Digital Electronics and Microprocessor Systems (ELEC211)

Dave McIntosh and Valerio Selis

dmc@liv.ac.uk

v.selis@liv.ac.uk

Digital 2: Multiplexers, decoders and encoders

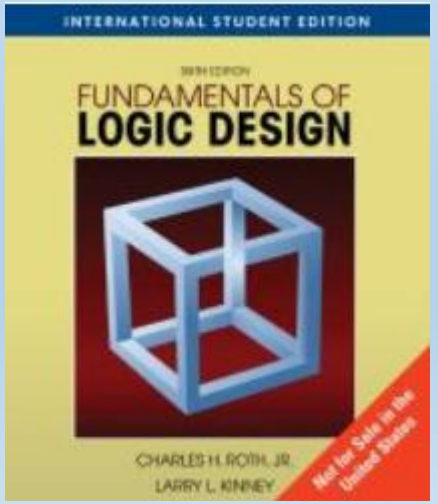
UNIVERSITY OF LIVERPOOL

Outline

https://www.youtube.com/watch?time_continue=12&v=2pLiVbrpUgE&feature=emb_logo

- Multiplexers
- Decoders
- Encoders

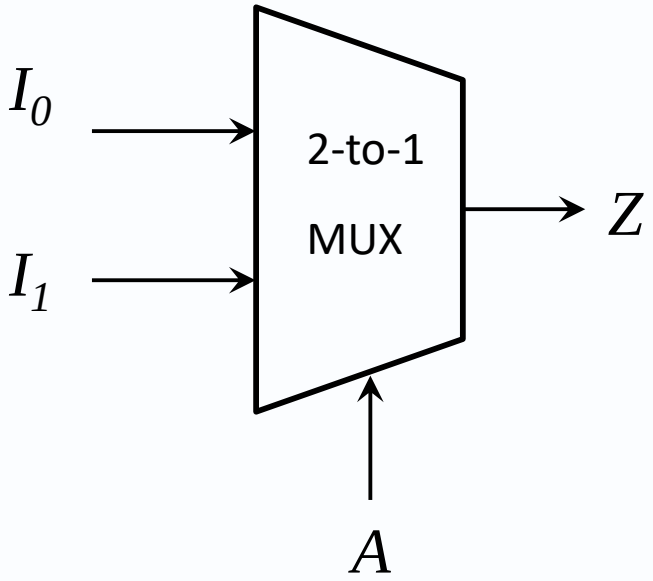
Course textbook – please borrow and use it!

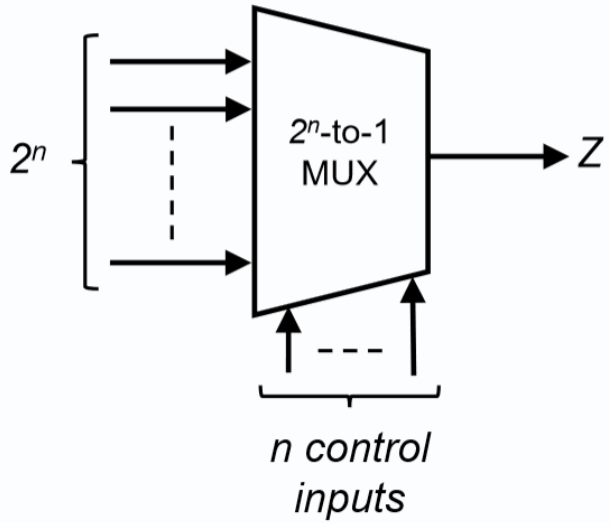


What is a multiplexer?

- ‘MUX’
- Digital switch or ‘data selector’
- Multiple data inputs
- Single output
- Switching is achieved through ‘select’ or ‘control’ inputs

- *Note: a demultiplexer does the opposite thing…*



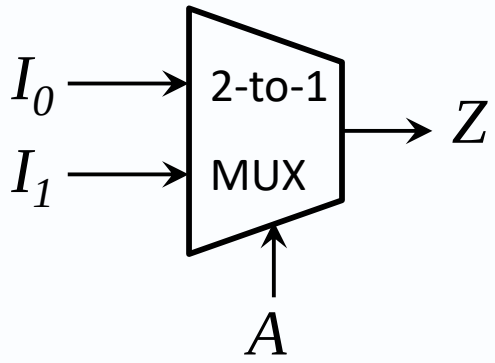


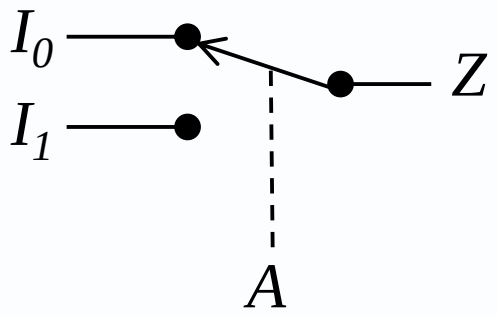
Multiplexer ('MUX')

- A multiplexer has:
  - At least two inputs.
  - One or more control inputs.
  - One output.

- The control inputs are used to select one of the data inputs and connect it to the output.
  - $A=0 \Rightarrow Z=I_0$
  - $A=1 \Rightarrow Z=I_1$
  - $Z=A'I_0+AI_1$

