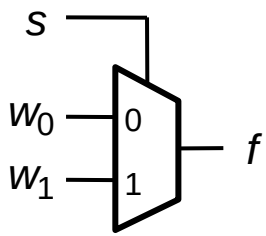
ECEN 248 –Introduction to Digital Systems Design (Spring 2008)

(Sections: 501, 502, 503, 507)

Prof. Xi Zhang
ECE Dept, TAMU, 333N WERC
http://www.ece.tamu.edu/~xizhang/ECEN248

Chapter 6.1 Multiplexers

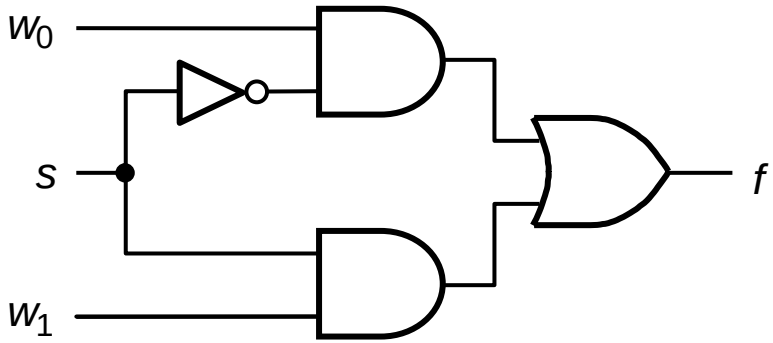
2-to-1 multiplexer



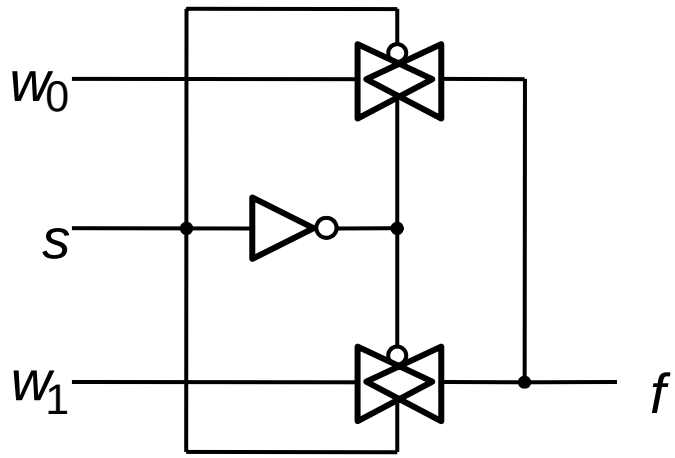
(a) Graphical symbol

s	f
0	w_0
1	w_1

(b) Truth table



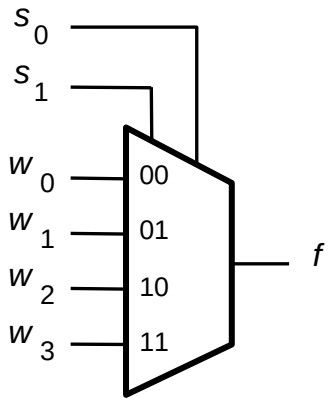
(c) Sum-of-products circuit



(d) Circuit with transmission gates

Figure 6.1. A 2-to-1 multiplexer.

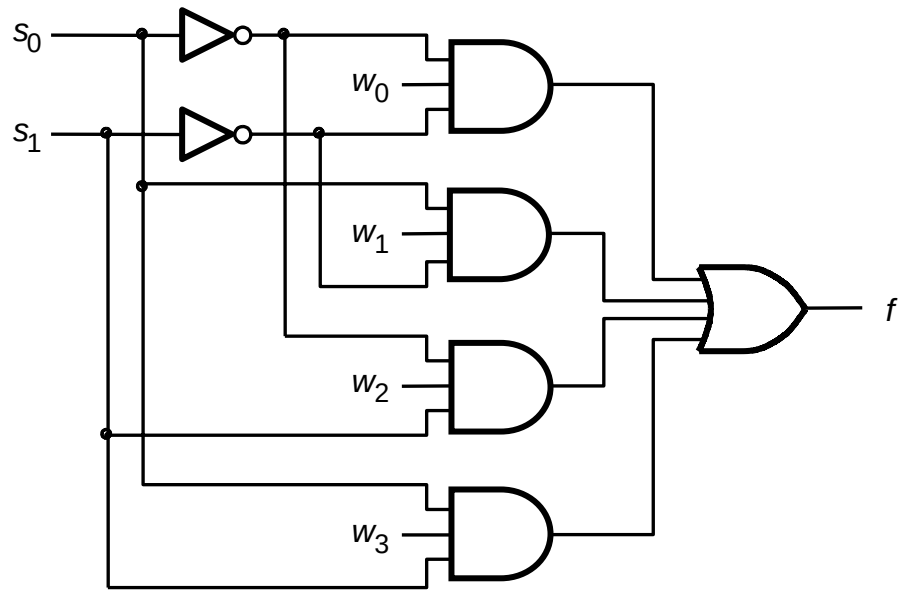
4-to-1 multiplexer



(a) Graphic symbol

s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

(b) Truth table



(c) Circuit

Figure 6.2. A 4-to-1 multiplexer.

4-to-1 multiplexer implemented by 2-to-1 multiplexer

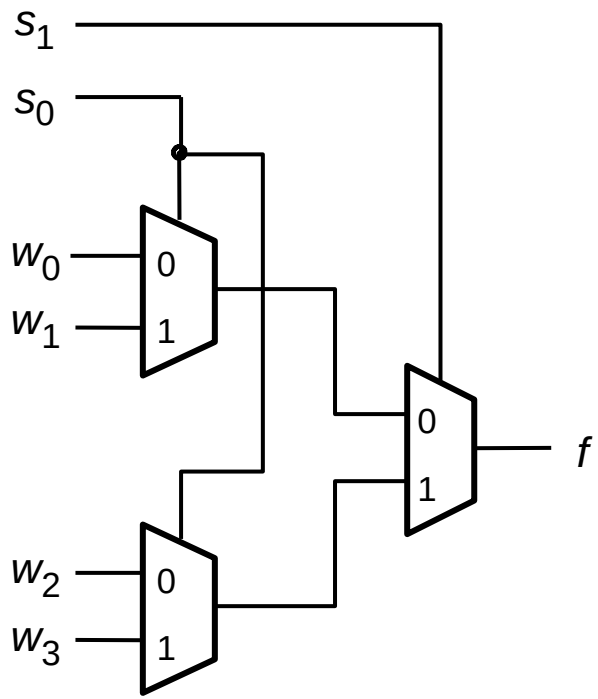


Figure 6.3. Using 2-to-1 multiplexers to build a 4-to-1 multiplexer.

16-to-1 multiplexer

- Building 16-to-1 multiplexer by using four 4-to-1 multiplexers.

