

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
Copyright © Alexander Stoytchev

Administrative Stuff

- **HW 7 is out**
- **It is due on Monday (Oct 17) @ 10pm**
- **We will start with Chapter 5 on Friday.**

Quick Review

Decoders

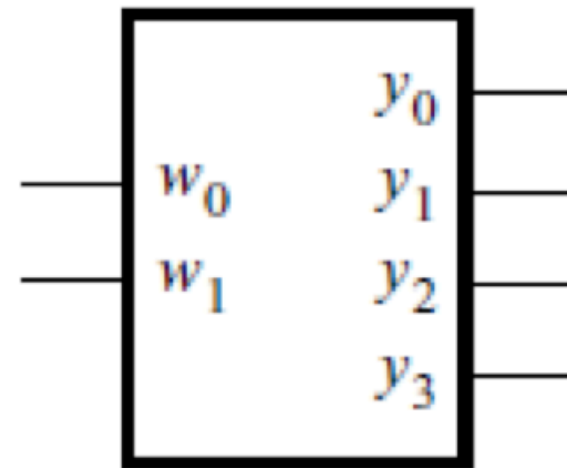
2-to-4 Decoder (Definition)

- 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



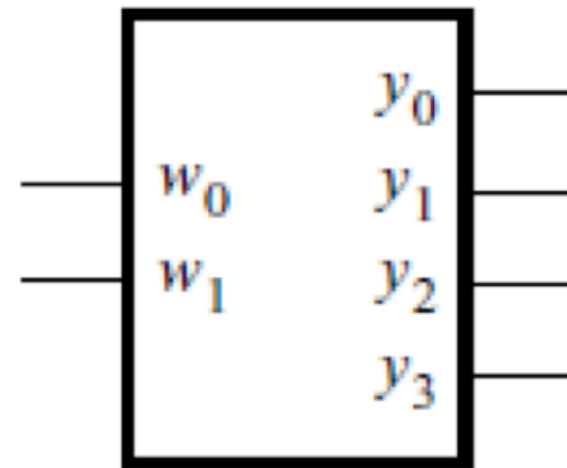
(b) Graphical symbol

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

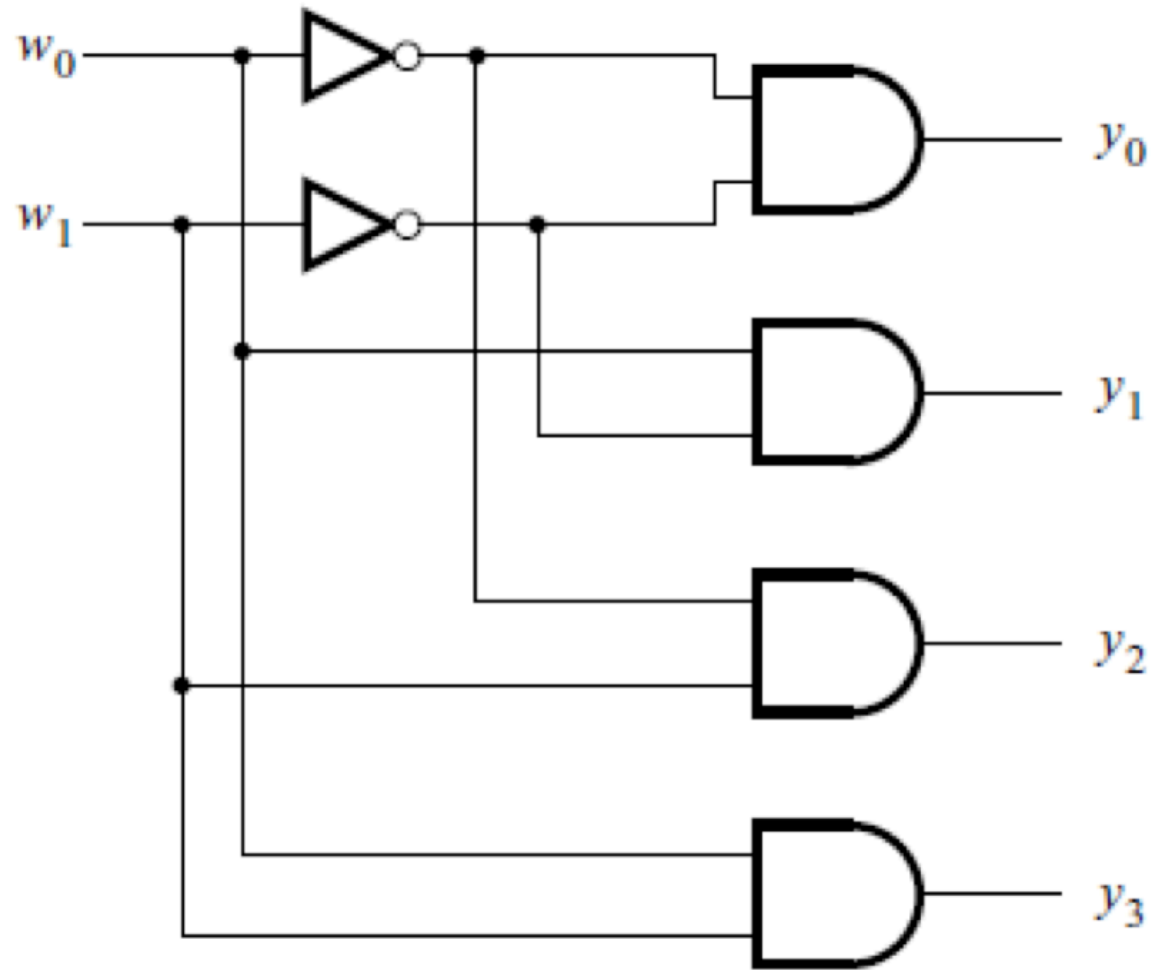
The outputs are “one-hot” encoded

(a) Truth table



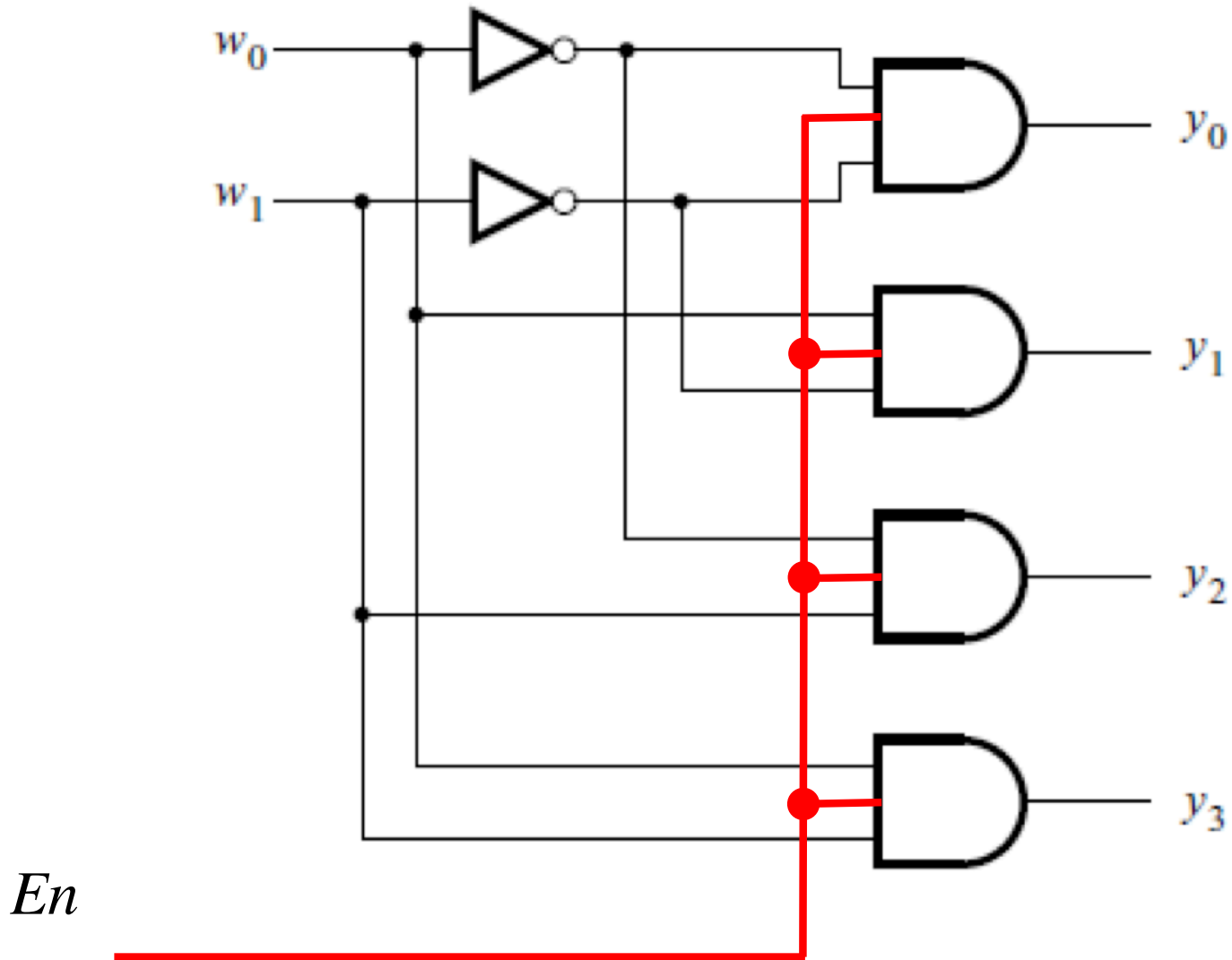
(b) Graphical symbol

Truth Logic Circuit for a 2-to-4 Decoder



[Figure 4.13c from the textbook]

Adding an Enable Input

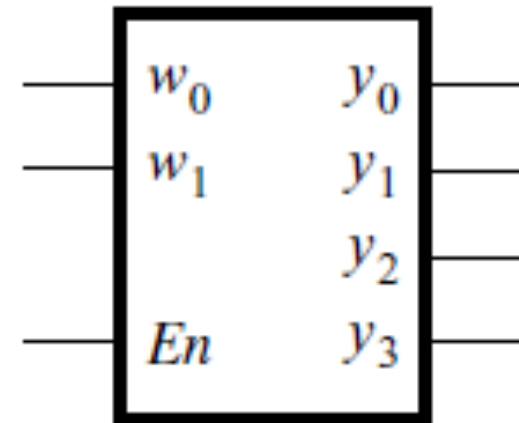


[Figure 4.13c from the textbook]

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

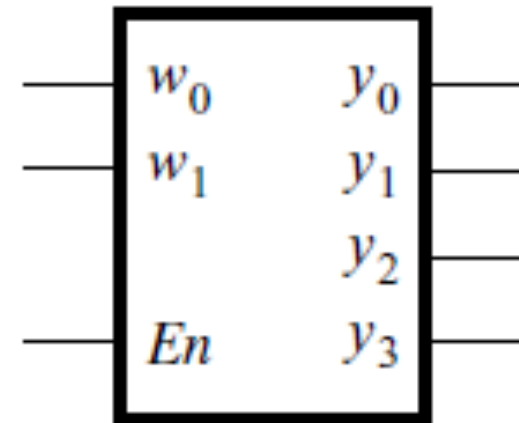


(b) Graphical symbol

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

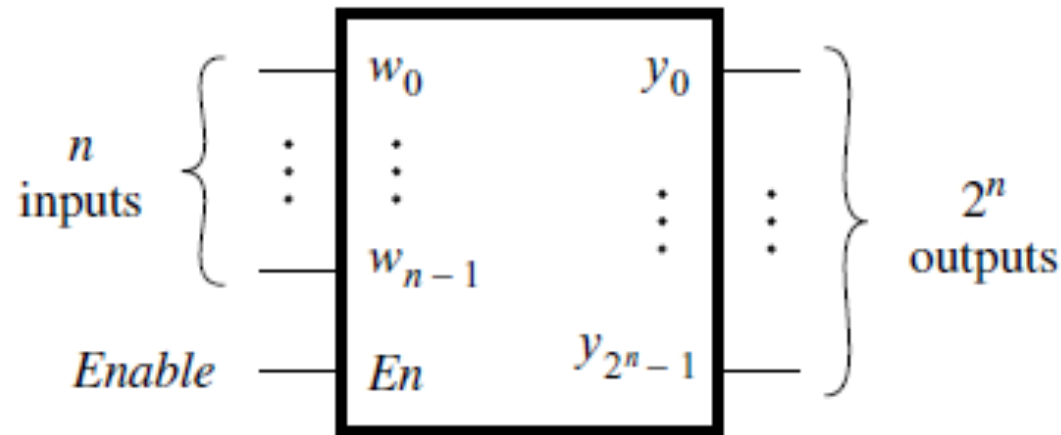
(a) Truth table



(b) Graphical symbol

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

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

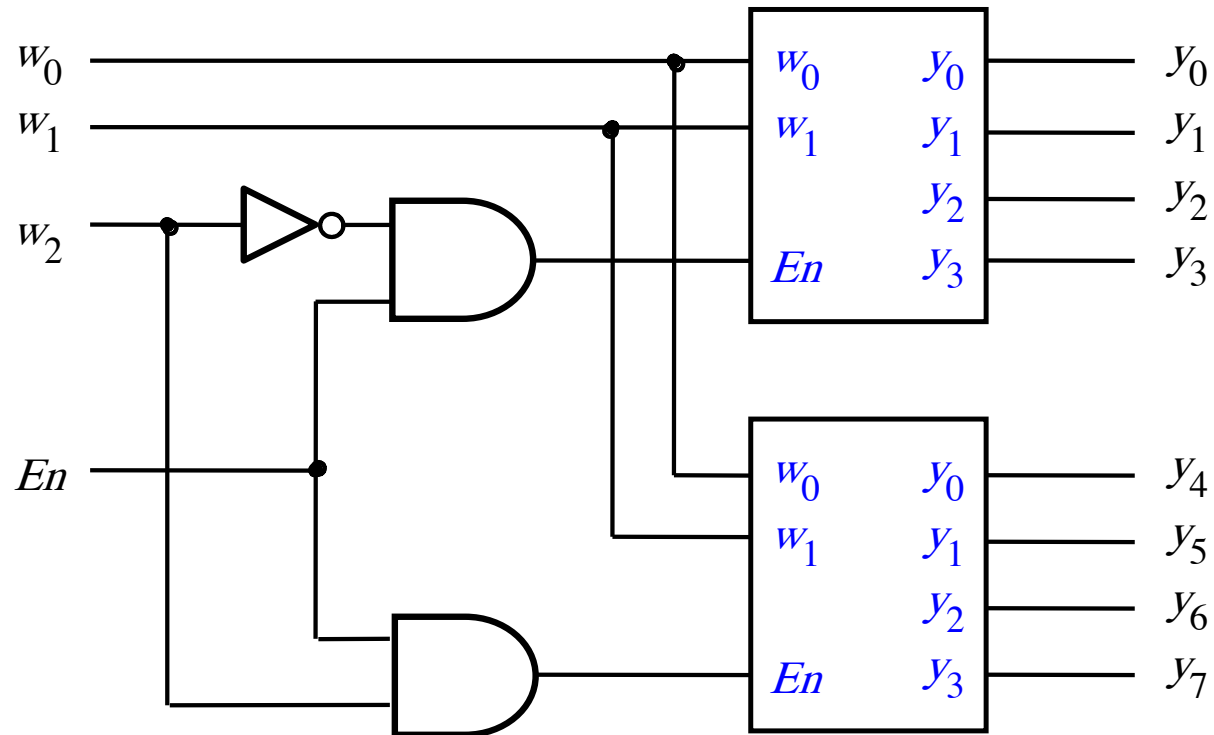


(d) An n -to- 2^n decoder

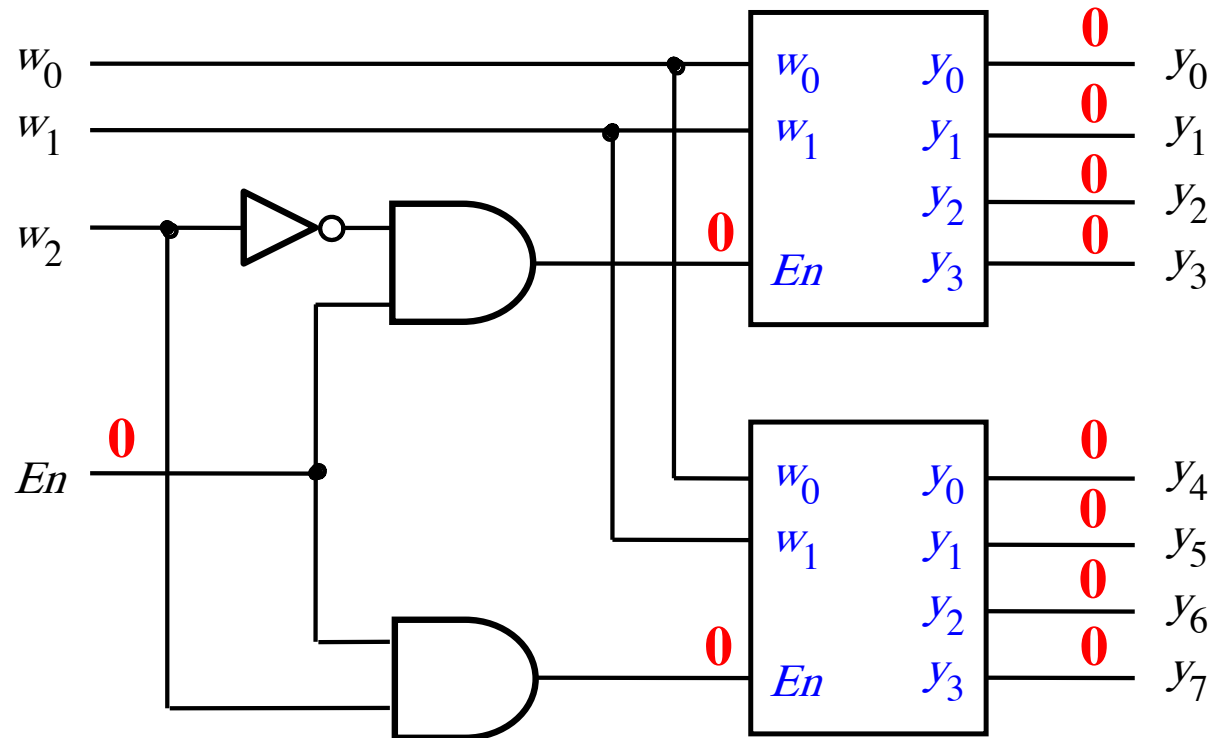
A binary decoder with n inputs has 2^n outputs

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*.

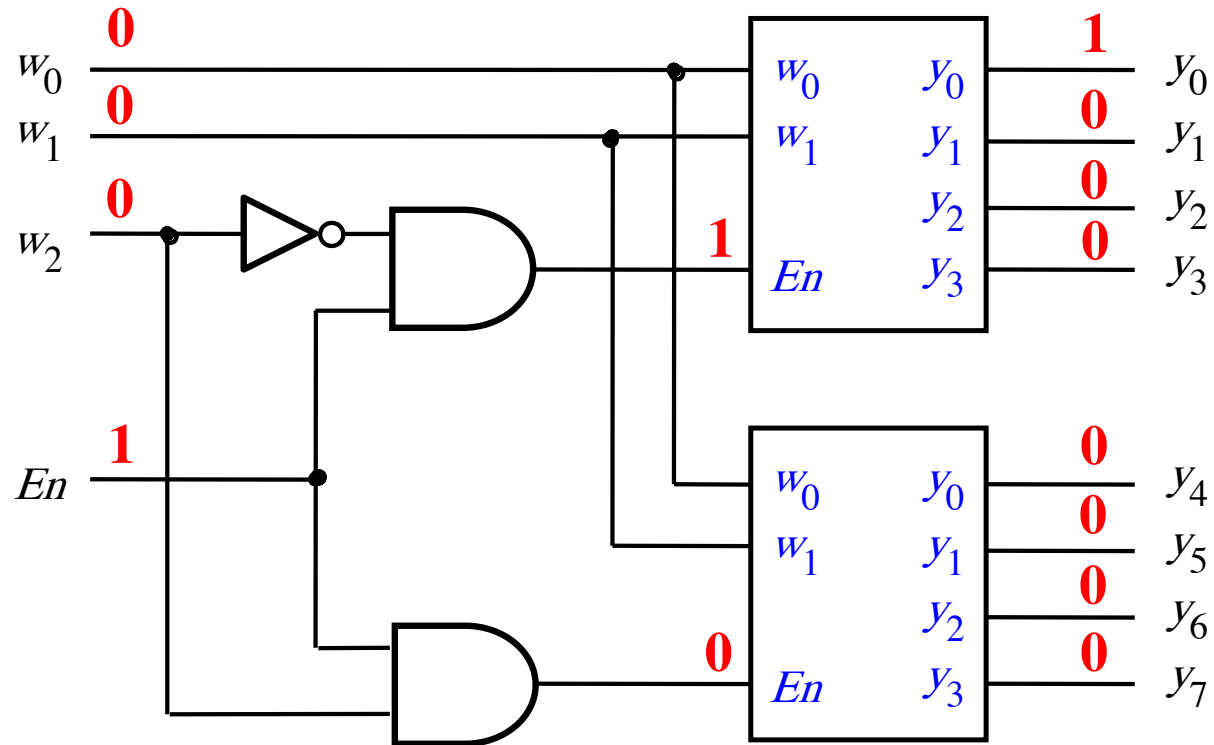
A 3-to-8 decoder using two 2-to-4 decoders



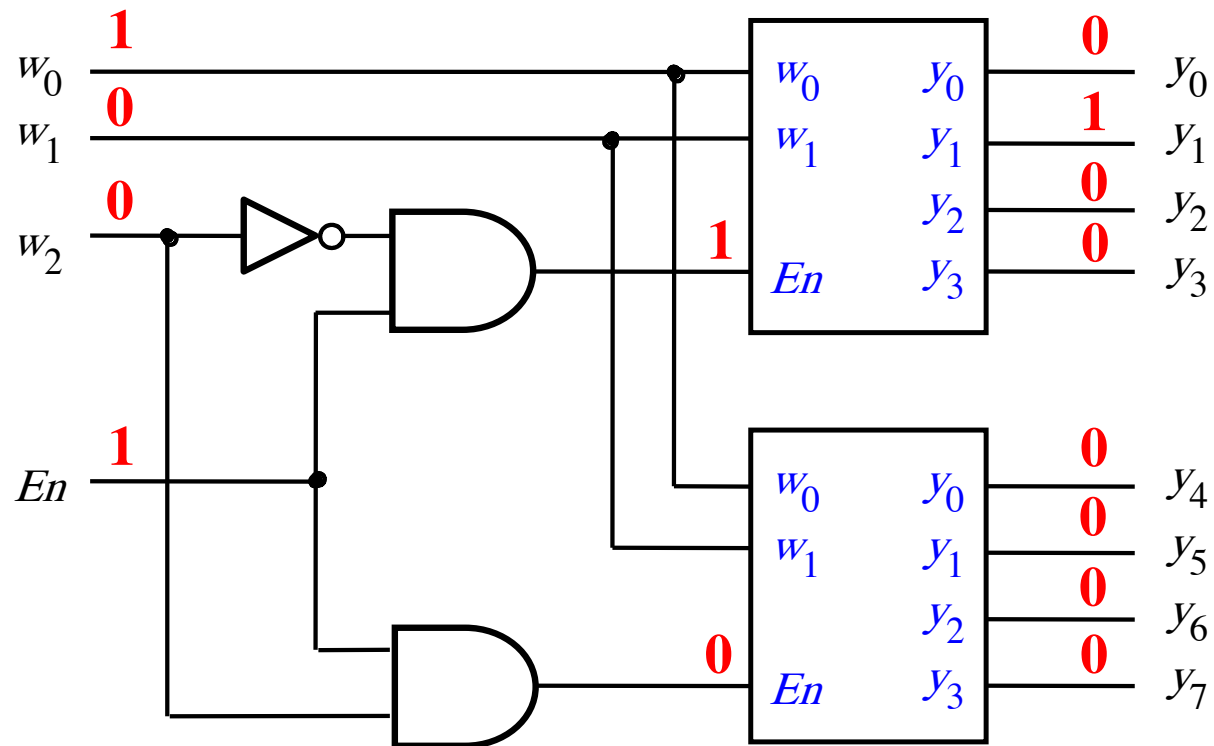
A 3-to-8 decoder using two 2-to-4 decoders



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

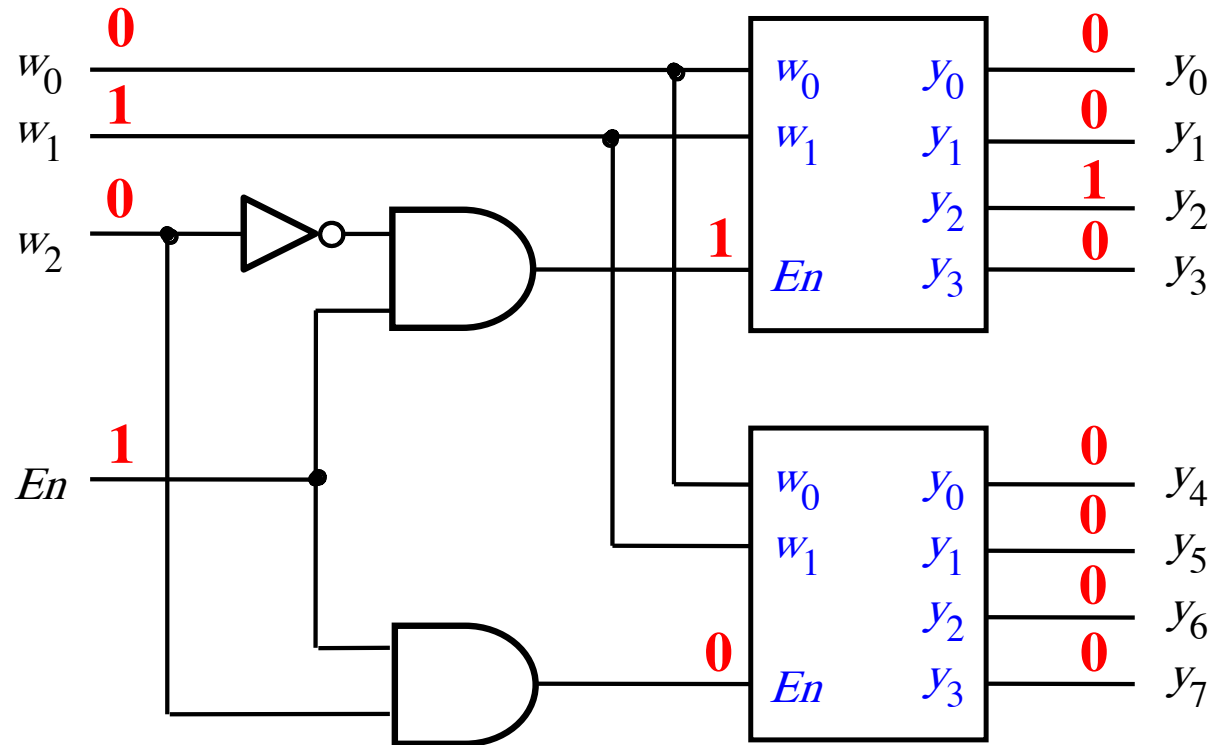


A 3-to-8 decoder using two 2-to-4 decoders



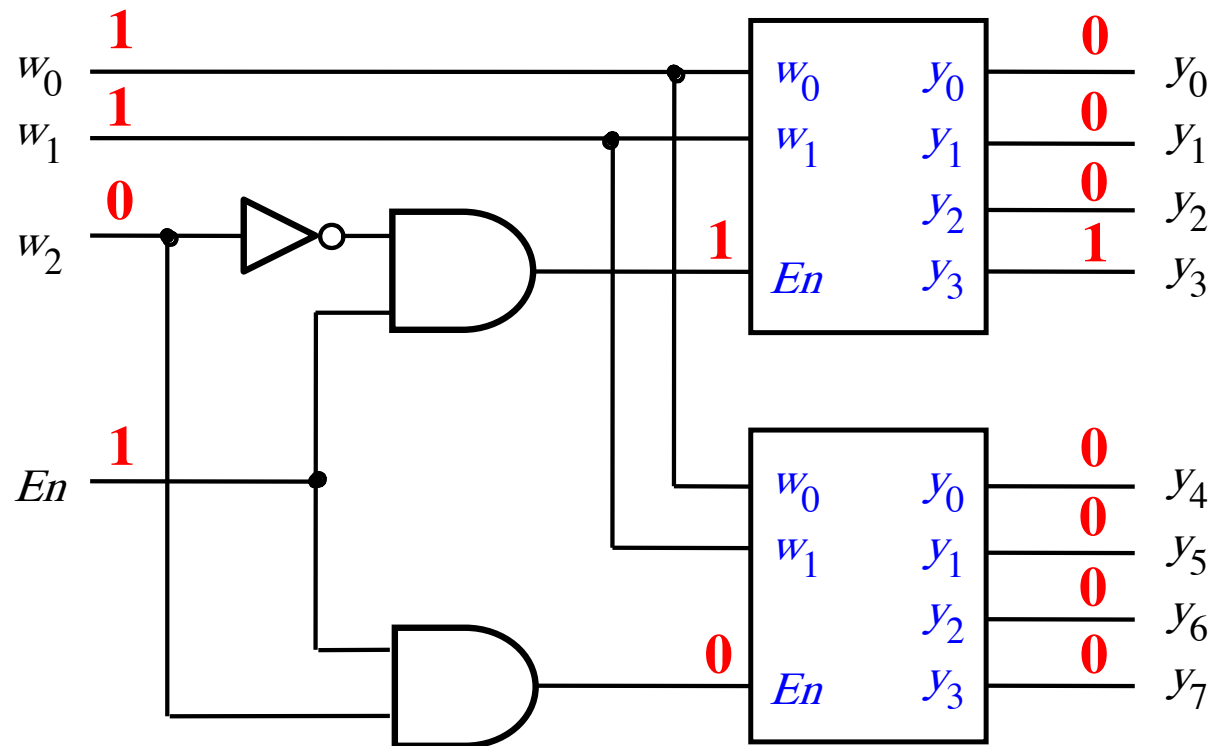
[Figure 4.15 from the textbook]

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

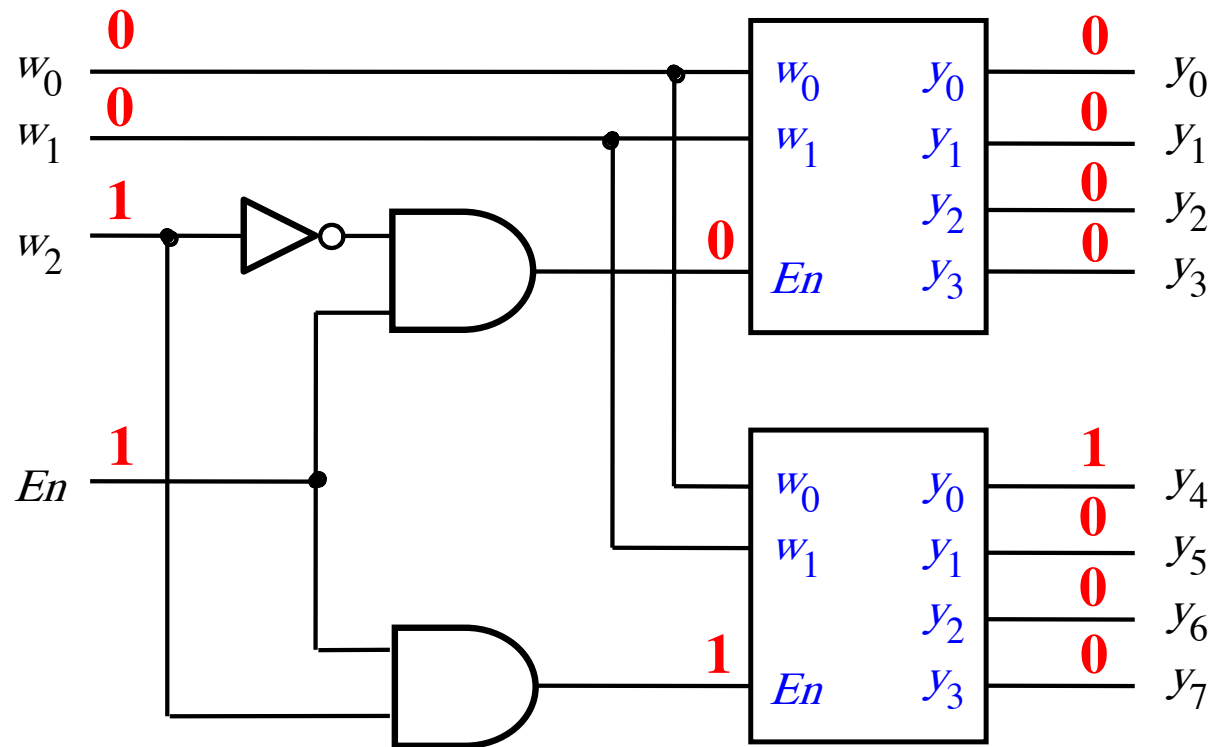


[Figure 4.15 from the textbook]

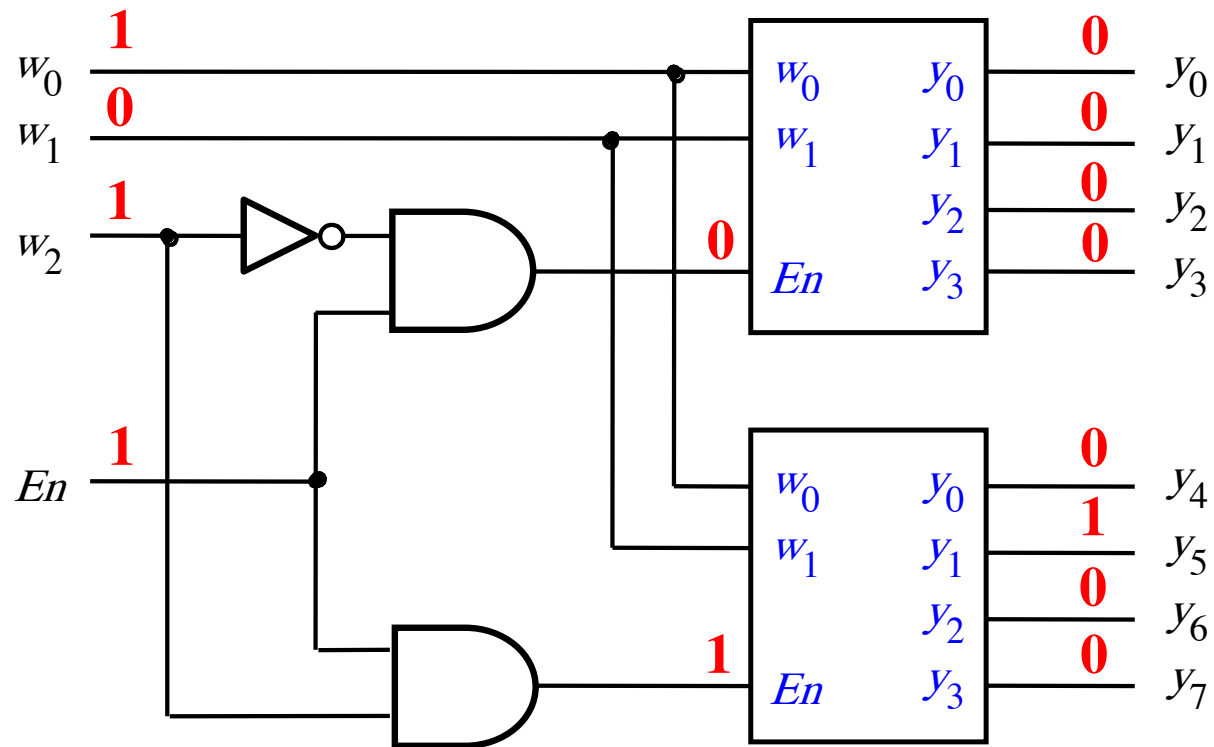
A 3-to-8 decoder using two 2-to-4 decoders