A	Z
0	I_0
1	I_1

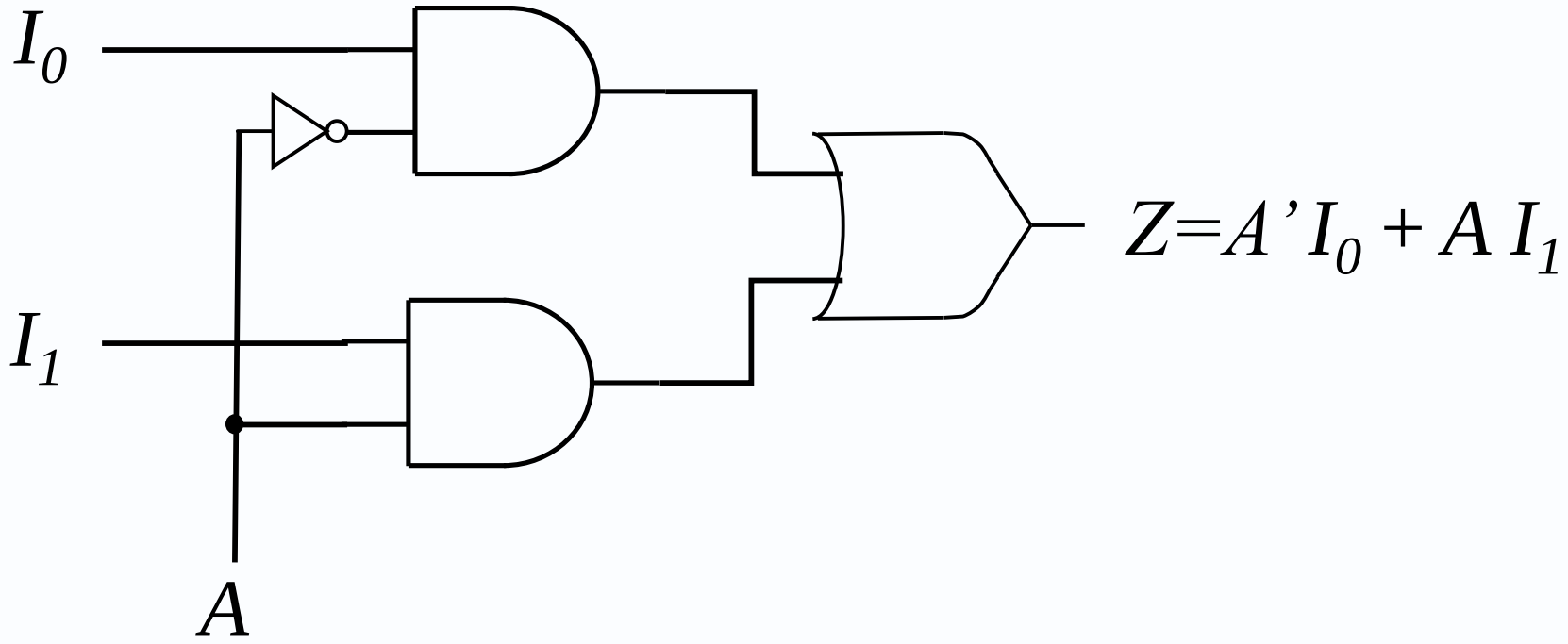




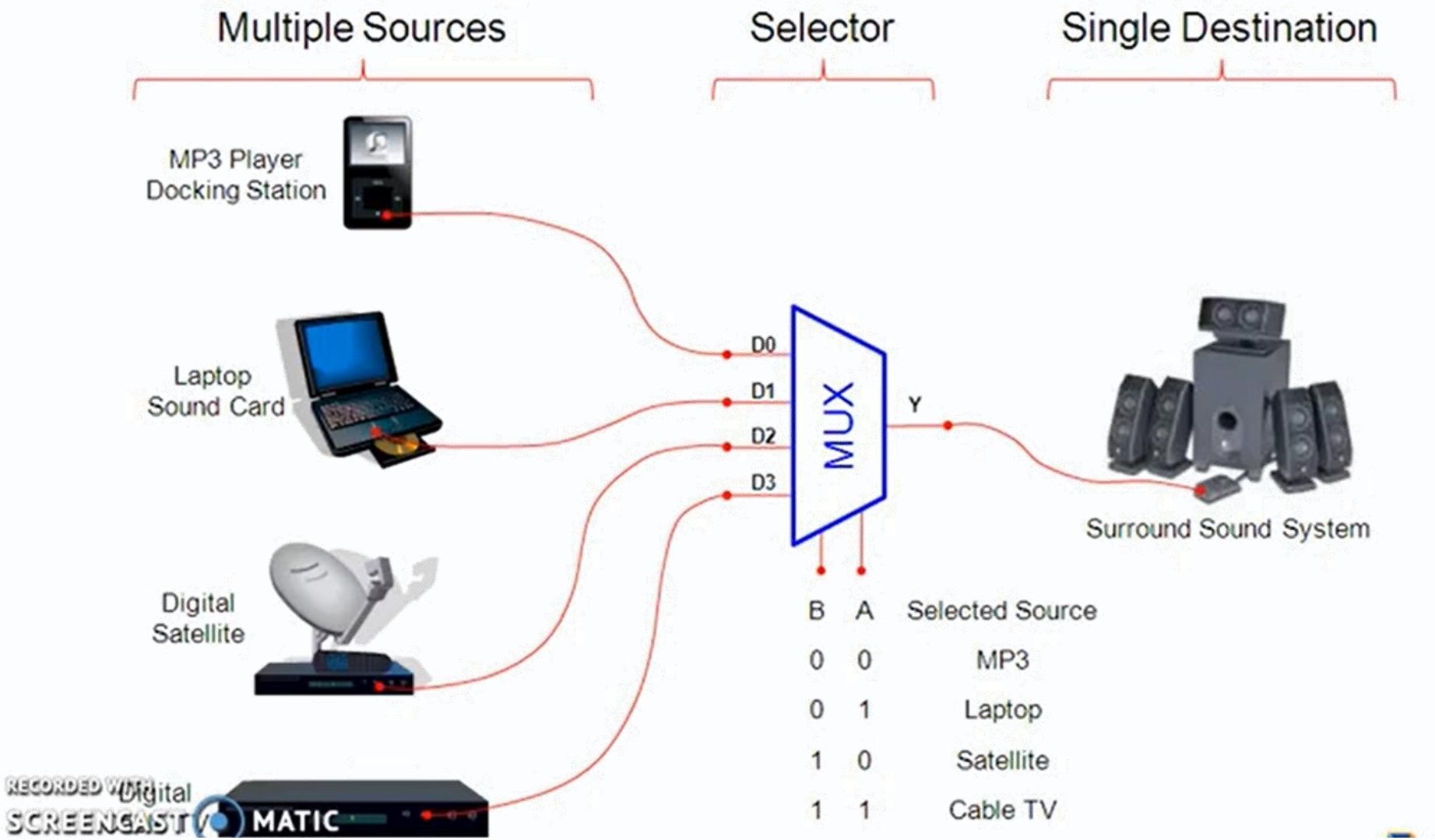
$$Z = \overline{A}I_0 + AI_1$$

Implementing it…

- Logic circuit for a 2-1 MUX

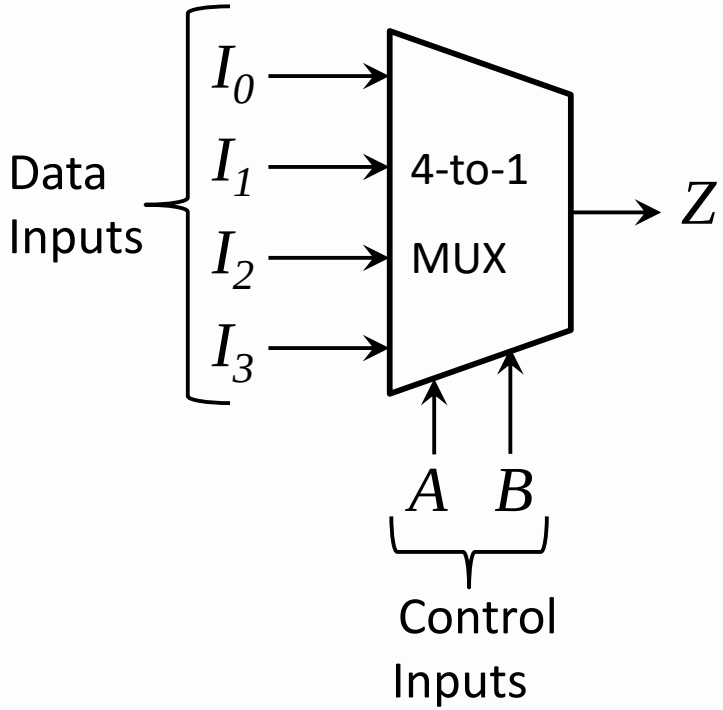


Possible application of a MUX

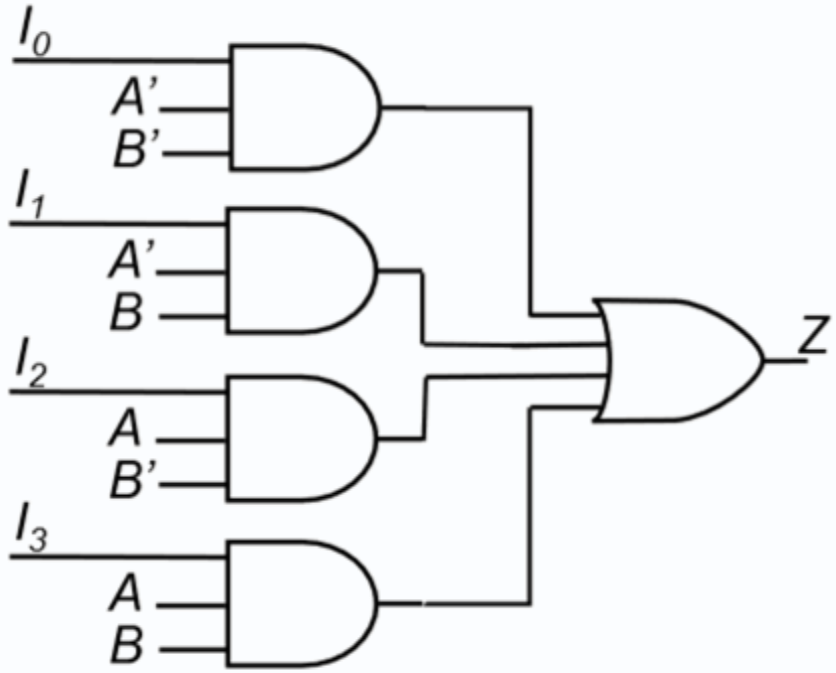


(Still from https://www.youtube.com/watch?v=7HcWmuJQcks)

MUX (4-to-1)



