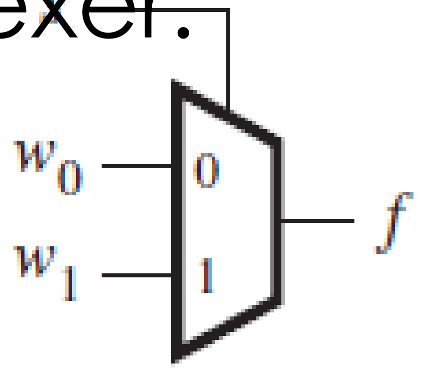
Multiplexers

A multiplexer circuit has a number of data inputs, one or more select inputs, and one output. It passes the signal value on one of the data inputs to the output. The data input is selected by the values of the select
inputs.

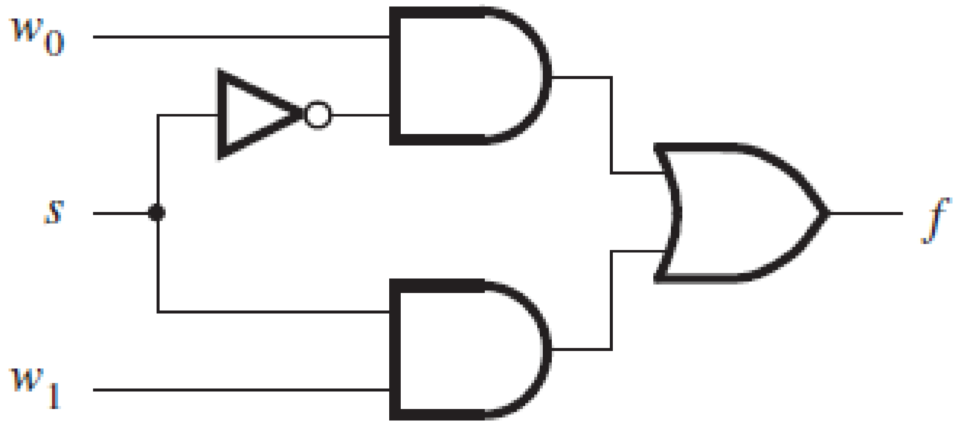
2-to-1 multiplexer.



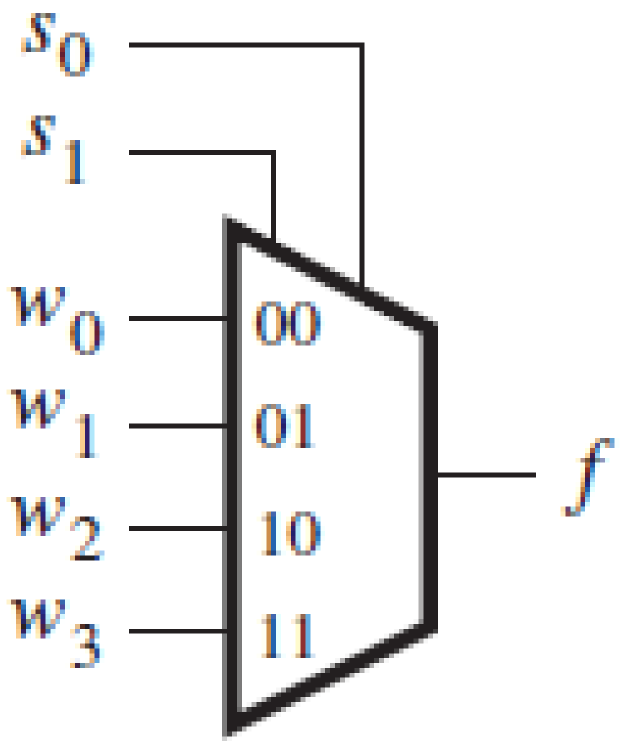
(a) Graphical symbol

s	f
0	w_0
1	w_1

(b) Truth table



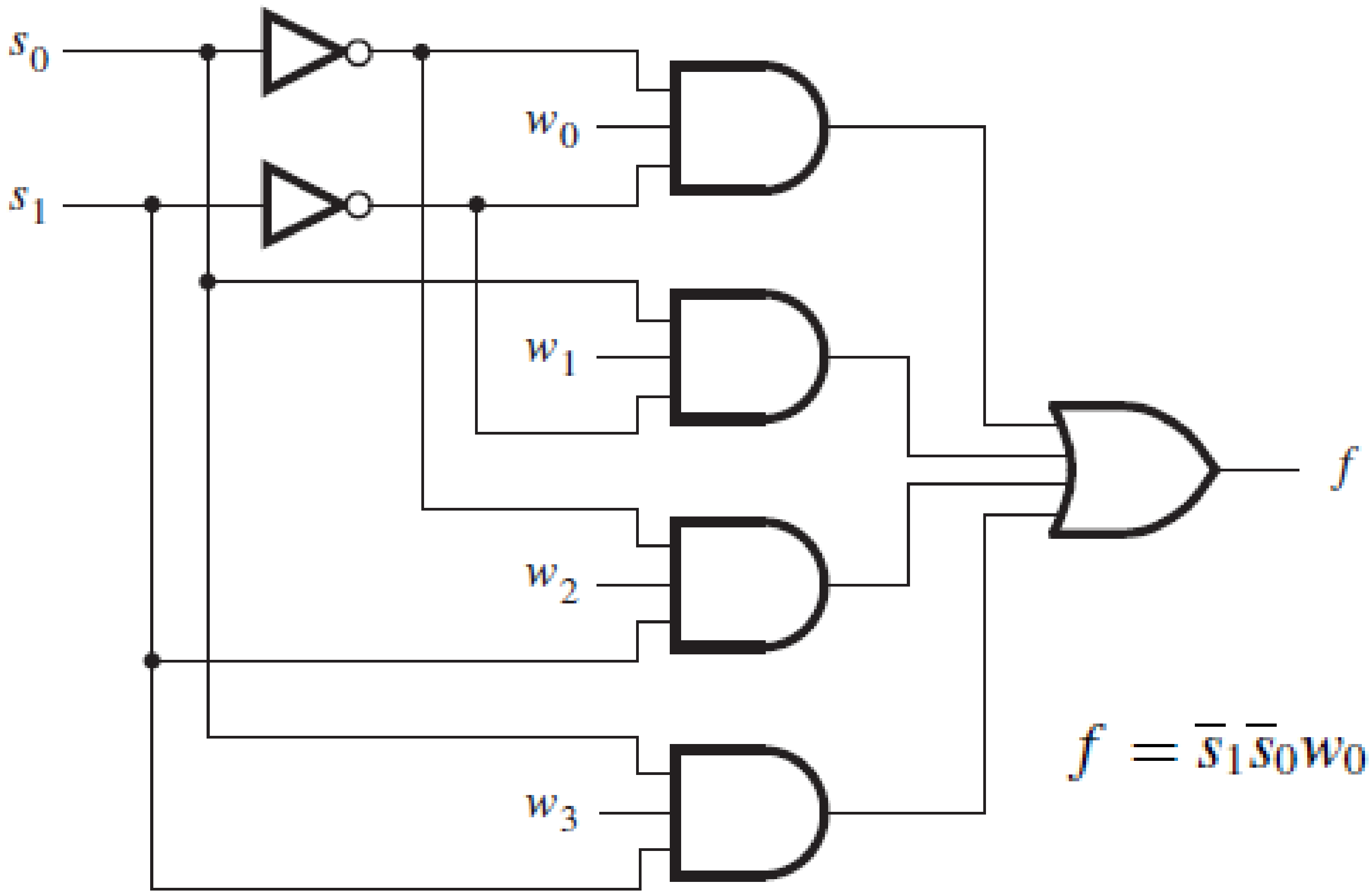
4-to-1 multiplexer.



