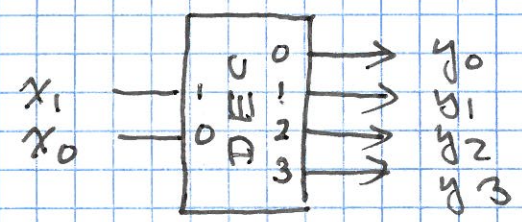
STANDARD COMBINATIONAL MODULES

- DECODERS
- ENCODERS
- MULTIPLEXERS (Selectors)
- DEMULTIPLEXERS (Distributors)
- SHIFTERS

CS M51A

OVERVIEW OF MAIN COMBINATIONAL MODULES

1. DECODER



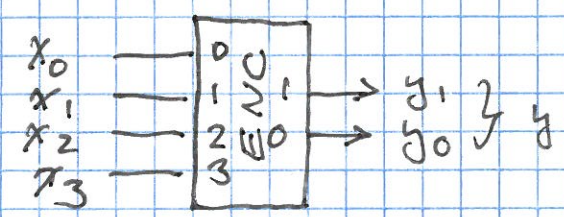
x_1	x_0	y_3	y_2	y_1	y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

BINARY — UNARY (1-OUT-OF-n)

$y_i = m_i(x_1, x_0)$

BINARY DECODER OF k INPUTS, PRODUCES 2^k OUTPUTS, CORRESPONDING TO ALL MINTERMS.

2. ENCODER



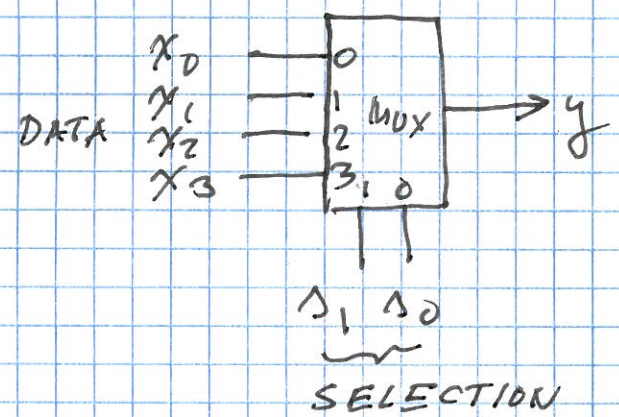
IN UNARY CODE — IN BINARY CODE

$y = i$ IF $x_i = 1$, $i \in \{0, 1, 2, 3\}$

INDEX OF INPUT WITH VALUE 1

CSM51A

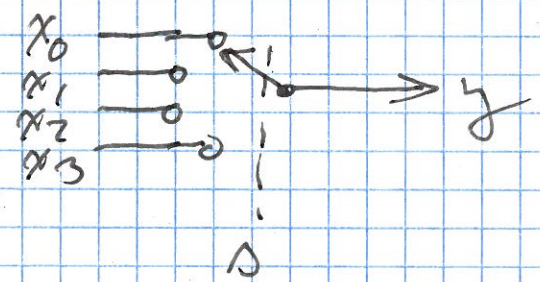
3. MULTIPLEXER — DATA ROUTING



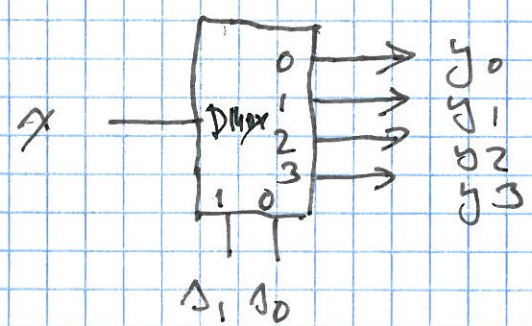
$s = 2s_1 + s_0 \in \{0, 1, 2, 3\}$

$y = x_s$

SWITCH EQUIVALENT:



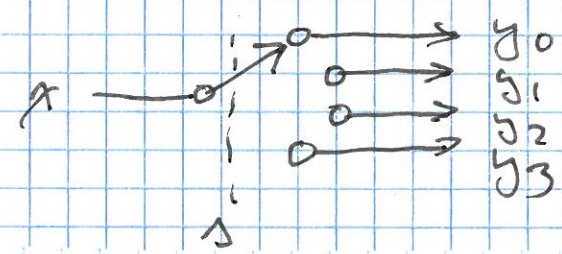
4. DEMULTIPLEXER — DATA ROUTING



$s = 2s_1 + s_0 \in \{0, 1, 2, 3\}$

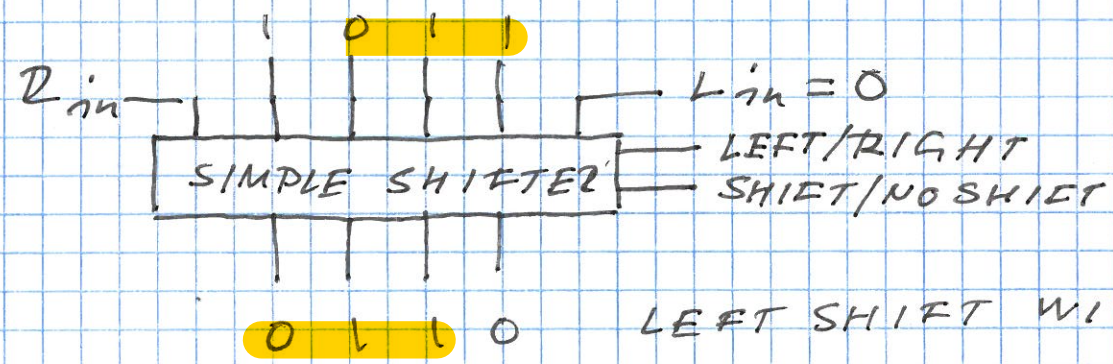
$y_s = x$

SWITCH EQUIVALENT:



CS M51A

5. SHIFTER — BIT-VECTOR MOVEMENT



LEFT SHIFT WITH 0 INSERT

THERE ARE OTHER WIDELY USED COMBINATIONAL MODULES. FOR EXAMPLE,

- ADDERS
- ARITHMETIC-LOGIC UNITS (ALUs)
- MULTIPLIERS

BINARY DECODERS

HIGH-LEVEL DESCRIPTION:

INPUTS: $\underline{x} = (x_{n-1}, \ldots, x_0), \quad x_j \in \{0, 1\}$

Enable $E \in \{0, 1\}$

OUTPUTS: $\underline{y} = (y_{2^n-1}, \ldots, y_0), \quad y_i \in \{0, 1\}$

FUNCTION: $y_i = \begin{cases} 1 \ \textbf{if} \ \ (x = i) \ \textbf{and} \ (E = 1) \\ 0 \ \textbf{otherwise} \end{cases}$

$$x = \sum_{j=0}^{n-1} x_j 2^j$$

and

$$i = 0, \ldots, 2^n - 1$$

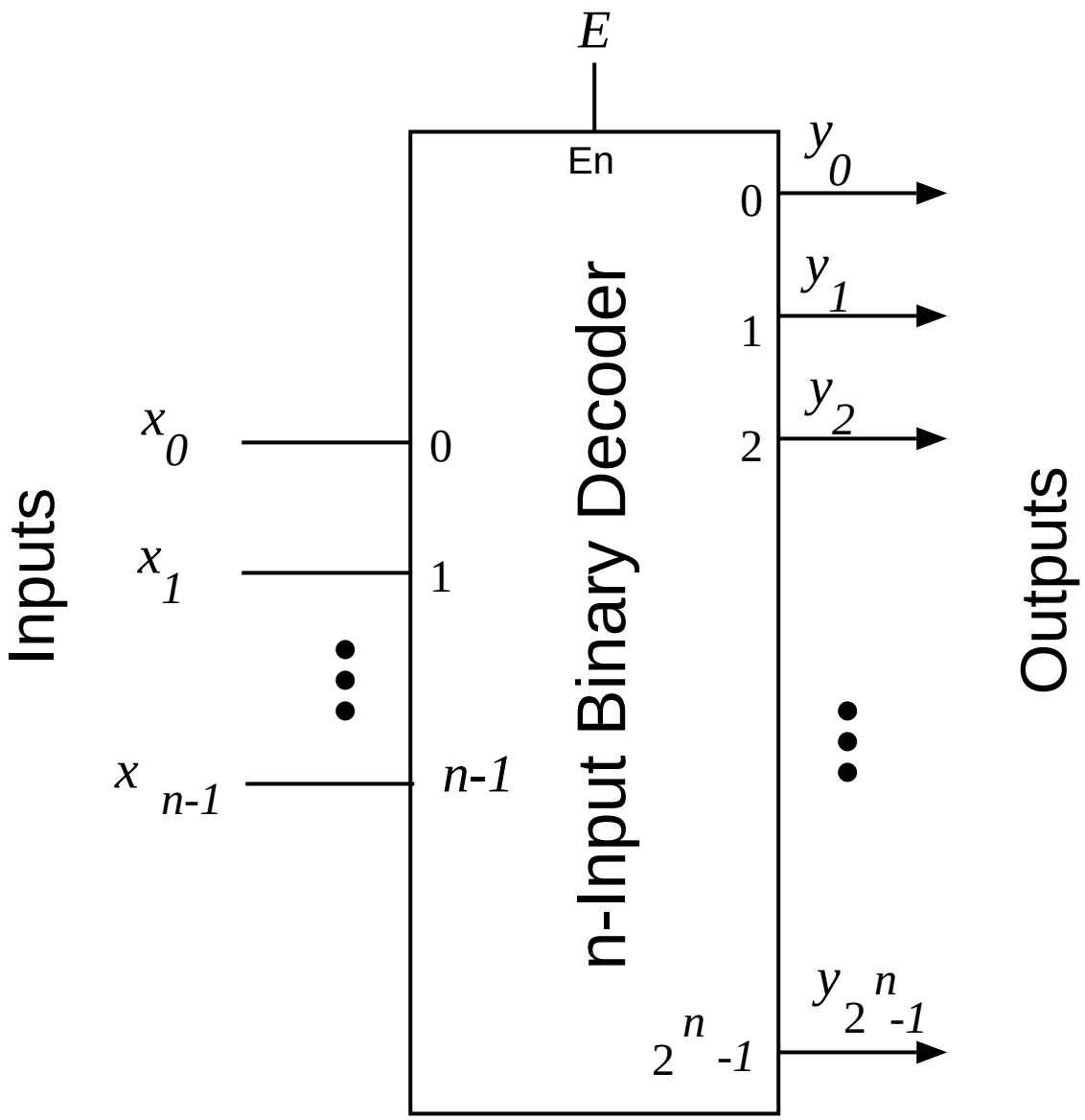


Figure 9.1: n-INPUT BINARY DECODER.