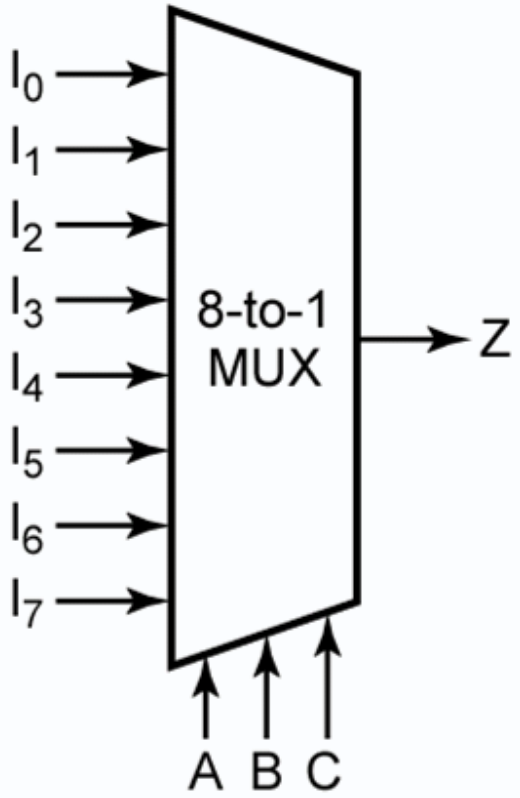
A	B	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



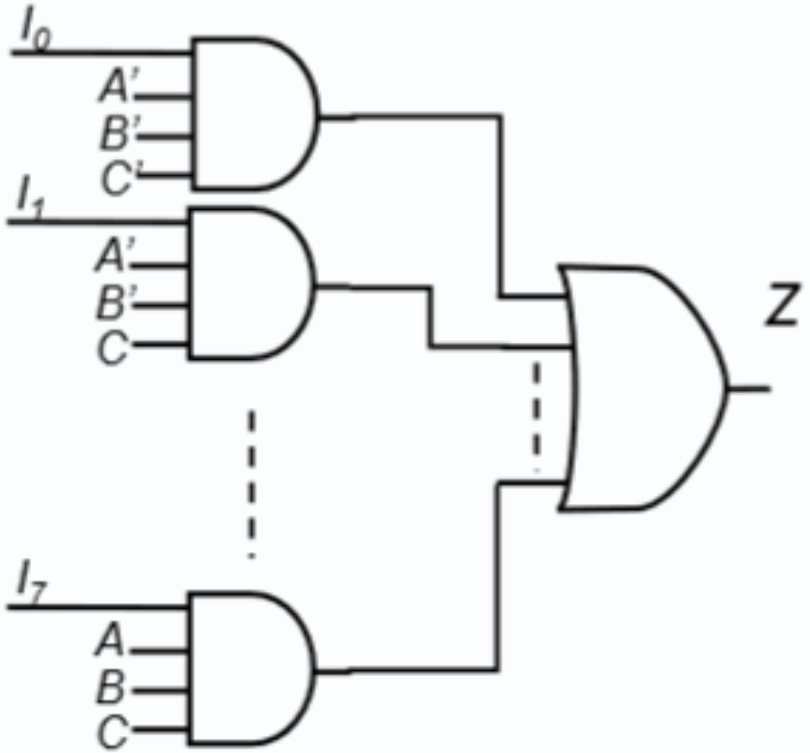
$$Z = \overline{A}\,\overline{B}I_0 + \overline{A}BI_1 + A\overline{B}I_2 + ABI_3$$

UNIVERSITY OF LIVERPOOL

MUX (8-to-1)

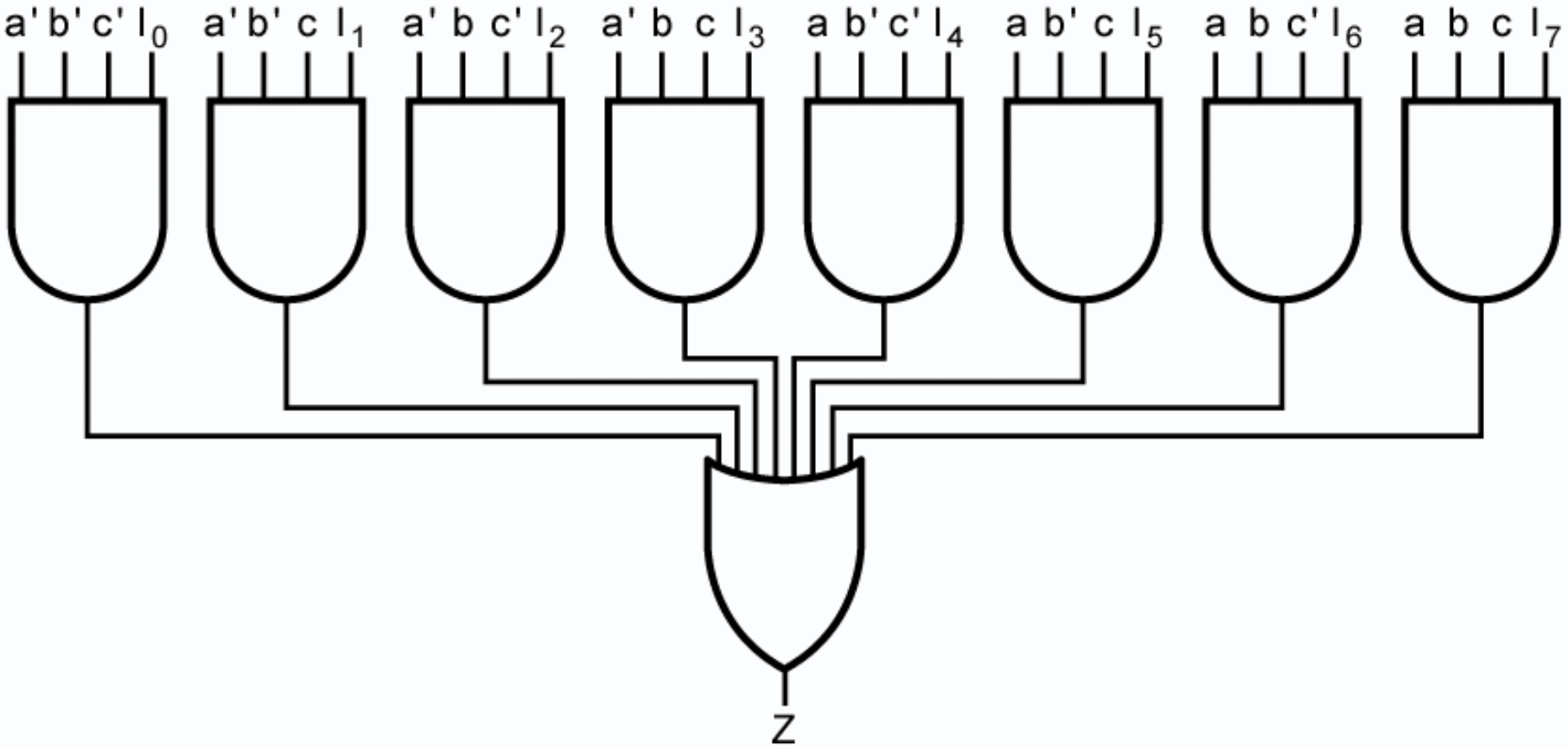


A	B	C	Z
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$

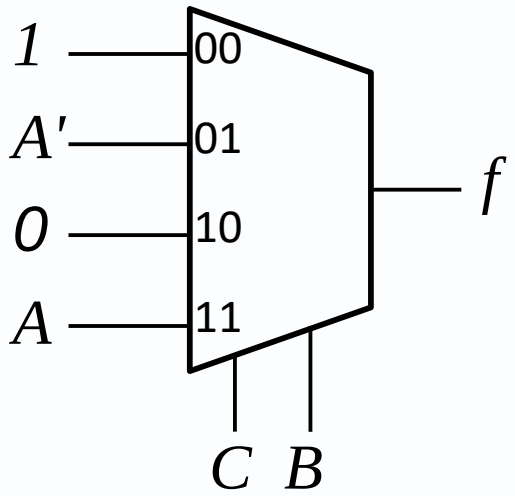
Internal logic for the 8-1 MUX



$$Z = a'b'c'I_0 + a'b'cI_1 + a'bc'I_2 + a'bcI_3 + ab'c'I_4 + ab'cI_5 + abc'I_6 + abcI_7$$

Question

- What function does this 4 to 1 MUX circuit implement?



