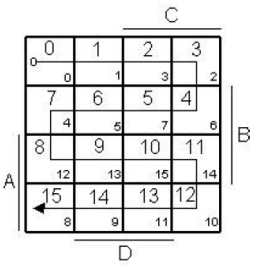
FUNCTIONAL BUILDING BLOCKS II

1. Code conversions: binary/Gray, binary/BCD
2. Multiplexers and demultiplexers
3. Comapartors

2

4-BIT GRAY CODE ON THE KARNAUGH MAP

mi	I	A	B	C	D
0	0	0	0	0	0
1	1	0	0	0	1
3	2	0	0	1	1
2	3	0	0	1	0
6	4	0	1	1	0
7	5	0	1	1	1
5	6	0	1	0	1
4	7	0	1	0	0
12	8	1	1	0	0
13	9	1	1	0	1
15	10	1	1	1	1
14	11	1	1	1	0
10	12	1	0	1	0
11	13	1	0	1	1
9	14	1	0	0	1
8	15	1	0	0	0



Construction rules of Gray code on 4-bits

3

BINARY/GRAY CONVERSION

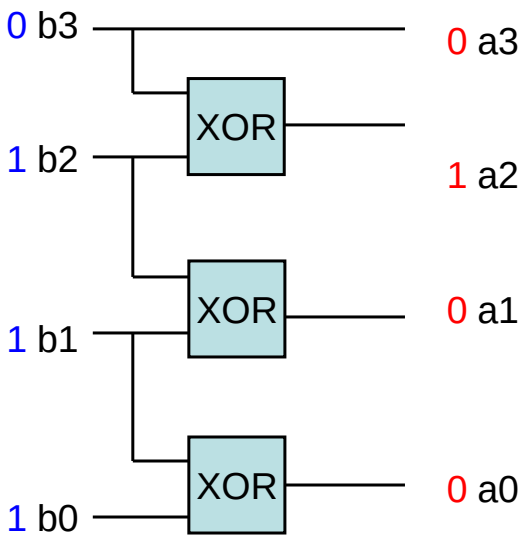
Binary 1 0 0 1 1 0 0 1

Gray 1 1 0 1 0 1 0 1

Simple algorithm for binary-to-Gray code conversion

4

BINARY/GRAY CODE COVERTER

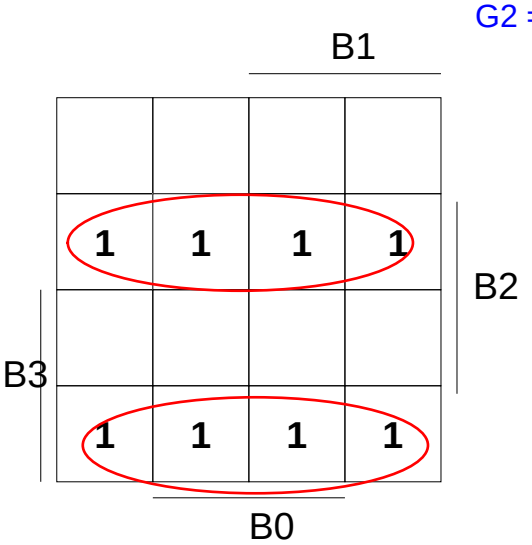


5

BIN/GRAY CONVERSION: G2

$G2 = \overline{B3}\,B2 + B3\,\overline{B2}$

$= B3 \oplus B2$



I	G3	G2	G1	G0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

6

BINARY/GRAY AND GRAY/BINARY CONVERSION ALGORITHMS

Binary: $b_3b_2b_1b_0$ Gray: $g_3g_2g_1g_0$

Binary $\rightarrow$ Gray: $g_i = b_{i+1} \oplus b_i$

Gray $\rightarrow$ Binary: $b_i = b_{i+1} \oplus g_i$

7

MULTIPLEXING

A multiplexer (MUX) is a device which selects one of many inputs to a single output. The selection is done by using an input address. Hence, a MUX can take many data bits and put them, one at a time, on a single output data line in a particular sequence. This is an example of transforming parallel data to serial data.

A demultiplexer (DEMUX) performs the inverse operation, taking one input and sending it to one of many possible outputs. Again the output line is selected using an address.

8

MULTIPLEXING (CONT.)

A MUX-DEMUX pair can be used to convert data to serial form for transmission, thus reducing the number of required transmission lines. The address bits are shared by the MUX and DEMUX at each end. If n data bits are to be transmitted, then after multiplexing, the number of separate lines required is $\log^2 n + 1$, compared to n without the conversion to serial. Hence for large n the saving can be substantial.

Multiplexers consist of two functionally separate components, a decoder and some switches or gates. The decoder interprets the input address to select a single data bit.

9

MULTIPLEXERS AND DEMULTIPLEXERS

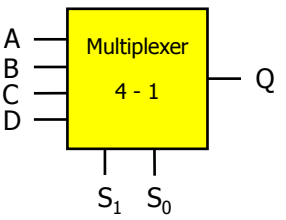
A multiplexer or mux is a device that selects one of many data-lines and ouputs that into a single line.

A demultiplexer or demux is a device that selects one of many output lines and connects the single input to the selected output line.

Sometimes the term data selector is used

MULTIPLEXER

It selects one (data)line from several input (data)lines.
2^n data inputs, one data output, n control/selector inputs.

