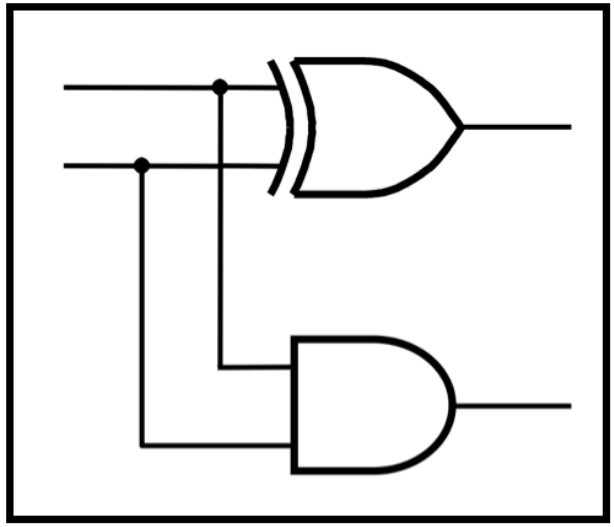


CprE 281: Digital Logic

Instructor: Alexander Stoytchev

http://www.ece.iastate.edu/~alexs/classes/

Code Converters

CprE 281: Digital Logic
Iowa State University, Ames, IA

Administrative Stuff

- **HW 7 is out**
- **It is due on Monday (Oct 18) @ 4pm**
- **We will start with Chapter 5 on Friday.**

Quick Review

Decoders

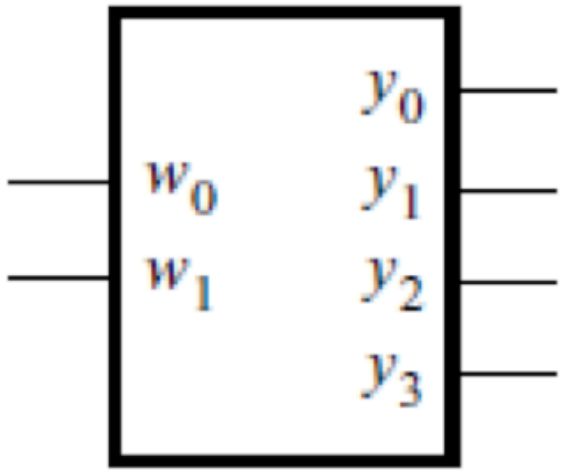
2-to-4 Decoder (Definition)

- Has two inputs: w_1 and w_0
- Has four outputs: y_0 , y_1 , y_2 , and y_3
- If w_1=0 and w_0=0, then the output y_0 is set to 1
- If w_1=0 and w_0=1, then the output y_1 is set to 1
- If w_1=1 and w_0=0, then the output y_2 is set to 1
- If w_1=1 and w_0=1, then the output y_3 is set to 1
- Only one output is set to 1. All others are set to 0.

Truth Table and Graphical Symbol for a 2-to-4 Decoder

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

(a) Truth table



(b) Graphical symbol

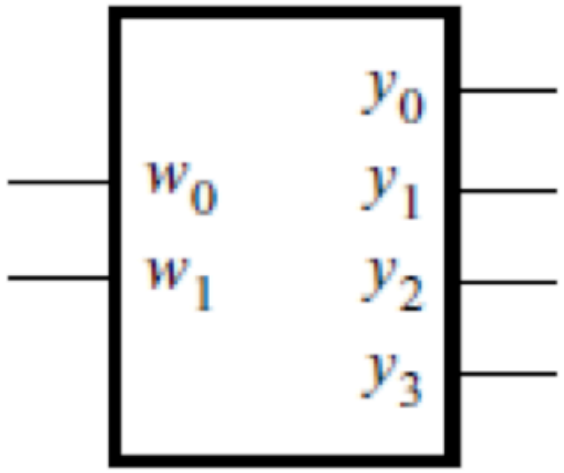
[Figure 4.13a-b from the textbook]

Truth Table and Graphical Symbol for a 2-to-4 Decoder

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

The outputs are "one-hot" encoded

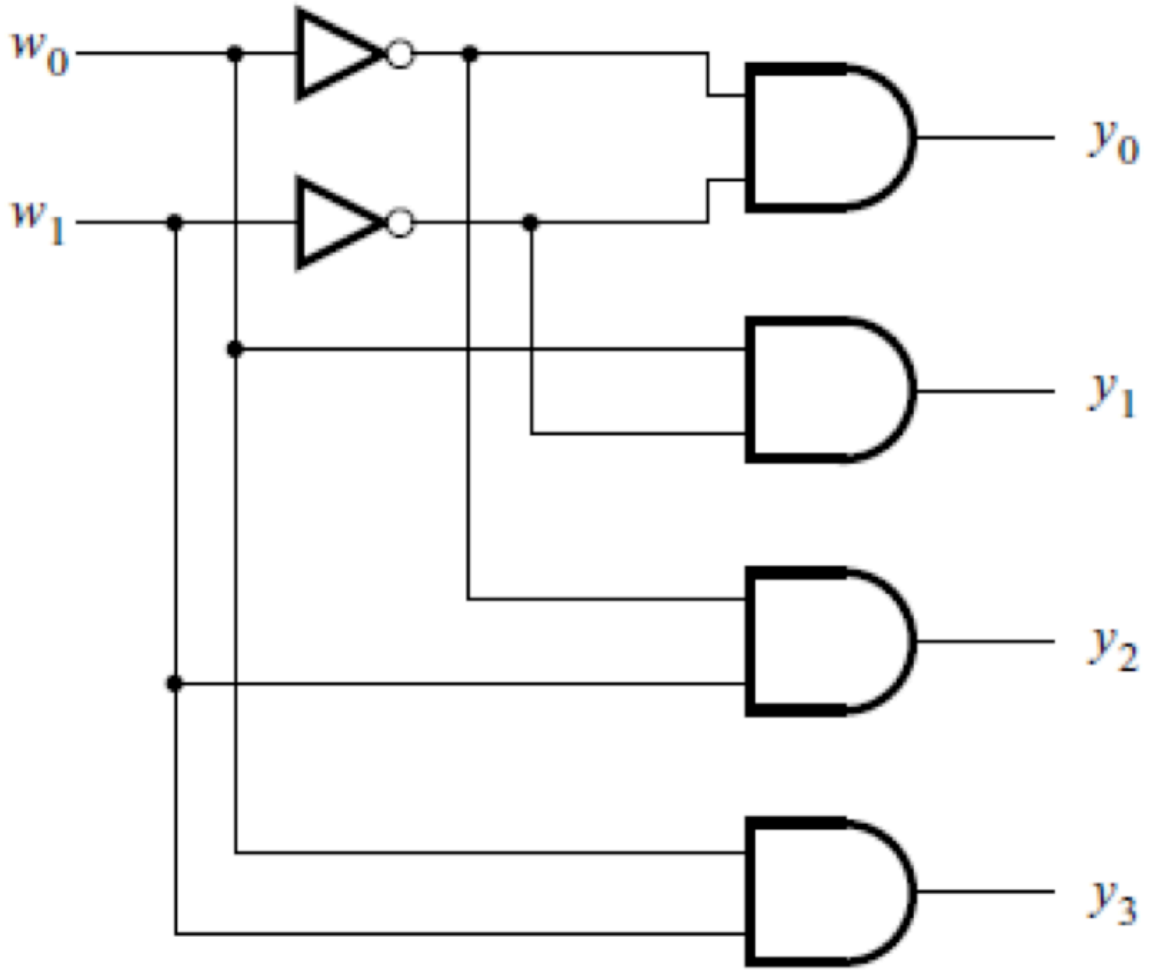
(a) Truth table



(b) Graphical symbol

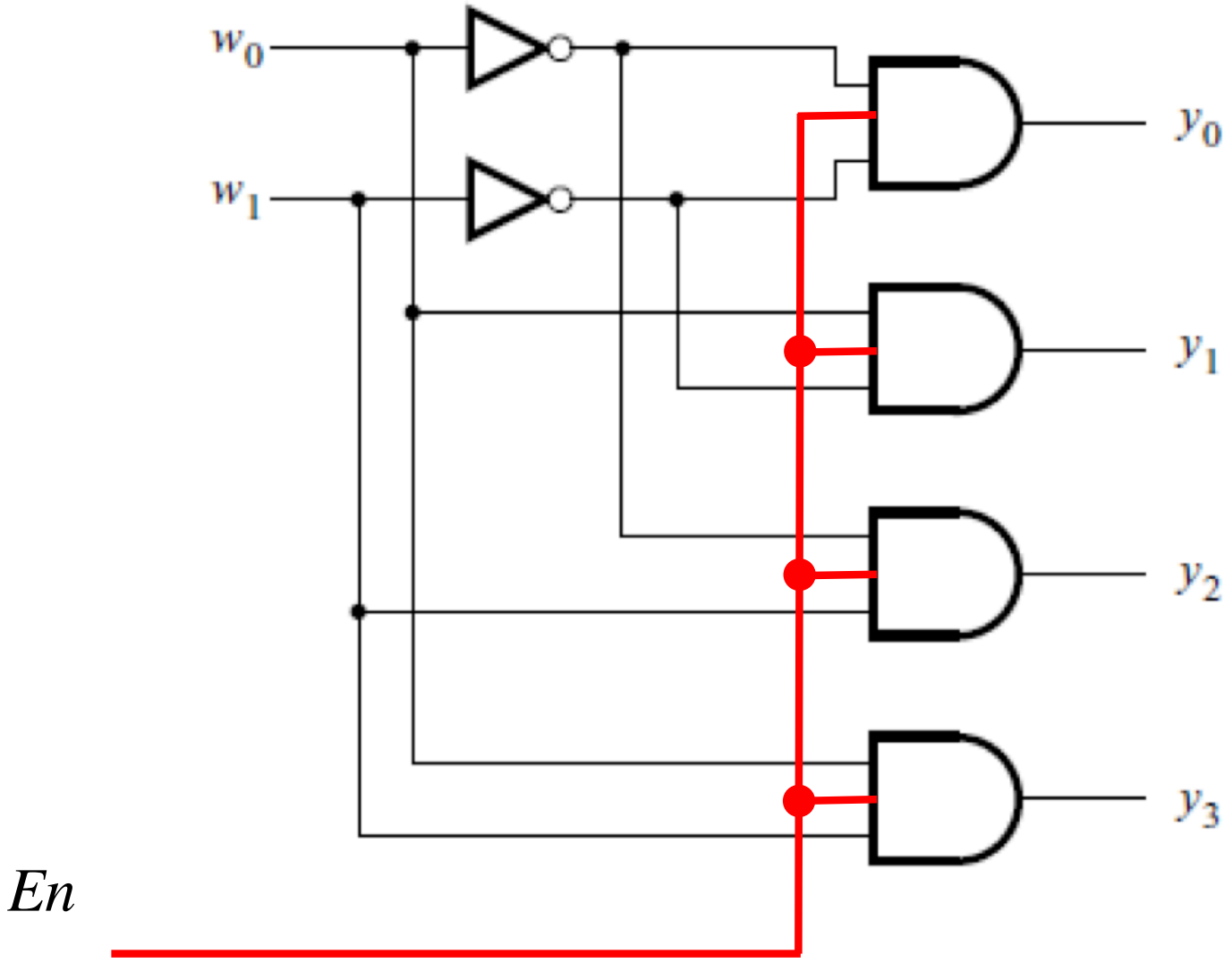
[Figure 4.13a-b from the textbook]

Truth Logic Circuit for a 2-to-4 Decoder



[Figure 4.13c from the textbook]

Adding an Enable Input

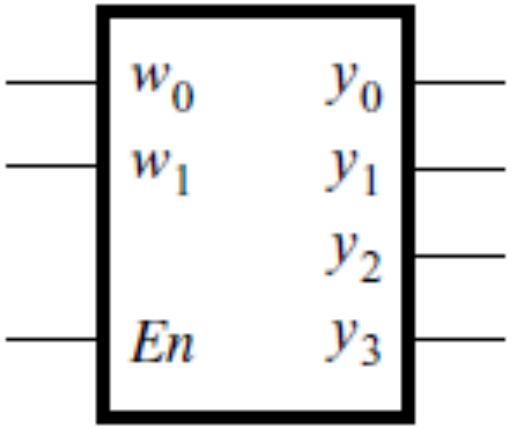


[Figure 4.13c from the textbook]

Truth Table and Graphical Symbol for a 2-to-4 Decoder with an Enable Input

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



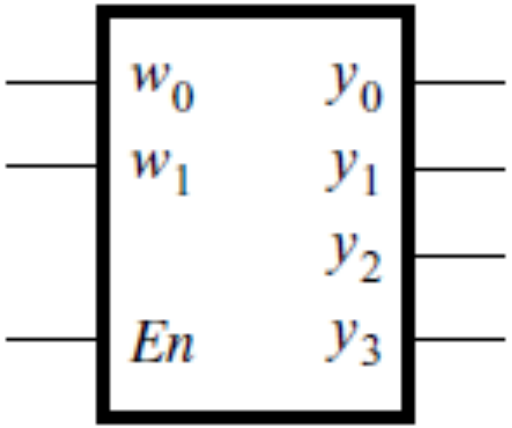
(b) Graphical symbol

[Figure 4.14a-b from the textbook]

Truth Table and Graphical Symbol for a 2-to-4 Decoder with an Enable Input

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

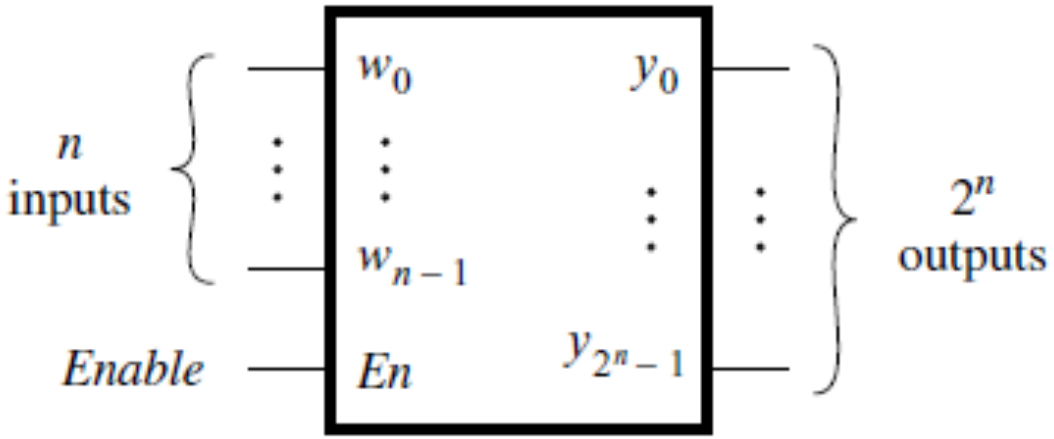


(b) Graphical symbol

x indicates that it does not matter what the value of these variable is for this row of the truth table

[Figure 4.14a-b from the textbook]

Graphical Symbol for a Binary n-to-2^n Decoder with an Enable Input



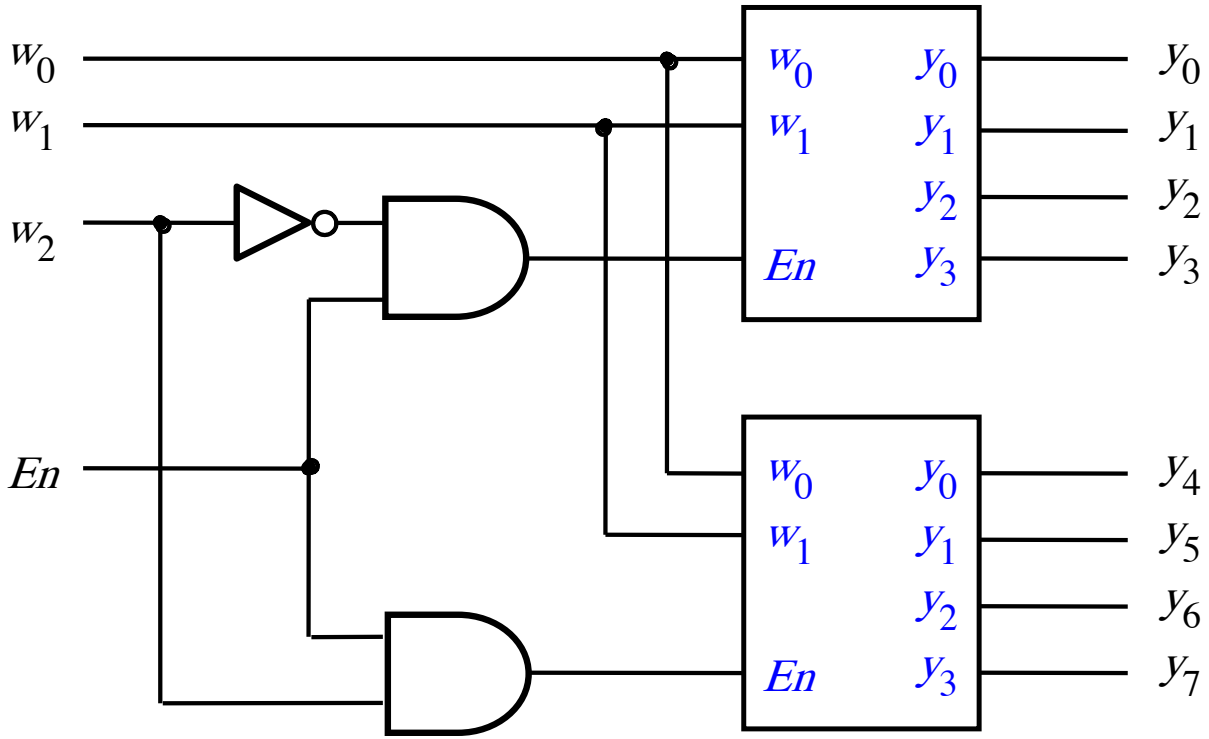
(d) An n-to-2^n decoder

A binary decoder with n inputs has 2^n outputs

The outputs of an enabled binary decoder are "one-hot" encoded, meaning that only a single bit is set to 1, i.e., it is *hot*.

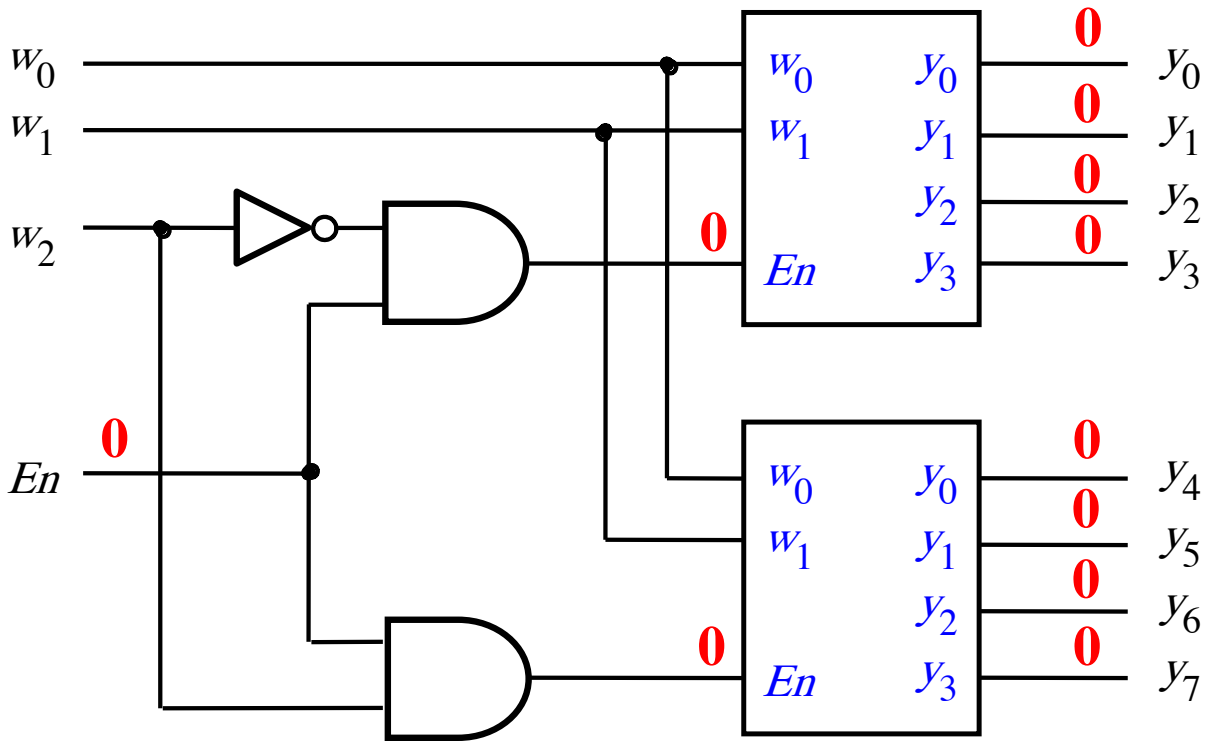
[Figure 4.14d from the textbook]

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



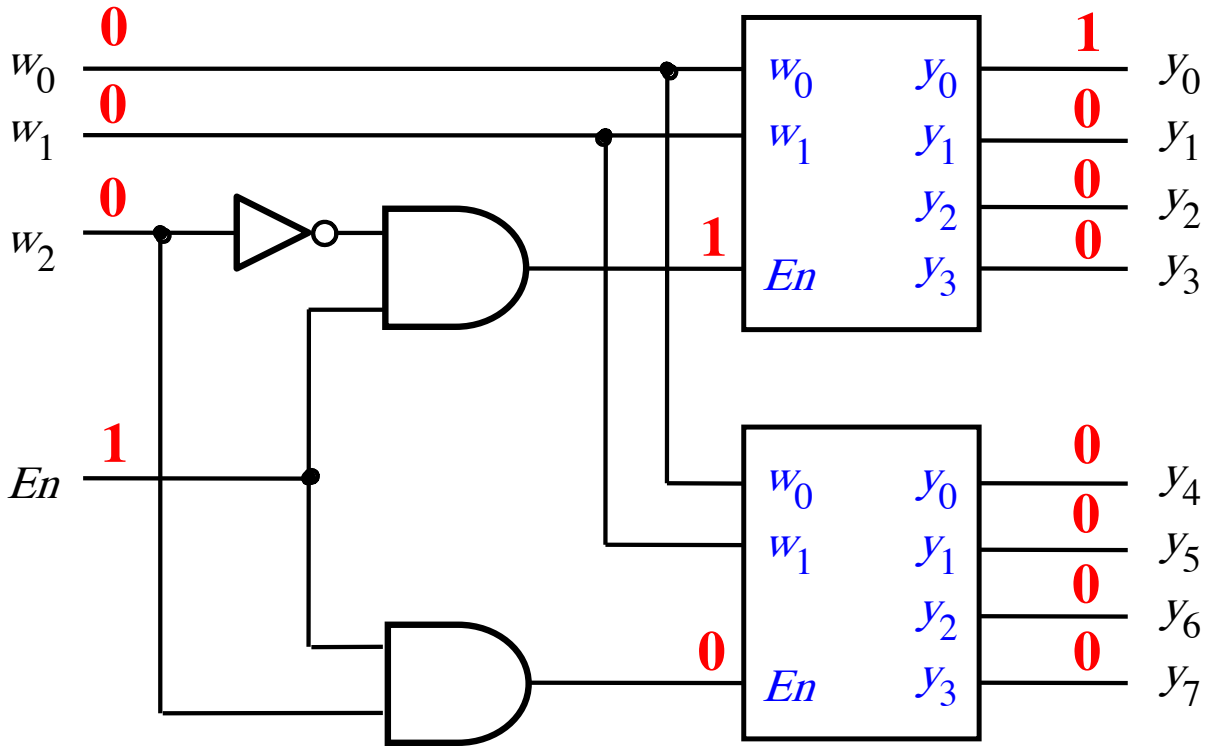
[Figure 4.15 from the textbook]

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



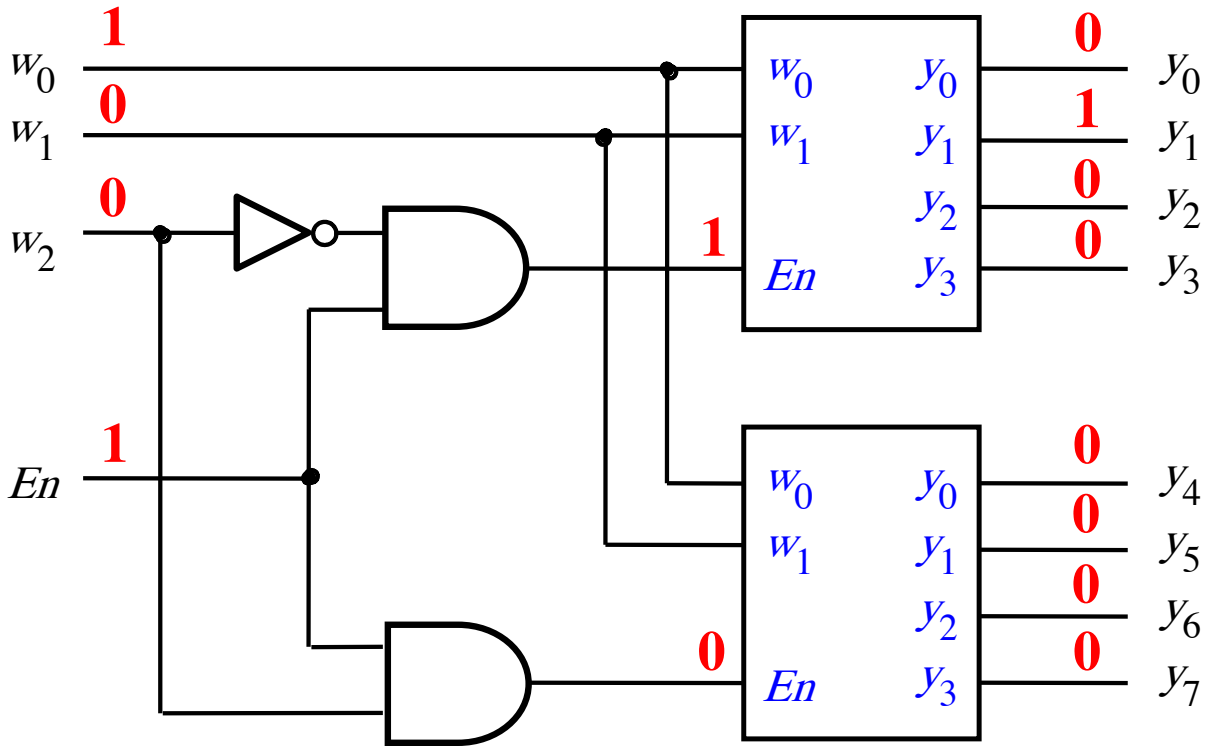
[Figure 4.15 from the textbook]

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



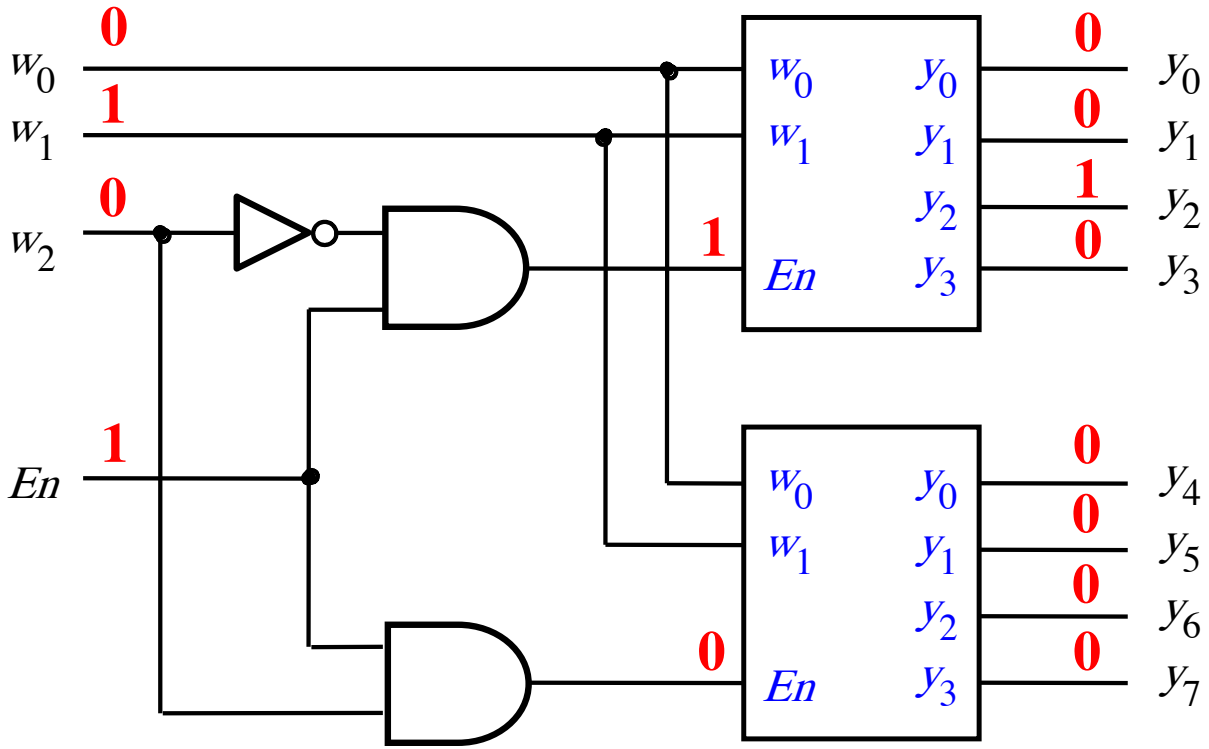
[Figure 4.15 from the textbook]

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



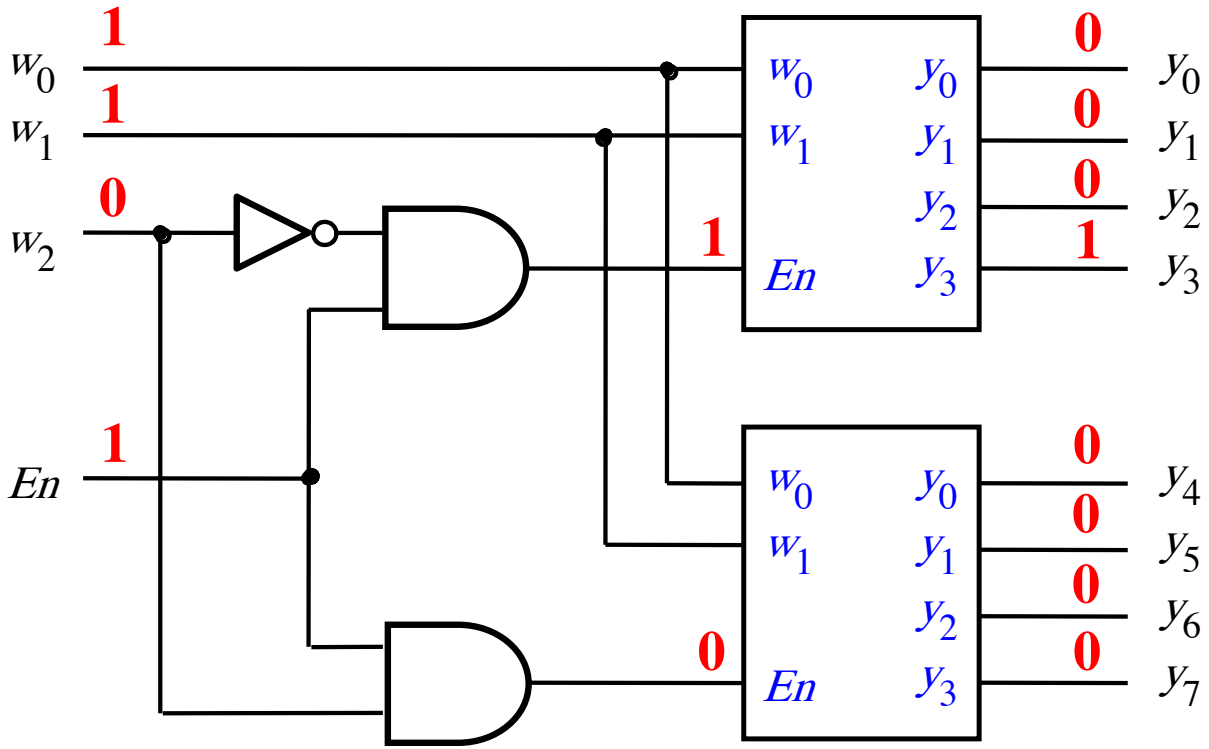
[Figure 4.15 from the textbook]

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



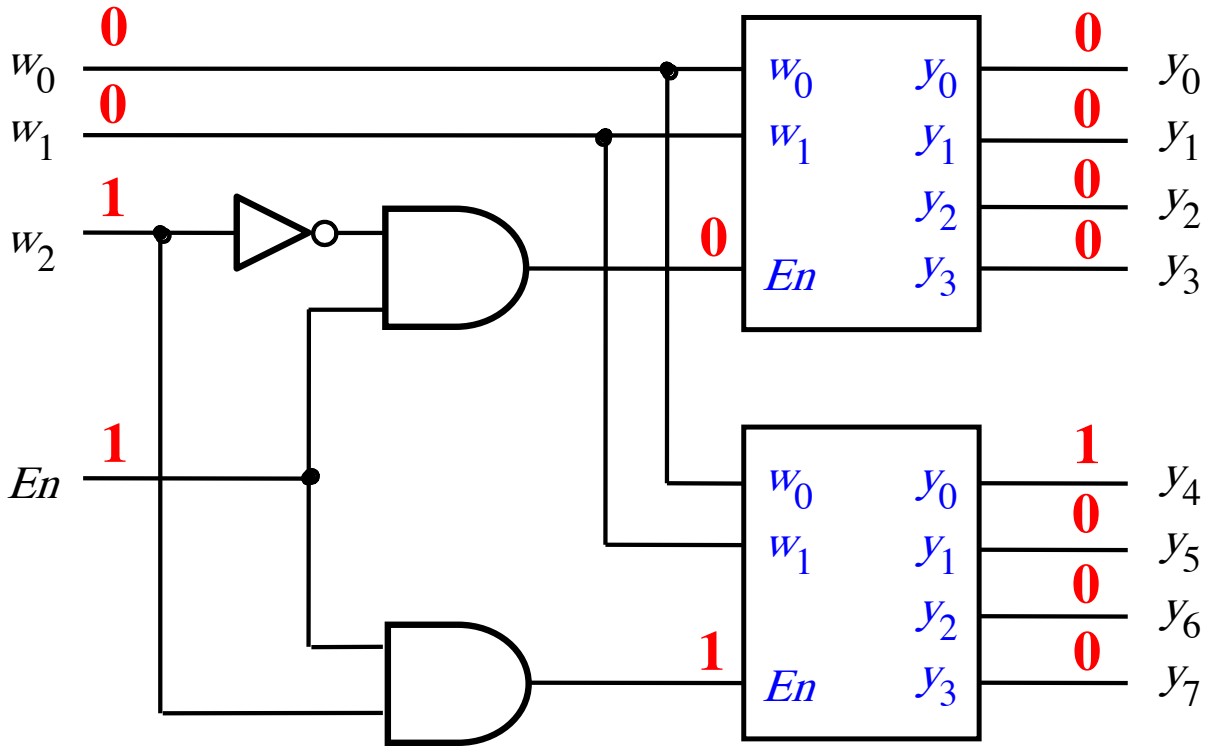
[Figure 4.15 from the textbook]

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



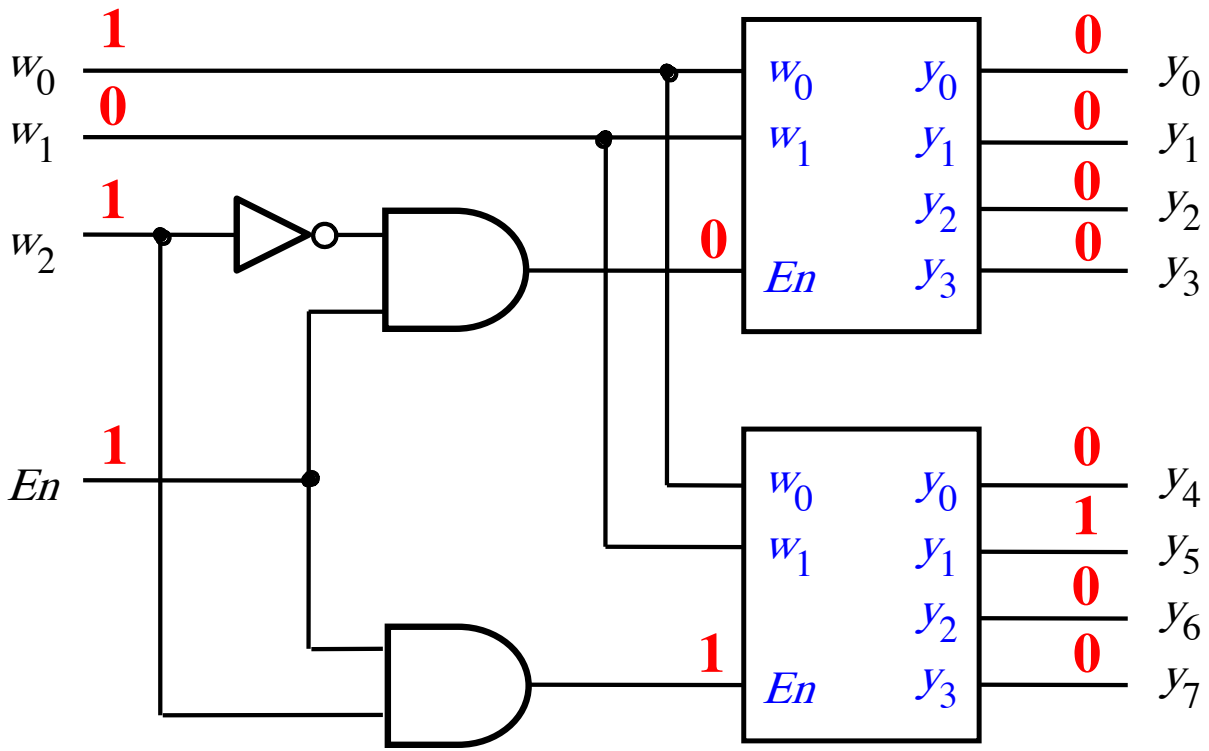
[Figure 4.15 from the textbook]