Multiplexers – application to logic circuit design

Multiplexers can also be used as a simple method for designing combinational logic circuits e.g.

$$f = B'A + CA'$$

$$= (C' + C)B'A + C(B' + B)A'$$

$$= C'B'A + CB'A + CB'A' + CBA'$$

C	B	A		minterm
0	0	0	m_0	$C'B'A'$
0	0	1	m_1	$C'B'A$
0	1	0	m_2	$C'BA'$
0	1	1	m_3	$C'BA$
1	0	0	m_4	$CB'A'$
1	0	1	m_5	$CB'A$
1	1	0	m_6	CBA'
1	1	1	m_7	CBA

Multiplexers – application to logic circuit design

Multiplexers can also be used as a simple method for designing combinational logic circuits e.g.

$$f = B'A + CA'$$

$$= (C' + C)B'A + C(B' + B)A'$$

$$= C'B'A + CB'A + CB'A' + CBA'$$

C	B	A	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

UNIVERSITY OF LIVERPOOL

Multiplexers – application to logic circuit design

Multiplexers can also be used as a simple method for designing combinational logic circuits e.g.

$$f = B'A + CA'$$

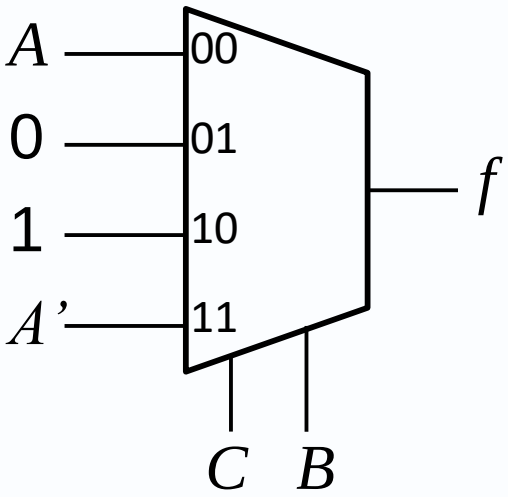
$$= (C' + C)B'A + C(B' + B)A'$$

$$= C'B'A + CB'A + CB'A' + CBA'$$

C	B	A	f	
0	0	0	0	} A
0	0	1	1	
0	1	0	0	} 0
0	1	1	0	
1	0	0	1	} 1
1	0	1	1	
1	1	0	1	} A'
1	1	1	0	

Multiplexers – application to logic circuit design

C	B	A	f	
0	0	0	0	} A
0	0	1	1	
0	1	0	0	} 0
0	1	1	0	
1	0	0	1	} 1
1	0	1	1	
1	1	0	1	} A'
1	1	1	0	



$$f = B'A + CA'$$

Question

- Design a circuit for the following expression using a 4 to 1 mux with B and C connected to the select inputs.

$$f = B'A' + BA + CB'$$

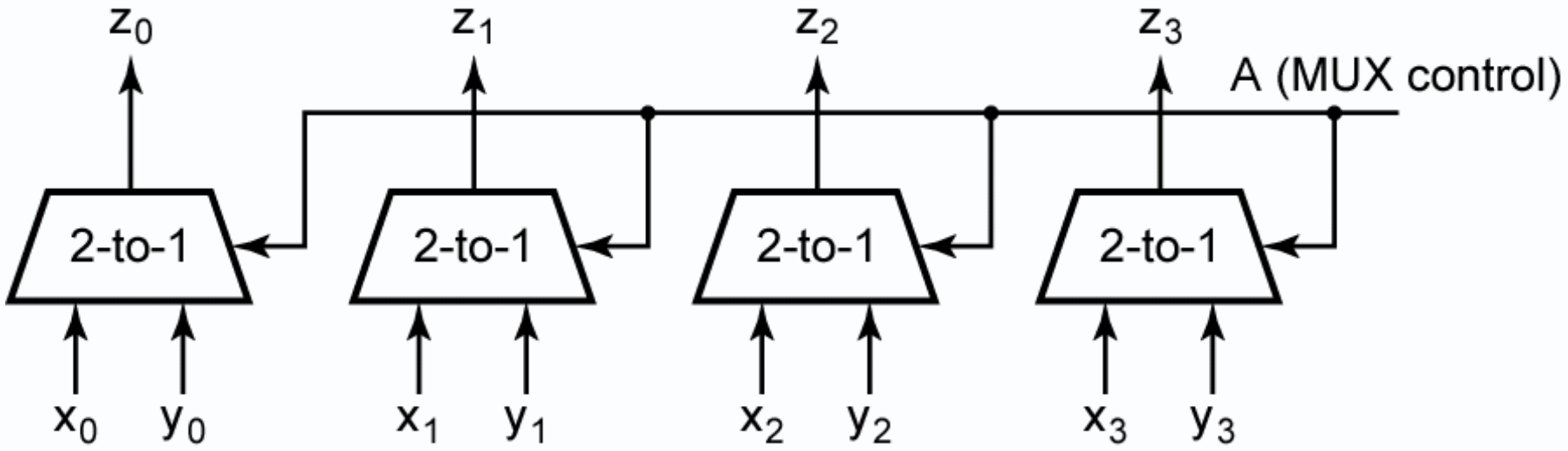
4-to-1 MUX has 4 data inputs

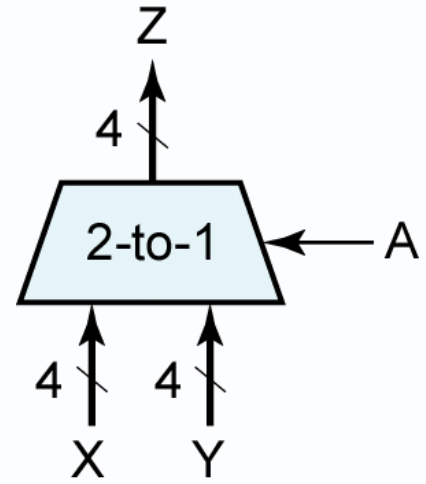
Given B and C as the 2 control / select inputs

(As there are 2^n data inputs, n = 2 makes sense)

Bus Multiplexers

- X is a 4 bit wide bus => x_3,x_2,x_1,x_0
- Y is a 4 bit wide bus => y_3,y_2,y_1,y_0
- Z is a 4 bit wide bus => z_3,z_2,z_1,z_0