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

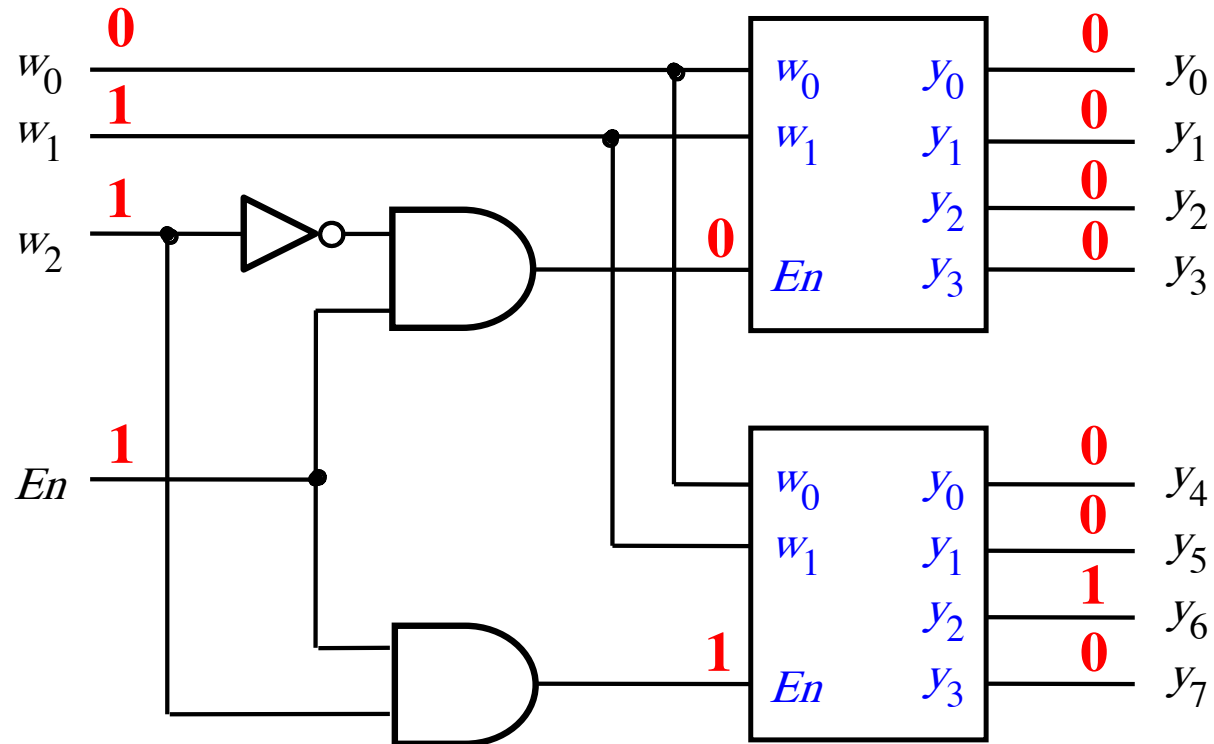


A 3-to-8 decoder using two 2-to-4 decoders

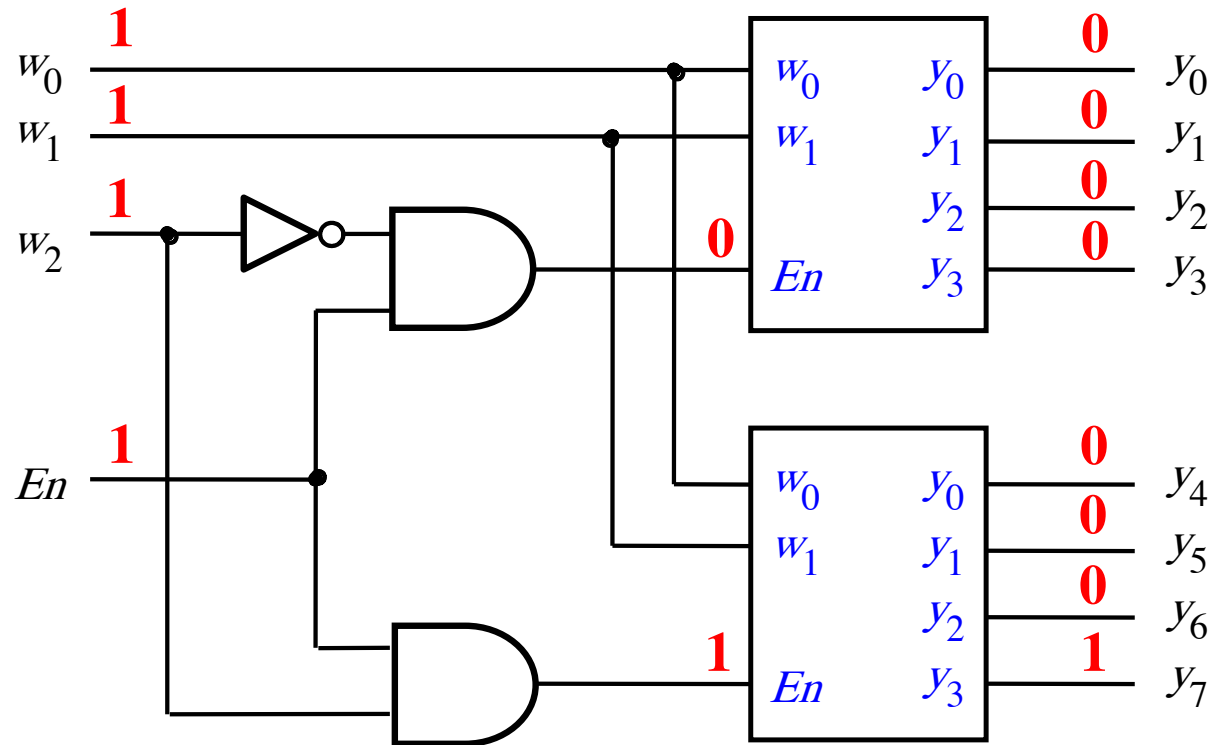


[Figure 4.15 from the textbook]

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

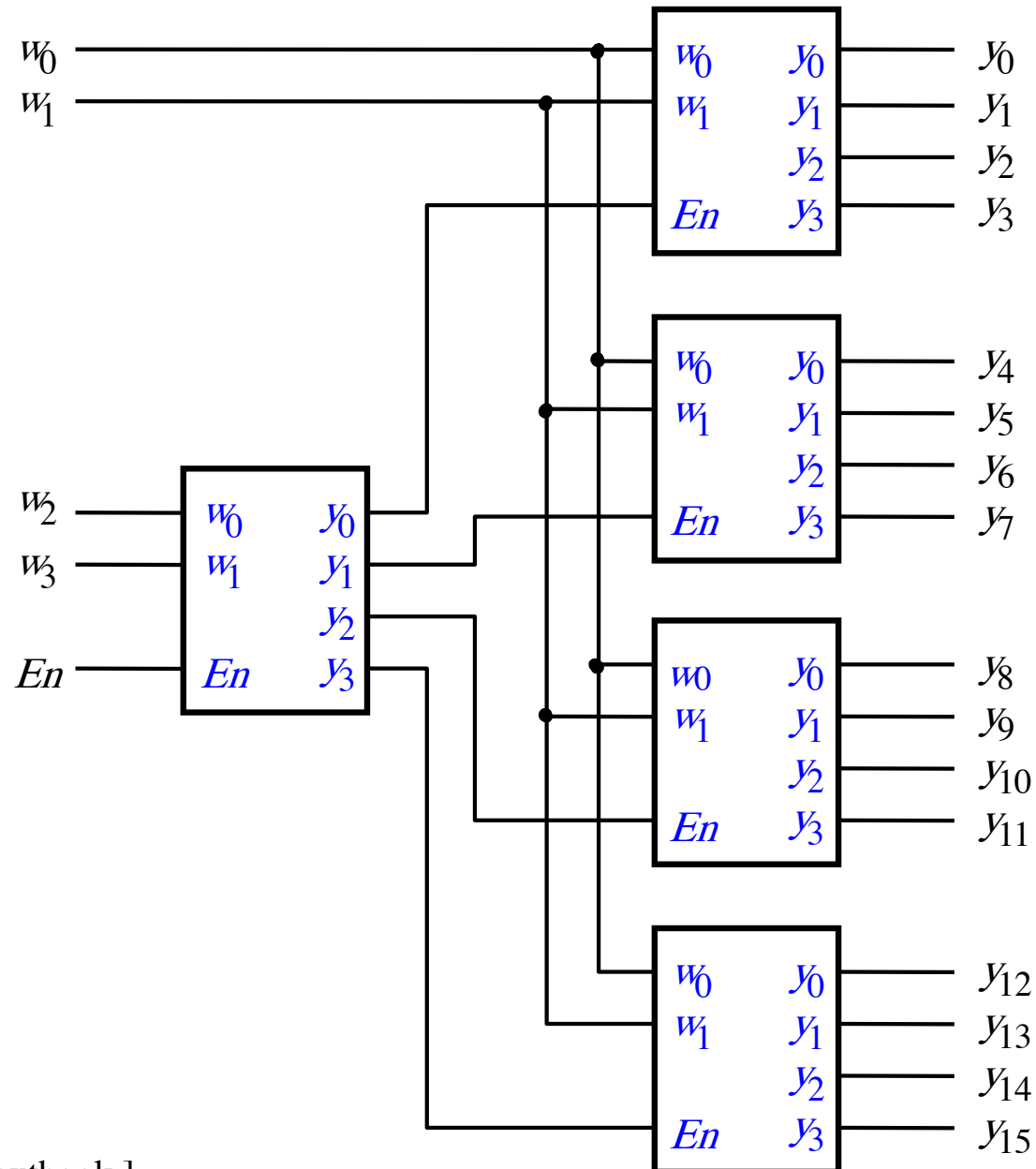


A 3-to-8 decoder using two 2-to-4 decoders



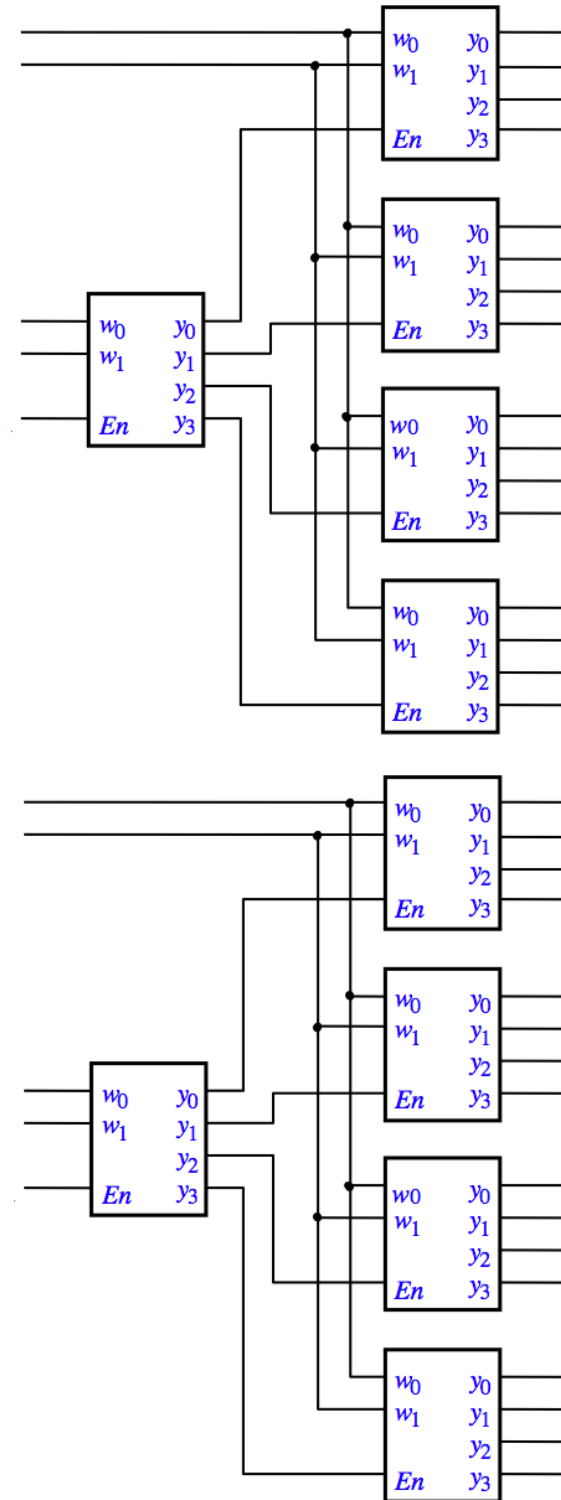
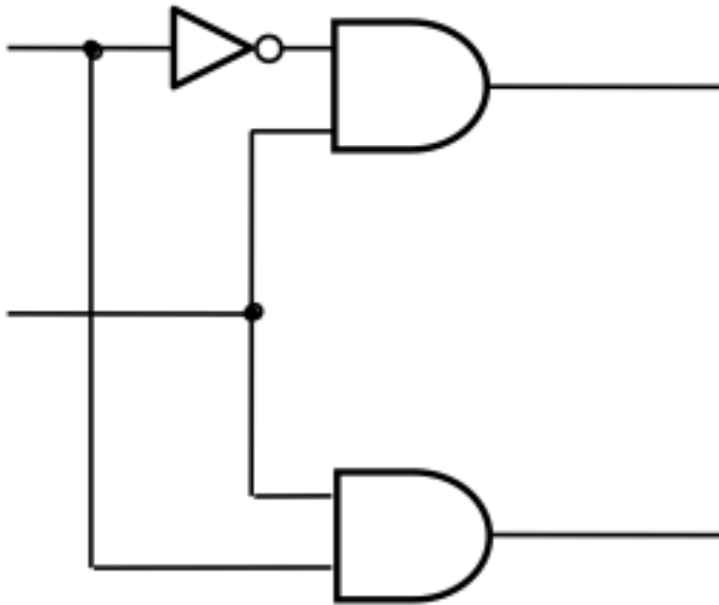
[Figure 4.15 from the textbook]

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

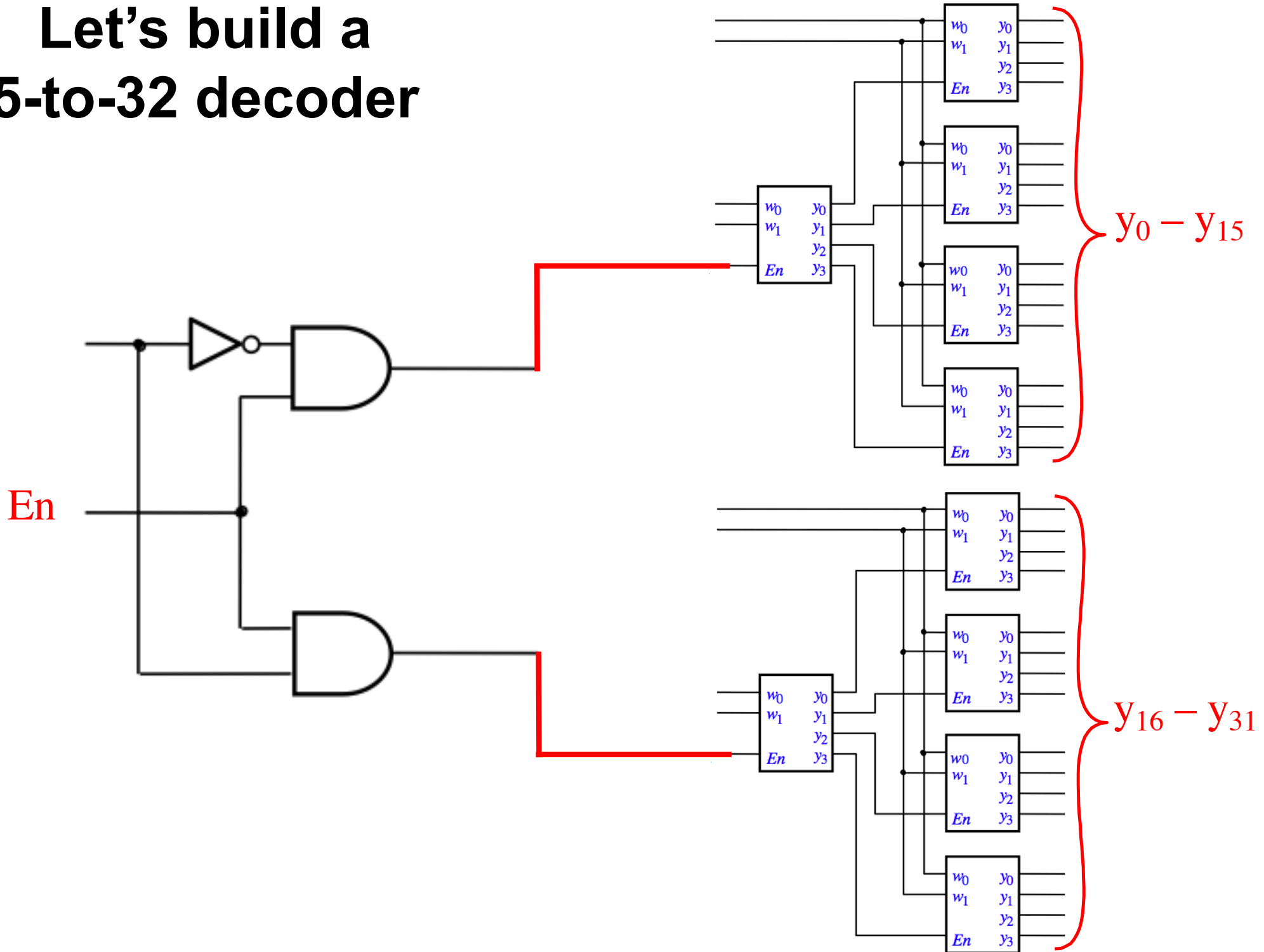


[Figure 4.16 from the textbook]

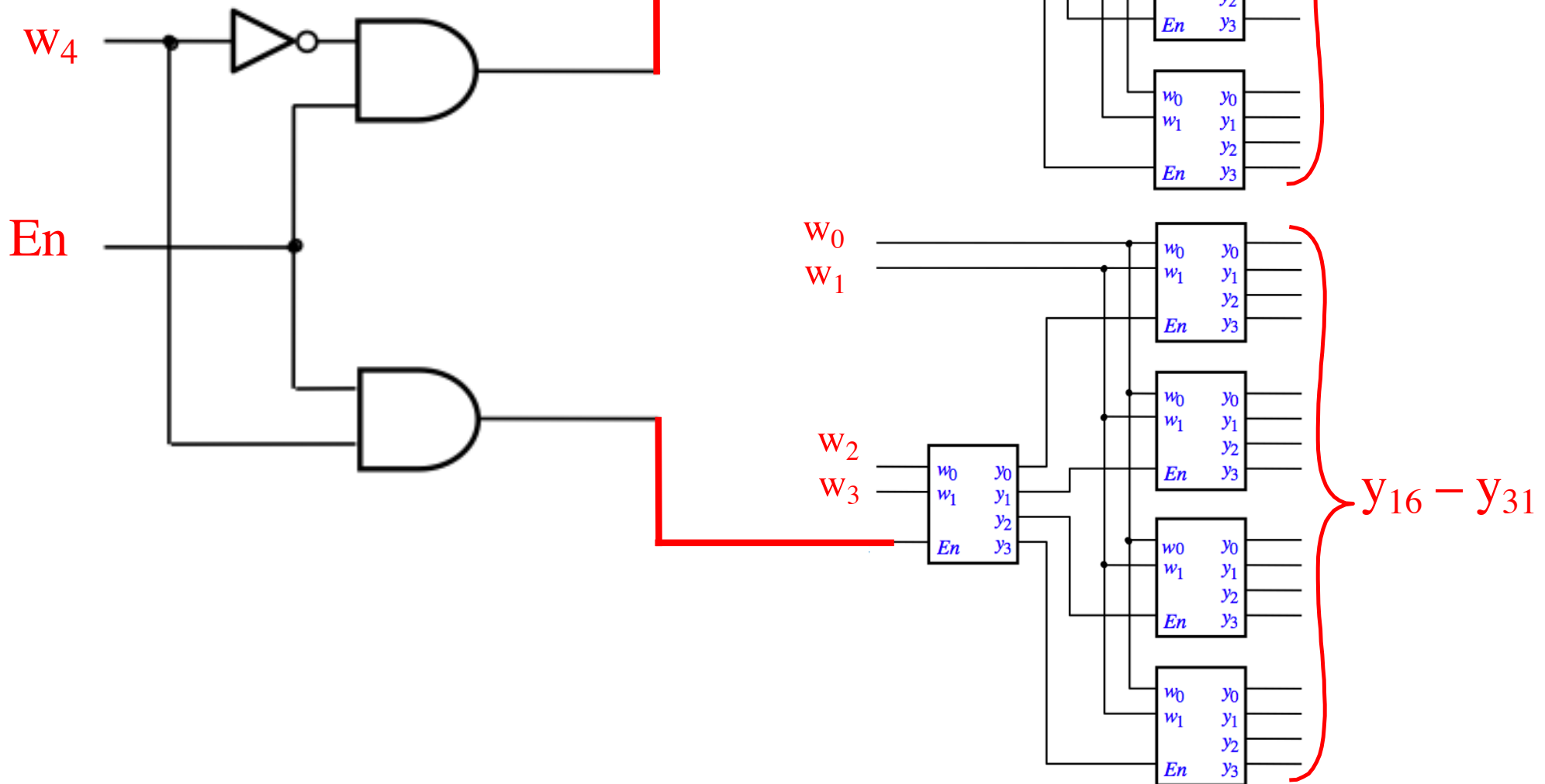
Let's build a 5-to-32 decoder



Let's build a 5-to-32 decoder



Let's build a 5-to-32 decoder

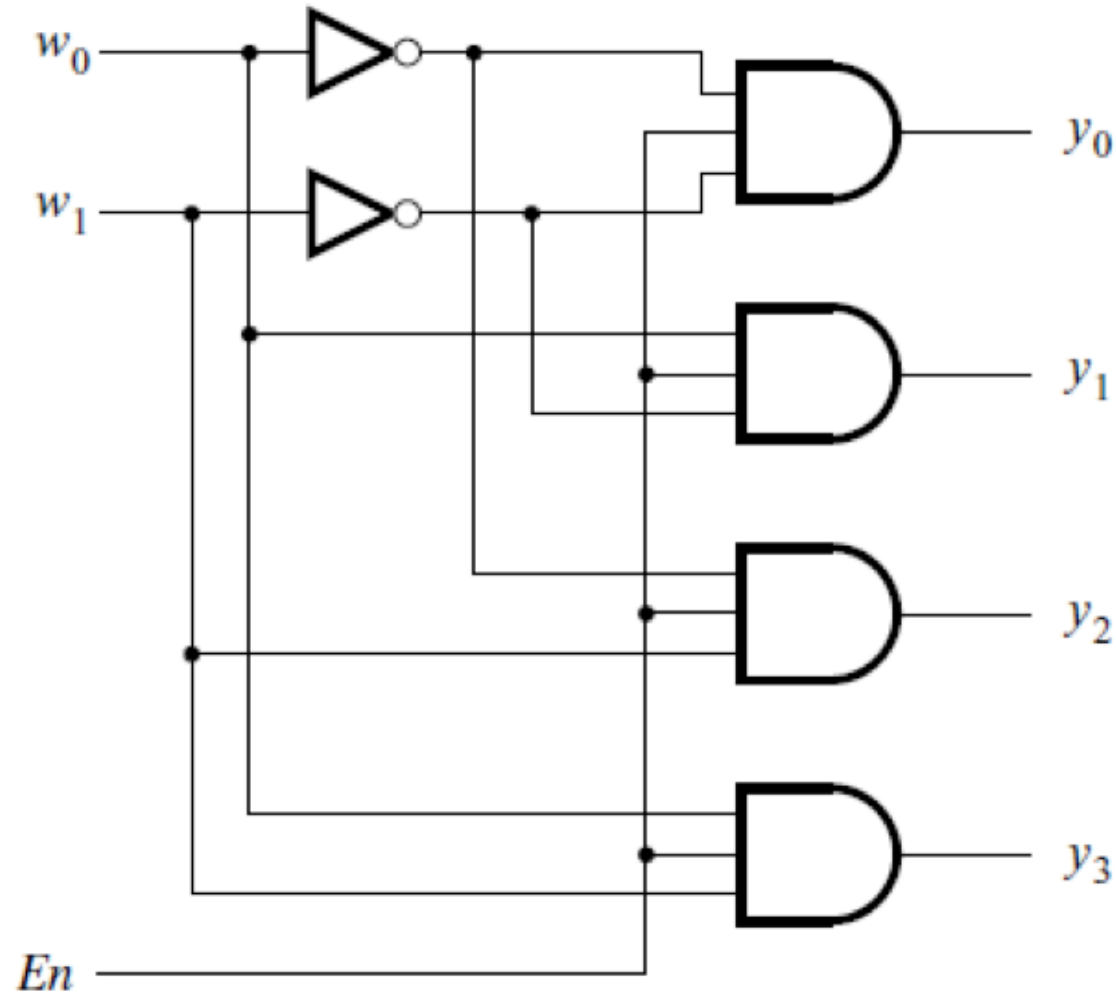


Demultiplexers

1-to-4 Demultiplexer (Definition)

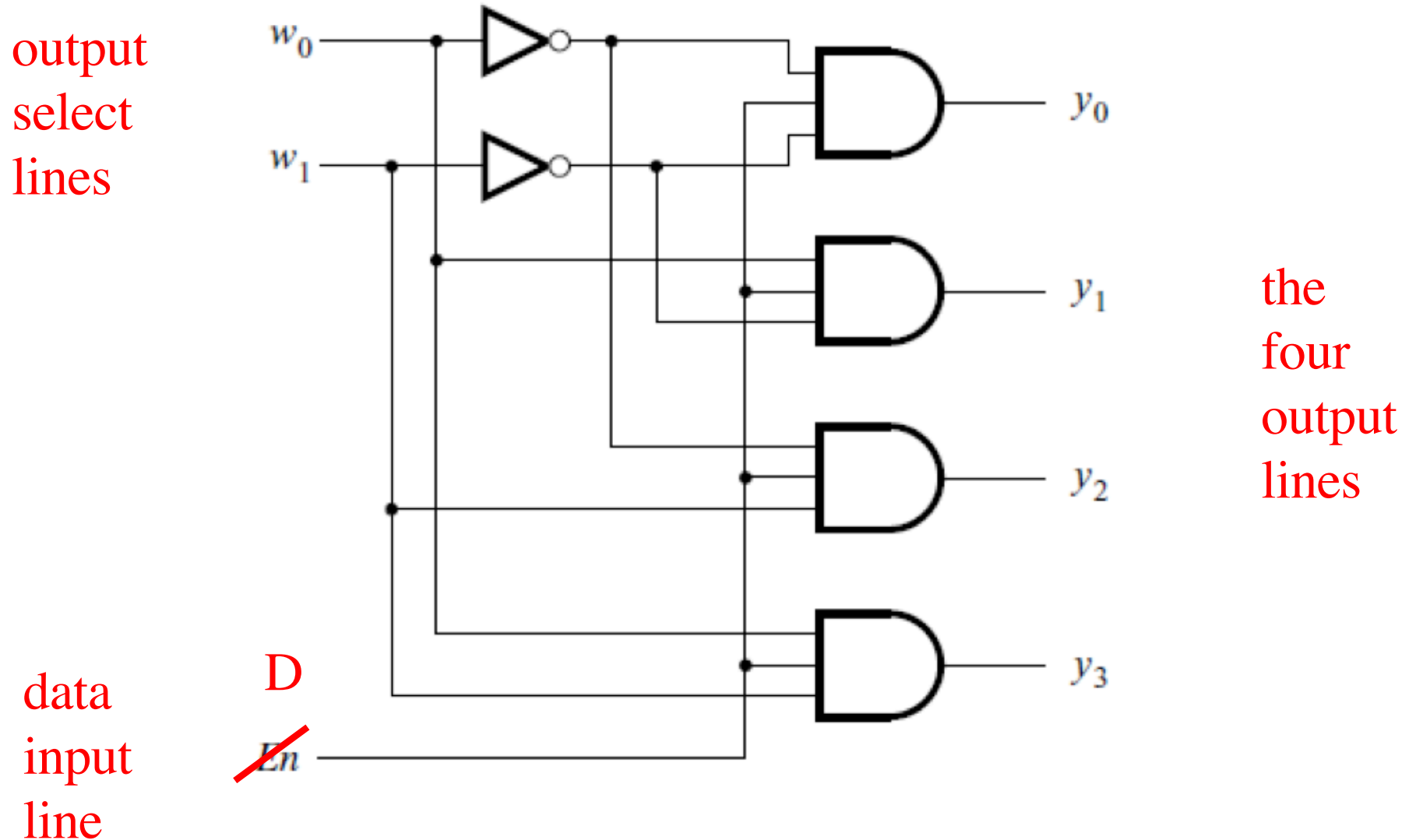
- Has one data input line: D
- Has two output select lines: w_1 and w_0
- Has four outputs: y_0 , y_1 , y_2 , and y_3
- If $w_1=0$ and $w_0=0$, then the output y_0 is set to D
- If $w_1=0$ and $w_0=1$, then the output y_1 is set to D
- If $w_1=1$ and $w_0=0$, then the output y_2 is set to D
- If $w_1=1$ 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 with enable



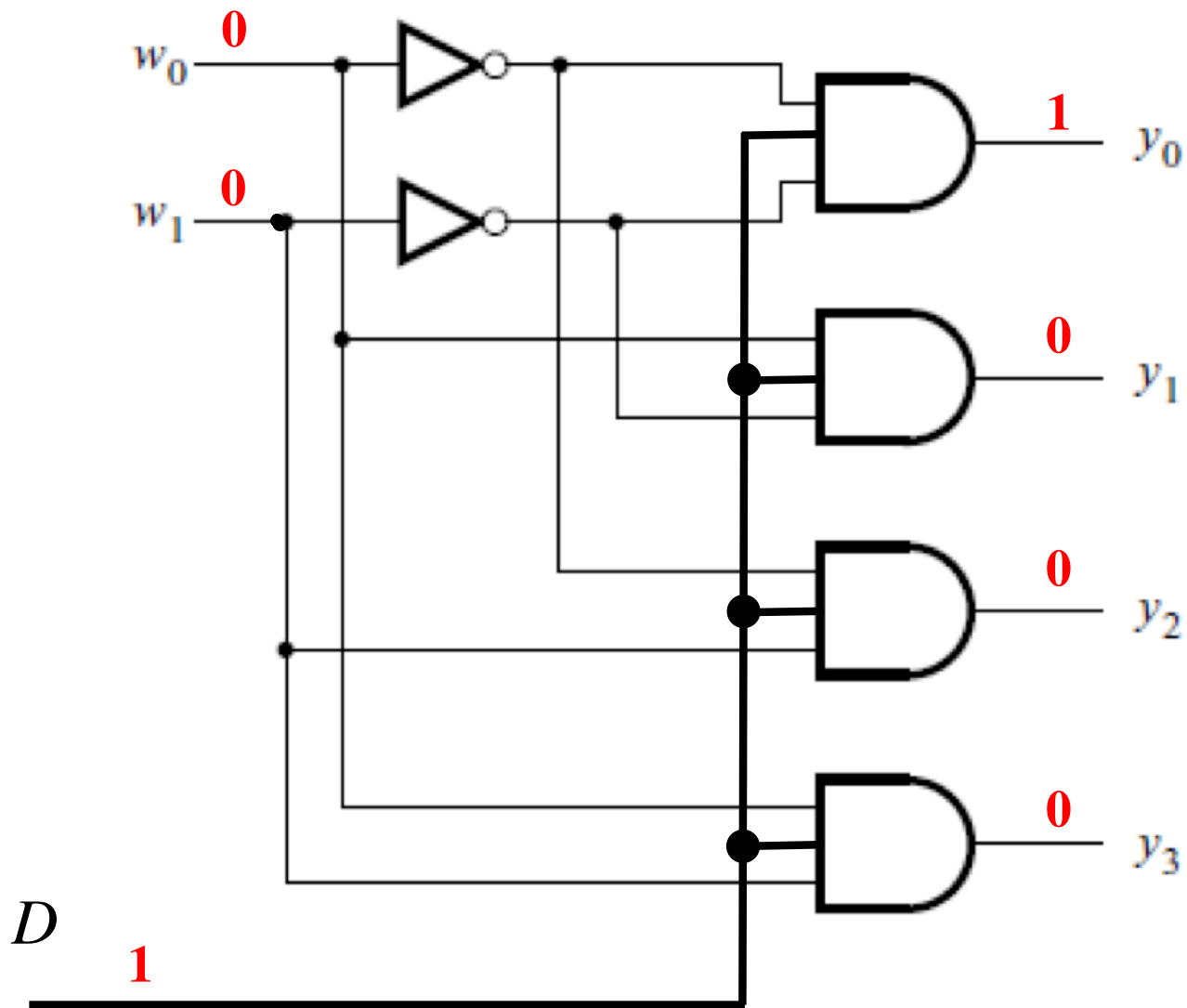
[Figure 4.14c from the textbook]

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

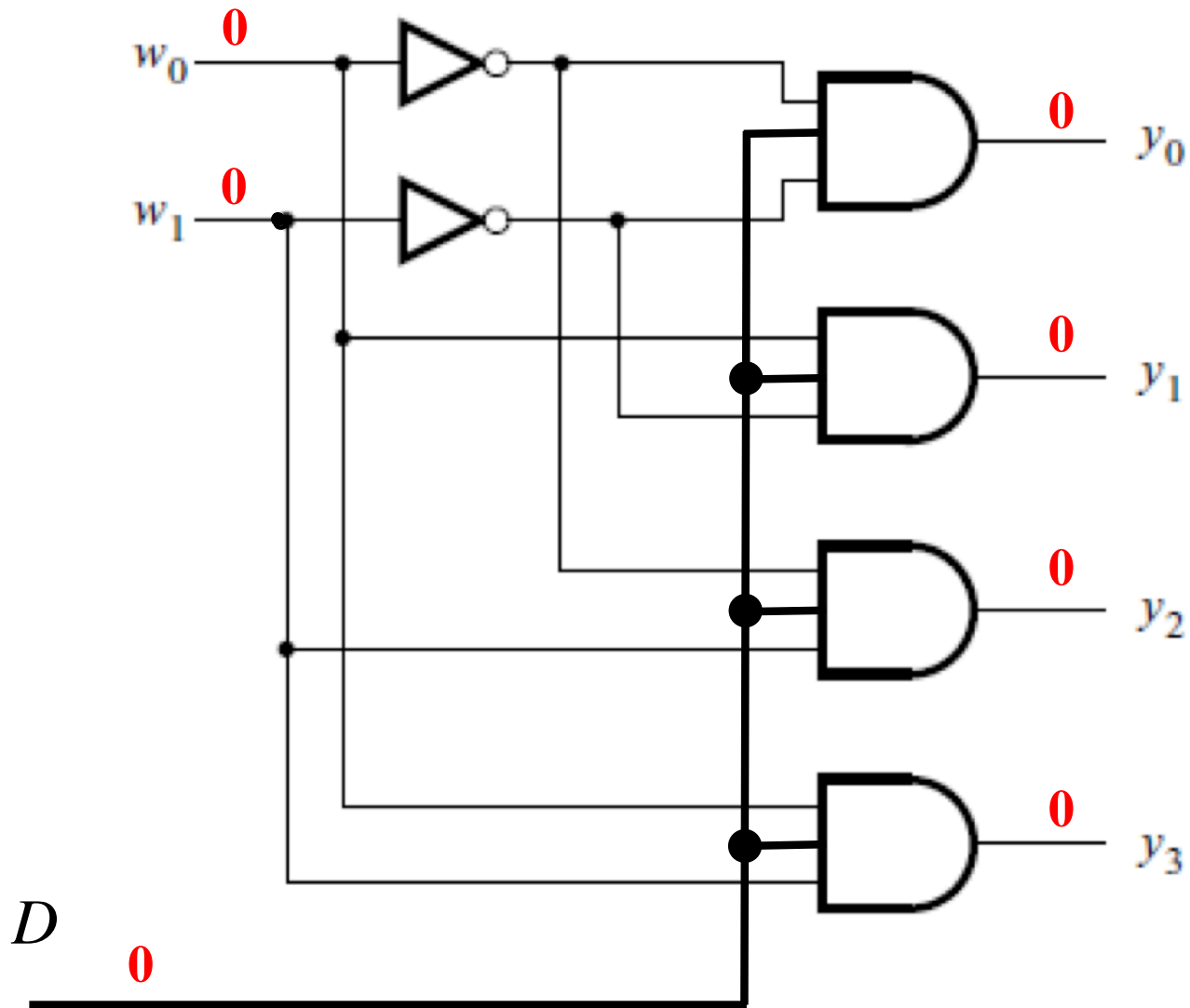


[Figure 4.14c from the textbook]

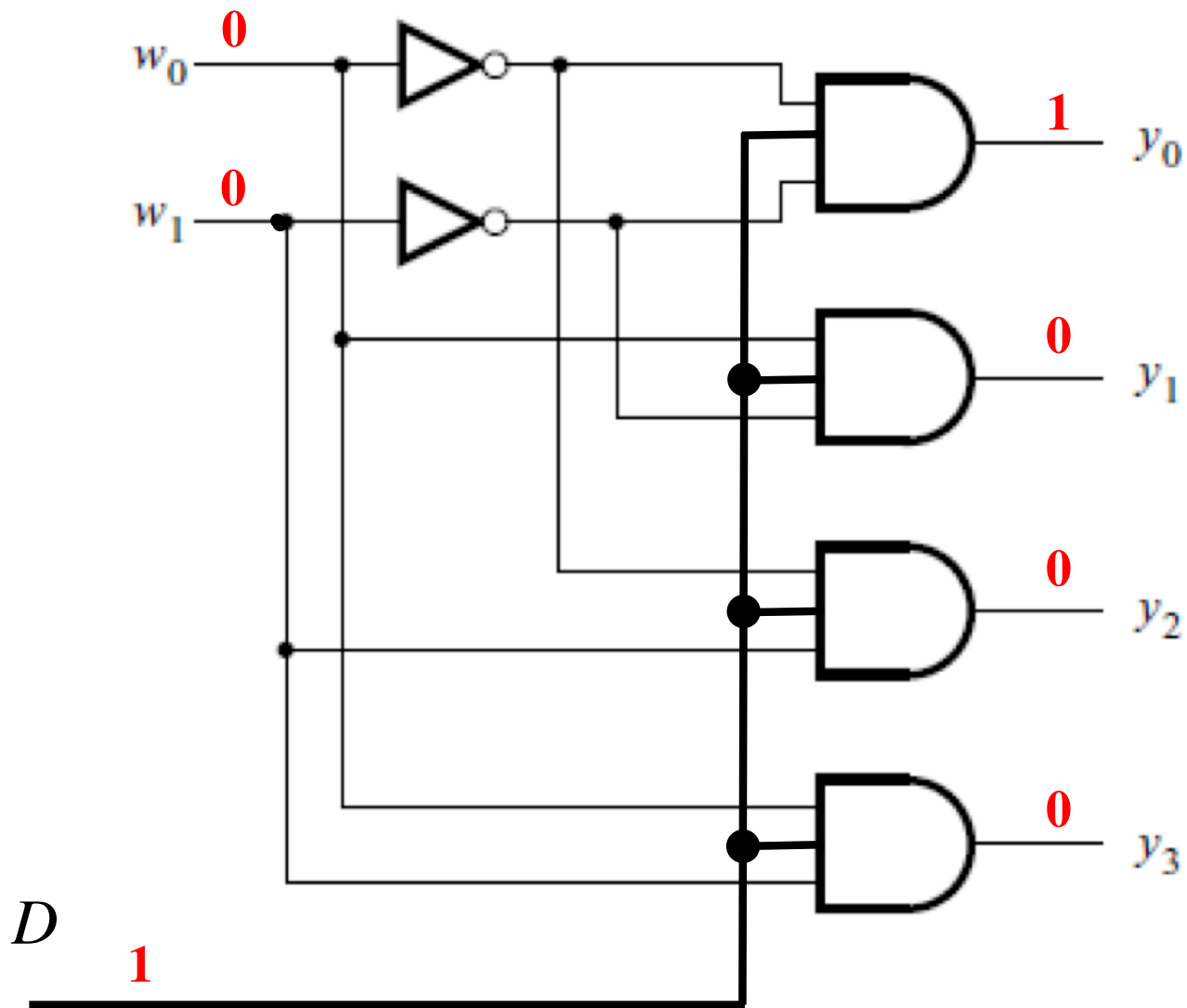
A 1-to-4 demultiplexer built with a 2-to-4 decoder with enable