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



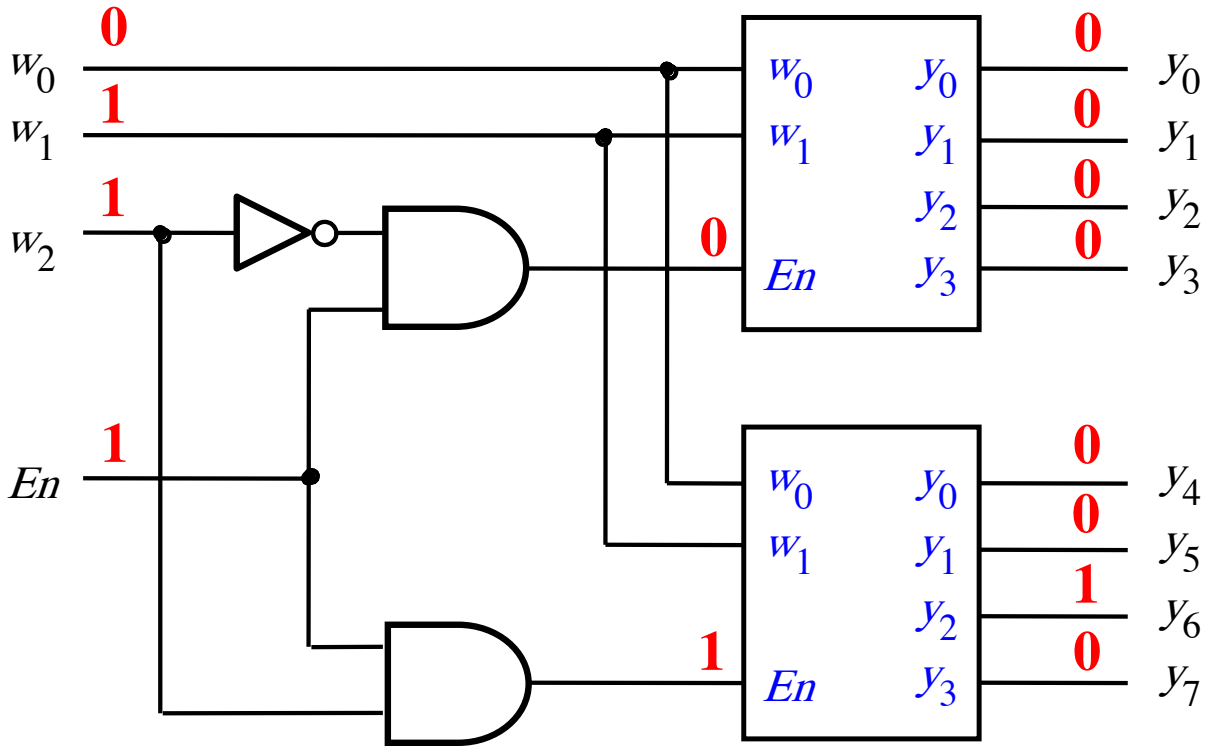
[Figure 4.15 from the textbook]

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



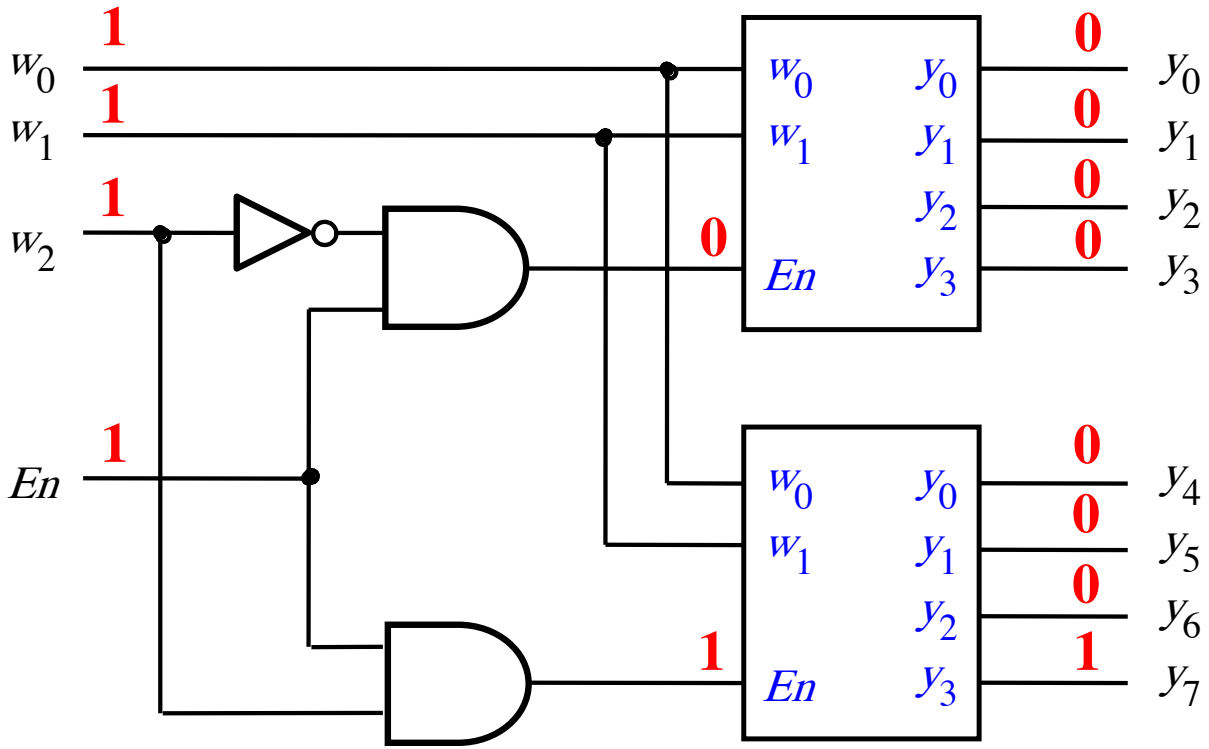
[Figure 4.15 from the textbook]

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



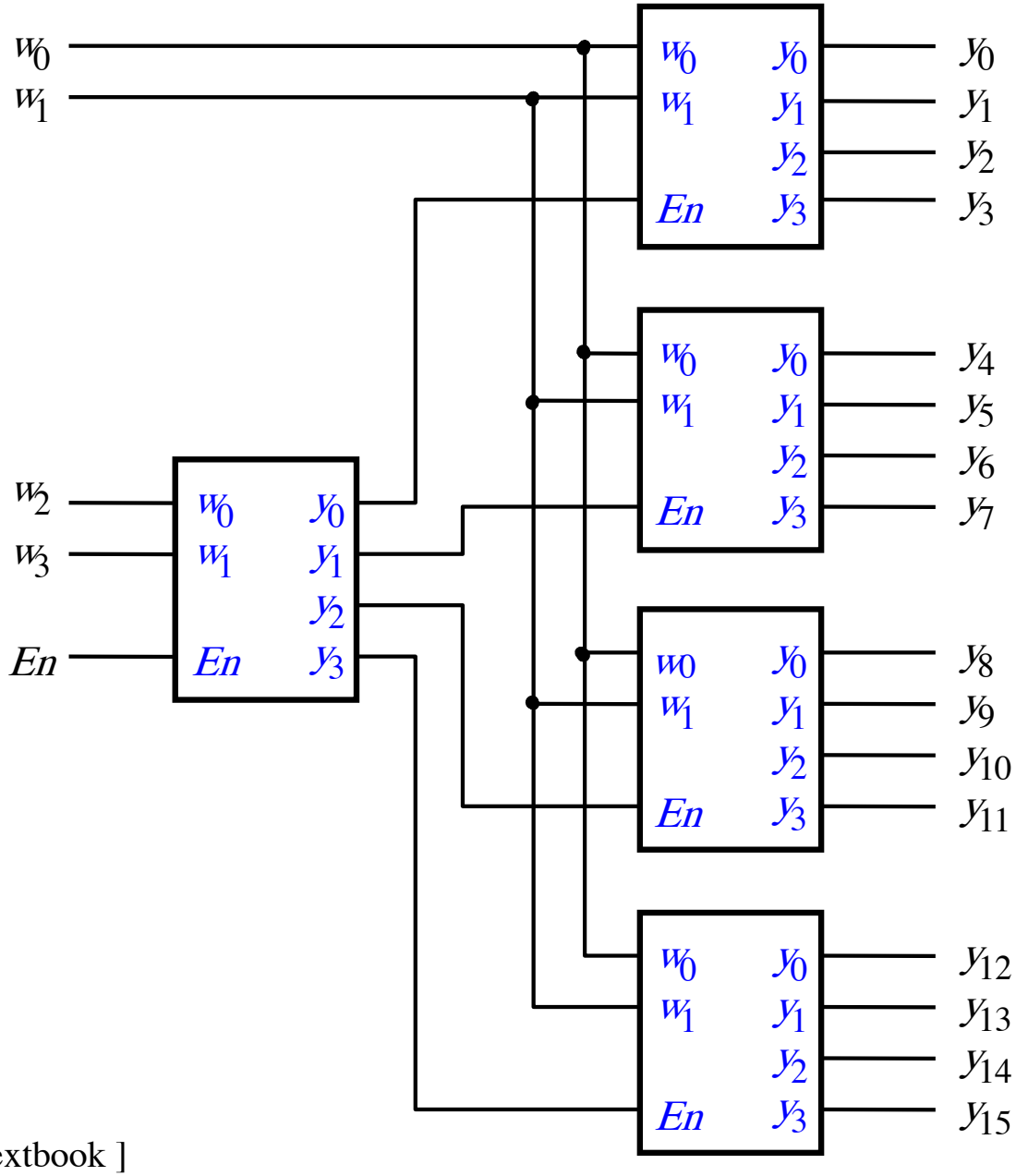
[Figure 4.15 from the textbook]

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



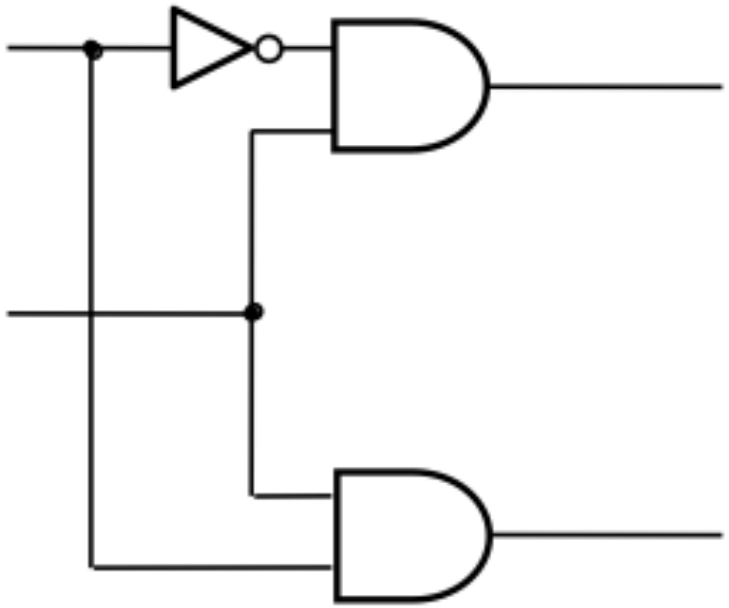
[Figure 4.15 from the textbook]

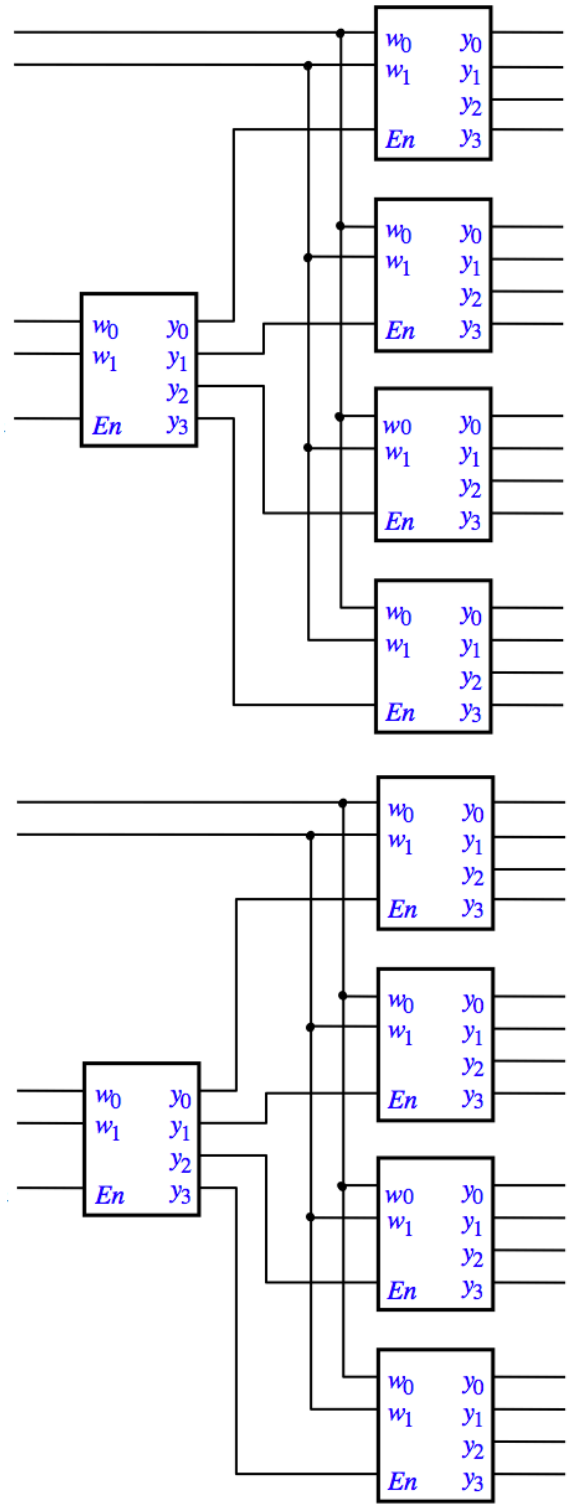
A 4-to-16 decoder built using a decoder tree



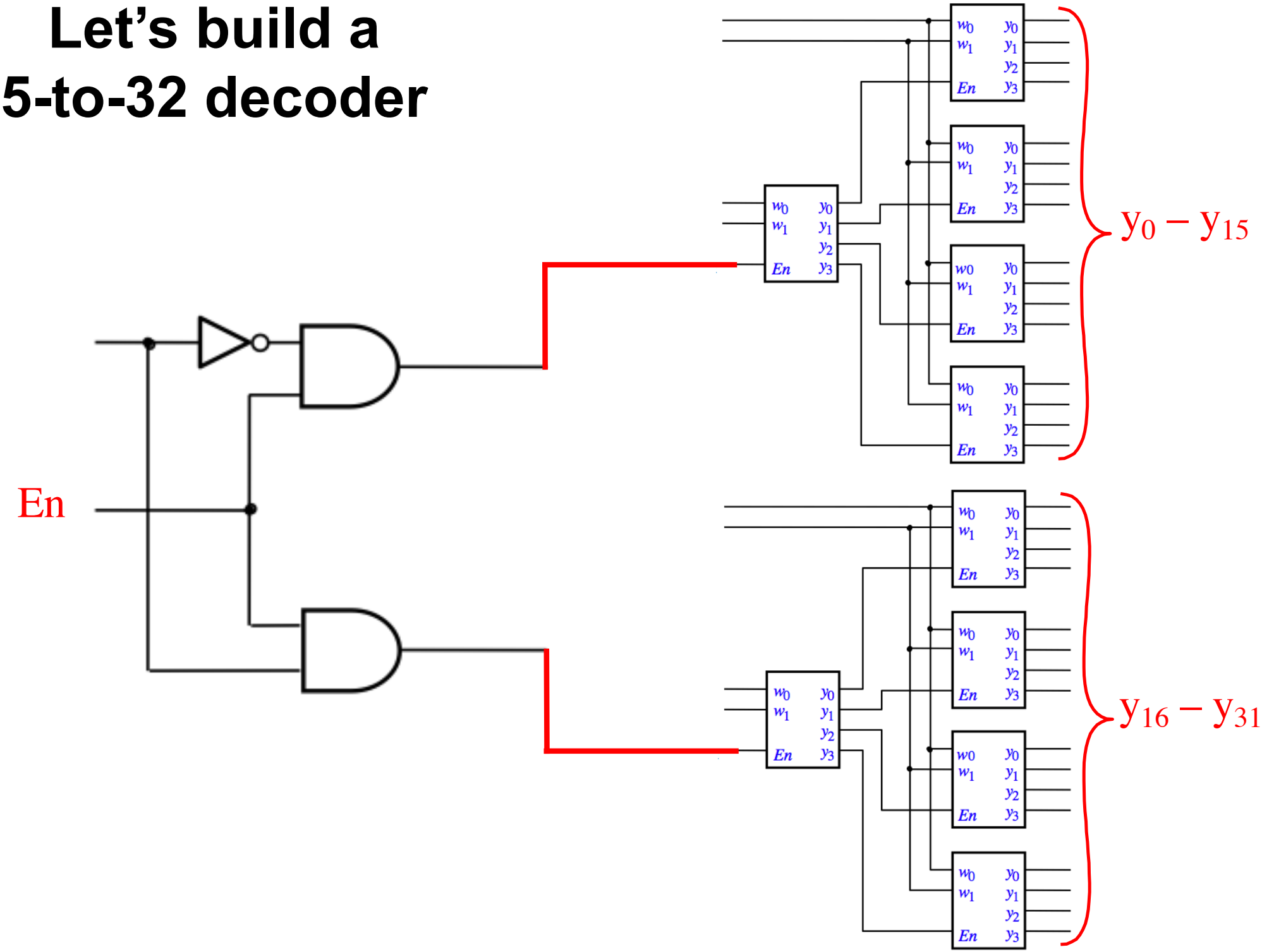
[Figure 4.16 from the textbook]

Let's build a 5-to-32 decoder

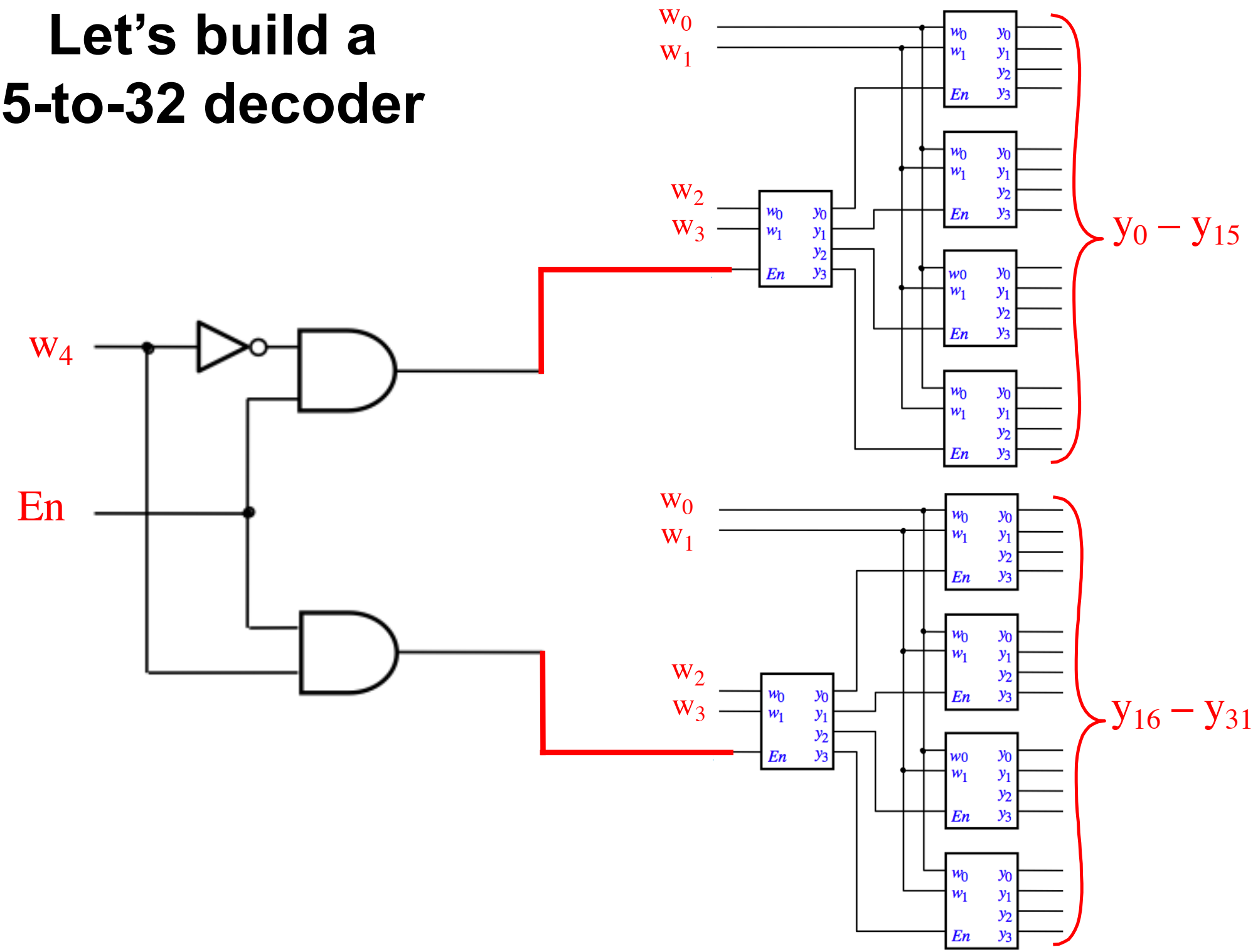




Let's build a 5-to-32 decoder



Let's build a 5-to-32 decoder

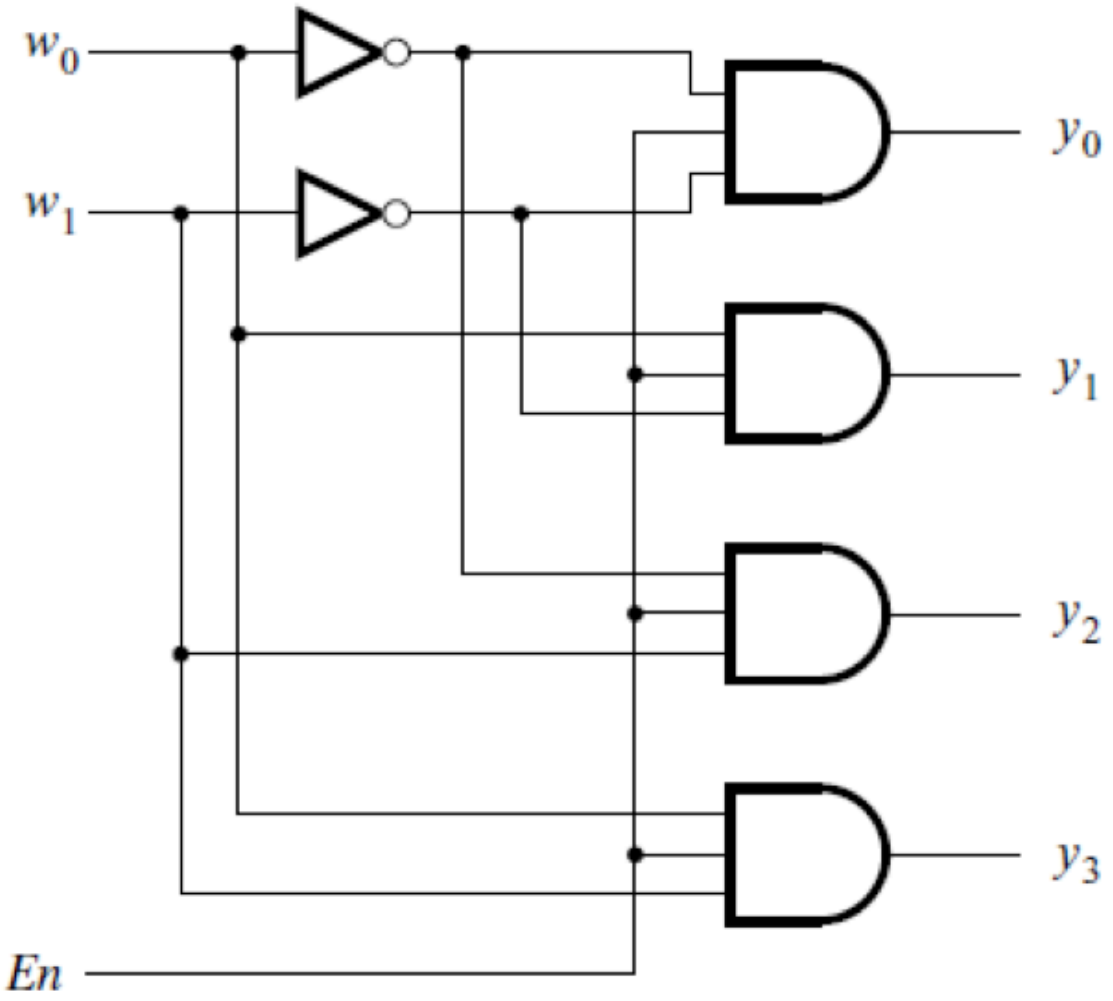


Demultiplexers

1-to-4 Demultiplexer (Definition)

- Has one data input line: D
- Has two output select lines: w_1 and w_0
- Has four outputs: y_0 , y_1 , y_2 , and y_3
- If w_1=0 and w_0=0, then the output y_0 is set to D
- If w_1=0 and w_0=1, then the output y_1 is set to D
- If w_1=1 and w_0=0, then the output y_2 is set to D
- If w_1=1 and w_0=1, then the output y_3 is set to D
- Only one output is set to D. All others are set to 0.

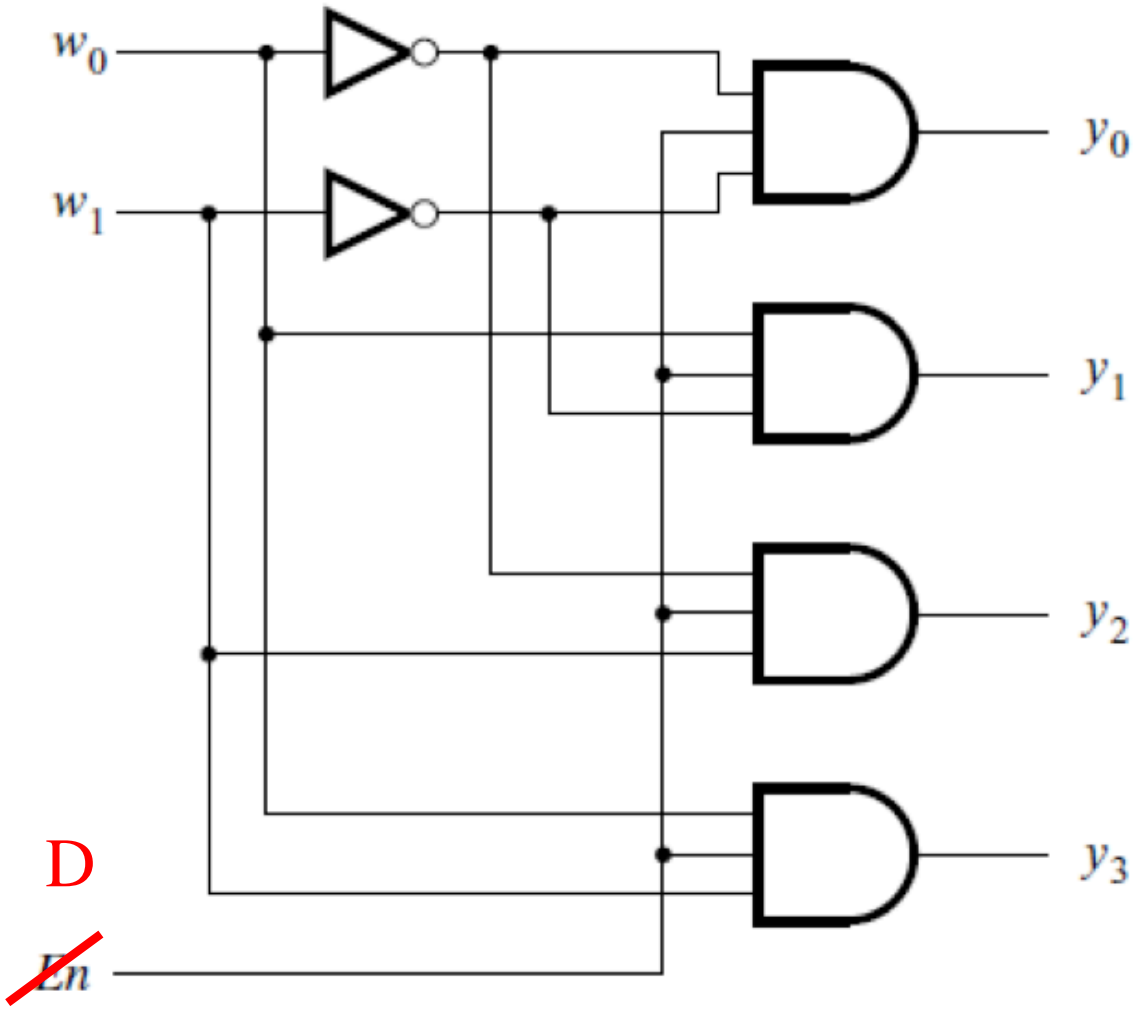
A 1-to-4 demultiplexer built with a 2-to-4 decoder



[Figure 4.14c from the textbook]

A 1-to-4 demultiplexer built with a 2-to-4 decoder

output select lines



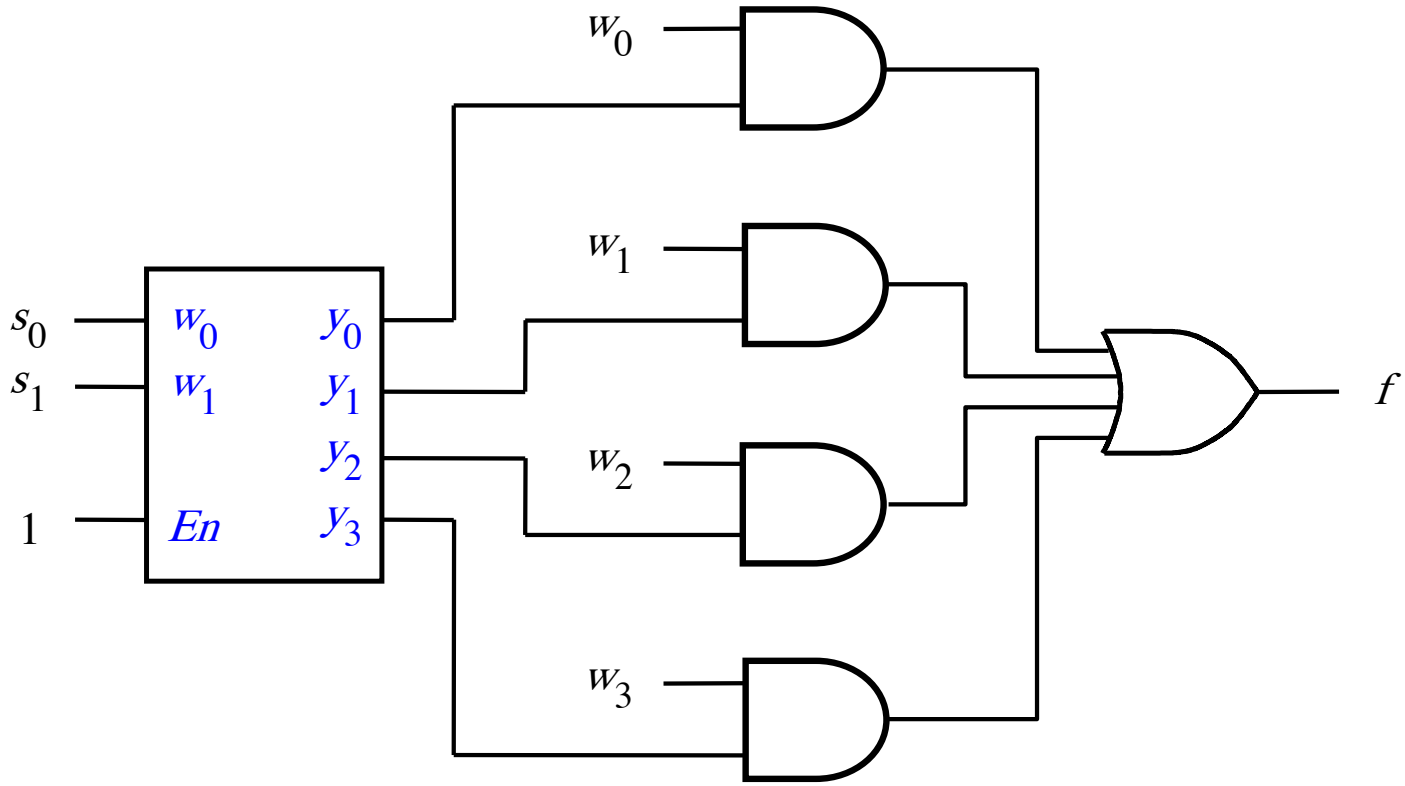
the four output lines

data input line

[Figure 4.14c from the textbook]

Multiplexers
(Implemented with Decoders)

A 4-to-1 multiplexer built using a 2-to-4 decoder



[Figure 4.17 from the textbook]

Encoders

Binary Encoders

4-to-2 Binary Encoder (Definition)

- Has four inputs: w_3 , w_2 , w_1 , and w_0
- Has two outputs: y_1 and y_0
- Only one input is set to 1 ("one-hot" encoded). All others are set to 0.
- If w_0=1 then y_1=0 and y_0=0
- If w_1=1 then y_1=0 and y_0=1
- If w_2=1 then y_1=1 and y_0=0
- If w_3=1 then y_1=1 and y_0=1

Truth table for a 4-to-2 binary encoder

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

[Figure 4.19 from the textbook]

Truth table for a 4-to-2 binary encoder

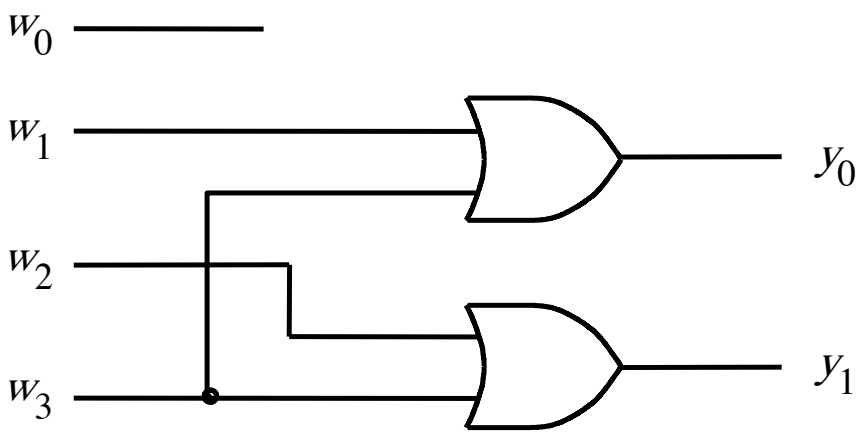
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

The inputs are “one-hot” encoded

[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

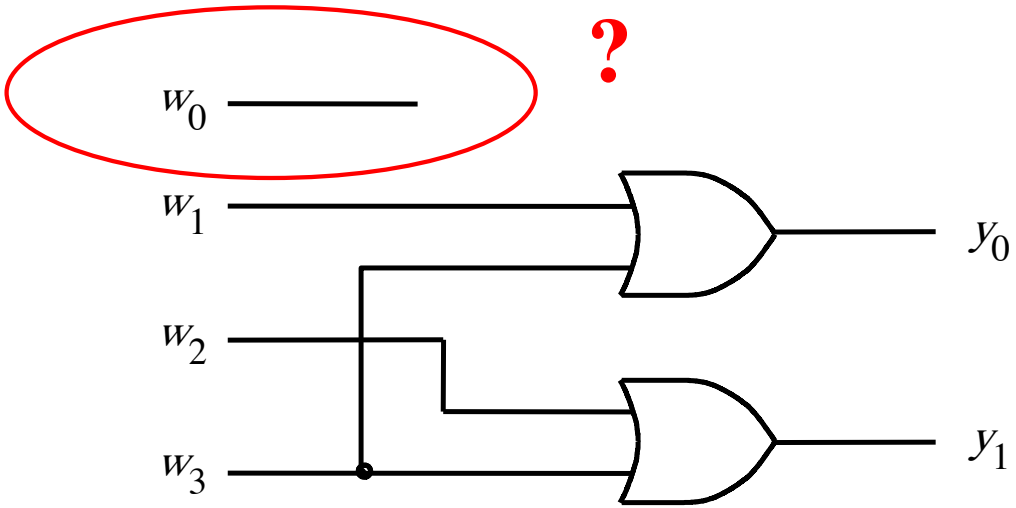
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

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

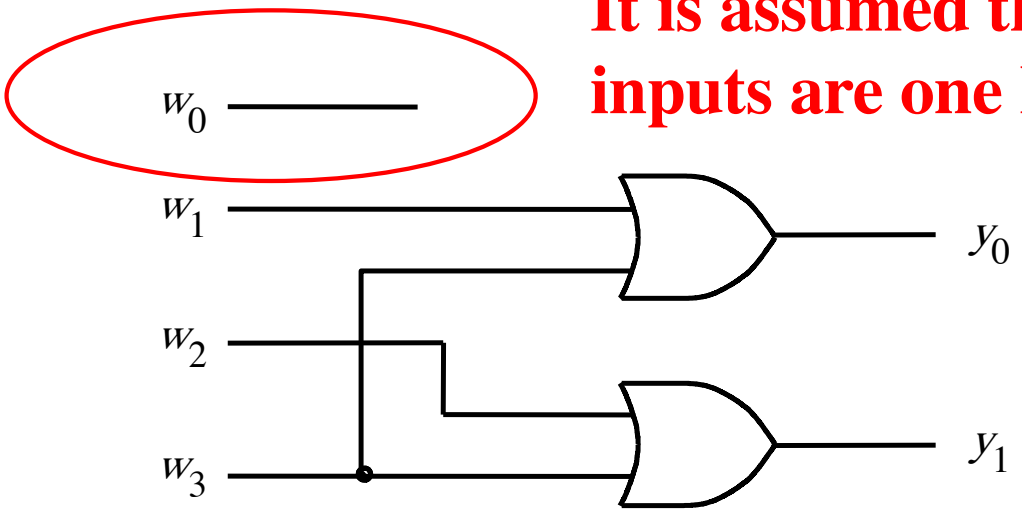


[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

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

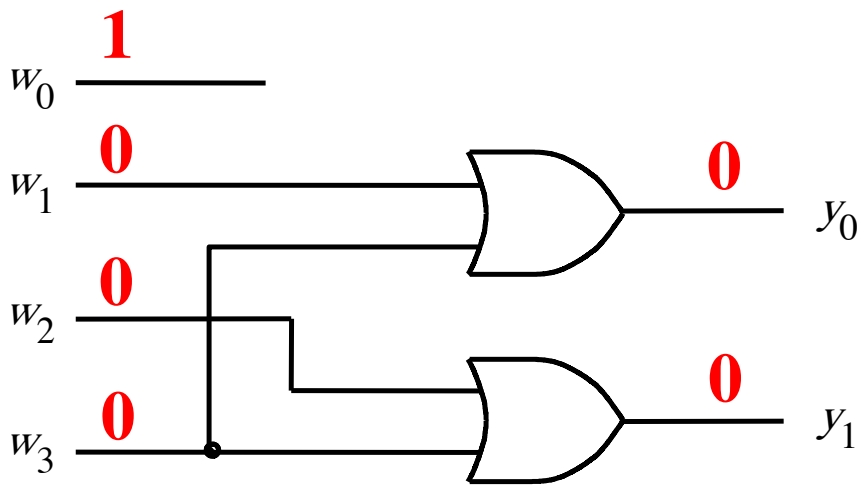
It is assumed that the inputs are one hot encoded



[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

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



[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

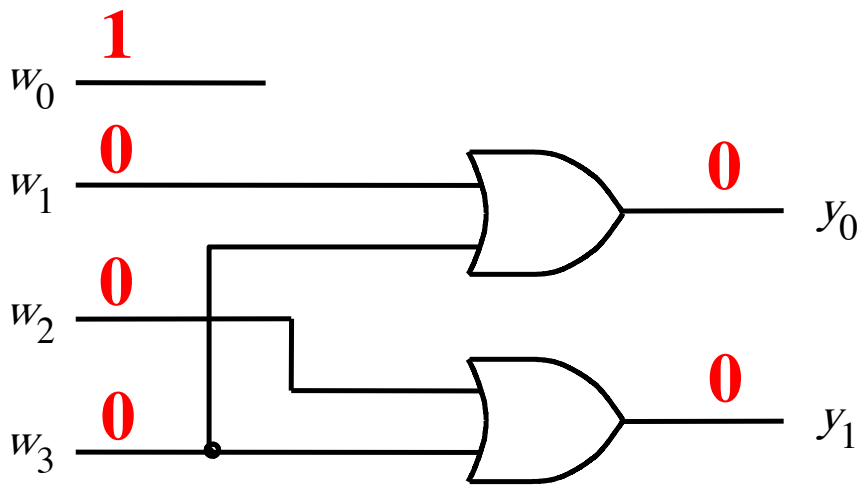
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

As this level of abstraction we need that w_0 input for this to be a proper 4-to-2 binary encoder.