EXAMPLE 9.1: 3-INPUT BINARY DECODER

E	x_2	x_1	x_0	x	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0	0	0	1	0
1	0	1	0	2	0	0	0	0	0	1	0	0
1	0	1	1	3	0	0	0	0	1	0	0	0
1	1	0	0	4	0	0	0	1	0	0	0	0
1	1	0	1	5	0	0	1	0	0	0	0	0
1	1	1	0	6	0	1	0	0	0	0	0	0
1	1	1	1	7	1	0	0	0	0	0	0	0
0	-	-	-	-	0	0	0	0	0	0	0	0

Example 9.1 (cont.)

BINARY SPECIFICATION:

INPUTS: $\underline{x} = (x_{n-1}, \ldots, x_0), \quad x_j \in \{0, 1\}$
$E \in \{0, 1\}$

OUTPUTS: $\underline{y} = (y_{2^n-1}, \ldots, y_0), \quad y_i \in \{0, 1\}$

FUNCTION: $y_i = E \cdot m_i(\underline{x}) \quad , \quad i = 0, \ldots, 2^n - 1$

EXAMPLE 9.2: IMPLEMENTATION OF 2-INPUT DECODER

$$y_0 = x_1' x_0' E \quad y_1 = x_1' x_0 E \quad y_2 = x_1 x_0' E \quad y_3 = x_1 x_0 E$$

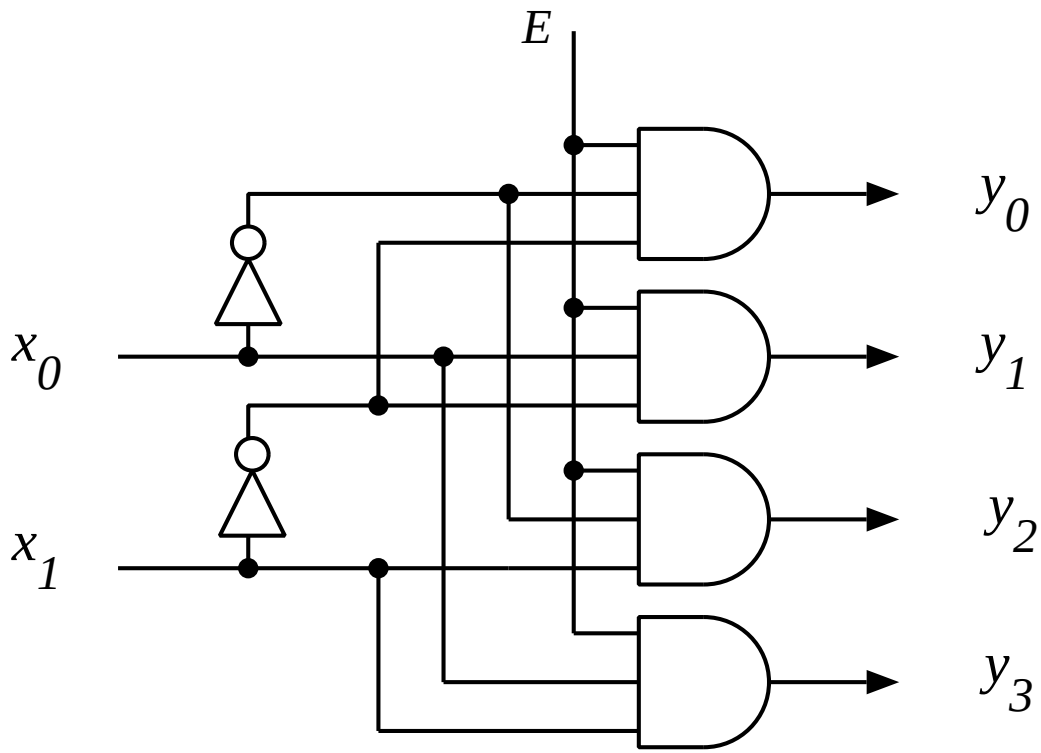


Figure 9.2: GATE NETWORK IMPLEMENTATION OF 2-INPUT BINARY DECODER.