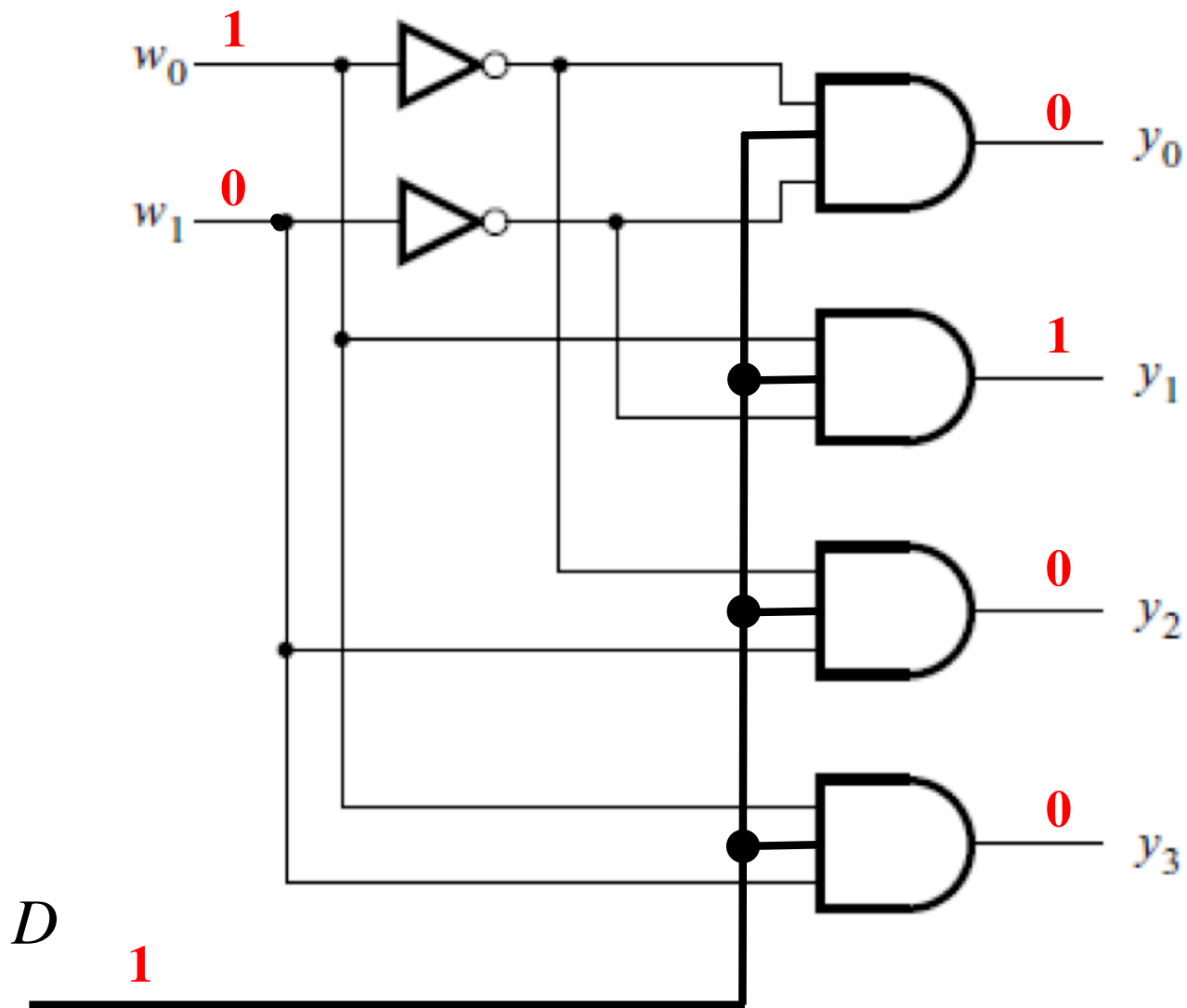
A 1-to-4 demultiplexer built with a 2-to-4 decoder with enable



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

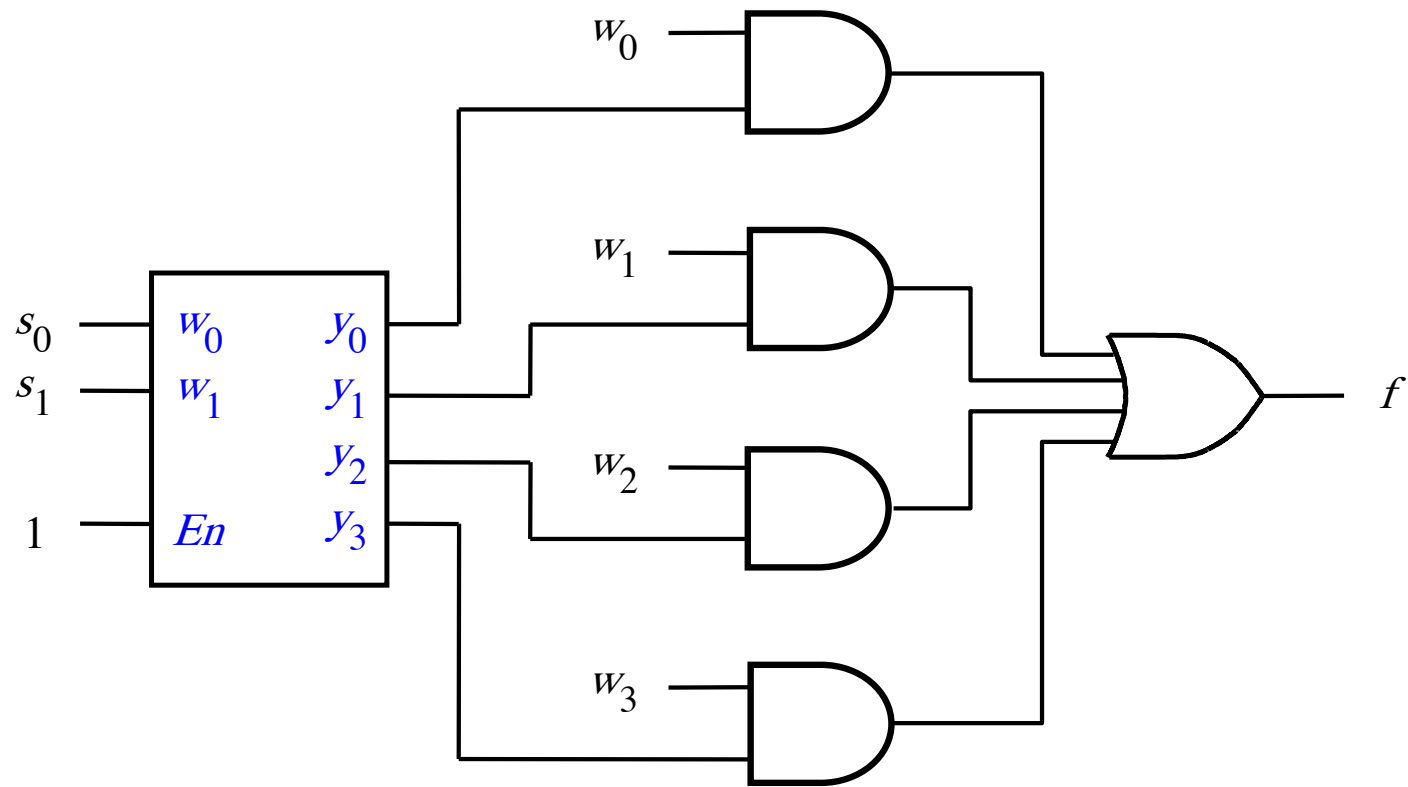


A 1-to-4 demultiplexer built with a 2-to-4 decoder with enable



Multiplexers (Implemented with Decoders)

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



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

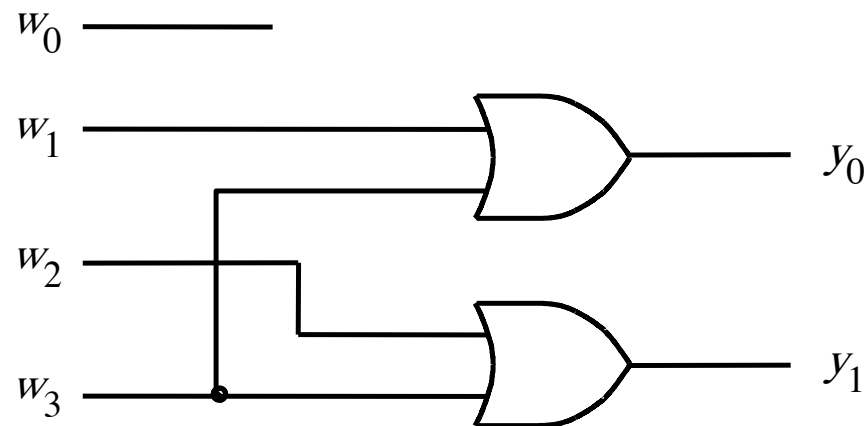
Truth table for a 4-to-2 binary encoder

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

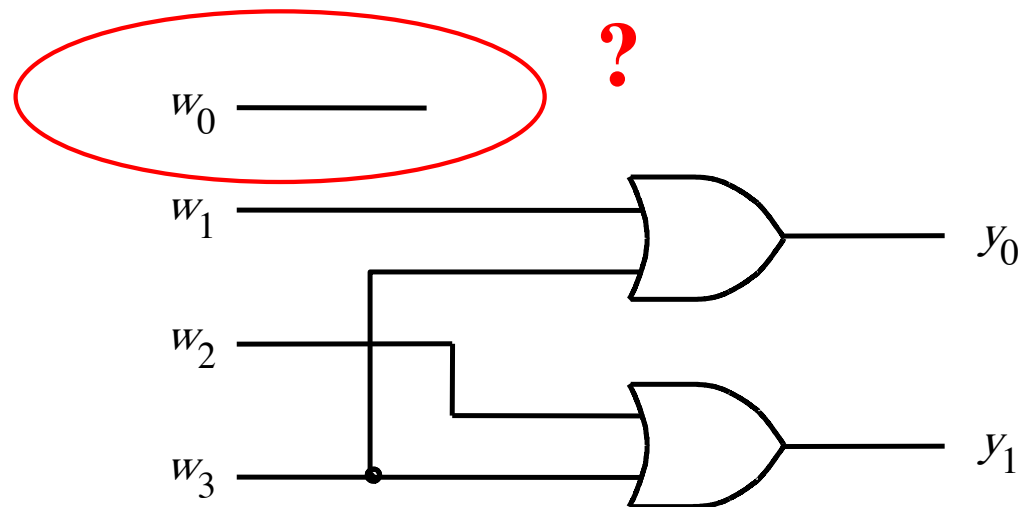
Circuit for a 4-to-2 binary encoder

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



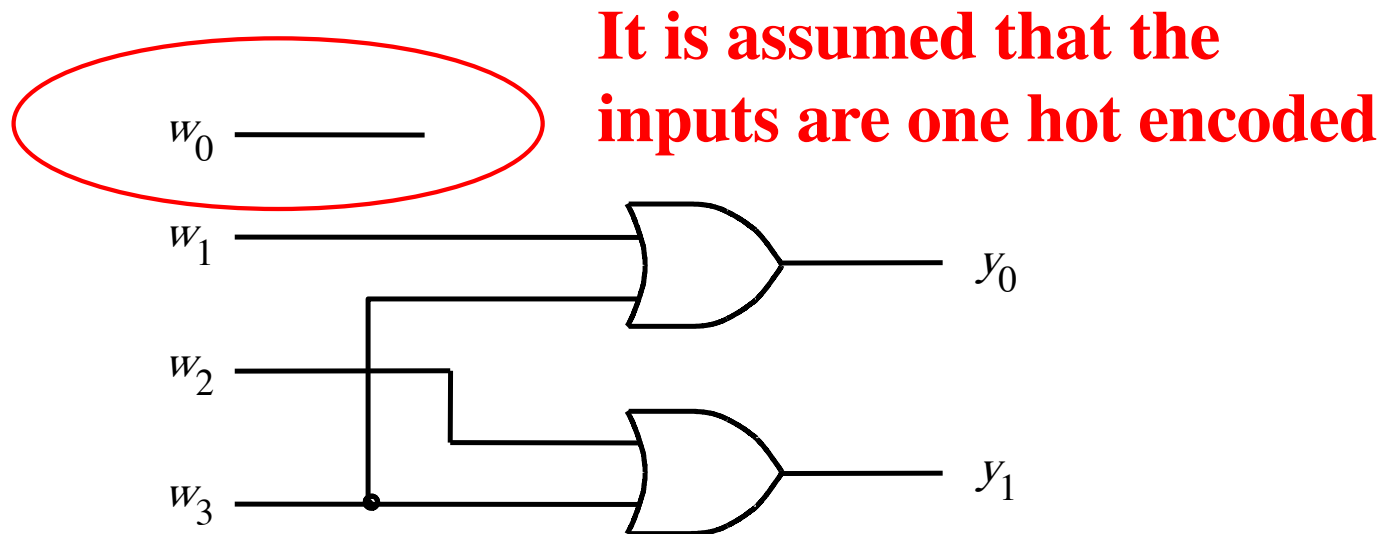
Circuit for a 4-to-2 binary encoder

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



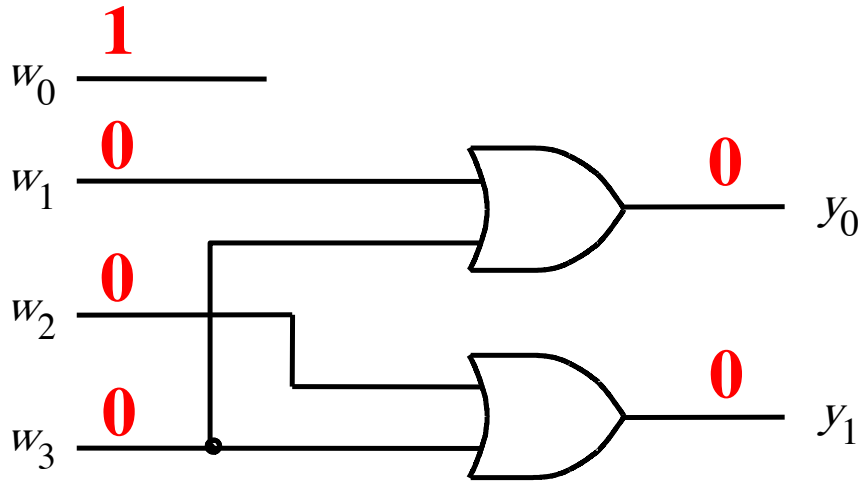
Circuit for a 4-to-2 binary encoder

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



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

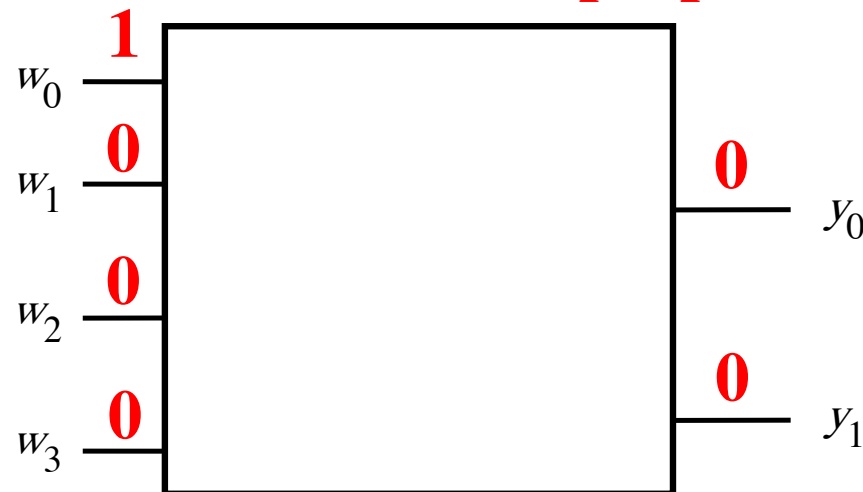


[Figure 4.19 from the textbook]

Circuit for a 4-to-2 binary encoder

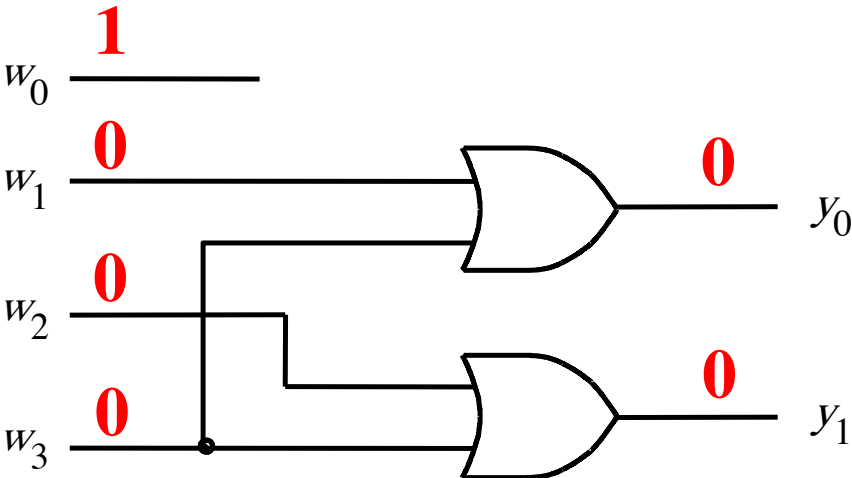
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

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



Circuit for a 4-to-2 binary encoder

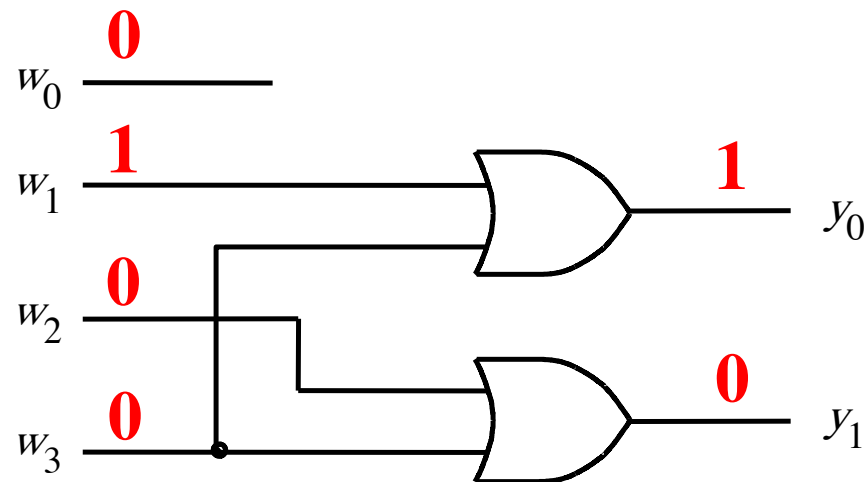
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1



[Figure 4.19 from the textbook]

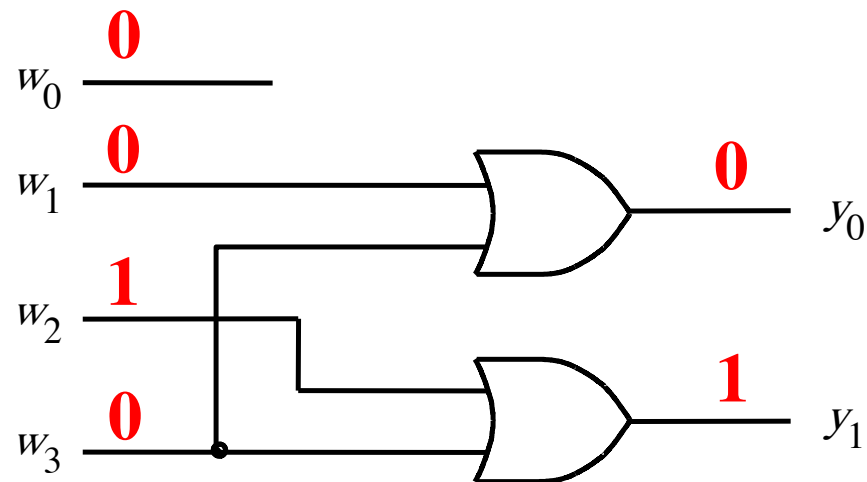
Circuit for a 4-to-2 binary encoder

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



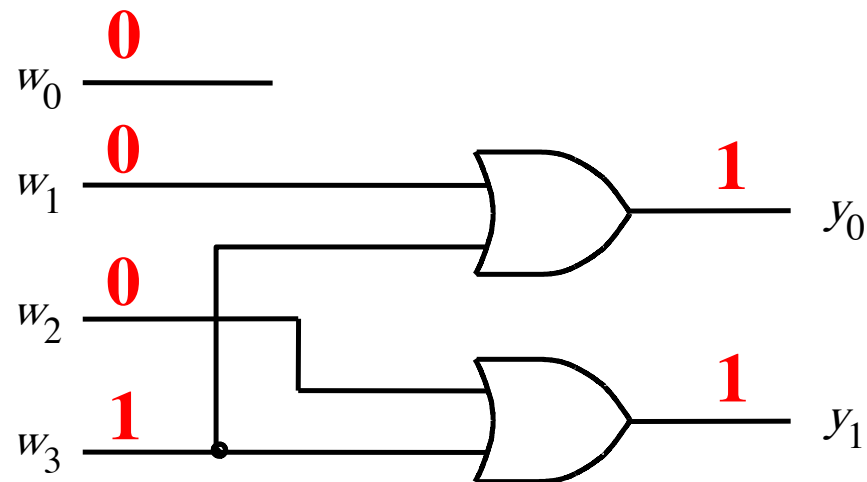
Circuit for a 4-to-2 binary encoder

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



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



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

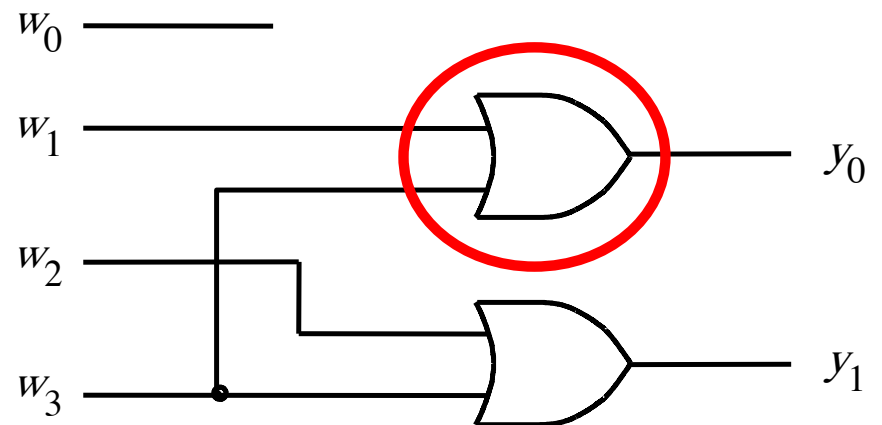
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

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

		$w_3 w_2$	00	01	11	10
$w_1 w_0$	00	d	0	d	1	
01	0	d	d	d		
11	d	d	d	d		
10	1	d	d	d		

$$y_0 = (w_1 + w_3)$$

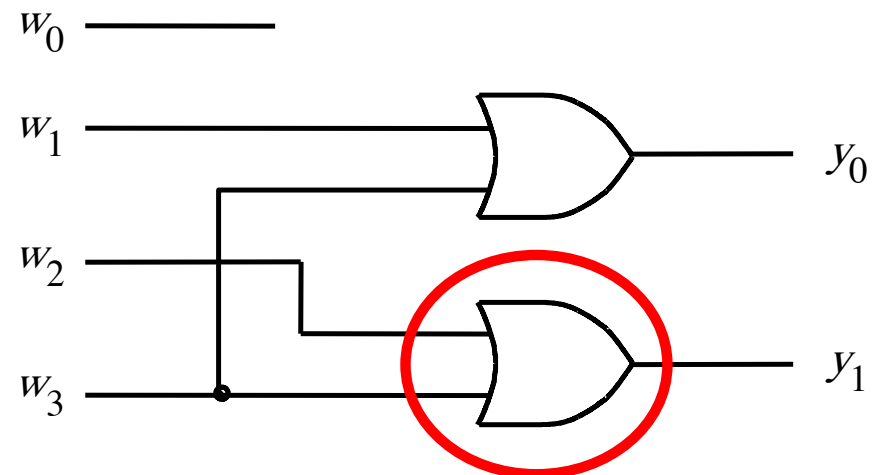


Expressions for 4-to-2 binary encoder

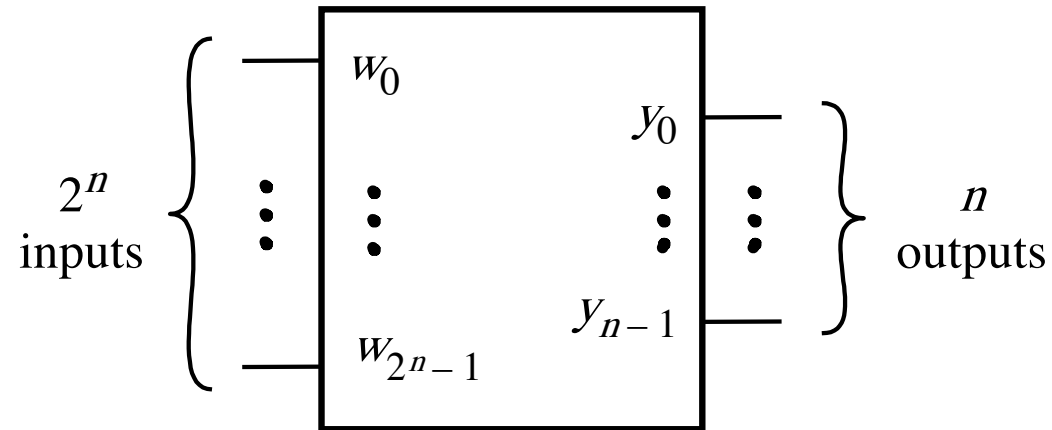
w_3	w_2	w_1	w_0	y_1	y_0
0	0	0	0	d	d
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	d	d
<hr/>					
0	1	0	0	1	0
0	1	0	1	d	d
0	1	1	0	d	d
0	1	1	1	d	d
<hr/>					
1	0	0	0	1	1
1	0	0	1	d	d
1	0	1	0	d	d
1	0	1	1	d	d
<hr/>					
1	1	0	0	d	d
1	1	0	1	d	d
1	1	1	0	d	d
1	1	1	1	d	d

$w_3 \ w_2$	$w_1 \ w_0$	00	01	11	10
00	00	d	1	d	1
01	00	0	d	d	d
11	00	d	d	d	d
10	00	0	d	d	d

$$y_1 = (w_3 + w_2)$$



A 2^n -to- n binary encoder



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

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

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

		$w_3 w_2$			
$w_1 w_0$		00	01	11	10
	00	d	1	1	1
01	0	1	1	1	
11	0	1	1	1	
10	0	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

		$w_3 w_2$			
$w_1 w_0$		00	01	11	10
00		d	0	1	1
01		0	0	1	1
11		1	0	1	1
10		1	0	1	1

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

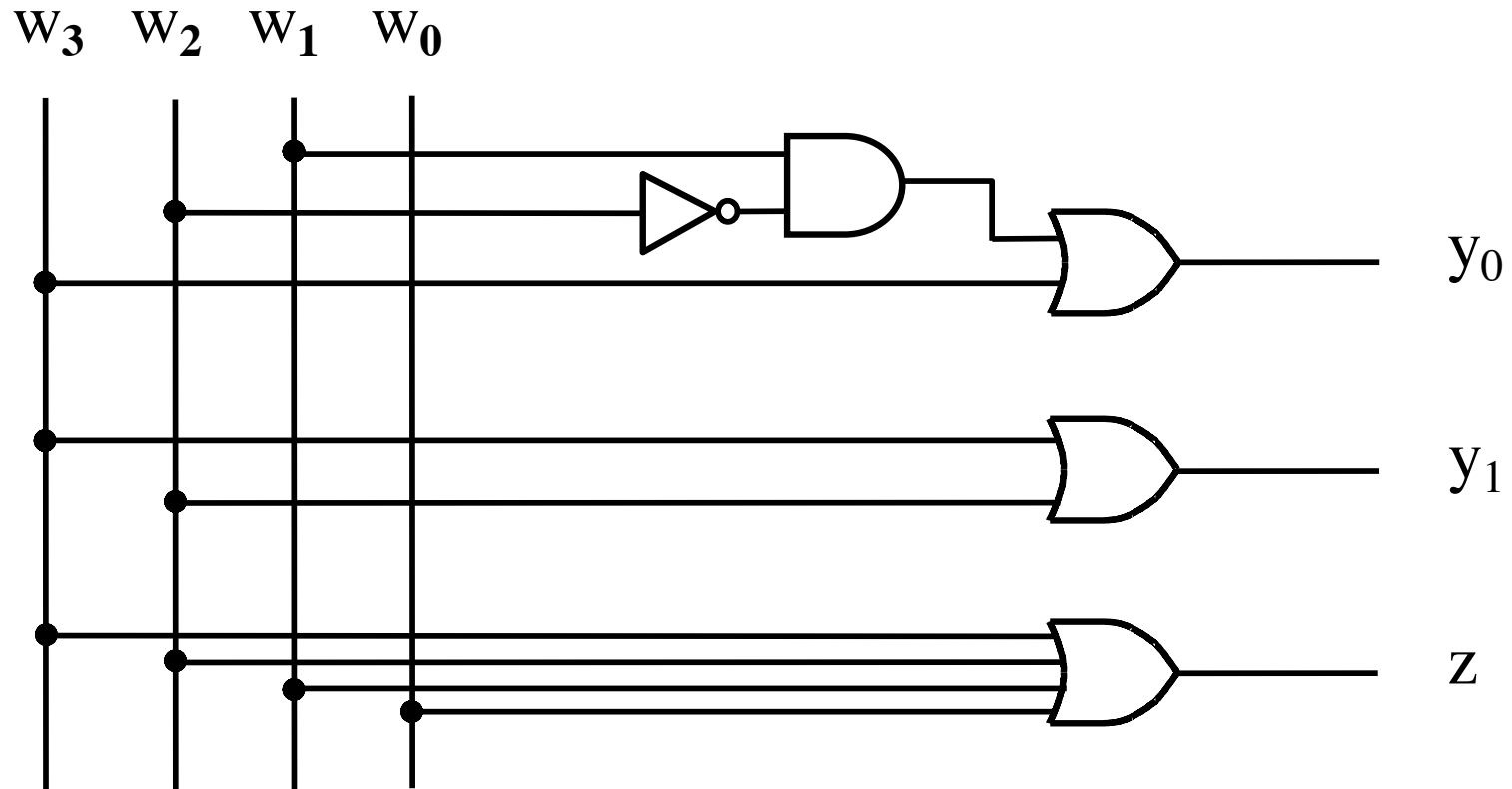
Expressions for 4-to-2 priority encoder

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

		$w_3 w_2$			
$w_1 w_0$		00	01	11	10
	00	0	1	1	1
01	1	1	1	1	
11	1	1	1	1	
10	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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

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

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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

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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

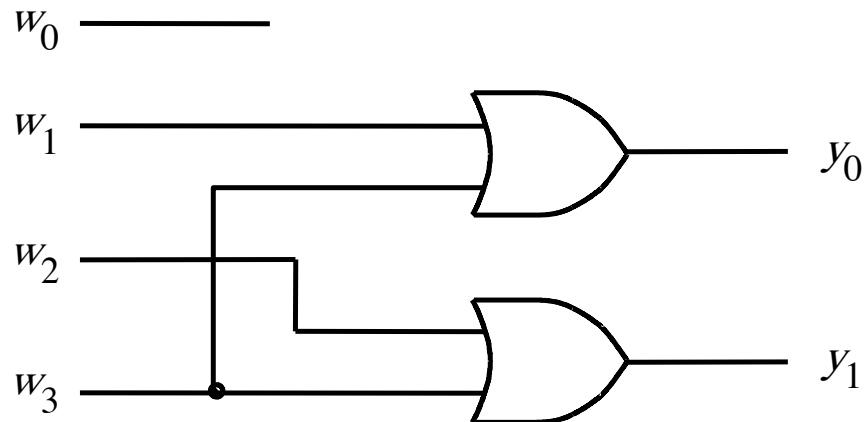
$$i_2 = \bar{w}_3 w_2$$

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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



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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

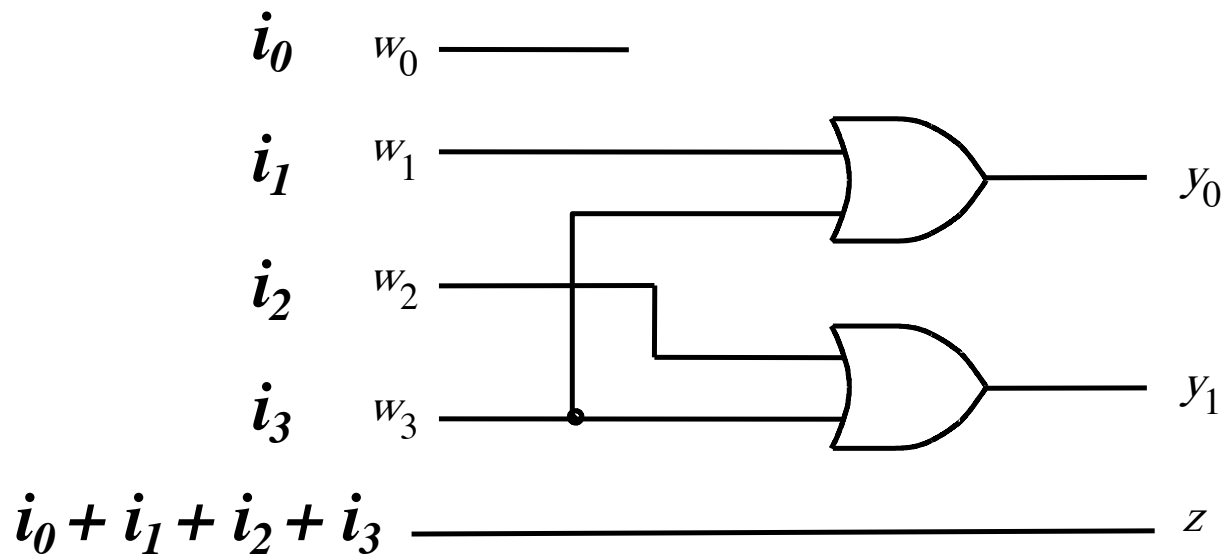
$$i_2 = \bar{w}_3 w_2$$

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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



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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

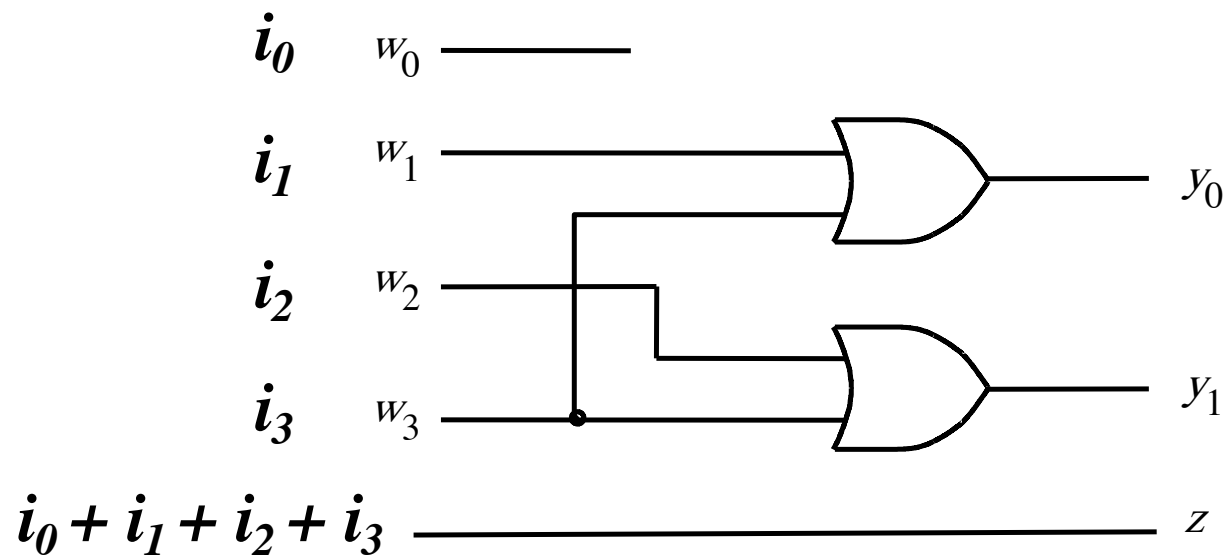
$$i_2 = \bar{w}_3 w_2$$

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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



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

Let's prove this for z

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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

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

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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

		$w_3 w_2$			
		00	01	11	10
$w_1 w_0$	00	0	1	1	1
	01	1	1	1	1
	11	1	1	1	1
	10	1	1	1	1

$z = ?$

Let's prove this for z

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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

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

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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

		$w_3 w_2$			
		00	01	11	10
$w_1 w_0$	00	0	1	1	1
	01	1	1	1	1
	11	1	1	1	1
	10	1	1	1	1

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

Let's prove this for y_0

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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

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

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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

		$w_3 w_2$			
		00	01	11	10
$w_1 w_0$	00	0	0	1	1
	01	0	0	1	1
	11	1	0	1	1
	10	1	0	1	1

$y_0 = ?$

Let's prove this for y_0

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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

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

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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

		w_3	w_2		
				w_1	w_0
		00	01	11	10
00		0	0	1	1
01		0	0	1	1
11		1	0	1	1
10		1	0	1	1

$$y_0 = w_3 + w_1 \bar{w}_2$$

Let's prove this for y_1

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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

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

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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

		w_3	w_2		
				w_1	w_0
		00	01	11	10
00		0	1	1	1
01		0	1	1	1
11		0	1	1	1
10		0	1	1	1

$$y_1 = ?$$

Let's prove this for y_1

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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

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

$$i_3 = w_3$$

$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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

		$w_3 w_2$			
		00	01	11	10
$w_1 w_0$	00	0	1	1	1
	01	0	1	1	1
	11	0	1	1	1
	10	0	1	1	1

$$y_1 = w_3 + w_2$$

Therefore, this circuit for the 4-to-2 priority encoder is equivalent to ..

