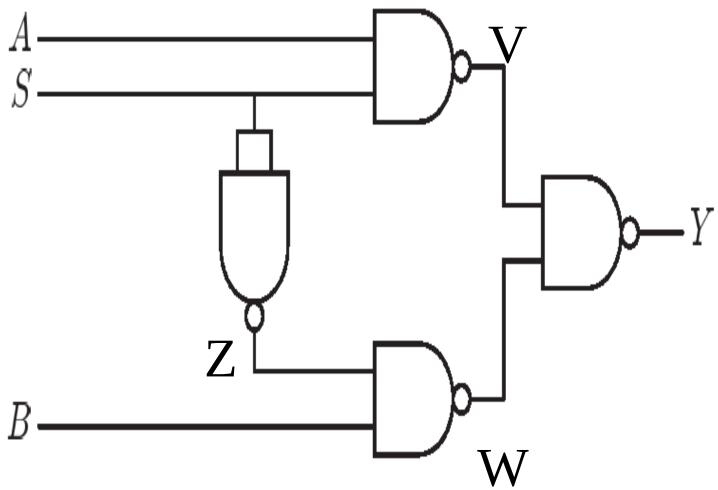
Computer Systems
Lecture 23

Overview

- Selector circuit
- Multiplexer
- Two-line decoder
- Data selector with two-line decoder
- Implementing a function with logic gates

Selector circuit



- This circuit implements the function:
  - Y = **not** (V **and** W), where,
  - V = **not** (A **and** S),
  - W= **not** (Z **and** B),
  - Z = **not** (S **and** S) = **not** S.
- Combining altogether we get:
  Y = **not** (**not** (A **and** S) **and** **not** (**not** (S) **and** B)).

V

Z

W

Truth table for selector circuit

A B S	A and S	V	Z	Z and B	W	V and W	Y
0 0 0	0	1	1	0	1	1	0
0 0 1	0	1	0	0	1	1	0
0 1 0	0	1	1	1	0	0	1
0 1 1	0	1	0	0	1	1	0
1 0 0	0	1	1	0	1	1	0
1 0 1	1	0	0	0	1	0	1
1 1 0	0	1	1	1	0	0	1
1 1 1	1	0	0	0	1	0	1

Selector function

- One may give a short definition of the function from the truth table:

If S = 1 then Y = A

if S = 0 then Y = B

Or **Y = (S&A)V(-S&B)**

Ex. Reimplement the selector circuit using AND, OR, NOT gates

6

Data selector, or multiplexer

Control W X Y Z	Data A B C D	OUT
0 0 0 1	a b c d	a
0 0 1 0	a b c d	b
0 1 0 0	a b c d	c
1 0 0 0	a b c d	d

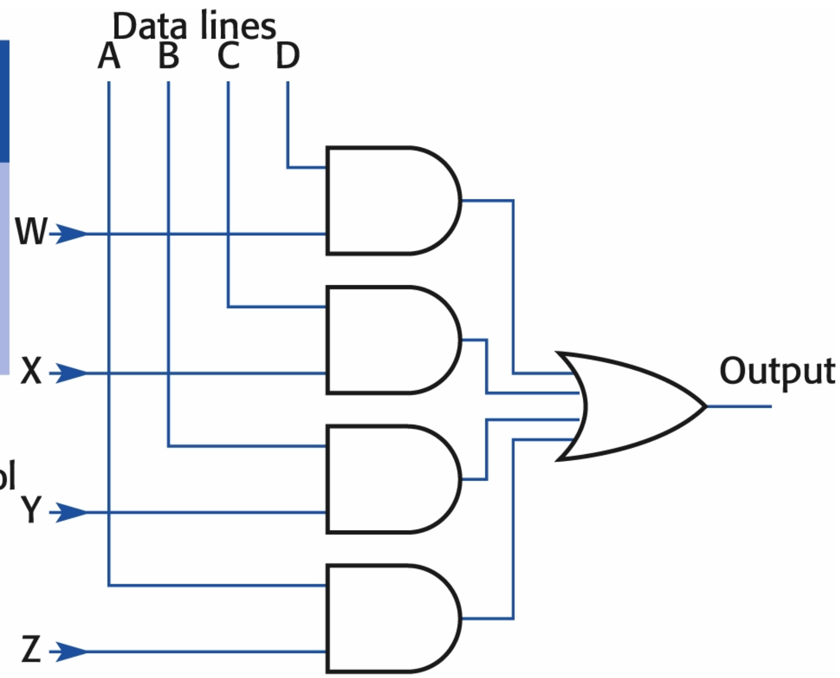


Fig. 4.5 Data selector circuit.

How to design the circuit?