[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

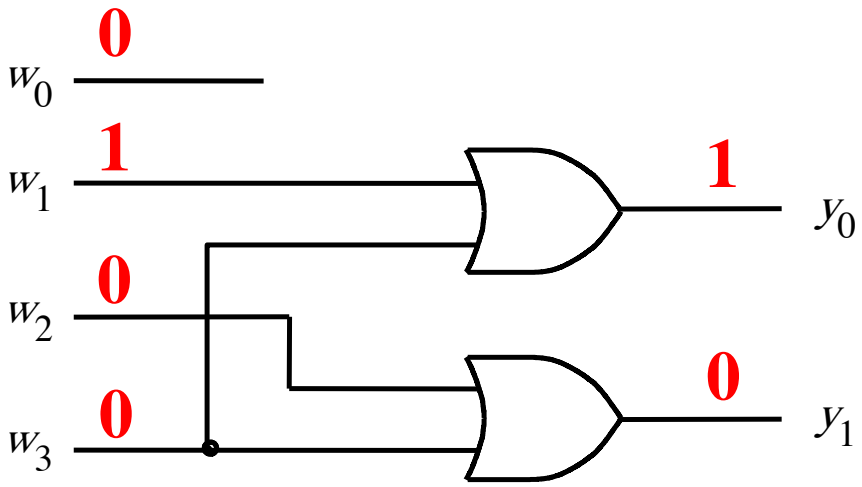
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

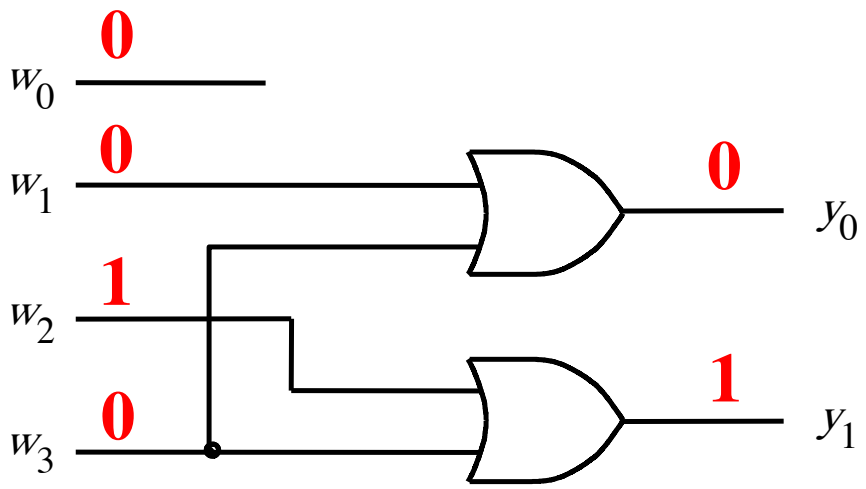
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

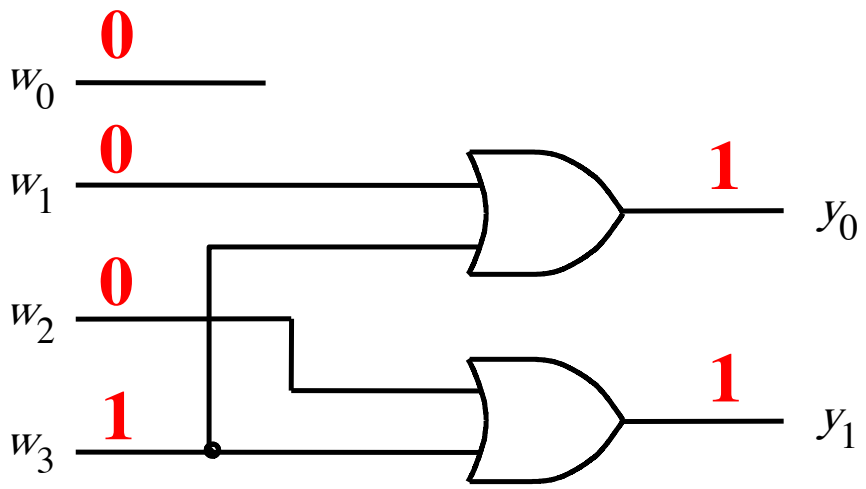
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

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



[Figure 4.19 from the textbook]

Expressions for 4-to-2 binary encoder

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

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

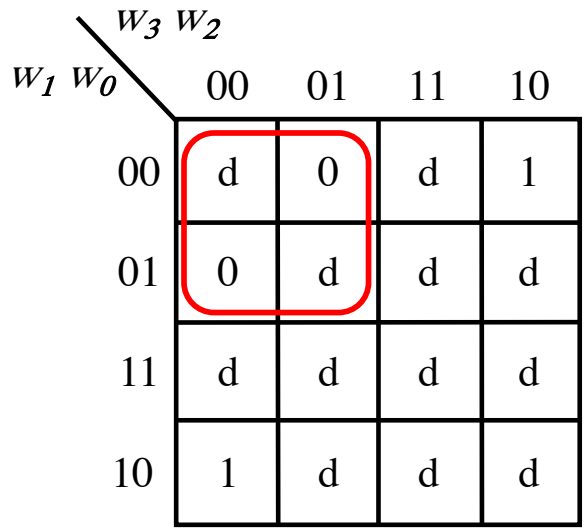
Expressions for 4-to-2 binary encoder

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

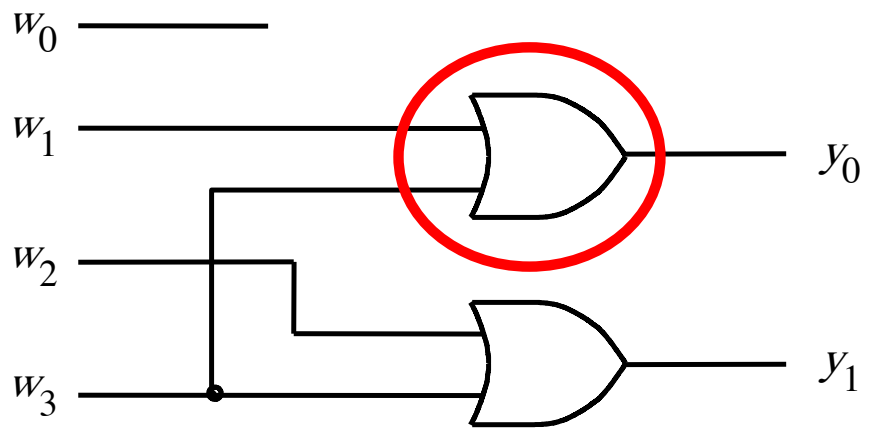
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Expressions for 4-to-2 binary encoder

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

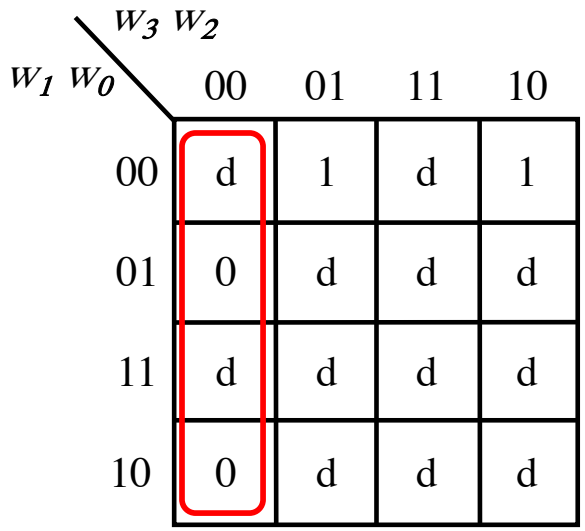


$$y_0 = (w_1 + w_3)$$

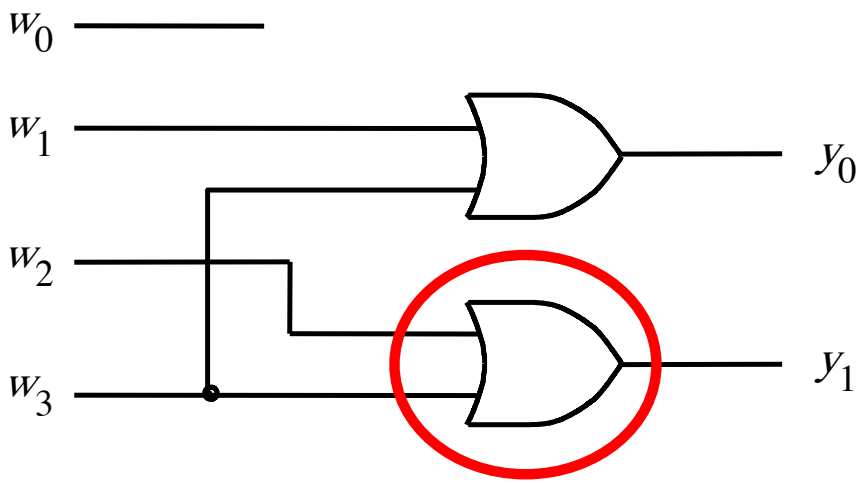


Expressions for 4-to-2 binary encoder

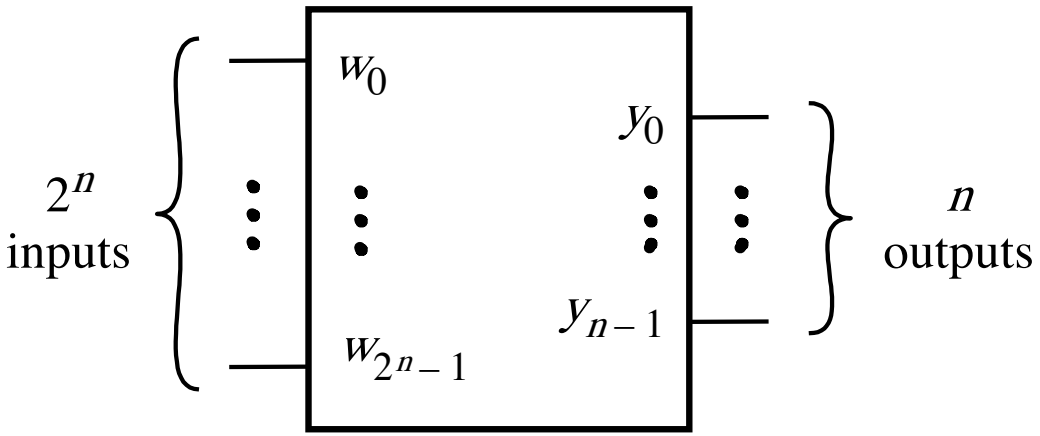
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	0	d	d
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	d	d
0	1	0	0	1	0
0	1	0	1	d	d
0	1	1	0	d	d
0	1	1	1	d	d
1	0	0	0	1	1
1	0	0	1	d	d
1	0	1	0	d	d
1	0	1	1	d	d
1	1	0	0	d	d
1	1	0	1	d	d
1	1	1	0	d	d
1	1	1	1	d	d



$$y_1 = (w_3 + w_2)$$



A 2^n-to-n binary encoder



[Figure 4.18 from the textbook]

Priority Encoders

4-to-2 Priority Encoder (Definition)

- Has four inputs: w_3 , w_2 , w_1 , and w_0
- Has two primary outputs: y_1 and y_0
- Has one other output: z

- The inputs are NOT "one-hot" encoded.
- More than one input can be set to 1 but they have priorities associated with them: w_3 – highest priority and w_0 – lowest priority.

- y_1=0 and y_0=0 (if w_0=1 and w_3=w_2=w_1=0)
- y_1=0 and y_0=1 (if w_1=1 and w_3=w_2=0)
- y_1=1 and y_0=0 (if w_2=1 and w_3=0)
- y_1=1 and y_0=1 (if w_3=1)

4-to-2 Priority Encoder (Definition)

- Has four inputs: w_3 , w_2 , w_1 , and w_0
- Has two primary outputs: y_1 and y_0
- Has one other output: z

- The inputs are NOT "one-hot" encoded.
- More than one input can be set to 1 but they have priorities associated with them: w_3 – highest priority and w_0 – lowest priority.

- y_1=0 and y_0=0 (if w_0=1 and w_3=w_2=w_1=0)
- y_1=0 and y_0=1 (if w_1=1 and w_3=w_2=0) w_0
- y_1=1 and y_0=0 (if w_2=1 and w_3=0) w_0 , w_1
- y_1=1 and y_0=1 (if w_3=1) w_0 , w_1 , w_2

these have lower priorities and can be either 0 or 1.

4-to-2 Priority Encoder (Definition)

- Has four inputs: w_3 , w_2 , w_1 , and w_0
- Has two primary outputs: y_1 and y_0
- Has one other output: z

- The inputs are NOT "one-hot" encoded.
- More than one input can be set to 1 but they have priorities associated with them: w_3 – highest priority and w_0 – lowest priority.

- y_1=0 and y_0=0 (if w_0=1 and w_3=w_2=w_1=0)
- y_1=0 and y_0=1 (if w_1=1 and w_3=w_2=0)
- y_1=1 and y_0=0 (if w_2=1 and w_3=0)
- y_1=1 and y_0=1 (if w_3=1)

- z =0 if w_3=w_2=w_1=w_0=0; otherwise z=1.

Truth table for a 4-to-2 priority encoder (abbreviated version)

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

[Figure 4.20 from the textbook]

Truth table for a 4-to-2 priority encoder (abbreviated version)

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

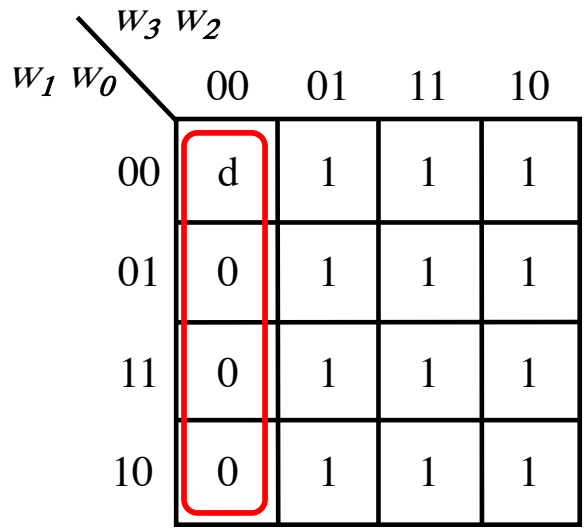
[Figure 4.20 from the textbook]

Truth table for a 4-to-2 priority encoder

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

Expressions for 4-to-2 priority encoder

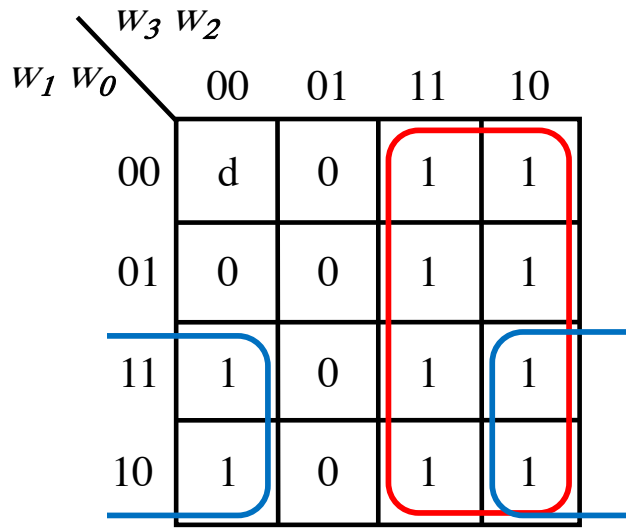
w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1



$$y_1 = w_3 + w_2$$

Expressions for 4-to-2 priority encoder

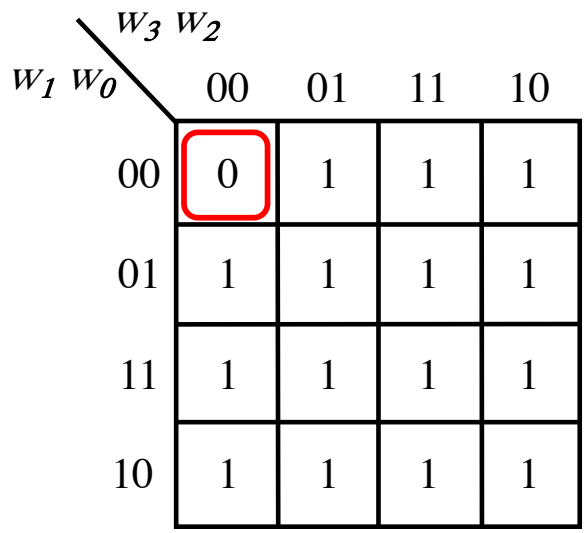
w_3	w_2	w_1	w_0	y_1	y_0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1



$$y_0 = w_3 + w_1 \overline{w_2}$$

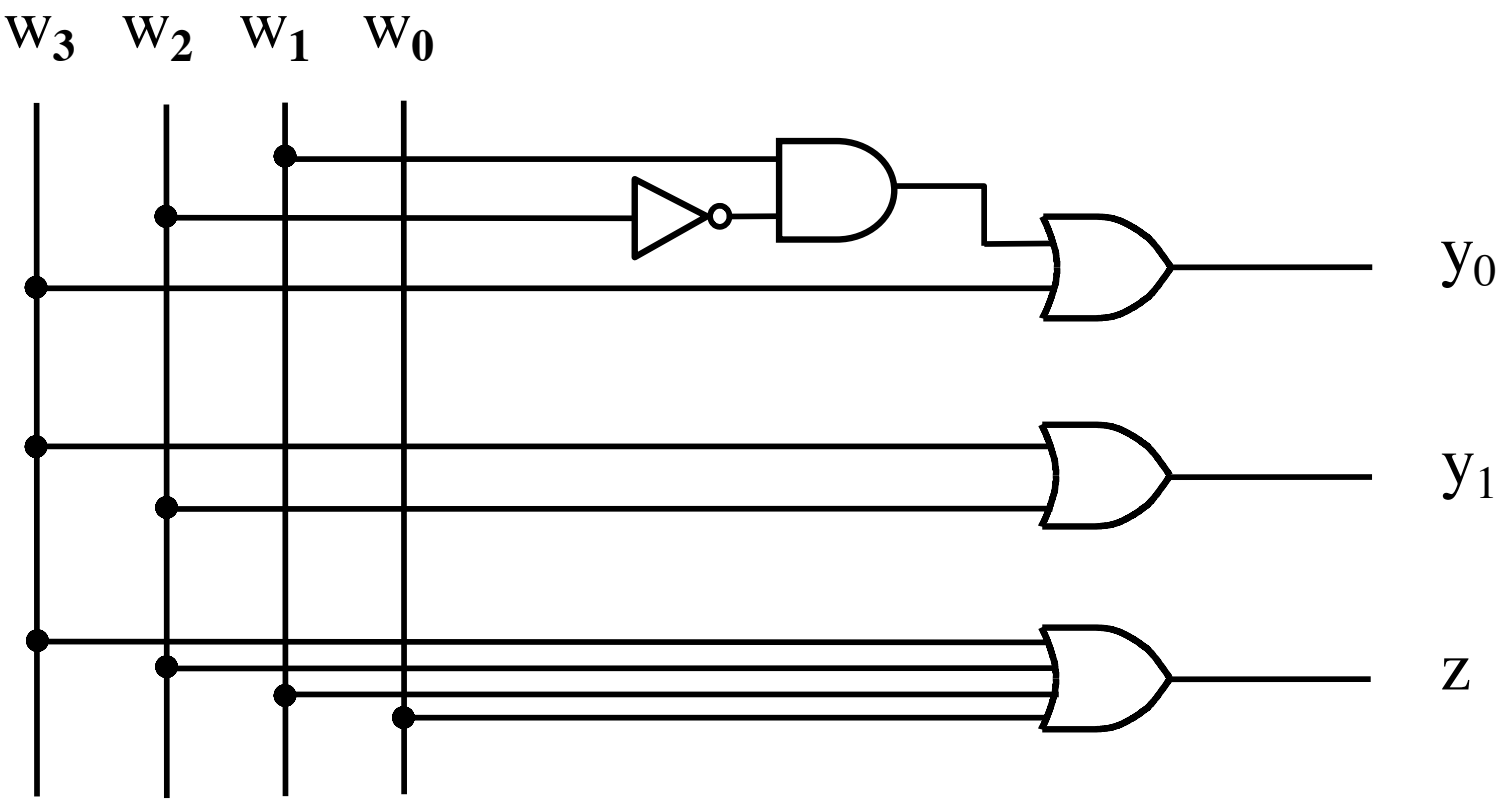
Expressions for 4-to-2 priority encoder

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	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1



$$z = w_3 + w_2 + w_1 + w_0$$

Circuit for the 4-to-2 priority encoder



The textbook derives a different circuit for the 4-to-2 priority encoder using a 4-to-2 binary encoder

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

$$i_0 = \overline{w}_3\overline{w}_2\overline{w}_1 w_0$$
$$i_1 = \overline{w}_3\overline{w}_2 w_1$$
$$i_2 = \overline{w}_3 w_2$$
$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$
$$y_1 = i_2 + i_3$$

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

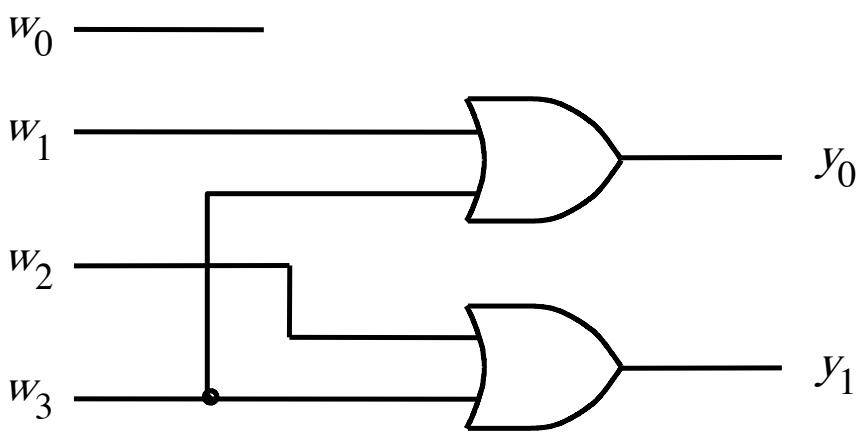
The textbook derives a different circuit for the 4-to-2 priority encoder using a 4-to-2 binary encoder

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

$$i_0 = \overline{w}_3\overline{w}_2\overline{w}_1 w_0$$
$$i_1 = \overline{w}_3\overline{w}_2 w_1$$
$$i_2 = \overline{w}_3 w_2$$
$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$
$$y_1 = i_2 + i_3$$

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



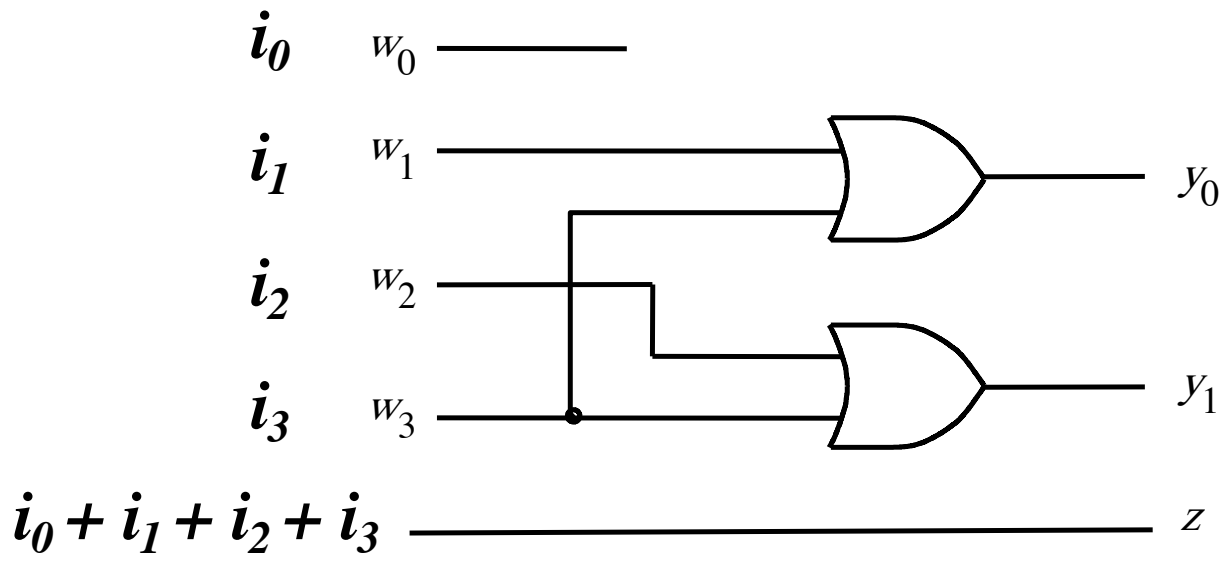
The textbook derives a different circuit for the 4-to-2 priority encoder using a 4-to-2 binary encoder

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

$$i_0 = \overline{w}_3\overline{w}_2\overline{w}_1 w_0$$
$$i_1 = \overline{w}_3\overline{w}_2 w_1$$
$$i_2 = \overline{w}_3 w_2$$
$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$
$$y_1 = i_2 + i_3$$

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



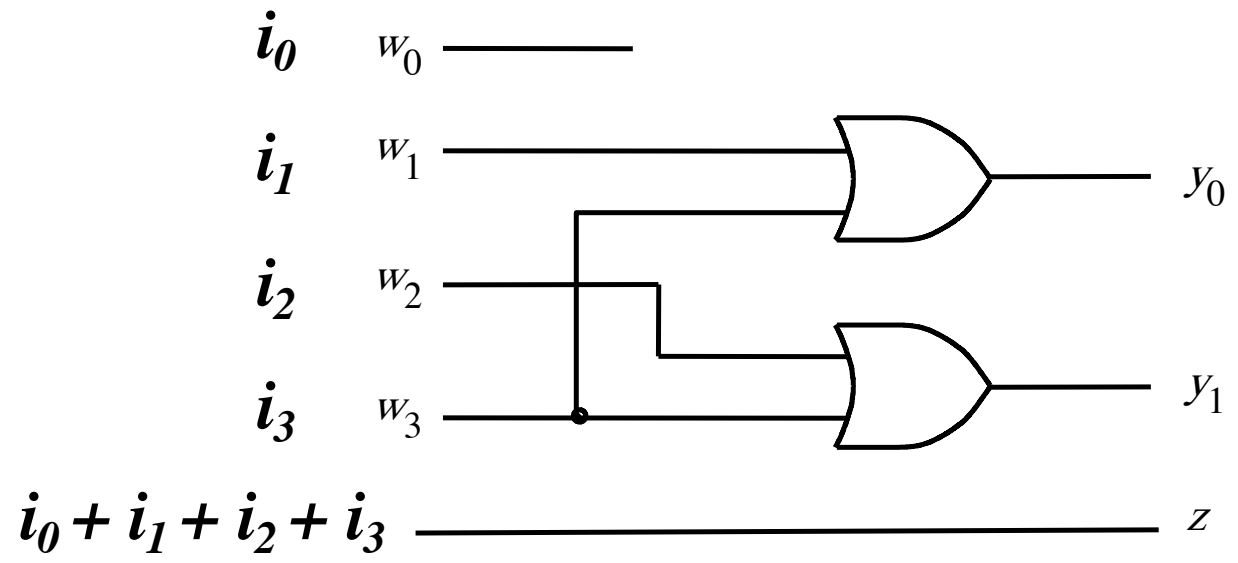
The textbook derives a different circuit for the 4-to-2 priority encoder using a 4-to-2 binary encoder

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

$$i_0 = \overline{w}_3\overline{w}_2\overline{w}_1 w_0$$
$$i_1 = \overline{w}_3\overline{w}_2 w_1$$
$$i_2 = \overline{w}_3 w_2$$
$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$
$$y_1 = i_2 + i_3$$

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



Try to prove that this is equivalent to the circuit that was derived above.

Code Converters

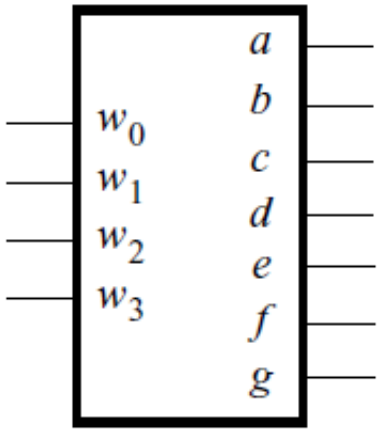
Code Converter (Definition)

- **Converts from one type of input encoding to a different type of output encoding.**

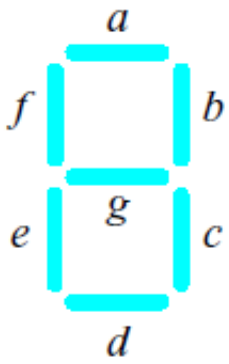
Code Converter (Definition)

- **Converts from one type of input encoding to a different type of output encoding.**
- **A decoder does that as well, but its outputs are always one-hot encoded so the output code is really only one type of output code.**
- **A binary encoder does that as well but its inputs are always one-hot encoded so the input code is really only one type of input code.**

A hex-to-7-segment display code converter



(a) Code converter



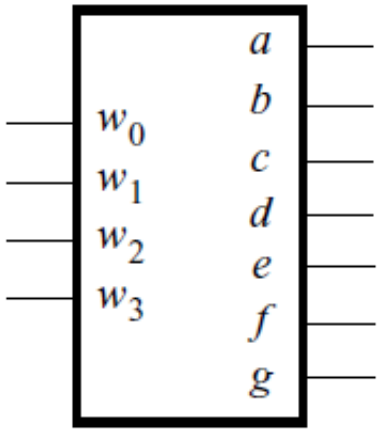
(b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

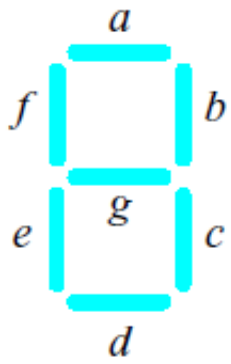
(c) Truth table

[Figure 4.21 from the textbook]

A hex-to-7-segment display code converter



(a) Code converter



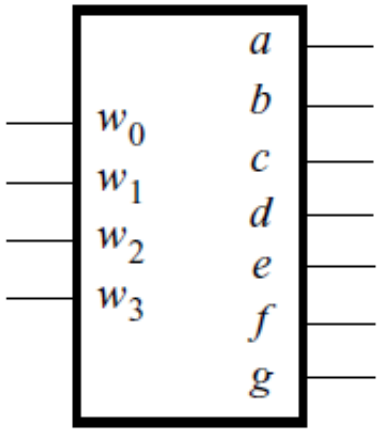
(b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

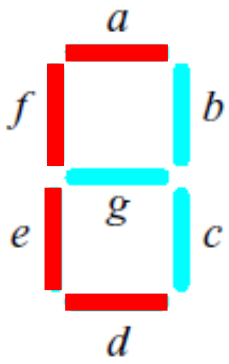
(c) Truth table

[Figure 4.21 from the textbook]

A hex-to-7-segment display code converter



(a) Code converter



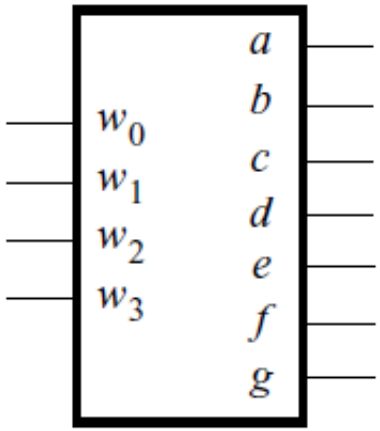
(b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

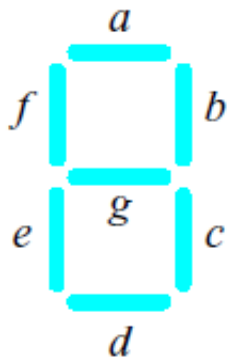
(c) Truth table

[Figure 4.21 from the textbook]

A hex-to-7-segment display code converter



(a) Code converter



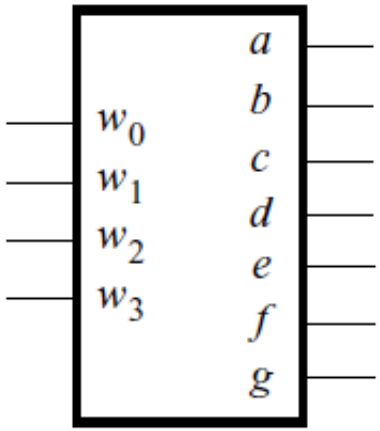
(b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

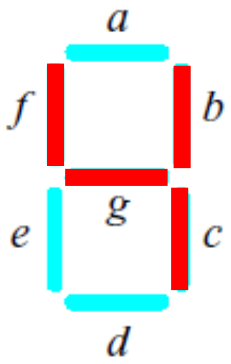
(c) Truth table

[Figure 4.21 from the textbook]

A hex-to-7-segment display code converter



(a) Code converter



(b) 7-segment display

w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

(c) Truth table

[Figure 4.21 from the textbook]

What is the circuit for this code converter?

x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

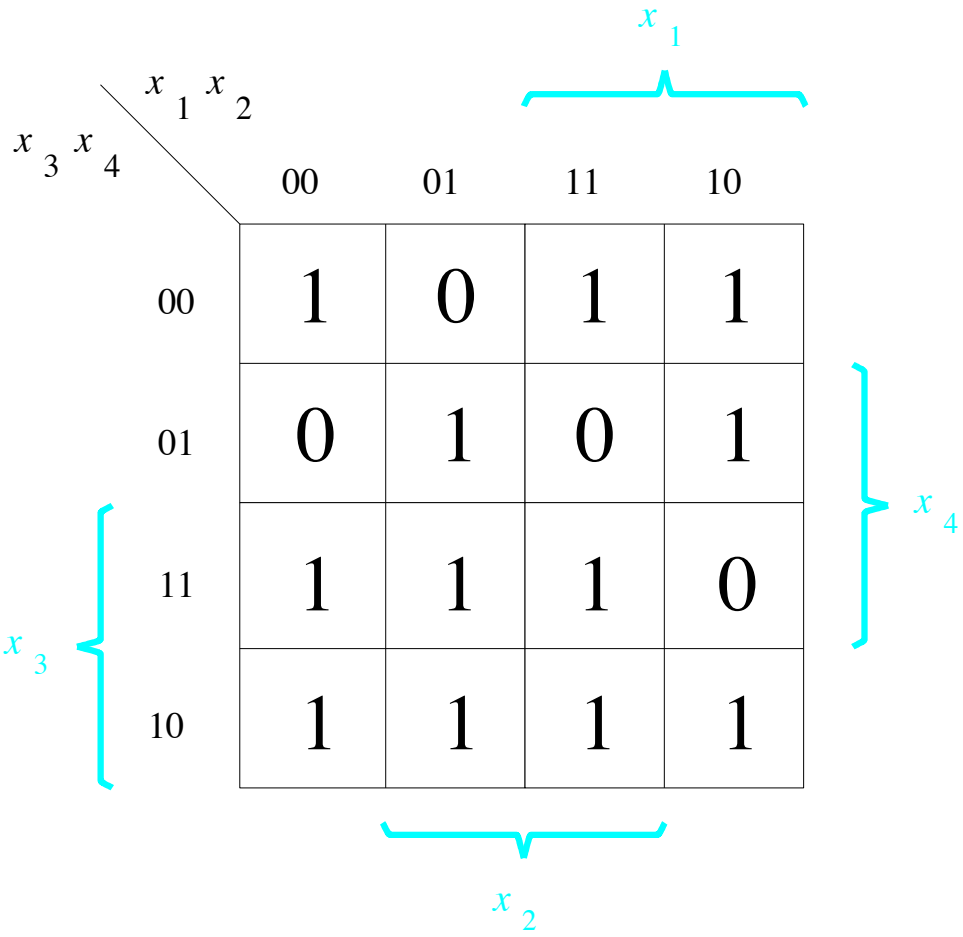
What is the circuit for this code converter?