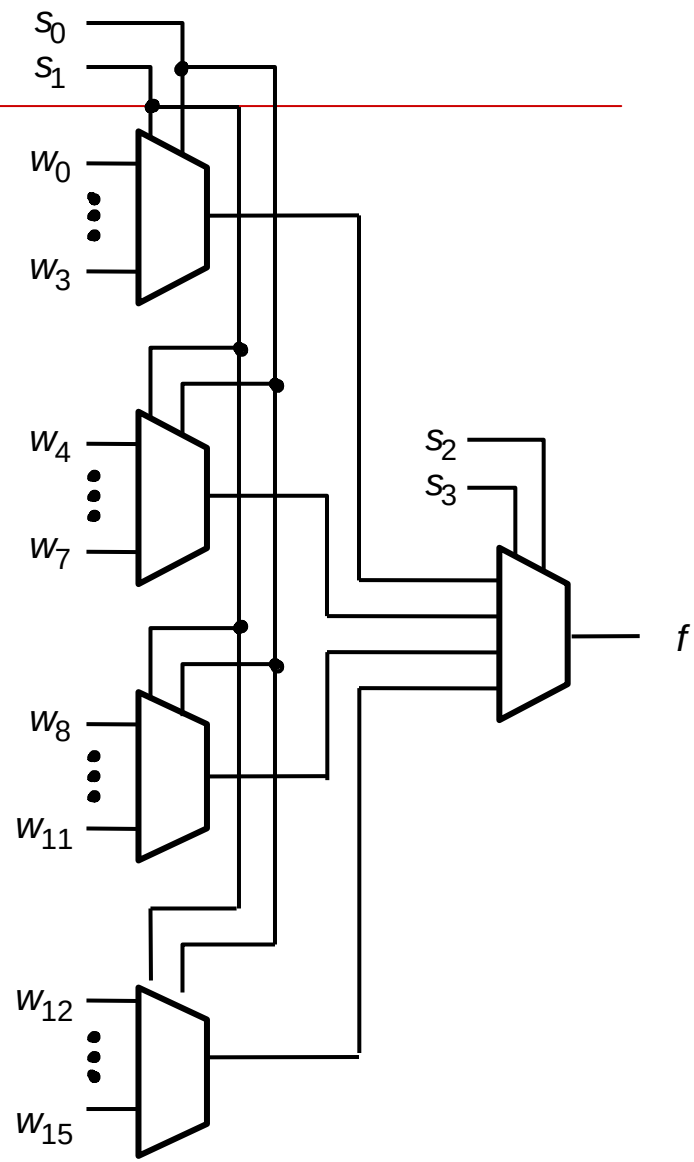
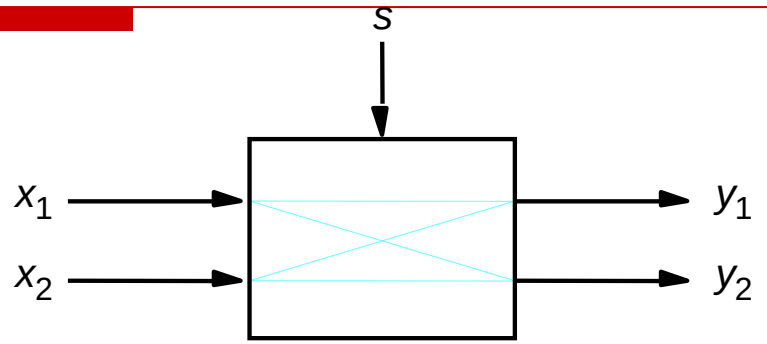


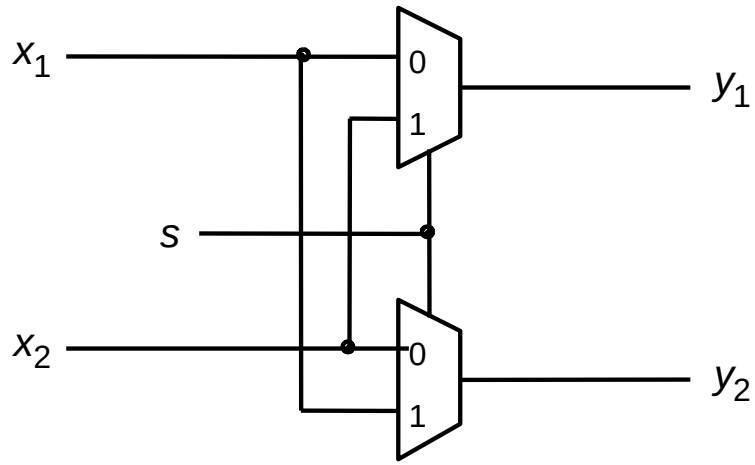
Figure 6.4. A 16-to-1 multiplexer.

A practical application of multiplexers

- $s = 0$
  - $x_1 \rightarrow y_1$
  - $x_2 \rightarrow y_2$
- $s = 1$
  - $x_2 \rightarrow y_1$
  - $x_1 \rightarrow y_2$