DECODER USES

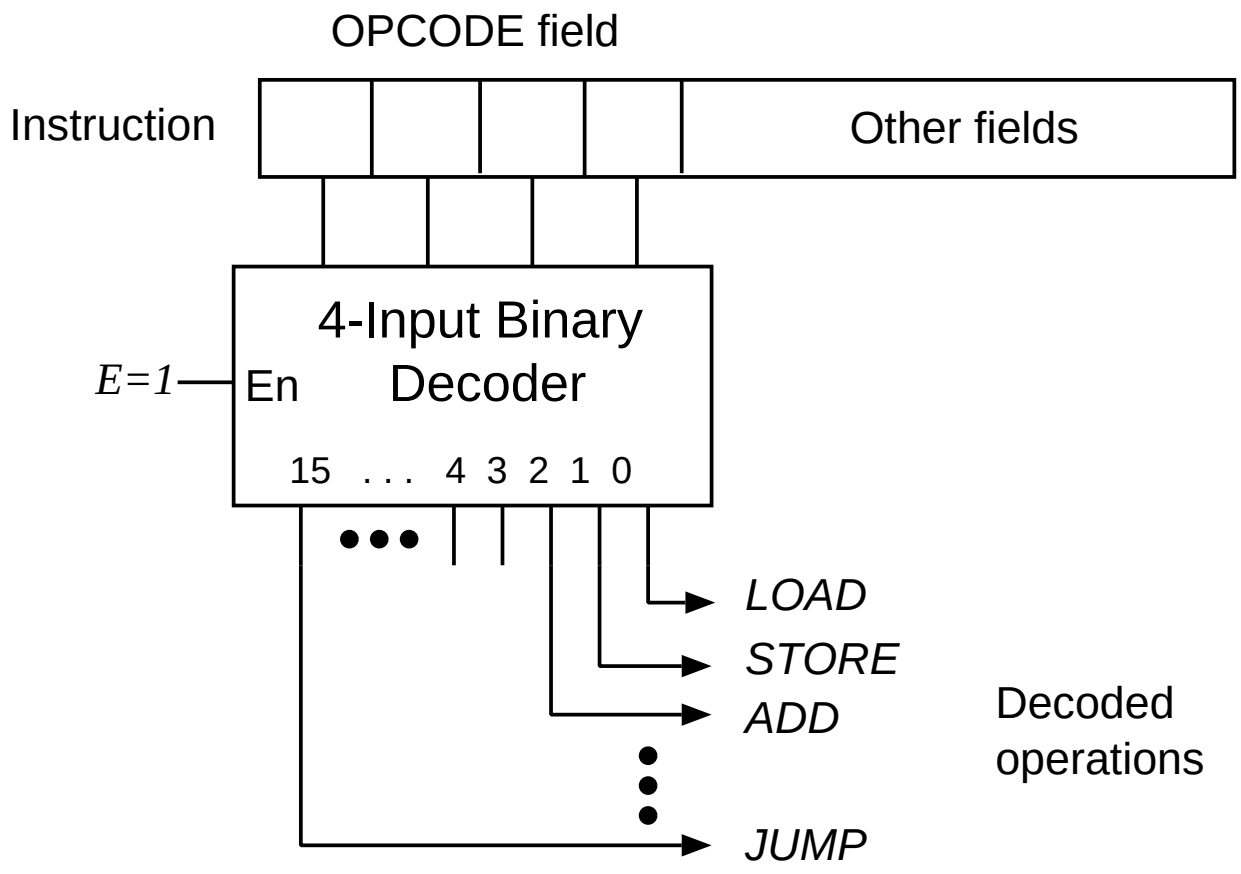
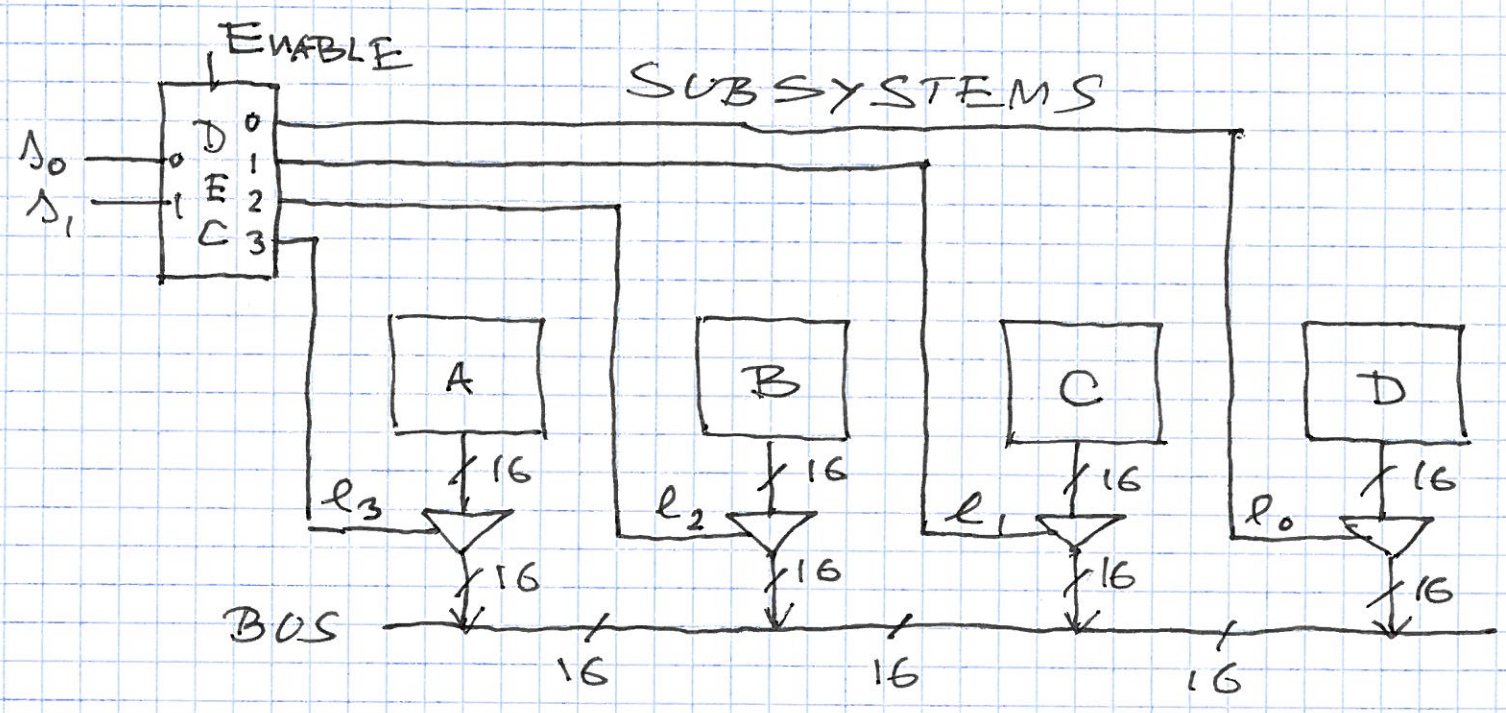