(a) Graphical 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

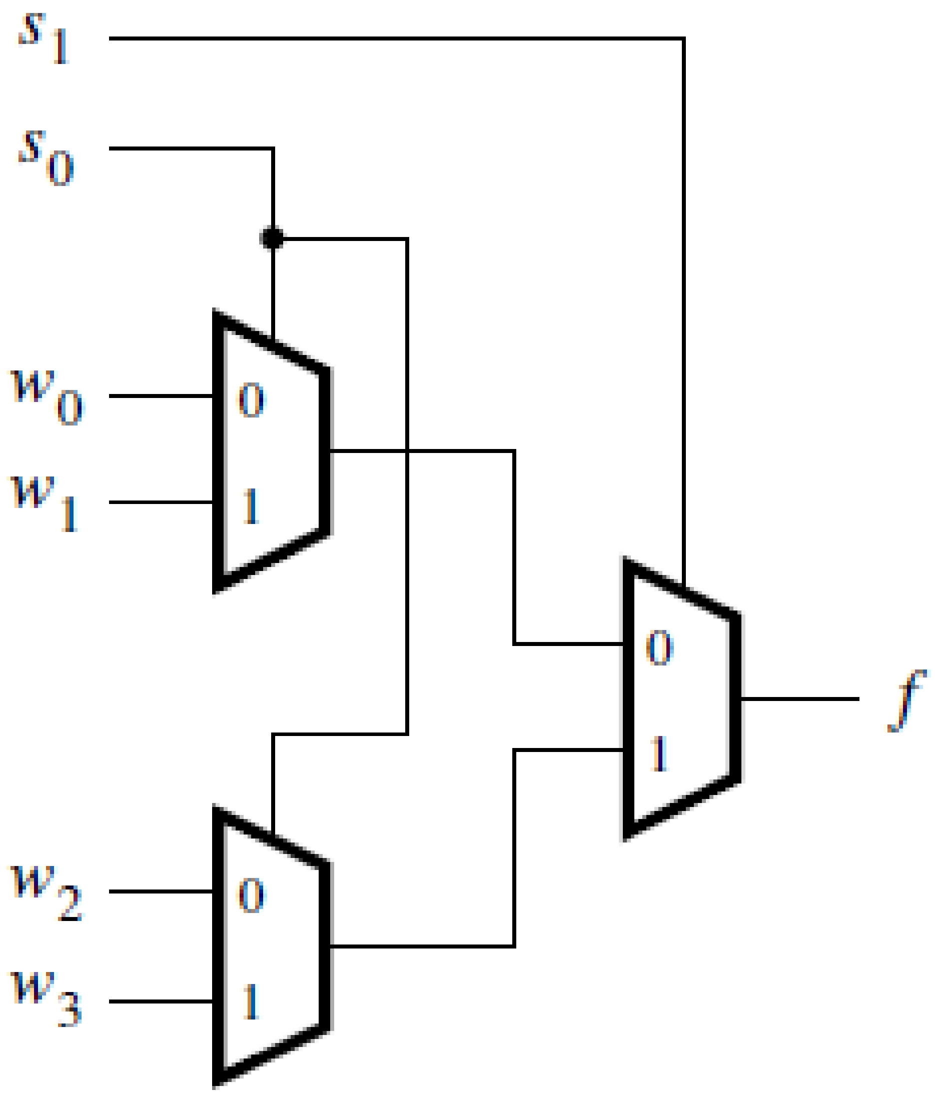
s_1	s_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3

$$f = \overline{s}_1\overline{s}_0 w_0 + \overline{s}_1 s_0 w_1 + s_1\overline{s}_0 w_2 + s_1 s_0 w_3$$

Usually, the number of data inputs, n, is an integer power of two. A multiplexer that has n data inputs, $w0, \ldots, wn-1$, requires $\lceil \log_2 n \rceil$ ct inputs and it has one output.

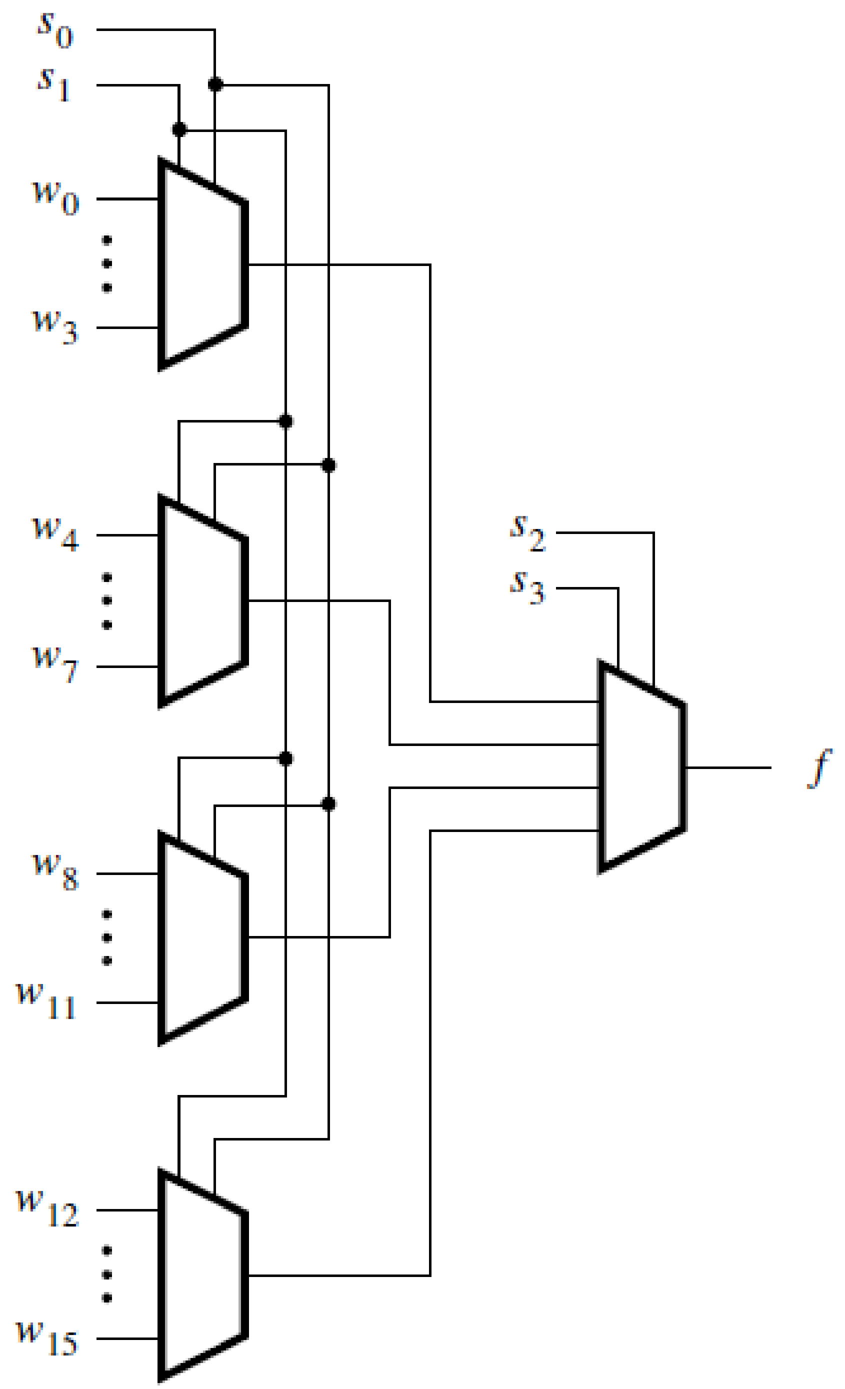
Larger multiplexers can also be constructed from smaller multiplexers. For example, the 4-to-1 multiplexer can be built using three 2-to-1 multiplexers as shown below:

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



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

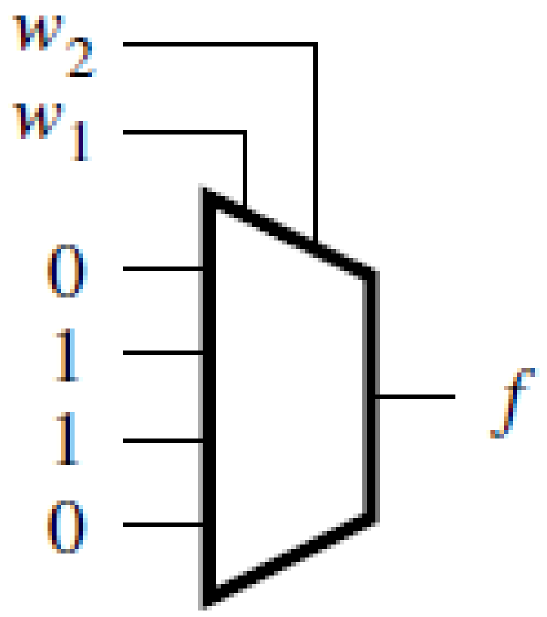
Similarly, 16-to-1 multiplexer can be constructed with five 4-to-1 multiplexers.