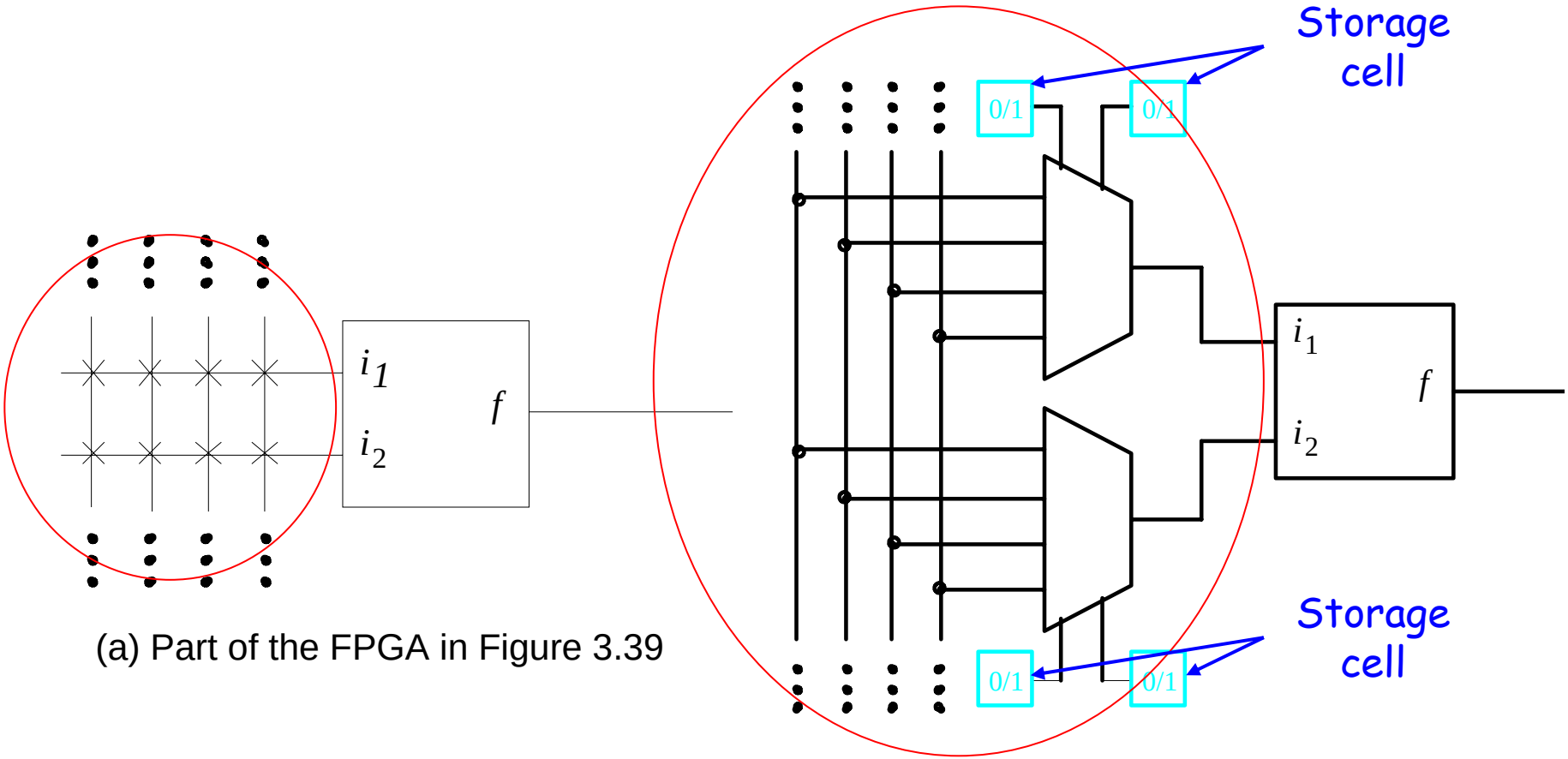
(a) A 2x2 crossbar switch



(b) Implementation using multiplexers

Figure 6.5. A practical application of multiplexers.

Implementing programmable switches in an FPGA



(a) Part of the FPGA in Figure 3.39

(c) Implementation using multiplexers

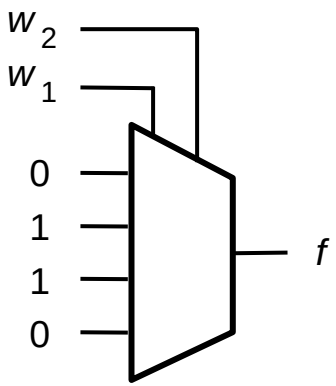
Figure 6.6. Implementing programmable switches in an FPGA.

Chapter 6.1.1 Synthesis of Logic Functions Using Multiplexers

Synthesis of a logic function using multiplexers

$$f = w_1 \oplus w_2$$

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0



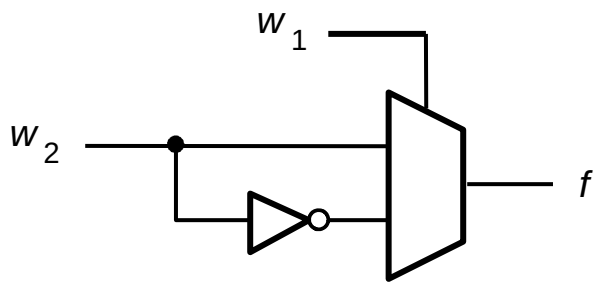
(a) Implementation using a 4-to-1 multiplexer

Not efficient

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	0	0

w_1	f
0	w_2
1	$\bar{w}_2$

(b) Modified truth table



(c) Circuit

More efficient

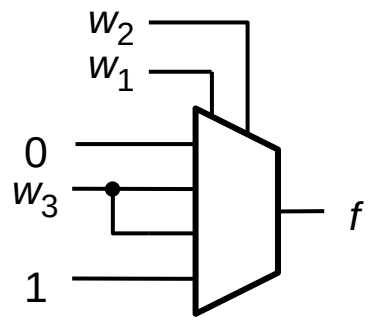
Figure 6.7. Synthesis of a logic function using multiplexers.

Three-input majority function by using a 4-to-1 multiplexer

w_1	w_2	w_3	f
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

w_1	w_2	f
0	0	0
0	1	w_3
1	0	w_3
1	1	1

(a) Modified truth table



(b) Circuit

- n-input Majority function:
  - The output is 1 if more than half inputs are 1;
  - Otherwise, the output is equal to 0.

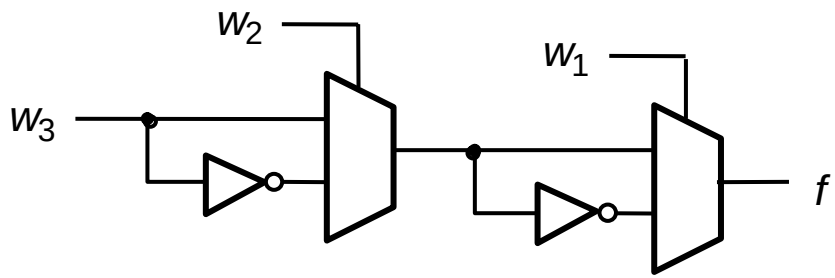
Figure 6.8. Implementation of the three-input majority function using a 4-to-1 multiplexer.

Three-input XOR by 2-to-1 multiplexers

$$f = w_1 \oplus w_2 \oplus w_3$$

w_1	w_2	w_3	f	
0	0	0	0	$w_2 \oplus w_3$
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	$\overline{w_2 \oplus w_3}$
1	0	1	0	
1	1	0	0	
1	1	1	1	

(a) Truth table



(b) Circuit

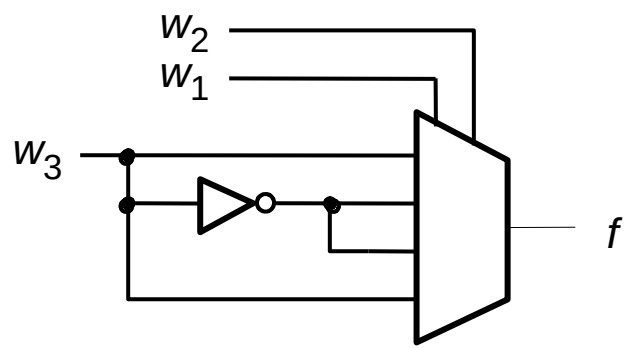
Figure 6.9. Three-input XOR implemented with 2-to-1 multiplexers.