Figure 9.3: OPERATION DECODING.

CSMSIA

BUS ACCESS CONTROL



(E=1)

s_1	s_0	e_3	e_2	e_1	e_0	BUS CONNECTED TO
0	0	0	0	0	1	D
0	1	0	0	1	0	C
1	0	0	1	0	0	B
1	1	1	0	0	0	A

(E=0) $e_i = 0 \ \forall i$ NO SUBSYSTEM CONNECTED TO THE BUS

DECODER USES

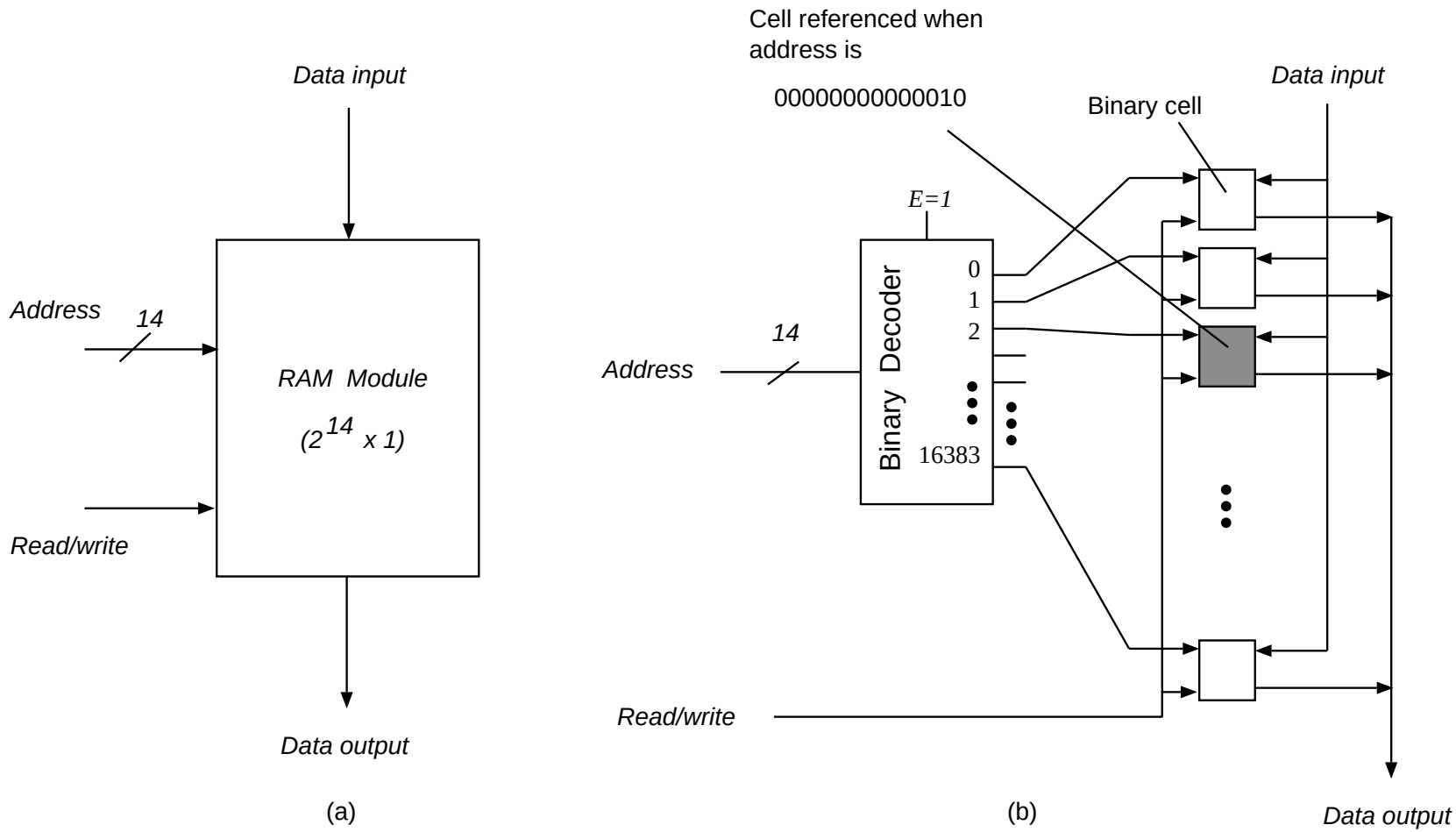
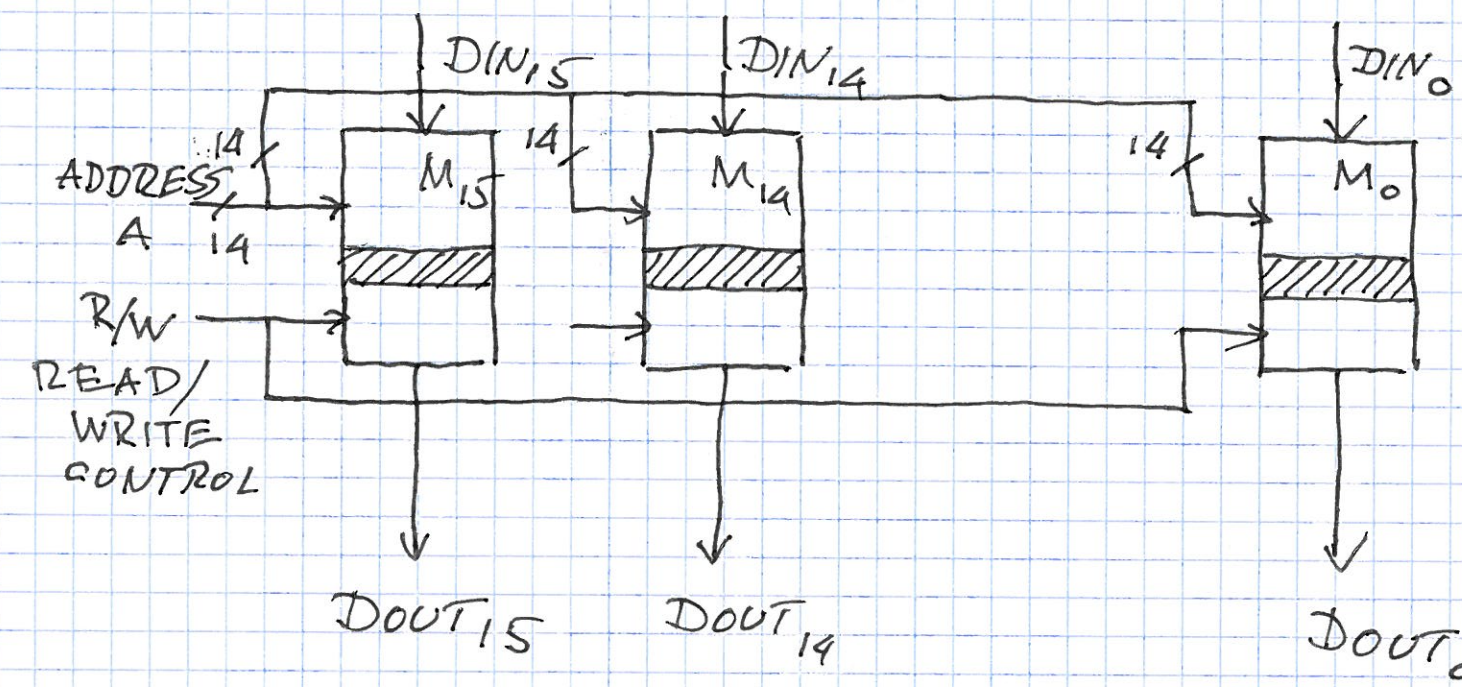