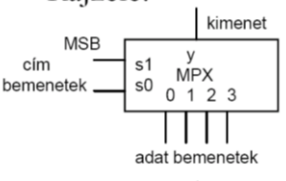


11

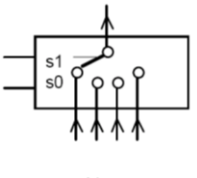
MULTIPLEXER: FUNCTIONING AND INTERNAL STRUCTURE

Multiplexer

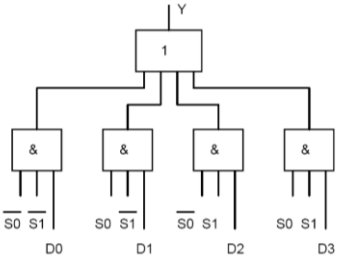
Rajzele:



Működése:



Internal structure:

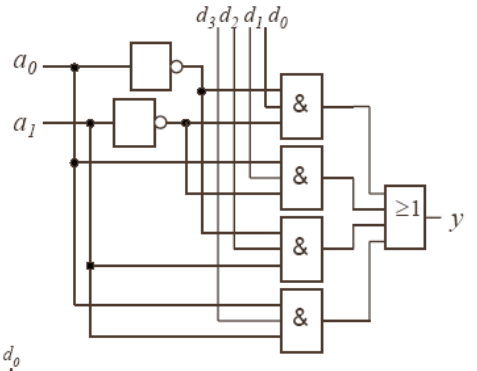


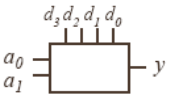
4-TO-1 LINE MULTIPLEXER

a_1	a_0	y
0	0	d_0
0	1	d_1
1	0	d_2
1	1	d_3

$$y = \overline{a_0} \cdot \overline{a_1} \cdot d_0 + a_0 \cdot \overline{a_1} \cdot d_1 + \overline{a_0} \cdot a_1 \cdot d_2 + a_0 \cdot a_1 \cdot d_3$$

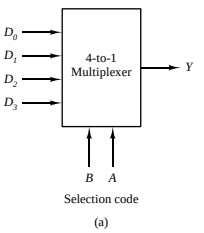
d_3	d_2	d_1	d_0	a_1	a_0	y
x	x	x	0	0	0	0
x	x	x	1	0	0	1
x	x	0	x	0	1	0
x	x	1	x	0	1	1
x	0	x	x	1	0	0
x	1	x	x	1	0	1
0	x	x	x	1	1	0
1	x	x	x	1	1	1





13

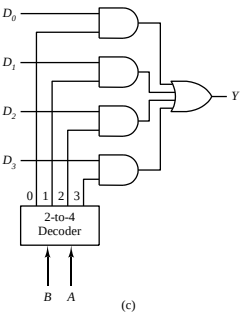
4-TO-1 MUX: INTERNAL STRUCTURE



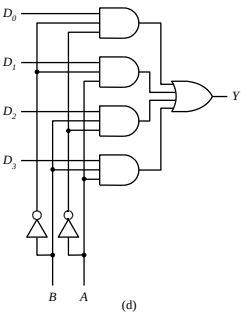
(a)

B	A	Y
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

(b)



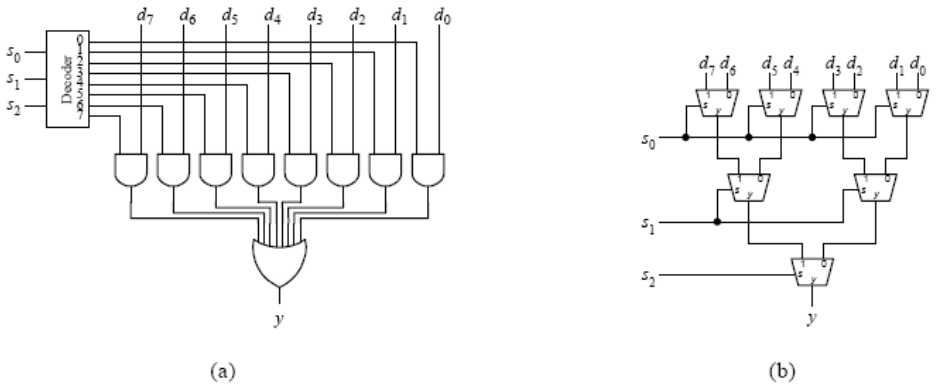
(c)



(d)

14

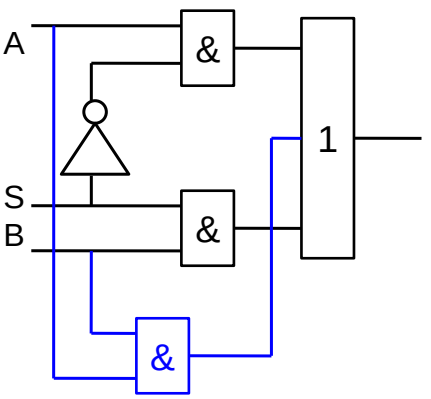
MUX IMPLEMENTATIONS



An 8-to-1 multiplexer implemented using:
(a) a 3-to-8 decoder; (b) seven 2-to-1 multiplexers

15

2-TO-1 MUX: DETAILED ANALYSIS



Logic equation:

$Y = S\,B + \overline{S}\,A$

Hazard! (Critical change in S, if A=B=1)