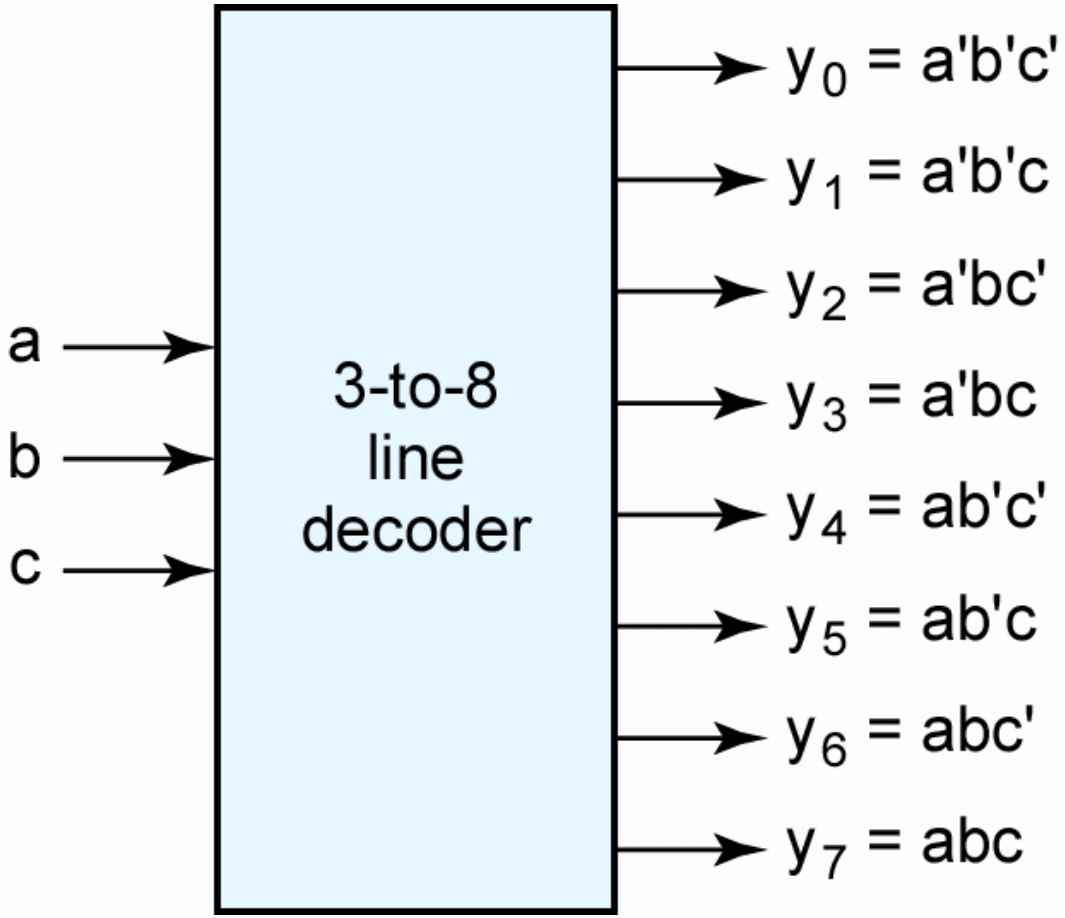




Quad multiplexer used to select data.

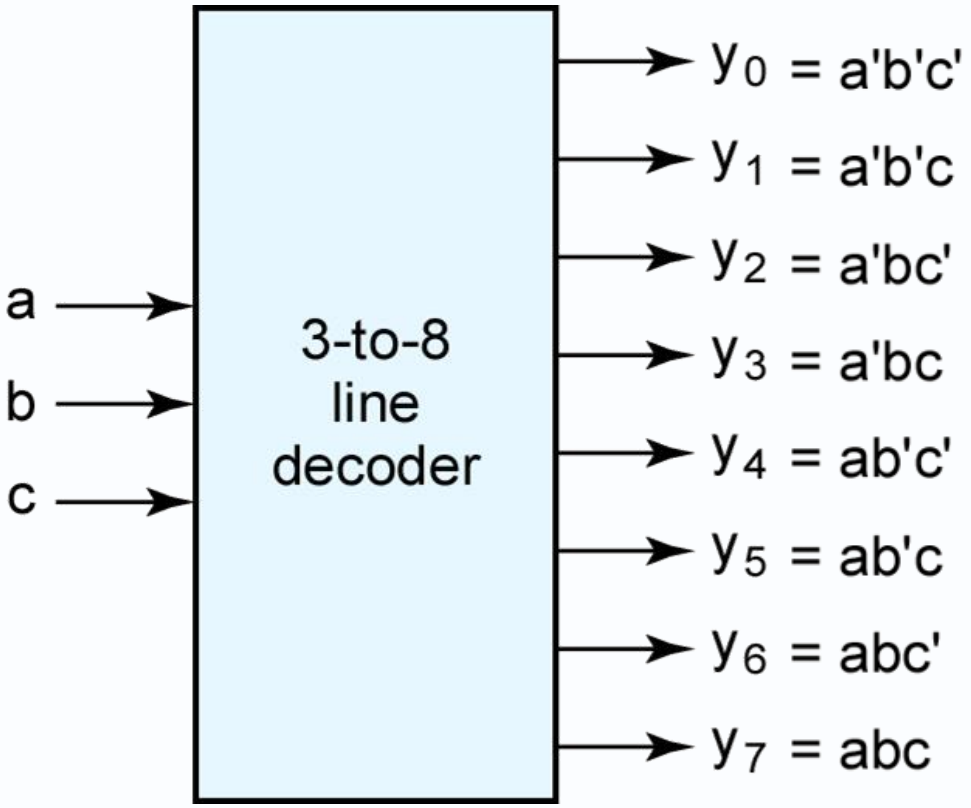
Input A selects one of two 4 bit data words.

Decoders



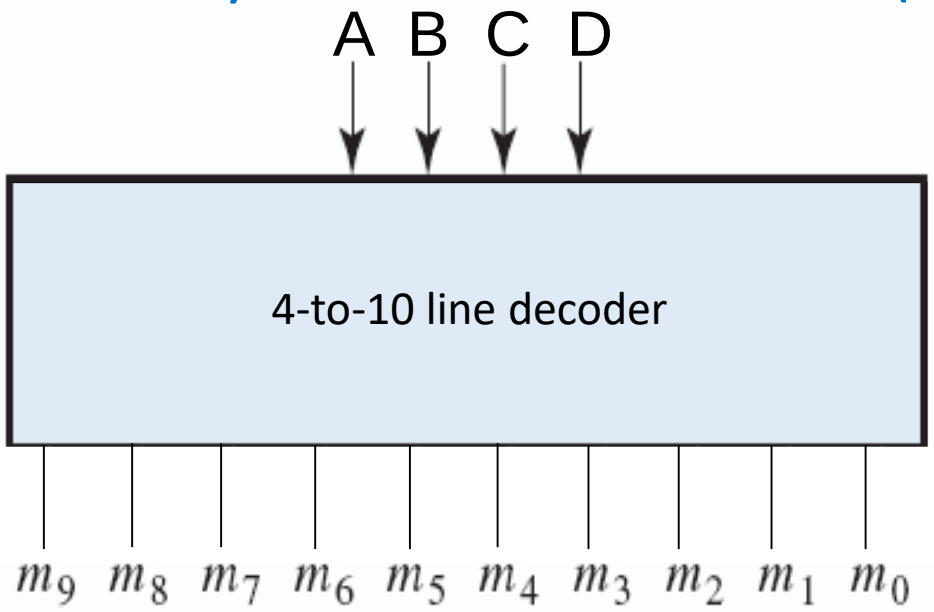
- **Exactly one** of the output lines will be 1 for each combination of the three input variables
- This decoder generates all of the **minterms** of the inputs
- A decoder converts binary information expressed by *n* inputs, to up to 2^n **unique** outputs.

3-to-8 line decoder truth table



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	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

Binary Coded Decimal (BCD) to decimal decoder – 4-to-10 line



Remember,
$m_0 = A' \cdot B' \cdot C' \cdot D'$,
$m_1 = A' \cdot B' \cdot C' \cdot D, etc.$