Three-input XOR implemented with a 4-to-1 multiplexer

$$f = w_1 \oplus w_2 \oplus w_3$$

w_1	w_2	w_3	f	
0	0	0	0	w_3
0	0	1	1	
0	1	0	1	$\bar{w}_3$
0	1	1	0	
1	0	0	1	$\bar{w}_3$
1	0	1	0	
1	1	0	0	w_3
1	1	1	1	

(a) Truth table



(b) Circuit

Figure 6.10. Three-input XOR function implemented with a 4-to-1 multiplexer.

Chapter 6.1.2 Multiplexer Synthesis Using Shannon's Expansions

Multiplexers synthesis using Shannon's Expansion

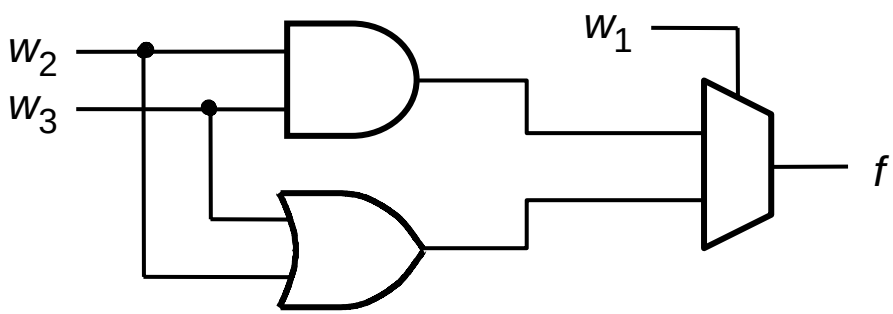
$$f = \overline{w}_1 w_2 w_3 + w_1 \overline{w}_2 w_3 + w_1 w_2 \overline{w}_3 + w_1 w_2 w_3$$

$$f = w_1(w_2 + w_3) + \overline{w}_1(w_2 w_3)$$

w_1	w_2	w_3	f
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

w_1	f
0	$w_2 w_3$
1	$w_2 + w_3$

(b) Truth table



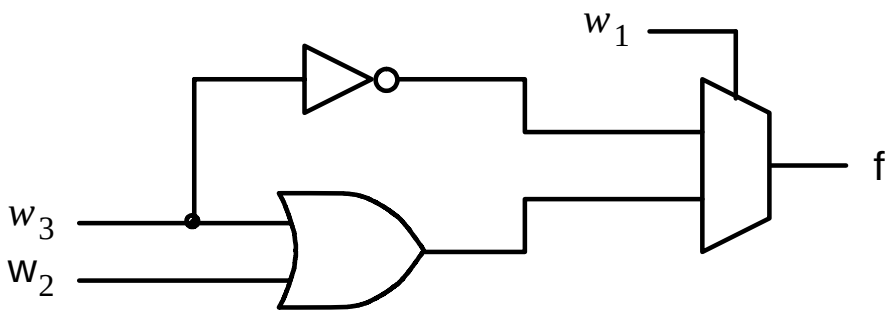
(b) Circuit

Figure 6.11. The three-input majority function implemented using a 2-to-1 multiplexer.

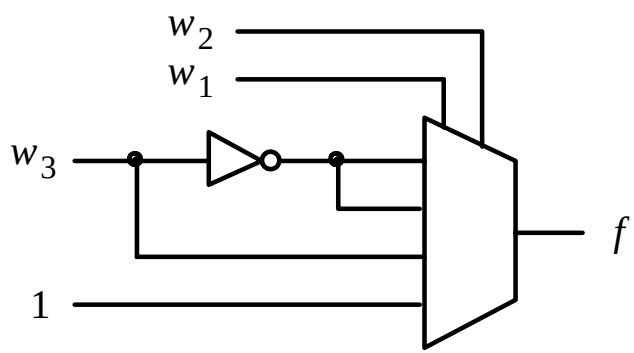
Synthesis for $f = \overline{w}_1\overline{w}_3 + w_1w_2 + w_1w_3$

$$f = \overline{w}_1\overline{w}_3 + w_1w_2 + w_1w_3$$
$$= \overline{w}_1(\overline{w}_3) + w_1(w_2 + w_3)$$

$$f = \overline{w}_1\overline{w}_3 + w_1w_2 + w_1w_3$$
$$= \overline{w}_1\overline{w}_2 f_{\overline{w}_1\overline{w}_2} + \overline{w}_1 w_2 f_{\overline{w}_1 w_2}$$
$$+ w_1\overline{w}_2 f_{w_1\overline{w}_2} + w_1w_2 f_{w_1w_2}$$
$$= \overline{w}_1\overline{w}_2(\overline{w}_3) + \overline{w}_1 w_2(\overline{w}_3) + w_1\overline{w}_2(w_3) + w_1w_2(1)$$



(a) Using a 2-to-1 multiplexer



(b) Using a 4-to-1 multiplexer

Figure 6.12. The circuits synthesized in Example 6.6.

Synthesis for 3-input majority function

$$f = w_1w_2 + w_1w_3 + w_2w_3$$
$$= \overline{w}_1(w_2w_3) + w_1(w_2 + w_3 + w_2w_3)$$
$$= \overline{w}_1(w_2w_3) + w_1(w_2 + w_3)$$