- The circuit on the previous slide is a straightforward implementation of the function:

$$O = (A \textbf{ and } Z) \textbf{ or } (B \textbf{ and } Y) \textbf{ or } (C \textbf{ and } X) \textbf{ or } (D \textbf{ and } W).$$

8

The problem with the circuit

- What happens if more than one control line has the signal 1?
  - The circuit (the function it implements) does not behave as a selector.
- Can we do better?
  - Yes, use **two-line decoder** to replace four control lines.

Two-line decoder

Selector Y X	Line d c b a
0 0	0 0 0 1
0 1	0 0 1 0
1 0	0 1 0 0
1 1	1 0 0 0

Fig. 4.6 A two-line decoder.

Implementation of two-line decoder

- The implementation follows the description:
  - Detect pattern 00 (on YX lines), output the result from line A.
  - Detect pattern 01, output the result from line B.
  - Detect pattern 10, output the result from line C.
  - Detect pattern 11, output the result from line D.

Data selector with two-line decoder

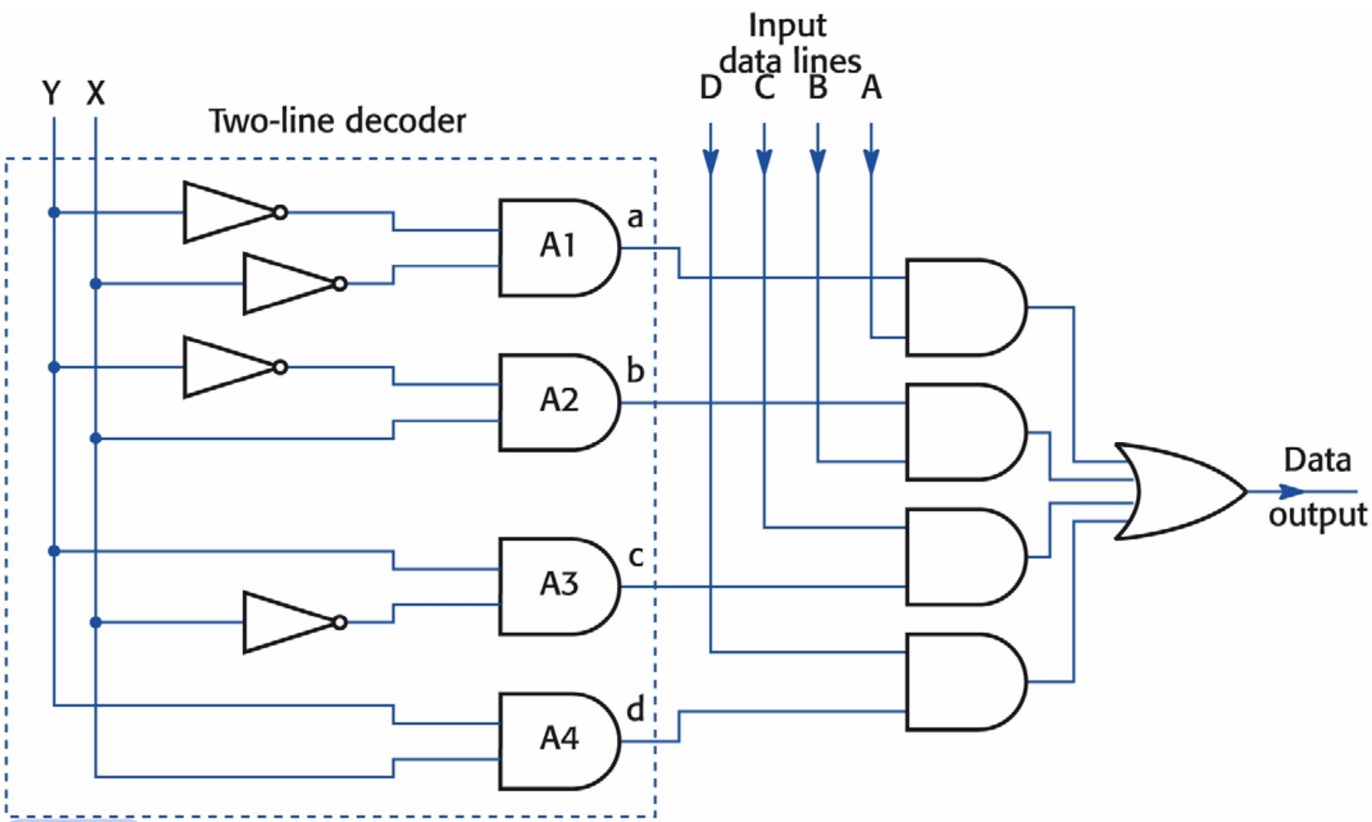


Fig. 4.7 Data multiplexer circuit using a two-line decoder.

Data selector with two-line decoder: logic equation

$O =$ (A **and** (**not** X **and** **not** Y))

or

(B **and** (X **and** **not** Y))

or

(C **and** (**not** X **and** Y))

or

(D **and** (X **and** Y)).