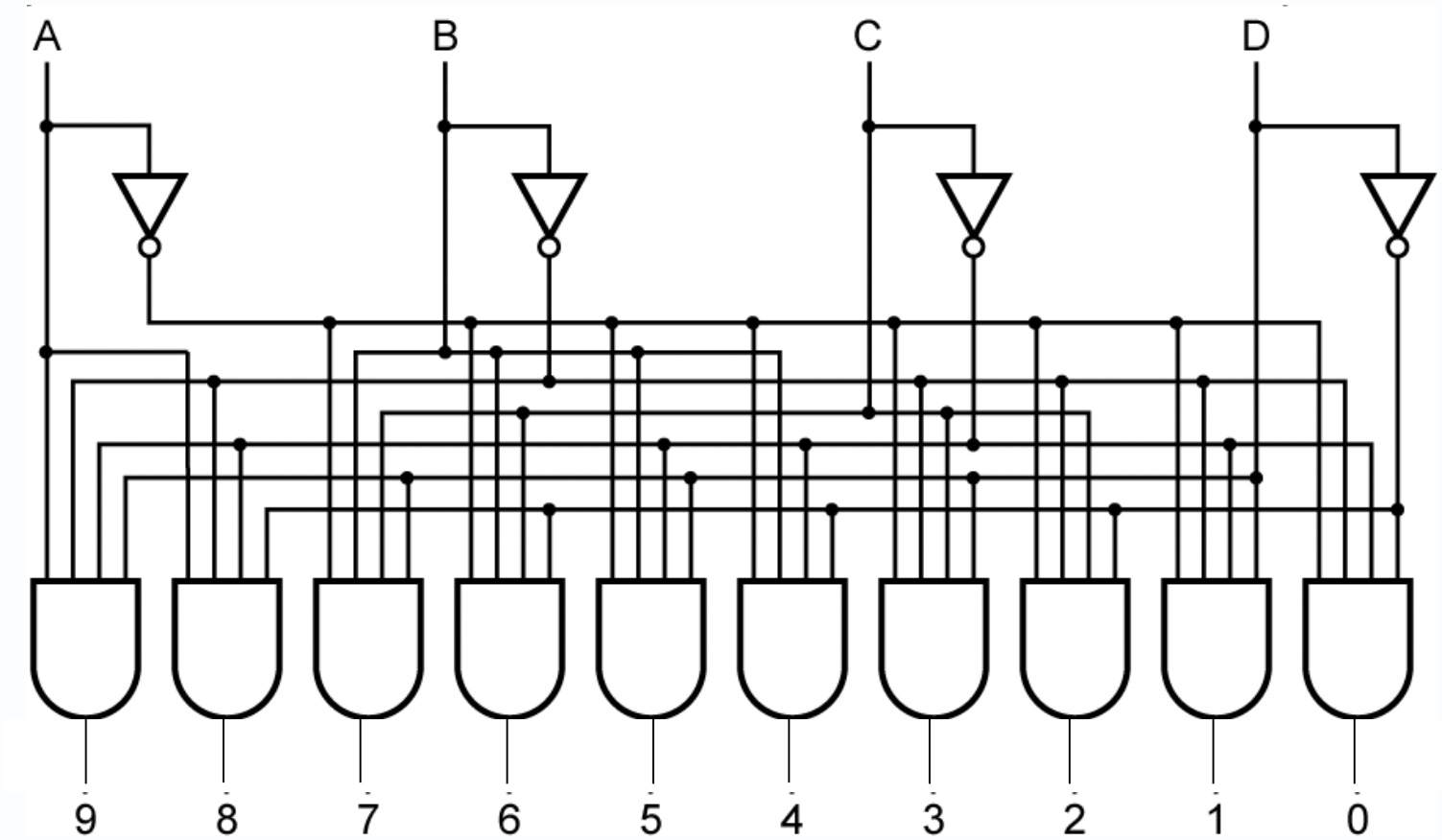
BCD Inputs				Outputs $(m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7,m_8,m_9)$									
A	B	C	D	0	1	2	3	4	5	6	7	8	9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0

All the output lines equal logic zero **except** the output representing the decimal number equivalent to the binary number ABCD. **This output equals logic one.**

e.g. when the input ABCD is 0101,

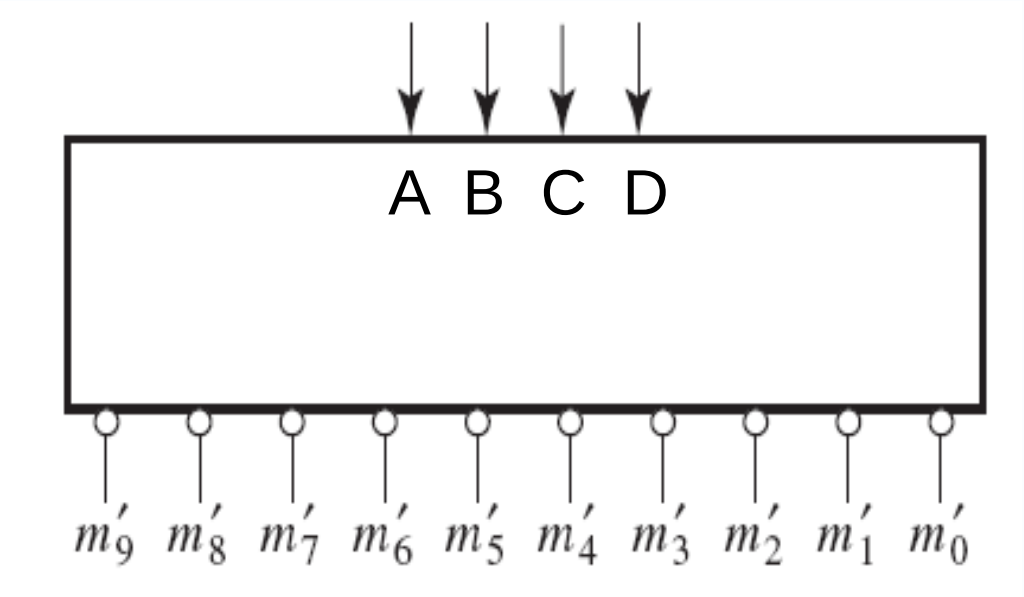
$$m_5 = A' \cdot B \cdot C' \cdot D = 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Circuit diagram for BCD to decimal decoder



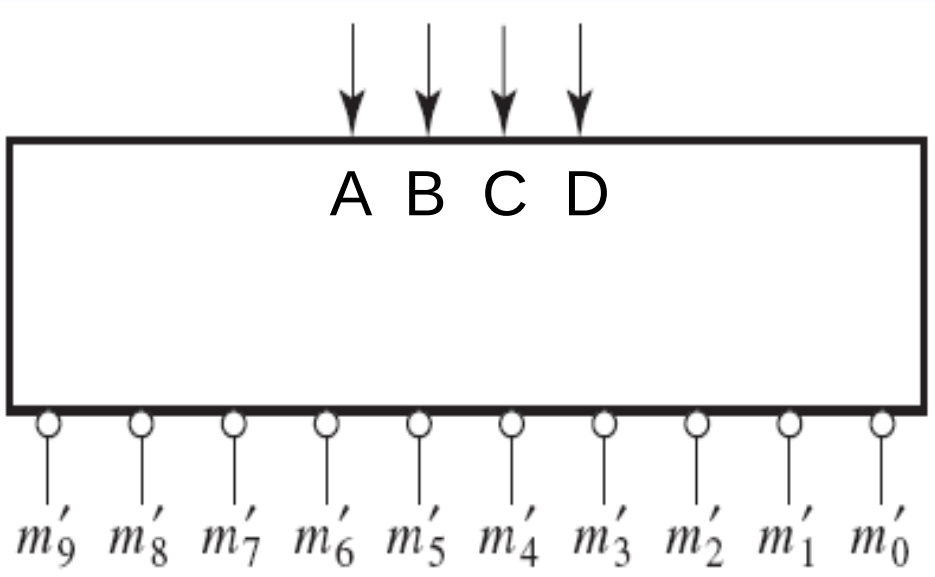
*Lots of AND gates here **...but...** NAND gates are cheaper to manufacture than AND gates (fewer transistors)*

BCD to decimal decoder
(4-to-10 line ***with inverted outputs***)



In this case, all the output lines equal logic **one** except the output that represents the decimal number equivalent to the binary number ABCD.
This output equals logic zero.

BCD to decimal decoder (4-to-10 line ***with inverted outputs***)