Let $g = w_1w_2$ *and* $h = w_1 + w_2$

$$g = \overline{w}_2(0) + w_2(w_3)$$
$$h = \overline{w}_2(w_3) + w_2(1)$$

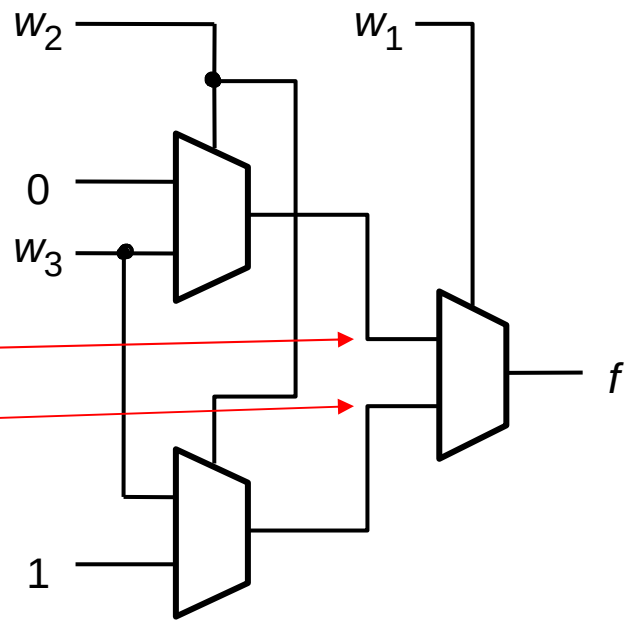


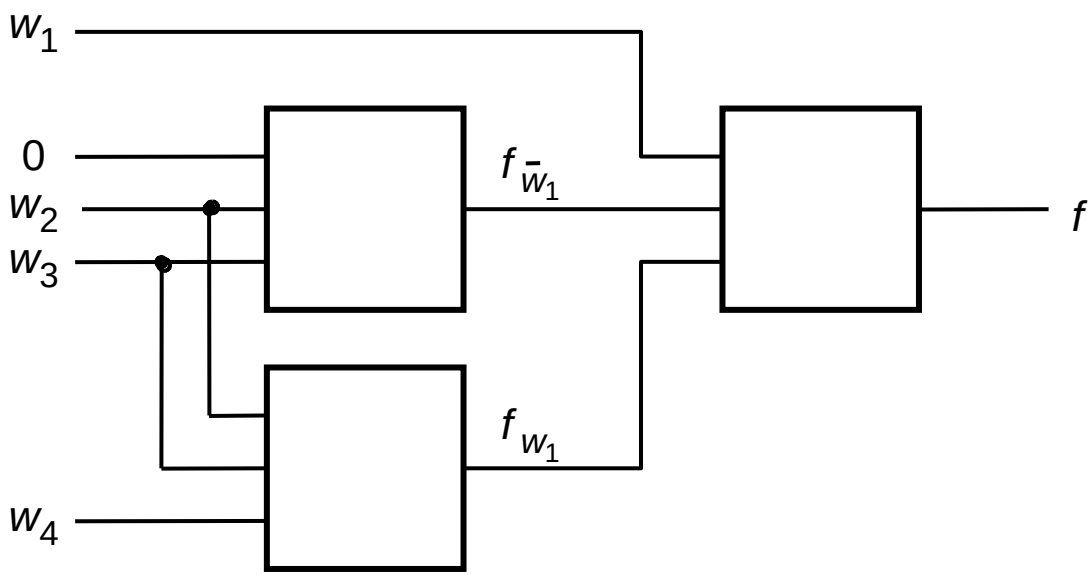
Figure 6.13. The circuit synthesized in Example 6.7

Circuits Ex. 6.8 by using 3-input lookup table

$$f = \overline{w}_2 w_3 + \overline{w}_1 w_2 \overline{w}_3 + w_2 \overline{w}_3 w_4 + w_1 \overline{w}_2 \overline{w}_4$$

$$f = \overline{w}_1 f_{\overline{w}_1} + w_1 f_{w_1}$$

$$= \overline{w}_1 \left(\overline{w}_2 w_3 + w_2 \overline{w}_3 \right) + w_1 \left(\overline{w}_2 w_3 + \overline{w}_2 w_3 w_4 + \overline{w}_2 \overline{w}_4 \right)$$



(a) Using three 3-LUTs

Figure 6.14. Circuits synthesized in Example 6.8

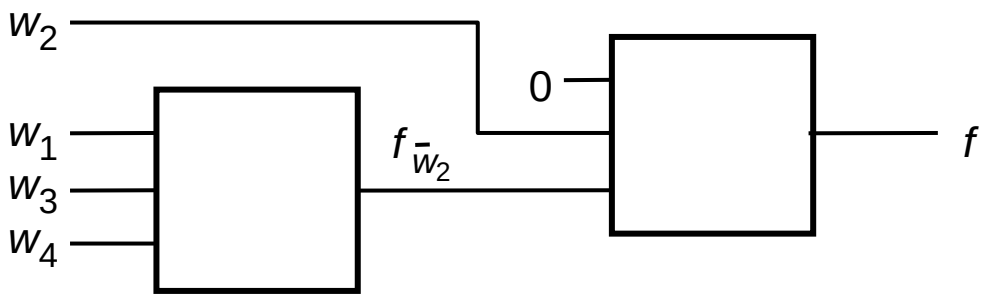
Circuits Ex. 6.8 by using 3-input lookup table

$$f = \overline{w}_2 w_3 + \overline{w}_1 w_2 \overline{w}_3 + w_2 \overline{w}_3 w_4 + w_1 \overline{w}_2 \overline{w}_4$$

$$f = \overline{w}_2 f_{\overline{w}_2} + w_2 f_{w_2}$$
$$= \overline{w}_2 (w_3 + w_1 \overline{w}_4) + w_2 (\overline{w}_1 \overline{w}_3 + \overline{w}_3 w_4)$$

Oberserving $\bar{f}_{\overline{w}_2} = f_{w_2}$ based on DeMorgan's theorem

We need only two 3-lookup tables,



(b) Using two 3-LUTs

Figure 6.14. Circuits synthesized in Example 6.8

Chapter 6.2 Decoders

Decoders

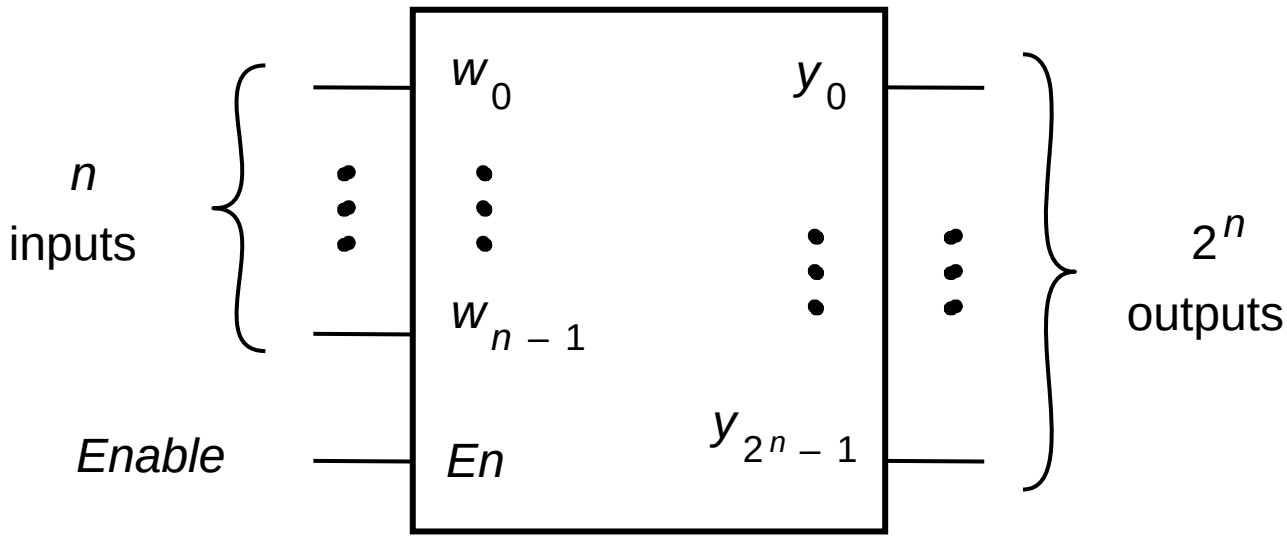
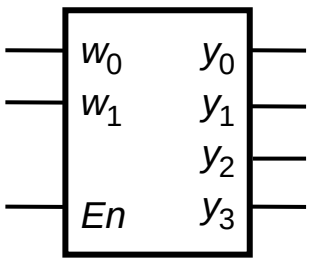


Figure 6.15. An n-to-2^n binary decoder.