x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

$$f(x_1, x_2, x_3, x_4) = \Sigma\, m(0, 2, 3, 5, 6, 7, 8, 9, 10, 12, 14, 15)$$

What is the circuit for this code converter?

x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1



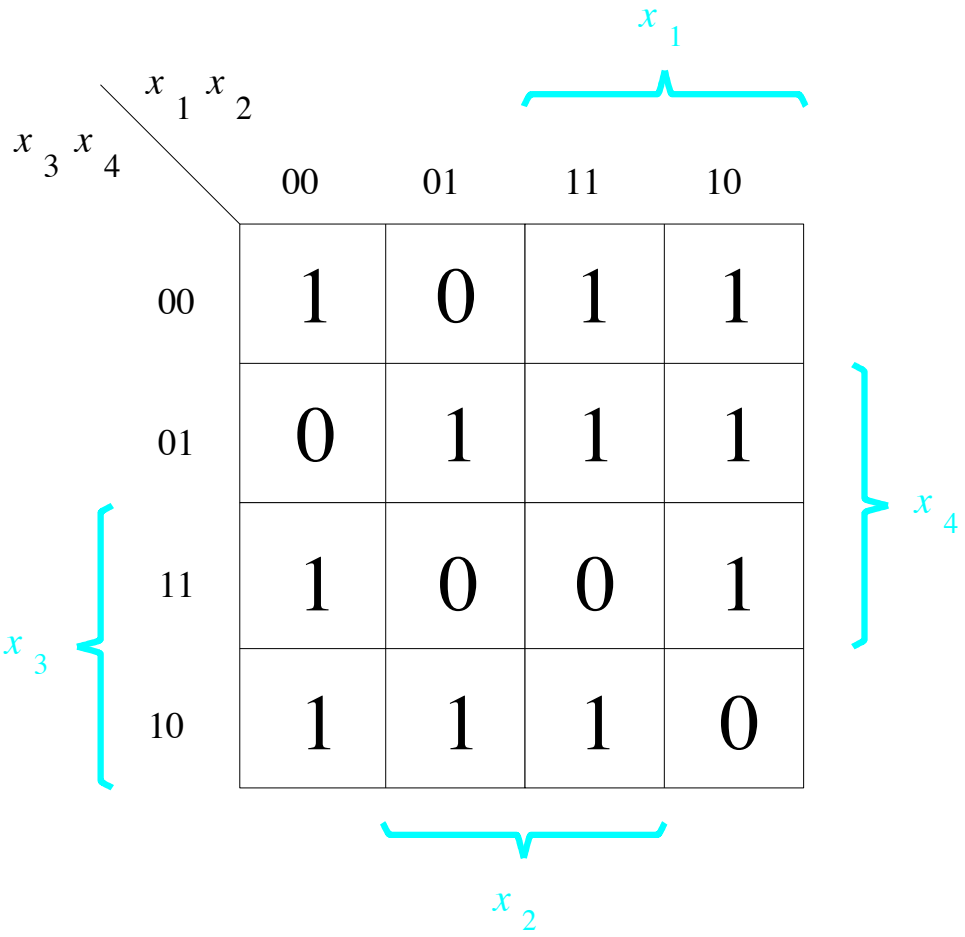
$$f(x_1, x_2, x_3, x_4) = \Sigma\, m(0, 2, 3, 5, 6, 7, 8, 9, 10, 12, 14, 15)$$

What is the circuit for this code converter?

x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

What is the circuit for this code converter?

x_3	x_2	x_1	x_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1



$$f(x_1, x_2, x_3, x_4) = \Sigma\, m(0, 2, 3, 5, 6, 8, 9, 11, 12, 13, 14)$$

Example Problems from Chapter 4

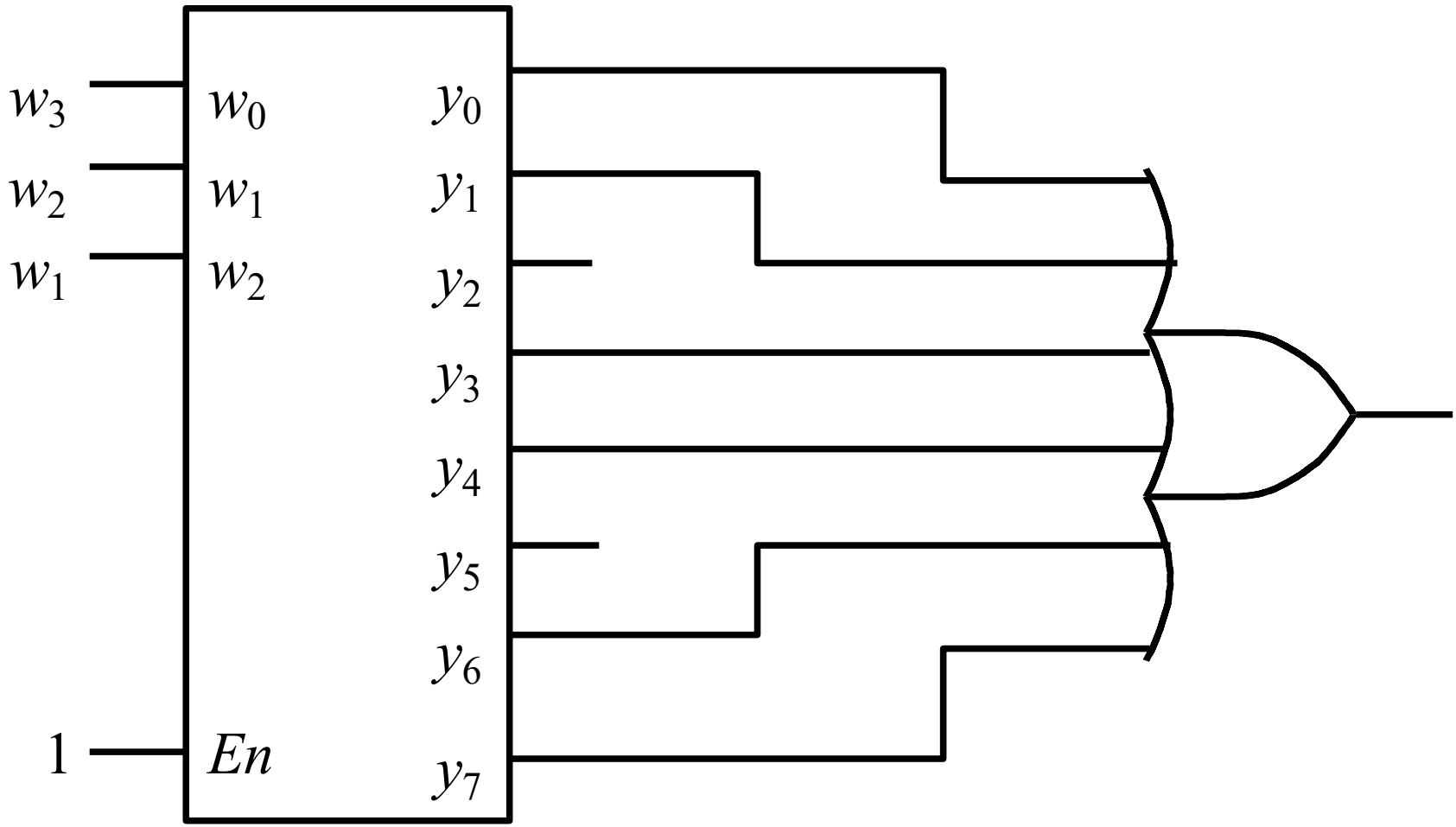
Example 1: SOP vs Decoders

Implement the function

$$f(w_1, w_2, w_3) = \Sigma\, m(0, 1, 3, 4, 6, 7)$$

by using a 3-to-8 binary decoder and one OR gate.

Solution Circuit

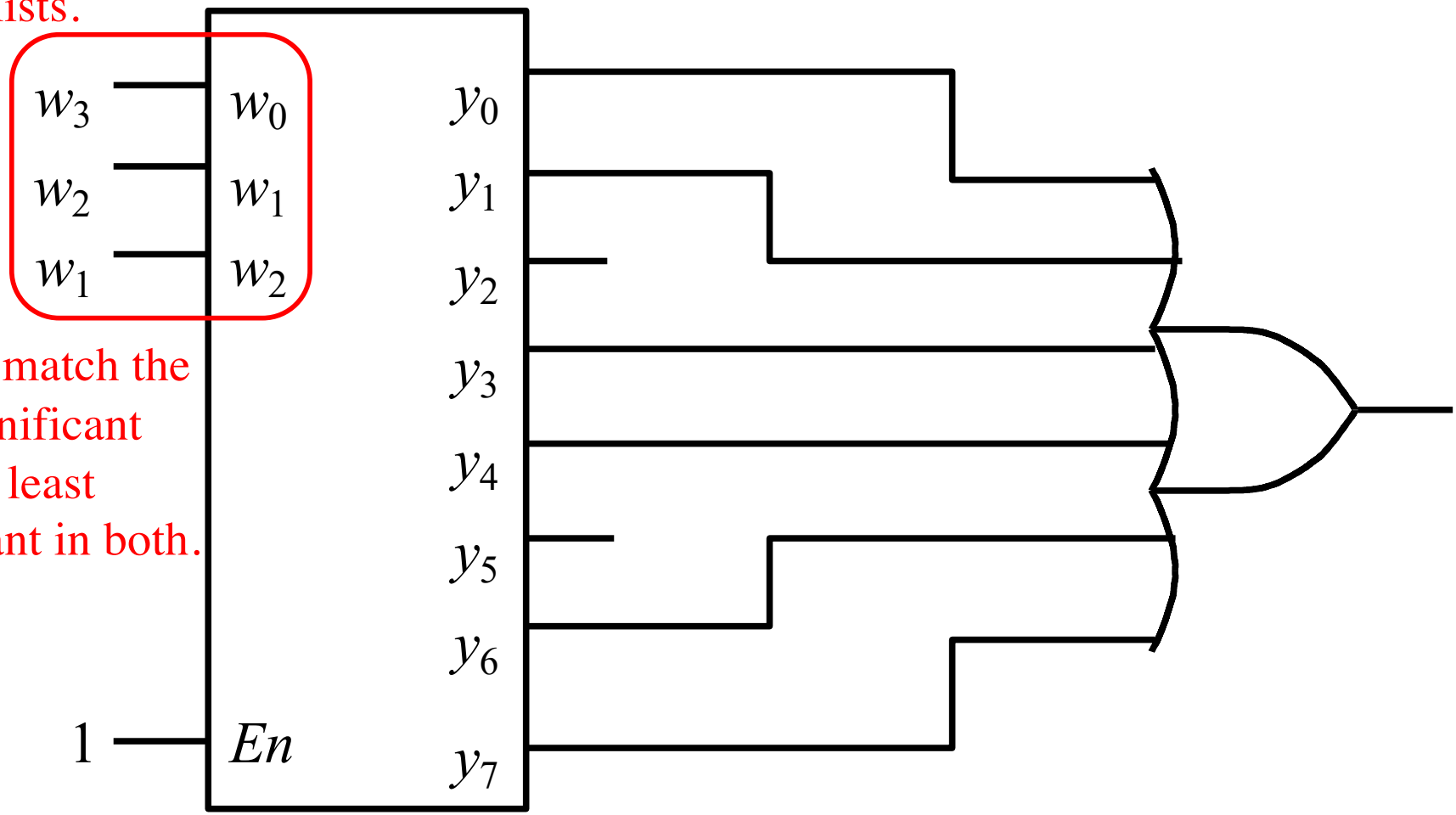


$$f(w_1, w_2, w_3) = \Sigma\, m(0, 1, 3, 4, 6, 7)$$

[Figure 4.44 from the textbook]

Solution Circuit

Notice this swap of variables in the two lists.



Need to match the least significant with the least significant in both.

$$f(w_1, w_2, w_3) = \Sigma\, m(0, 1, 3, 4, 6, 7)$$

[Figure 4.44 from the textbook]

Example 2: Implement an 8-to-3 binary encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

[Figure 4.45 from the textbook]

Example 2: Implement an 8-to-3 binary encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$y_0 = w_1 + w_3 + w_5 + w_7$$
$$y_1 = w_2 + w_3 + w_6 + w_7$$
$$y_2 = w_4 + w_5 + w_6 + w_7$$

[Figure 4.45 from the textbook]

Example 2: Implement an 8-to-3 binary encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$y_0 = w_1 + w_3 + w_5 + w_7$$
$$y_1 = w_2 + w_3 + w_6 + w_7$$
$$y_2 = w_4 + w_5 + w_6 + w_7$$

Example 2: Implement an 8-to-3 binary encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

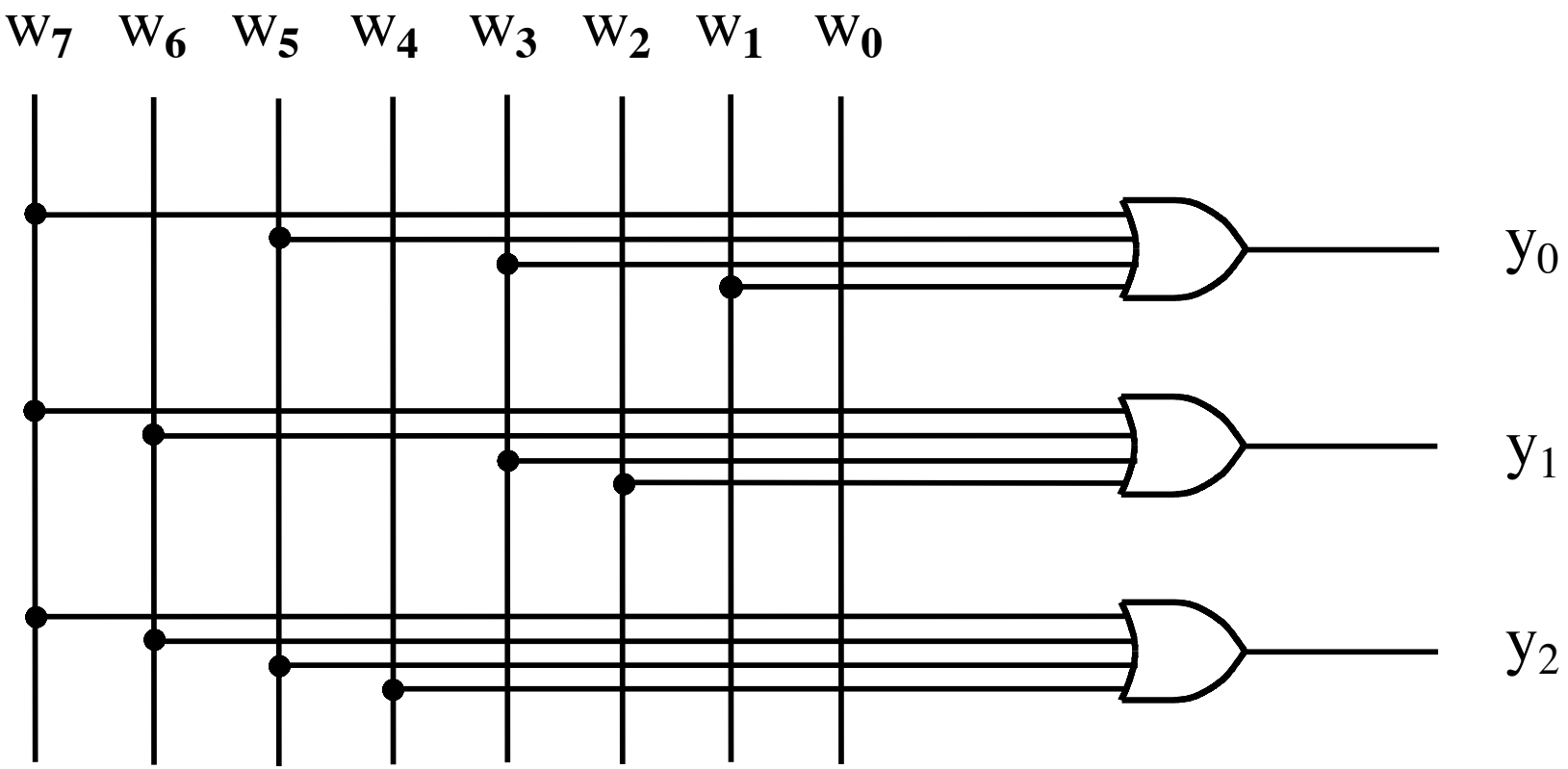
$$y_0 = w_1 + w_3 + w_5 + w_7$$
$$y_1 = w_2 + w_3 + w_6 + w_7$$
$$y_2 = w_4 + w_5 + w_6 + w_7$$

Example 2: Implement an 8-to-3 binary encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$y_0 = w_1 + w_3 + w_5 + w_7$$
$$y_1 = w_2 + w_3 + w_6 + w_7$$
$$y_2 = w_4 + w_5 + w_6 + w_7$$

Circuit for the 8-to-3 binary encoder



$$y_0 = w_1 + w_3 + w_5 + w_7$$
$$y_1 = w_2 + w_3 + w_6 + w_7$$
$$y_2 = w_4 + w_5 + w_6 + w_7$$

Example 3: Implement an 8-to-3 priority encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Example 3: Implement an 8-to-3 priority encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

Example 3: Implement an 8-to-3 priority encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0	z
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	x	0	0	1	1
0	0	0	0	0	1	x	x	0	1	0	1
0	0	0	0	1	x	x	x	0	1	1	1
0	0	0	1	x	x	x	x	1	0	0	1
0	0	1	x	x	x	x	x	1	0	1	1
0	1	x	x	x	x	x	x	1	1	0	1
1	x	x	x	x	x	x	x	1	1	1	1
0	0	0	0	0	0	0	0	d	d	d	0

Example 3: Implement an 8-to-3 priority encoder

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0	z
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	x	0	0	1	1
0	0	0	0	0	1	x	x	0	1	0	1
0	0	0	0	1	x	x	x	0	1	1	1
0	0	0	1	x	x	x	x	1	0	0	1
0	0	1	x	x	x	x	x	1	0	1	1
0	1	x	x	x	x	x	x	1	1	0	1
1	x	x	x	x	x	x	x	1	1	1	1
0	0	0	0	0	0	0	0	d	d	d	0

Example 3: Implement an 8-to-3 priority encoder

$i_0 = \overline{w_7}\,\overline{w_6}\,\overline{w_5}\,\overline{w_4}\,\overline{w_3}\,\overline{w_2}\,\overline{w_1}\,w_0$

$i_1 = \overline{w_7}\,\overline{w_6}\,\overline{w_5}\,\overline{w_4}\,\overline{w_3}\,\overline{w_2}\,w_1$

$i_2 = \overline{w_7}\,\overline{w_6}\,\overline{w_5}\,\overline{w_4}\,\overline{w_3}\,w_2$

$i_3 = \overline{w_7}\,\overline{w_6}\,\overline{w_5}\,\overline{w_4}\,w_3$

$i_4 = \overline{w_7}\,\overline{w_6}\,\overline{w_5}\,w_4$

$i_5 = \overline{w_7}\,\overline{w_6}\,w_5$

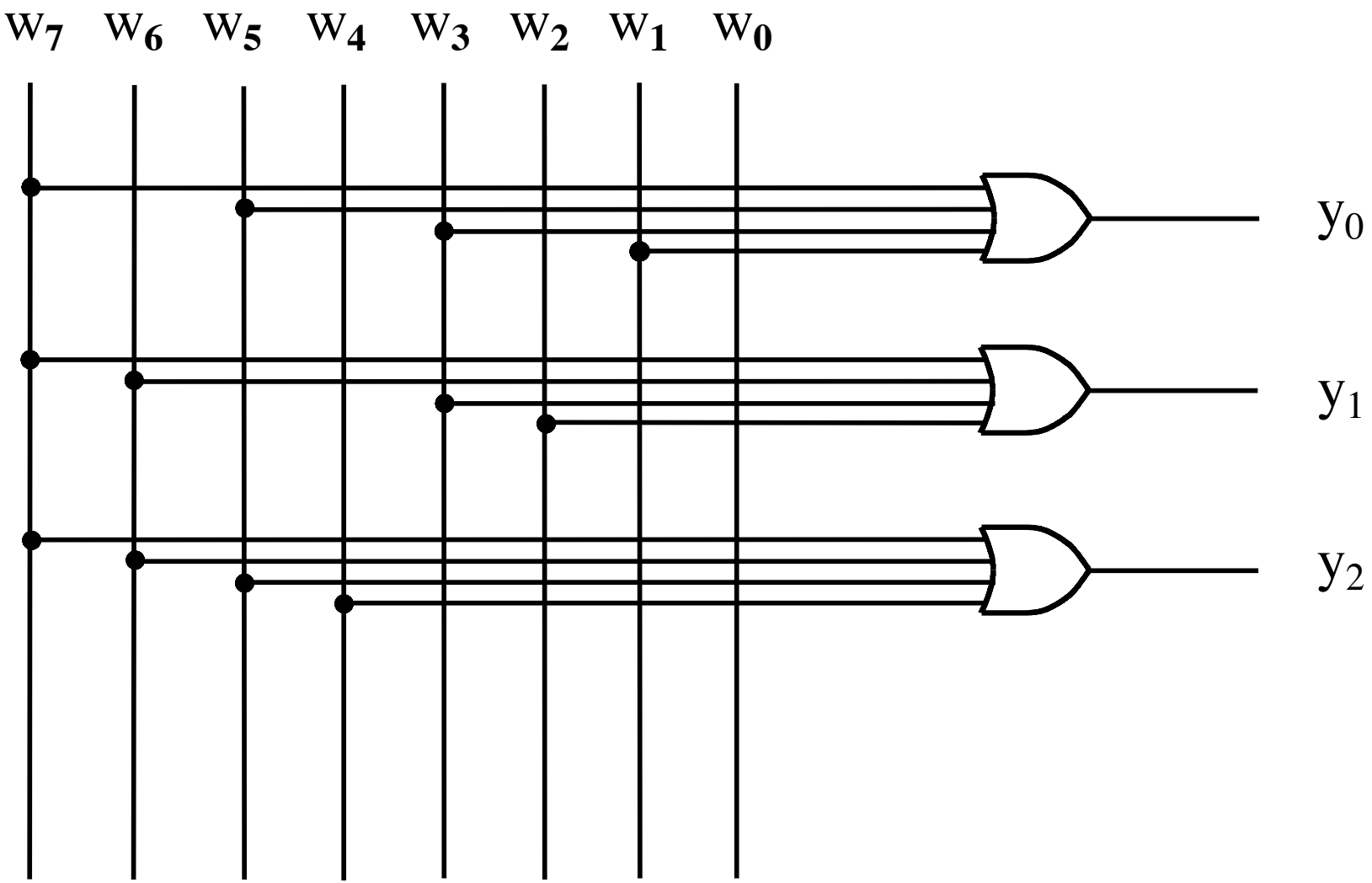
$i_6 = \overline{w_7}\,w_6$

$i_7 = w_7$

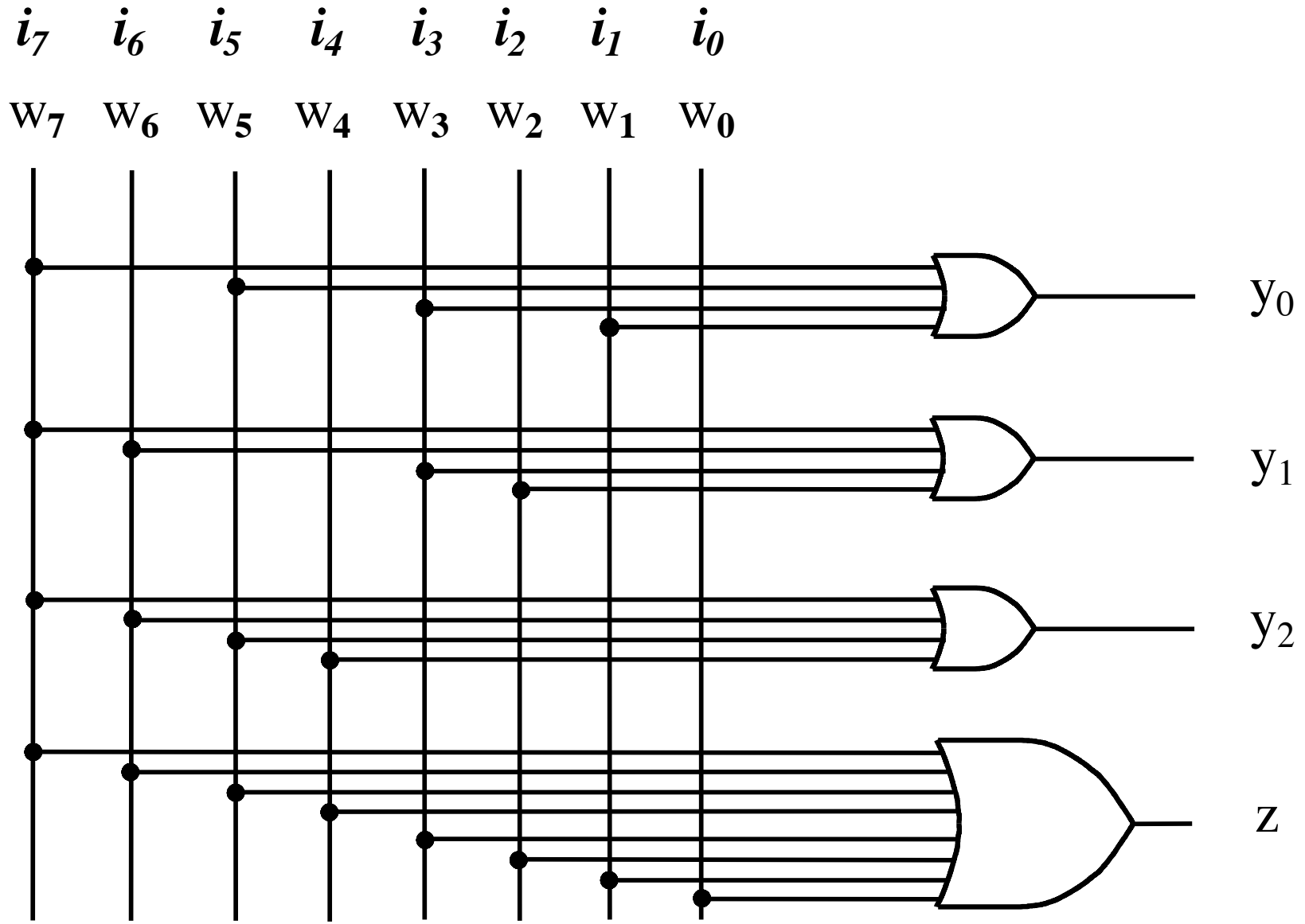
$z = i_0+i_1+i_2+i_3+i_4+i_5+i_6+i_7$

w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0	z
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	x	0	0	1	1
0	0	0	0	0	1	x	x	0	1	0	1
0	0	0	0	1	x	x	x	0	1	1	1
0	0	0	1	x	x	x	x	1	0	0	1
0	0	1	x	x	x	x	x	1	0	1	1
0	1	x	x	x	x	x	x	1	1	0	1
1	x	x	x	x	x	x	x	1	1	1	1
0	0	0	0	0	0	0	0	d	d	d	0

Circuit for the 8-to-3 binary encoder



Circuit for the 8-to-3 priority encoder



Example 4:Circuit implementation using a multiplexer

Implement the function

$$f(w_1, w_2, w_3, w_4, w_5) = \bar{w}_1\bar{w}_2\bar{w}_4\bar{w}_5 + w_1w_2 + w_1w_3 + w_1w_4 + w_3w_4w_5$$

using a 4-to-1 multiplexer

Some Boolean Algebra Leads To

$$\bar{w}_1\bar{w}_2\bar{w}_4\bar{w}_5 + w_1w_2 + w_1w_3 + w_1w_4 + w_3w_4w_5$$

$$\bar{w}_1\bar{w}_4\,(\bar{w}_5\,\bar{w}_2) + w_4\,(w_3w_5) + w_1\,(w_2 + w_3) + w_1w_4\,(1)$$

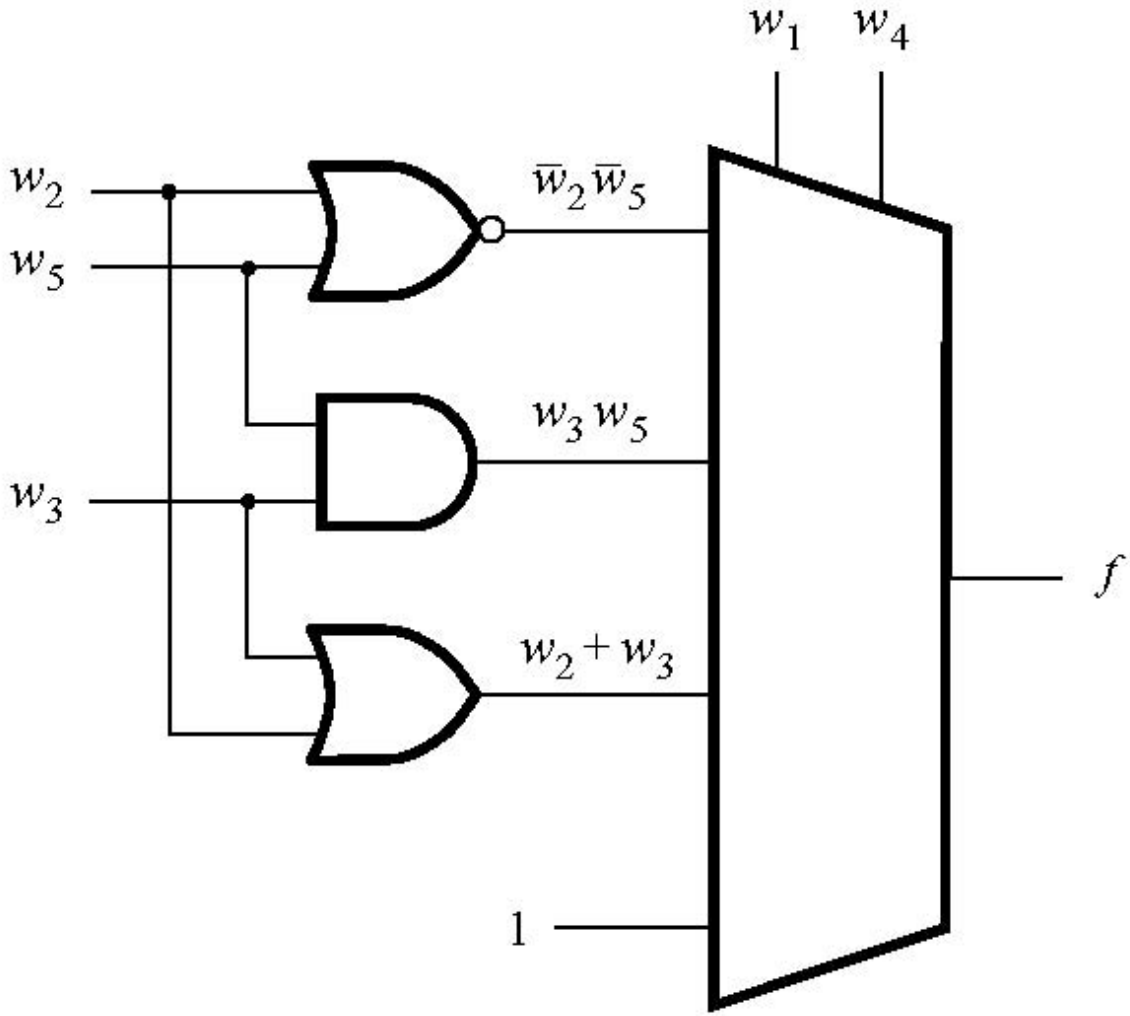
$$\bar{w}_1\bar{w}_4\,(\bar{w}_5\,\bar{w}_2) + (\,\bar{w}_1 + w_1)w_4\,(w_3w_5) + w_1(\,\bar{w}_4 + w_4)\,(w_2 + w_3) + w_1w_4\,(1)$$

$$\bar{w}_1\bar{w}_4\,(\bar{w}_5\,\bar{w}_2) + \bar{w}_1w_4\,(w_3w_5) + w_1\bar{w}_4\,(w_2 + w_3) + w_1w_4\,(\,w_3w_5 + (w_2 + w_3) + 1)$$

$$\bar{w}_1\bar{w}_4\,(\bar{w}_5\,\bar{w}_2) + \bar{w}_1w_4\,(w_3w_5) + w_1\bar{w}_4\,(w_2 + w_3) + w_1w_4\,(1)$$

Note that the split is by w_1 and w_4, not w_1 and w_2

Solution Circuit



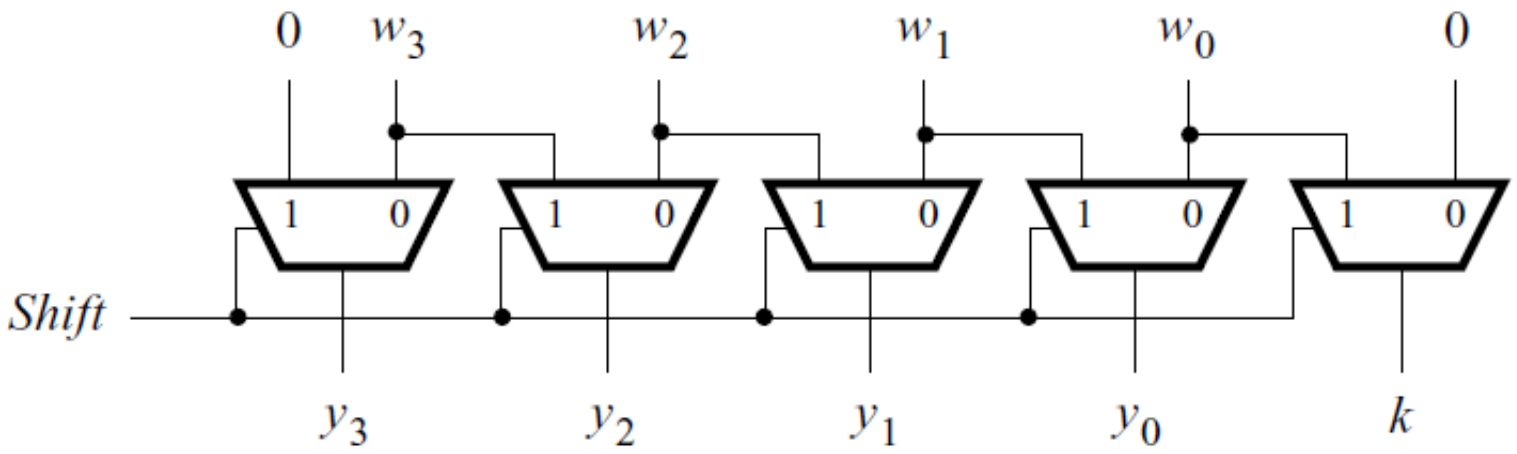
$$\overline{w}_1\overline{w}_4(\overline{w}_5\overline{w}_2) + \overline{w}_1 w_4(w_3 w_5) + w_1\overline{w}_4(w_2 + w_3) + w_1 w_4(1)$$

[Figure 4.46 from the textbook]