Synthesis of Logic Functions Using Multiplexers

Two input XOR function using 4 to 1 multiplexer

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

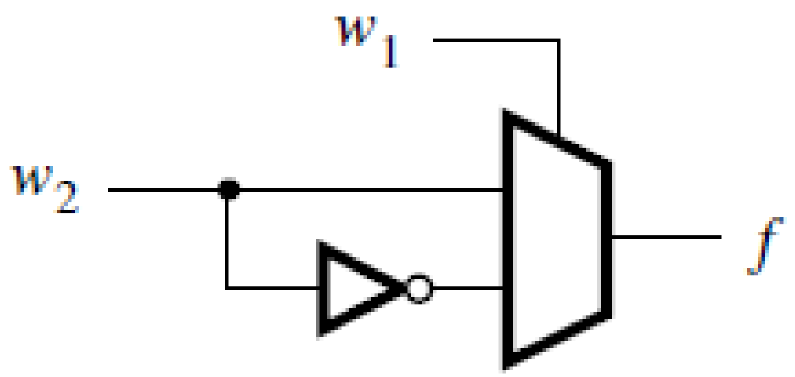


(a) Implementation using a 4-to-1 multiplexer

Example 1 : Two input XOR function using 2 to 1 multiplexer

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

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

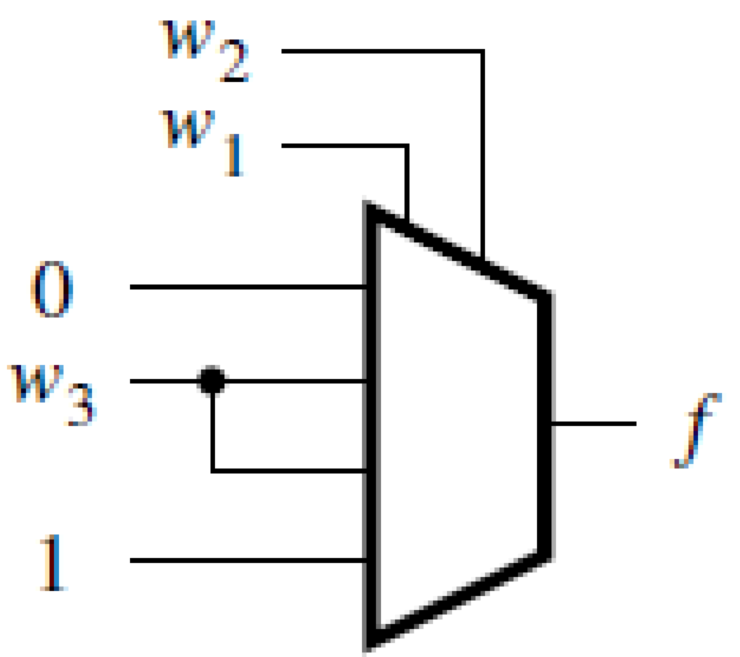


(c) Circuit

Example 2 : Three input majority function using 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



(b) Circuit

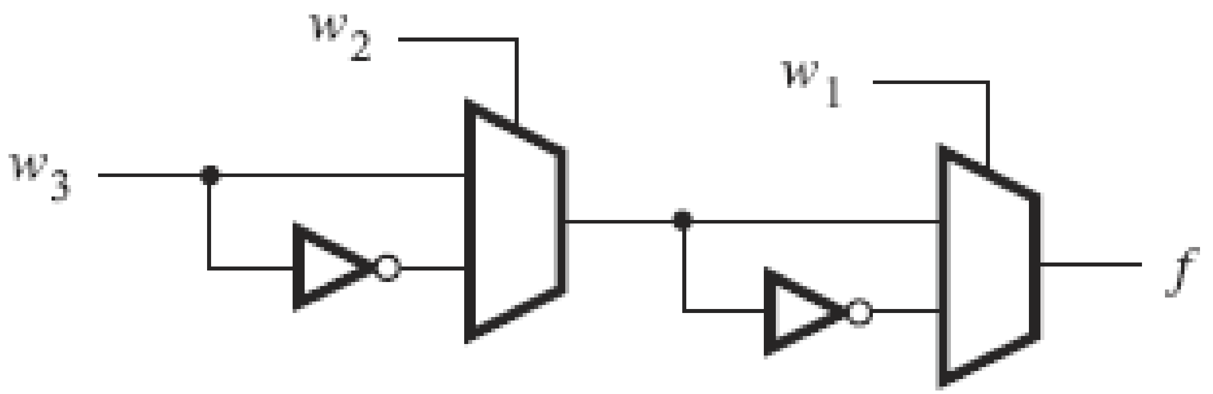
Example 3: Implement a three-input XOR gate using 2-to-1 multiplexer.

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

$\left.\begin{matrix} 0\\1\\1\\0 \end{matrix}\right\} w_2 \oplus w_3$

$\left.\begin{matrix} 1\\0\\0\\1 \end{matrix}\right\} \overline{w_2 \oplus w_3}$

(a) Truth table

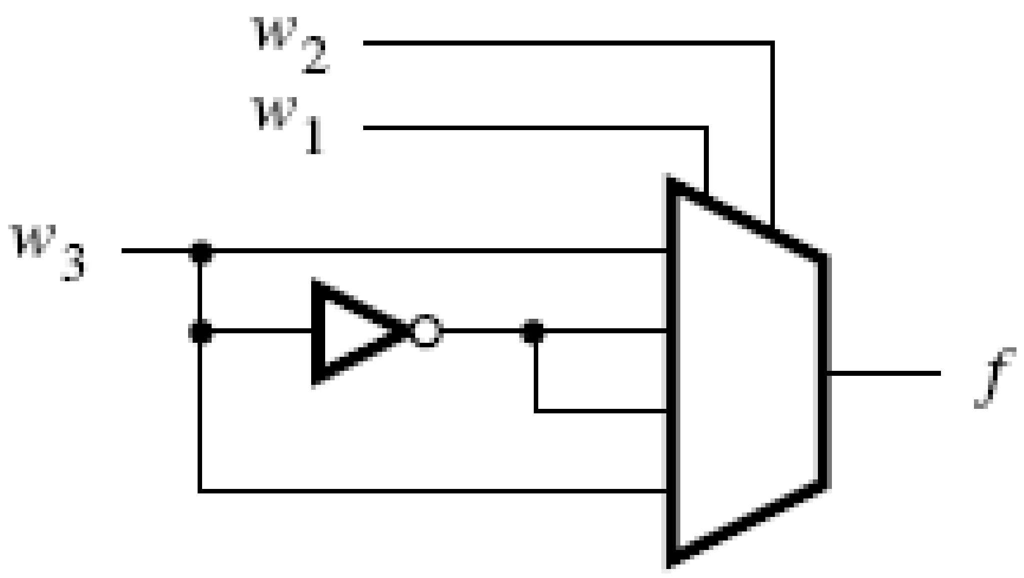


(b) Circuit

Example 4: Implement a three-input XOR gate using 4-to-1 multiplexer.

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

(a) Truth table



(b) Circuit

Example 5 : Three input majority function using 2 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

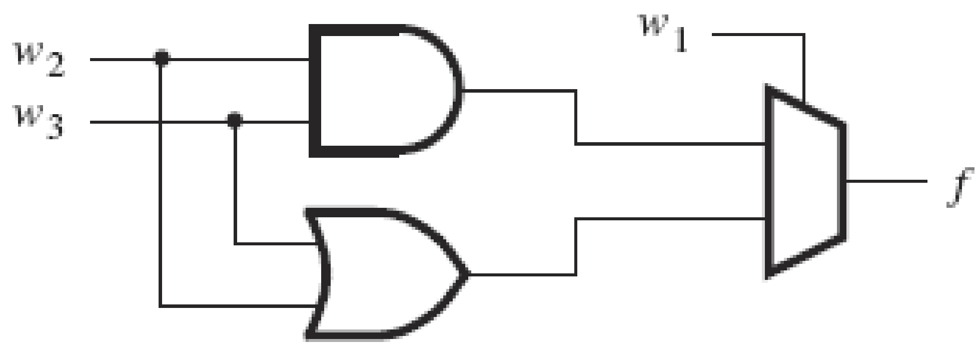
(a) Truth table

-

w_2w_3

w_2+w_3

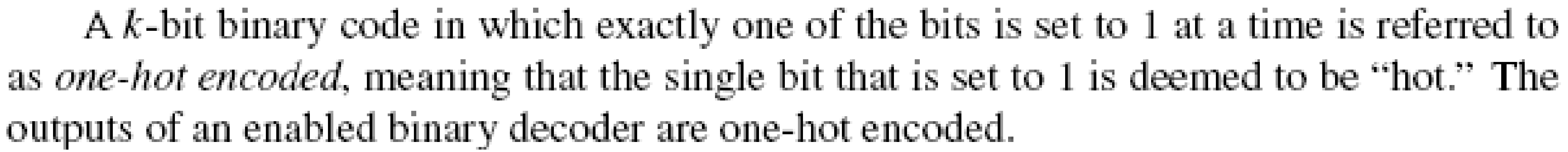
w_1	f
0	w_2w_3
1	$w_2 + w_3$



(b) Circuit

Decoders

- A binary decoder is a logic circuit with *n* inputs and 2^n outputs.
- Only one output is asserted at a time, and each output corresponds to one valuation of the inputs.
- The decoder also has an enable input *En*
- If *En*=1, the valuation of $w_{n-1} \ldots w_1 w_0$ determines which of the outputs is asserted.
- If *En*=0, then none of the decoder outputs is asserted.