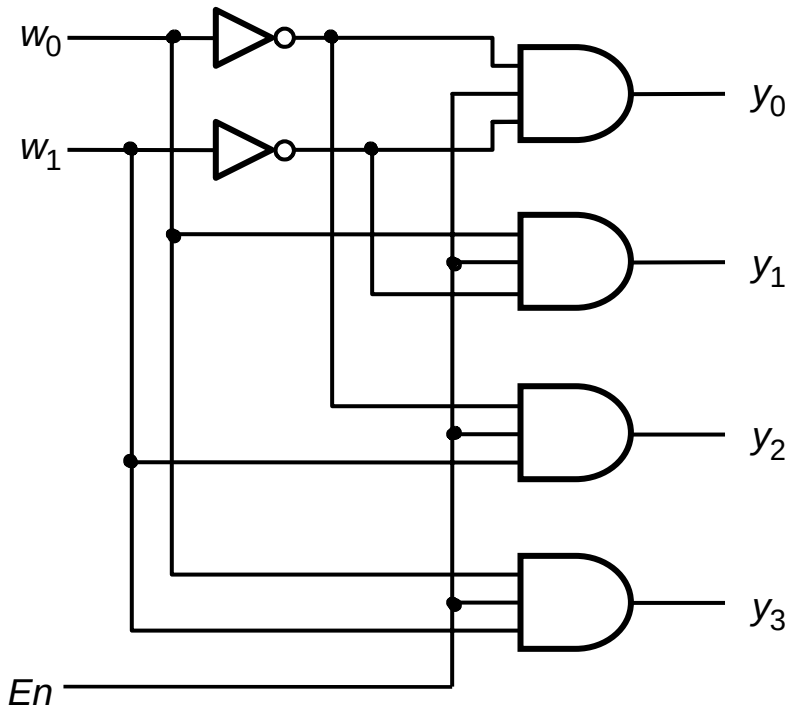
2-to-4 decoder

En	w_1	w_0	y_0	y_1	y_2	y_3
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
0	x	x	0	0	0	0

(a) Truth table



(b) Graphical symbol



(c) Logic circuit

Figure 6.16. A 2-to-4 decoder.

3-to-8 decoder by using 2-to-4 decoders

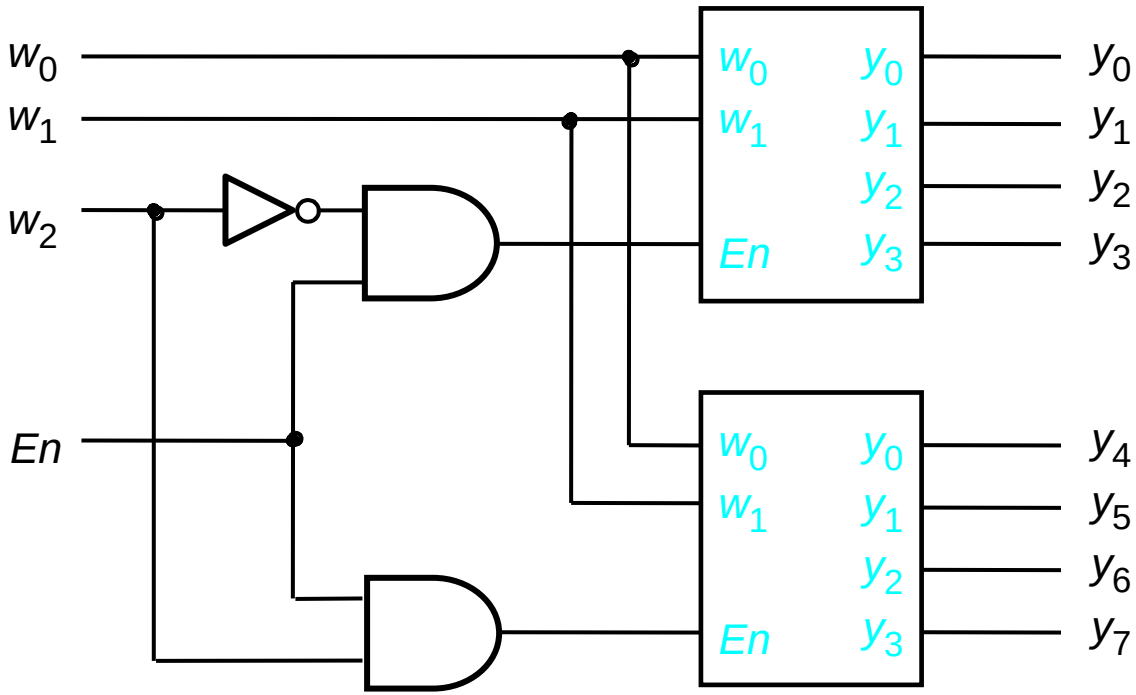


Figure 6.17. A 3-to-8 decoder using two 2-to-4 decoders.