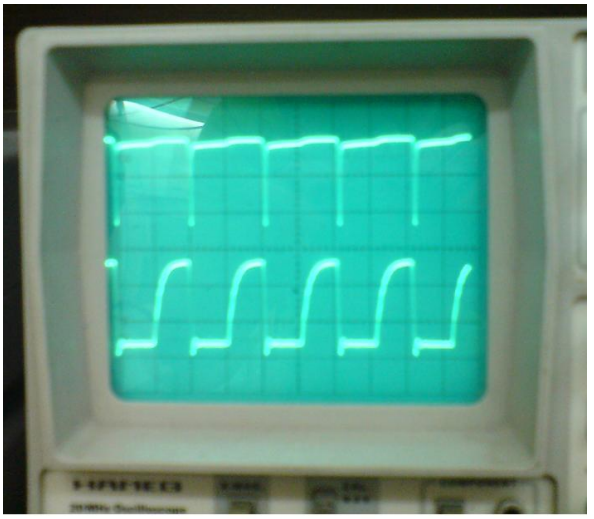
Hazard elimination:

$Y = S\,B + \overline{S}\,A + A\,B$

The network without the AB gate, when impemented with four NAND (74LS00, t_{pd}=9,5 ns) gates in a breadboard fashion exhibited the hazardous operation.

16

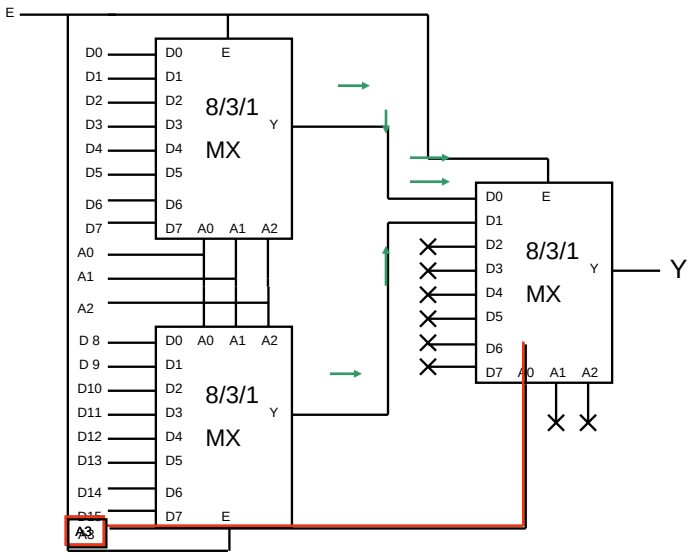
2-TO-1 MULTIPLEXER



Static hazard seen on the output of the MUX

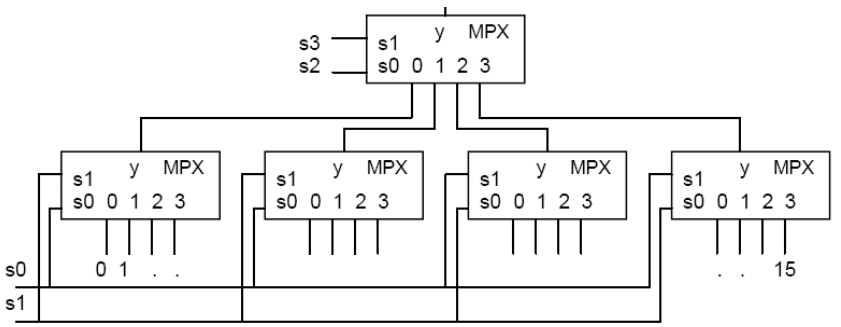
17

MULTIPLEXER EXTENSION



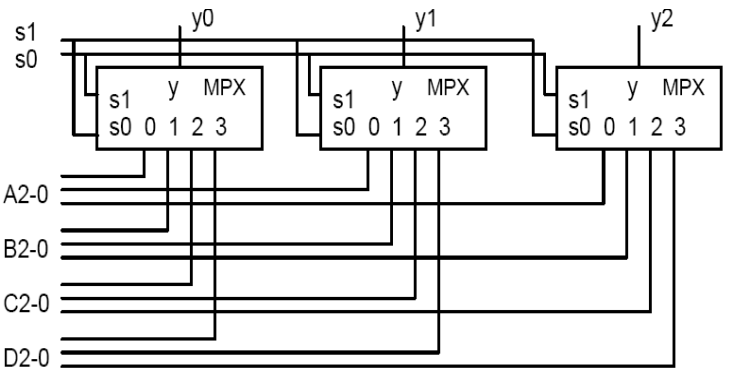
18

MULTIPLEXER: CHANNEL NUMBER EXTENSION



Larger multiplexers can be constructed by using smaller multiplexers by chaining them together. E. g. multiplexer with 16-data inputs from 4-data input units.
The number of levels can be increased in principle at free will, however each added level increases the time delay of the whole circuit.

MULTIPLEXER: CHANNEL WIDTH EXTENSION



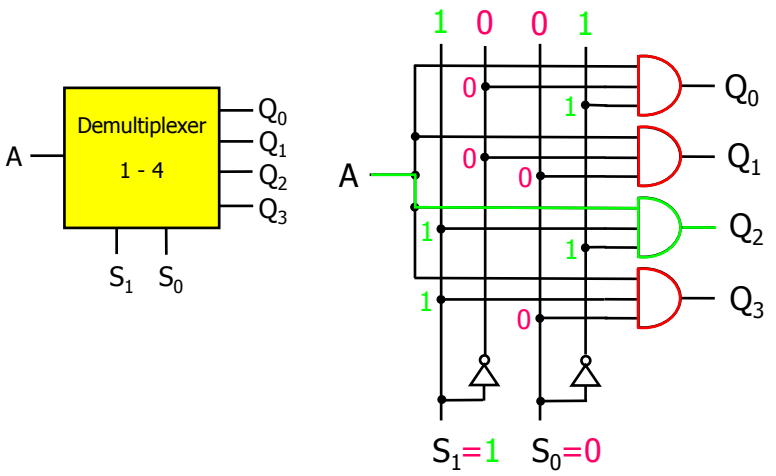
Multiplexing four 3-bit wide channels (buses) using three 4-bit multiplexers.