Cost comparison in gate count

- Multiplexer: 4 AND gates + 1 OR gate
- 2-line decoder: 8 AND gates + 1 OR gate + 4 NOT gates

Implementing a function

- Given a truth table for a logic function, to implement the function by a logic circuit one may proceed as follows:
  - Implement detectors (i.e. using AND/NOT gates) for all input patterns on which the function gives the output 1.
  - Connect the outputs of all detectors to the inputs by an OR gate.

15

Implementing a function

Truth table

i_1	i_2	i_3	i_4	O_1
1	1	0	0	1
1	1	0	1	1
0	0	1	0	1
1	0	1	0	1
0	1	0	1	1
....				0
…				0
....				0

The function gives 0 on all other inputs.

Short form

i_1	i_2	i_3	i_4	O_1
1	1	0	*	1
*	0	1	0	1
0	1	0	1	1

16

Implementing a function

- Logic expression:

$$O_1 = (i_1 \textbf{ and } i_2 \textbf{ and } (\textbf{not } i_3))$$
$$\textbf{or}$$
$$((\textbf{not } i_2) \textbf{ and } i_3 \textbf{ and } (\textbf{not } i_4))$$
$$\textbf{or}$$
$$((\textbf{not } i_1) \textbf{ and } i_2 \textbf{ and } (\textbf{not } i_3) \textbf{ and } i_4).$$

Sum-of-products Implementation

O = (*i1* and *i2* and (not *i3*)) or
((not *i2*) and *i3* and (not *i4*)) or
((not *i1*) and *i2* and (not *i3*) and *i4*)

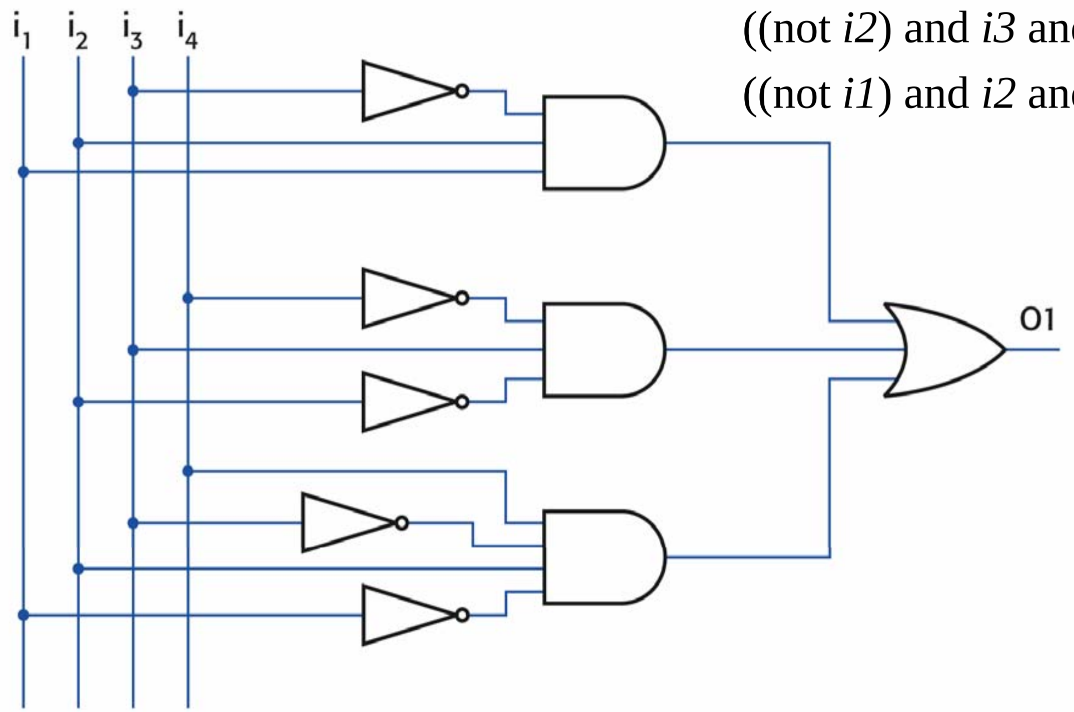


Fig. 4.9 Implementing a sum-of-products logic equation.

Ex. Implement the following function using Boolean gates

Truth table

i_1	i_2	i_3	i_4	O_1
1	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	0	1
0	1	0	1	0
....				0
…				0
....				0

The function gives 0 on all the other inputs.

Short form

i_1	i_2	i_3	i_4	O_1

19

Exercise

- Draw a Boolean circuit to implement the XOR function using NOT, AND, OR gates

Readings

- [Wil06] Chapter 4, sections 4.3, 4.4, 4.6, 4.7.