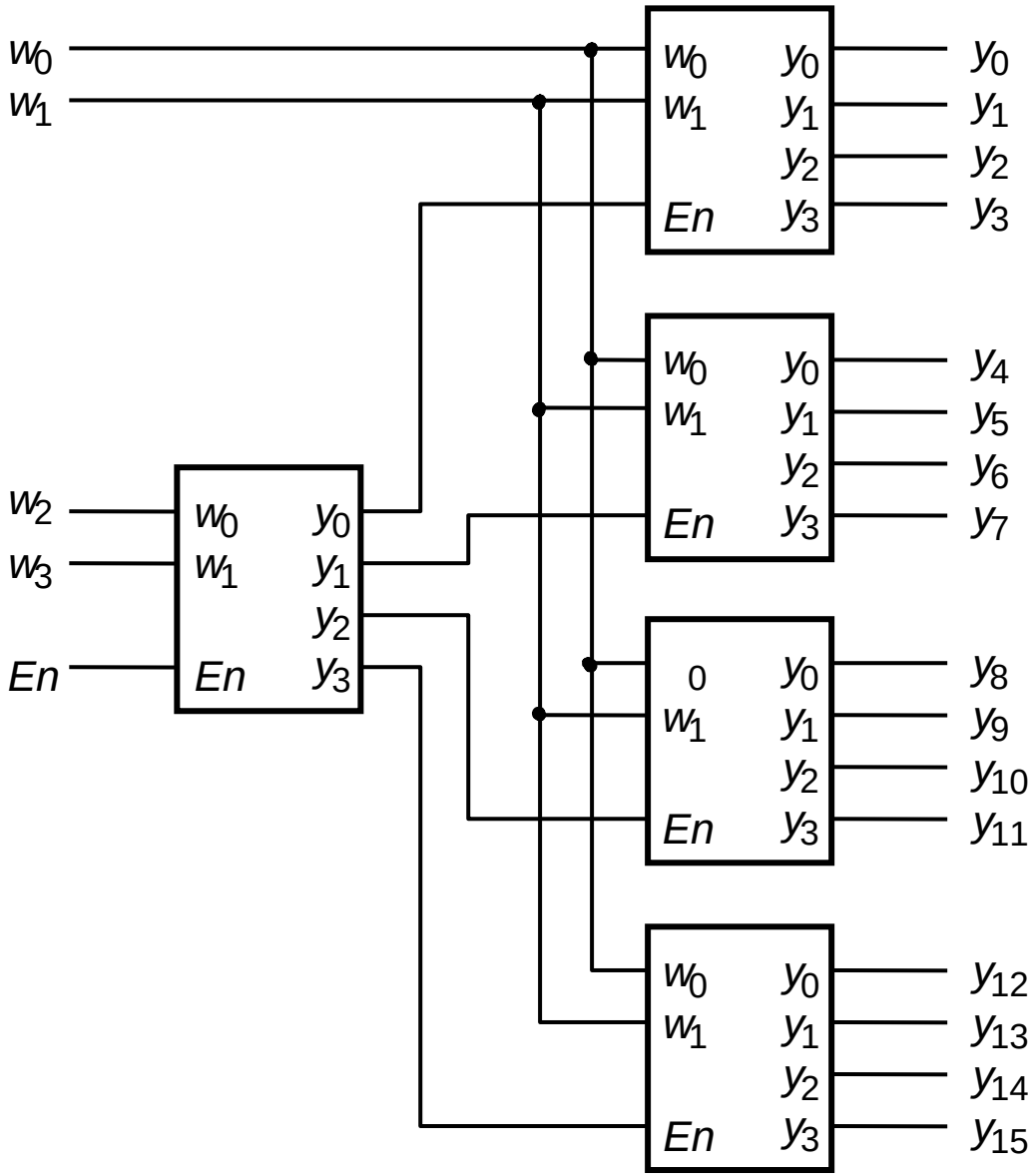


Figure 6.18. A 4-to-16 decoder built using a decoder tree.

Using decoder to build multiplexer

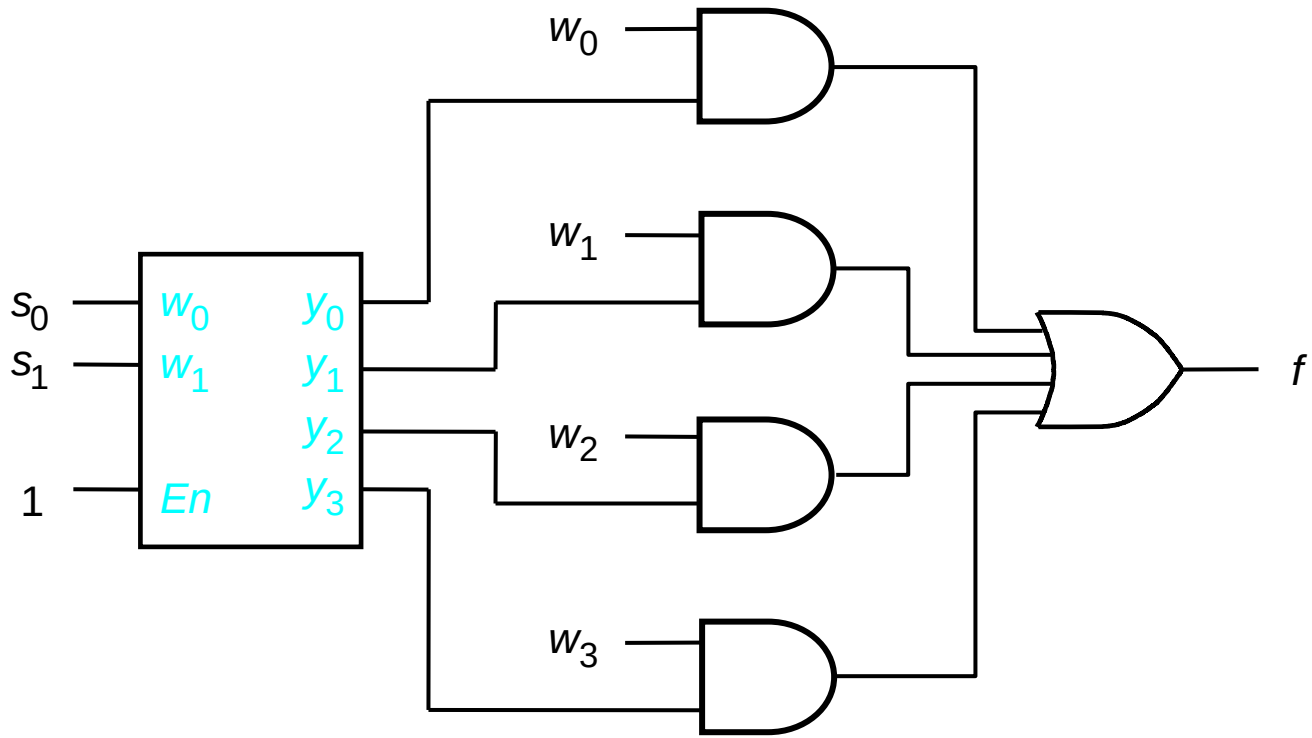
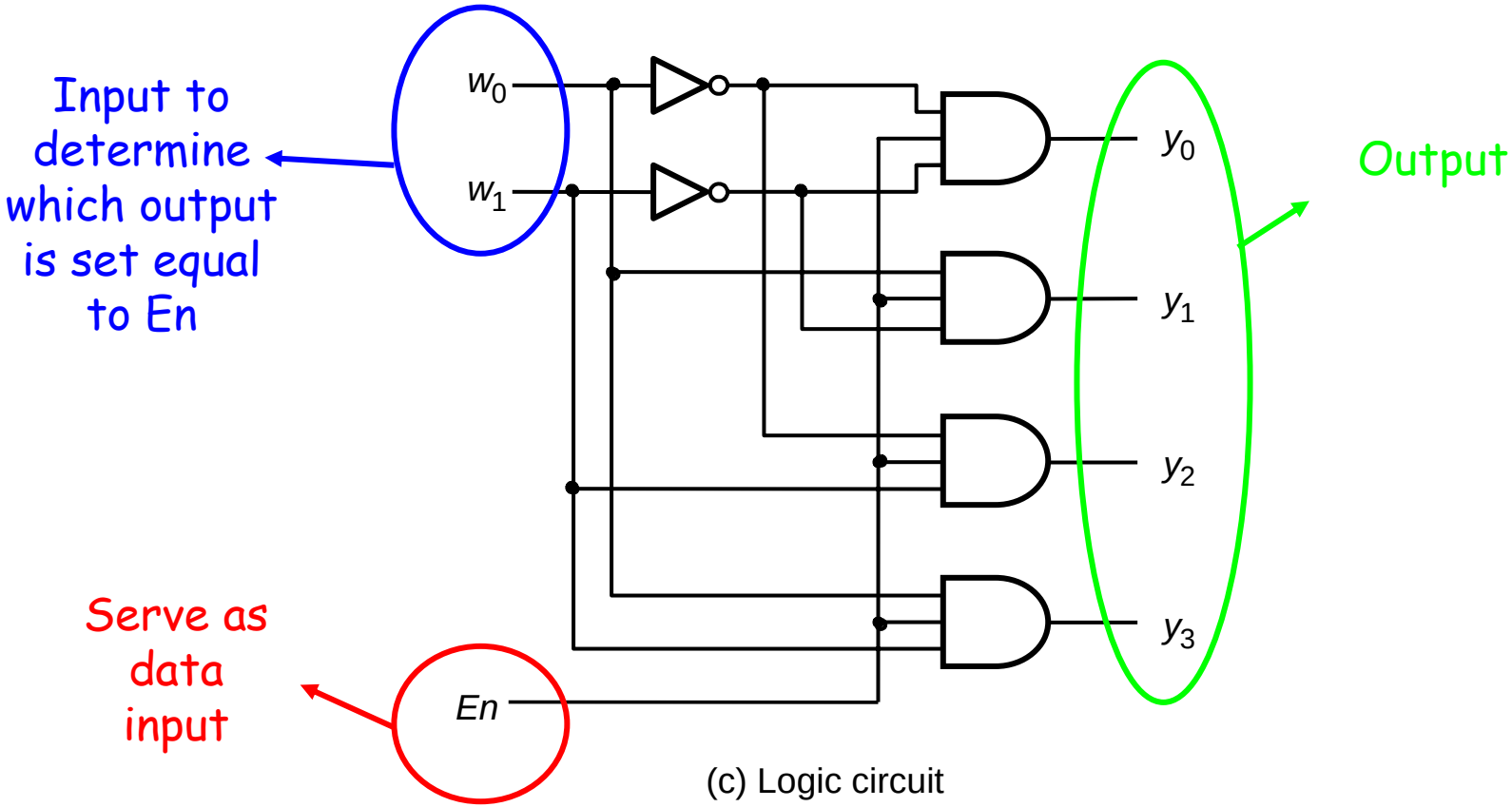