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

$$i_0 = \bar{w}_3 \bar{w}_2 \bar{w}_1 w_0$$

$$i_1 = \bar{w}_3 \bar{w}_2 w_1$$

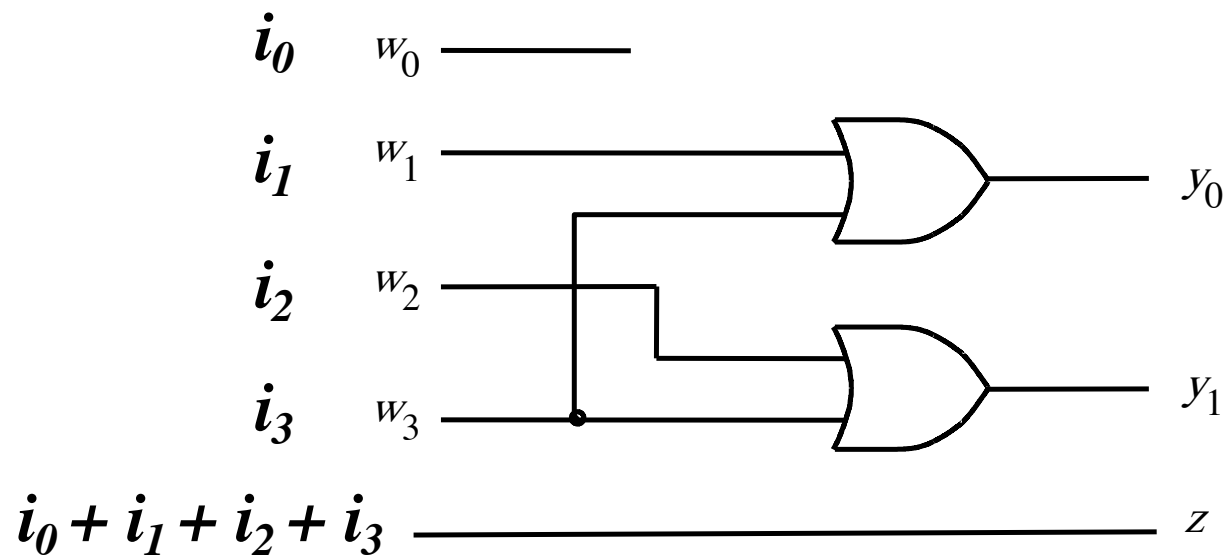
$$i_2 = \bar{w}_3 w_2$$

$$i_3 = w_3$$

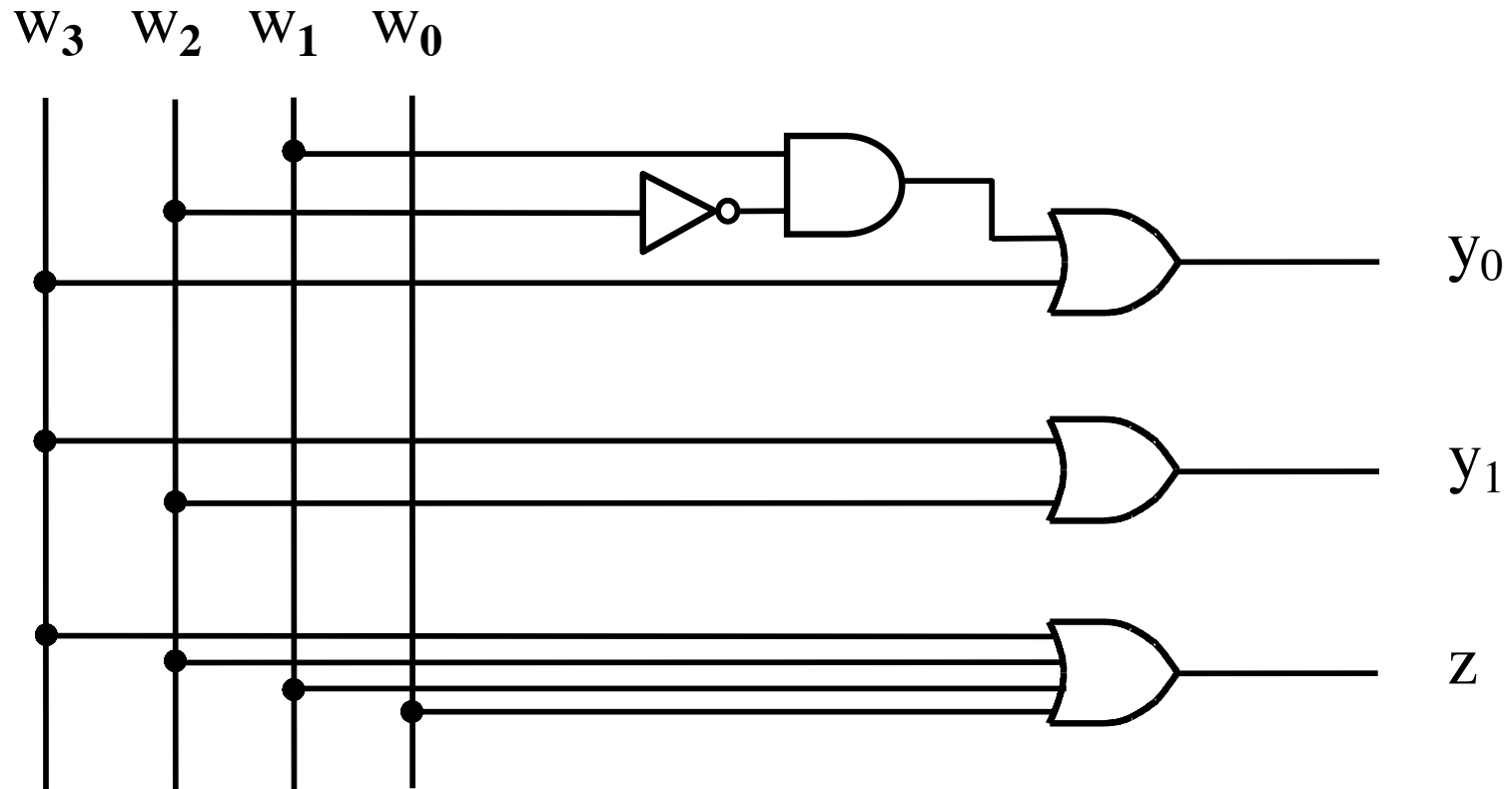
$$y_0 = i_1 + i_3$$

$$y_1 = i_2 + i_3$$

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



... this circuit for the 4-to-2 priority encoder



Code Converters

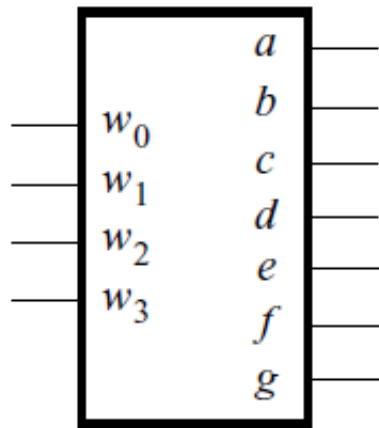
Code Converter (Definition)

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

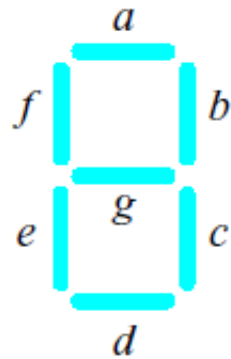
Code Converter (Definition)

- **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

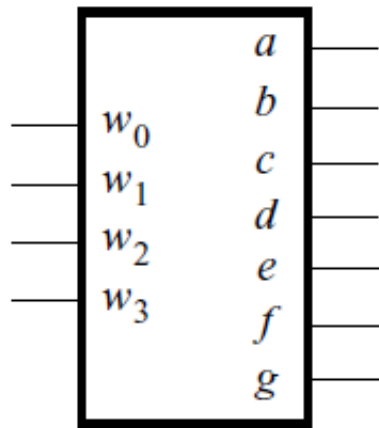


(b) 7-segment display

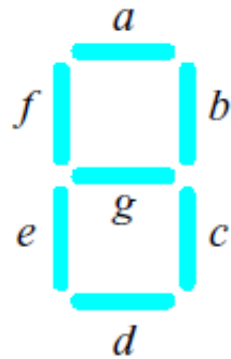
w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0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

A hex-to-7-segment display code converter



(a) Code converter

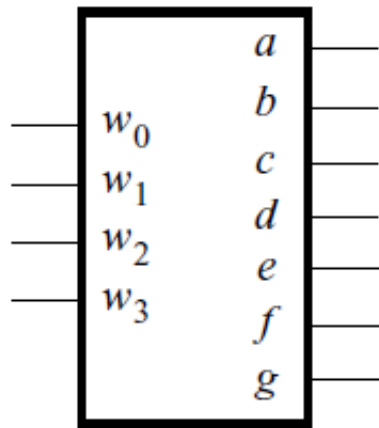


(b) 7-segment display

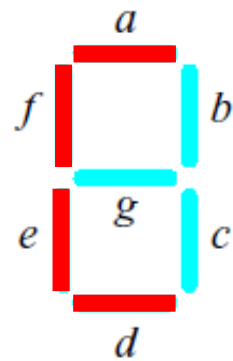
w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0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

A hex-to-7-segment display code converter



(a) Code converter

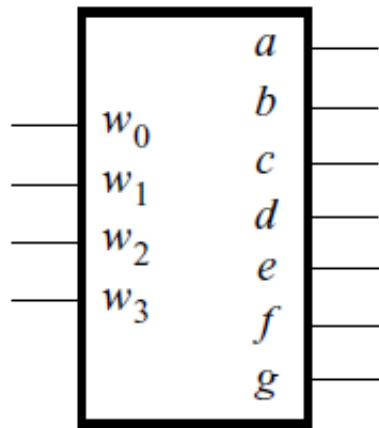


(b) 7-segment display

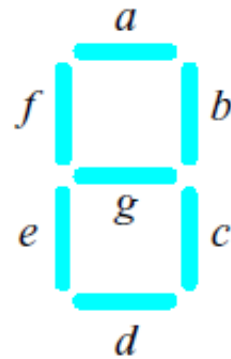
w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0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

A hex-to-7-segment display code converter



(a) Code converter

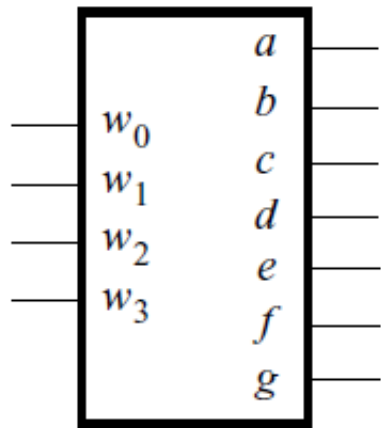


(b) 7-segment display

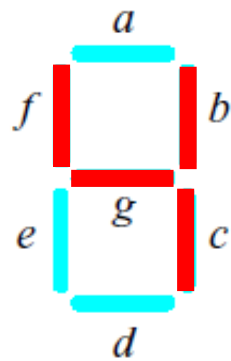
w_3	w_2	w_1	w_0	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0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

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

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

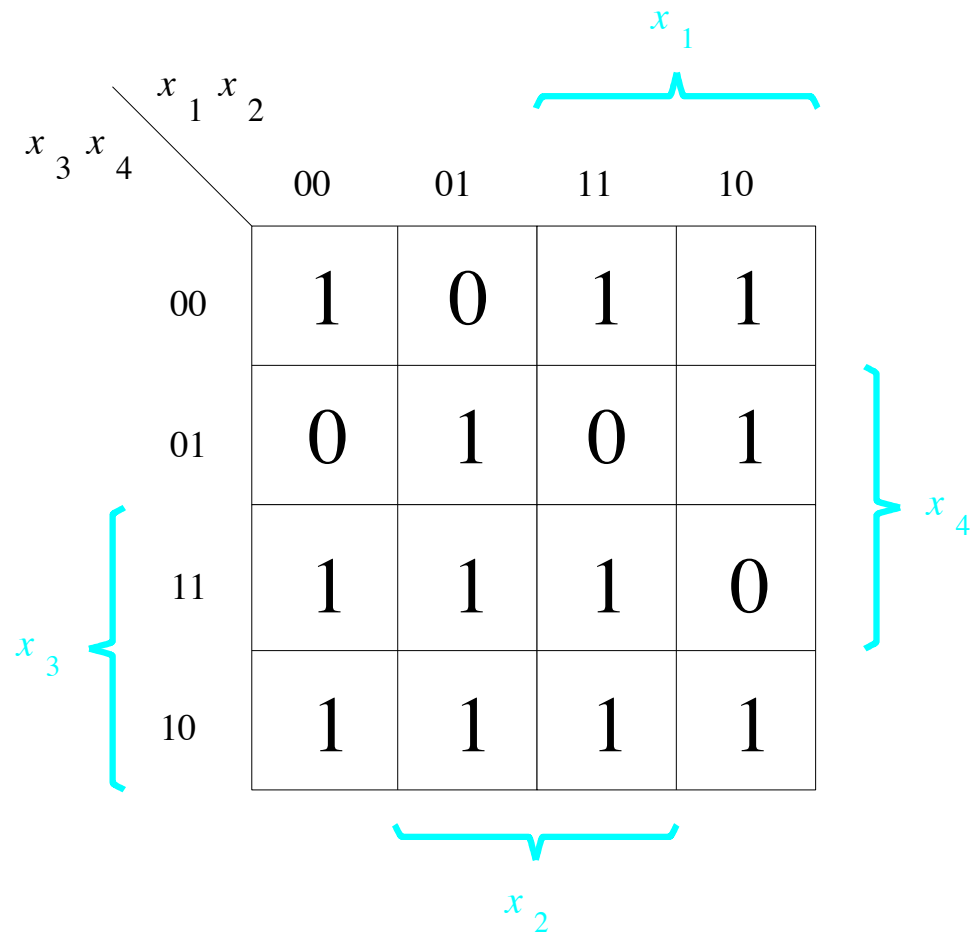
What is the circuit for this code converter?

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

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

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



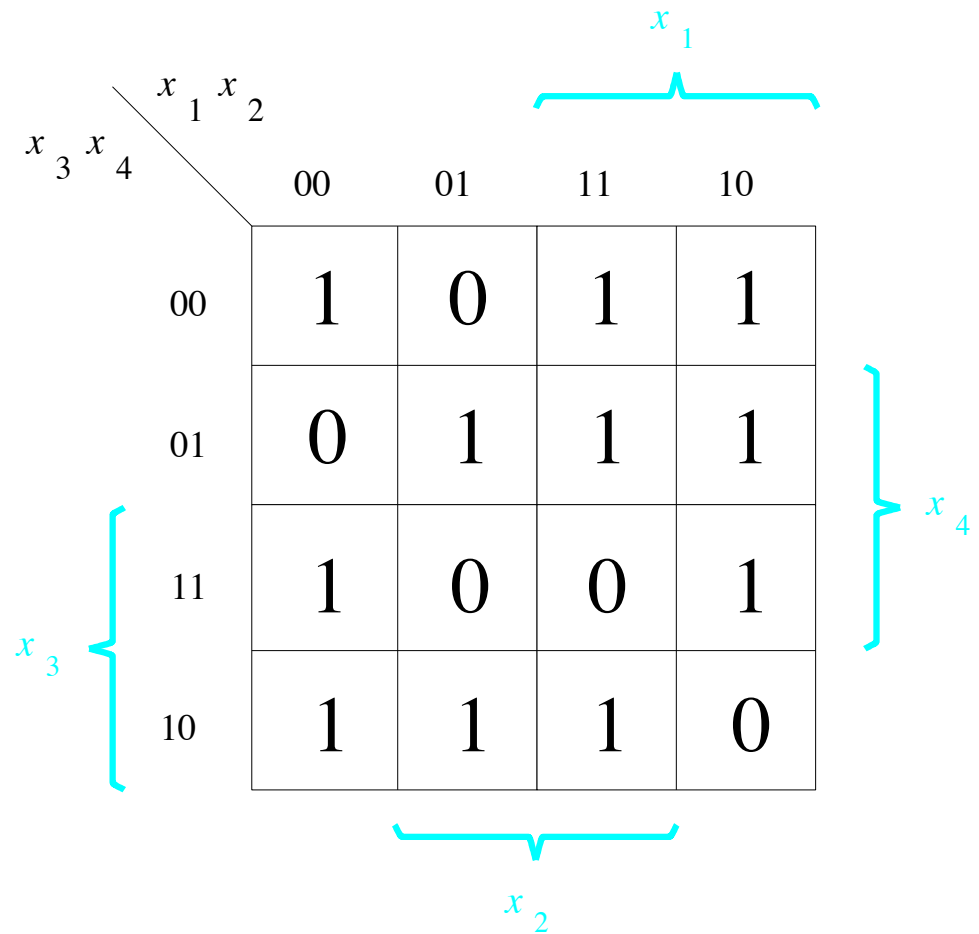
$$f(x_1, x_2, x_3, x_4) = \sum m(0, 2, 3, 5, 6, 7, 8, 9, 10, 12, 14, 15)$$

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



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

Example Problems from Chapter 4

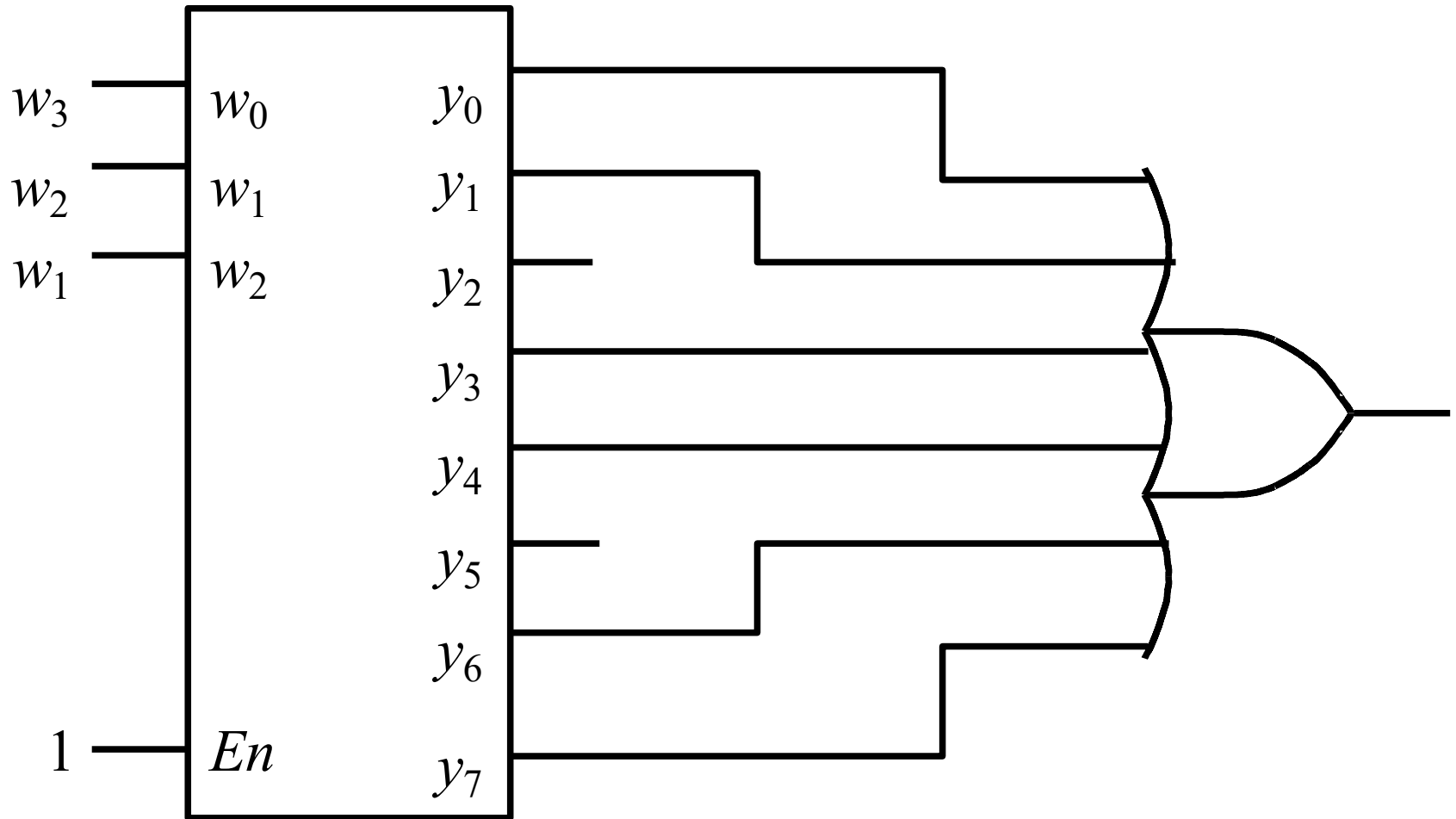
Example 1: SOP vs Decoders

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.

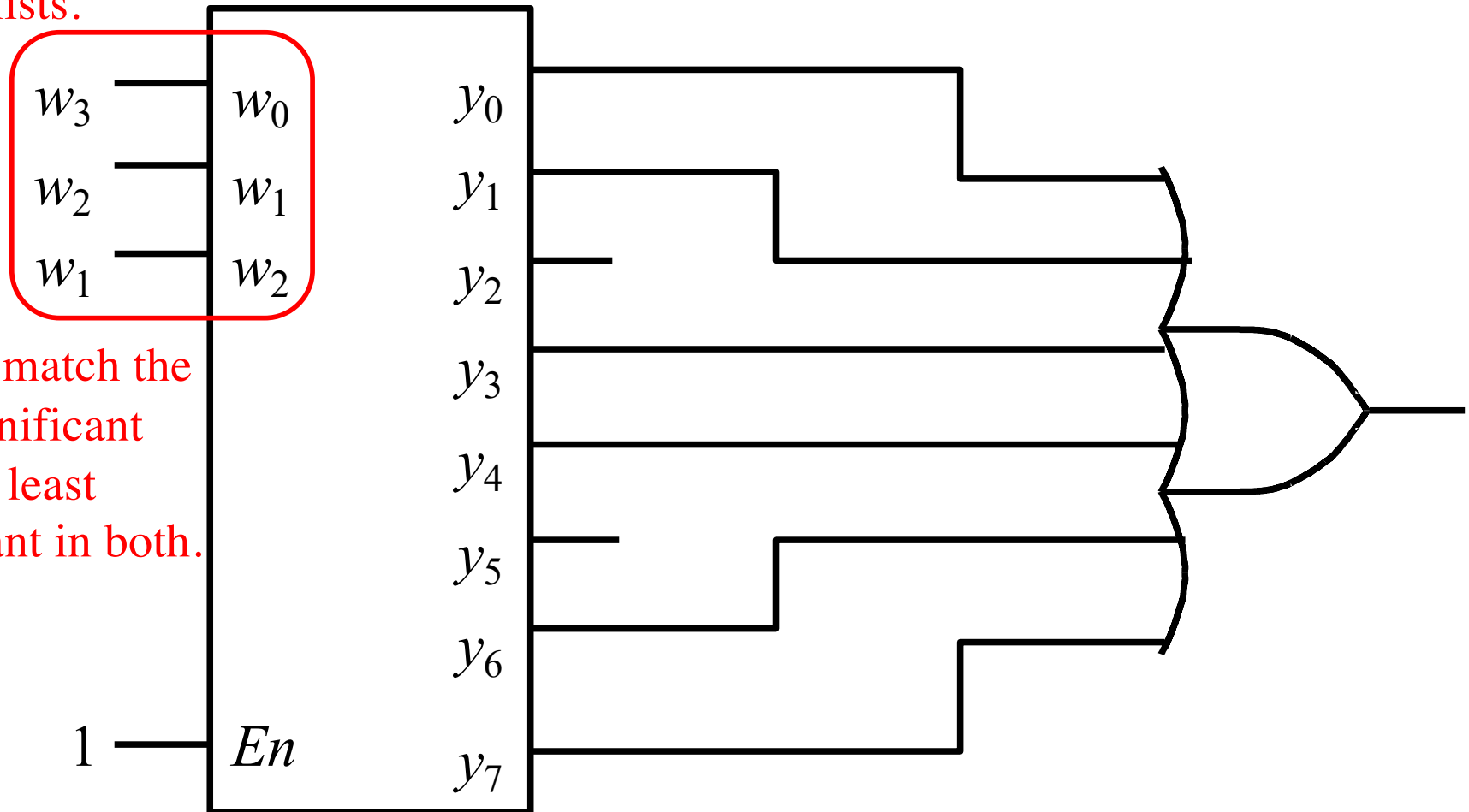
Solution Circuit



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

Solution Circuit

Notice this swap
of variables in
the two lists.



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

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

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

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

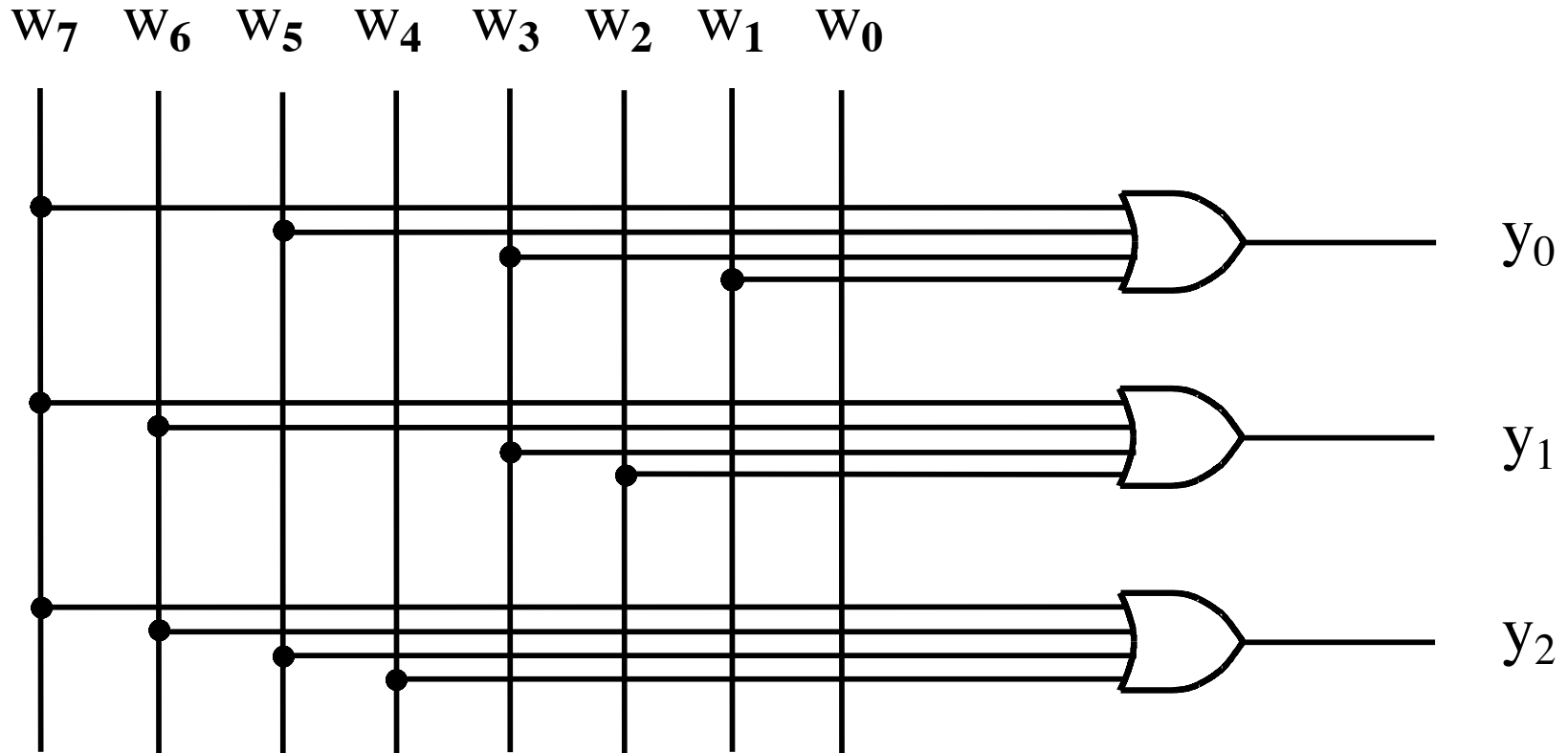
w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

$$y_0 = w_1 + w_3 + w_5 + w_7$$

$$y_1 = w_2 + w_3 + w_6 + w_7$$

$$y_2 = w_4 + w_5 + w_6 + w_7$$

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

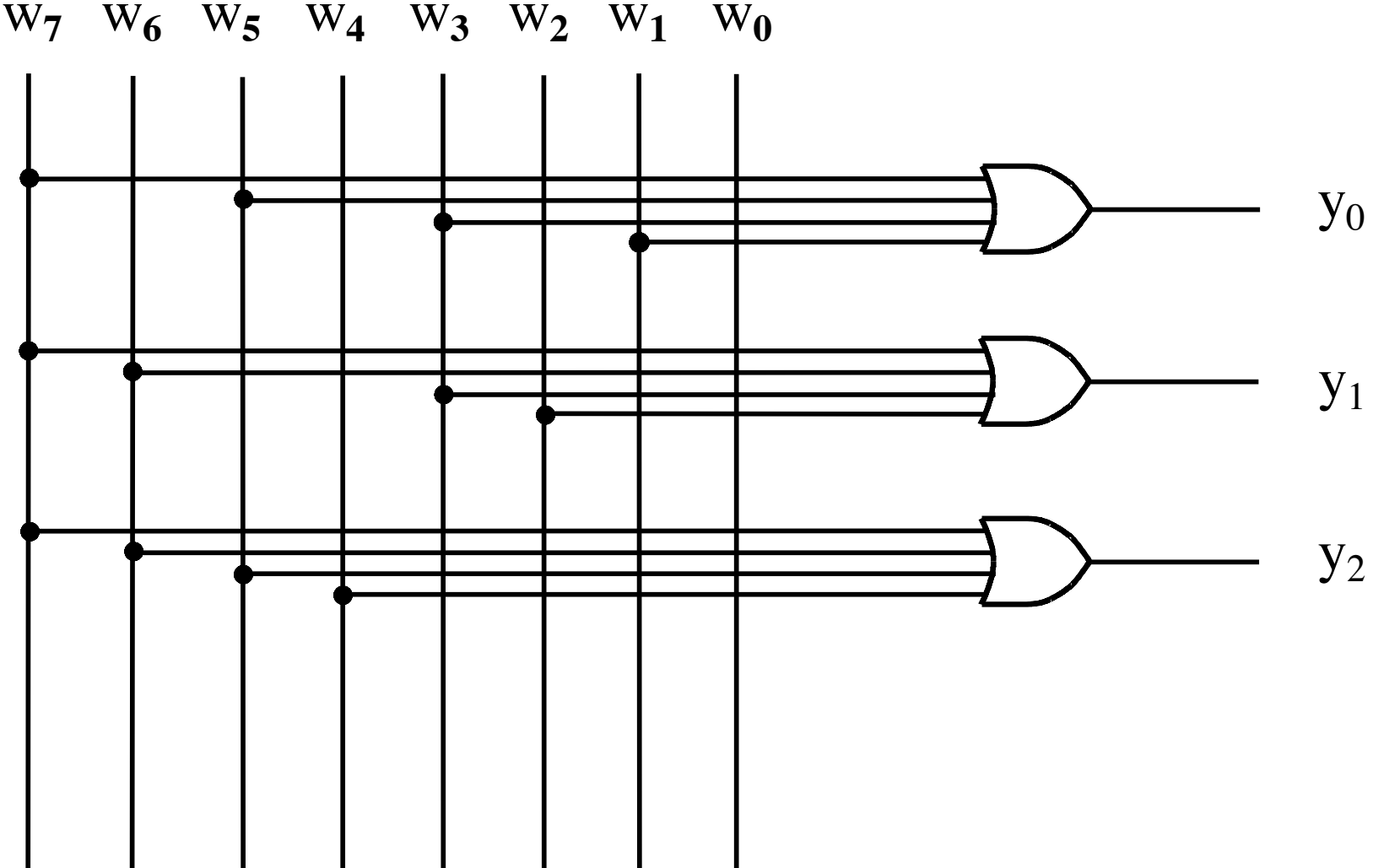
Example 3: Implement an 8-to-3 priority encoder

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

	w_7	w_6	w_5	w_4	w_3	w_2	w_1	w_0	y_2	y_1	y_0	z
$i_0 = \overline{w_7} \overline{w_6} \overline{w_5} \overline{w_4} \overline{w_3} \overline{w_2} \overline{w_1} w_0$	0	0	0	0	0	0	0	1	0	0	0	1
$i_1 = \overline{w_7} \overline{w_6} \overline{w_5} \overline{w_4} \overline{w_3} \overline{w_2} w_1$	0	0	0	0	0	0	1	X	0	0	1	1
$i_2 = \overline{w_7} \overline{w_6} \overline{w_5} \overline{w_4} \overline{w_3} w_2$	0	0	0	0	0	1	X	X	0	1	0	1
$i_3 = \overline{w_7} \overline{w_6} \overline{w_5} \overline{w_4} w_3$	0	0	0	0	1	X	X	X	0	1	1	1
$i_4 = \overline{w_7} \overline{w_6} \overline{w_5} w_4$	0	0	0	1	X	X	X	X	1	0	0	1
$i_5 = \overline{w_7} \overline{w_6} w_5$	0	0	1	X	X	X	X	X	1	0	1	1
$i_6 = \overline{w_7} w_6$	0	1	X	X	X	X	X	X	1	1	0	1
$i_7 = w_7$	1	X	X	X	X	X	X	X	1	1	1	1
$z = i_0 + i_1 + i_2 + i_3 + i_4 + i_5 + i_6 + i_7$	0	0	0	0	0	0	0	0	d	d	d	0

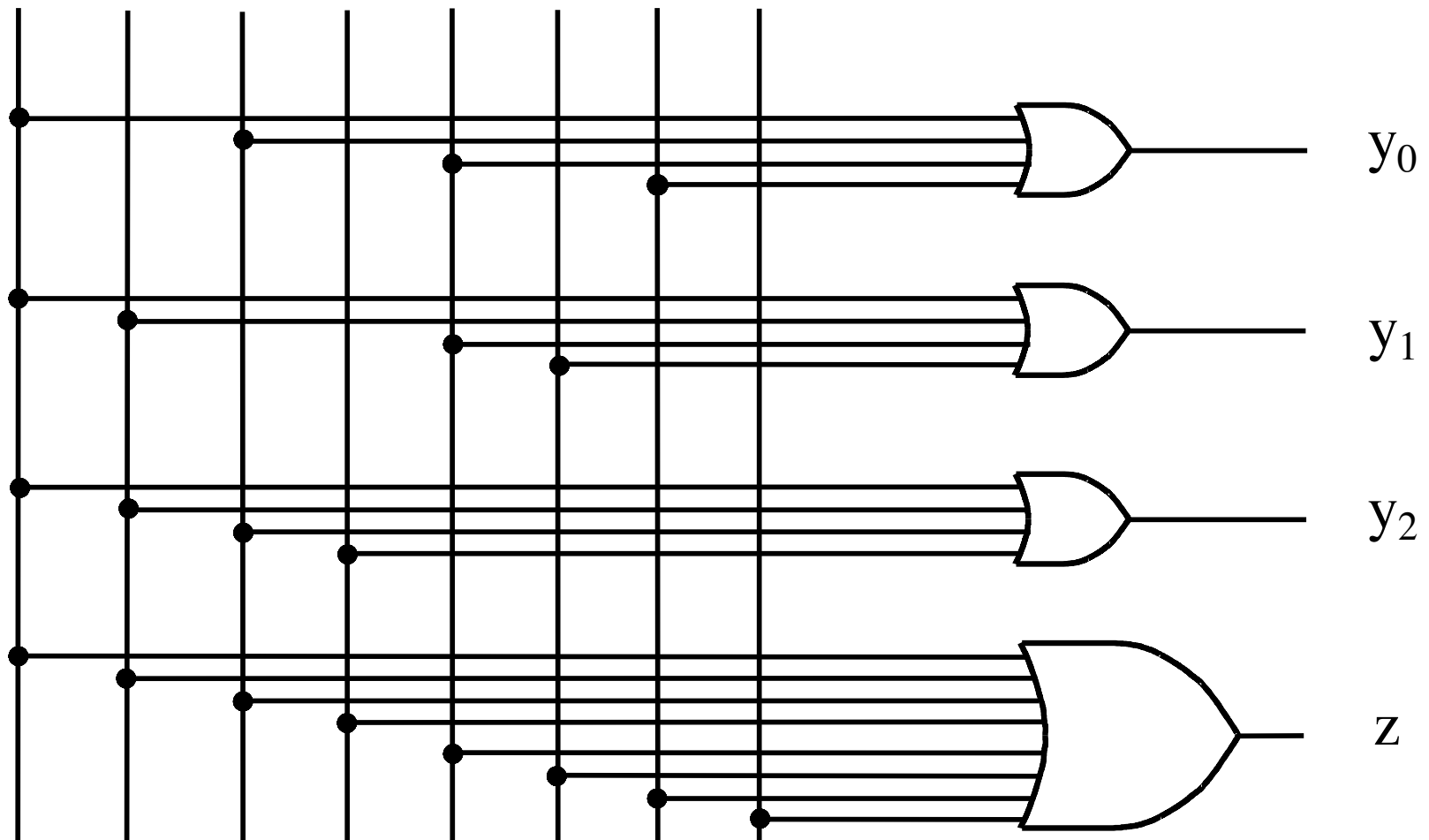
Circuit for the 8-to-3 binary encoder



Circuit for the 8-to-3 priority encoder

i_7 i_6 i_5 i_4 i_3 i_2 i_1 i_0

w_7 w_6 w_5 w_4 w_3 w_2 w_1 w_0



Example 4: Circuit implementation using a multiplexer

Implement the function

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

using a 4-to-1 multiplexer

Some Boolean Algebra Leads To

$$\overline{w_1} \overline{w_2} \overline{w_4} \overline{w_5} + w_1 w_2 + w_1 w_3 + w_1 w_4 + w_3 w_4 w_5$$

$$\overline{w_1} \overline{w_4} (\overline{w_5} \overline{w_2}) + w_4 (w_3 w_5) + w_1 (w_2 + w_3) + w_1 w_4 (1)$$