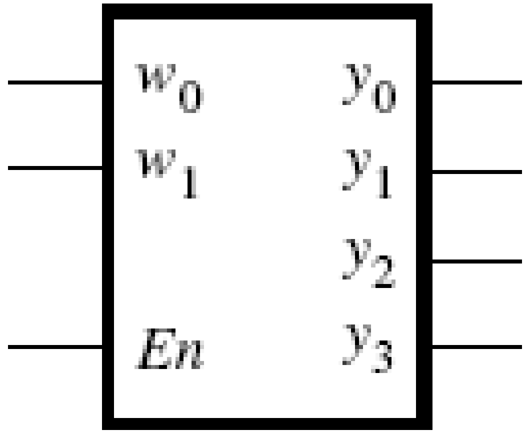


A *k*-bit binary code in which exactly one of the bits is set to 1 at a time is referred to as *one-hot encoded*, meaning that the single bit that is set to 1 is deemed to be "hot." The outputs of an enabled binary decoder are one-hot encoded.

2-to-4 Decoders

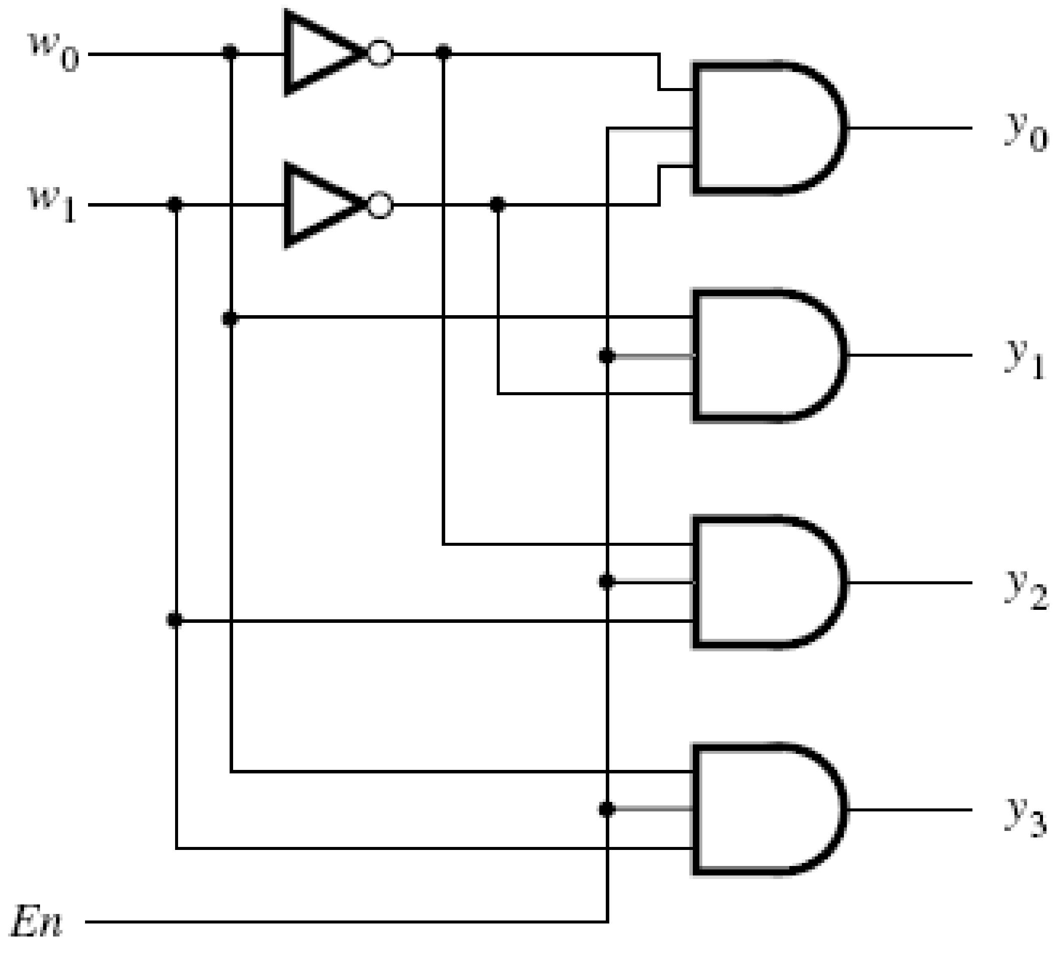
En	w_1	w_0	y_0	y_1	y_2	y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