DEMULTIPLEXER

- One (data)input line to be connected to any of several outputs
- One (data)input,
  2^n (data)outputs,
  n control/selector inputs.

21

DEMULTIPLEXER



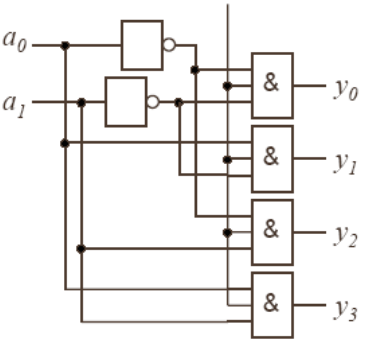
22

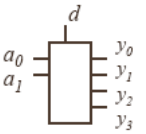
1-TO-4 LINE DEMULTIPLEXER

a_1	a_0	y_3	y_2	y_1	y_0
0	0	0	0	0	d
0	1	0	0	d	0
1	0	0	d	0	0
1	1	d	0	0	0

d	a_1	a_0	y_3	y_2	y_1	y_0
0	0	0	0	0	0	**0**
1	0	0	0	0	0	**1**
0	0	1	0	0	**0**	0
1	0	1	0	0	**1**	0
0	1	0	0	**0**	0	0
1	1	0	0	**1**	0	0
0	1	1	**0**	0	0	0
1	1	1	**1**	0	0	0

$$y_0 = \overline{a_0} \cdot \overline{a_1} \cdot d$$
$$y_1 = a_0 \cdot \overline{a_1} \cdot d$$
$$y_2 = \overline{a_0} \cdot a_1 \cdot d$$
$$y_3 = a_0 \cdot a_1 \cdot d$$



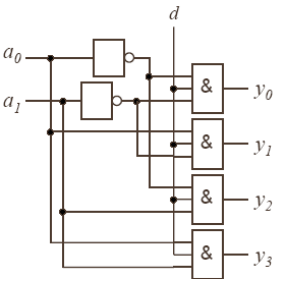


23

COMPARISON OF DECODER AND DEMULTIPLEXER

$N=4 \rightarrow n=2$

$$y_0 = \overline{a_0} \cdot \overline{a_1} \cdot d$$
$$y_1 = a_0 \cdot \overline{a_1} \cdot d$$
$$y_2 = \overline{a_0} \cdot a_1 \cdot d$$
$$y_3 = a_0 \cdot a_1 \cdot d$$

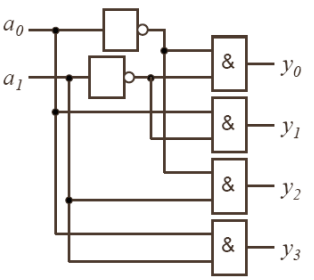


pl. **4**-ből **1** dekódoló

$N=4 \rightarrow n=2$

a_1	a_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$$y_0 = \overline{a_0} \cdot \overline{a_1}$$
$$y_1 = a_0 \cdot \overline{a_1}$$
$$y_2 = \overline{a_0} \cdot a_1$$
$$y_3 = a_0 \cdot a_1$$



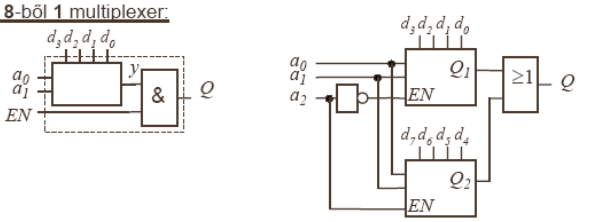
24

MUX AND DEMUX EXTENSION

Multiplexer bővítés

2 db **N**-ből **1** multiplexerből összeállítható egy **2N**-ből **1** multiplexer:

pl. **8**-ból **1** multiplexer:

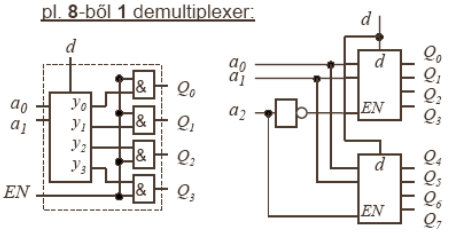


a_2	a_1	a_0	Q
0	0	0	d_0
0	0	1	d_1
0	1	0	d_2
0	1	1	d_3
1	0	0	d_4
1	0	1	d_5
1	1	0	d_6
1	1	1	d_7

Demultiplexer bővítés

2 db **N**-ből **1** demultiplexerből összeállítható egy **2N**-ből **1** demultiplexer:

pl. **8**-ból **1** demultiplexer:



a_2	a_1	a_0	Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	0	0	0	d
0	0	1	0	0	0	0	0	0	d	0
0	1	0	0	0	0	0	0	d	0	0
0	1	1	0	0	0	0	d	0	0	0
1	0	0	0	0	0	d	0	0	0	0
1	0	1	0	0	d	0	0	0	0	0
1	1	0	0	d	0	0	0	0	0	0
1	1	1	d	0	0	0	0	0	0	0