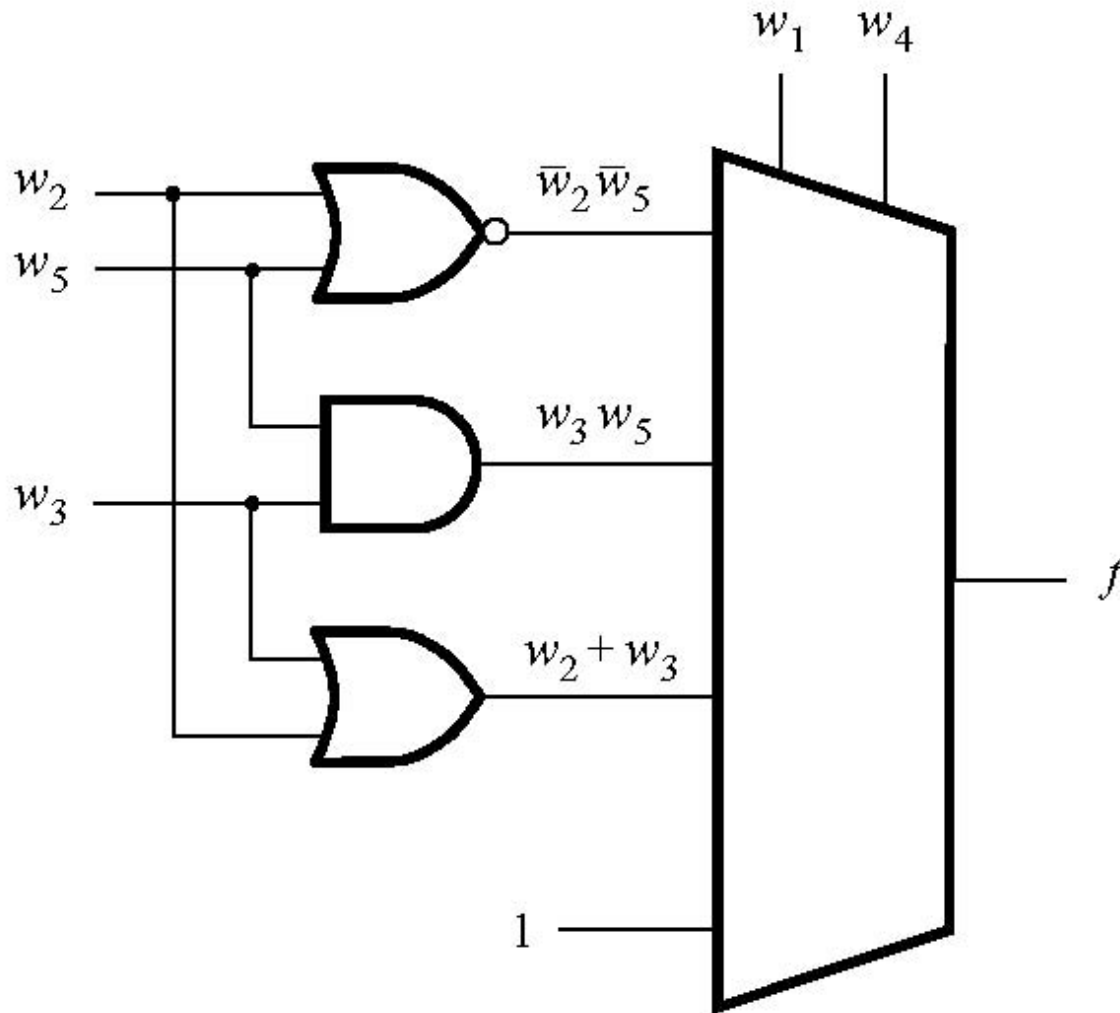
$$\overline{w_1} \overline{w_4} (\overline{w_5} \overline{w_2}) + (\overline{w_1} + w_1) w_4 (w_3 w_5) + w_1 (\overline{w_4} + w_4) (w_2 + w_3) + w_1 w_4 (1)$$

$$\overline{w_1} \overline{w_4} (\overline{w_5} \overline{w_2}) + \overline{w_1} w_4 (w_3 w_5) + w_1 \overline{w_4} (w_2 + w_3) + w_1 w_4 (w_3 w_5 + (w_2 + w_3) + 1)$$

$$\overline{w_1} \overline{w_4} (\overline{w_5} \overline{w_2}) + \overline{w_1} w_4 (w_3 w_5) + w_1 \overline{w_4} (w_2 + w_3) + w_1 w_4 (1)$$

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

Solution Circuit



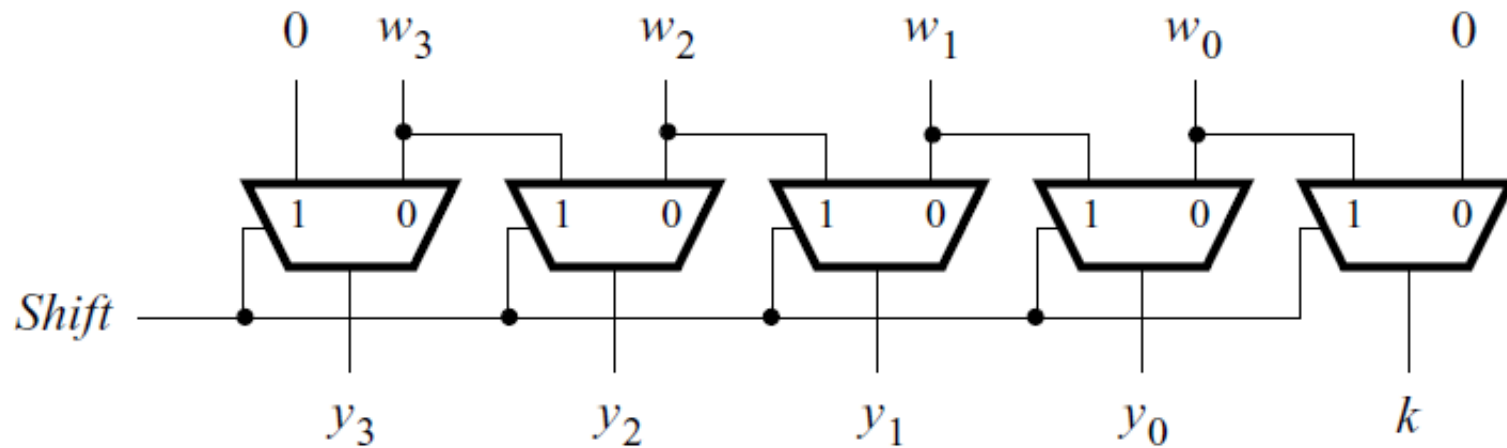
$$\bar{w}_1 \bar{w}_4 (\bar{w}_5 \bar{w}_2) + \bar{w}_1 w_4 (w_3 w_5) + w_1 \bar{w}_4 (w_2 + w_3) + w_1 w_4 (1)$$

[Figure 4.46 from the textbook]

Some Final Things from Chapter 4

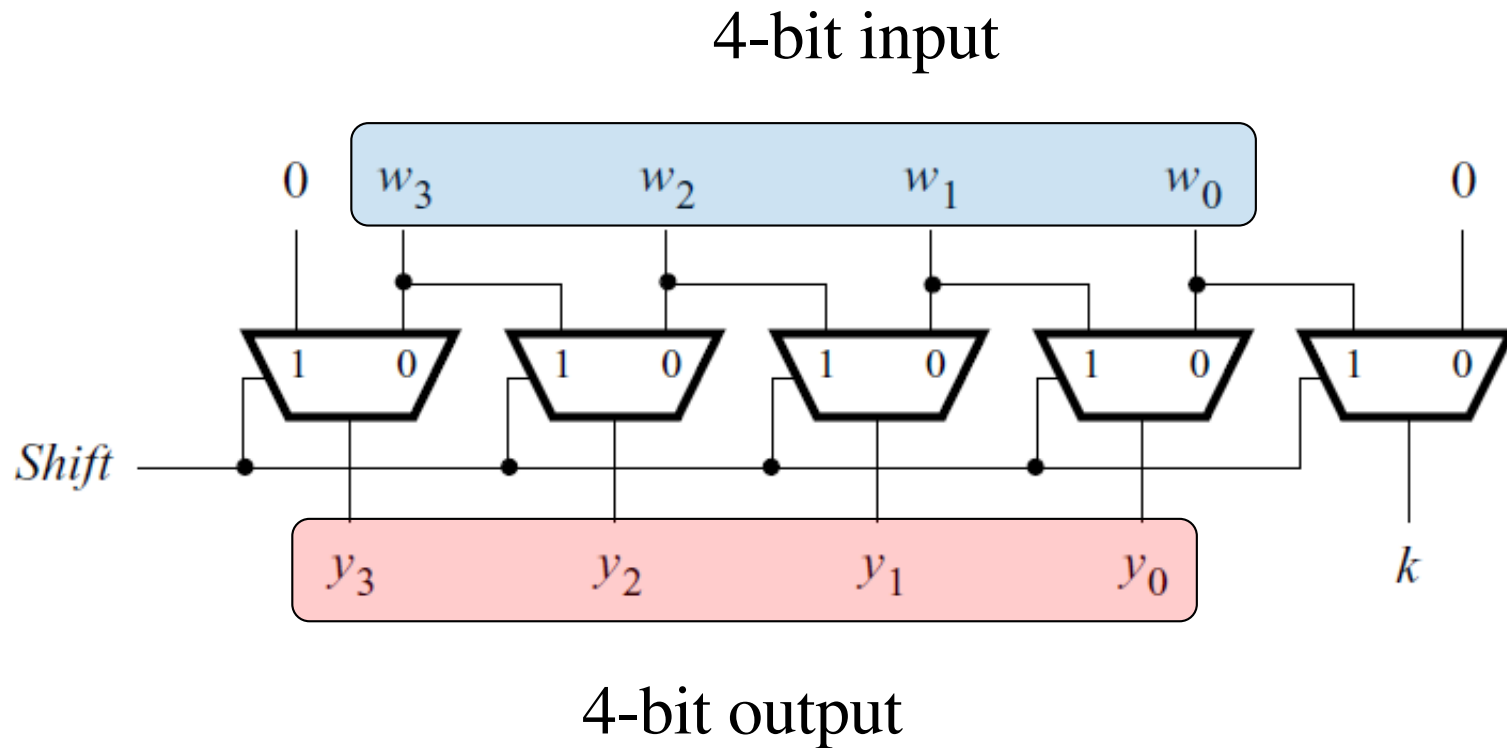
Shifter Circuit

Hold / Shift-Right Circuit

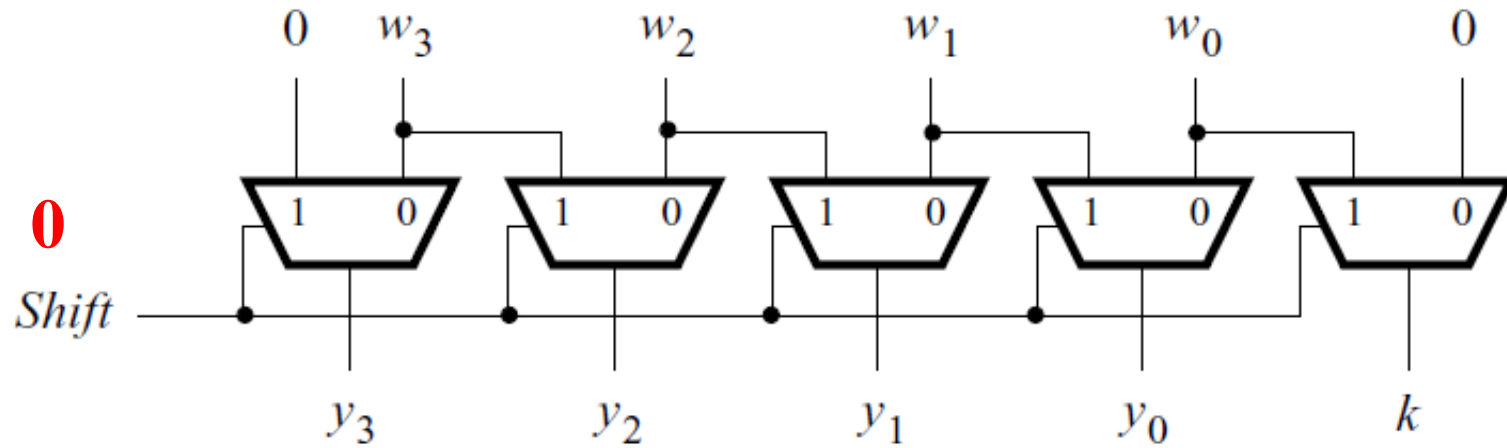


[Figure 4.50 from the textbook]

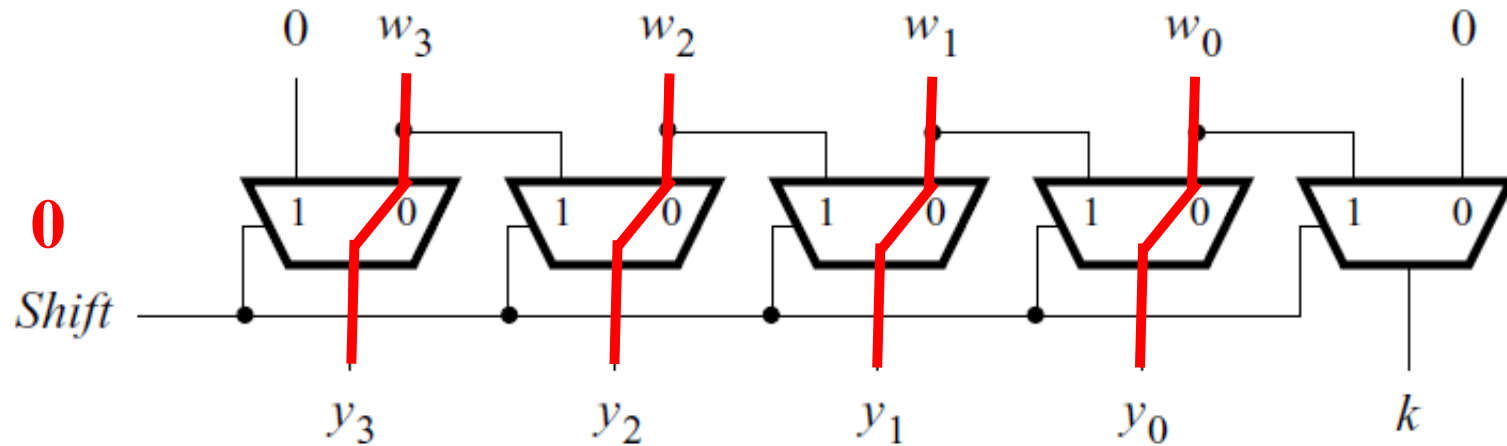
Hold / Shift-Right Circuit



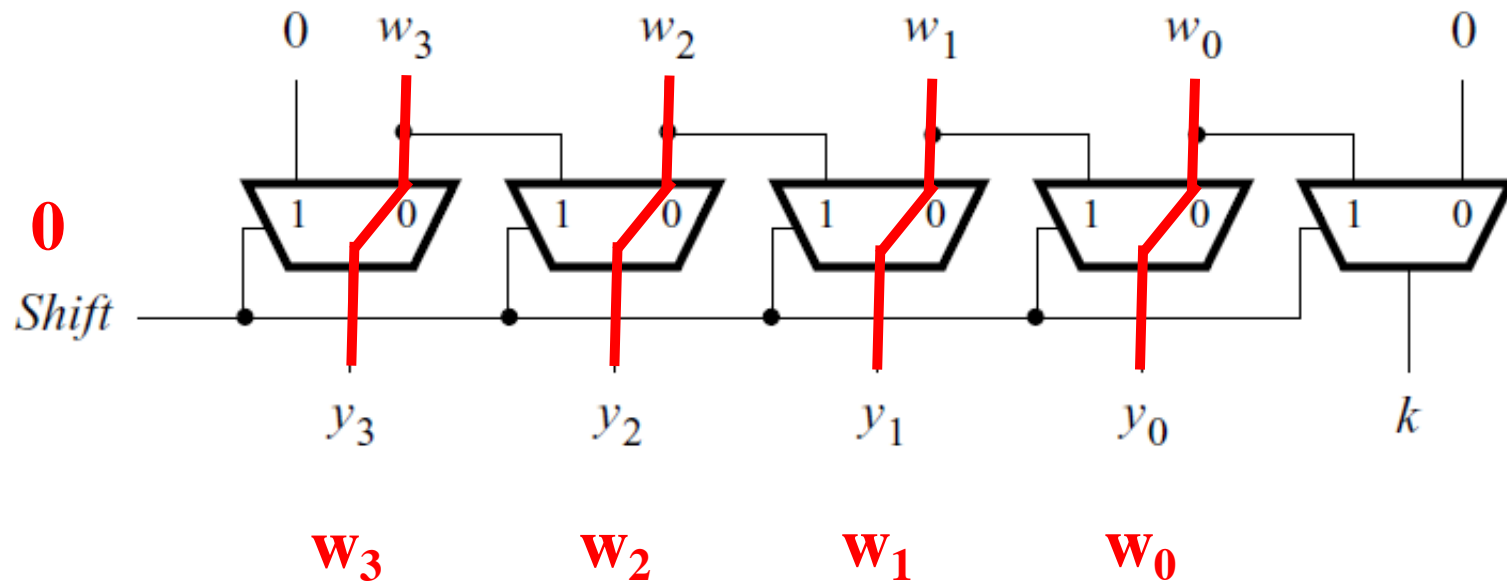
No shift if the control signal is 0



No shift if the control signal is 0

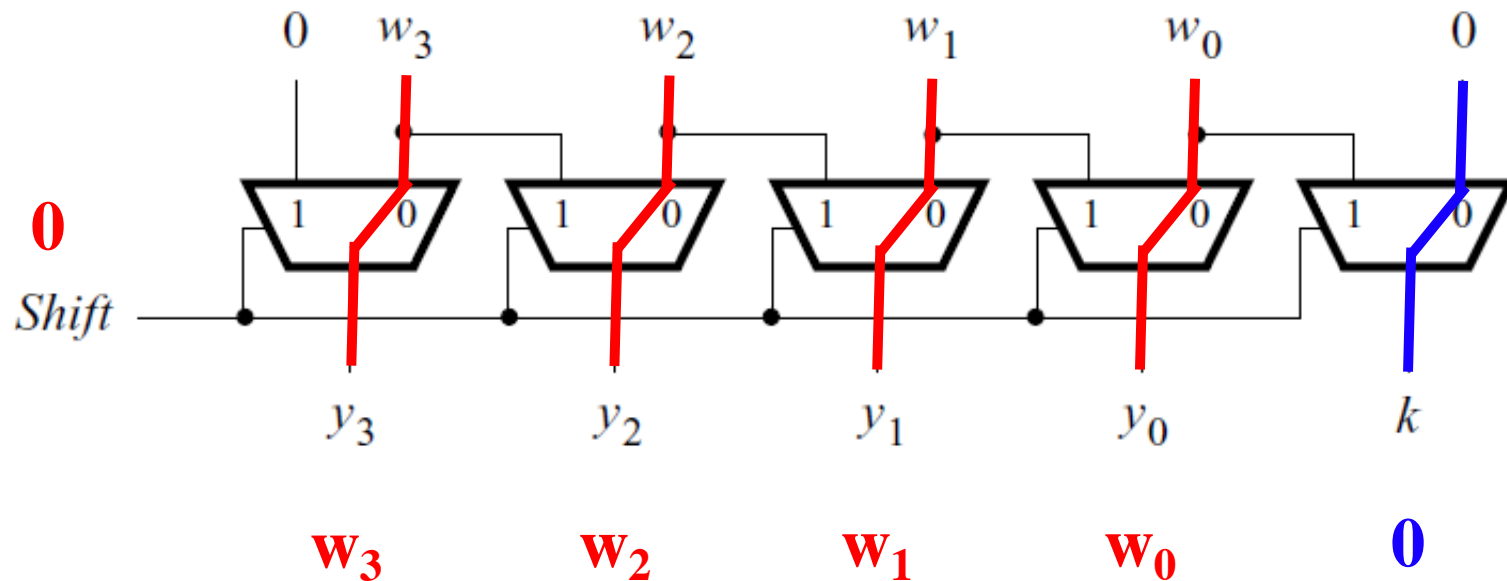


No shift if the control signal is 0



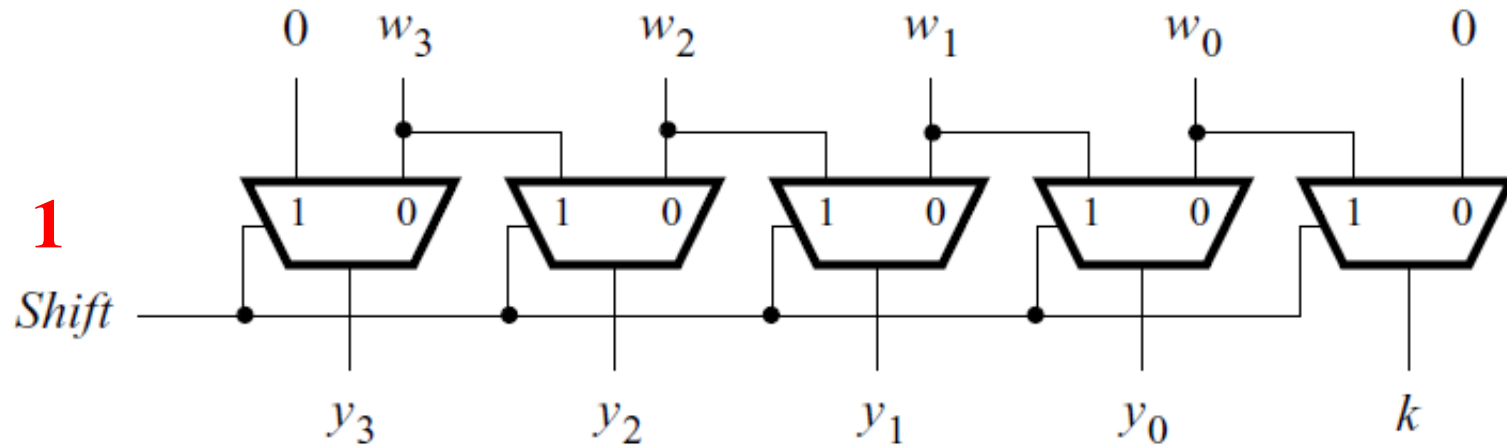
No shift in this case.

No shift if the control signal is 0

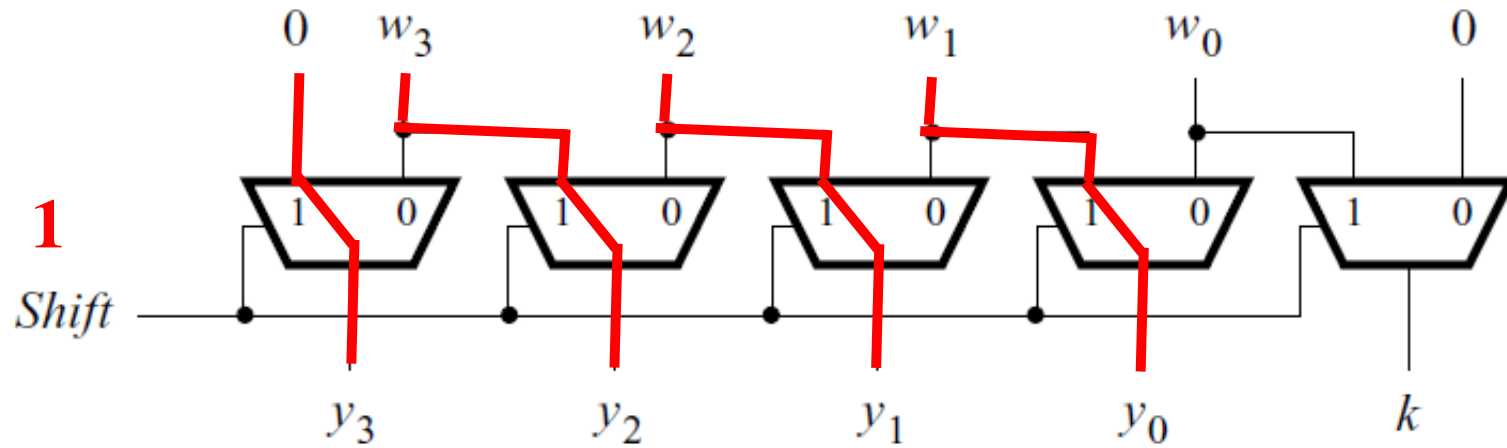


No shift in this case.