Figure 6.19. A 4-to-1 multiplexer built using a decoder.

Demultiplexer

- Multiplexer
  - The purpose of Multiplexer is to multiplex the n data inputs onto the single data output under control of the select inputs.
- Demultiplexer
  - Perform the opposite function of the multiplexer.
  - Place the value of a single data input onto multiple data outputs.
  - Can be implemented by using a decoder circuit.

Example: 1-to-4 demultiplexer by using 2-to-4 decoder



(c) Logic circuit

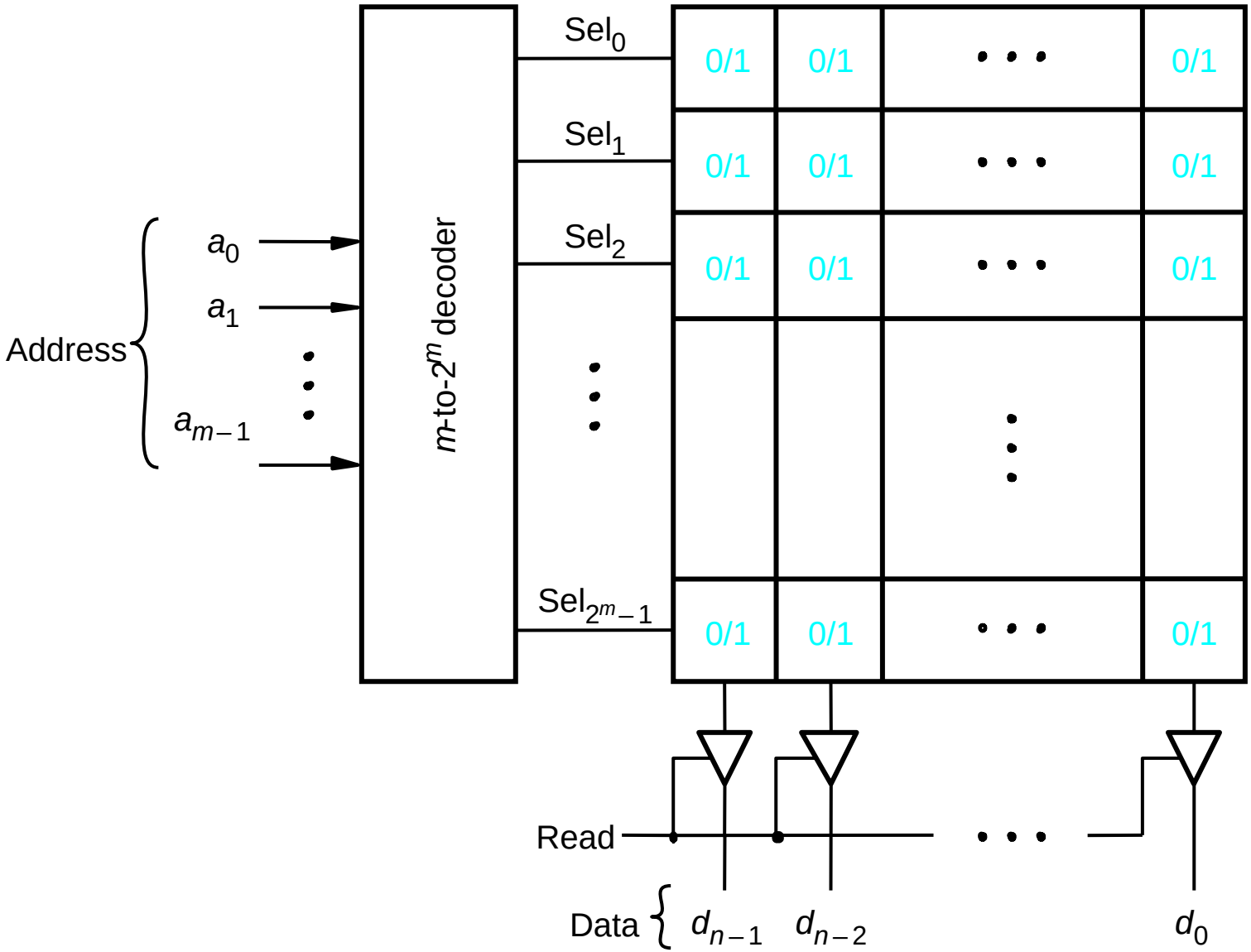


Figure 6.21. A 2^m x n read-only memory (ROM) block.