(a) Truth table



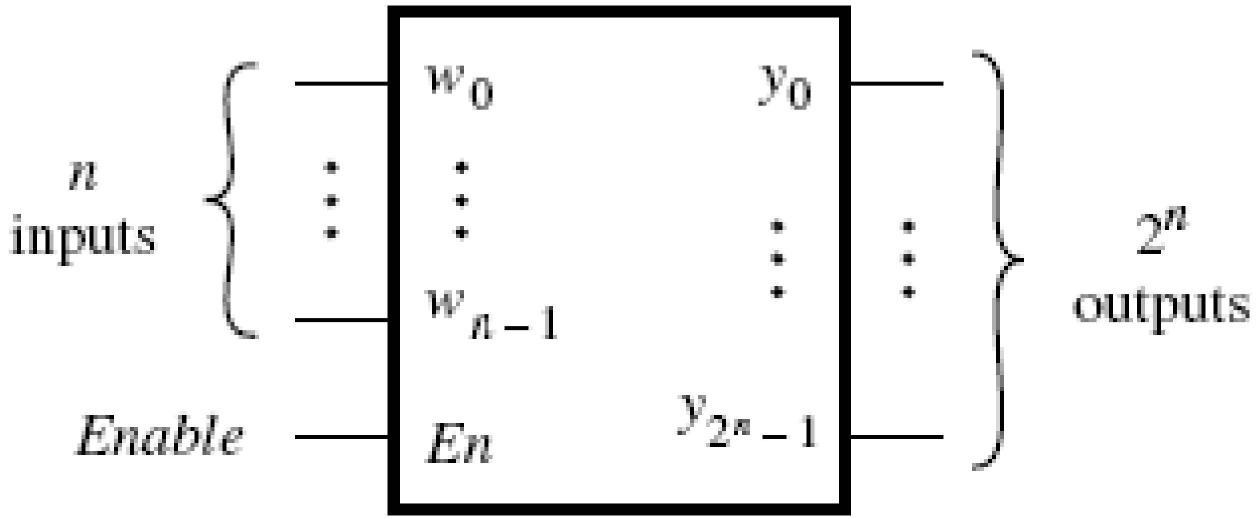
(b) Graphical symbol

2-to-4 Decoders



(c) Logic circuit

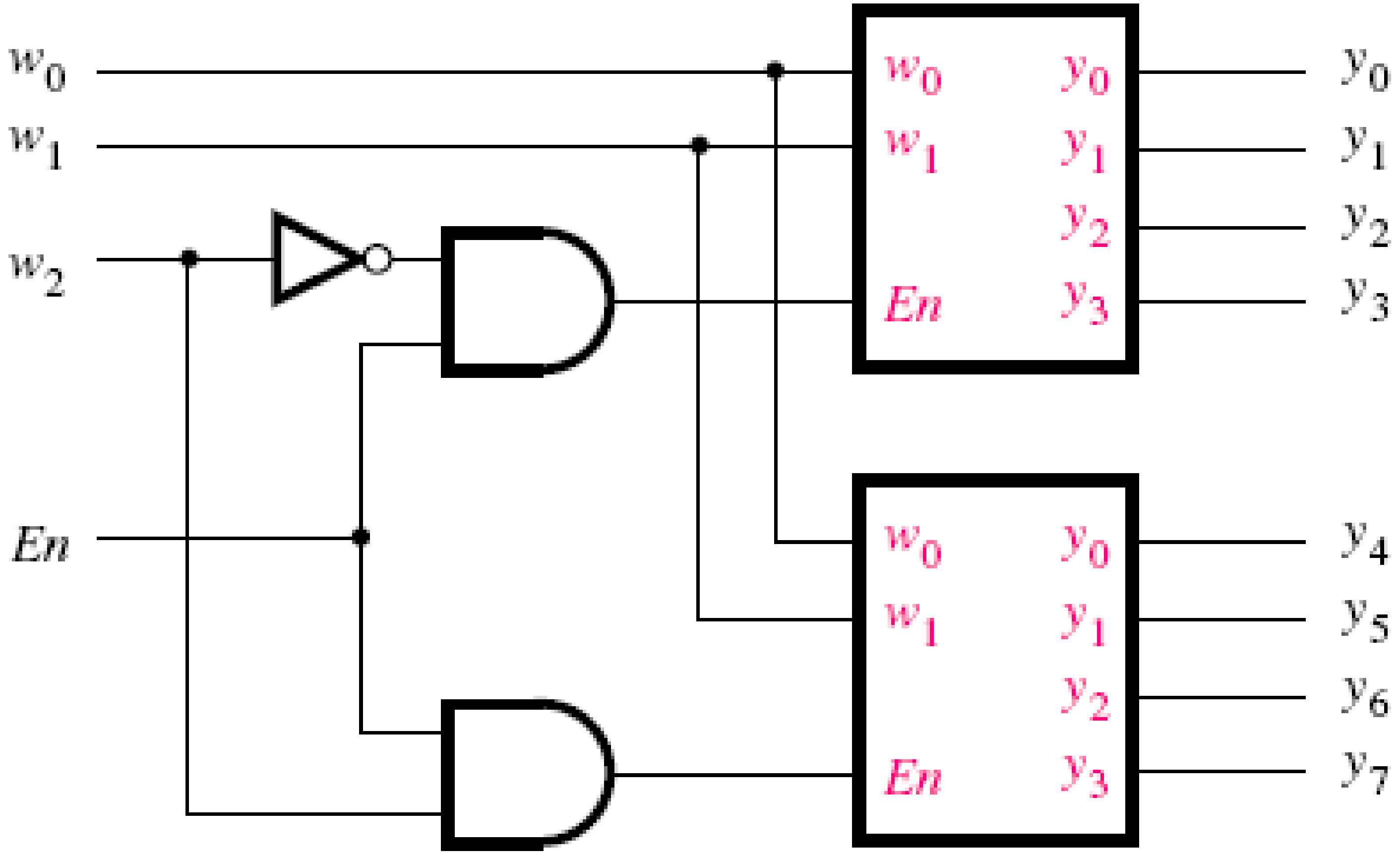
Decoders



(d) An n-to-2^n decoder

Decoders

Larger decoders can be built using the sum-of-products structure or else they can be constructed from smaller decoders.

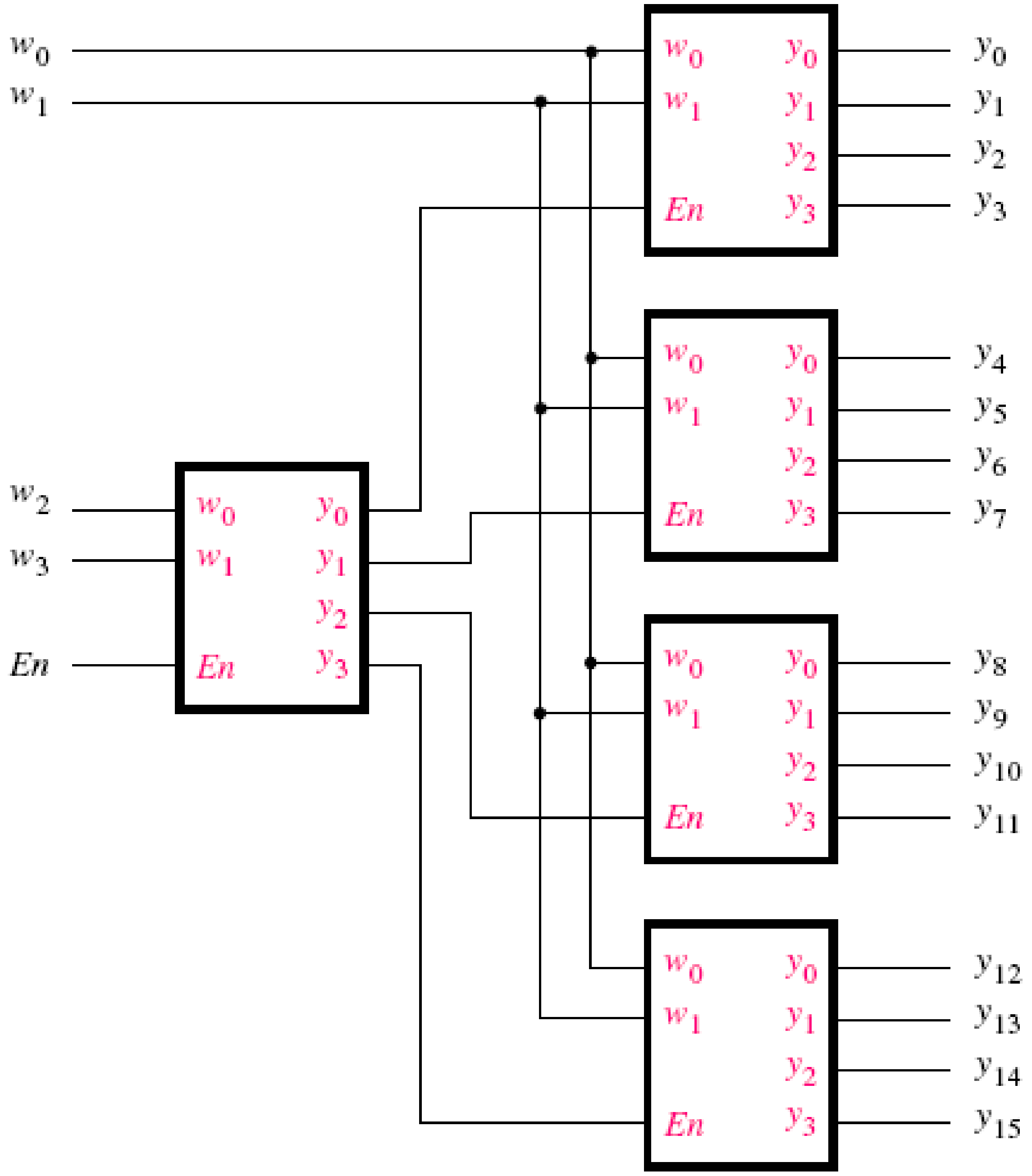


A 3-to-8 decoder using two 2-to-4 decoders.

The w2 input drives the enable input of the two decoders.

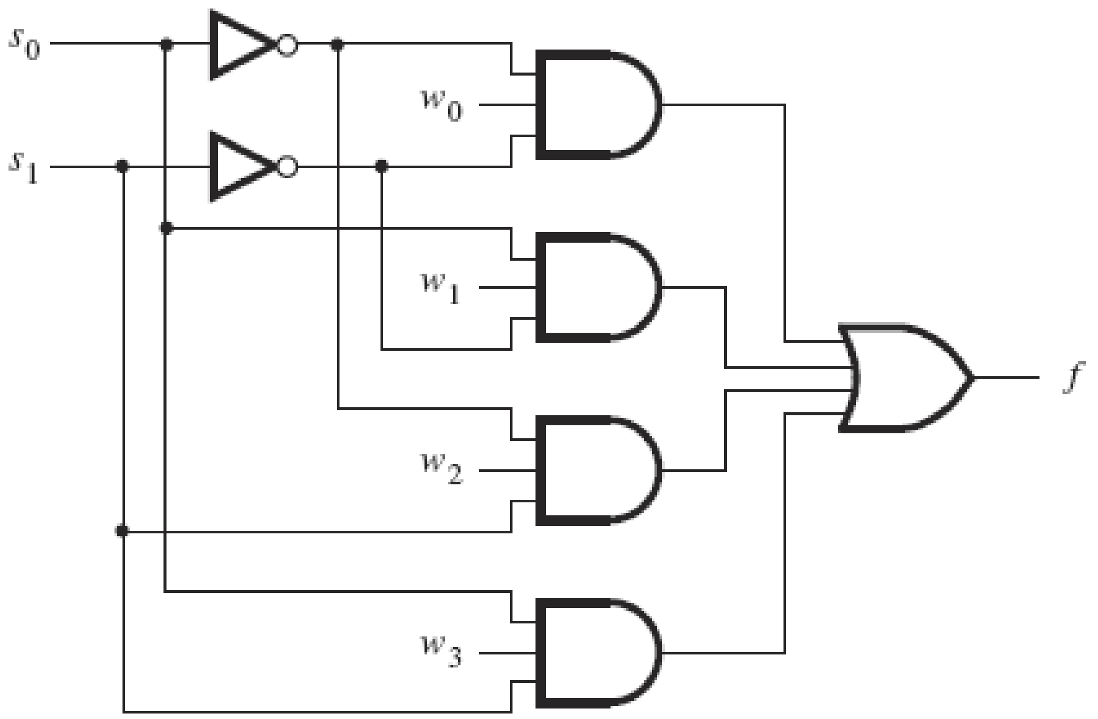
4-to-16 Decoder

This type of circuit is referred to as a decoder tree.