Some Final Things from Chapter 4

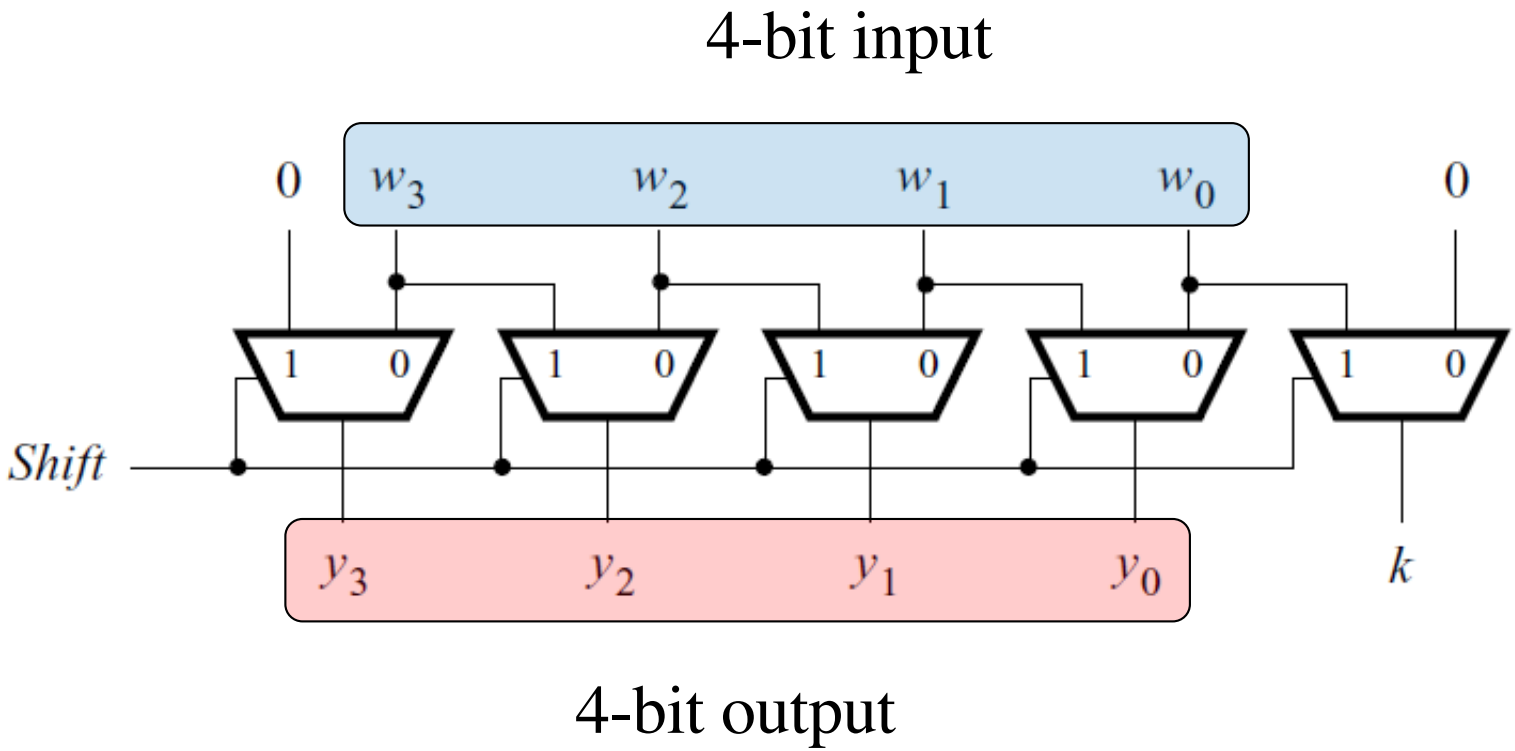
Shifter Circuit

Hold / Shift-Right Circuit



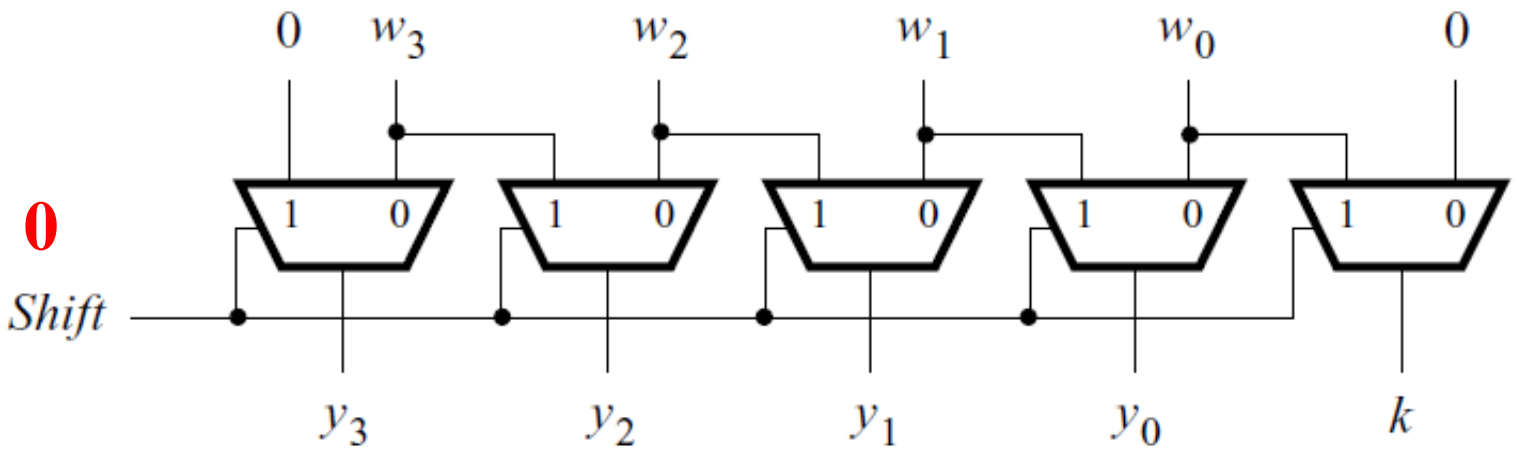
[Figure 4.50 from the textbook]

Hold / Shift-Right Circuit

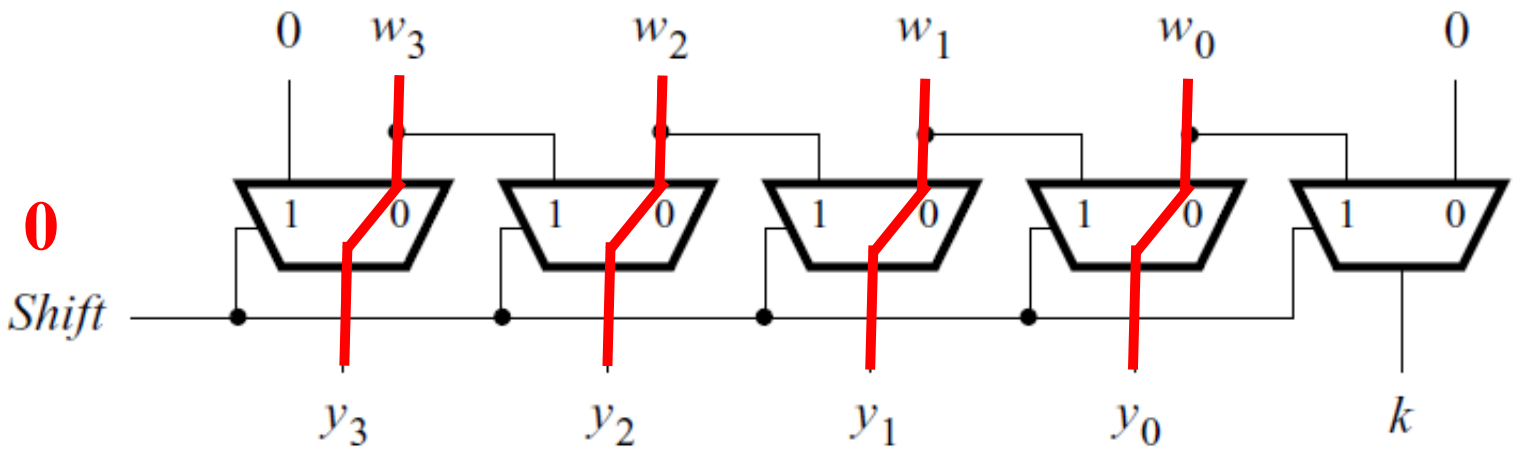


[Figure 4.50 from the textbook]

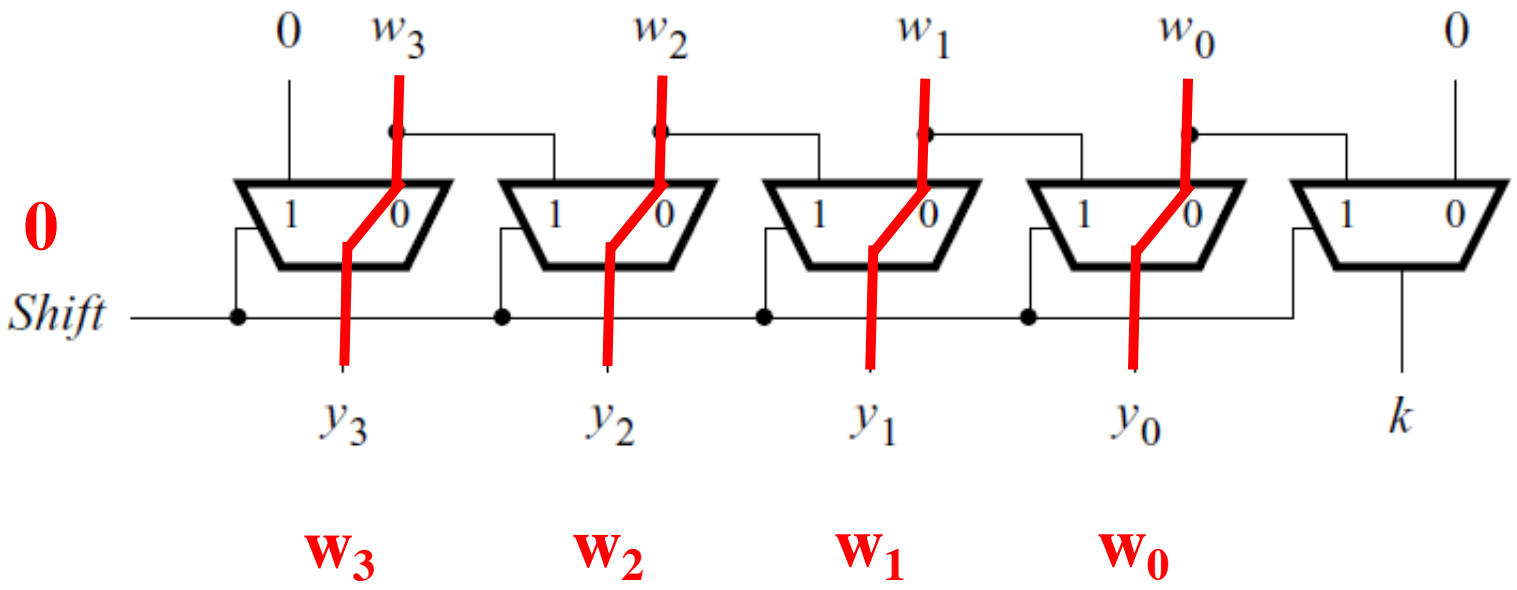
No shift if the control signal is 0



No shift if the control signal is 0

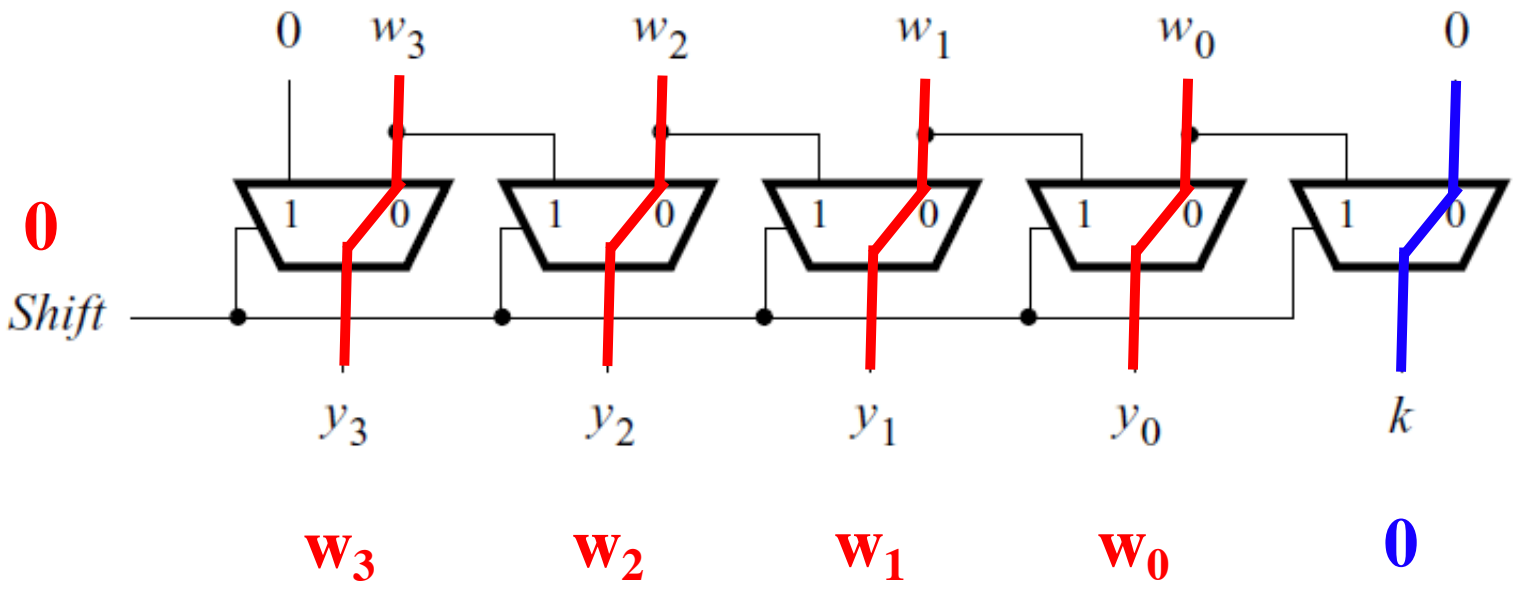


No shift if the control signal is 0



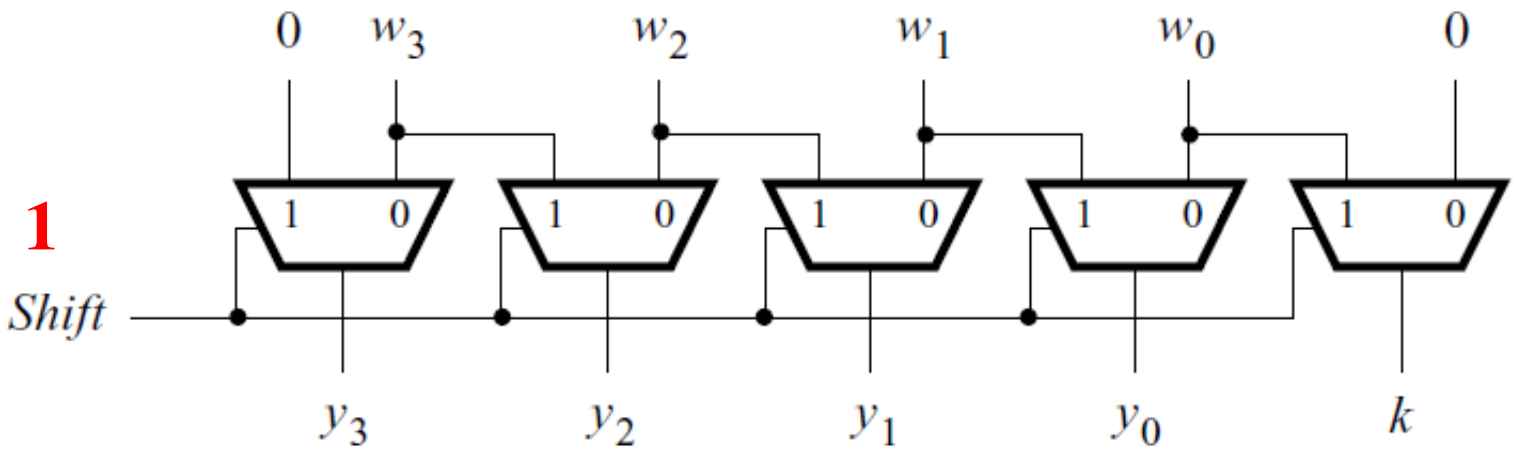
No shift in this case.

No shift if the control signal is 0

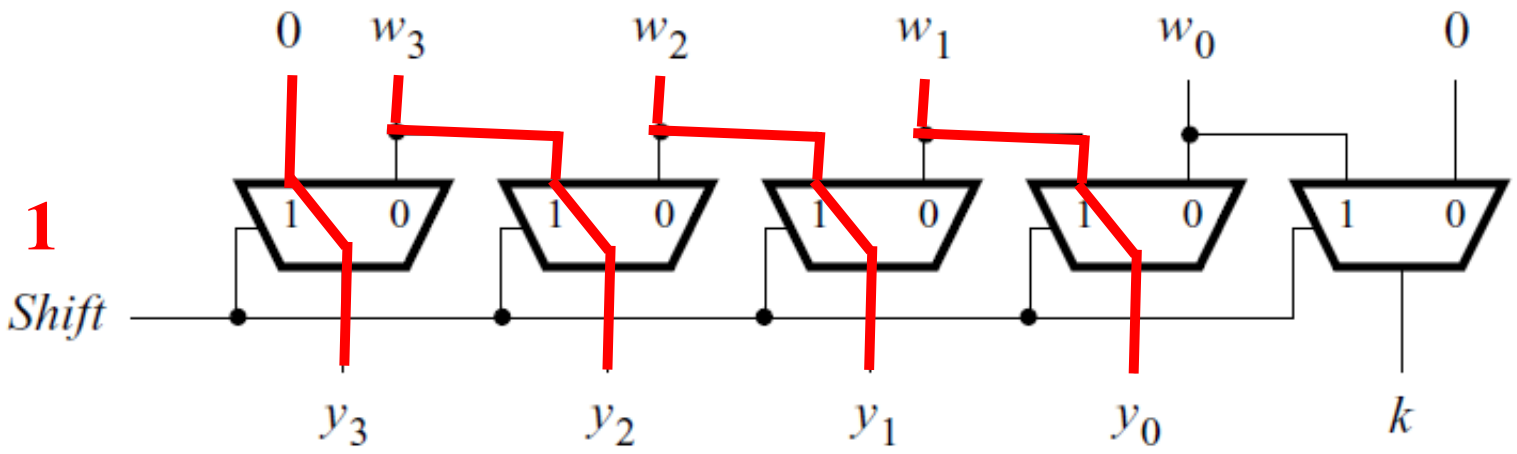


No shift in this case.

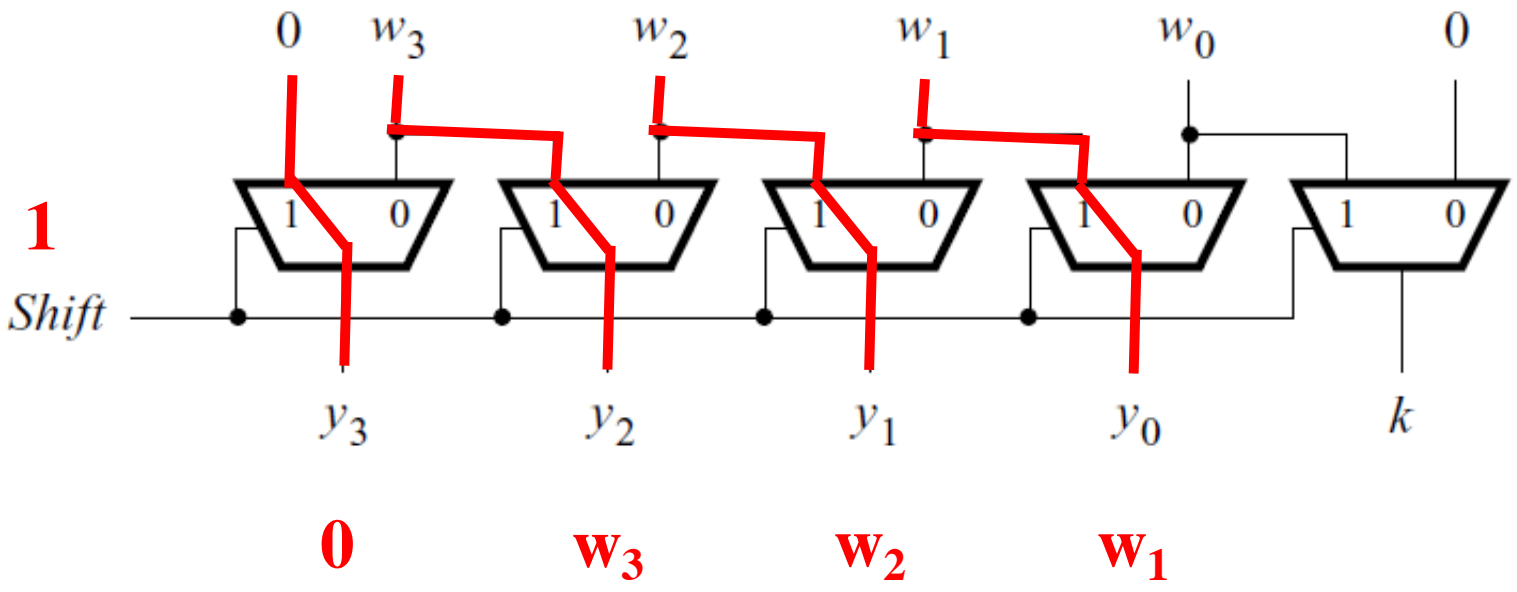
Shift right if the control signal is 1



Shift right if the control signal is 1

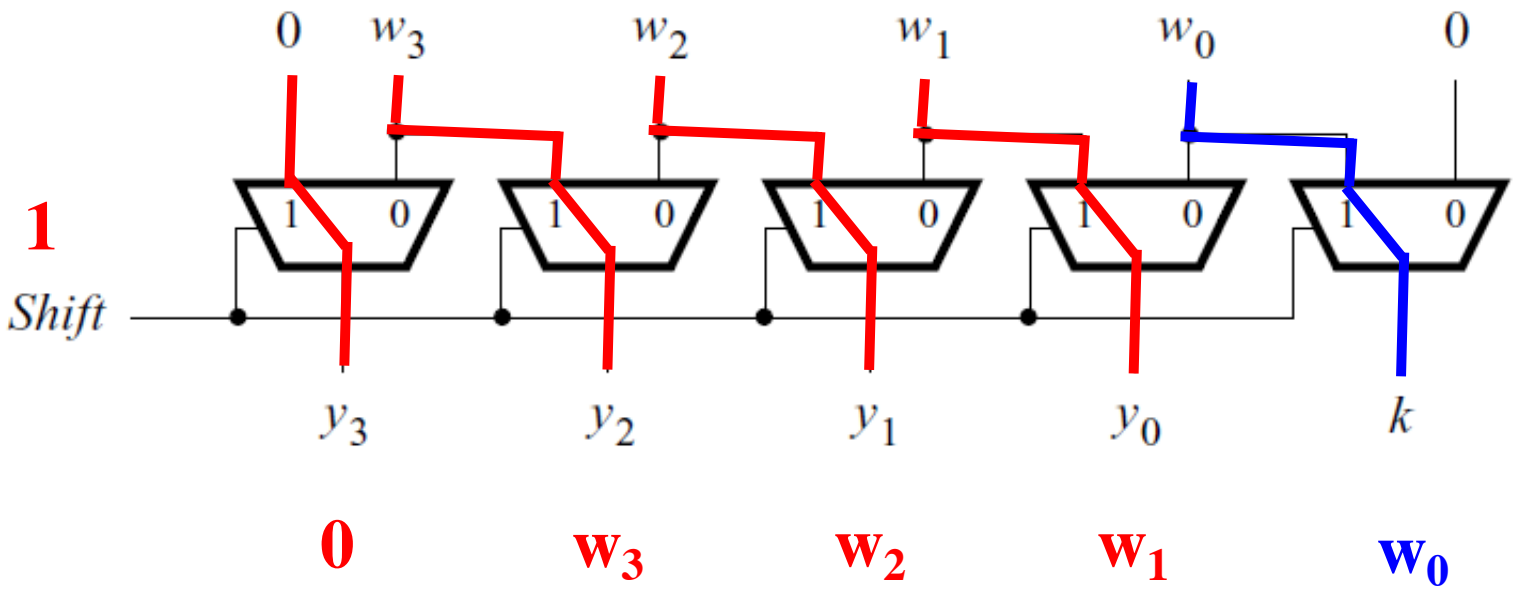


Shift right if the control signal is 1



Shift to the right by 1 bit

A shifter circuit



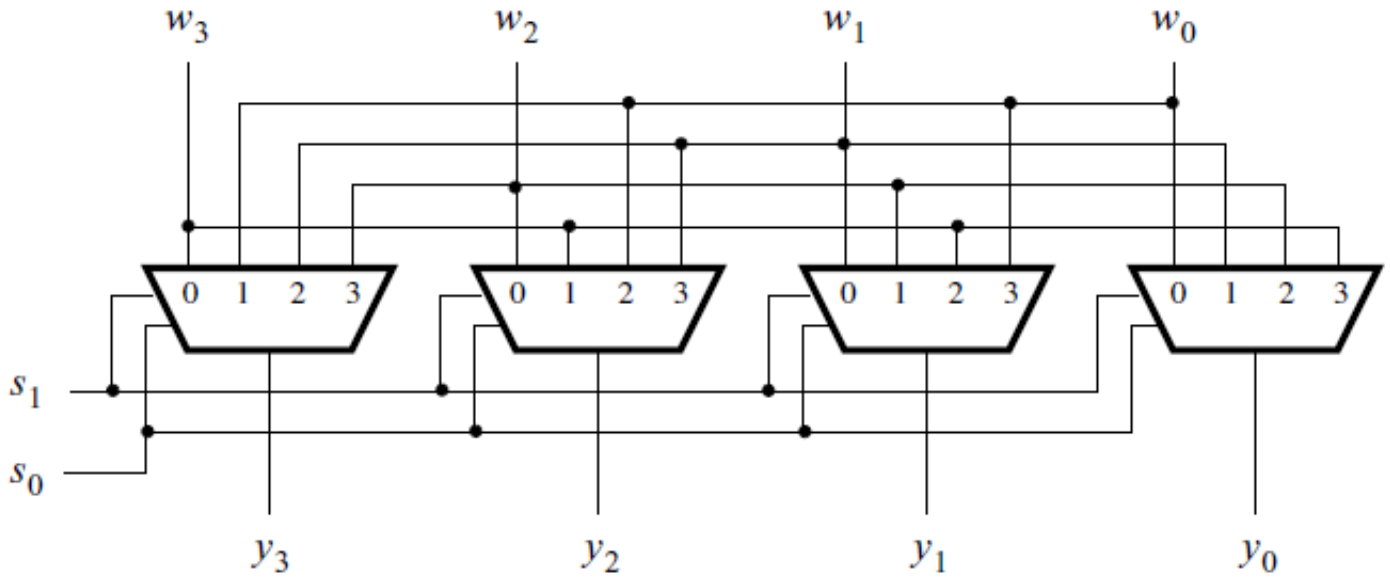
Shift to the right by 1 bit

Barrel Shifter

A barrel shifter circuit

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

(a) Truth table



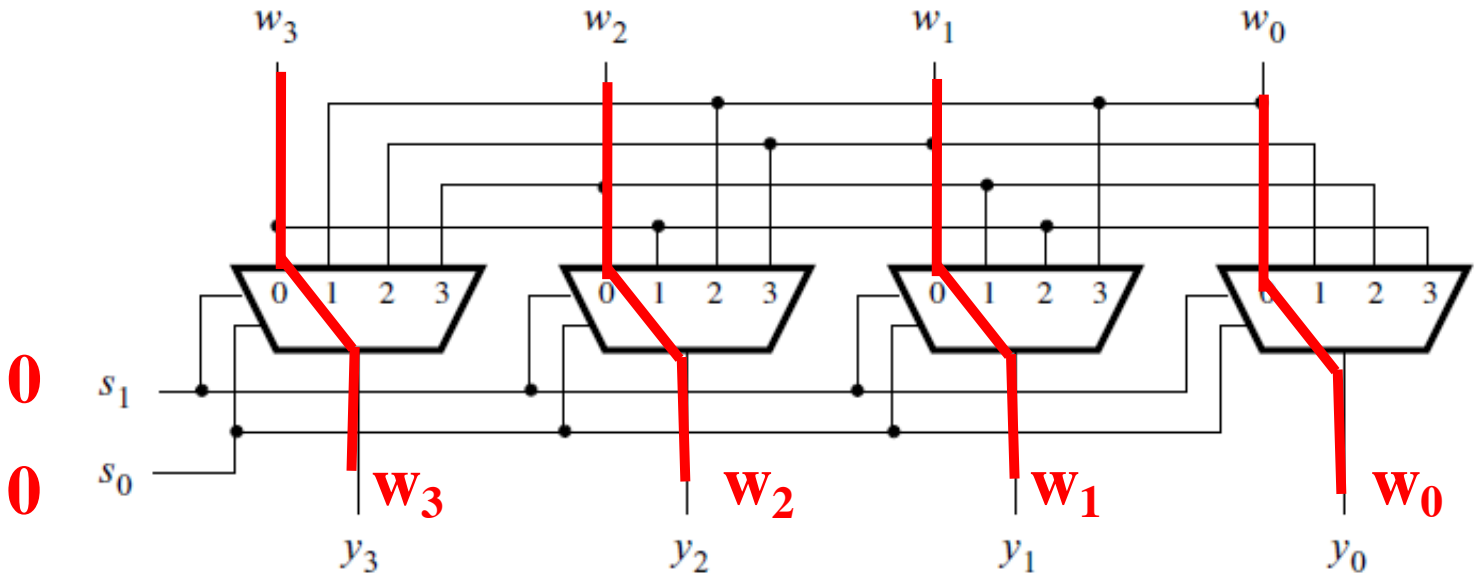
(b) Circuit

[Figure 4.51 from the textbook]

A barrel shifter circuit

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

(a) Truth table



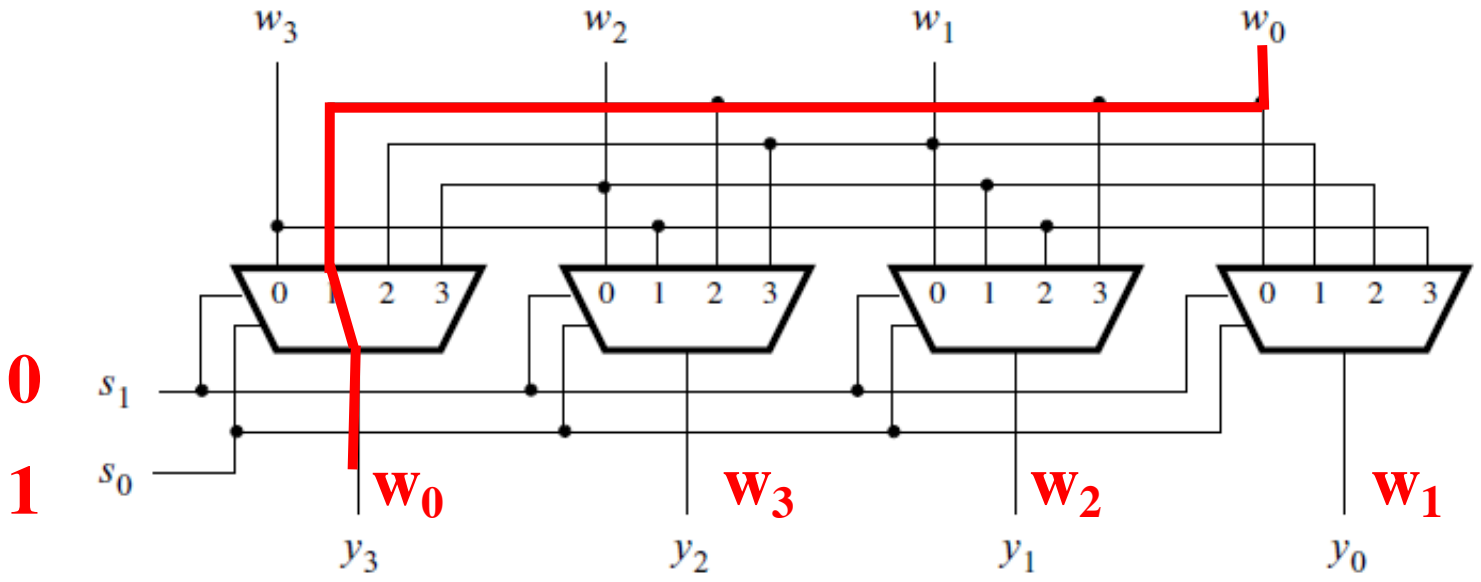
(b) Circuit

[Figure 4.51 from the textbook]

A barrel shifter circuit

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

(a) Truth table



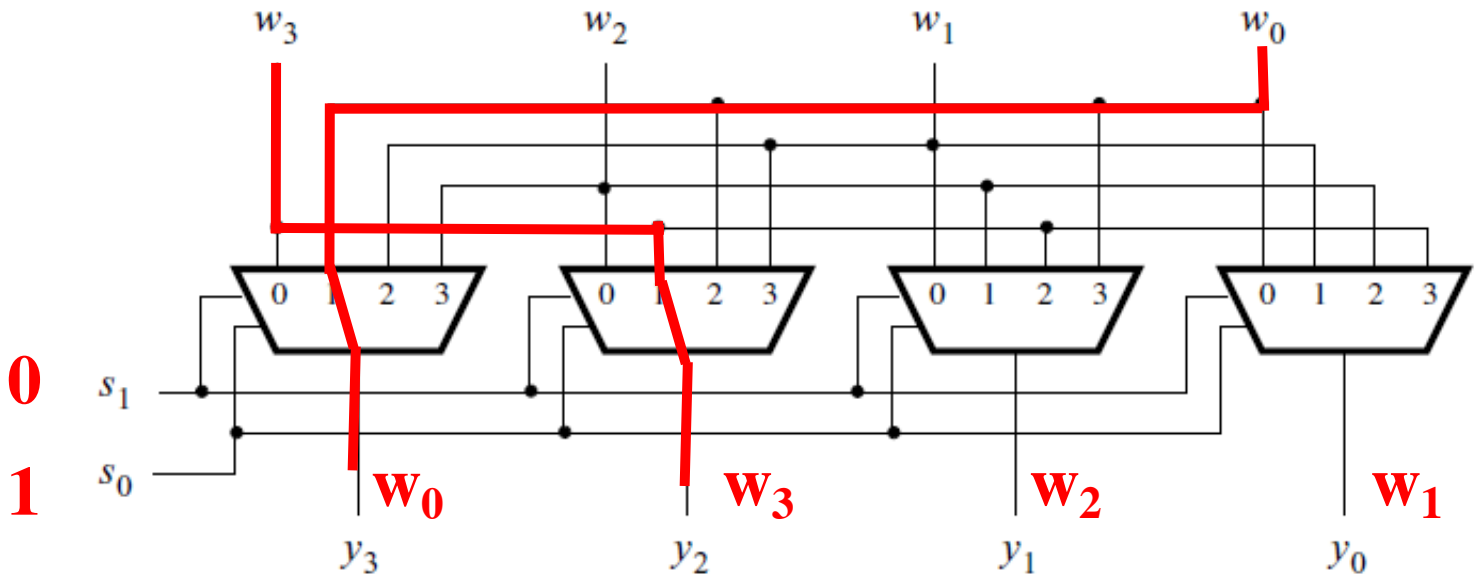
(b) Circuit

[Figure 4.51 from the textbook]

A barrel shifter circuit

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

(a) Truth table



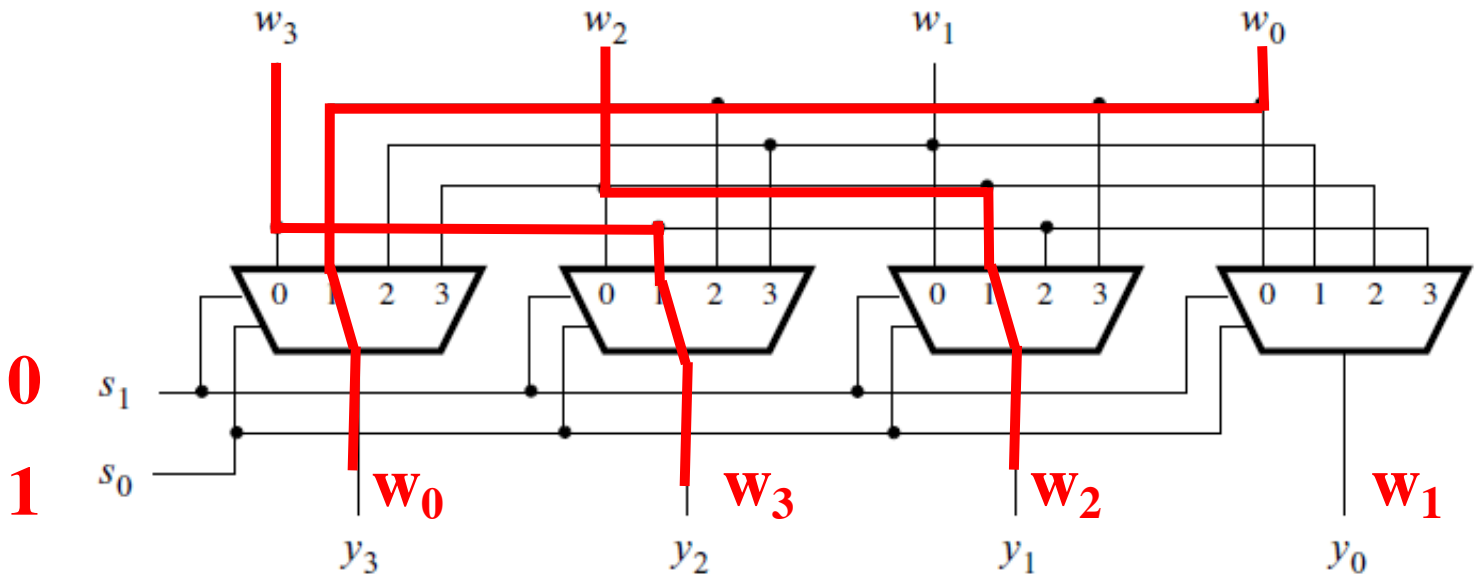
(b) Circuit

[Figure 4.51 from the textbook]

A barrel shifter circuit

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

(a) Truth table



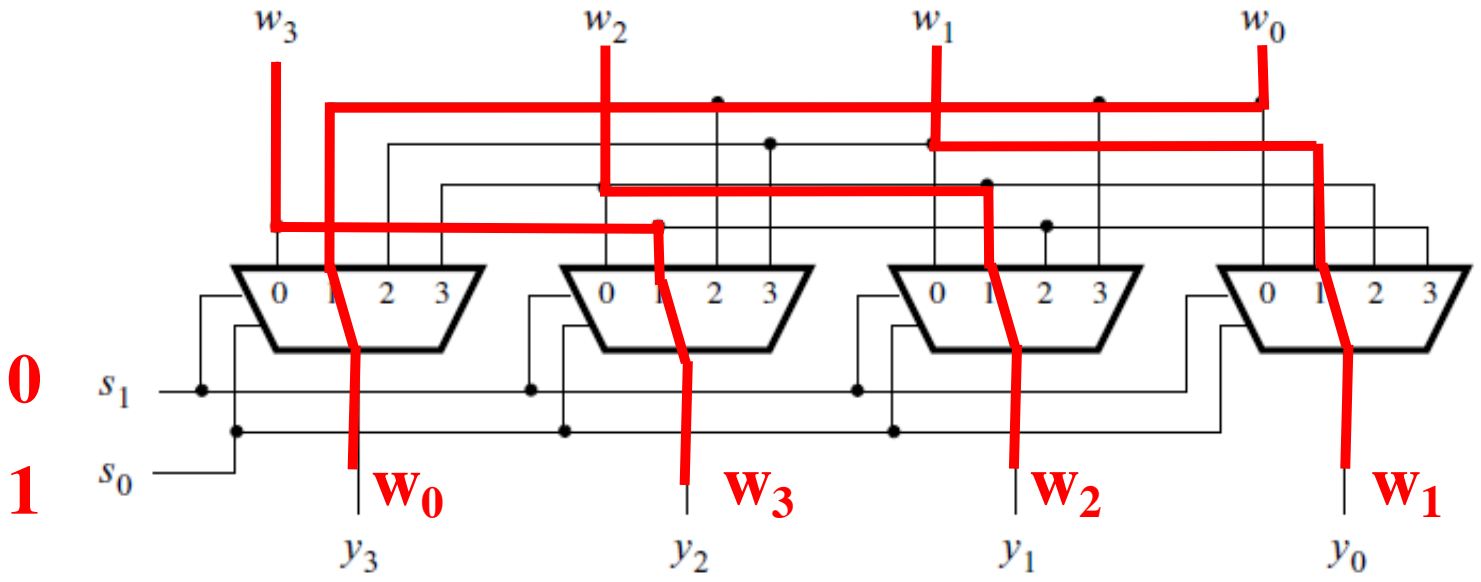
(b) Circuit

[Figure 4.51 from the textbook]