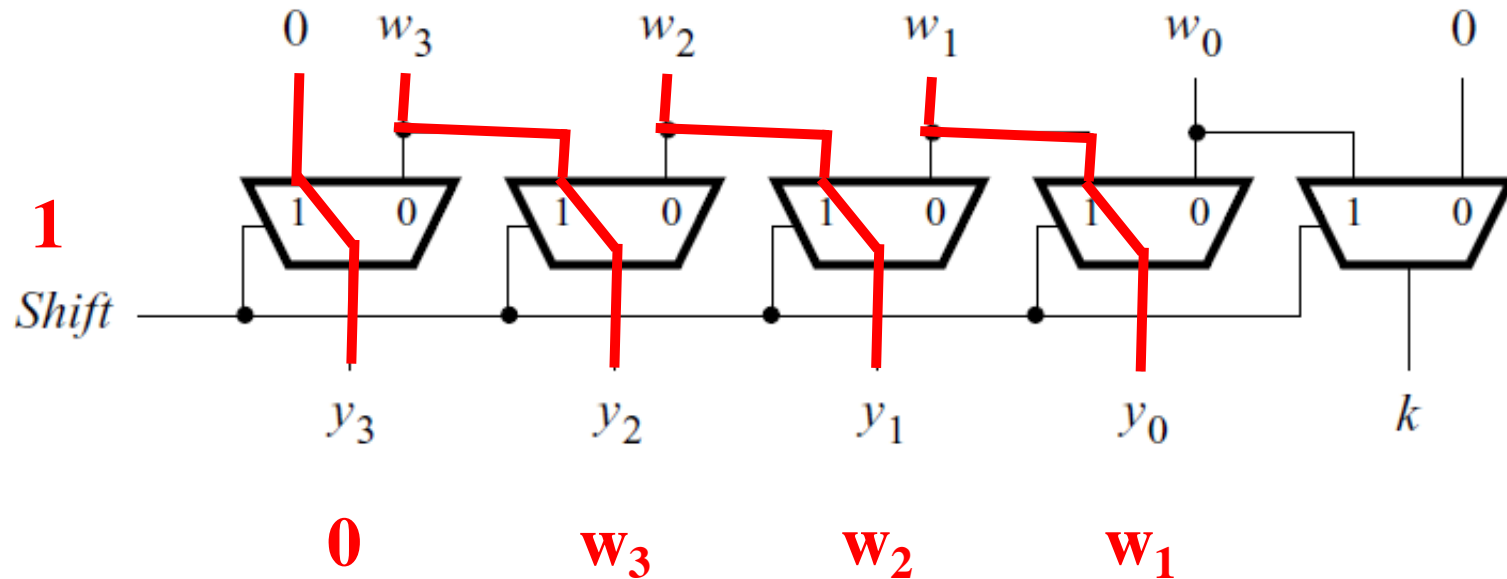
Shift right if the control signal is 1



Shift right if the control signal is 1

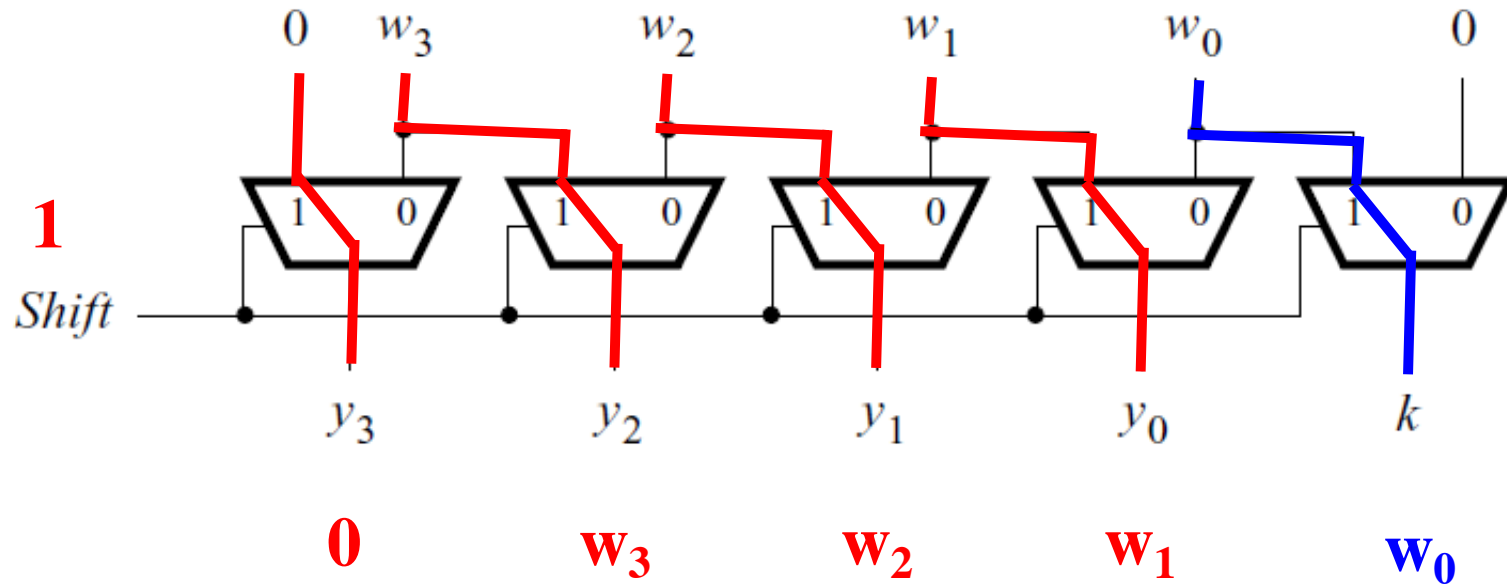


Shift right if the control signal is 1



Shift to the right by 1 bit

A shifter circuit



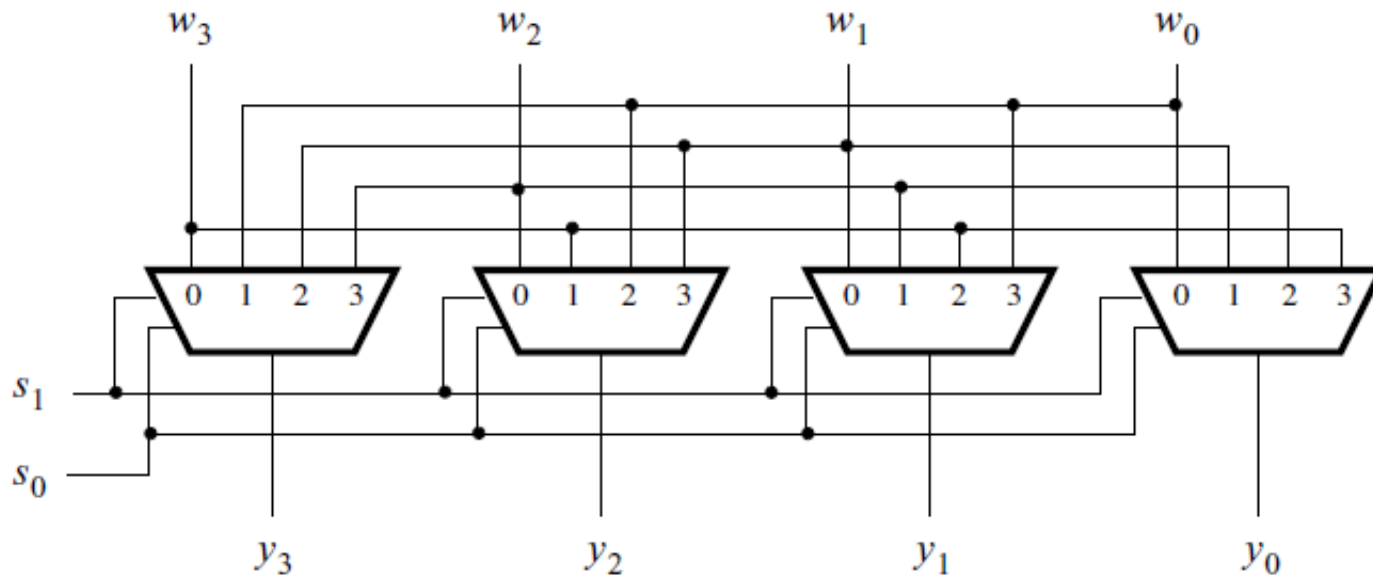
Shift to the right by 1 bit

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

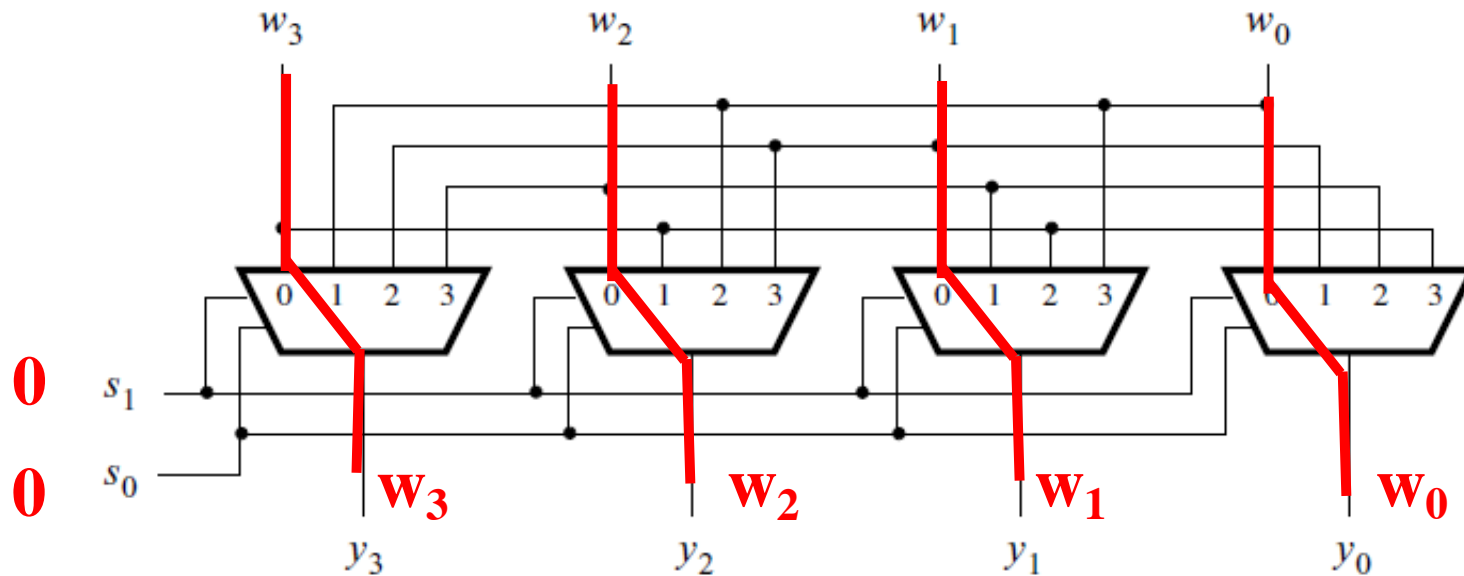


(b) Circuit

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

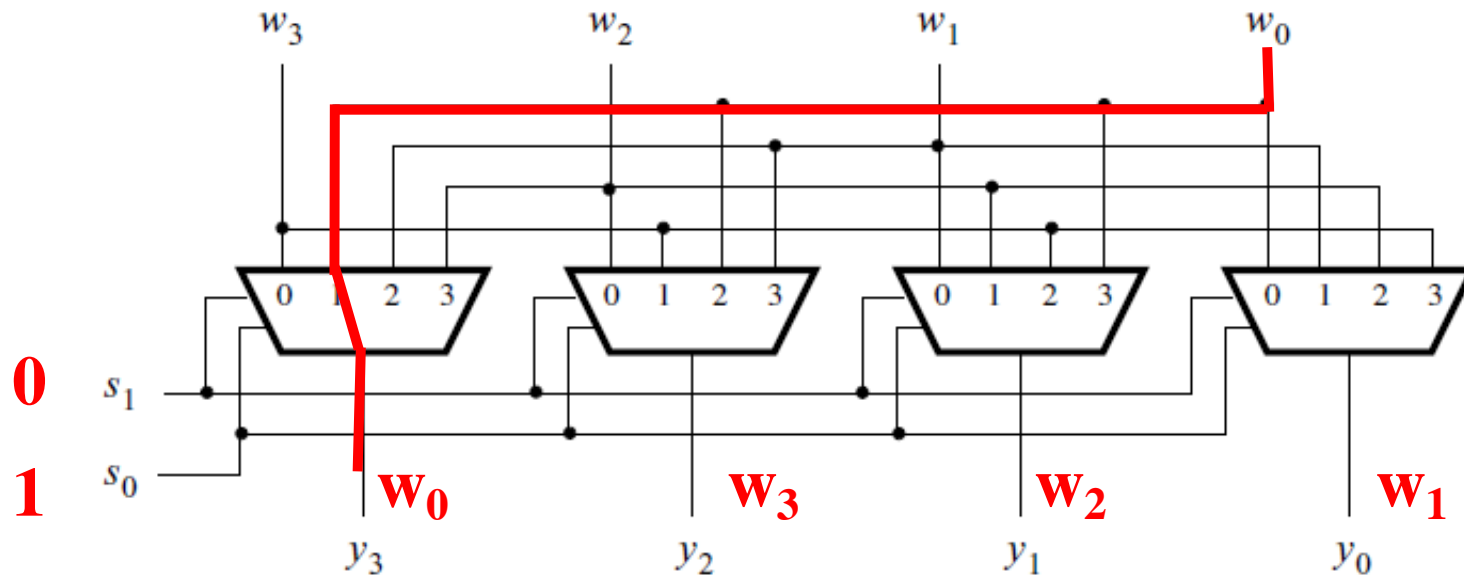


(b) Circuit

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

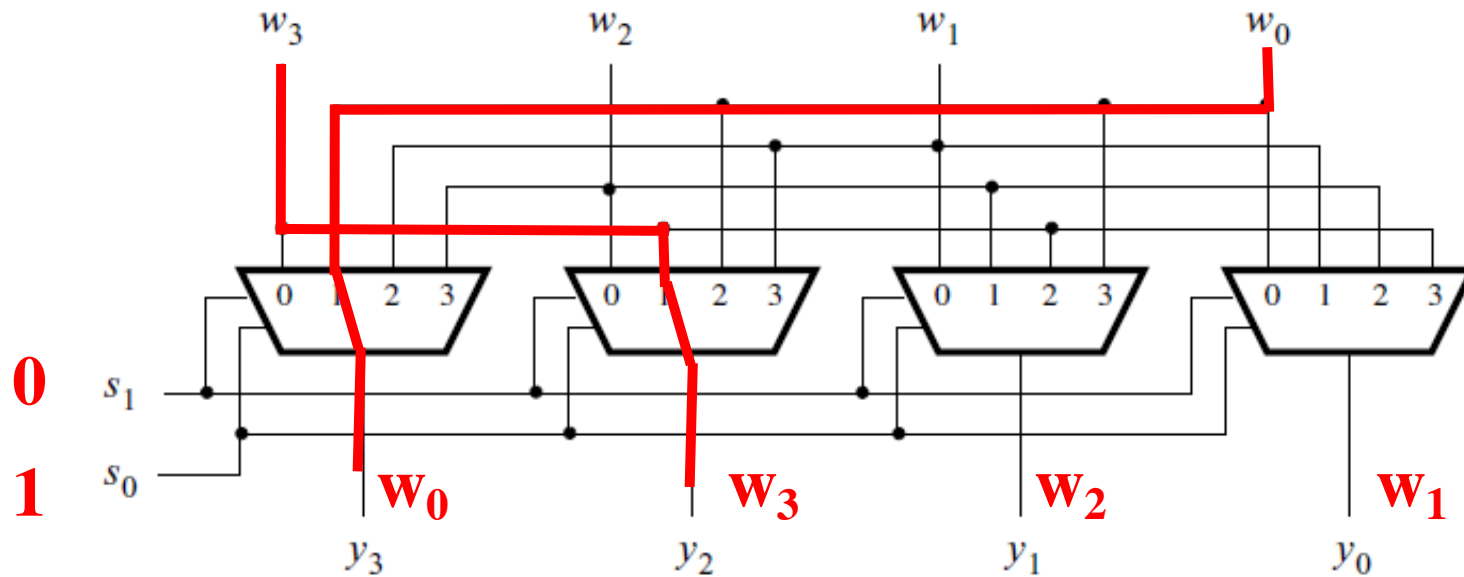


(b) Circuit

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

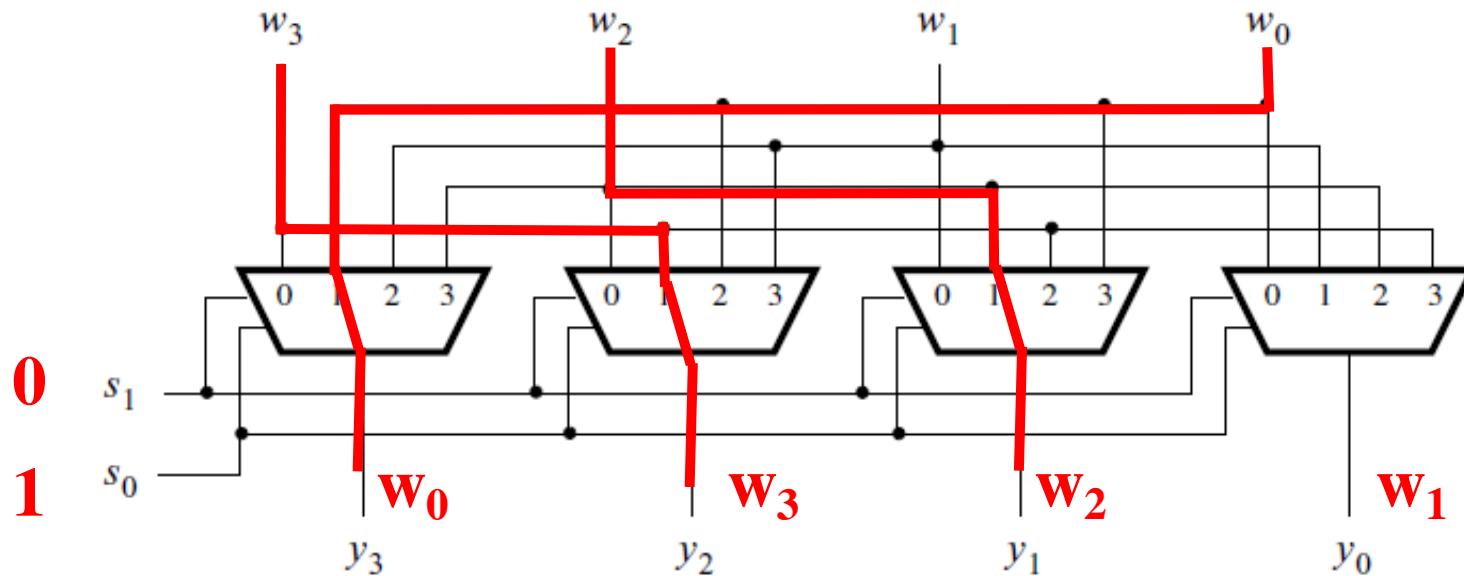


(b) Circuit

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

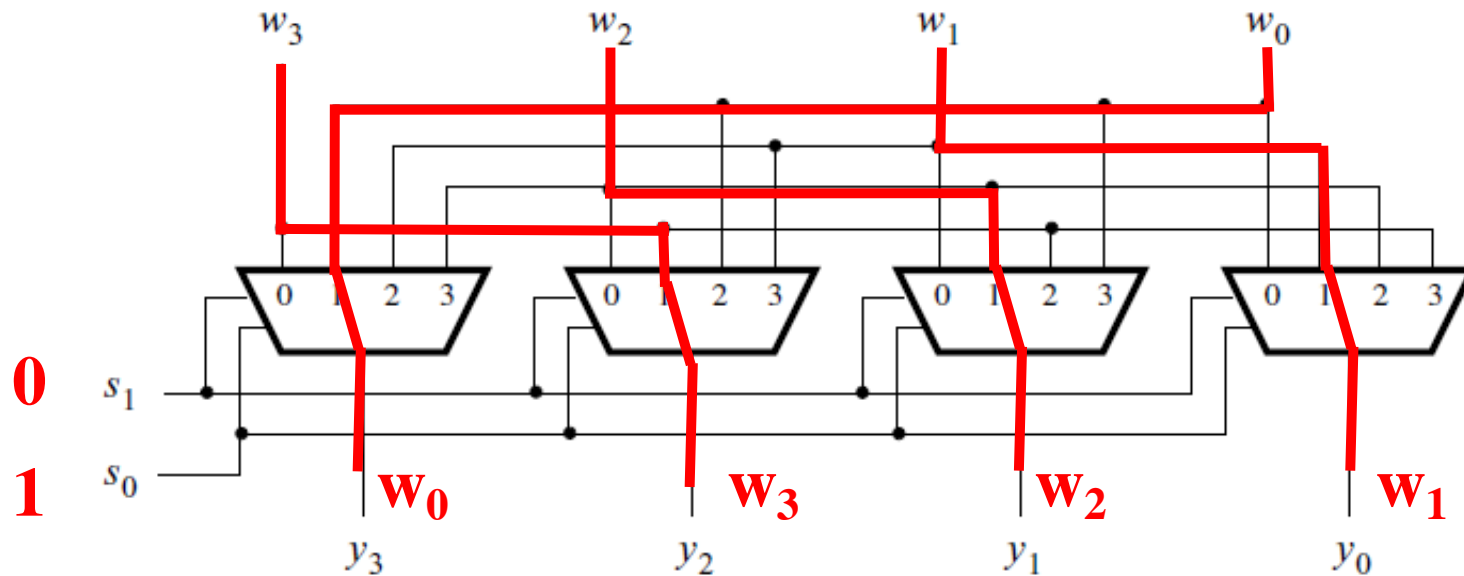


(b) Circuit

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

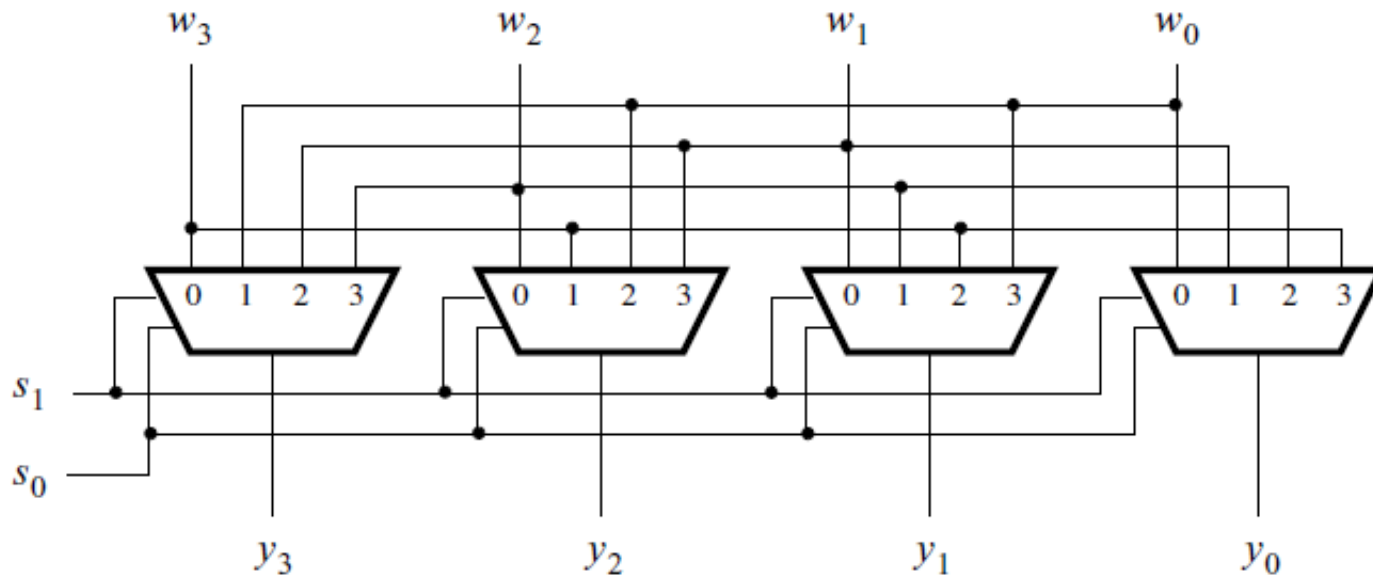


(b) Circuit

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

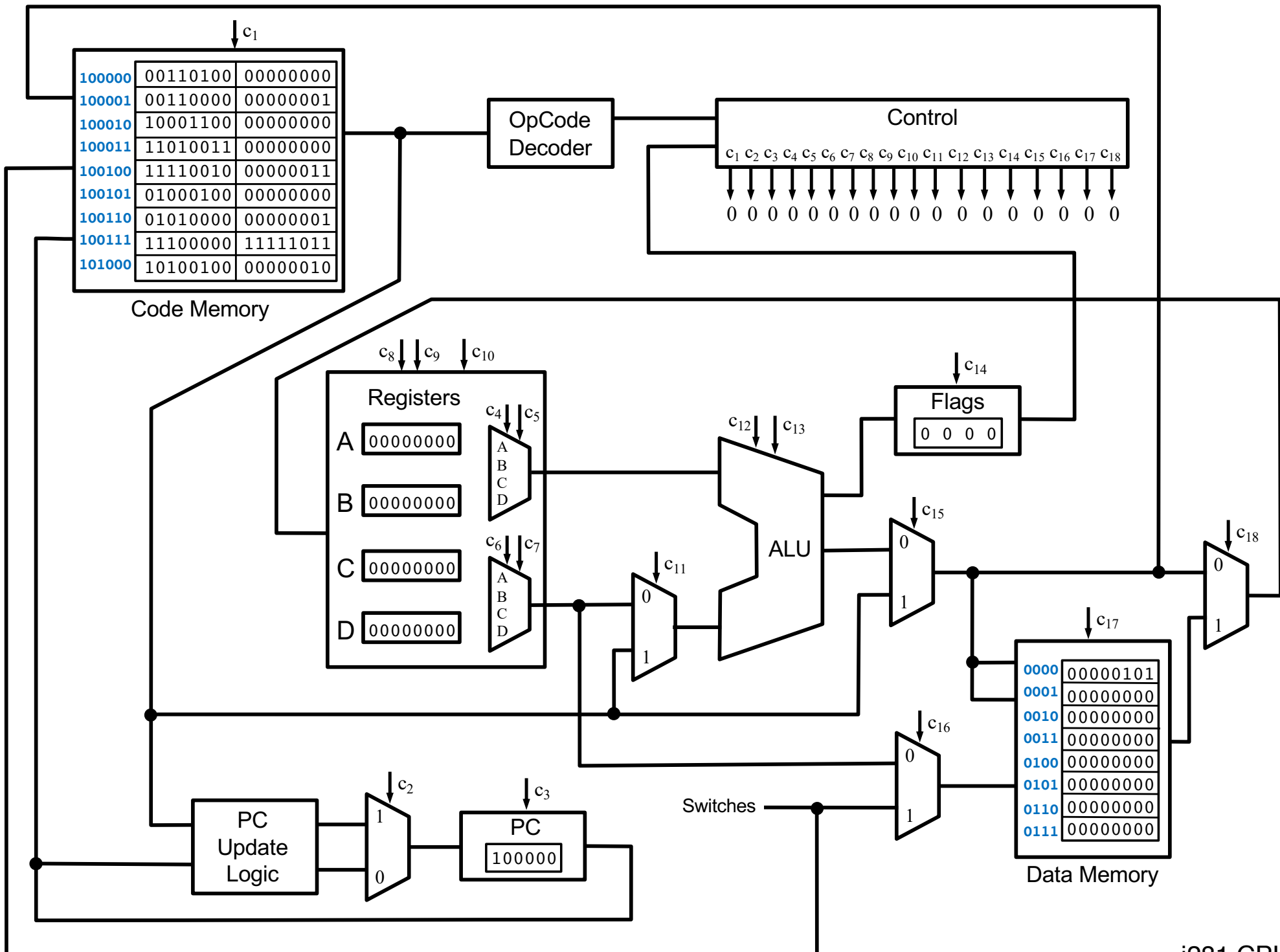
(a) Truth table

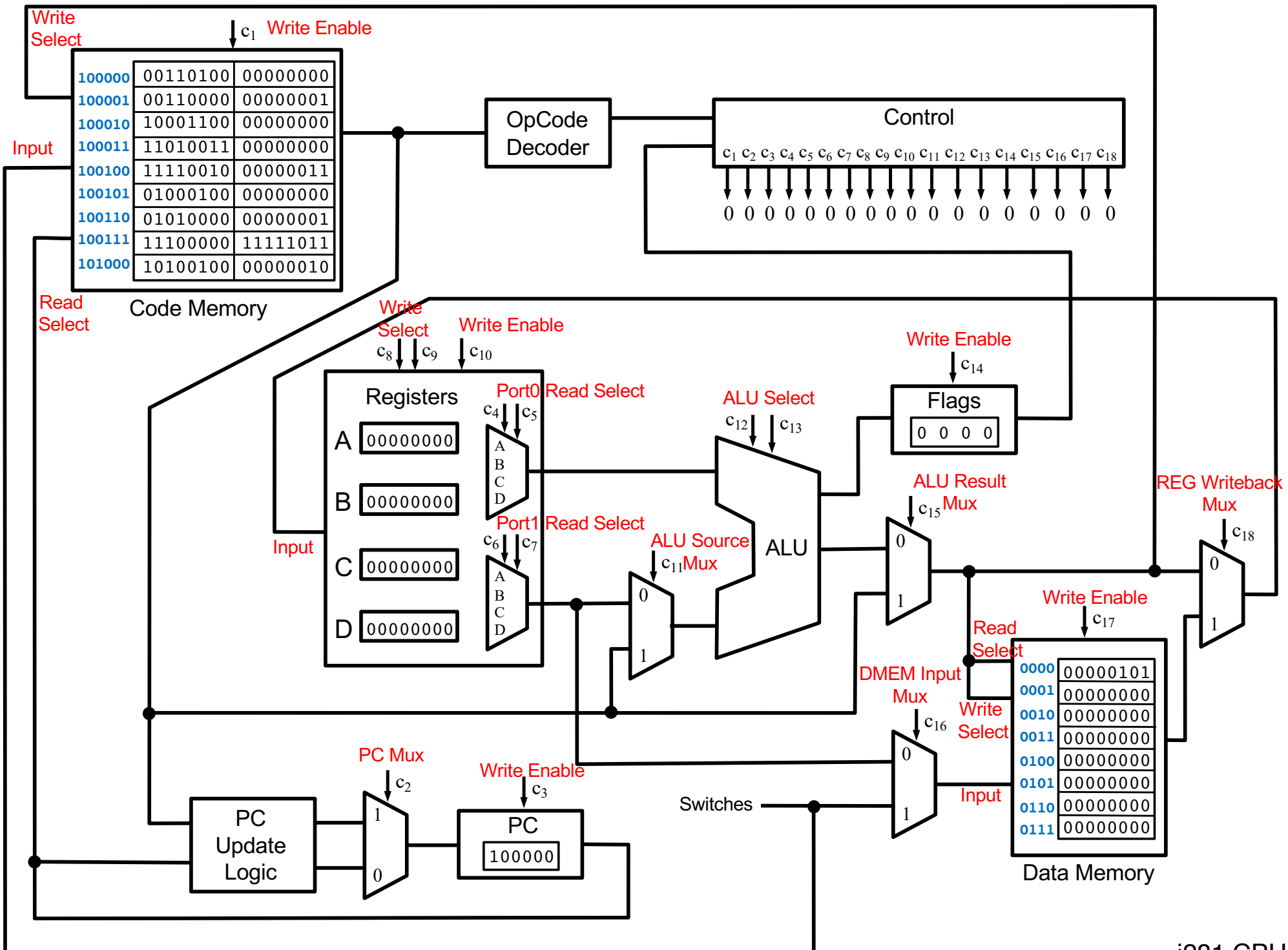


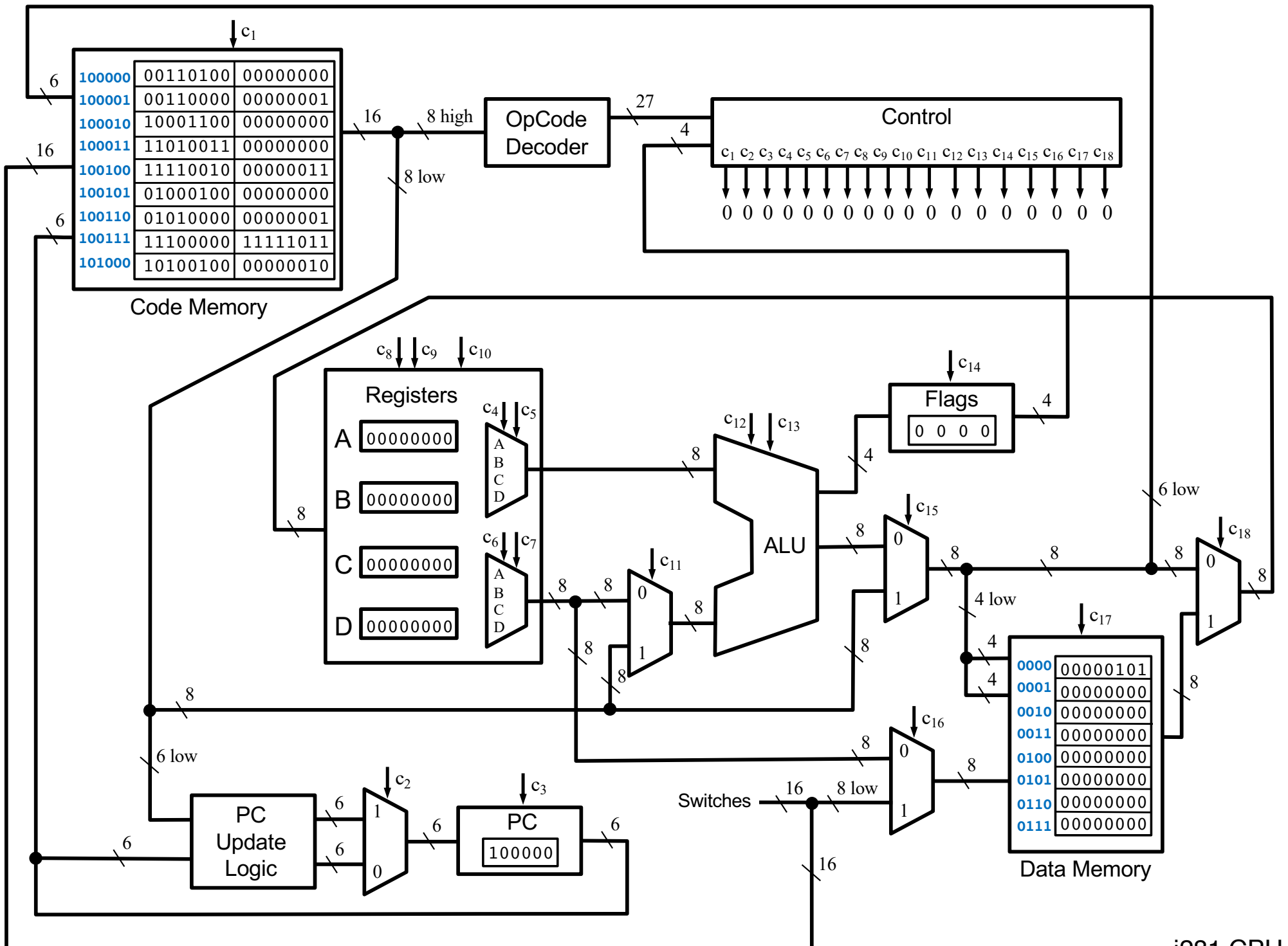
(b) Circuit

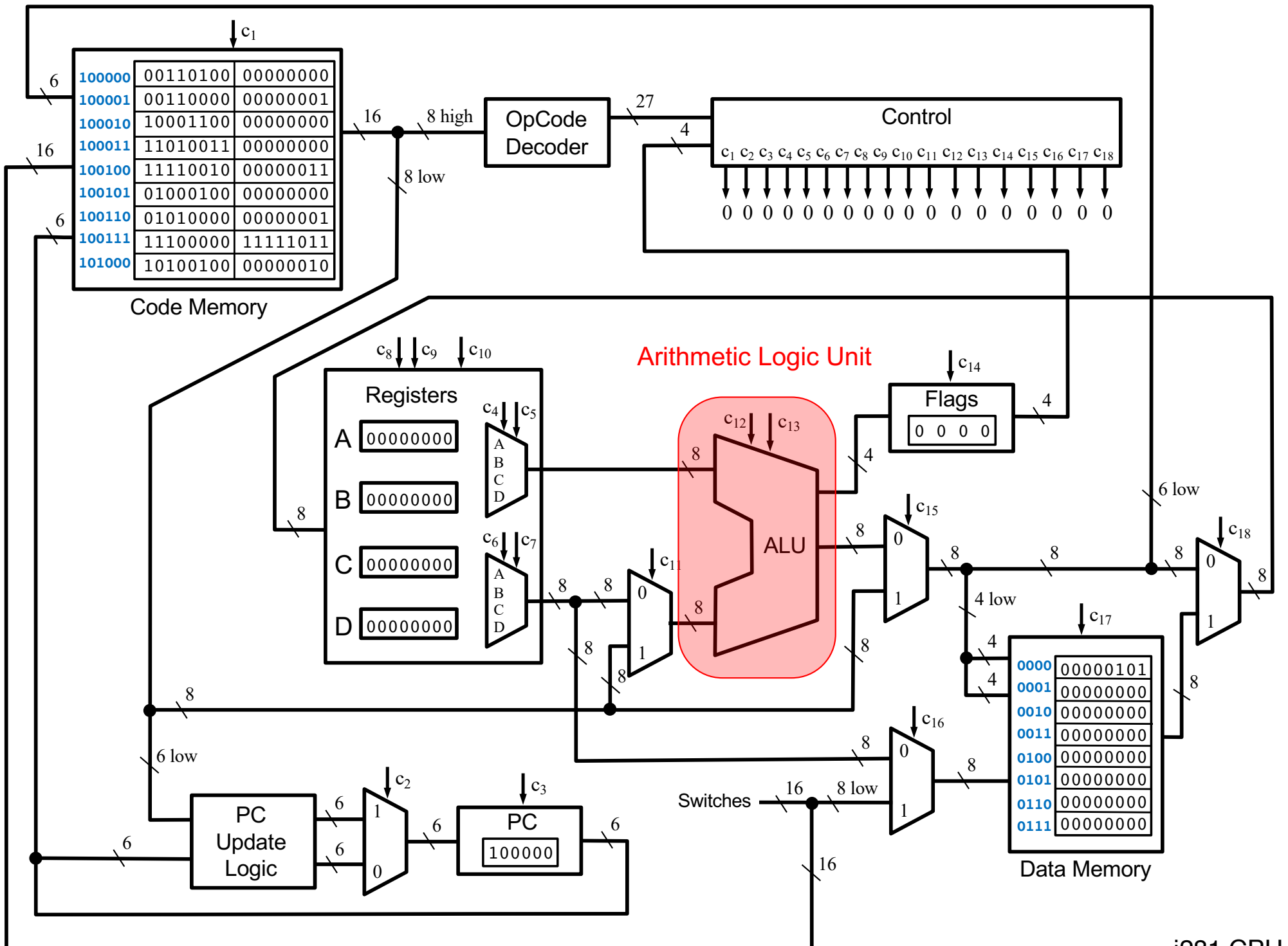
Arithmetic Logic Unit (ALU)