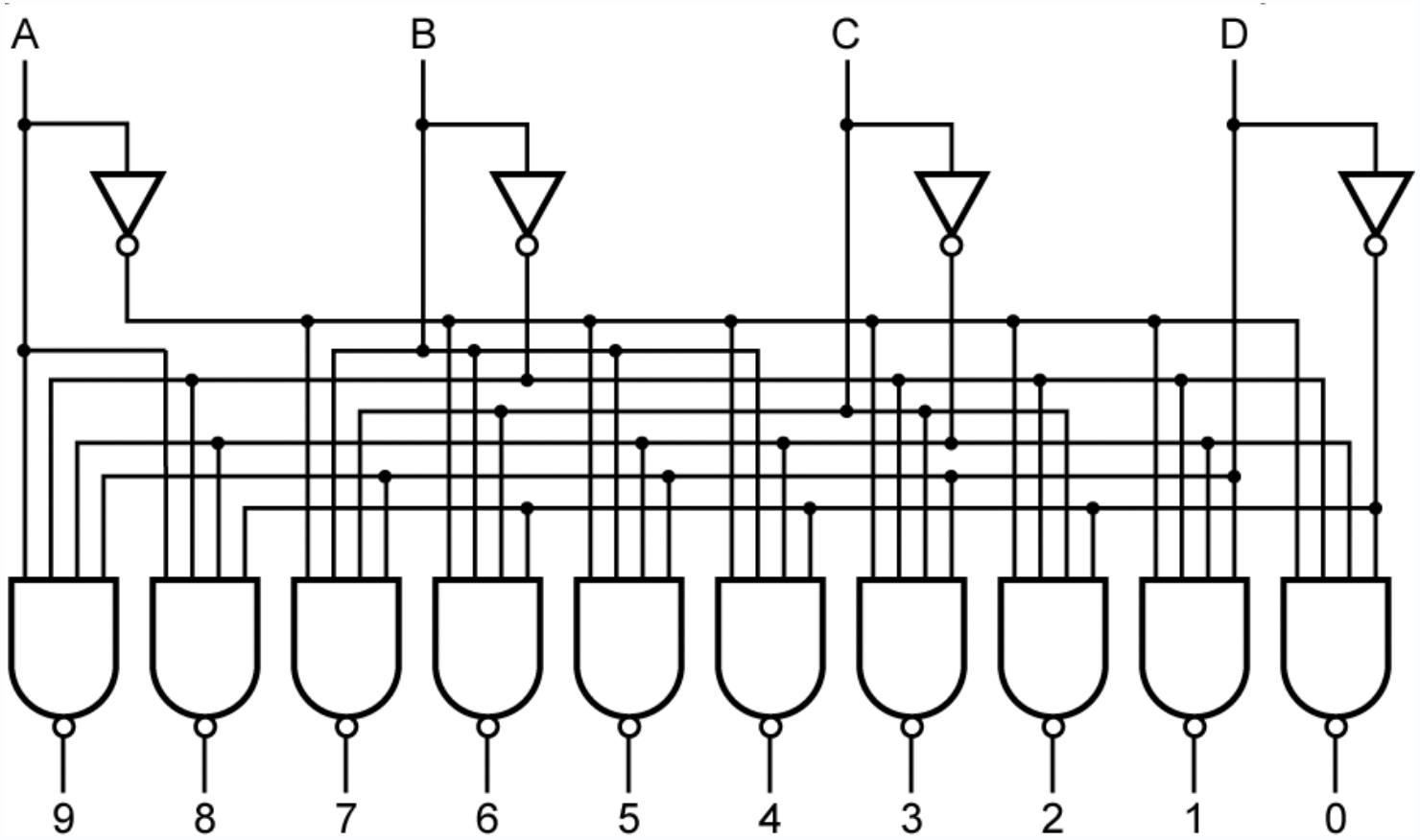


All the output lines equal logic **one** except the output that represents the decimal number equivalent to ABCD.
This output equals logic zero.

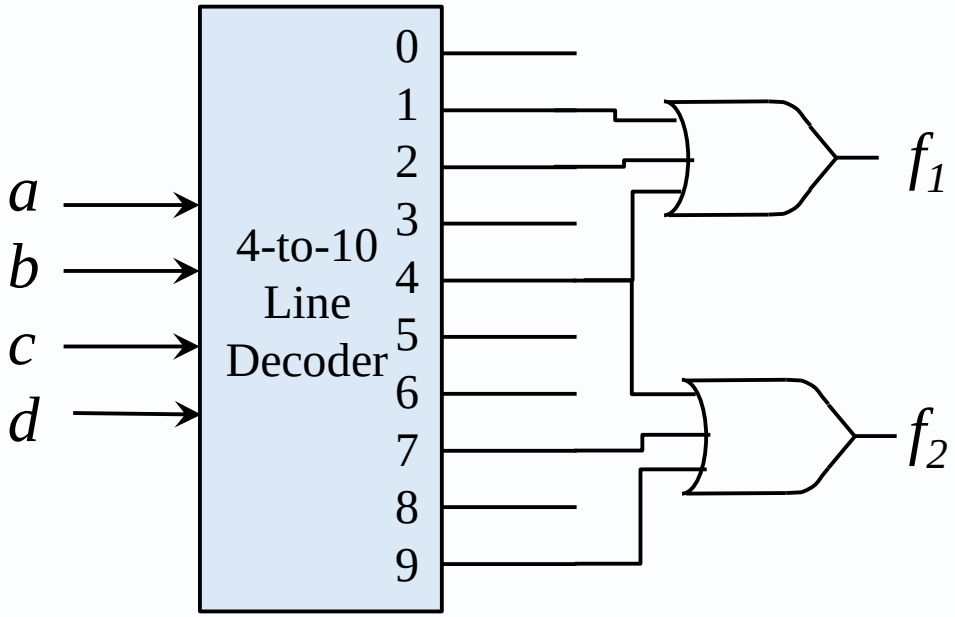
When input is greater than 9, all the output lines equal logic one.

BCD Inputs				Outputs									
A	B	C	D	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Circuit diagram for BCD to decimal decoder *(inverted outputs)*



Using a decoder to realise a function



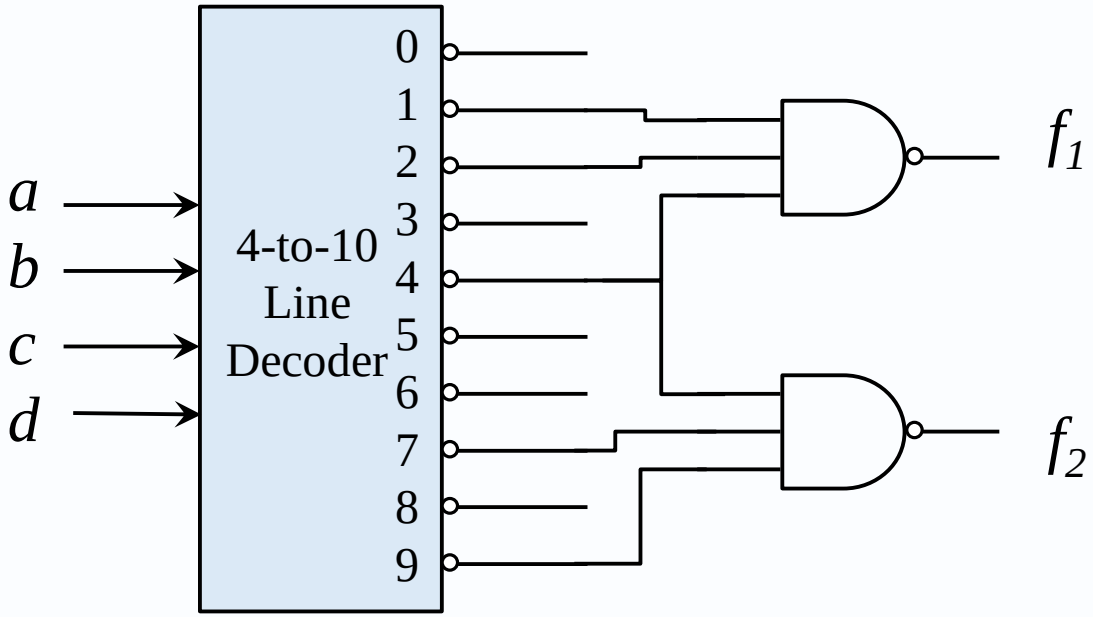
A decoder with *n* inputs generates all the minterms of *n* variables.

So *n*-variable functions can be realised by ORing together selected minterm outputs

$$f_1 = m_1 + m_2 + m_4$$

$$f_2 = m_4 + m_7 + m_9$$

Using a decoder to realise a function



A decoder with *n* inputs generates all the minterms of *n* variables.