5

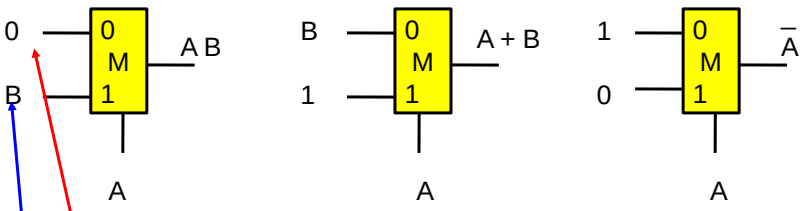
MULTIPLEXER AS AN UNIVERSAL COMBINATIONAL CIRCUIT

From the point of view of output(s) the multiplexer can be considered as a one level combinational circuit.

Its characteristics is the fast response time.

For the selected input the time delay corresponds to the unit gate delay.

BASIC GATES WITH MUX



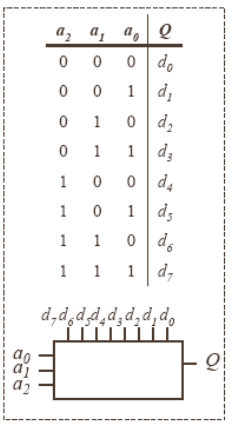
Background: expansion theorem

$F(A,B) = A\,F(1,B) + \overline{A}\,F(0,B)$

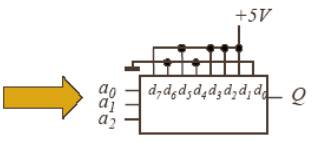
Application to $F(A,B) = A\,B$
Expanded form:

$A\,B + \overline{A}\,0 = A\,B$ *QED*

MULTIPLEXER AS AN UNIVERSAL COMBINATIONAL NETWORK



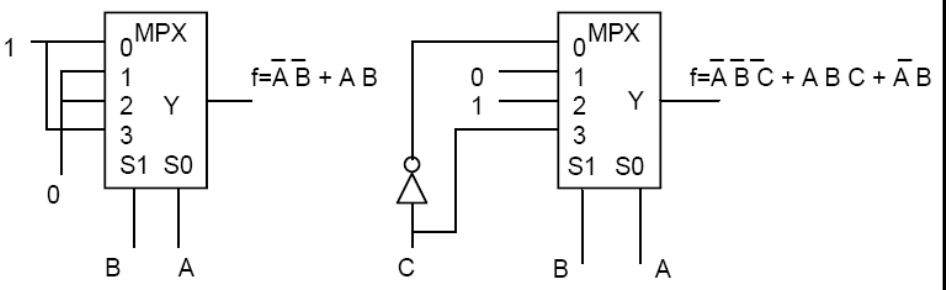
a_2	a_1	a_0	Q
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



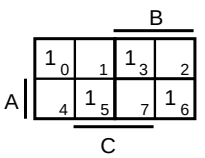
$Q = \Sigma^3(1,2,3,5,7)$

A multiplexer can be used to implement the truth table. It generates directly the minterms. E.g. using a 8-to-1 MUX any 3-variable function can be realized using one MSI package. [28]

MULTIPLEXER AS AN UNIVERSAL COMBINATIONAL NETWORK



IMPLEMENTATION OF LOGIC FUNCTIONS WITH MUX

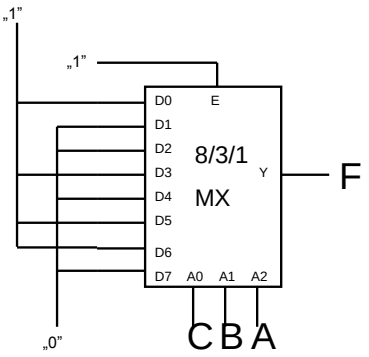


$$F(A,B,C) = \sum^{3}(0,3,5,6)$$

$$F(A,B,C) = \bar{A}\cdot\bar{B}\cdot\bar{C} + \bar{A}\cdot B\cdot C + A\cdot\bar{B}\cdot C + A\cdot B\cdot\bar{C}$$

Using gates: min three modules/packges

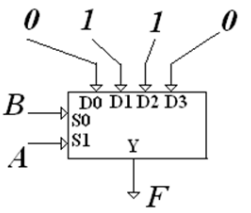
Using multiplexer: one module/package



MULTIPLEXER BASED IMPELEMTATION OF XOR FUNCTION

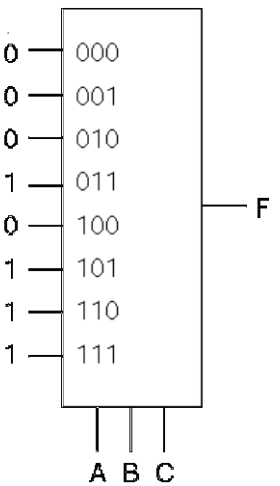
$$F = A\overline{B} + \overline{A}B$$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



Implementing the Majority Function with an 8-1 Mux

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Principle: Use the mux select to pick out the selected minterms of the function.

32

MORE EFFICIENCY

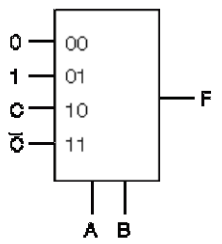
However, there is a better technique available for doing the same. In this, a 2^n-to-1 MUX can be used to implement a Boolean function with n + 1 variables. The procedure is as follows. Out of n + 1 variables, n are connected to the n selection lines of the 2^n-to-1 multiplexer. The left-over variable is used with the input lines. Various input lines are tied to one of the following: '0', '1', the left-over variable and the complement of the left-over variable. Which line is given what logic status can be easily determined with the help of a simple procedure.