Shifter Circuit of the i281 CPU

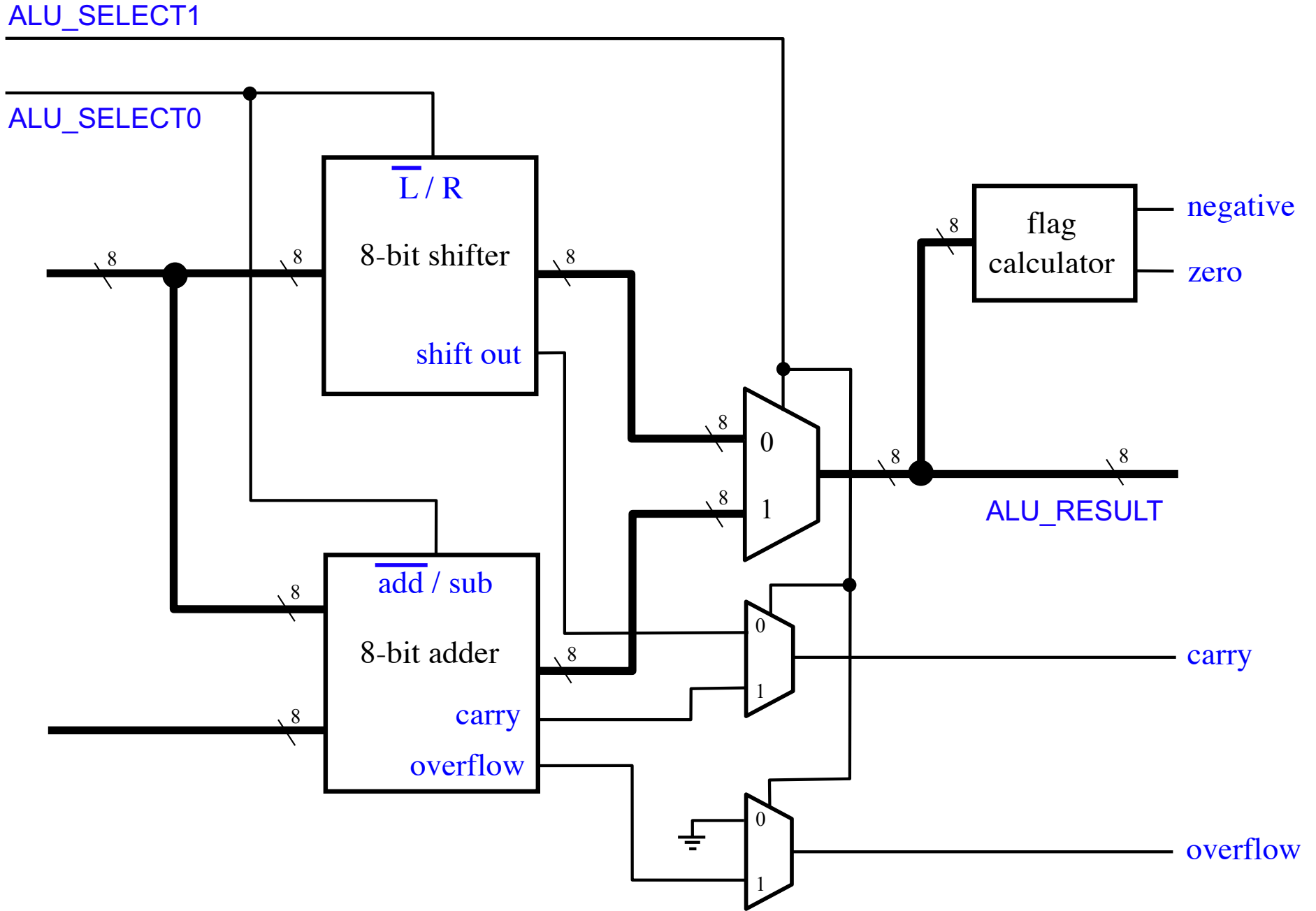




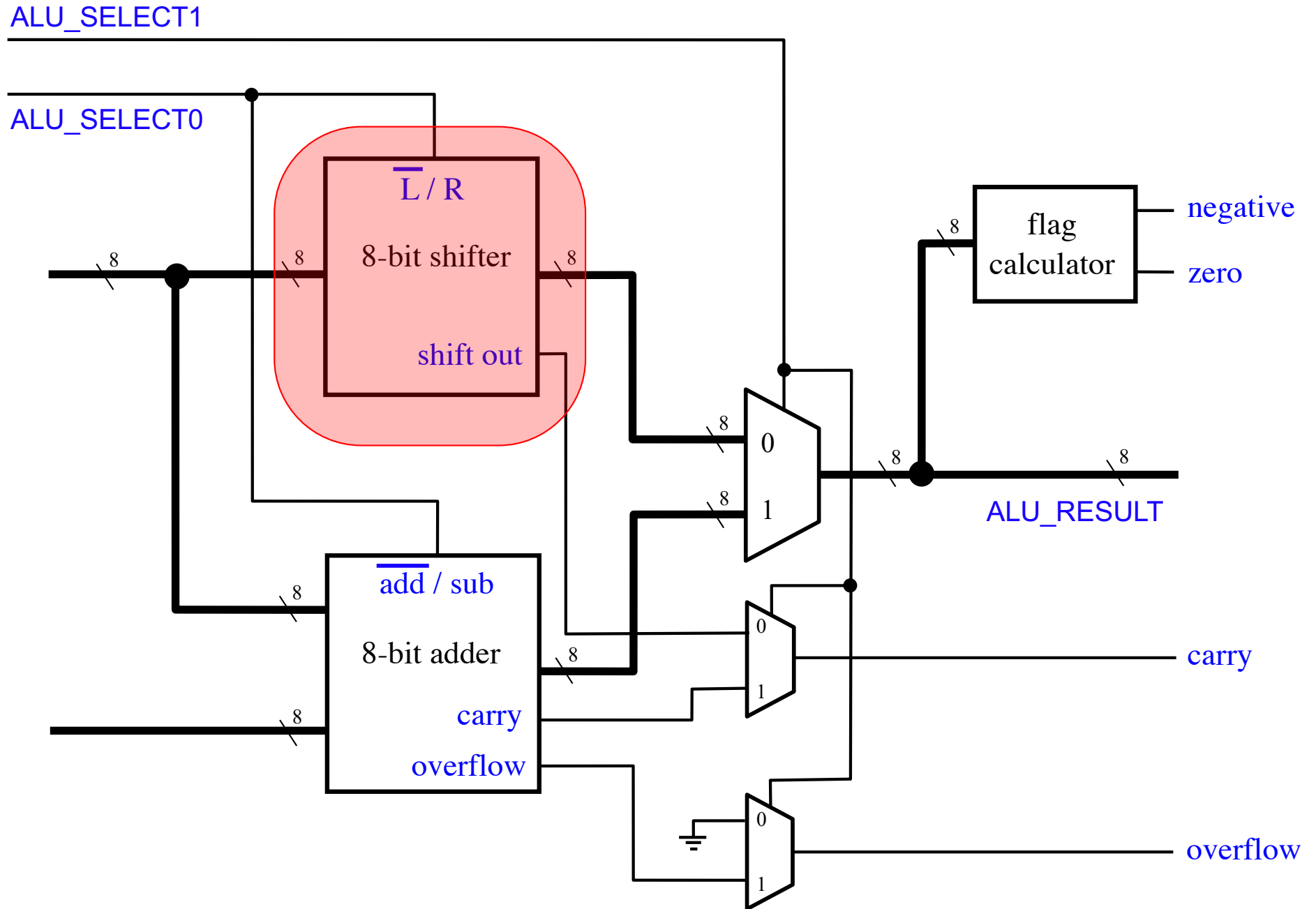




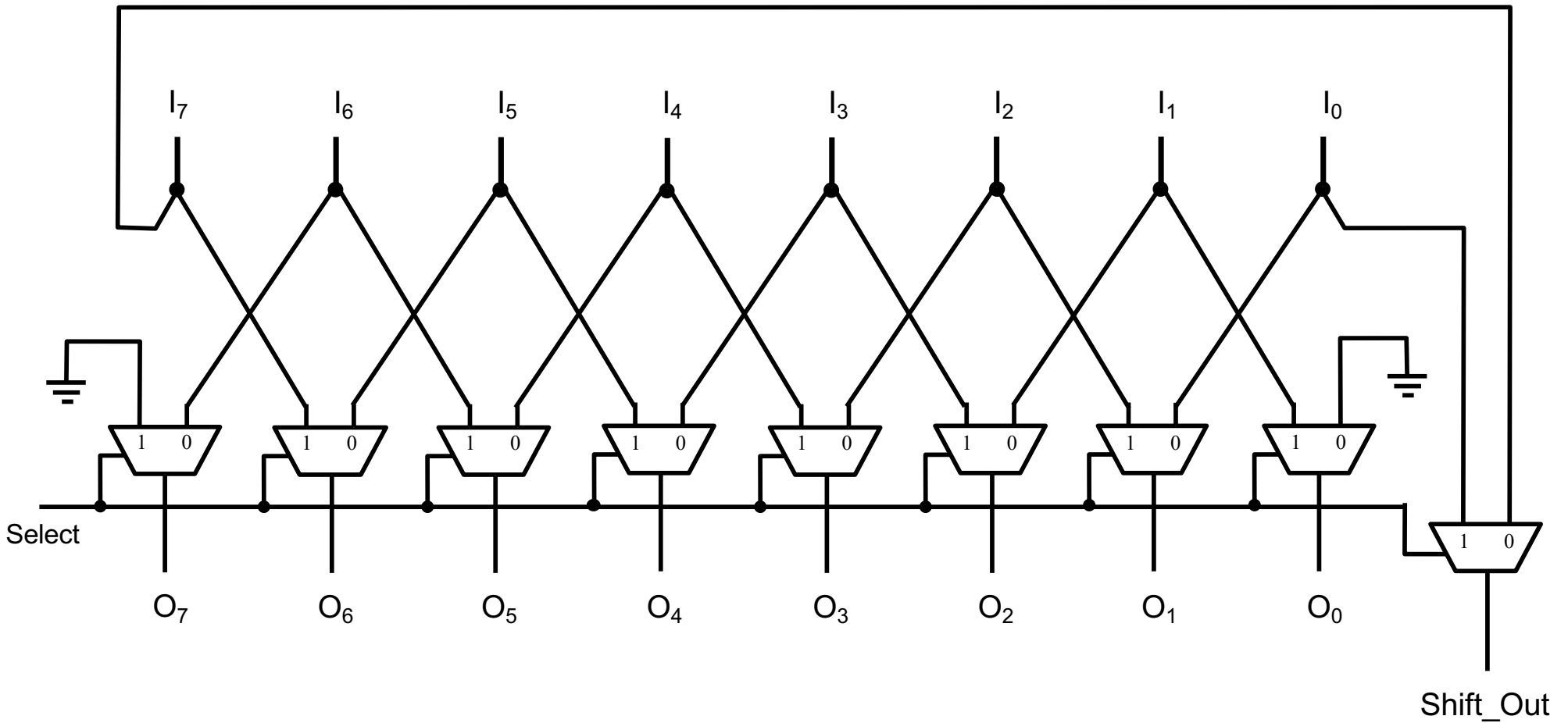
The ALU



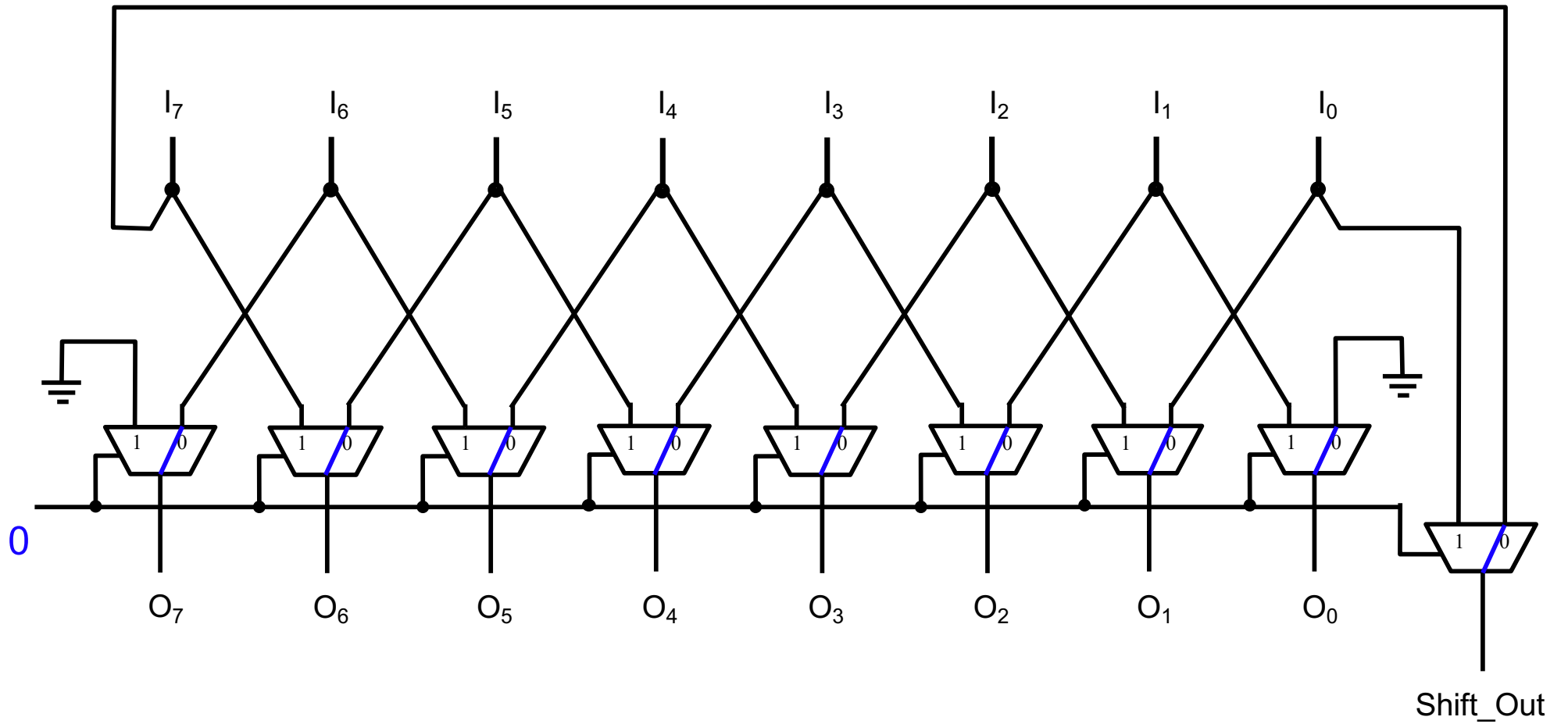
The Shifter Circuit



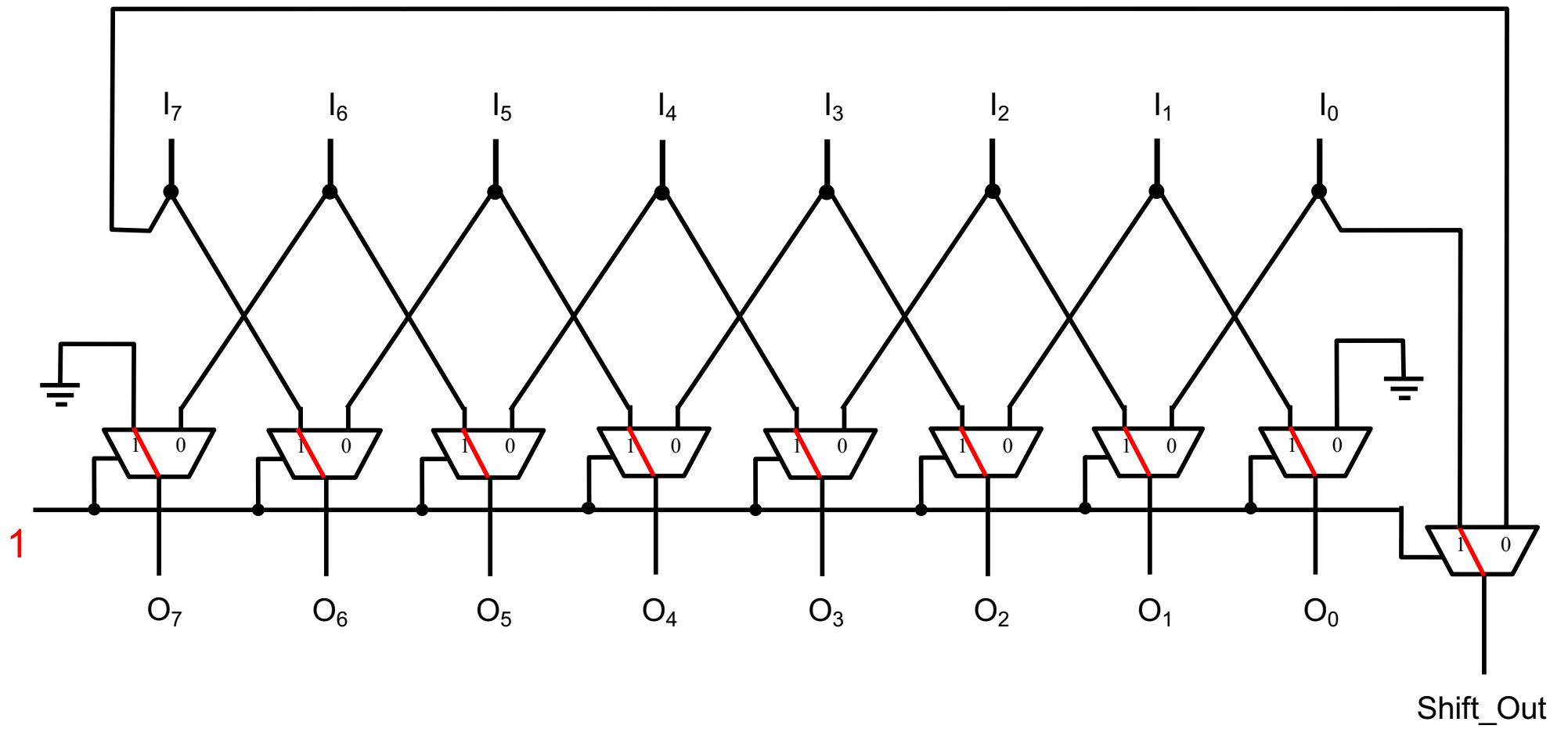
The i281 CPU Shifter Circuit



i281 CPU

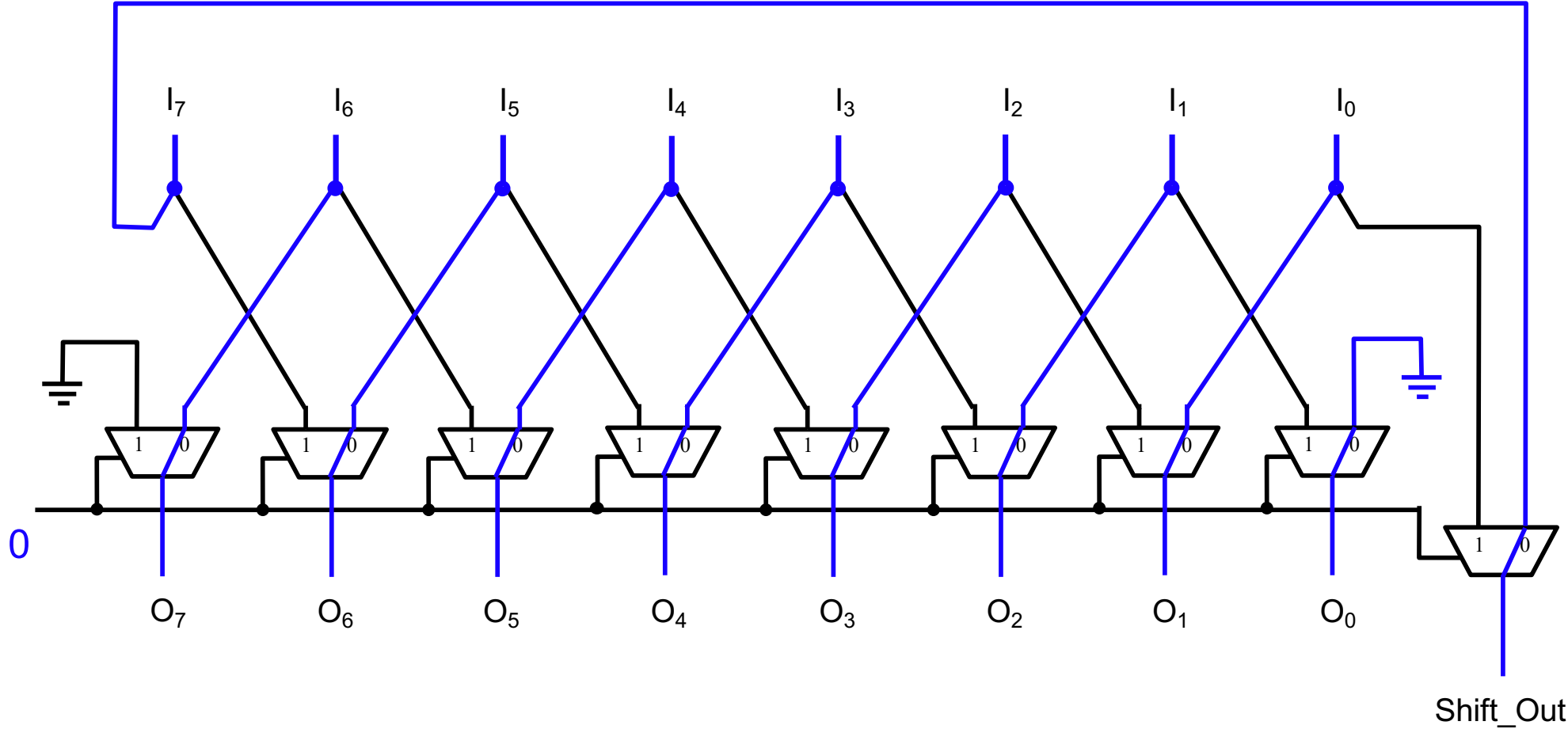


i281 CPU

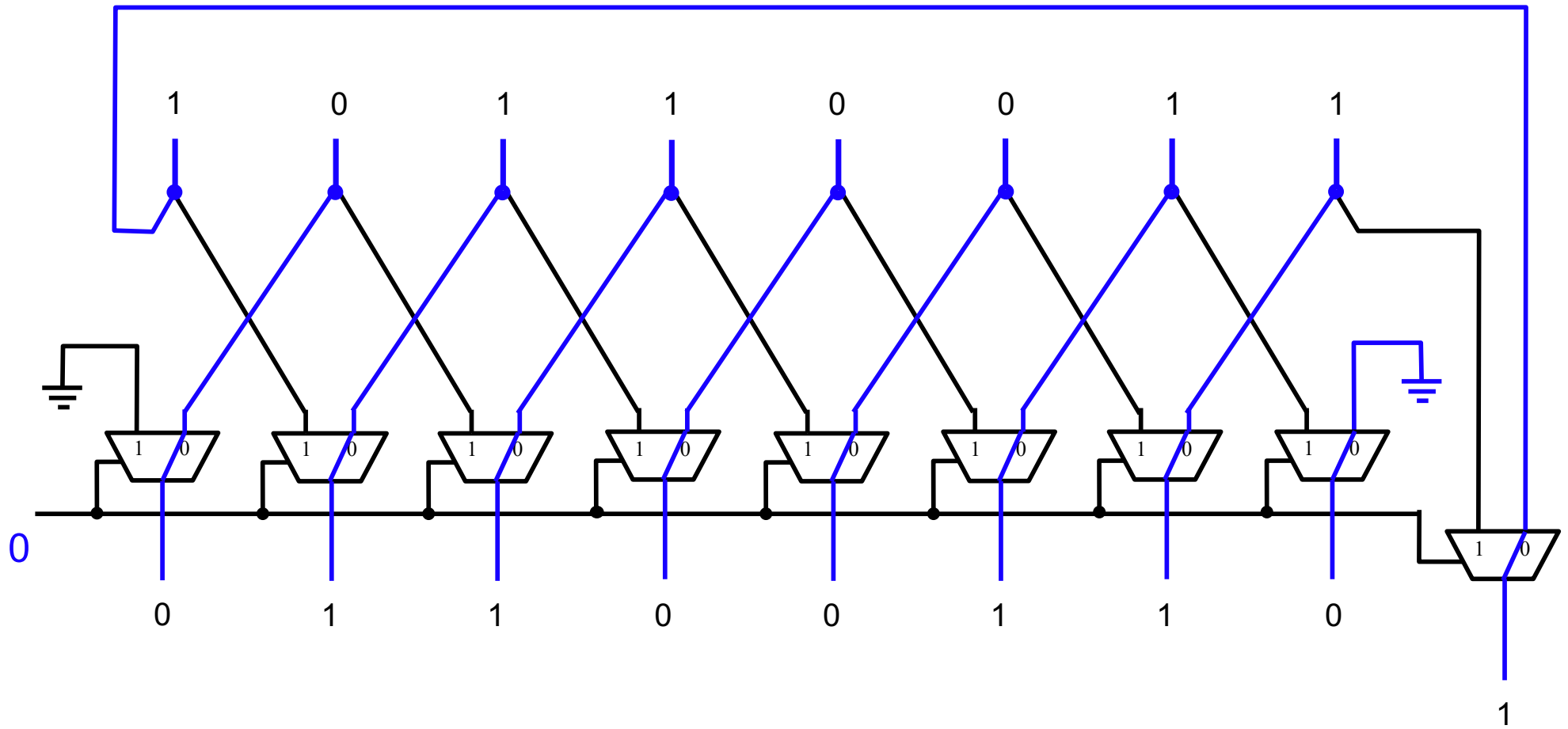


i281 CPU

Shift Left

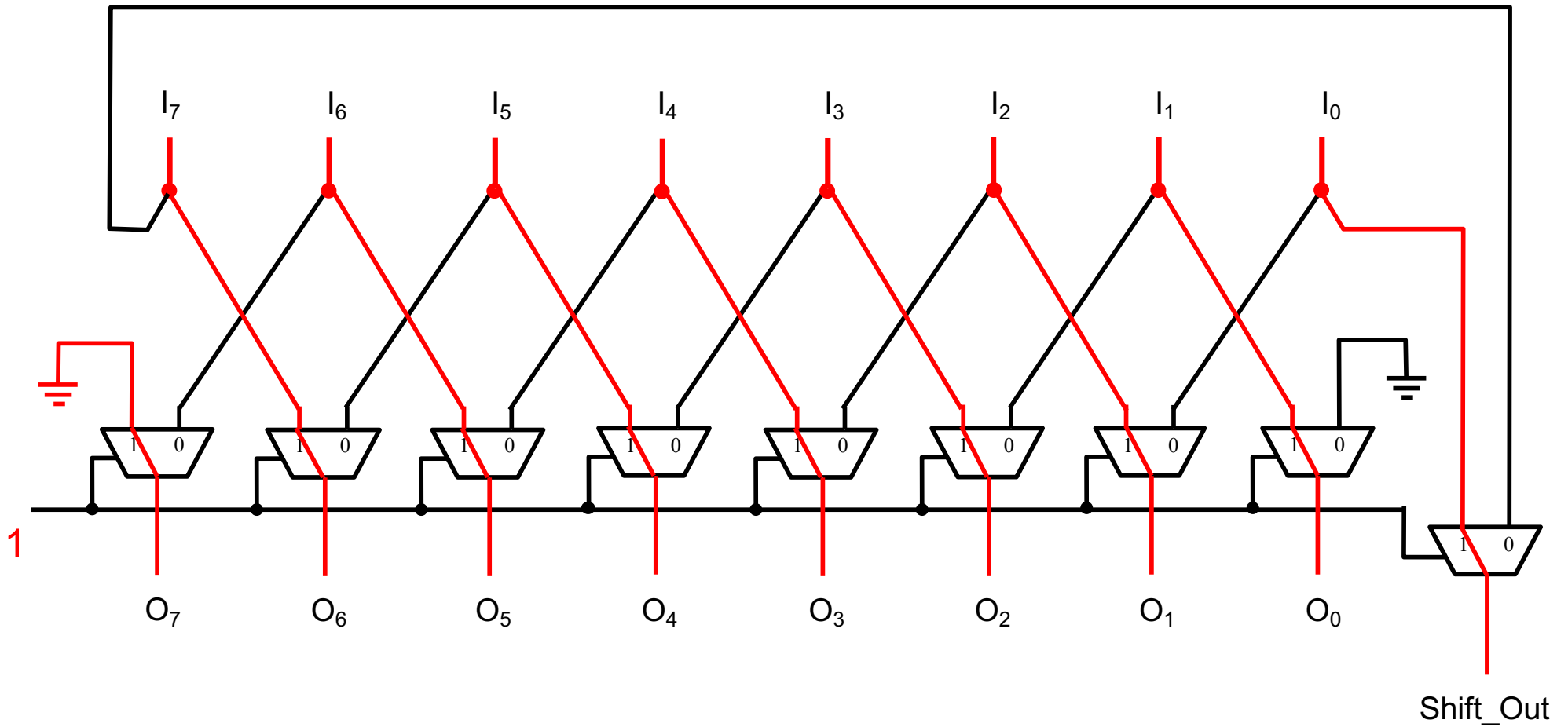


Shift Left

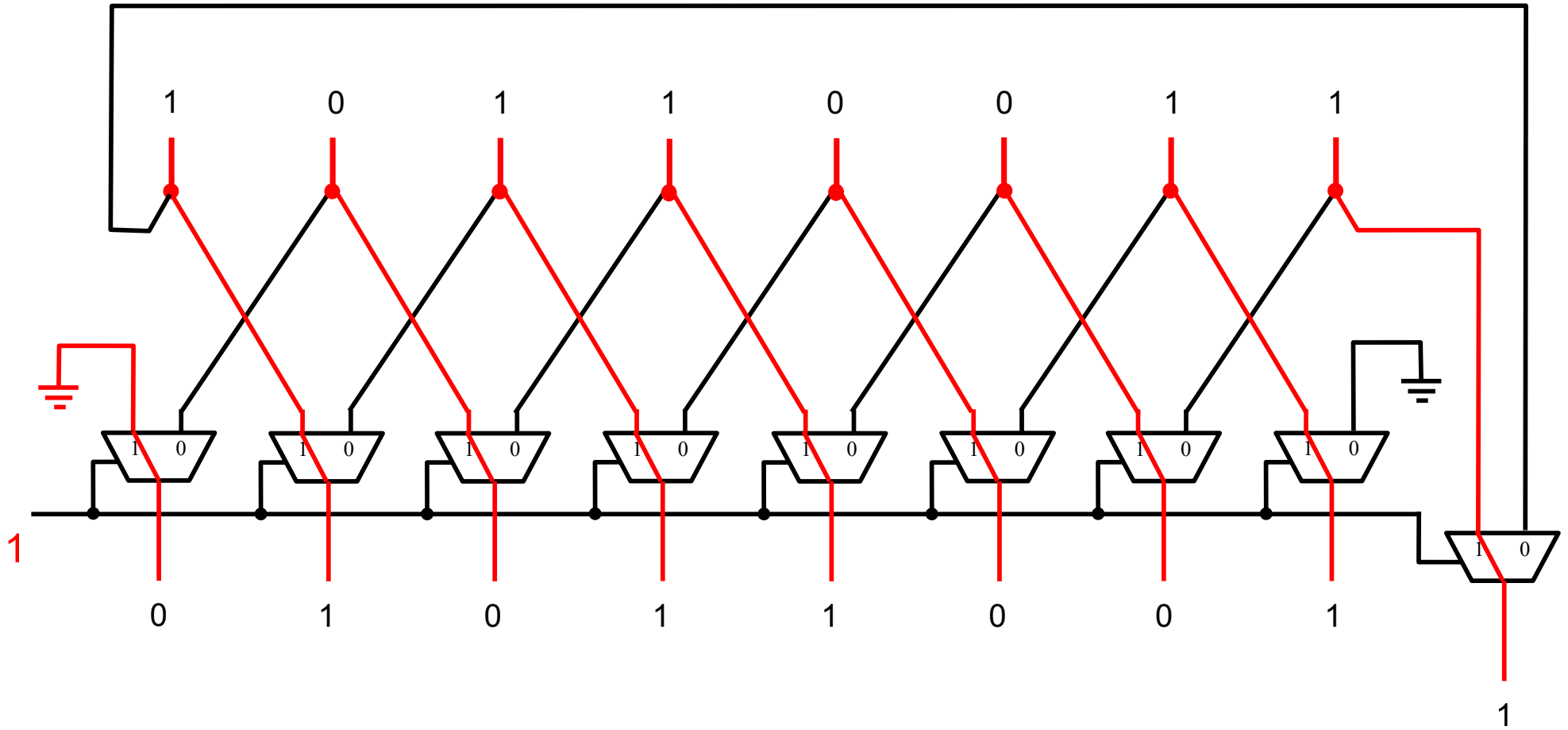


i281 CPU

Shift Right



Shift Right



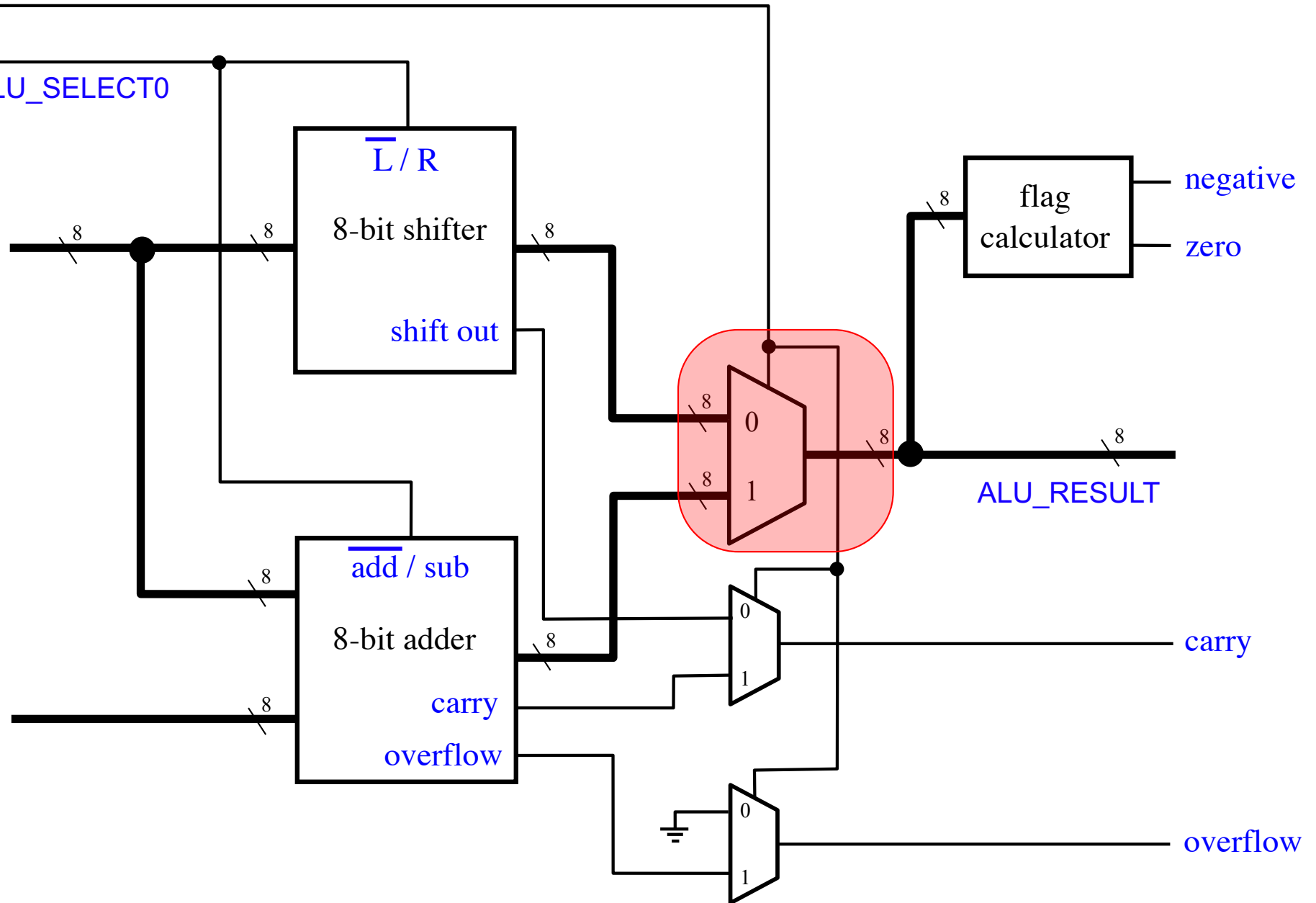
i281 CPU

Bus Multiplexer

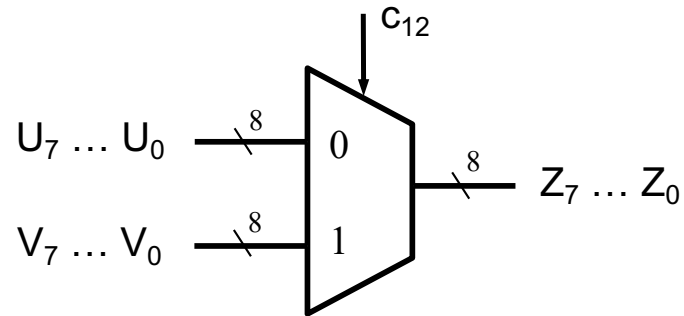
The Internal ALU Bus Multiplexer

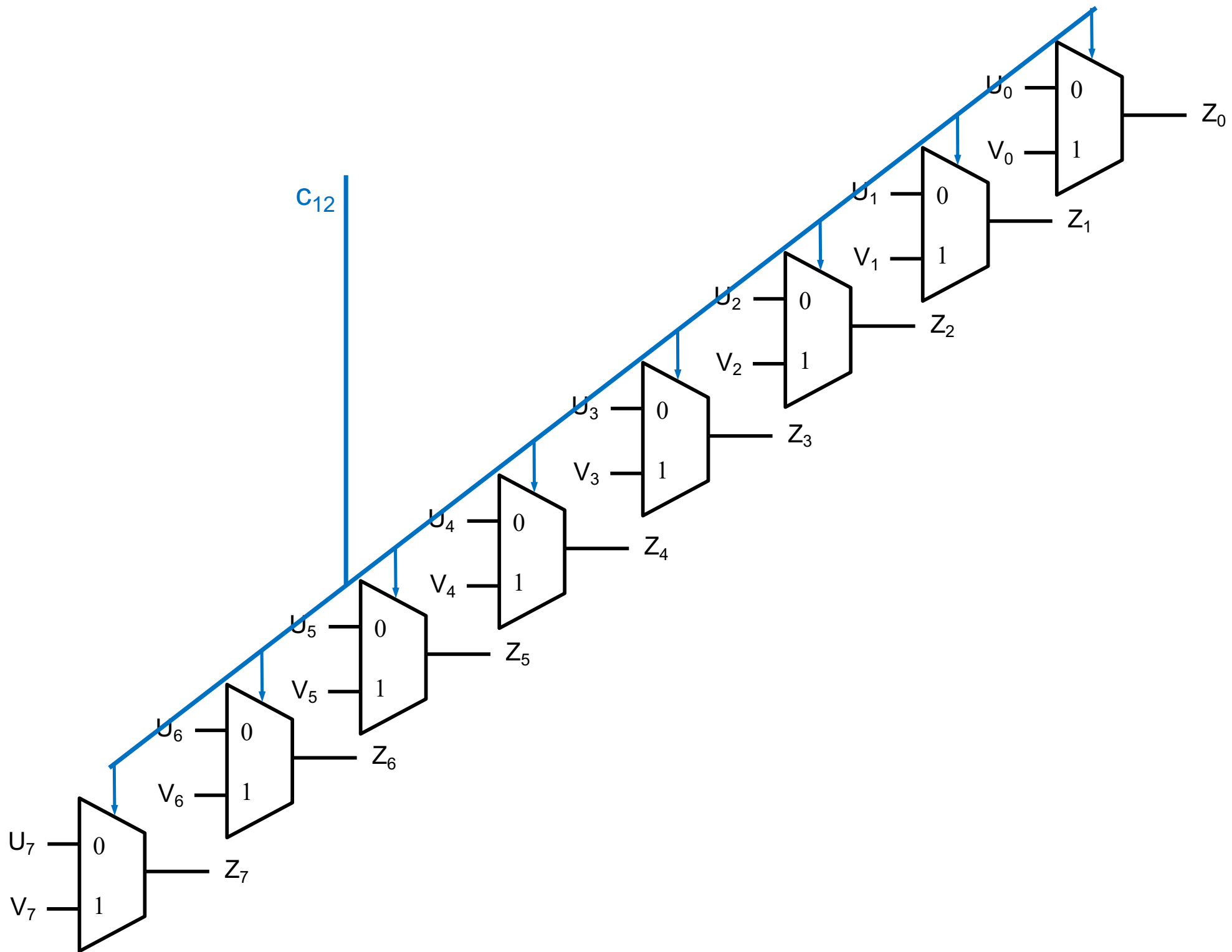
ALU_SELECT1

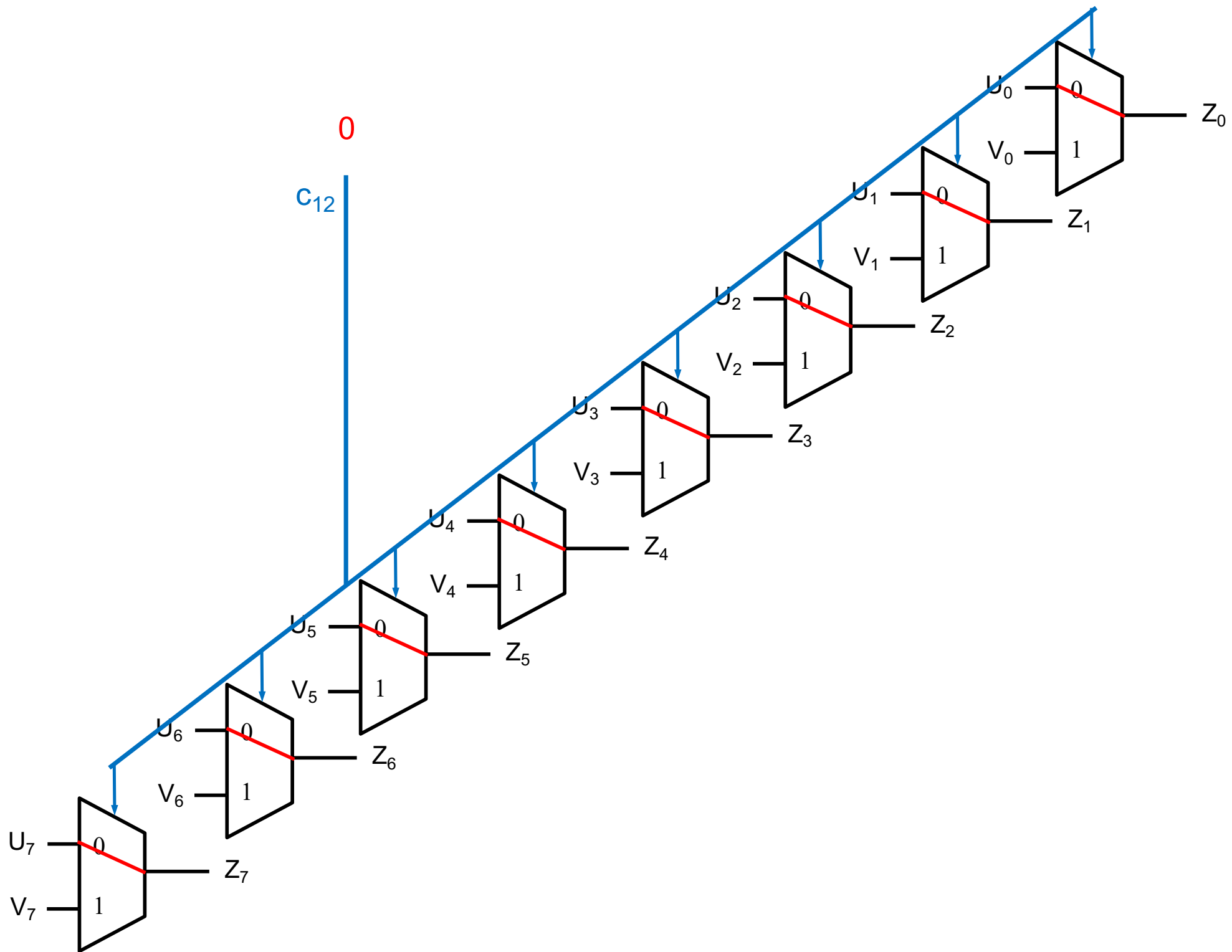
ALU_SELECT0

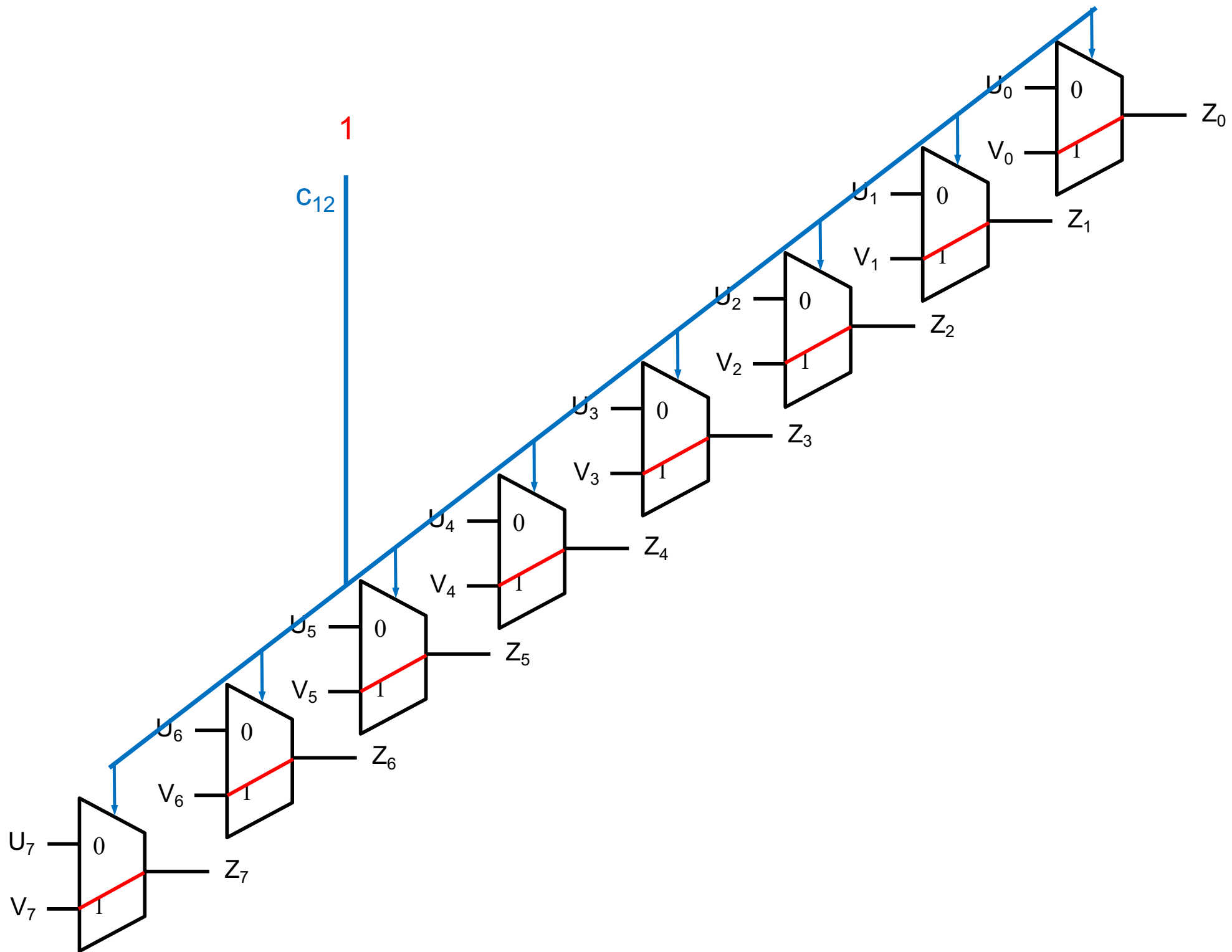


2-to-1 Bus Multiplexer (with 8-bit lines)