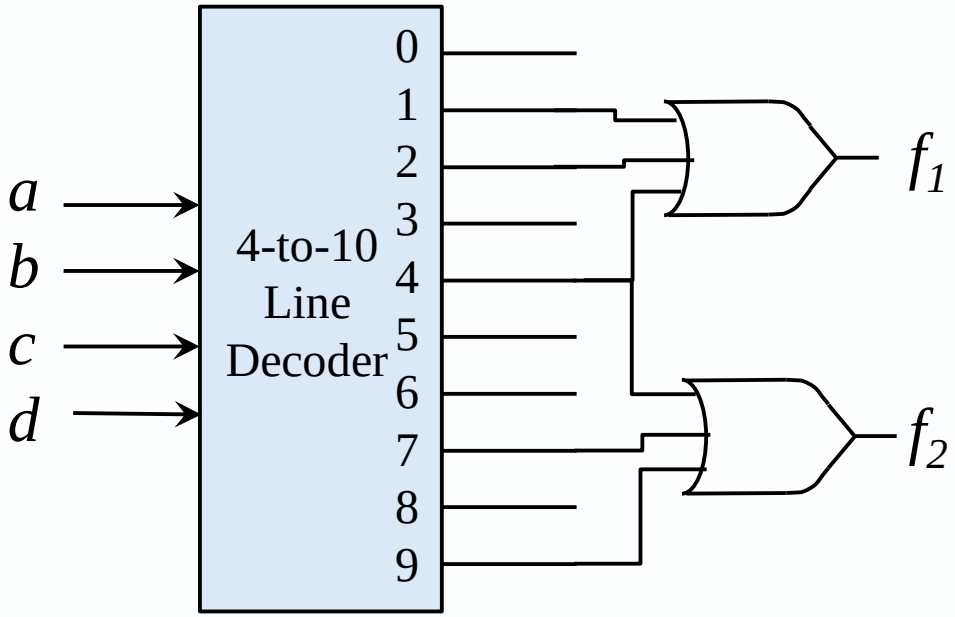
So *n*-variable functions can be realised by ORing together selected minterm outputs

$$f_1 = m_1 + m_2 + m_4 = (m_1' m_2' m_4')'$$

$$f_2 = m_4 + m_7 + m_9 = (m_4' m_7' m_9')'$$

Inverted outputs (and NAND gates instead of OR gates)

Using a decoder to realise a function



A decoder with *n* inputs generates all the minterms of *n* variables.

So *n*-variable functions can be realised by ORing together selected minterm outputs

$$f_1 = m_1 + m_2 + m_4$$

$$f_2 = m_4 + m_7 + m_9$$

Method

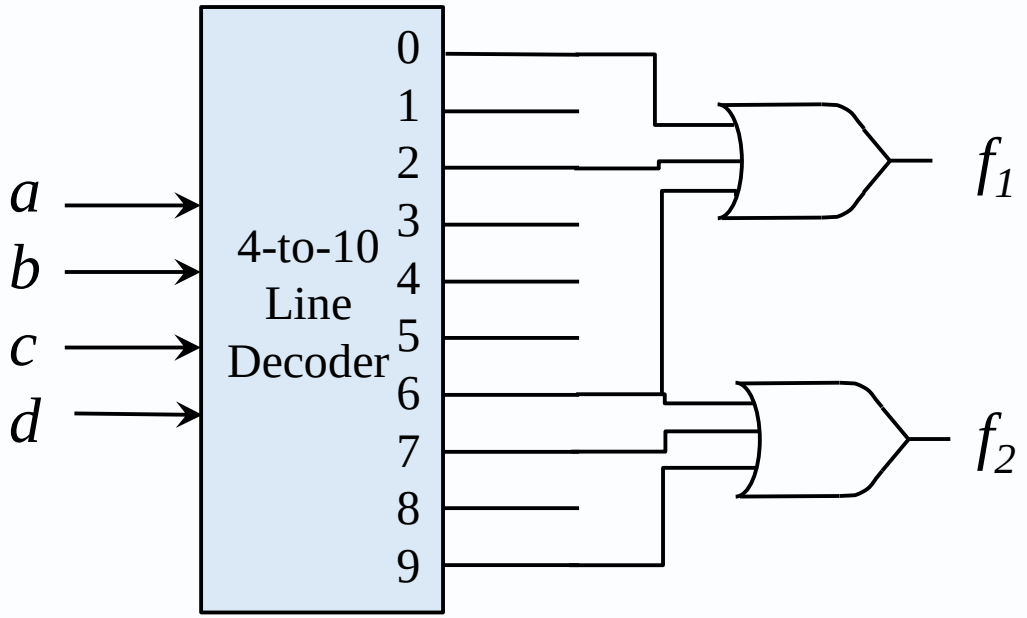
$$f_1 = m_1 + m_2 + m_4$$

$$f_2 = m_4 + m_7 + m_9$$

a	b	c	d	f_1	f_2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	1	1
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	1
1	0	0	0	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	0	0

Question

- What functions f_1 and f_2 does this circuit implement? Find a sum of products expression for each one.



Question

- Design a circuit for the following expression using a BCD to decimal decoder and two OR gates.

$$f_1 = a'b'c' + a'c'd$$

$$f_2 = a'bd + ab'c'd'$$

Encoders

An encoder performs the inverse function of a decoder. If input y_i is 1 and the other inputs are 0, then *abc* outputs represent a binary number equal to *i*.

For example, if $y_3 = 1$, then *abc* = 011.

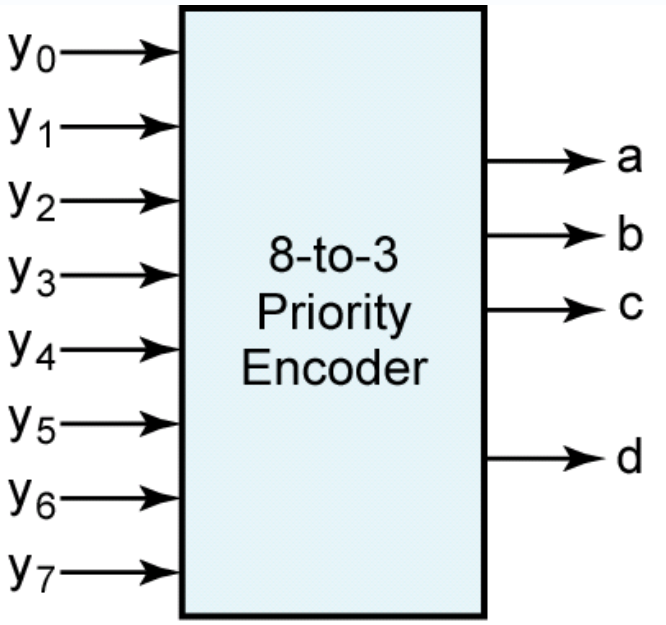
If more than one input is 1, the highest numbered input determines the output.

An extra output, *d*, is 1 if any input is 1, otherwise *d* is 0. This signal is needed to distinguish the case of all 0 inputs from the case where only y_0 is 1.

Priority Encoder

An encoder performs the inverse function of a decoder.

Inputs								Outputs			
y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7	a	b	c	d
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
X	1	0	0	0	0	0	0	0	0	1	1
X	X	1	0	0	0	0	0	0	1	0	1
X	X	X	1	0	0	0	0	0	1	1	1
X	X	X	X	1	0	0	0	1	0	0	1
X	X	X	X	X	1	0	0	1	0	1	1
X	X	X	X	X	X	1	0	1	1	0	1
X	X	X	X	X	X	X	1	1	1	1	1



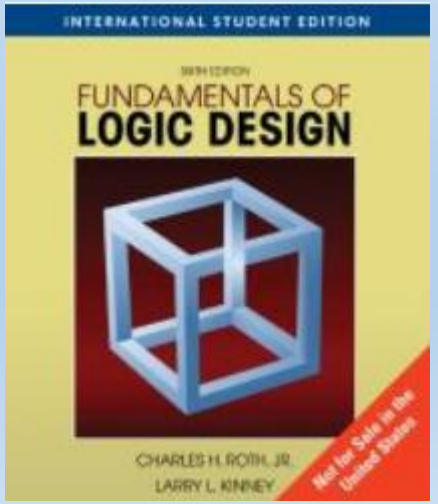
Summary and suggested reading

- Multiplexers (Section 9.2)
- Decoders (Section 9.4)
- Encoders (Section 9.4)

Next lecture is **tomorrow at 14.00 in 502-LT2**:

ROM, PLA and PAL…

Roth and Kinney *Fundamentals of Logic Design*



UNIVERSITY OF LIVERPOOL