A barrel shifter circuit

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

(a) Truth table



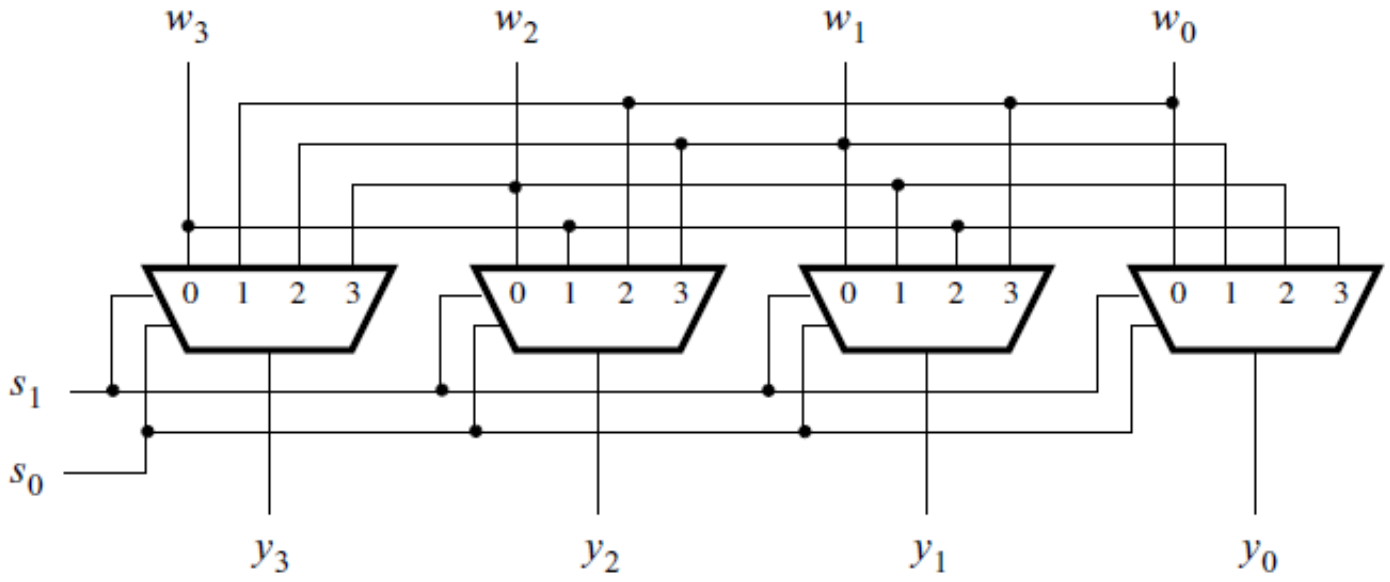
(b) Circuit

[Figure 4.51 from the textbook]

A barrel shifter circuit

s_1	s_0	y_3	y_2	y_1	y_0
0	0	w_3	w_2	w_1	w_0
0	1	w_0	w_3	w_2	w_1
1	0	w_1	w_0	w_3	w_2
1	1	w_2	w_1	w_0	w_3

(a) Truth table

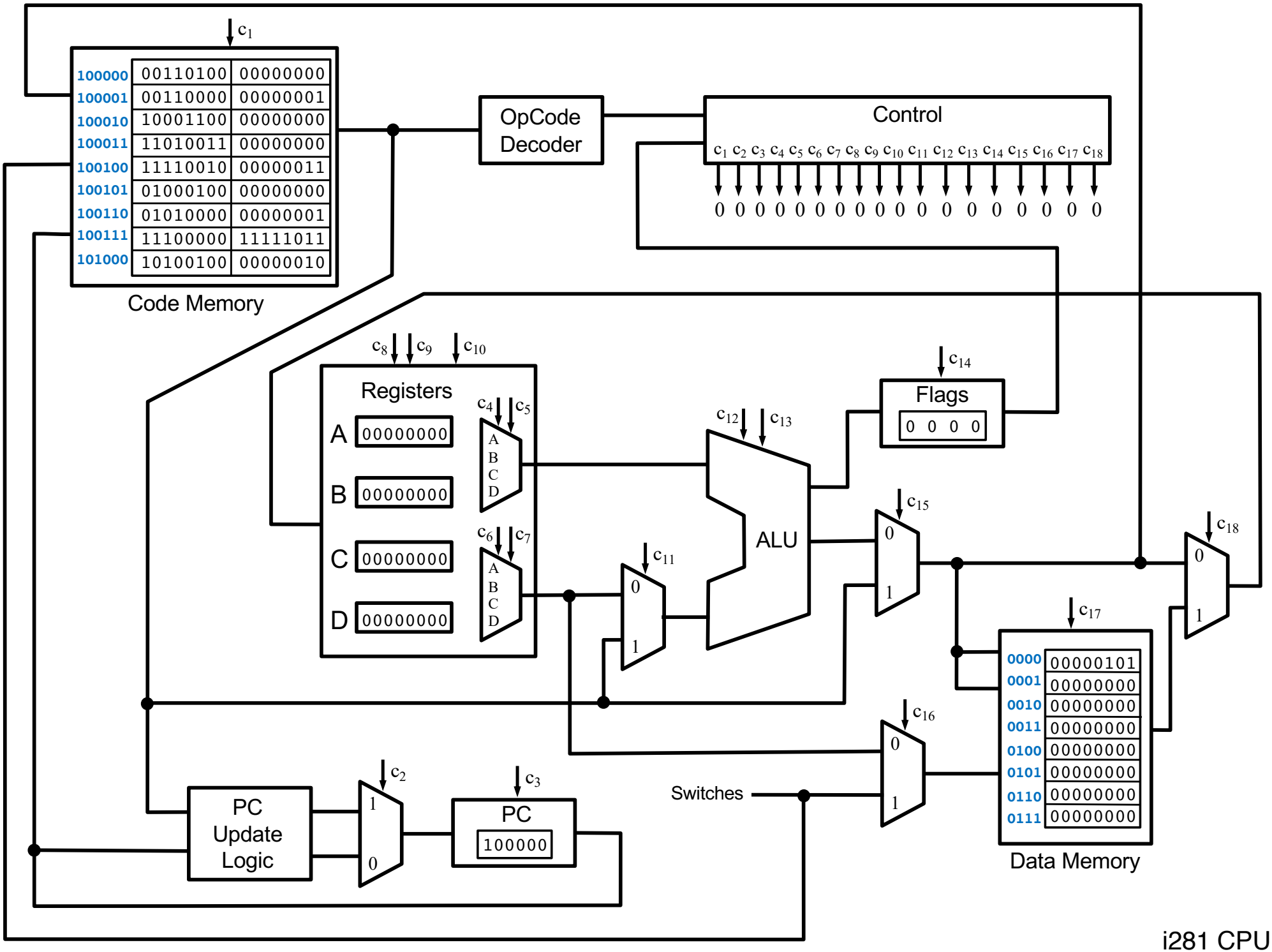


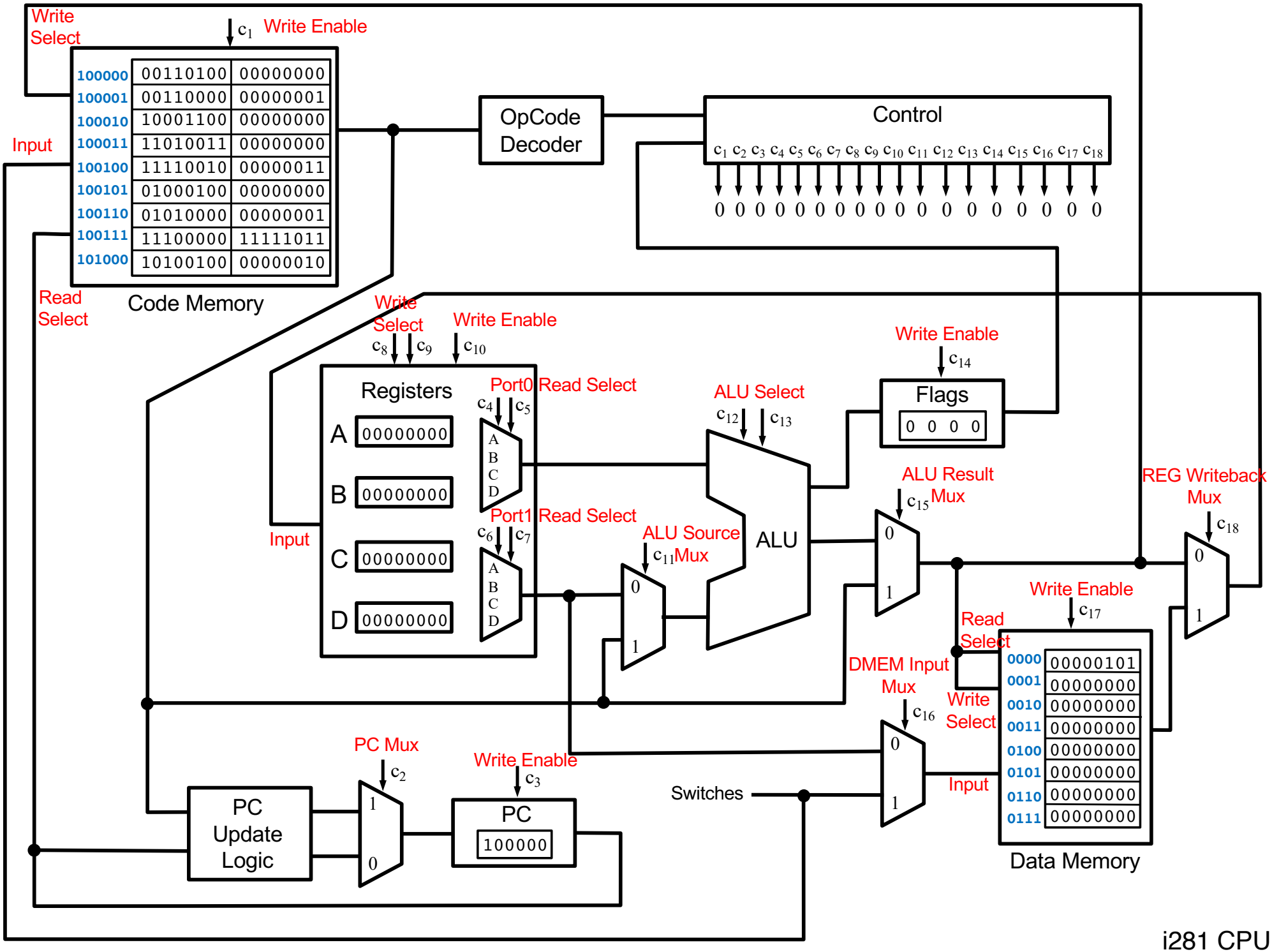
(b) Circuit

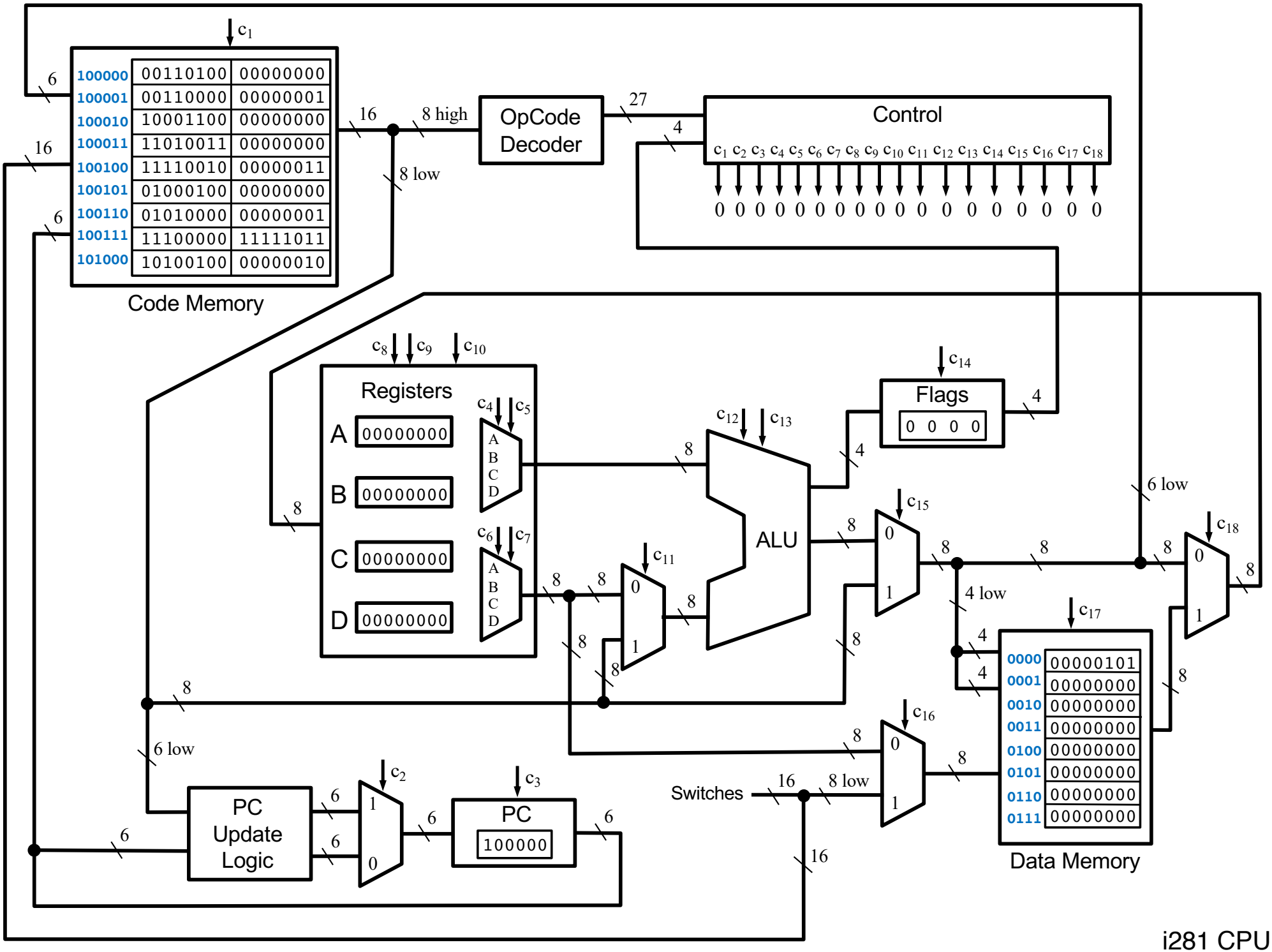
[Figure 4.51 from the textbook]

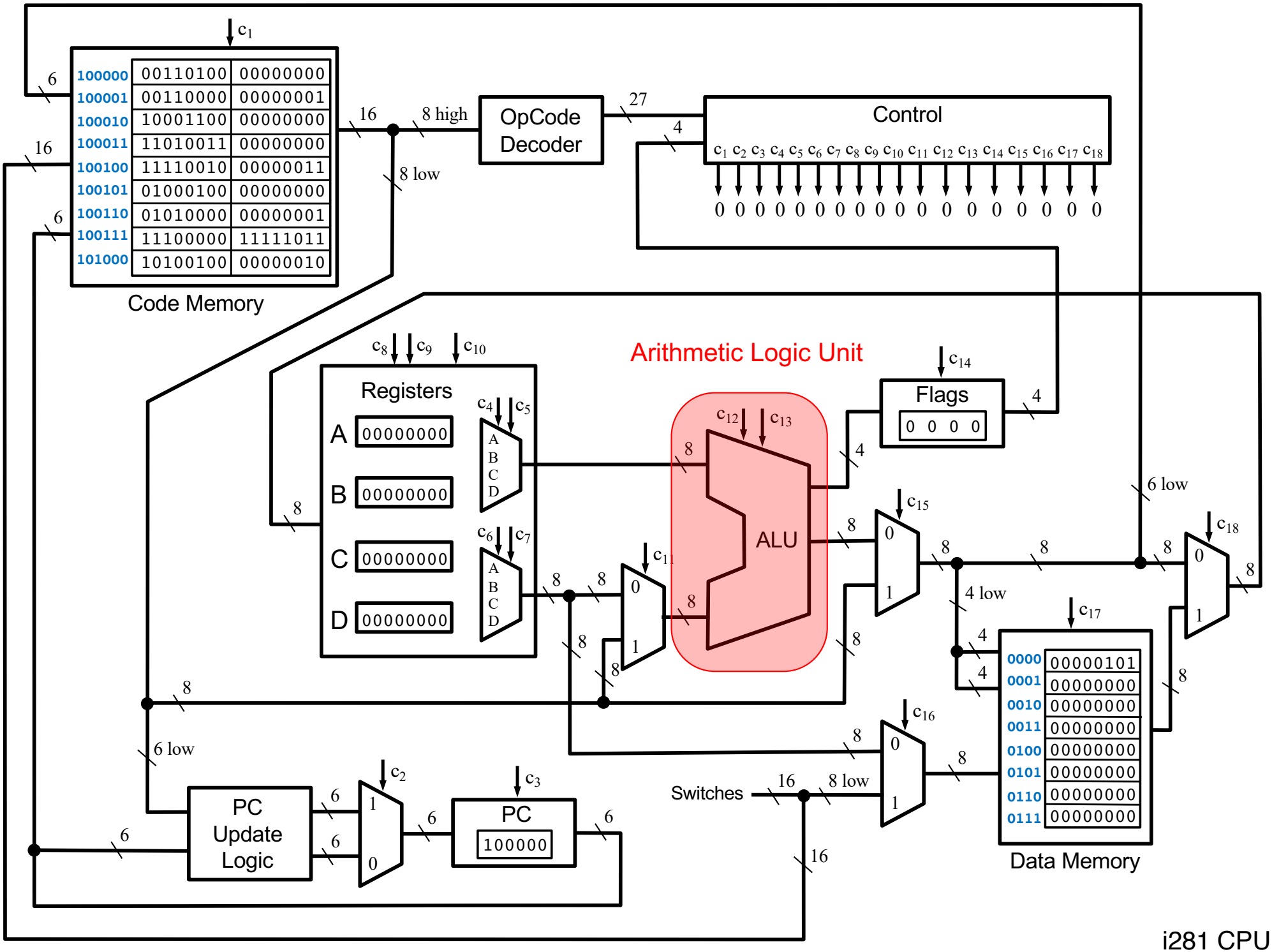
Arithmetic Logic Unit (ALU)

Shifter Circuit of the i281 CPU

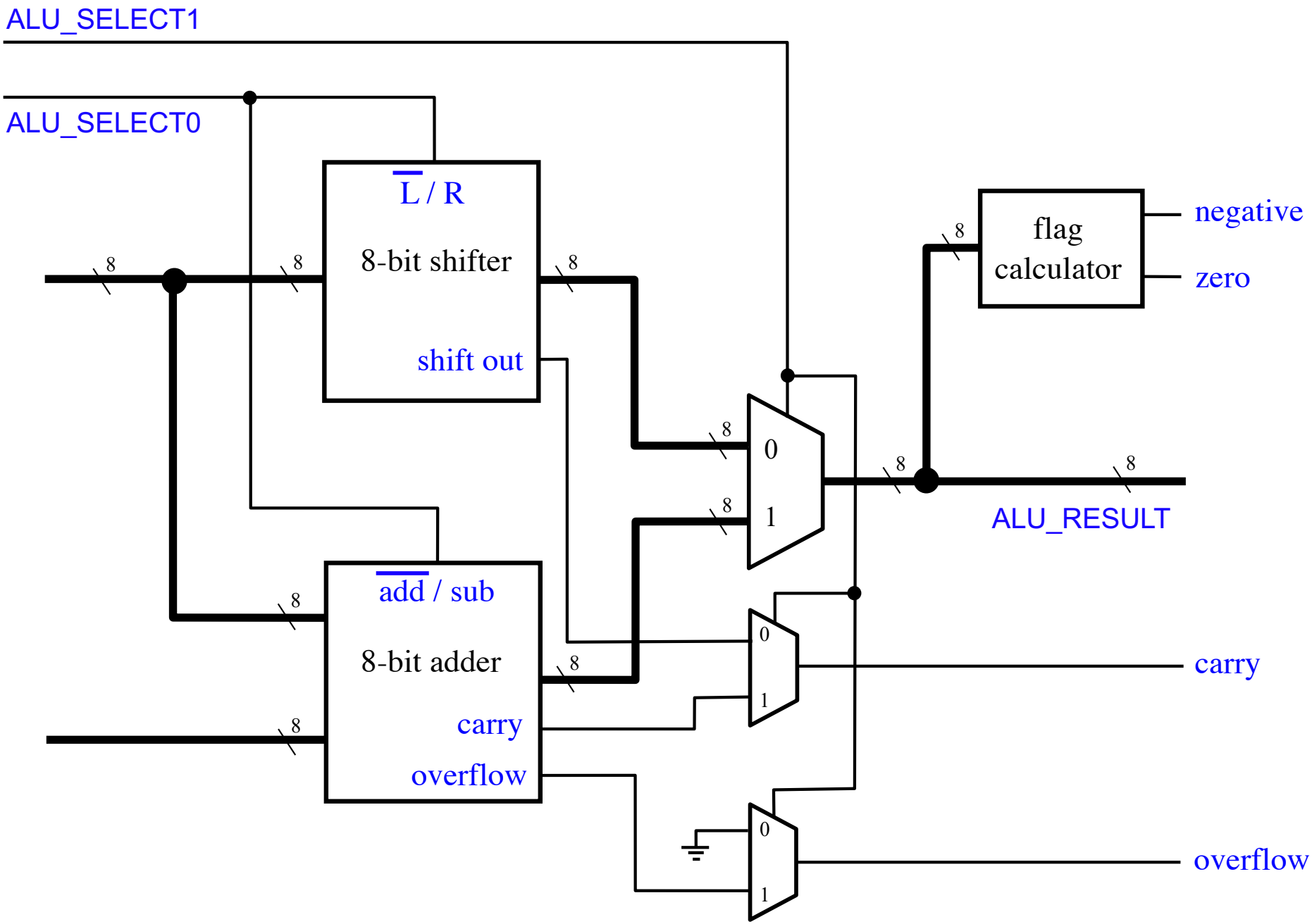

c1
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
c14
Flags
0 0 0 0
c12 c13
ALU
c15
c11
0 1
c18
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
c2
c3
PC Update Logic
PC
100000
i281 CPU


Write Select
c1 Write Enable
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
Input
Read Select
OpCode Decoder
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Write Select
c8 c9
Write Enable
c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
Port0 Read Select
c4 c5
Port1 Read Select
c6 c7
A B C D
Input
ALU Select
c12 c13
ALU
ALU Source
c11 Mux
Write Enable
c14
Flags
0 0 0 0
ALU Result
c15 Mux
REG Writeback
Mux
c18
Write Enable
c17
Read Select
Write Select
DMEM Input
Mux
c16
Input
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
PC Mux
c2
Write Enable
c3
PC
100000
PC Update Logic
Switches
i281 CPU

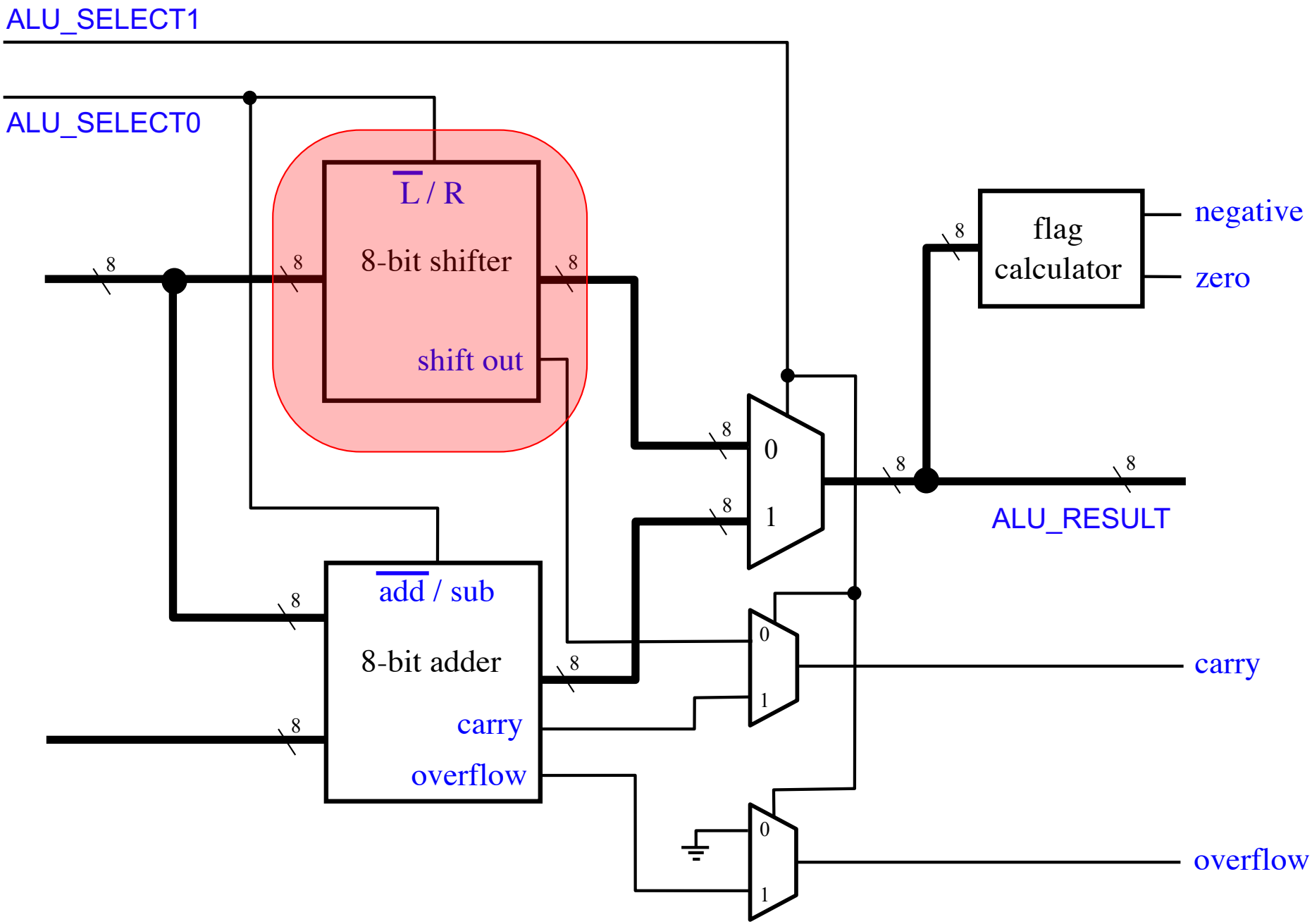

c1
6
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
8
c14
Flags
0 0 0 0
4
c12 c13
ALU
c11
c15
c16
c17
c18
6 low
4 low
Switches
16
8 low
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c2
c3
PC Update Logic
PC
100000
i281 CPU


c1
6
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Arithmetic Logic Unit
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
c12 c13
ALU
c14
Flags
0 0 0 0
4
c11
c15
c18
6 low
4 low
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
8 low
c2
c3
PC Update Logic
PC
100000
i281 CPU

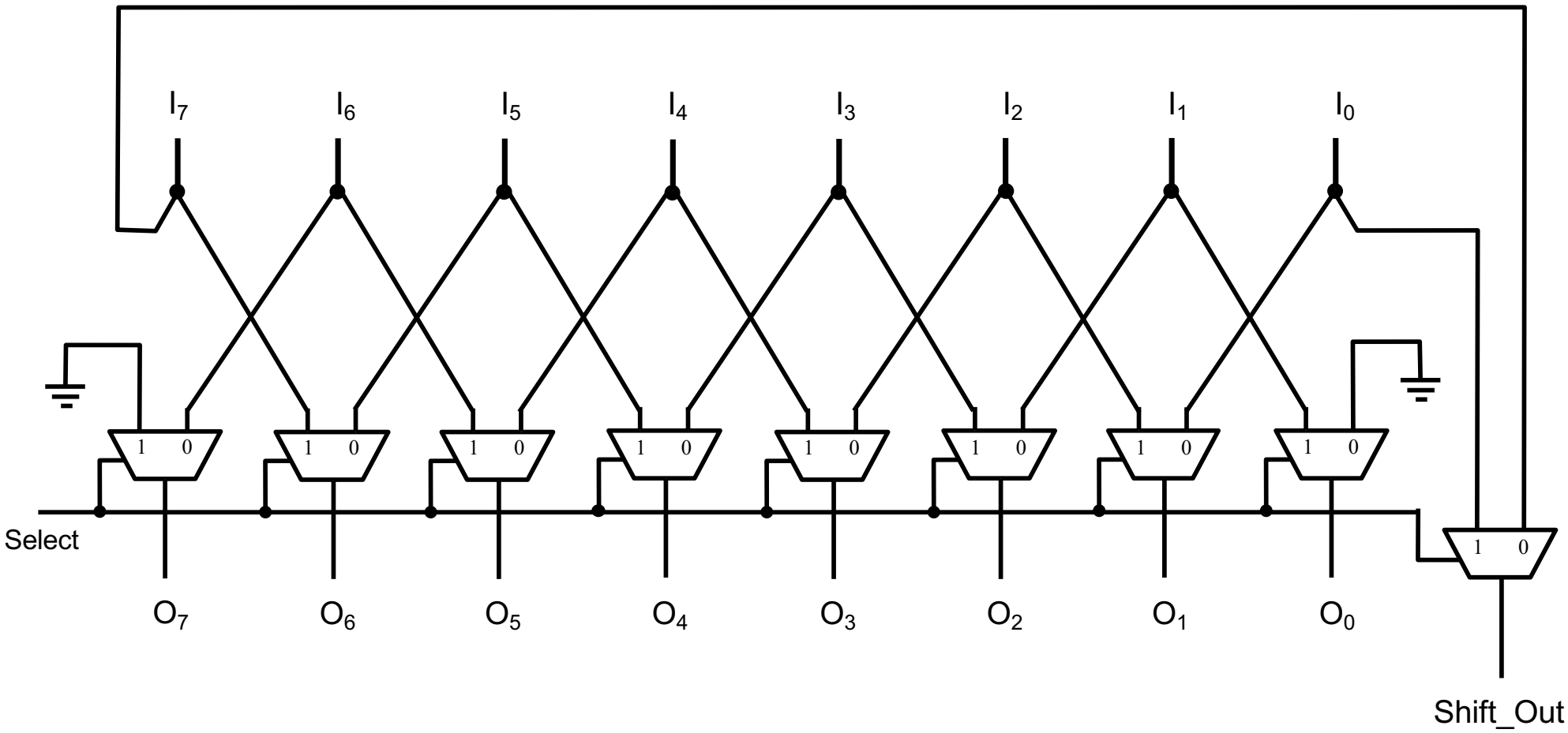
The ALU

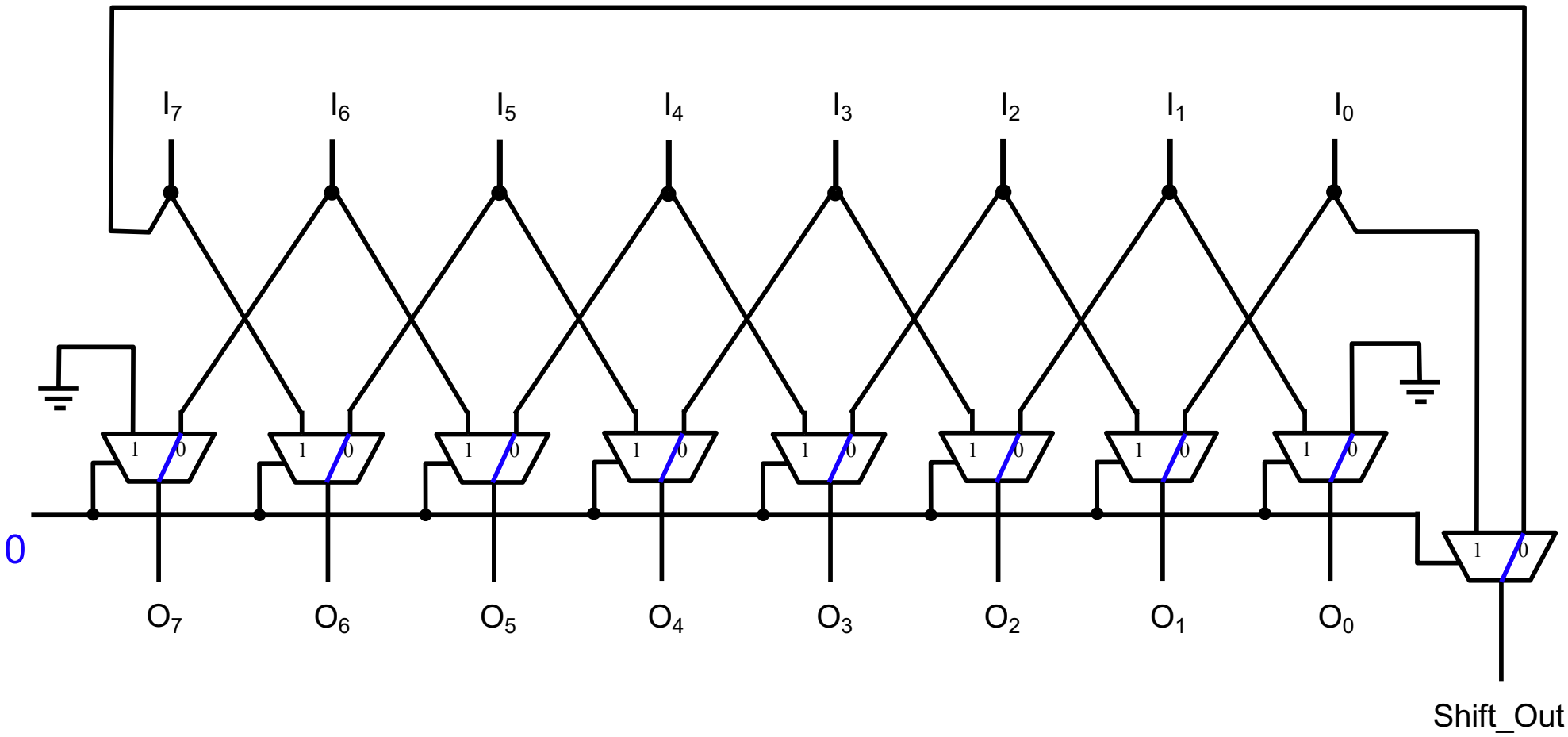

ALU_SELECT1
ALU_SELECT0
L / R
8-bit shifter
shift out
8
add / sub
8-bit adder
carry
overflow
flag
calculator
negative
zero
0
1
ALU_RESULT
carry
overflow