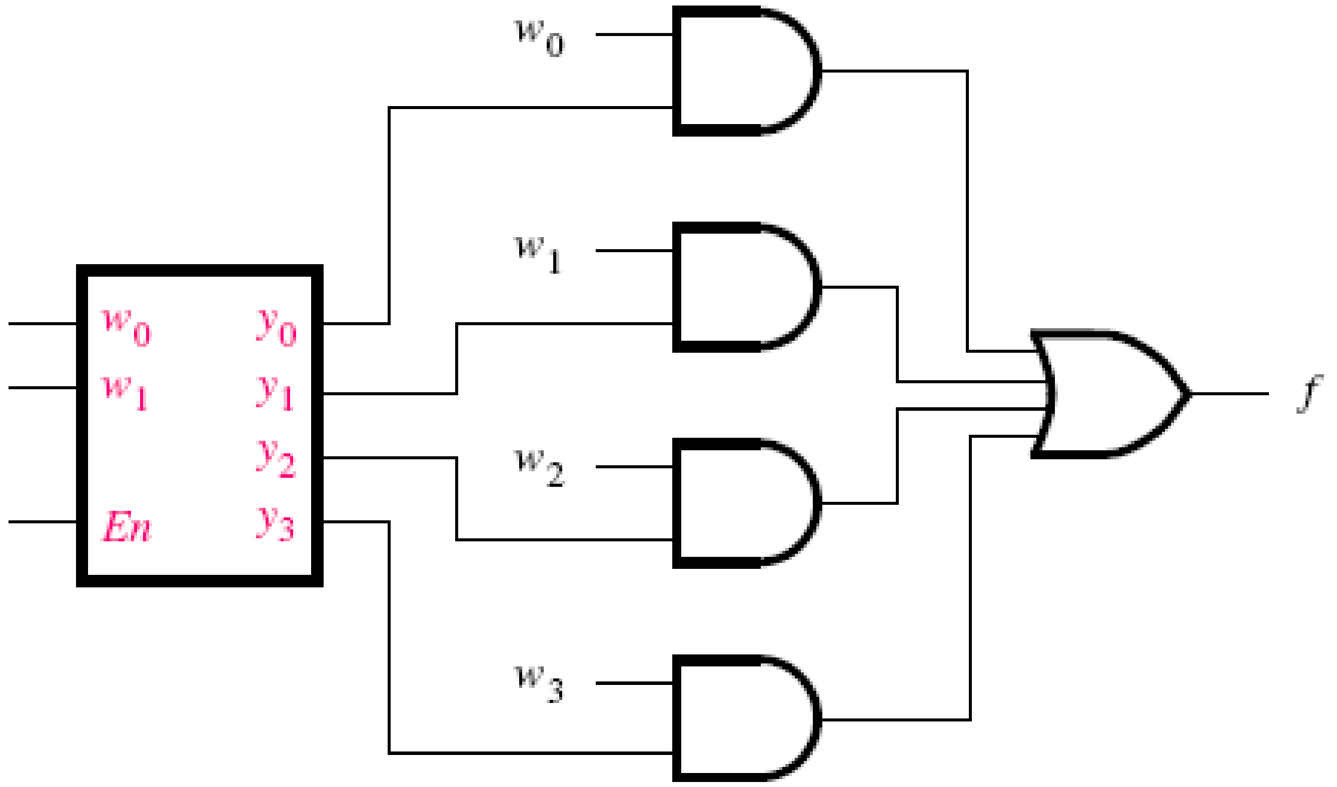


w_0
w_1
w_2
w_3
En
w_0 w_1 En y_0 y_1 y_2 y_3
w_0 w_1 En y_0 y_1 y_2 y_3
w_0 w_1 En y_0 y_1 y_2 y_3
w_0 w_1 En y_0 y_1 y_2 y_3
w_0 w_1 En y_0 y_1 y_2 y_3
y_0
y_1
y_2
y_3
y_4
y_5
y_6
y_7
y_8
y_9
y_{10}
y_{11}
y_{12}
y_{13}
y_{14}
y_{15}

4-to-1 Multiplexer using Decoder



4-to-1 Multiplexer



4-to-1 Multiplexer using decoder

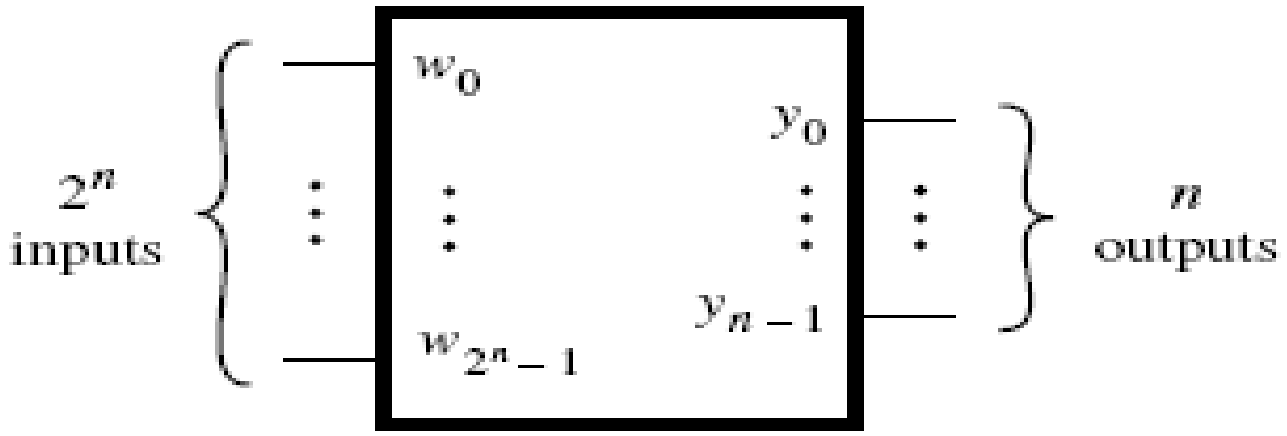
Encoders

An encoder performs the opposite function of a decoder.

Binary Encoders

A *binary encoder* encodes information from 2^n inputs into an n-bit code.

Exactly one of the input signals should have a value of 1, and the outputs present the binary number that identifies which input is equal to 1.

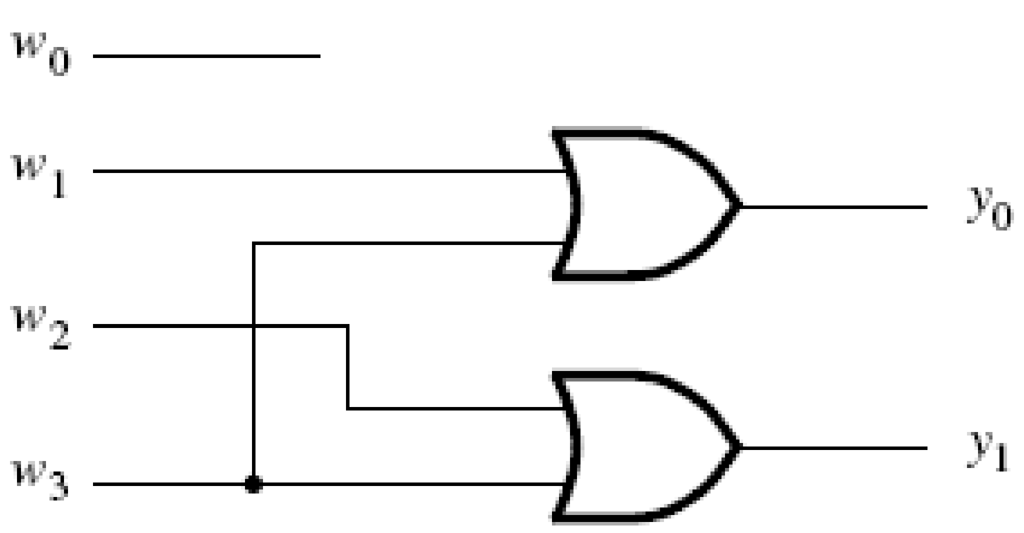


A 2^n-to-n binary encoder.