33

MORE EFFICIENCY: USING 4-TO-1 MUX TO IMPLEMENT A 3-VARIABLE FUNCTION

Function to be implemented: F = $\Sigma^4(2,3,5,6)$

A	B	C	F	
0	0	0	0	0
0	0	1	0	
0	1	0	1	1
0	1	1	1	
1	0	0	0	C
1	0	1	1	
1	1	0	1	$\overline{C}$
1	1	1	0	

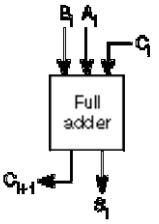


Principle: Use the A and B inputs to select a pair of minterms. The value applied to the MUX input is selected from {0, 1, C, C'} to pick the desired behavior of the minterm pair.

34

Example: Using Multiplexers to Implement an Adder

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

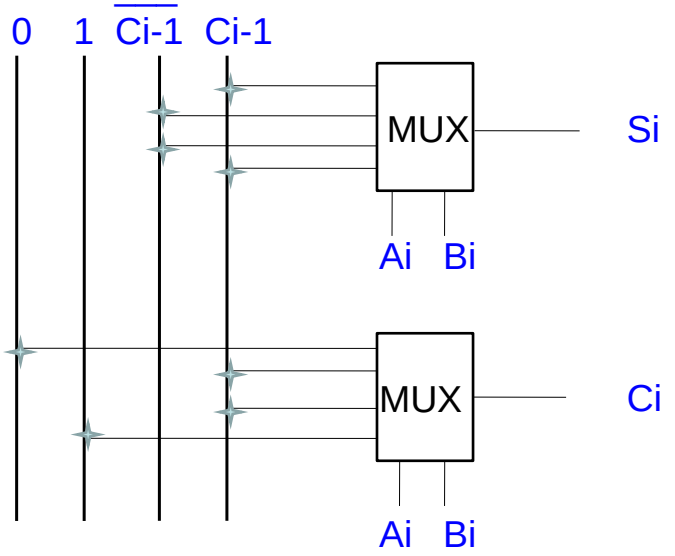


A_i	B_i	S_i	C_{i+1}
0	0	C_i	0
0	1	$\overline{C}_i$	C_i
1	0	$\overline{C}_i$	C_i
1	0	C_i	1

Rearrange truth table:

Use A_i, B_i to select MUX output, connect C_i and C_i' to MUX data inputs.
Implement with two 4-to-1 multiplexers and one inverter (to generate C_i')

FULL ADDER: 4/2/1 MUX IMPLEMENTATION



36

CASE STUDY: LOGICAL FUNCTIONAL UNIT

- Multi-purpose Function Block
  - 3 control inputs to specify operation to perform on operands
  - 2 data inputs for operands
  - 1 output of the same bit-width as operands

C0	C1	C2	Function	Comments
0	0	0	1	always 1
0	0	1	A + B	logical OR
0	1	0	(A • B)'	logical NAND
0	1	1	A XOR B	logical XOR
1	0	0	A XNOR B	logical XNOR
1	0	1	A • B	logical AND
1	1	0	(A + B)'	logical NOR
1	1	1	0	always 0

3 control inputs: C0, C1, C2
2 data inputs: A, B
1 output: F

IMPLEMENTATION WITH LOGIC GATES

C0	C1	C2	A	B	F	
0	0	0	0	0	1	0
0	0	0	0	1	1	1
0	0	0	1	0	1	2
0	0	0	1	1	1	3
0	0	1	0	0	0	
0	0	1	0	1	1	5
0	0	1	1	0	1	6
0	0	1	1	1	1	7
0	1	0	0	0	1	8
0	1	0	0	1	1	9
0	1	0	1	0	1	10
0	1	0	1	1	0	
0	1	1	0	0	0	
0	1	1	0	1	1	13
0	1	1	1	0	1	14
0	1	1	1	1	0	
1	0	0	0	0	1	16
1	0	0	0	1	0	
1	0	0	1	0	0	
1	0	0	1	1	1	19
1	0	1	0	0	0	
1	0	1	0	1	0	
1	0	1	1	0	0	
1	0	1	1	1	1	23
1	1	0	0	0	1	24
1	1	0	0	1	0	
1	1	0	1	0	0	
1	1	0	1	1	0	
1	1	1	0	0	0	
1	1	1	0	1	0	
1	1	1	1	0	0	
1	1	1	1	1	0	

$F = \Sigma^5(0\text{-}3,5\text{-}10,13,14,16,19,23,24)$

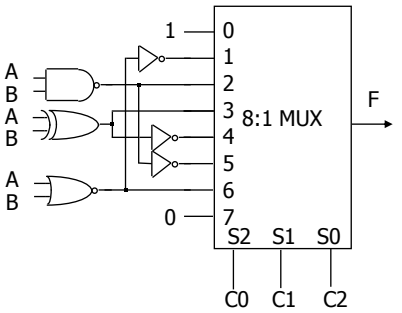
Minimization on 5 variable Karnaugh map:

four 4-cubes

IMPLEMENTATION WITH MULTIPLEXER

C0	C1	C2	A	B	F
0	0	0	0	0	1
0	0	0	0	1	1
0	0	0	1	0	1
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	1
0	1	0	0	1	1
0	1	0	1	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	0	1	0
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	0
1	0	1	1	0	0
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	0	1	0
1	1	0	1	0	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	1	0	1	0
1	1	1	1	0	0
1	1	1	1	1	0

choose implementation technology
5-variable K-map to discrete gates
multiplexer implementation
the target operations are pair wise inverse of each other



COMPARATOR

A digital comparator compares two numbers in binary form and generates a one or zero at its output depending on whether they are the same or not.