The Shifter Circuit

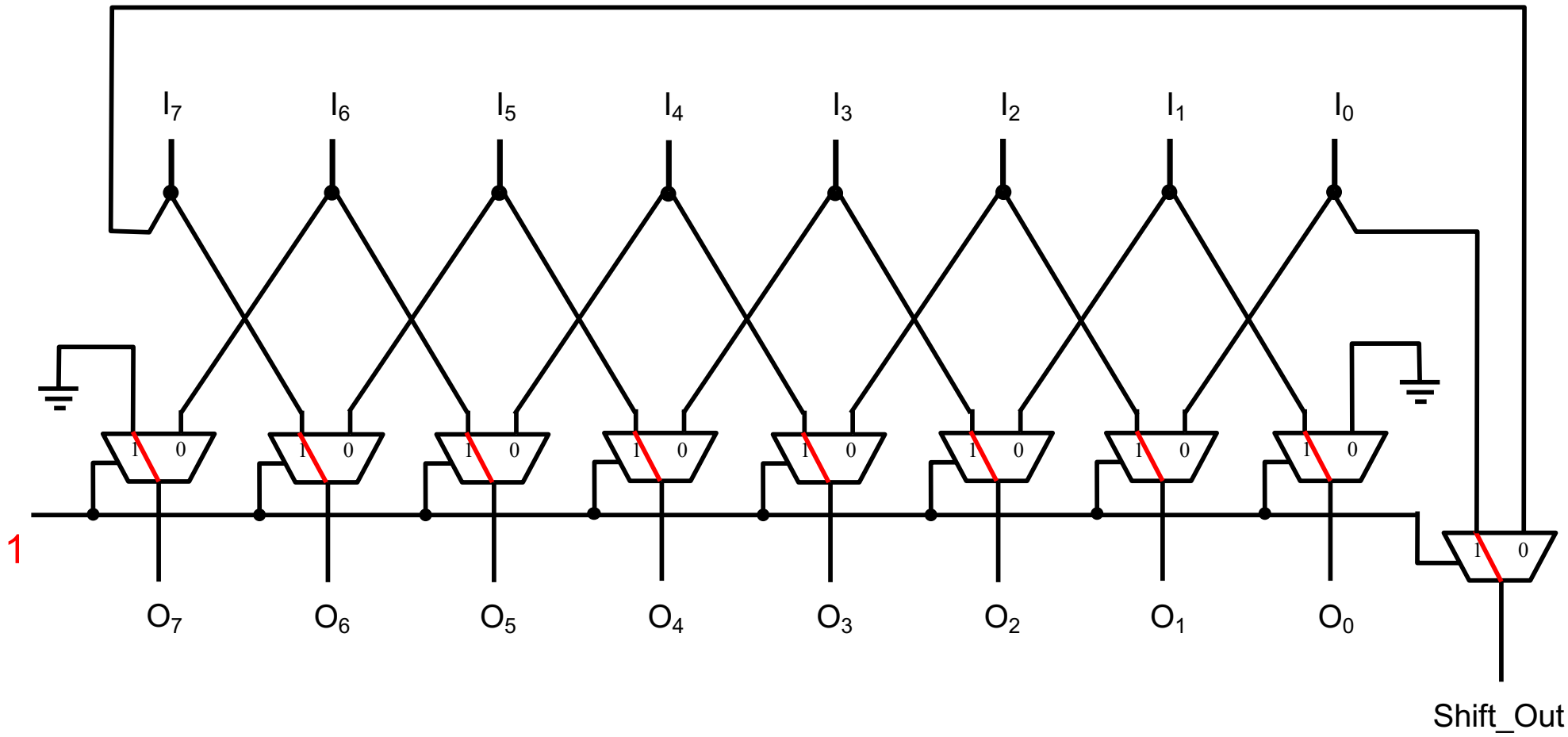


The i281 CPU Shifter Circuit



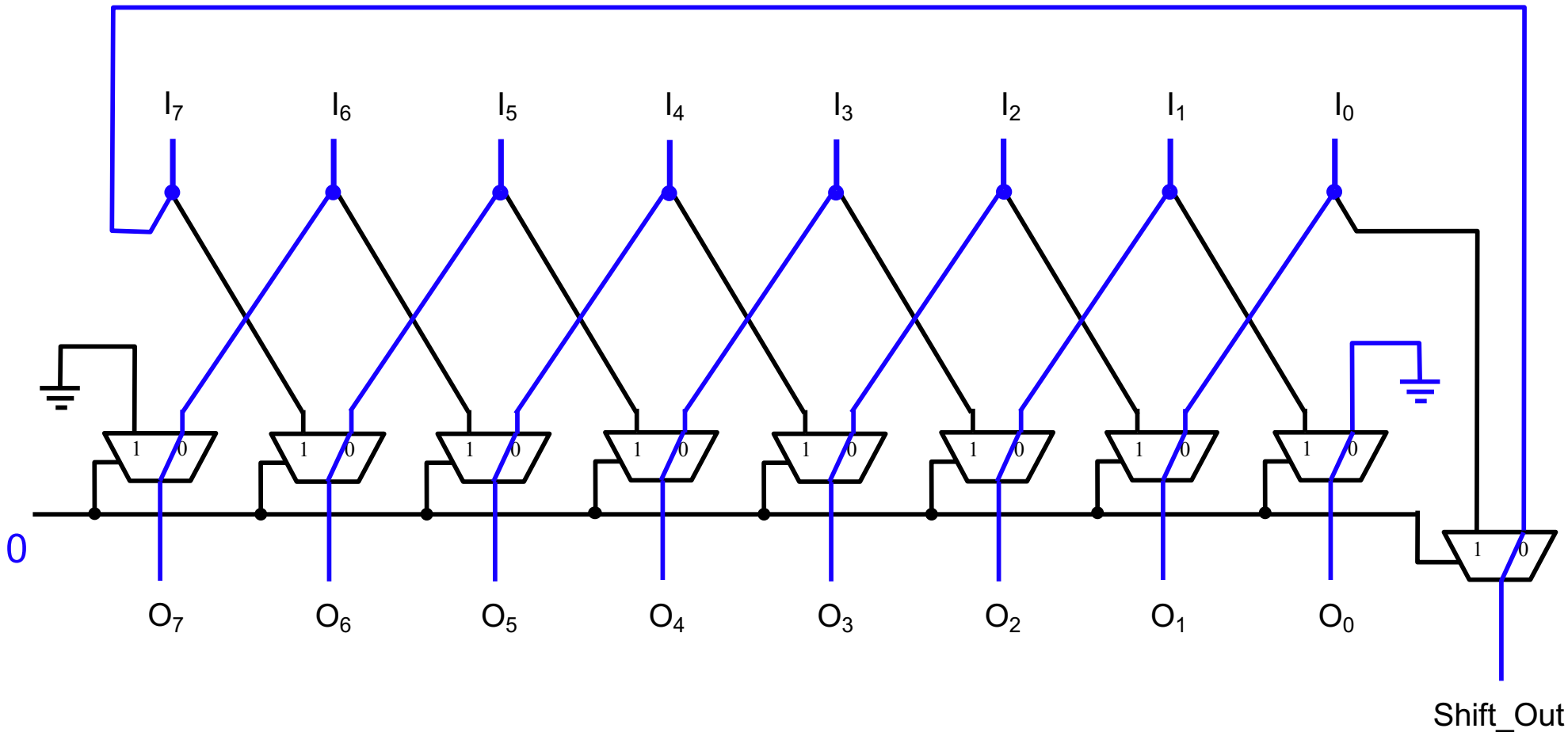


i281 CPU

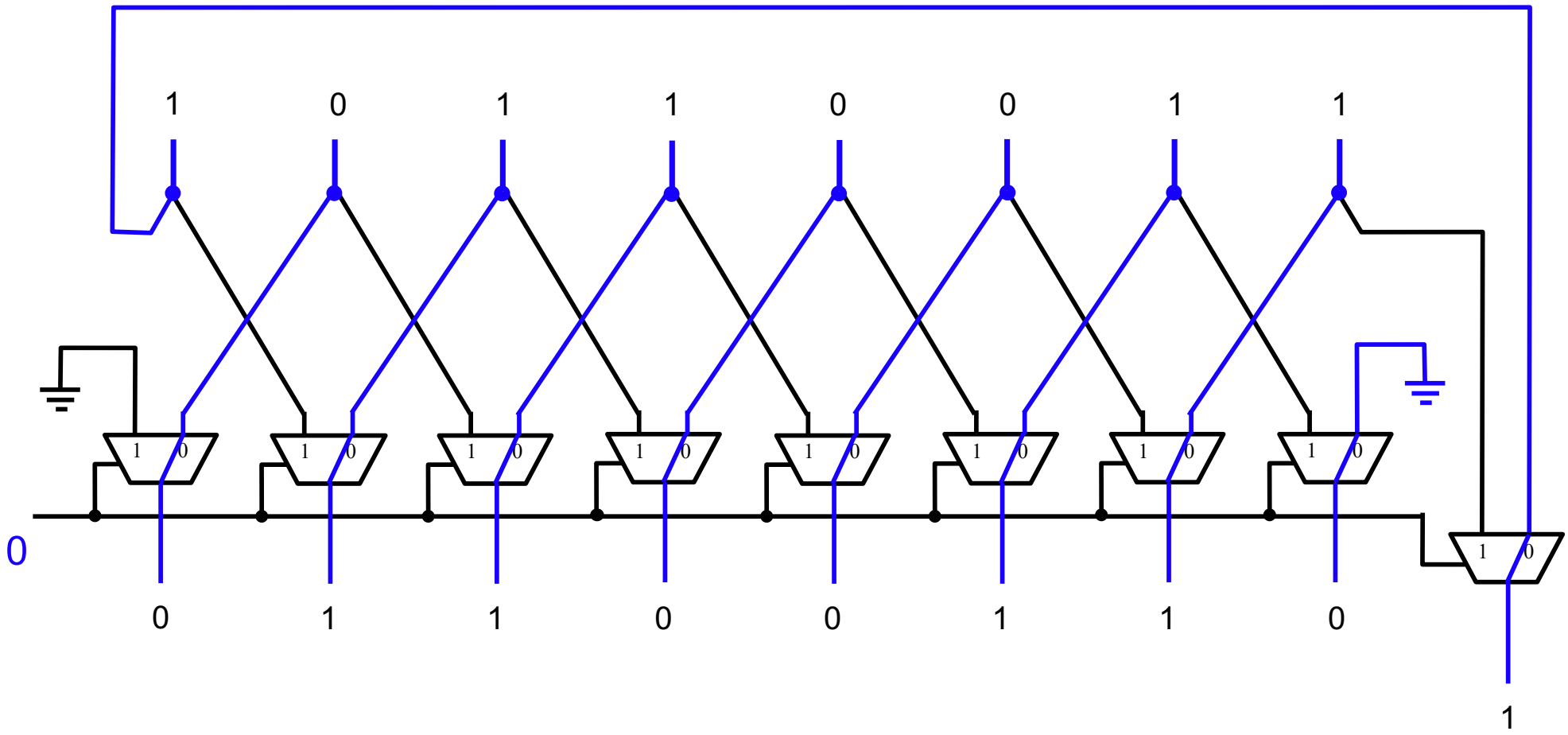


i281 CPU

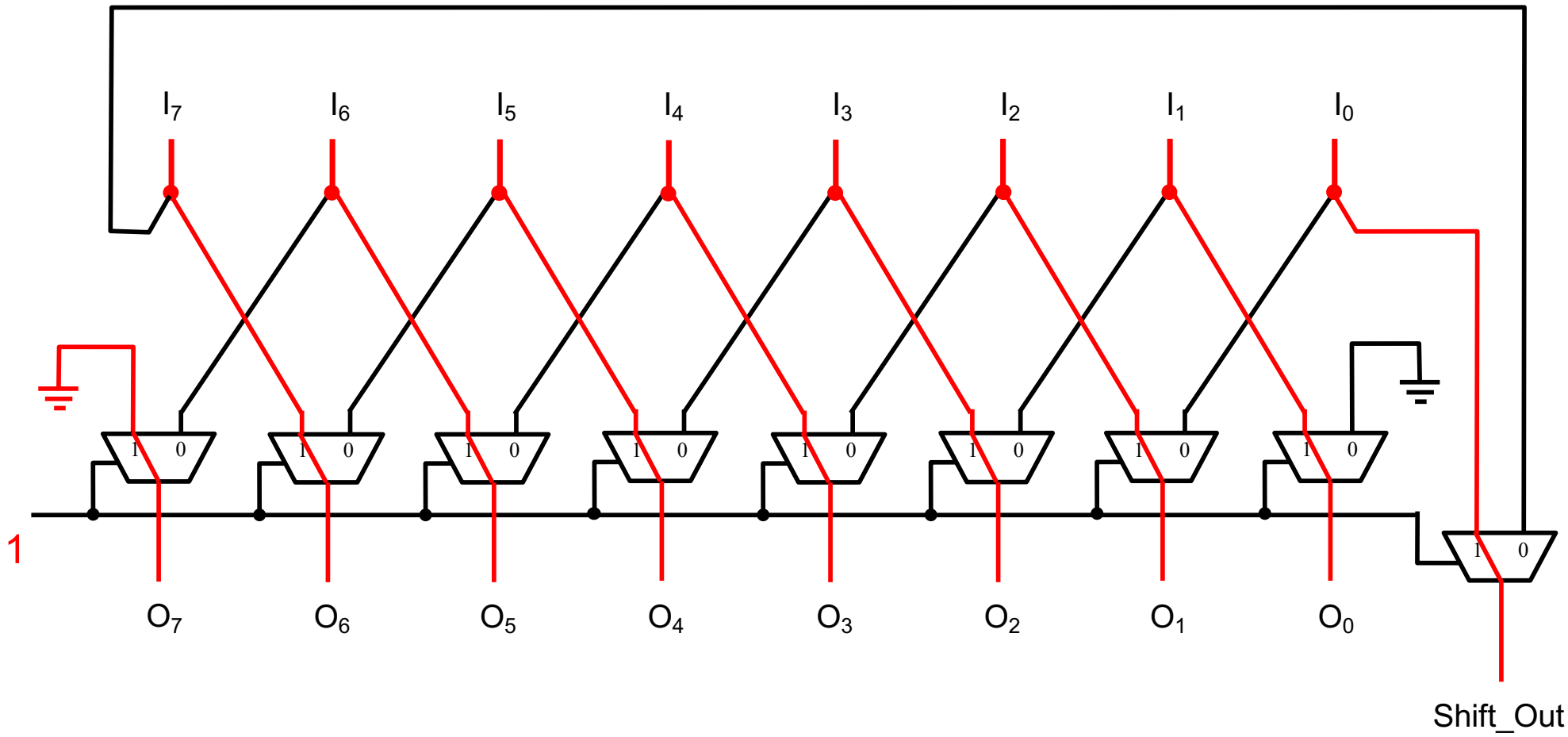
Shift Left



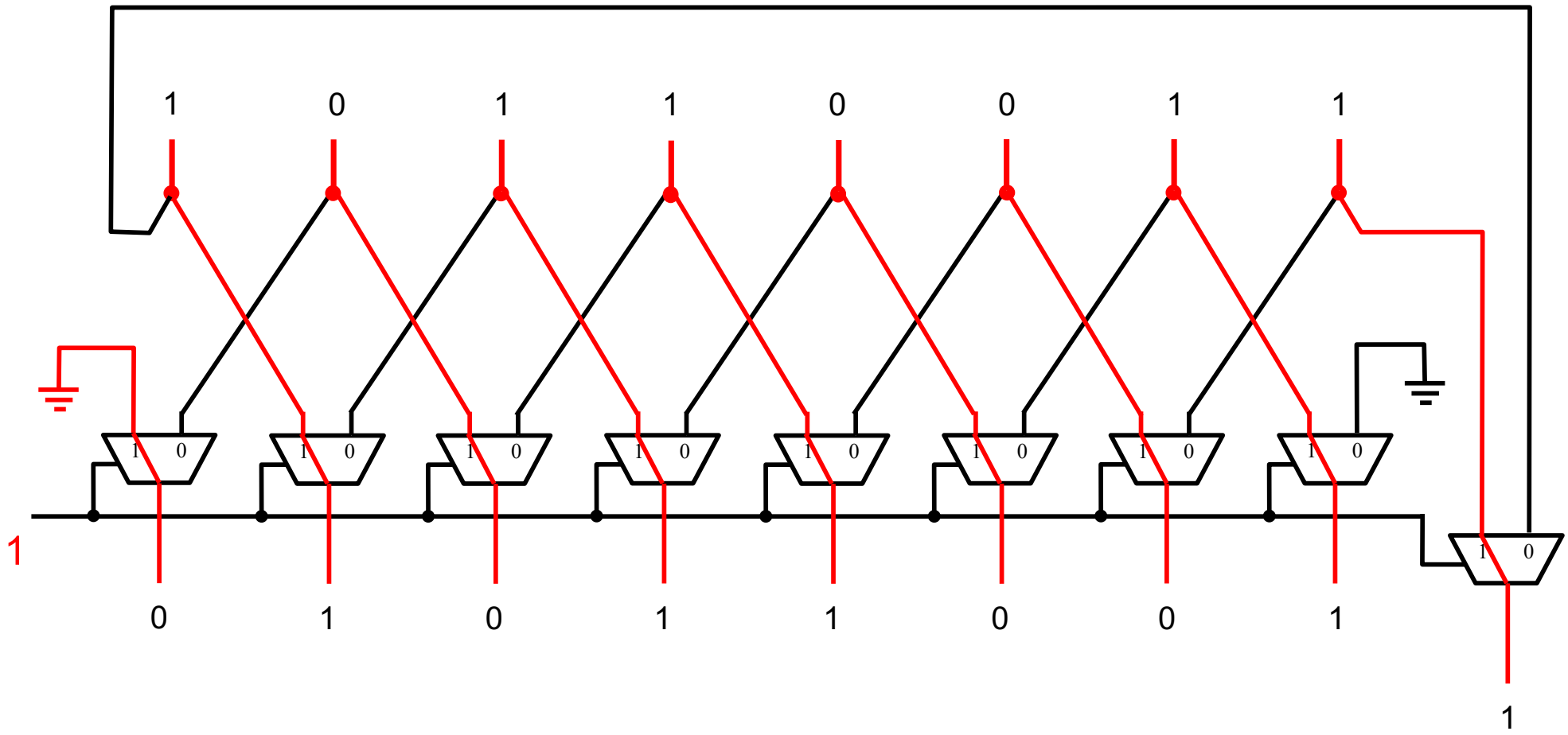
Shift Left



Shift Right

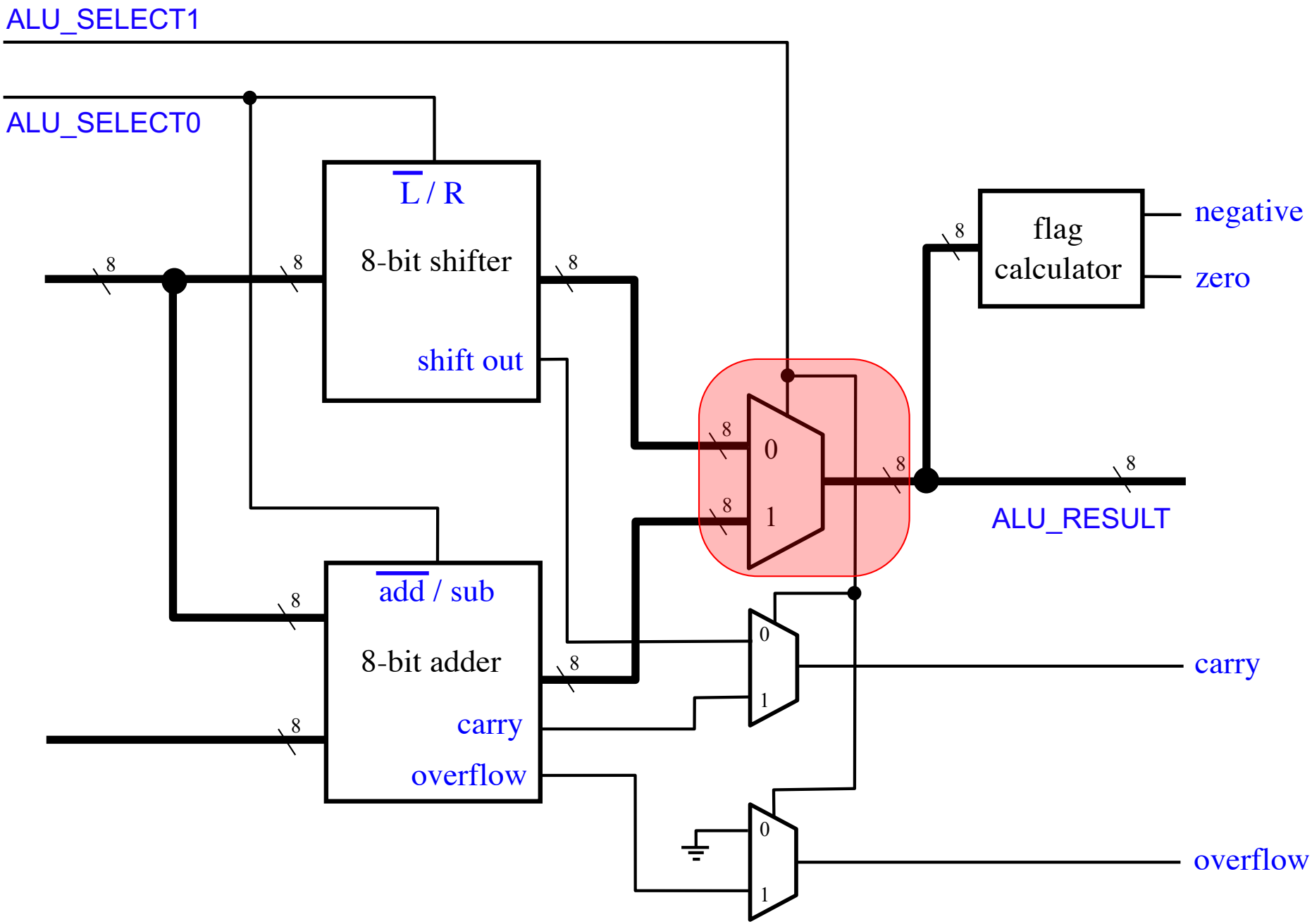


Shift Right

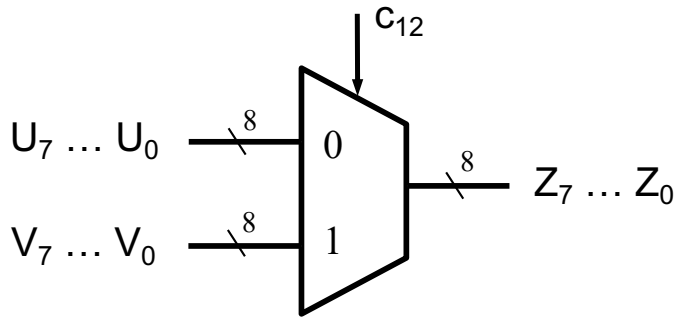


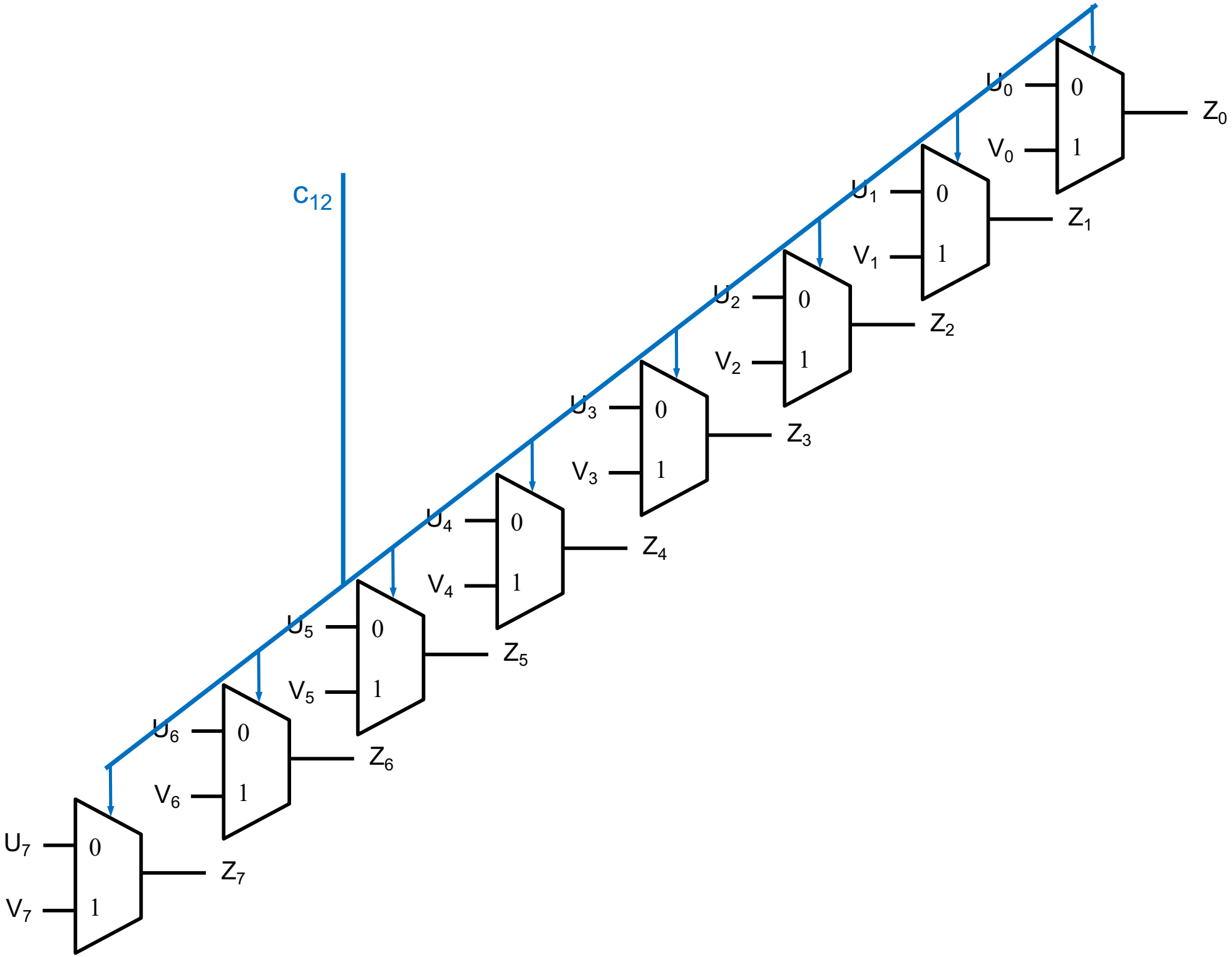
Bus Multiplexer

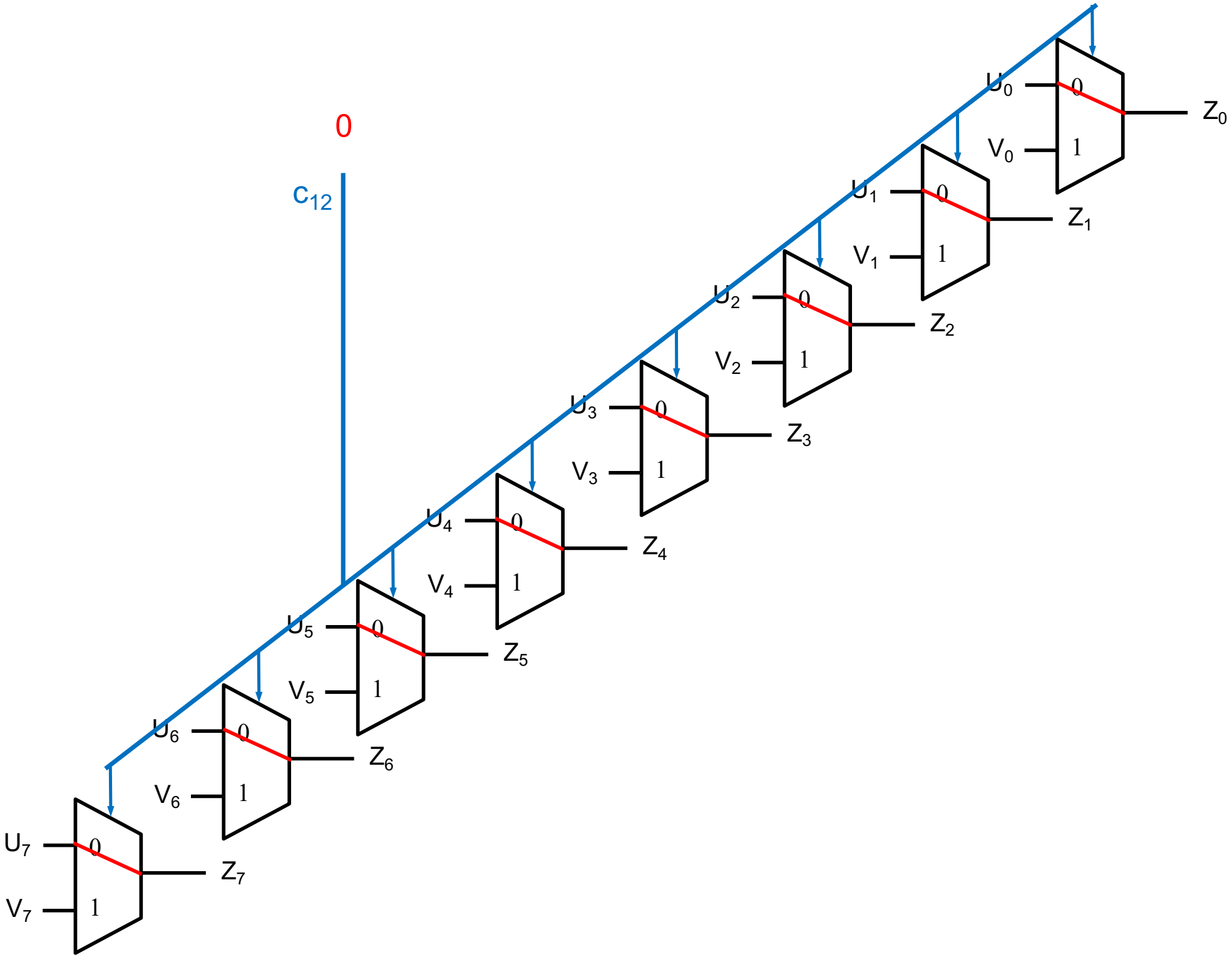
The Internal ALU Bus Multiplexer

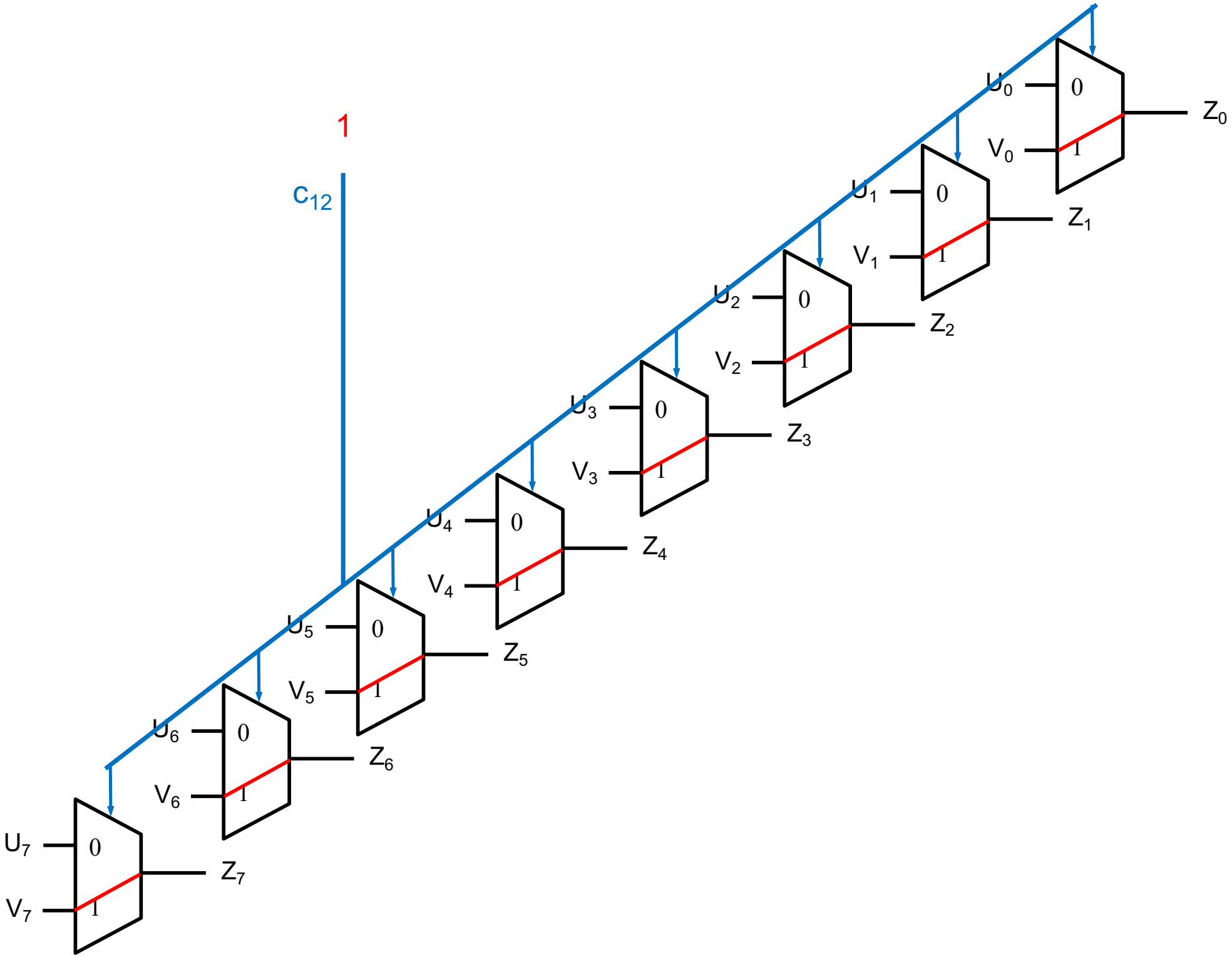


2-to-1 Bus Multiplexer (with 8-bit lines)