4-to-2 Binary Encoders

w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

(a) Truth table



(b) Circuit

The inputs are one-hot encoded. The input patterns that have multiple inputs set to 1 are treated as don’t care conditions.
Encoders are used for transmitting information in a digital system.

Priority Encoders

In a priority encoder each input has a priority level associated with it.
The encoder outputs indicate the active input that has the highest priority.
When an input with high priority is asserted, the inputs with lower priority are ignored.

w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

$$i_0 = \overline{w}_3\overline{w}_2\overline{w}_1 w_0 \qquad y_0 = i_1 + i_3$$

$$i_1 = \overline{w}_3\overline{w}_2 w_1 \qquad y_1 = i_2 + i_3$$

$$i_2 = \overline{w}_3 w_2$$

$$i_3 = w_3$$

$$z = i_0 + i_1 + i_2 + i_3$$

Z is set to 1 when at least one of the inputs is equal to 1.

w_0 has the lowest priority and w_3 the highest

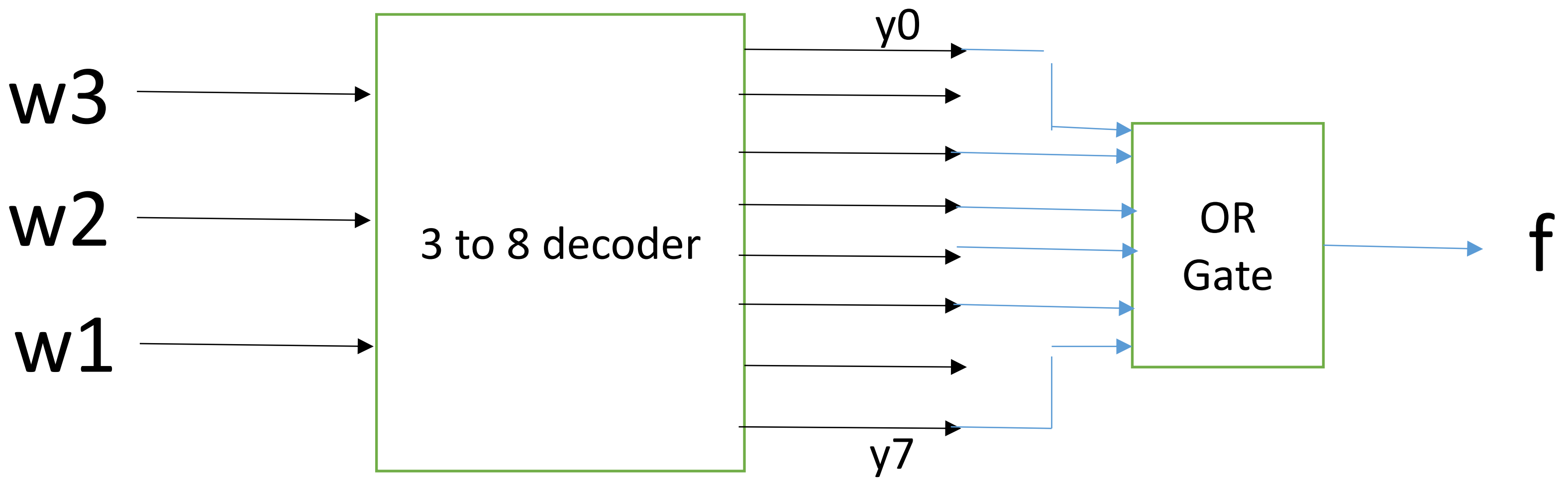
Priority encoder having w0 highest priority and w3 least priority

w3	w2	w1	w0	y1	y0	z
x	x	x	1	0	0	1
x	x	1	0	0	1	1
x	1	0	0	1	0	1
1	0	0	0	1	1	1
0	0	0	0	x	x	0

Problems:

1. Show how the function $f(w1, w2, w3) = \sum m(0, 2, 3, 4, 5, 7)$ can be implemented using a 3-to-8 binary decoder and an OR gate.

2. Show how the function $f(w1, w2, w3) = \sum m(1, 2, 3, 5, 6)$ can be implemented using a 3-to-8 binary decoder and an OR gate.


w3
w2
w1
3 to 8 decoder
y0
y7
OR Gate
f

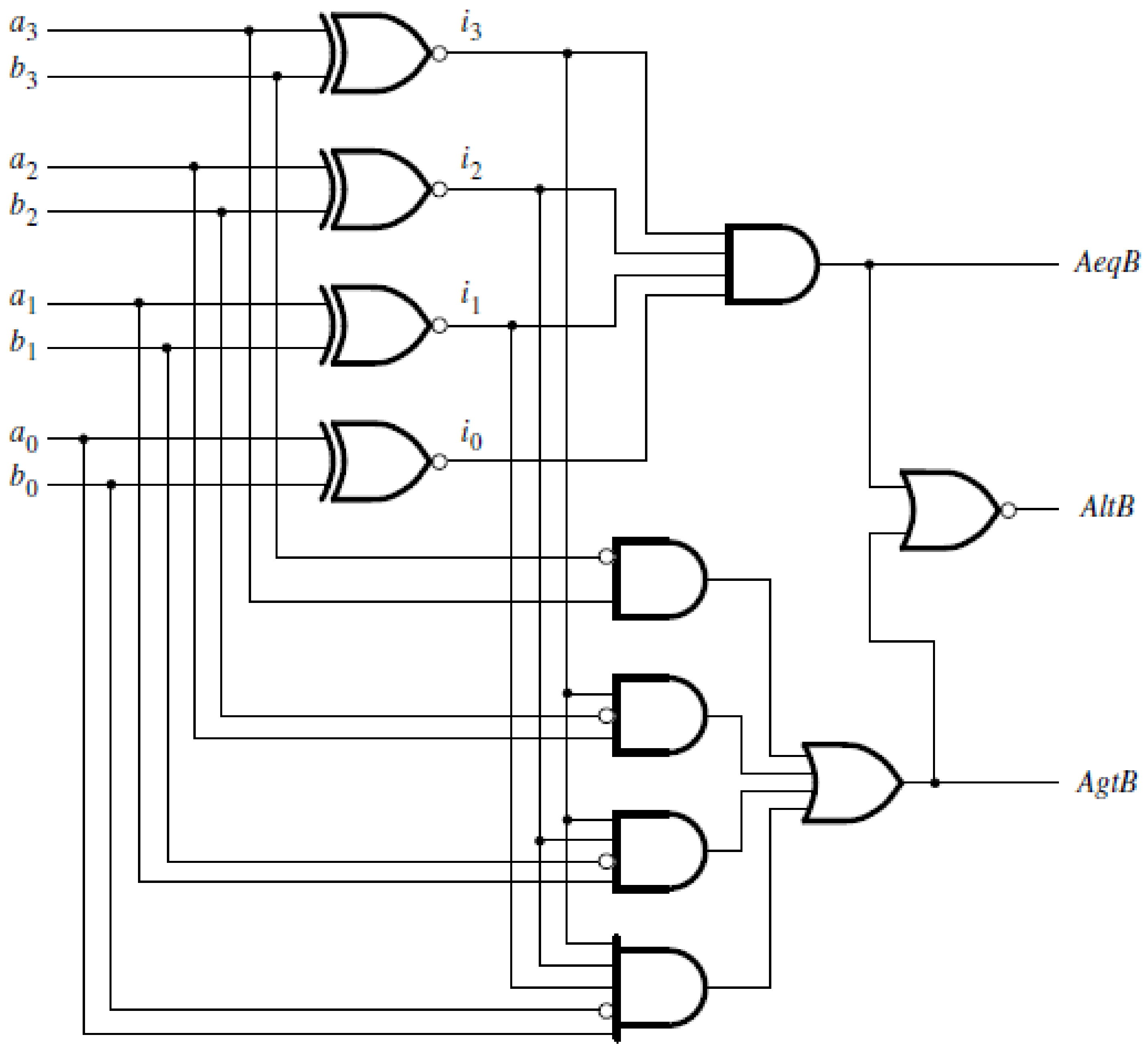
Arithmetic Comparison Circuits (Unsigned)

Let *A* = *a*3*a*2*a*1*a*0 and *B* = *b*3*b*2*b*1*b*0. Define a set of intermediate signals called *i*3, *i*2, *i*1, and *i*0. Each signal, *ik* , is 1 if the bits of *A* and *B* with the same index are equal.