Comparators can be used in a central processing unit (CPU) or microcontroller in branching software.

A comparator can be simulated by subtracting the two values (A, B) in question and checking if the result is zero.

COMAPRATORS

Comparing two numbers three outcomes are possible

$X < Y$

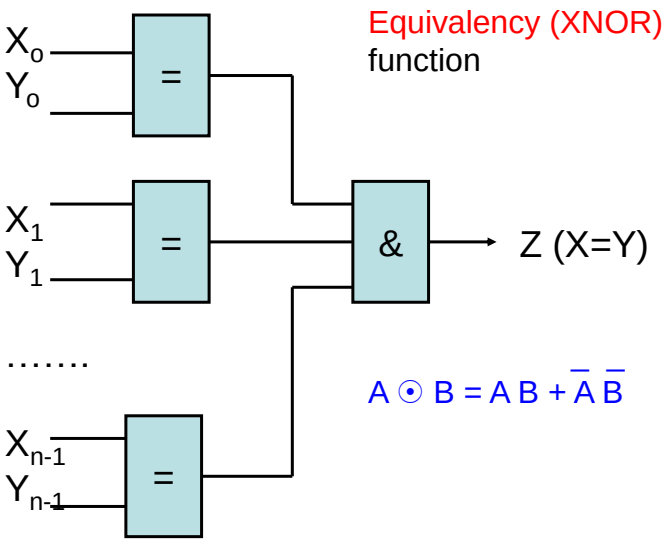
$X = Y$

$X > Y$

Two binary numbers are equal, if all their bits are equal. In this case the output of the comparator is 1, otherwise is 0.

41

N-BIT COMPARATOR



42

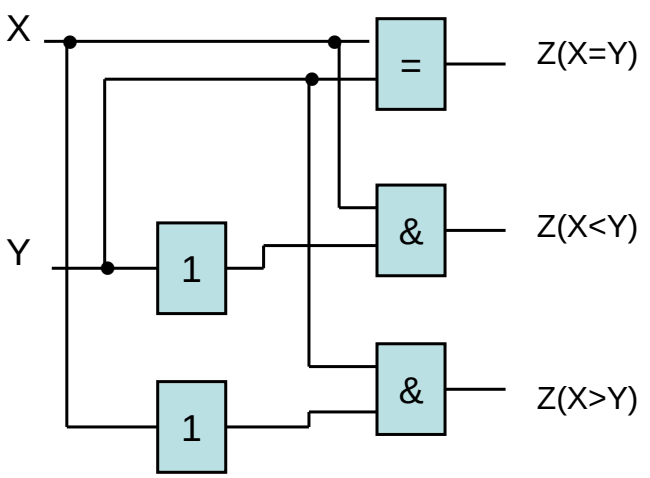
MAGNITUDE COMPARATOR (1-BIT)

X	Y	Z(X>Y)	Z(X=Y)	Z(X<Y)
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$Z(X>Y) = X \overline{Y}$ $Z(X=Y) = X Y + \overline{X}\,\overline{Y}$ $Z(X<Y) = \overline{X} Y$

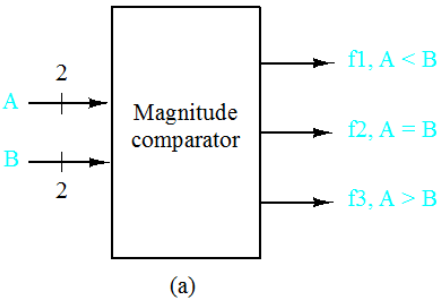
43

1-BIT MAGNITUDE COMPARATOR



44

2-BIT MAGNITUDE COMPARATOR

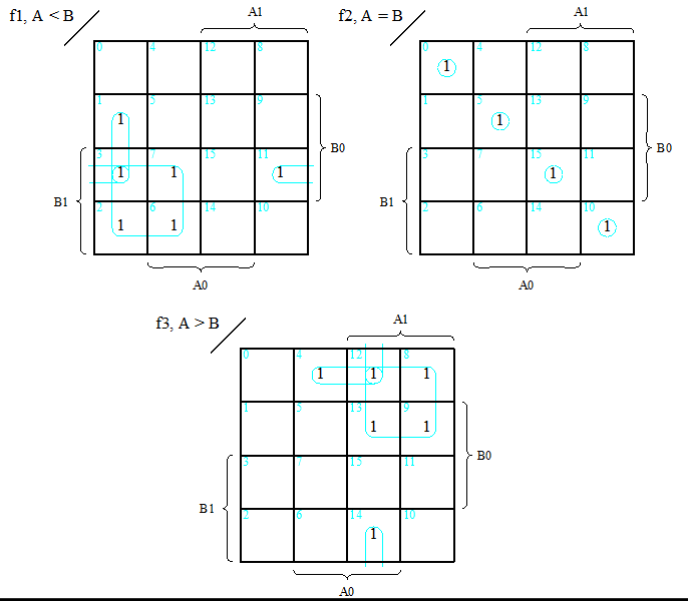


(a)