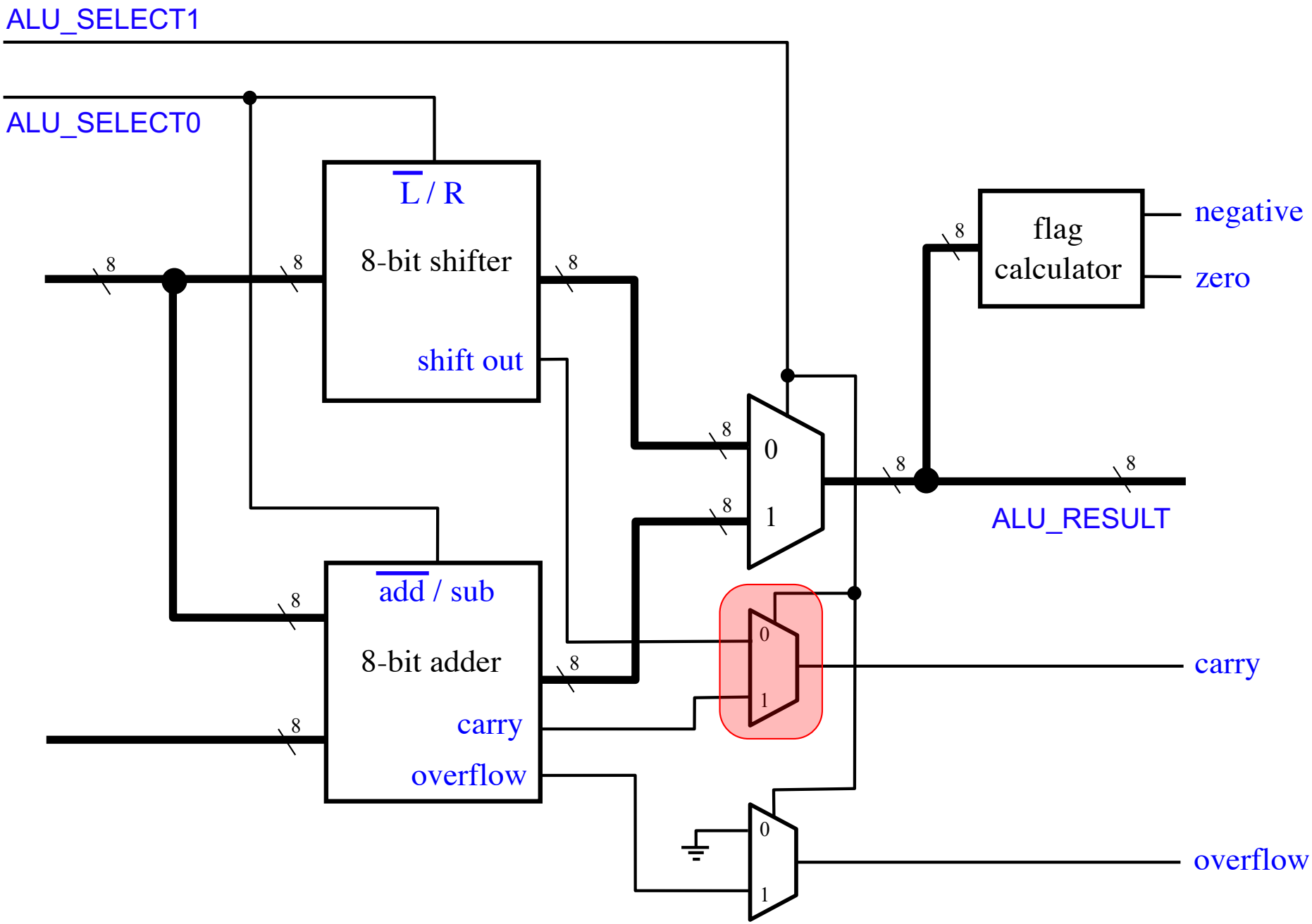



C12
U0
V0
0
1
Z0
U1
V1
Z1
U2
V2
Z2
U3
V3
Z3
U4
V4
Z4
U5
V5
Z5
U6
V6
Z6
U7
V7
Z7

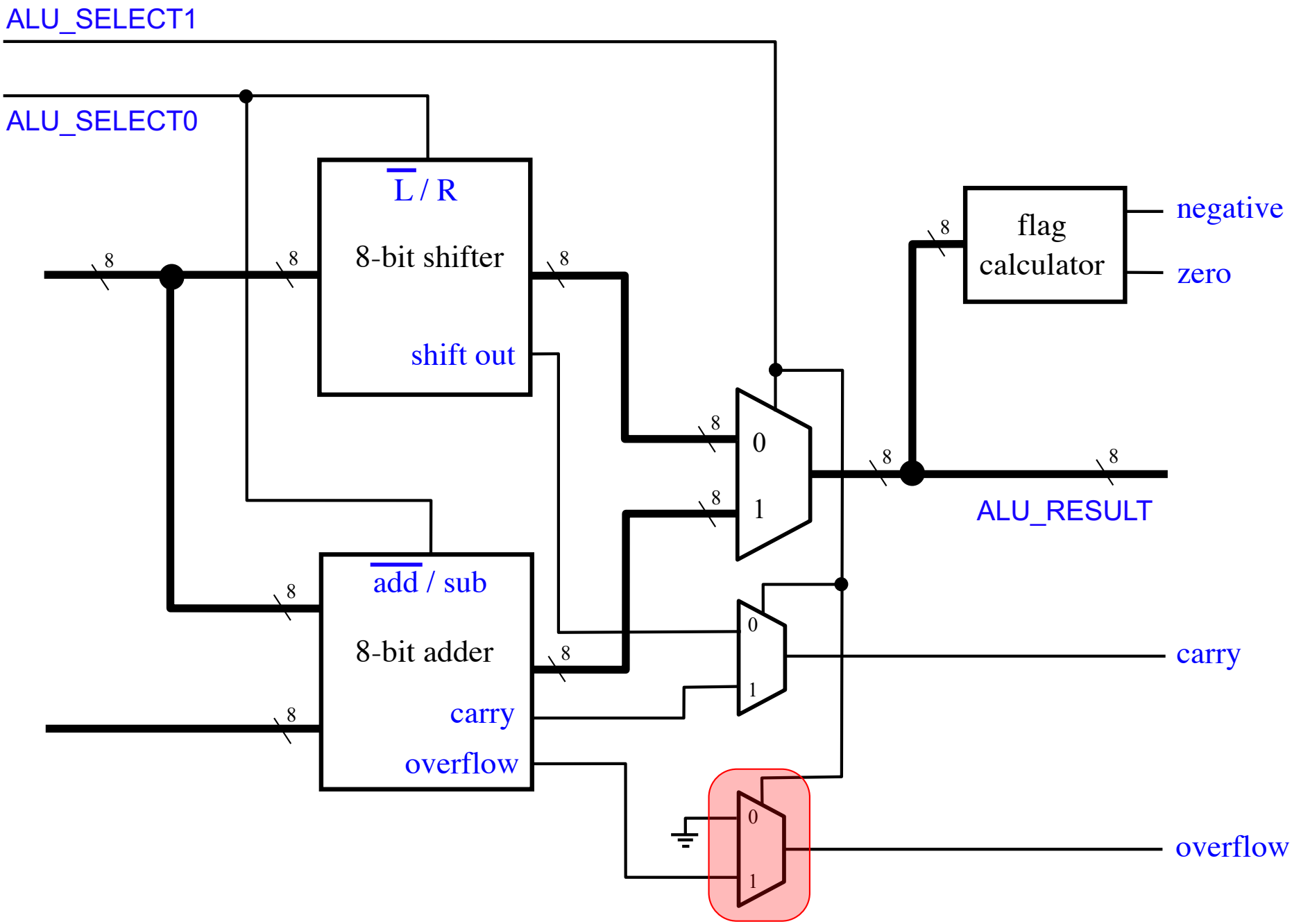

0
c_{12}
U_0
V_0
Z_0
U_1
V_1
Z_1
U_2
V_2
Z_2
U_3
V_3
Z_3
U_4
V_4
Z_4
U_5
V_5
Z_5
U_6
V_6
Z_6
U_7
V_7
Z_7
0
1


1
C_{12}
U_0
V_0
Z_0
U_1
V_1
Z_1
U_2
V_2
Z_2
U_3
V_3
Z_3
U_4
V_4
Z_4
U_5
V_5
Z_5
U_6
V_6
Z_6
U_7
V_7
Z_7
0
1

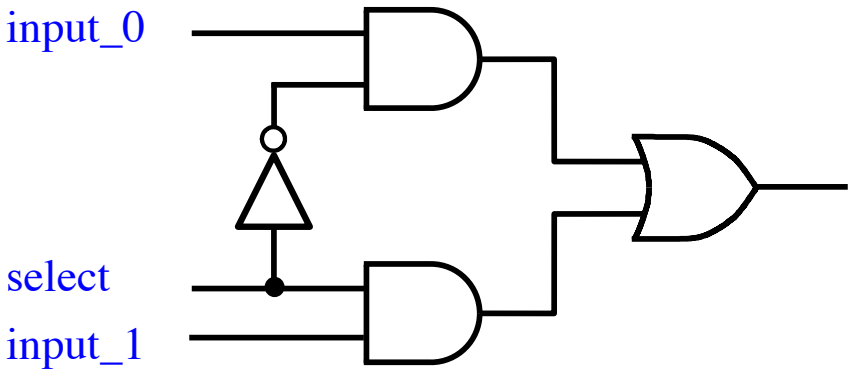
2-to-1 Multiplexer



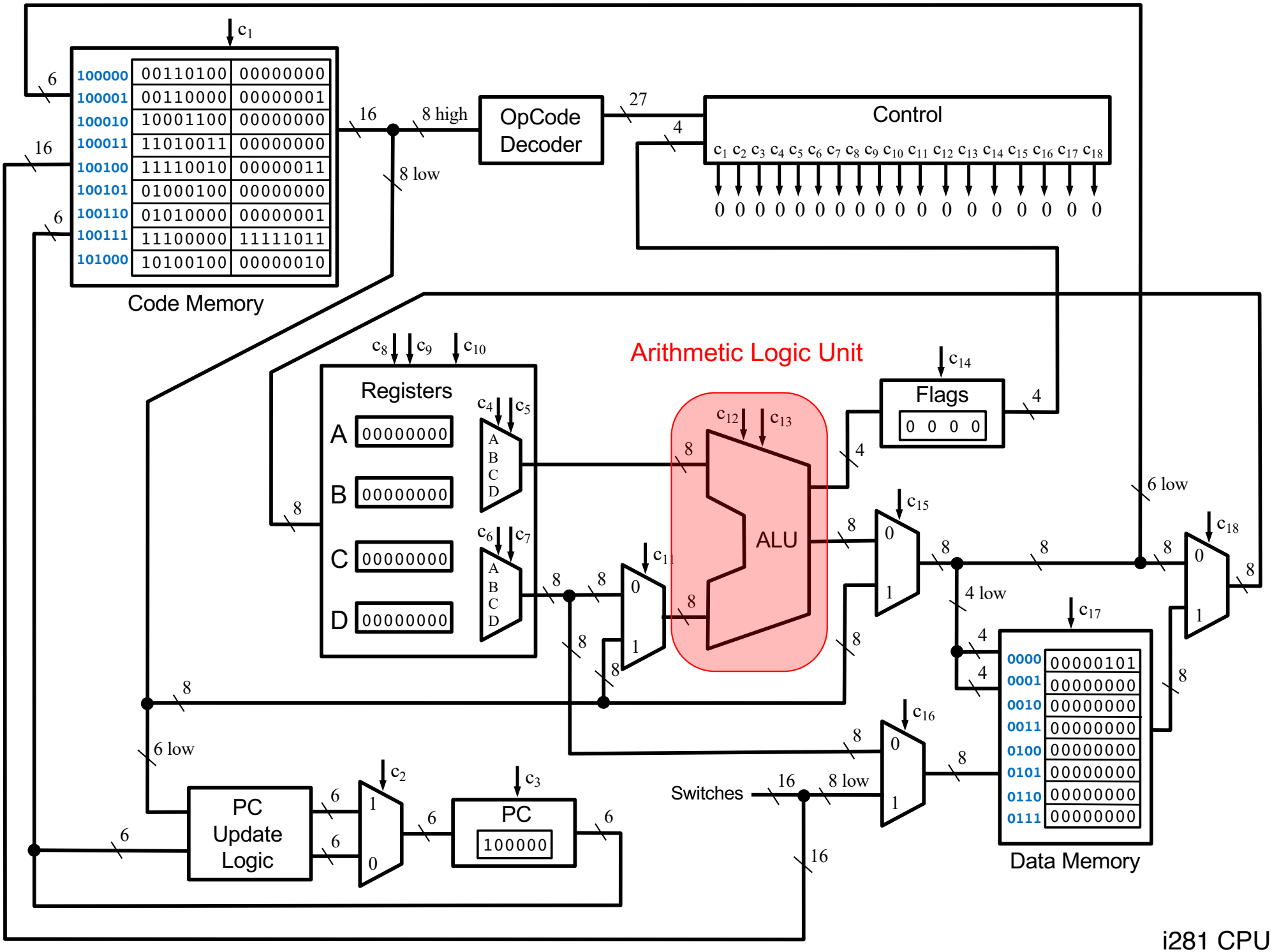
2-to-1 Multiplexer



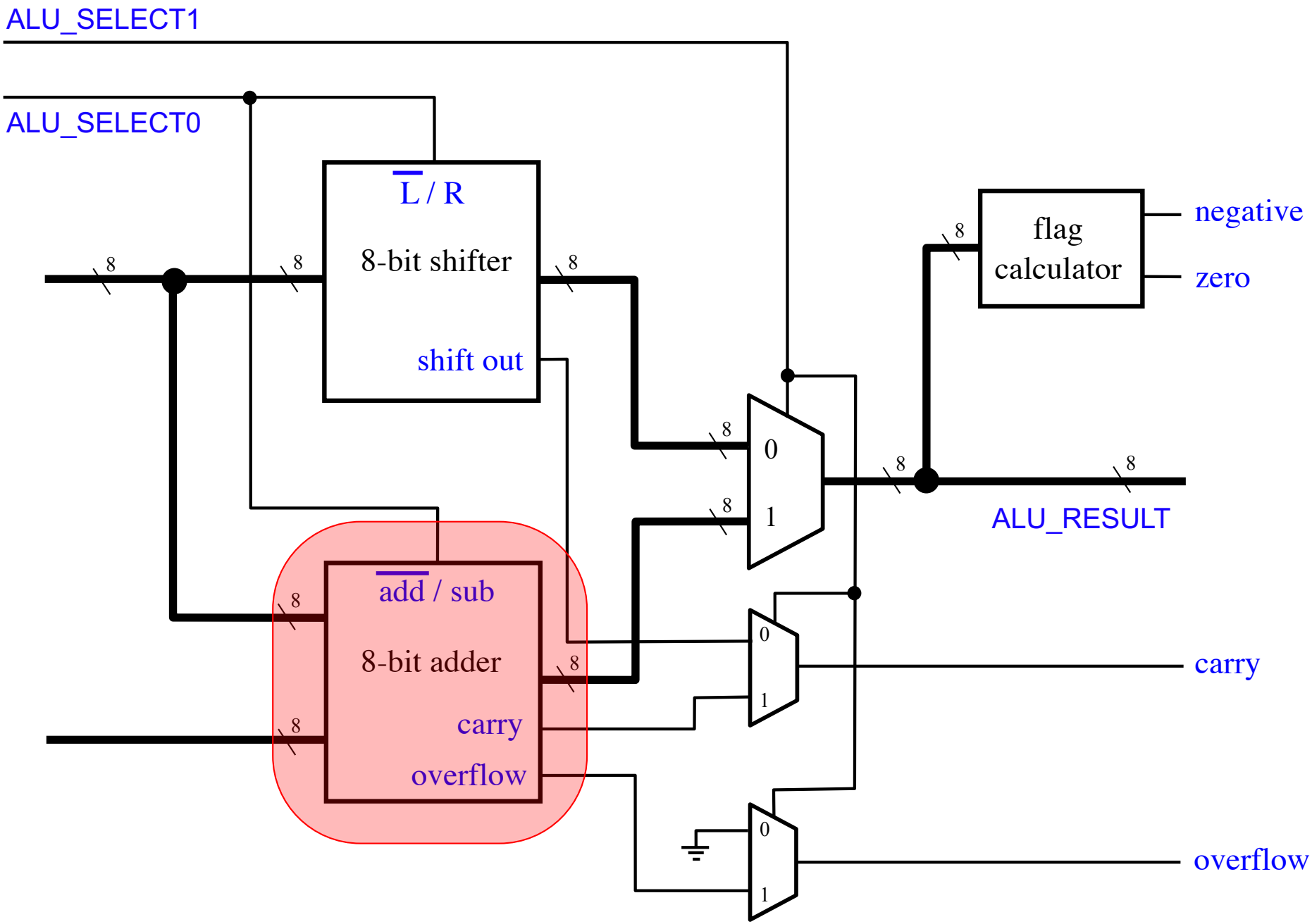
2-to-1 Multiplexer



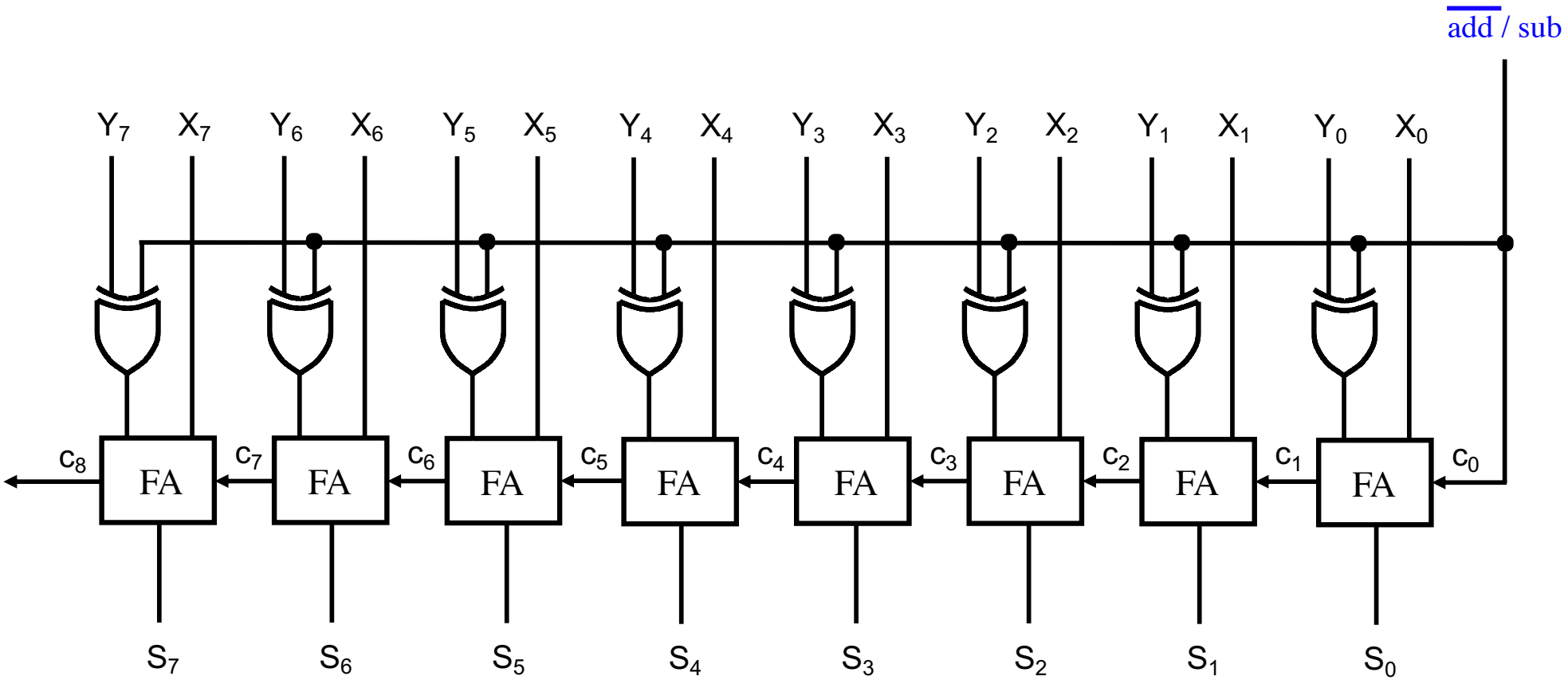
The adder/subtractor of the i281 CPU


c1
6
100000 00110100 00000000
100001 00110000 00000001
100010 10001100 00000000
100011 11010011 00000000
100100 11110010 00000011
100101 01000100 00000000
100110 01010000 00000001
100111 11100000 11111011
101000 10100100 00000010
Code Memory
16
8 high
8 low
OpCode Decoder
27
4
Control
c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17 c18
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Arithmetic Logic Unit
c8 c9 c10
Registers
A 00000000
B 00000000
C 00000000
D 00000000
c4 c5
c6 c7
A B C D
c12 c13
ALU
c14
Flags
0 0 0 0
4
c11
c15
c18
6 low
4 low
c17
0000 00000101
0001 00000000
0010 00000000
0011 00000000
0100 00000000
0101 00000000
0110 00000000
0111 00000000
Data Memory
c16
Switches
8 low
c2
c3
PC Update Logic
PC
100000
i281 CPU

The Adder / Subtractor

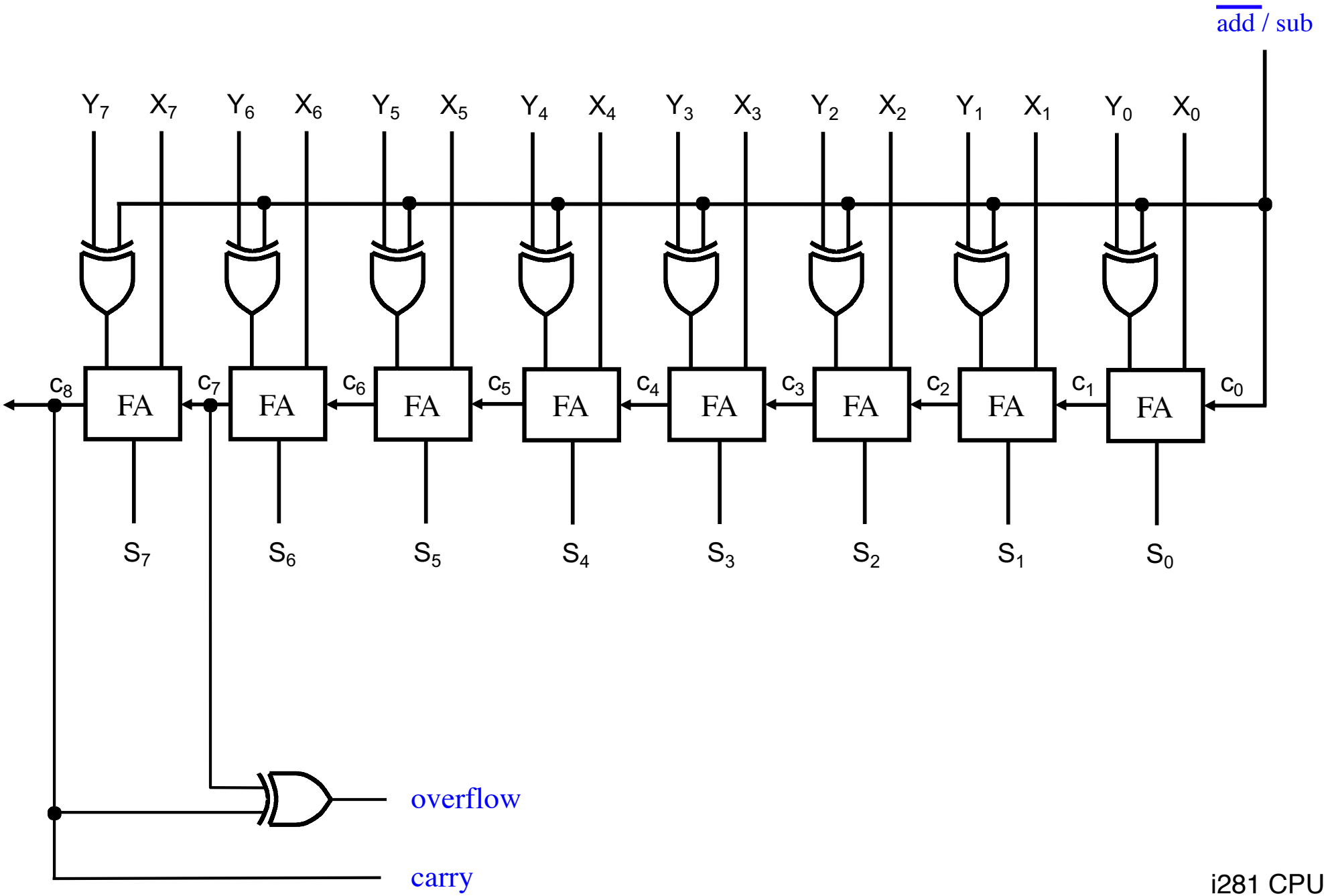


The Adder / Subtractor

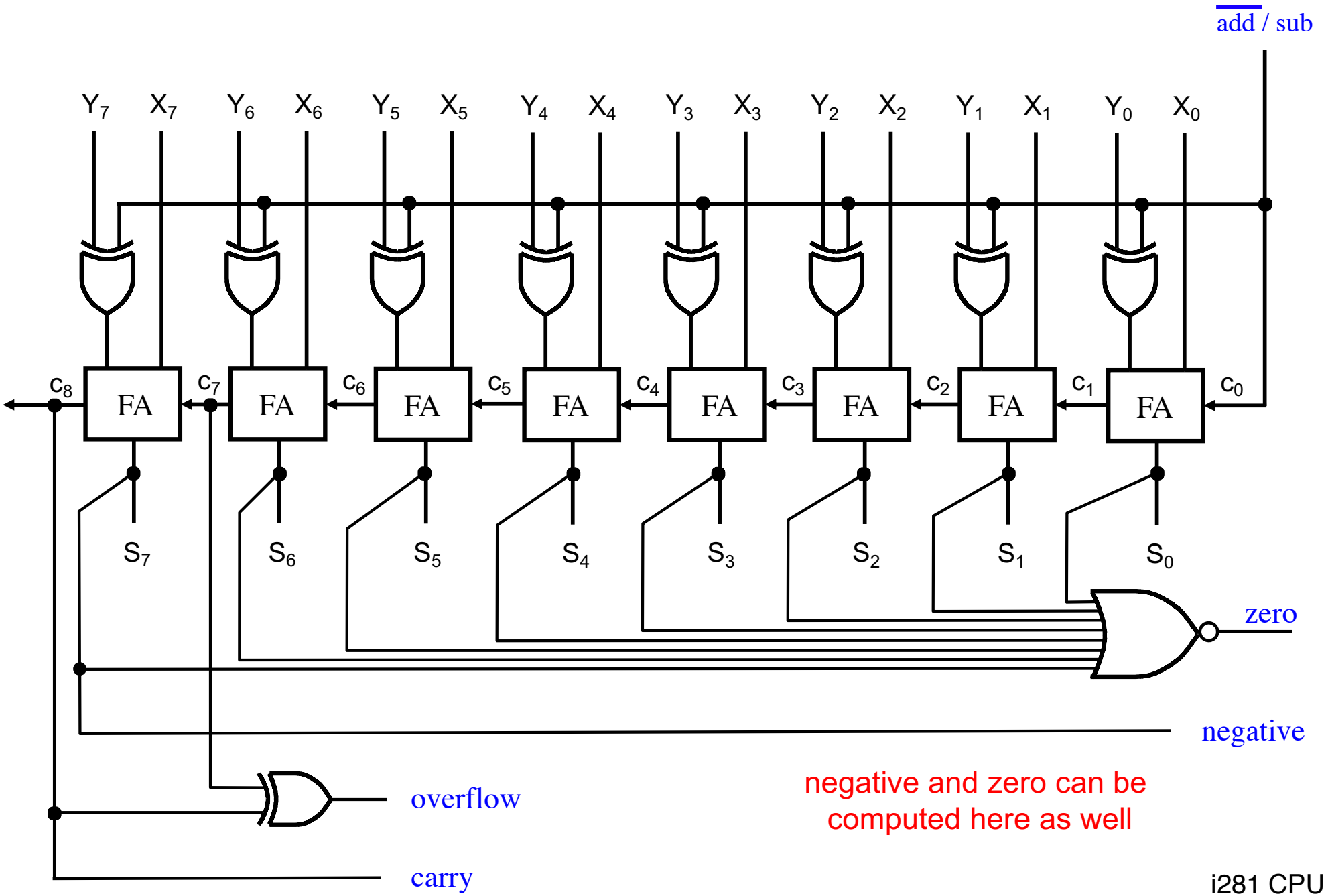


This is an 8-bit ripple-carry adder. Note that the X and Y lines are swapped.

The Adder / Subtractor



The Adder / Subtractor



Abbreviations for the Flags

- **Carry Flag (CF)**
- **Overflow Flag (OF)**
- **Negative Flag (NF)**
- **Zero Flag (ZF)**

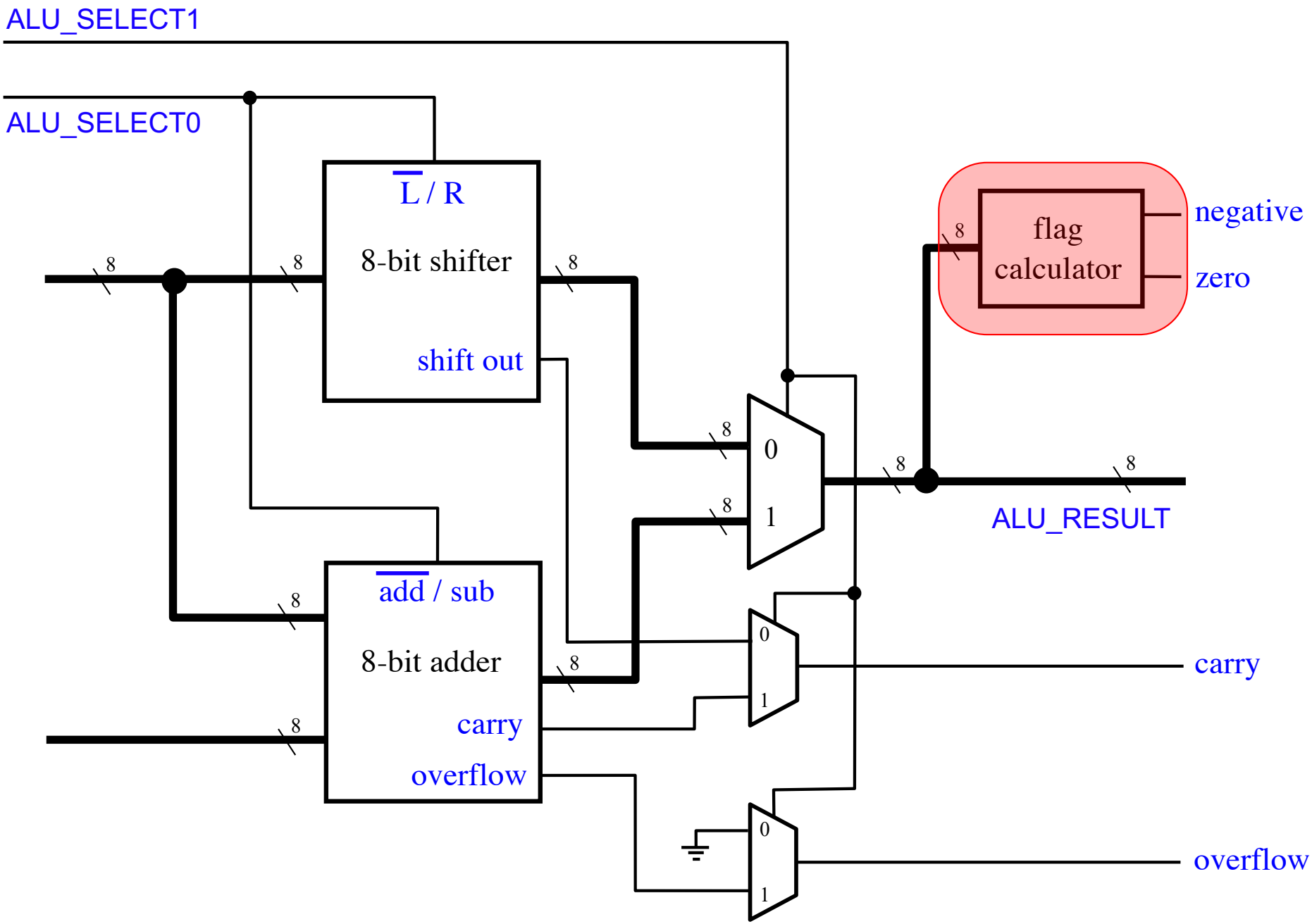
Abbreviations for the Flags

- **Carry Flag (CF)**
- **Overflow Flag (OF)**
- **Negative Flag (NF)**
- **Zero Flag (ZF)**

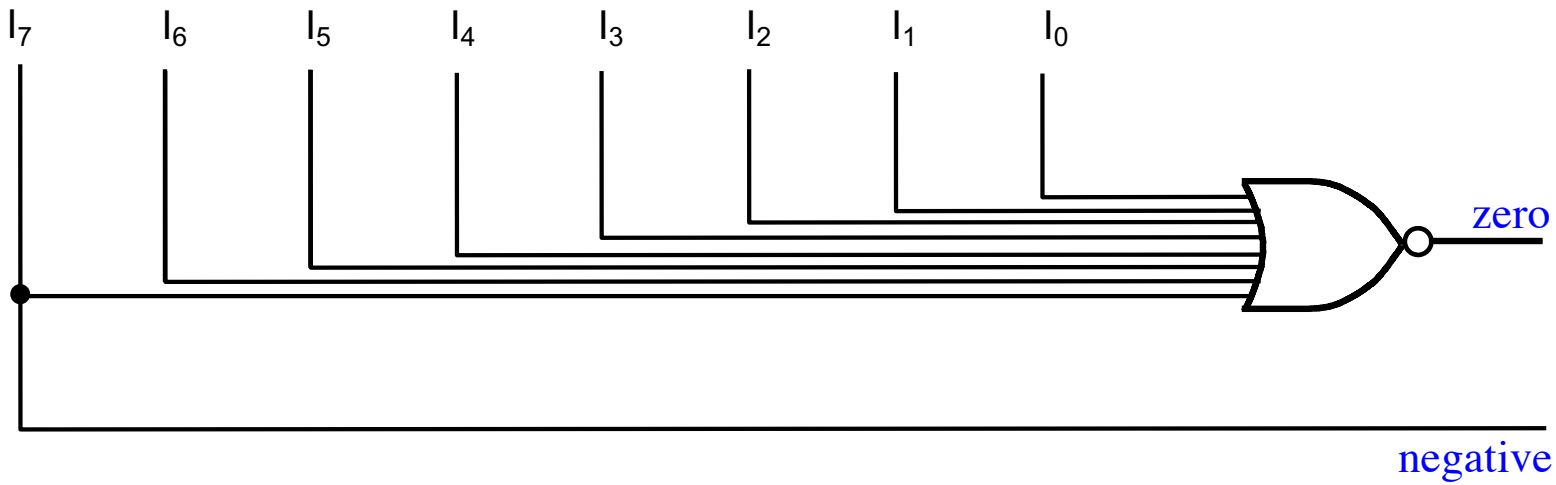
In some CPU architectures the carry flag means borrow. And it could be inverted relative to the previous diagram.

The flag calculator of the i281 CPU

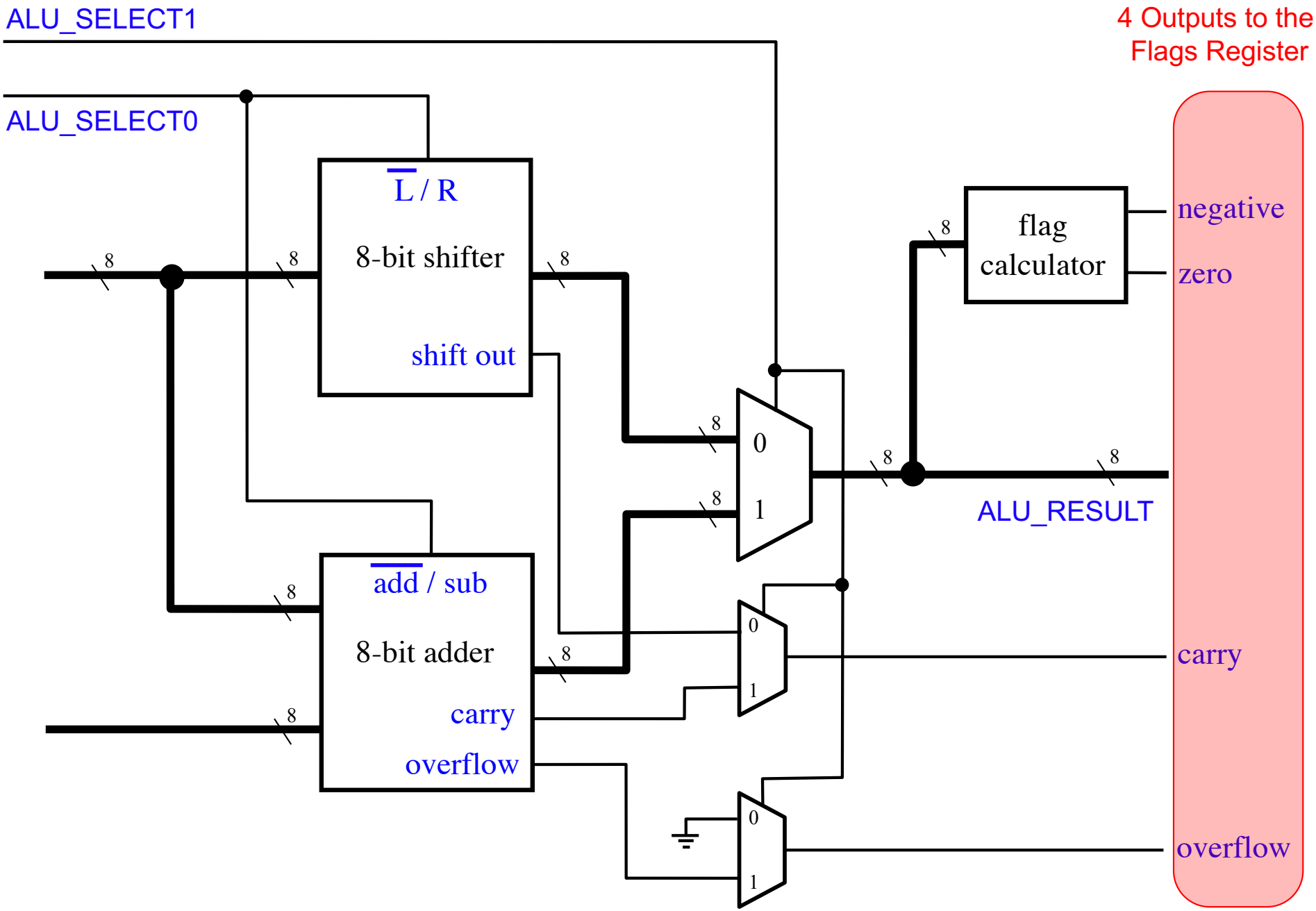
The ALU Flag Calculator



The ALU Flag Calculator



ALU Outputs to the Flags Register



Comparison of Signed Numbers

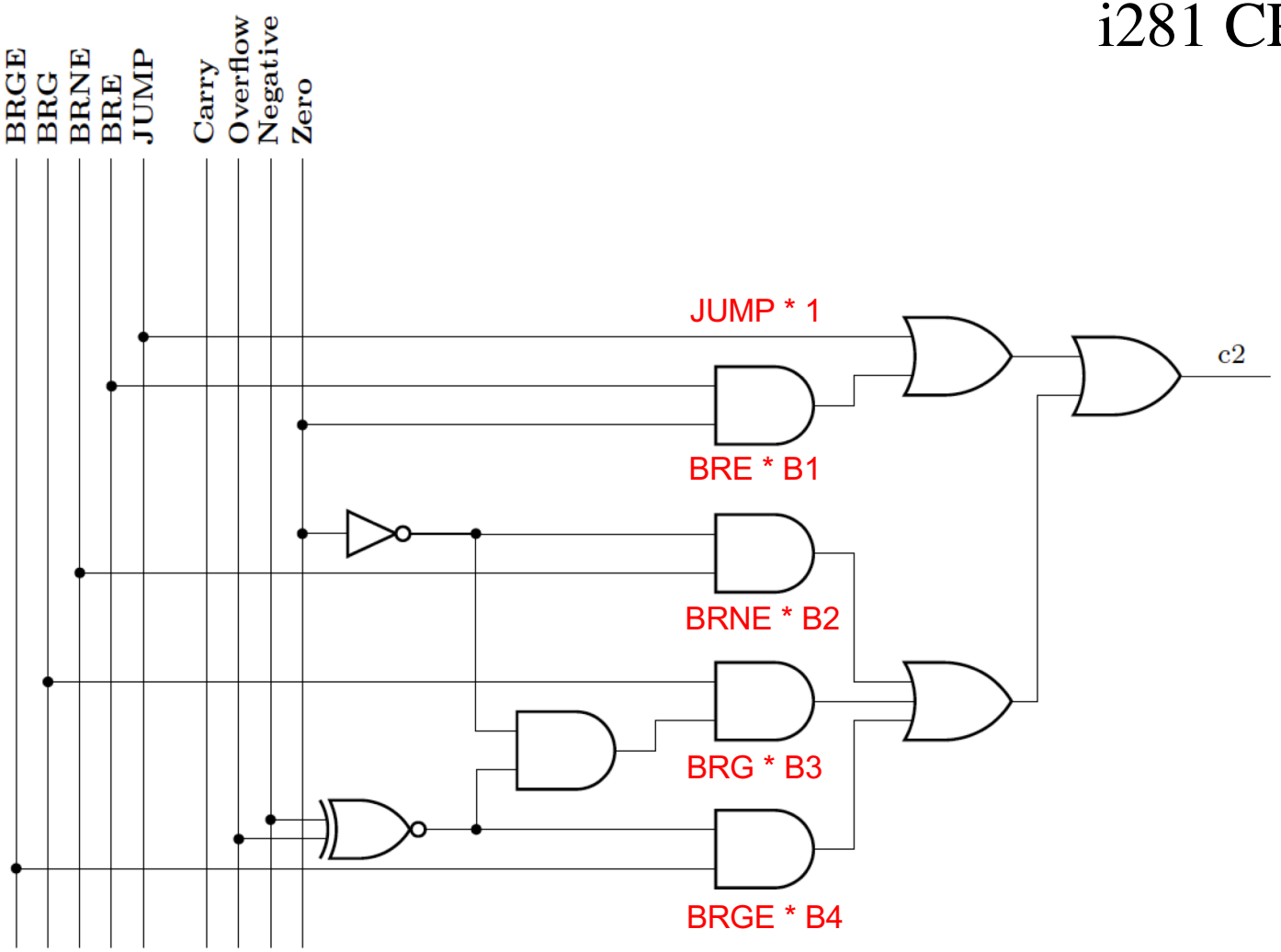
Comparison of Signed Numbers

- **Equal** — **ZF = 1**
- **Not equal** — **ZF = 0**
- **Greater** — **ZF = 0 and NF = OF**
- **Greater or Equal** — **NF = OF**
- **Less** — **NF != OF**
- **Less or Equal** — **ZF = 1 or NF != OF**

Comparison of Signed Numbers

- **Equal** — **ZF**
- **Not equal** — $\overline{ZF}$
- **Greater** — $\overline{ZF} \bullet XNOR(NF, OF)$
- **Greater or Equal** — **XNOR(NF, OF)**
- **Less** — **XOR(NF, OF)**
- **Less or Equal** — **ZF + XOR(NF, OF)**

i281 CPU



JUMP		1	1															
BRE/BRZ		B1	1															
BRNE/BRNZ		B2	1															
BRG		B3	1															
BRGE		B4	1															

C_2 is the OR of these five times the OPCODE

```
B1=  ZF
B2= ~ZF
B3= AND(~ZF, XNOR(NF, OF))
B4= XNOR(NF, OF)
```

Zero Flag (ZF)
Negative Flag (NF)
Overflow Flag (OF)

Some Interesting Dualities

- $\overline{\text{Equal}} = \text{Not Equal}$
- $\overline{\text{Greater}} = \text{Less or Equal}$
- $\overline{\text{Less}} = \text{Greater or Equal}$

Comparison of Unsigned Numbers (not supported by this CPU)

Comparison of Unsigned Numbers

- Equal
- Not equal
- Greater
- Greater or equal
- Less
- Less or Equal

Comparison of Unsigned Numbers

- Equal
- Not equal
- Greater / Above
- Greater or Equal / Above or Equal
- Less / Below
- Less or Equal / Below or Equal

Comparison of Unsigned Numbers

- Equal — ZF = 1
- Not equal — ZF = 0
- Greater — ZF = 0 and CF = 1
- Greater or Equal — CF = 1
- Less — CF = 0
- Less or Equal — ZF = 1 or CF = 0

Comparison of Unsigned Numbers

- Equal — ZF
- Not equal — $\overline{ZF}$
- Greater — $\overline{ZF} \cdot CF$
- Greater or Equal — CF
- Less — $\overline{CF}$
- Less or Equal — $ZF + \overline{CF}$

Comparison of Unsigned Numbers

- Equal — ZF
- Not equal — $\overline{ZF}$
- Above — $\overline{ZF} \cdot CF$
- Above or Equal — CF
- Below — $\overline{CF}$
- Below or Equal — $ZF + \overline{CF}$

Questions?

THE END