That is, *ik* = (*ak* ⊕ *bk*)' . The comparator's *AeqB* output is then given by *AeqB* = *i*3*i*2*i*1*i*0

$$AgtB = a_3\overline{b}_3 + i_3a_2\overline{b}_2 + i_3i_2a_1\overline{b}_1 + i_3i_2i_1a_0\overline{b}_0$$

$$AltB = \overline{AeqB + AgtB}$$


a_3
b_3
a_2
b_2
a_1
b_1
a_0
b_0
i_3
i_2
i_1
i_0
AeqB
AltB
AgtB

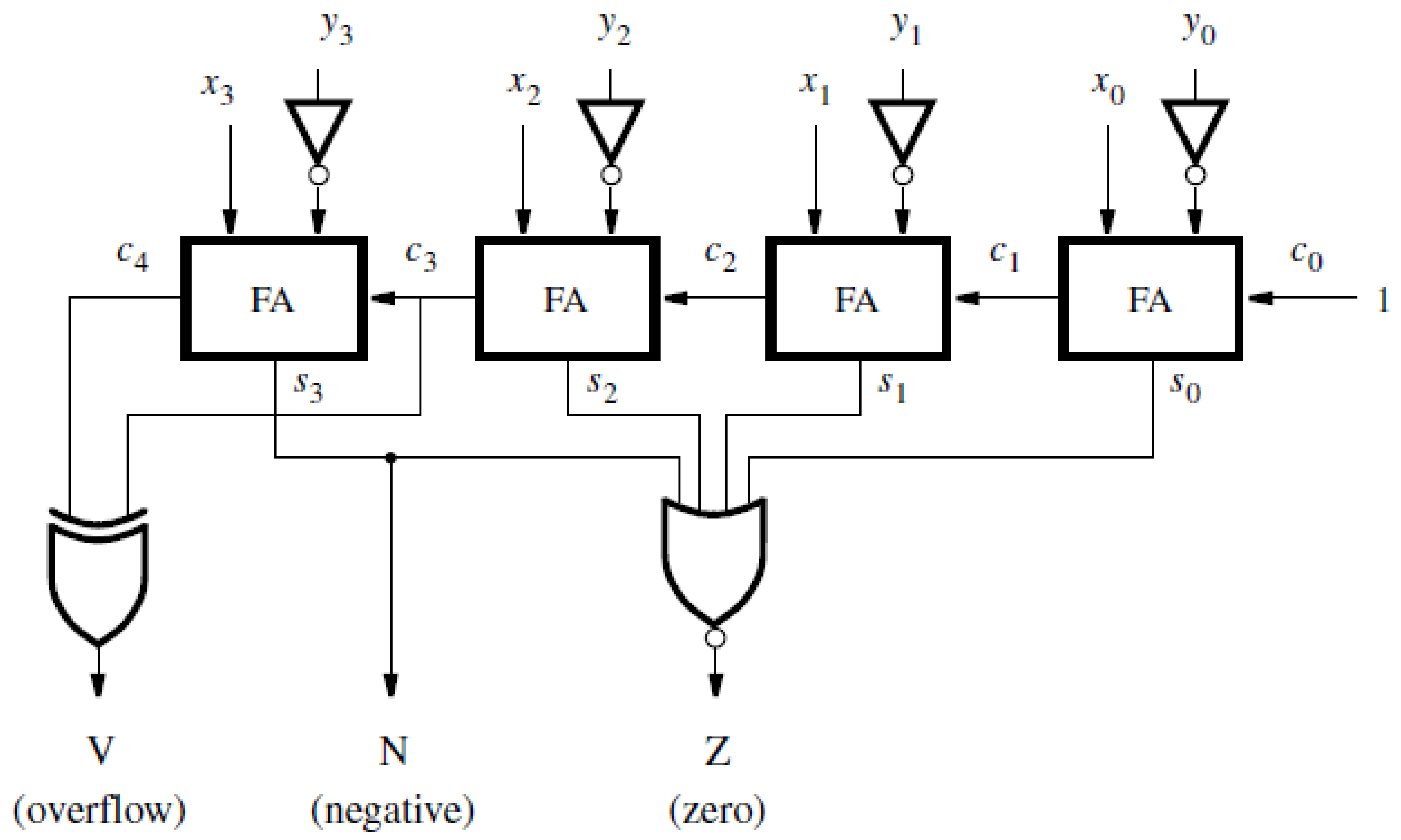
Arithmetic Comparison Circuits (signed)

Example

Problem: In computer computations it is often necessary to compare numbers. Two four-bit signed numbers, $X = x3x2x1x0$ and $Y = y3y2y1y0$, can be compared by using the subtractor circuit. The three outputs denote the following:

- $Z = 1$ if the result is 0; otherwise $Z = 0$
- $N = 1$ if the result is negative; otherwise $N = 0$
- $V = 1$ if arithmetic overflow occurs; otherwise $V = 0$

Show how Z, N, and V can be used to determine the cases $X = Y$, $X < Y$, $X \leq Y$, $X > Y$, and $X \geq Y$.


y_3
y_2
y_1
y_0
x_3
x_2
x_1
x_0
c_4
c_3
c_2
c_1
c_0
FA
FA
FA
FA
1
s_3
s_2
s_1
s_0
V
(overflow)
N
(negative)
Z
(zero)

Case 1: $X < Y$

•**Same sign**:

If X and Y have the same sign and $X < Y$, subtraction ($X - Y$) will not overflow.

$\rightarrow V = 0$ and the result is negative ($N = 1$).

•**Different signs (X negative, Y positive)**:

If $X < Y$ and they have different signs, the result is normally negative.

But, if overflow occurs (V = 1), the result's sign flips and N = 0.

•This leads to the rule:

$$X < Y \; if N \oplus V = 1$$

3. Case 2: $X = Y$

•This is straightforward:

$$X = Y \; if Z = 1$$

4. Case 3: $X \leq Y$

•If $X \leq Y$, then either:

- $X = Y \rightarrow Z = 1$
- $X < Y \rightarrow N \oplus V = 1$

•Combined:

$$X \leq Y \; if Z + (N \oplus V) = 1$$

5. Case 4: $X > Y$

•This is simply the **NOT** of $X \leq Y$:

$$X > Y \ if\left(Z + (N \oplus V)\right)' = 1$$

6. Case 5: $X \geq Y$

•This is the **NOT** of $X < Y$:

$$X \geq Y \ if\left(N \oplus V\right)' = 1$$

Consider first the case $X < Y$, where the following possibilities may arise:

- If X and Y have the same sign there will be no overflow, hence $V = 0$. Then for both positive and negative X and Y the difference will be negative ($N = 1$).
- If X is negative and Y is positive, the difference will be negative ($N = 1$) if there is no overflow ($V = 0$); but the result will be positive ($N = 0$) if there is overflow ($V = 1$).

Therefore, if $X < Y$ then $N \oplus V = 1$.

The case $X = Y$ is detected by $Z = 1$. Then, $X \leq Y$ is detected by $Z + (N \oplus V) = 1$.

$X > Y$ if $(Z + (N \oplus V))' = 1$

$X \geq Y$ if $(N \oplus V)' = 1$.

Z	N	V	$N \oplus V$	$Z \vee (N \oplus V)$	Interpretation
0	0	0	0	0	$X > Y$
0	0	1	1	1	$X < Y$
0	1	0	1	1	$X < Y$
0	1	1	0	0	$X > Y$
1	0	0	0	1	$X = Y$

- When $Z = 1$ (result is zero) the only valid flag values are $N = 0$ and $V = 0$. The other three rows with $Z = 1$ are impossible for two's-complement subtraction, so they are marked **Impossible**.
- The table shows:
  - $X < Y$ when $N \oplus V = 1$.
  - $X = Y$ when $Z = 1$.
  - $X \leq Y$ when $Z \vee (N \oplus V) = 1$.
  - $X > Y$ when $\left(Z \vee (N \oplus V)\right)' = 1$.
  - $X \geq Y$ when $(N \oplus V)' = 1$.