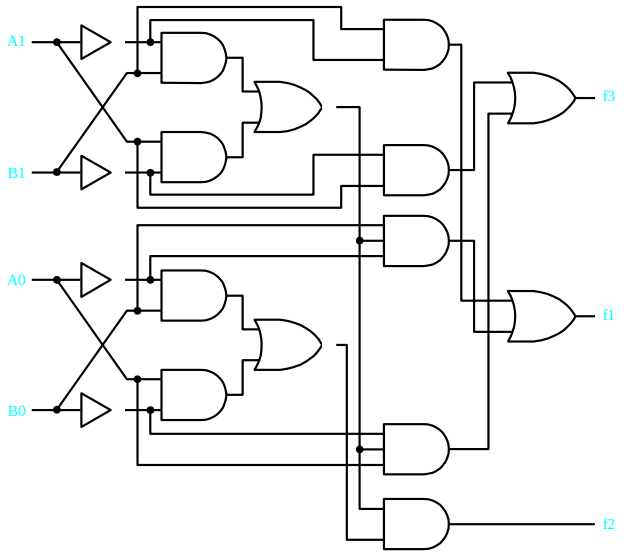
i	A1	A0	B1	B0	f1	f2	f3
0	0	0	0	0	0	1	0
1	0	0	0	1	1	0	0
2	0	0	1	0	1	0	0
3	0	0	1	1	1	0	0
4	0	1	0	0	0	0	1
5	0	1	0	1	0	1	0
6	0	1	1	0	1	0	0
7	0	1	1	1	1	0	0
8	1	0	0	0	0	0	1
9	1	0	0	1	0	0	1
10	1	0	1	0	0	1	0
11	1	0	1	1	1	0	0
12	1	1	0	0	0	0	1
13	1	1	0	1	0	0	1
14	1	1	1	0	0	0	1
15	1	1	1	1	0	1	0

(b)

2-BIT MAGNITUDE COMPARATOR



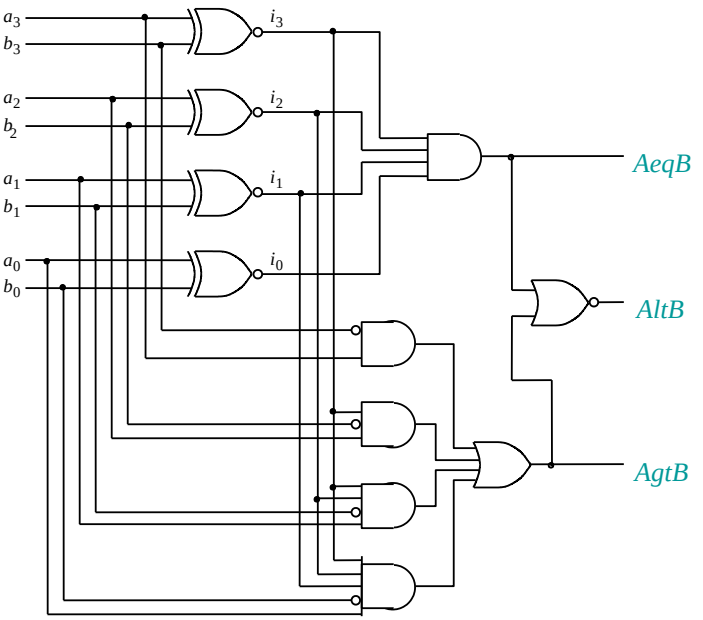
2-BIT MAGNITUDE COMPARATOR



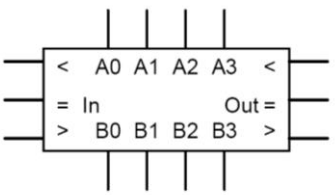
COMPARATORS

- Outputs:
  - AeqB: A is equal to B
  - AgtB: A is greater than B
  - AltB: A is less than B
- Operation?
  - Bitwise comparison beginning from the most significant bit (MSB). As the first difference occurs, the number in which the bit in question is 1, is the greater.

4-BIT COMPARATOR

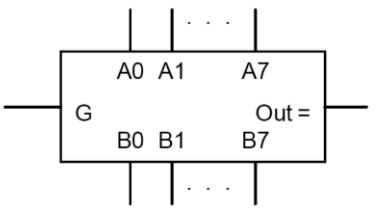


4- AND 8-BIT COMPARATORS: MSI

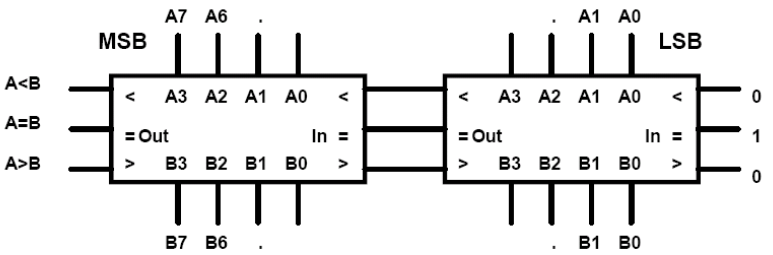


4-bit comparator three relation outputs (A>B, A=B, A<B) and three relation inputs for chaining purposes.

8-bit comparator with one relation (A=B) output, and one relation input for chaining purposes.



CHAIN CONNECTED COMPARATORS



The relation inputs of the first comparator (LSB) should be set for equality.
The delay times of the chained units will be added.

THE END

52