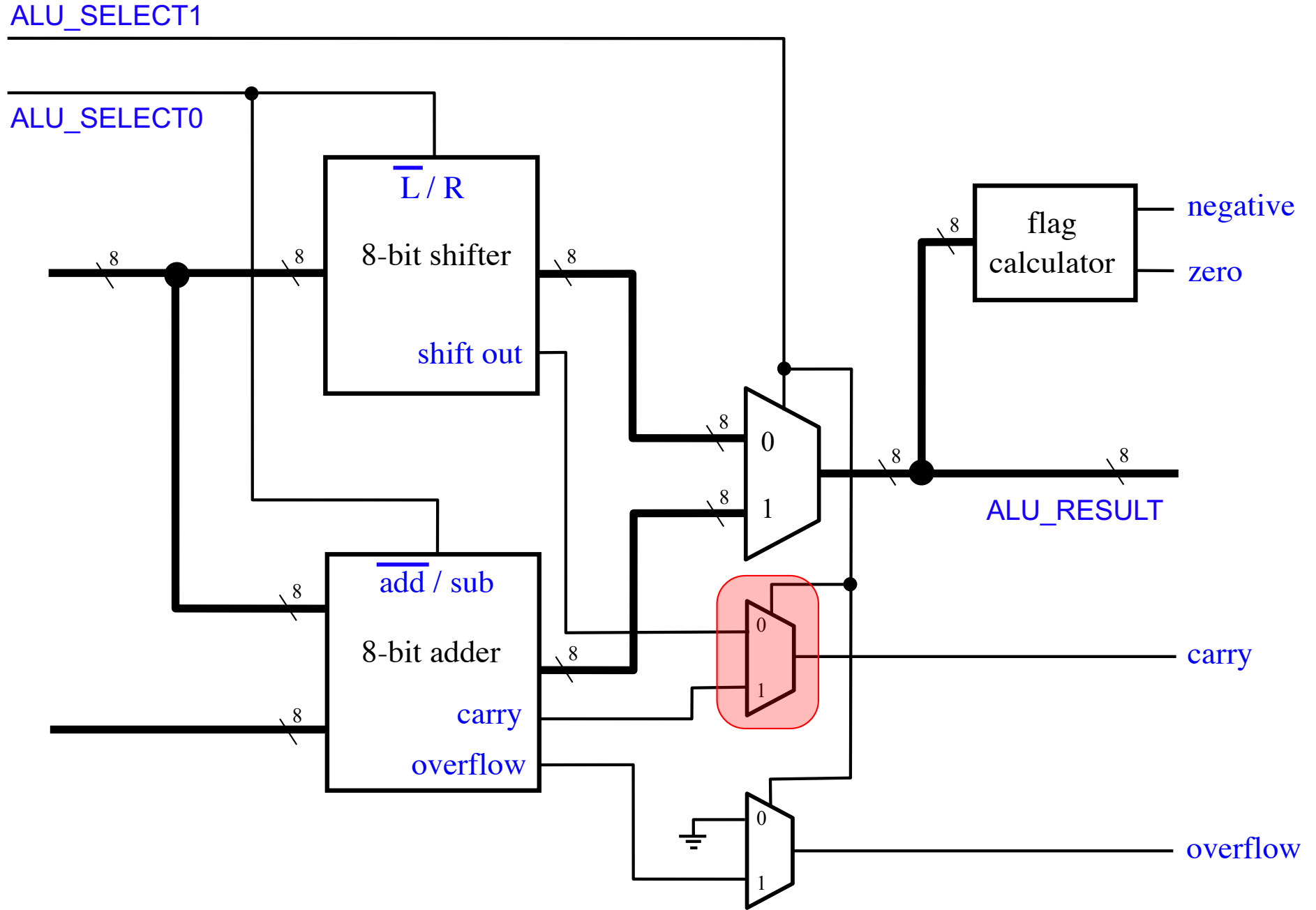




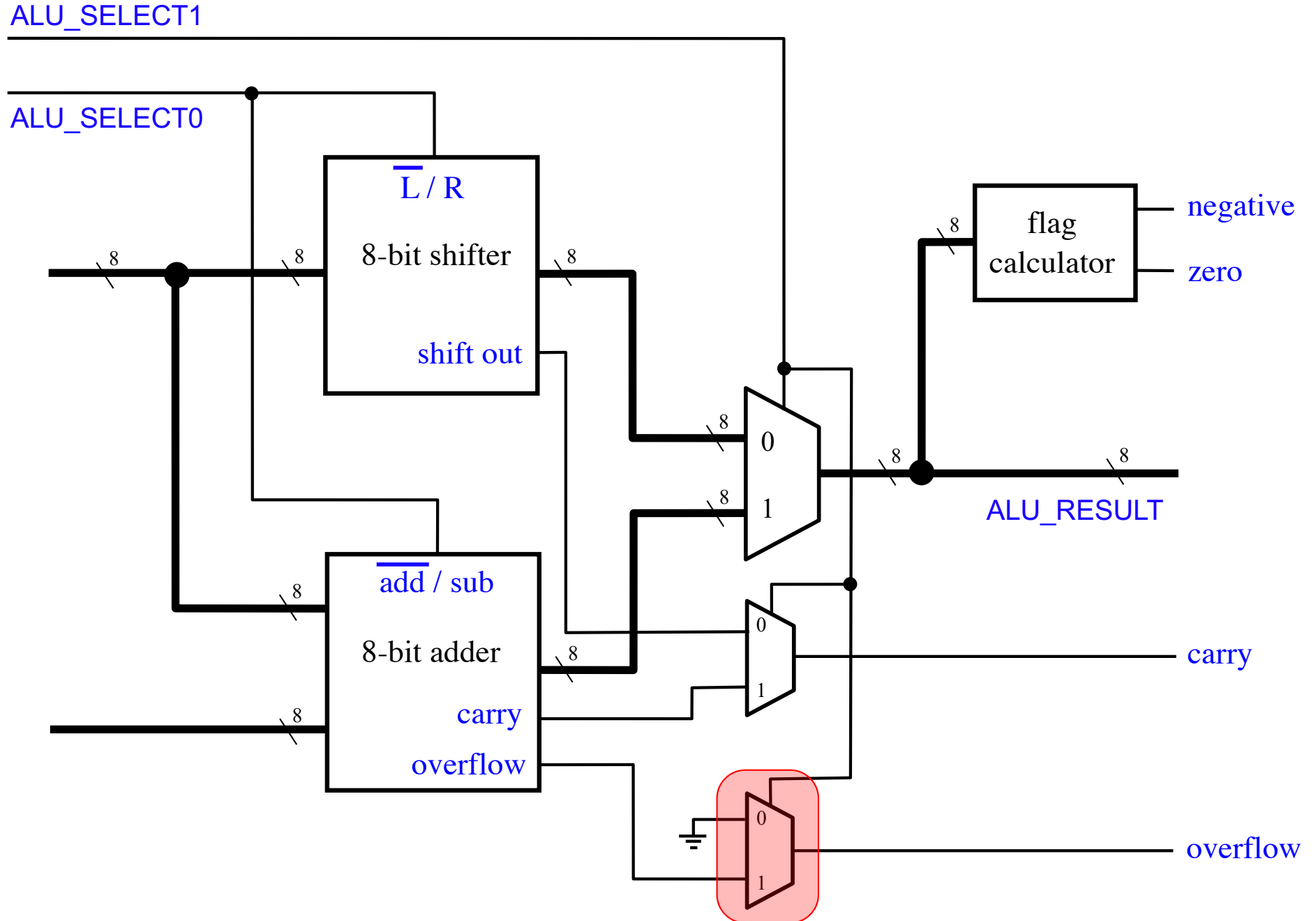




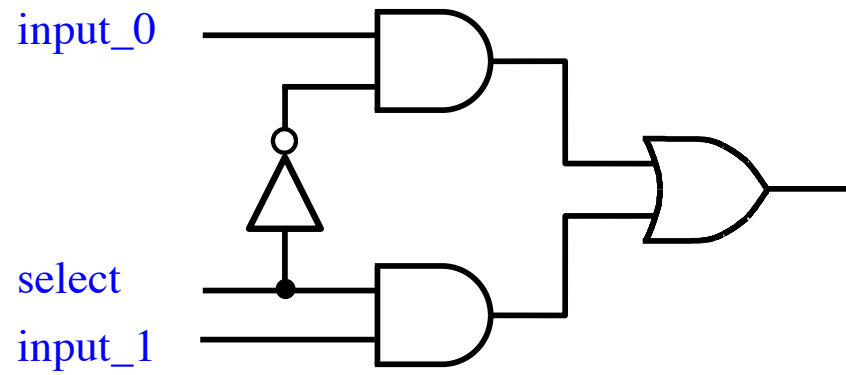
2-to-1 Multiplexer



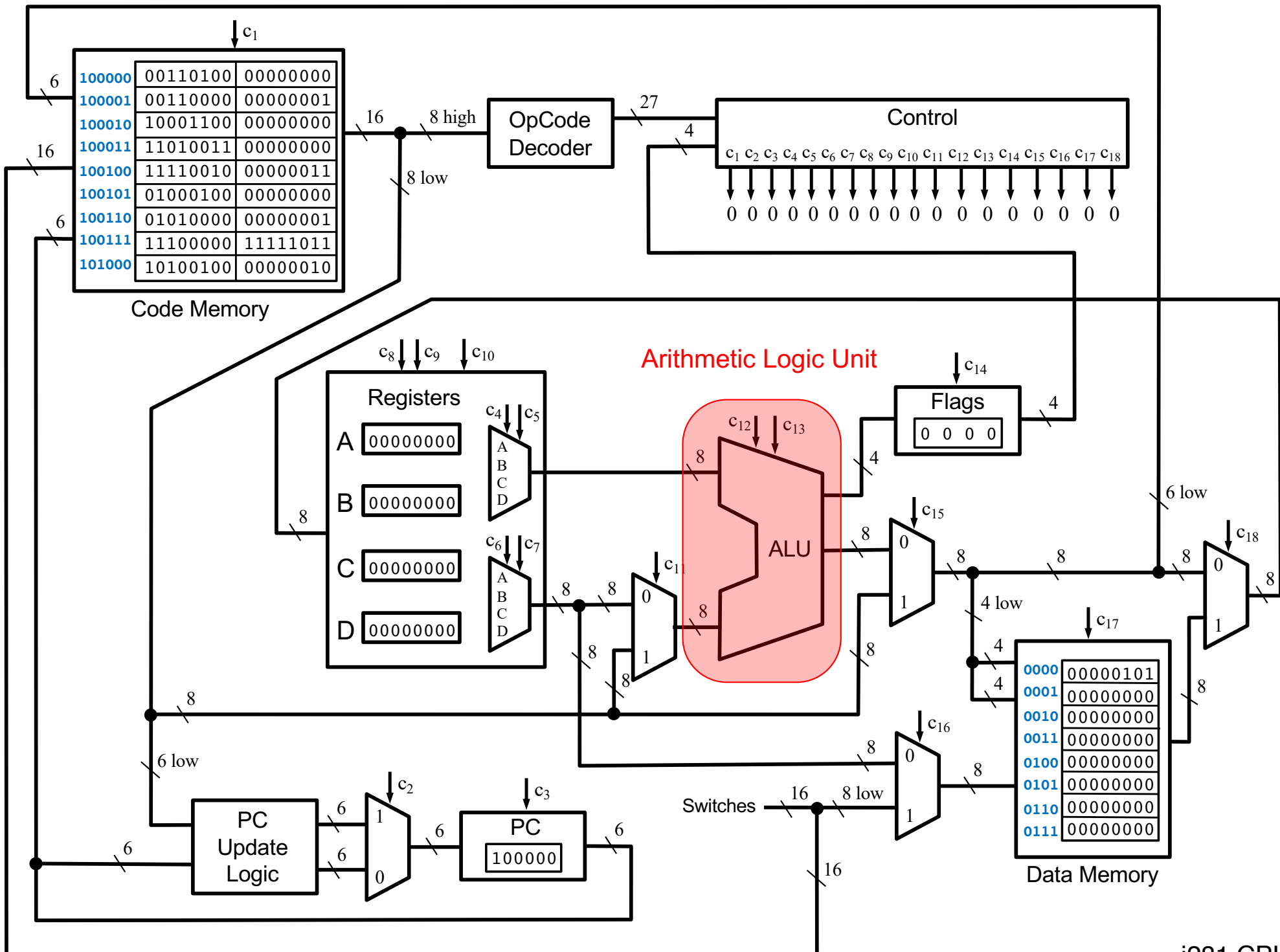
2-to-1 Multiplexer



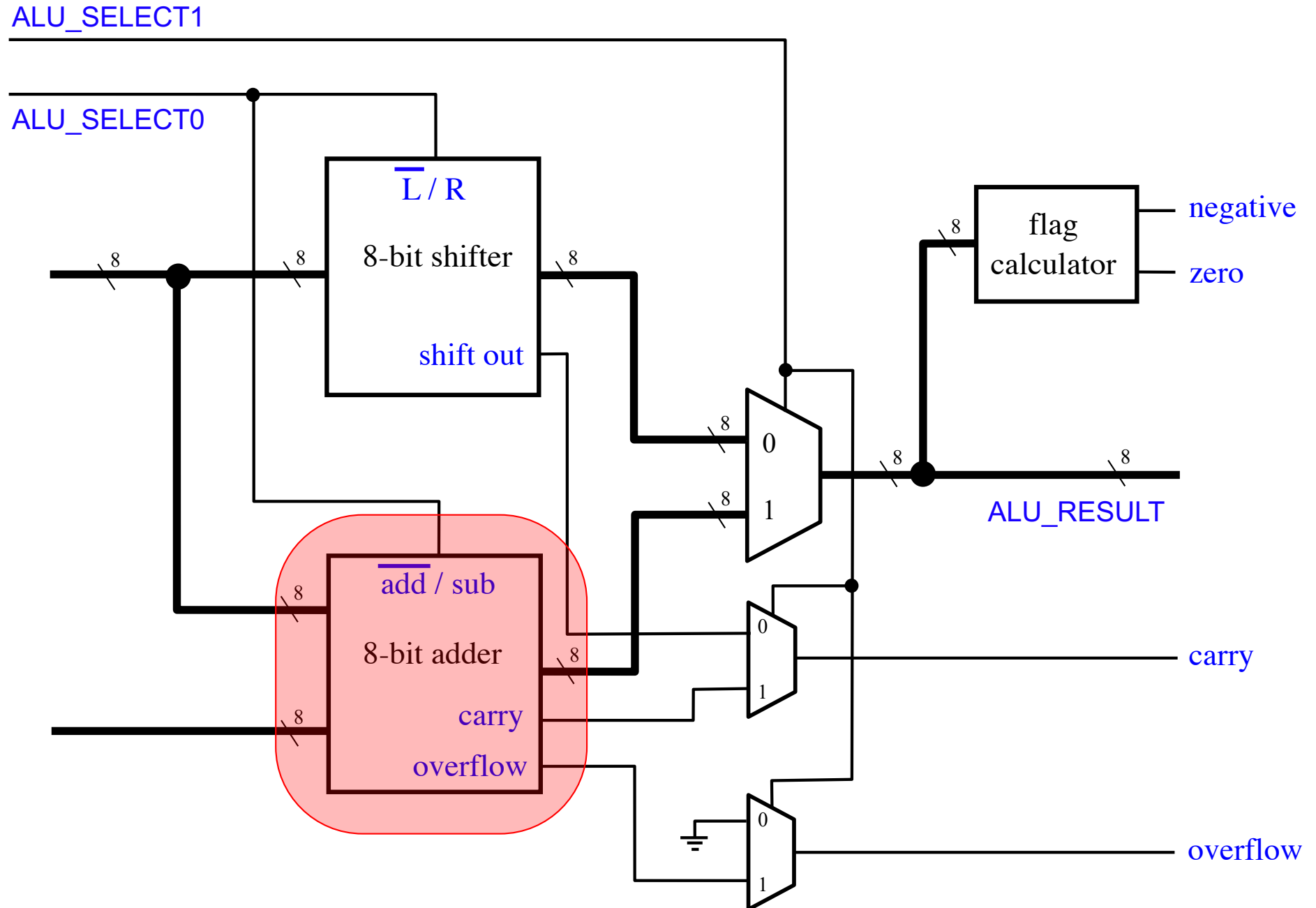
2-to-1 Multiplexer



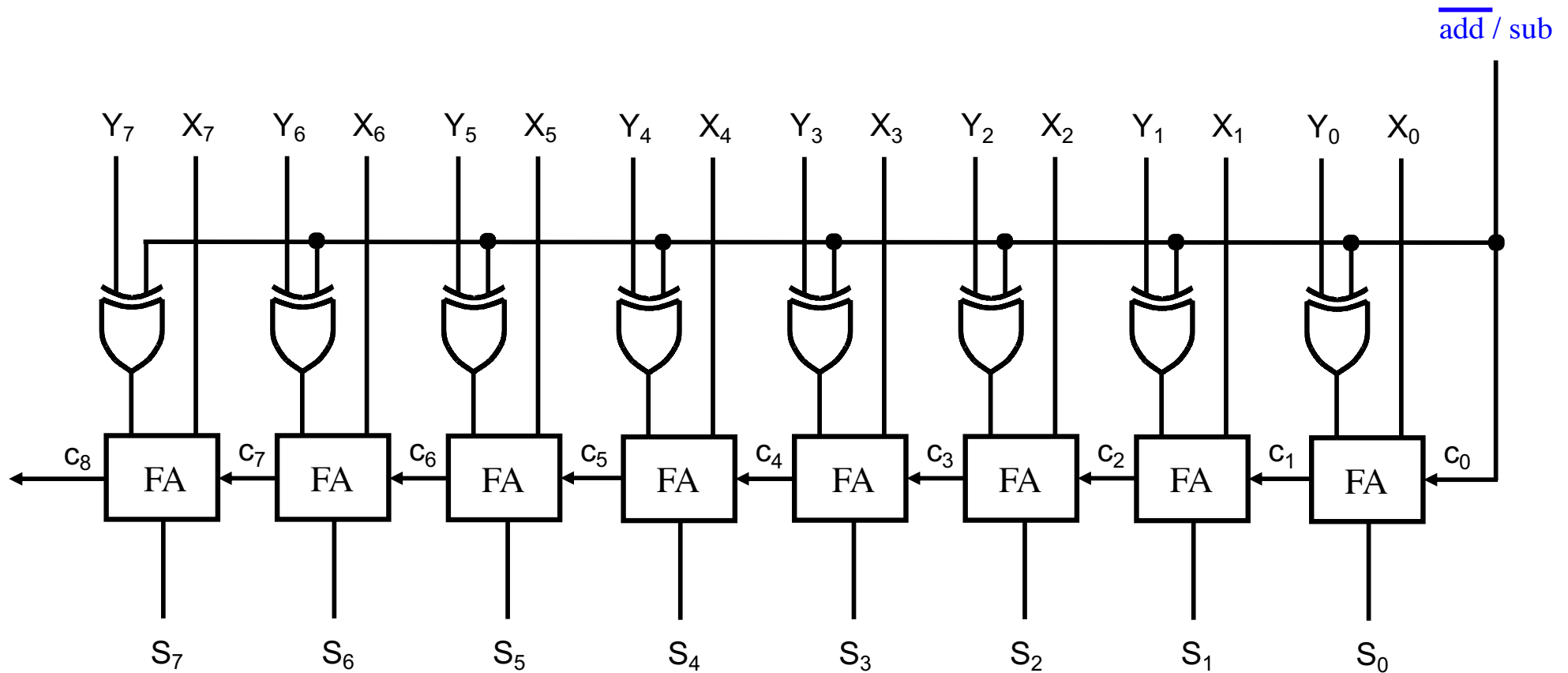
The adder/subtractor of the i281 CPU



The Adder / Subtractor

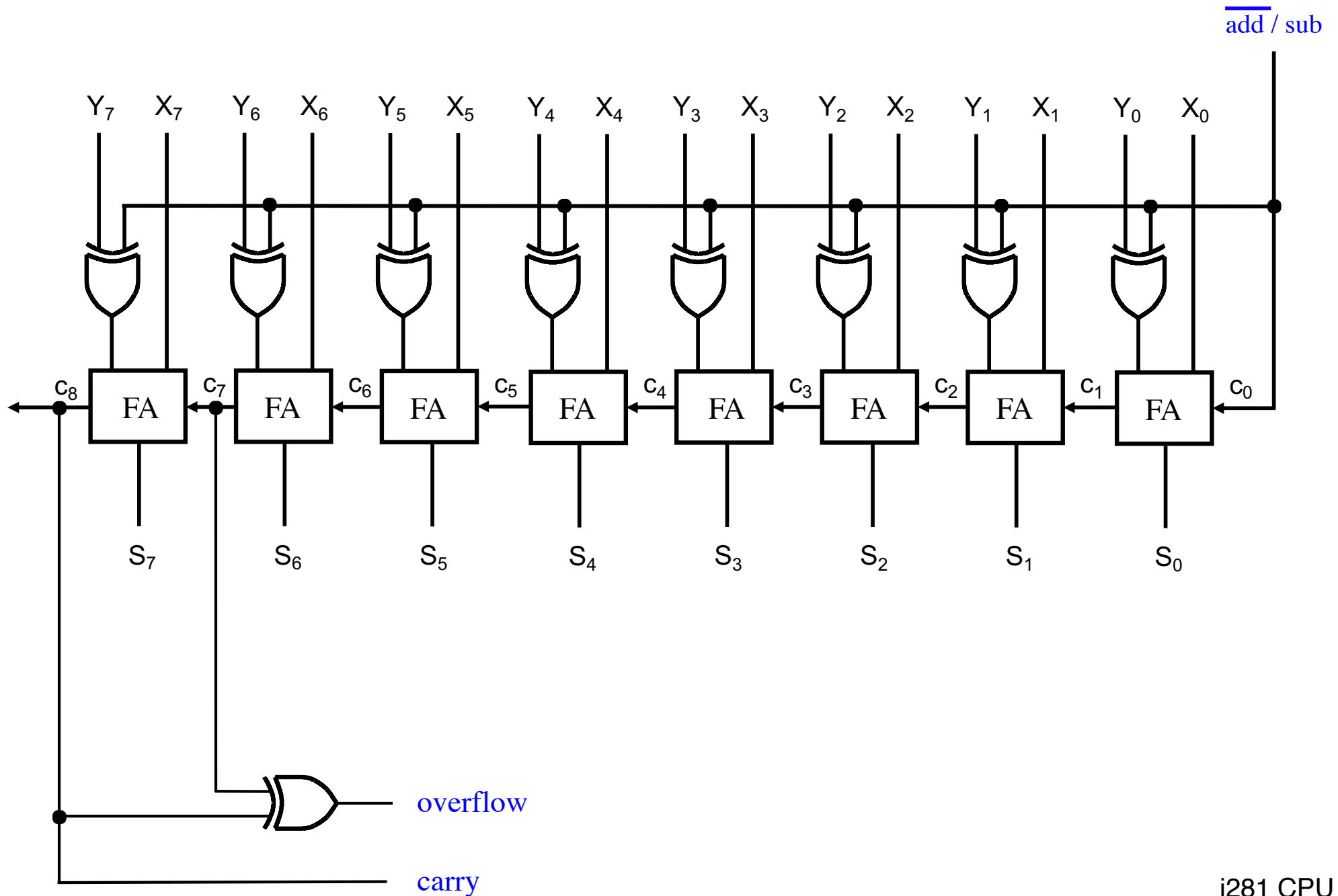


The Adder / Subtractor



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

The Adder / Subtractor



Abbreviations for the Flags

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

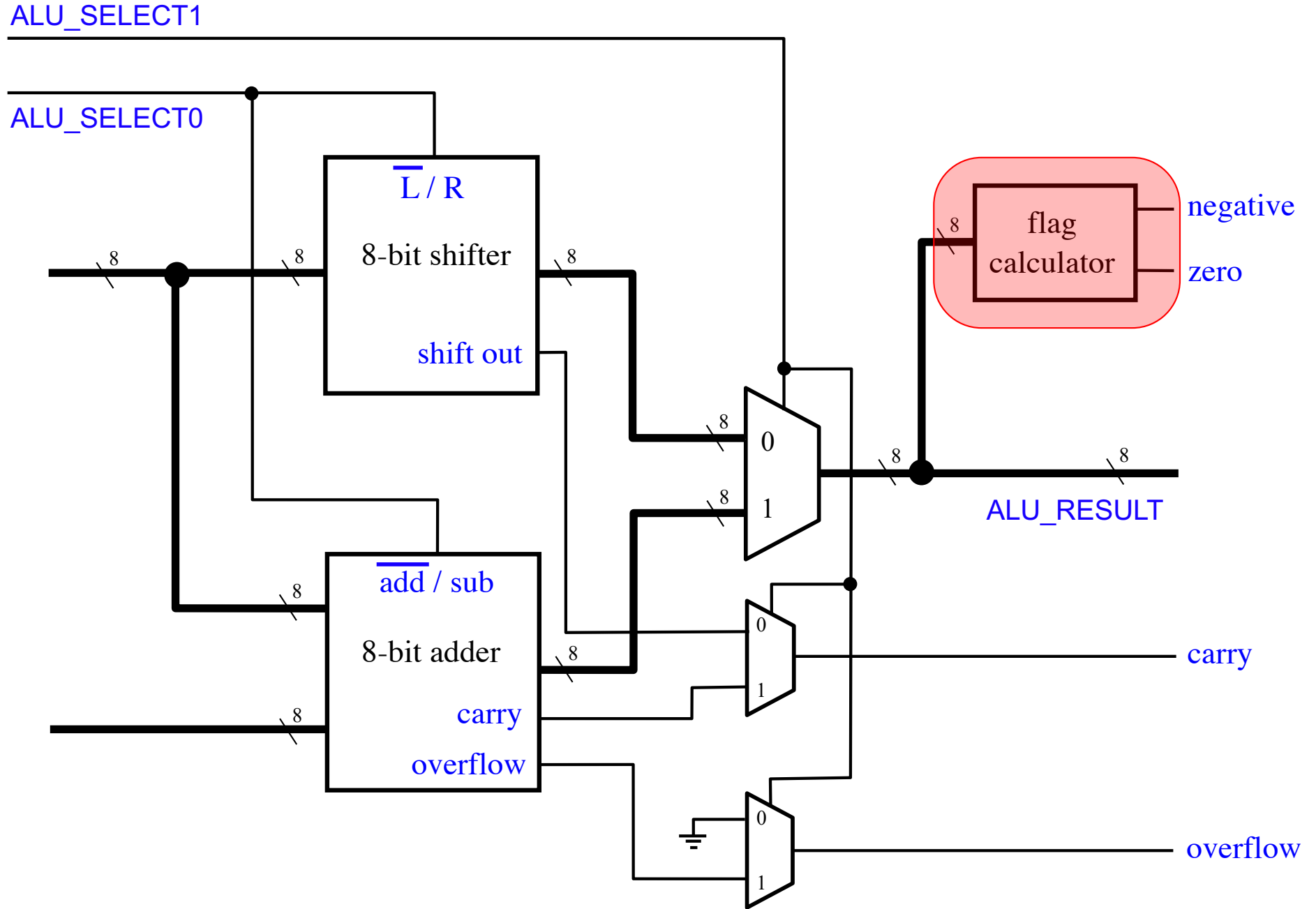
Abbreviations for the Flags

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

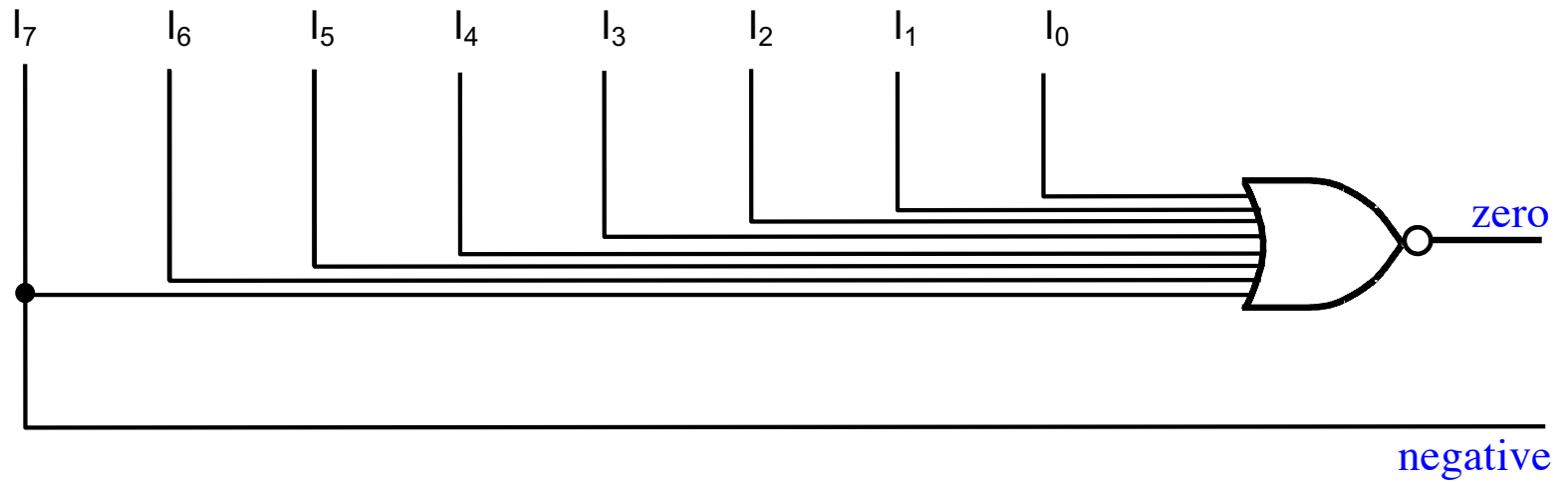
In some CPU architectures the carry flag means borrow. And it could be inverted relative to the previous diagram.

The flag calculator of the i281 CPU

The ALU Flag Calculator



The ALU Flag Calculator

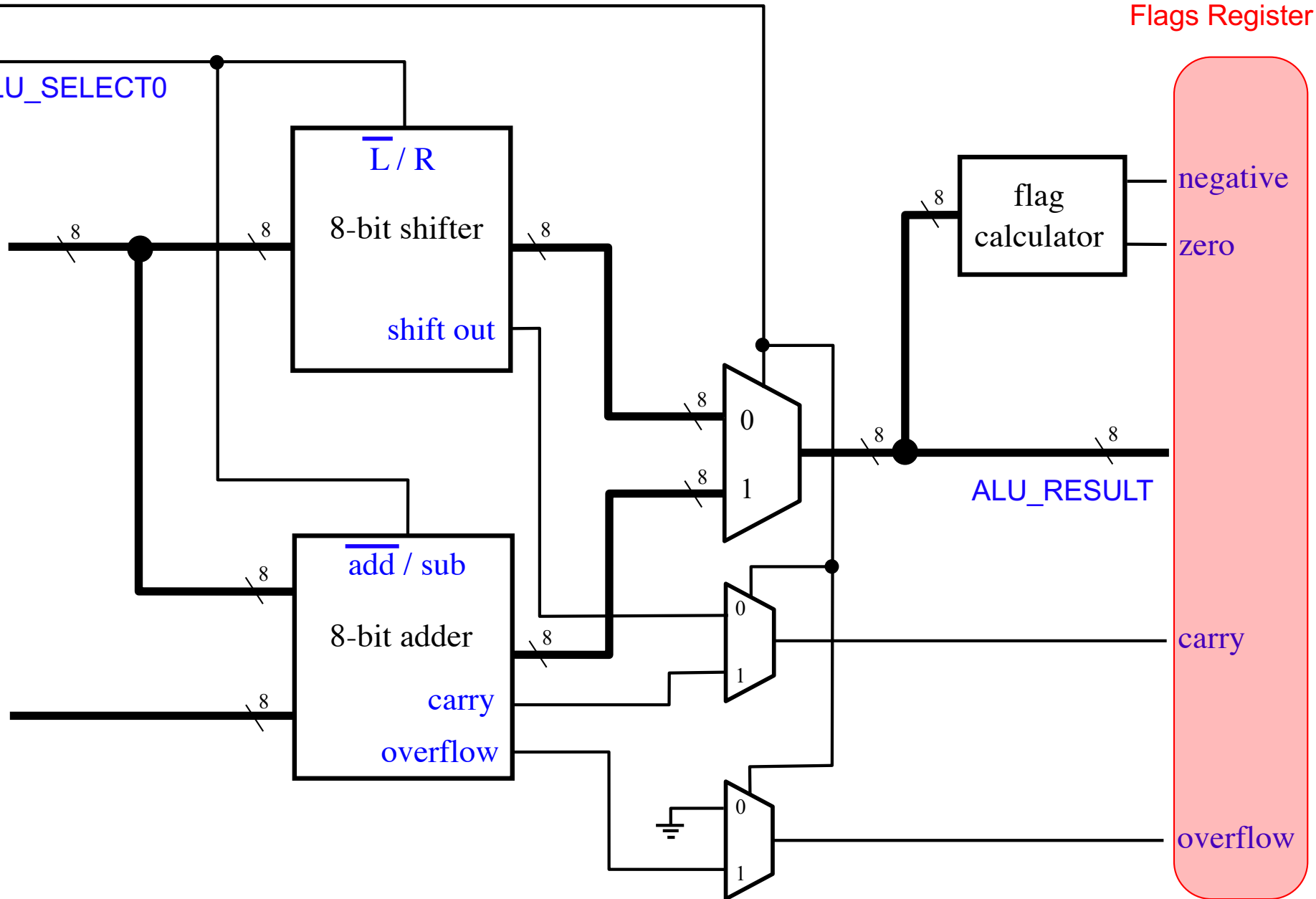


ALU Outputs to the Flags Register

ALU_SELECT1

ALU_SELECT0

4 Outputs to the
Flags Register



negative

zero

ALU_RESULT

carry

overflow

Comparison of Signed Numbers

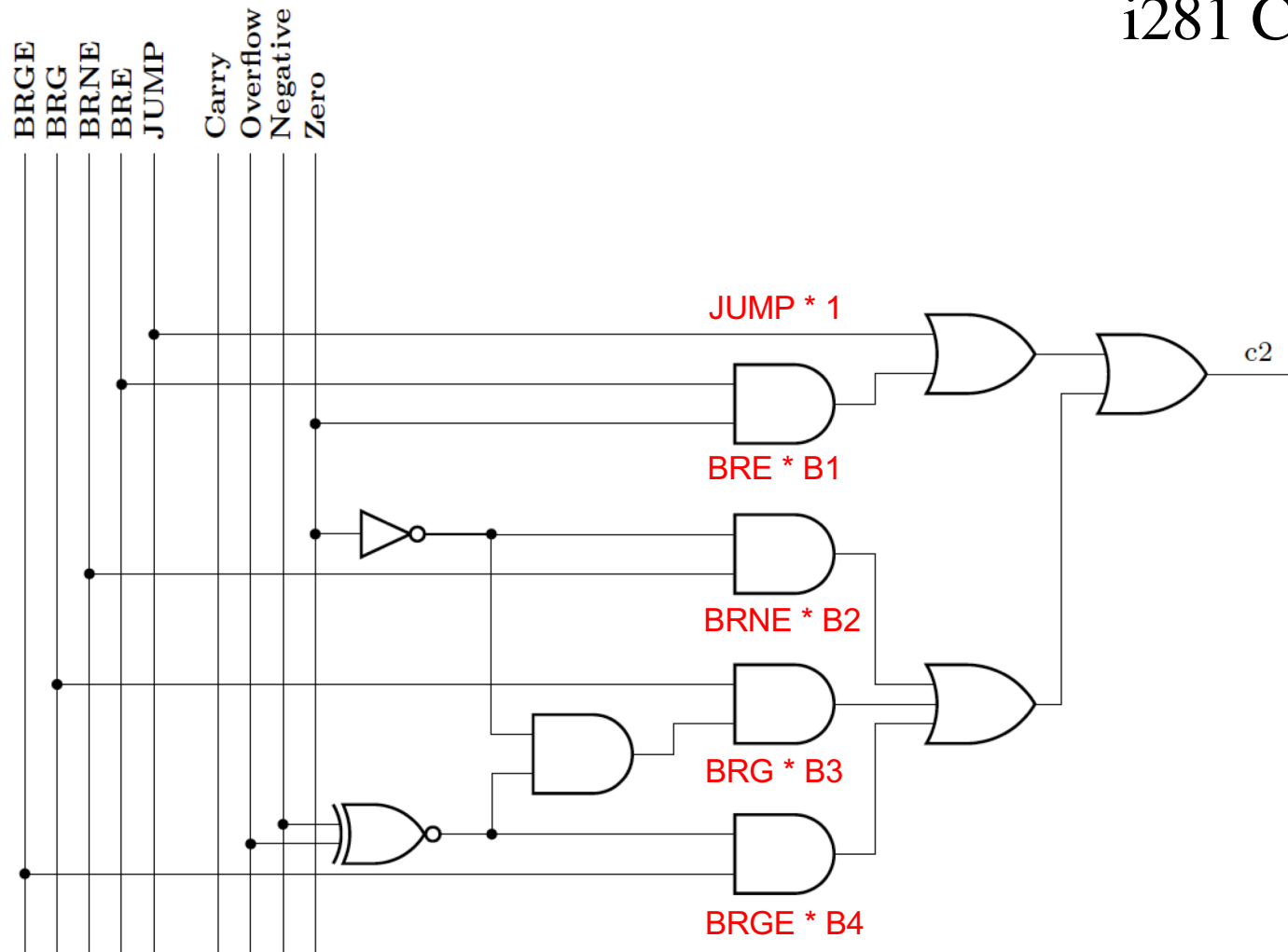
Comparison of Signed Numbers

- **Equal** $ZF = 1$
- **Not equal** $ZF = 0$
- **Greater** $ZF = 0$ and $NF = OF$
- **Greater or Equal** $NF = OF$
- **Less** $NF \neq OF$
- **Less or Equal** $ZF = 1$ or $NF \neq OF$

Comparison of Signed Numbers

- **Equal** ZF
- **Not equal** \overline{ZF}
- **Greater** $\overline{ZF} \cdot XNOR(NF, OF)$
- **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																

C₂ is the OR of these five times the OPCODE

B1= ZF
 B2= $\sim ZF$
 B3= $\text{AND}(\sim ZF, \text{XNOR}(\text{NF}, \text{OF}))$
 B4= $\text{XNOR}(\text{NF}, \text{OF})$

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