Figure 9.4: RANDOM ACCESS MEMORY (RAM): a) MODULE; b) ADDRESSING OF BINARY CELLS.

CS MS1A

MULTIBIT MEMORY: $2^{14} \times 16$-BIT RAM



DATA-IN

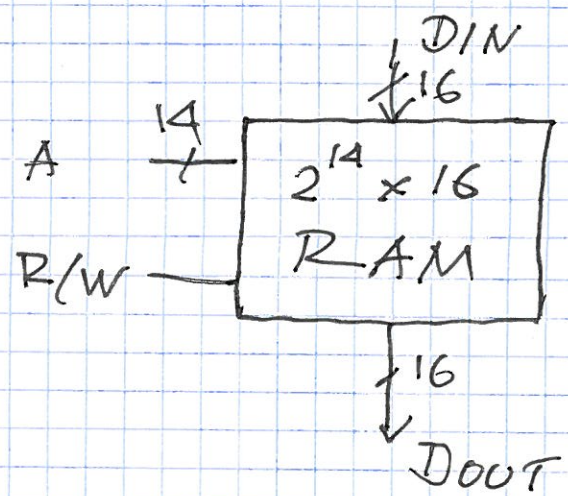
$DIN = (DIN_{15}, \ldots, DIN_0)$

$M_i, \; i = 0, \ldots, 15$

$2^{14} \times 1$ RAM MODULE

DATA-OUT

$DOUT = (DOUT_{15}, \ldots, DOUT_0)$



$$\begin{cases} 2^{14} \text{ WORDS} \\ 16 \text{ BITS / WORD} \end{cases}$$

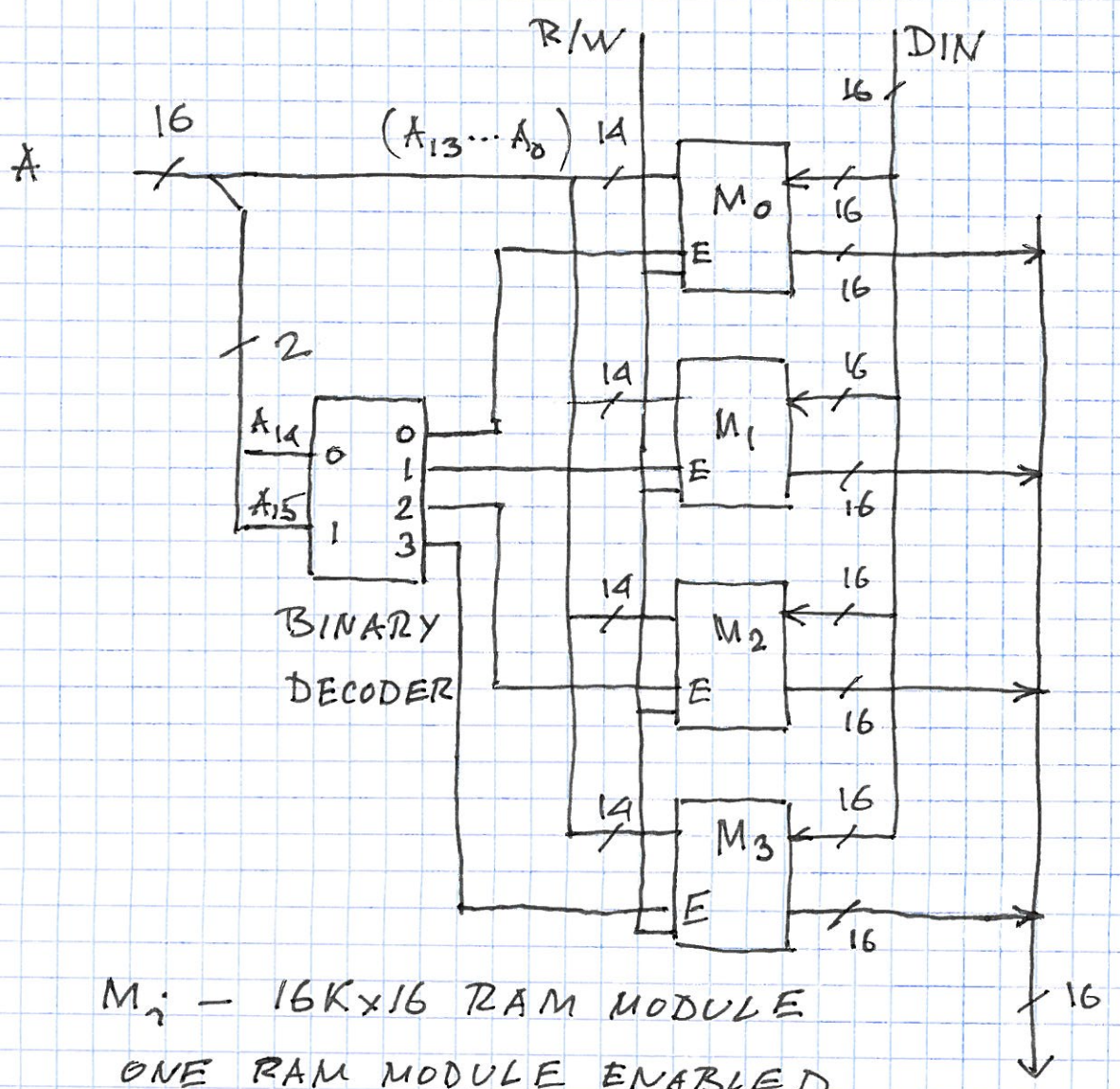
$\rightarrow$ 16K $\times$ 16 RAM

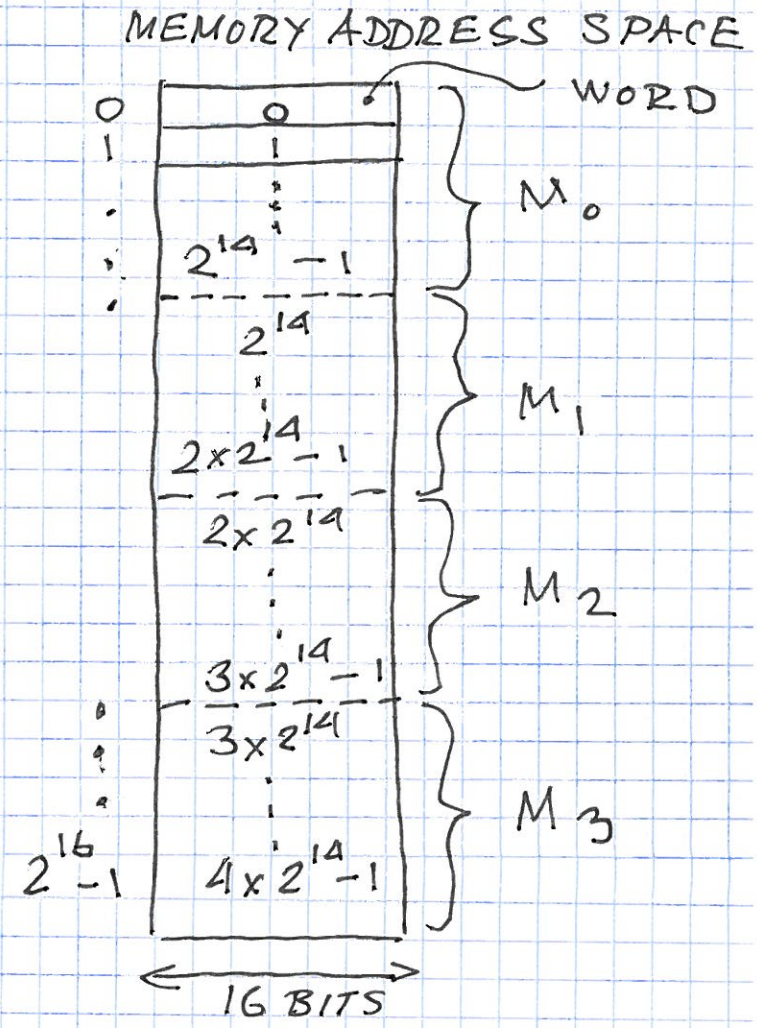
$K = 2^{10}$

CS M51A

MULTIMODULE MEMORY:

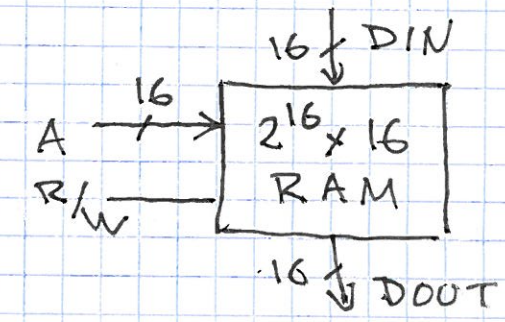
64K x16 USING 16K x16 RAM MODULES





M_i — 16K x16 RAM MODULE

ONE RAM MODULE ENABLED AT A TIME FOR READ OR WRITE OPERATION



BINARY DECODER AND OR GATE

- UNIVERSAL

EXAMPLE 9.5:

$x_2x_1x_0$	z_2	z_1	z_0
000	0	1	0
001	1	0	0
010	0	0	1
011	0	1	0
100	0	0	1
101	1	0	1
110	0	0	0
111	1	0	0

$$
\begin{aligned}
(y_7, \ldots, y_0) &= \mathrm{DEC}(x_2, x_1, x_0, 1) \\
z_2(x_2, x_1, x_0) &= y_1 + y_5 + y_7 \\
z_1(x_2, x_1, x_0) &= y_0 + y_3 \\
z_0(x_2, x_1, x_0) &= y_2 + y_4 + y_5
\end{aligned}
$$

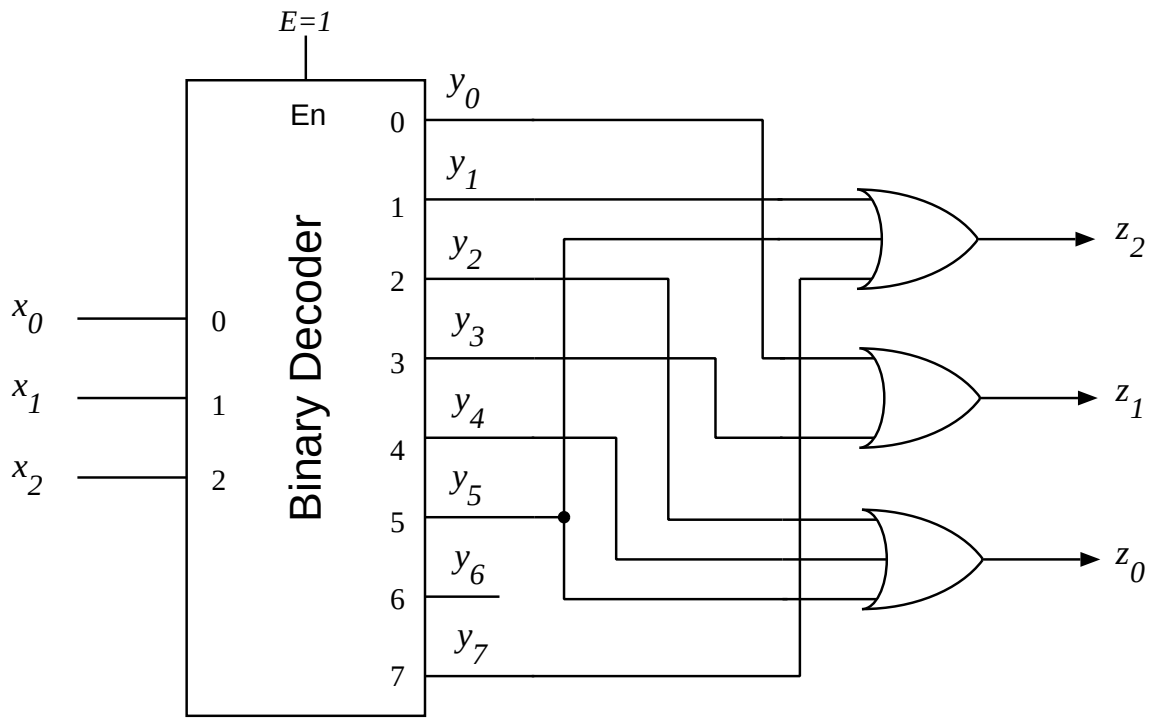
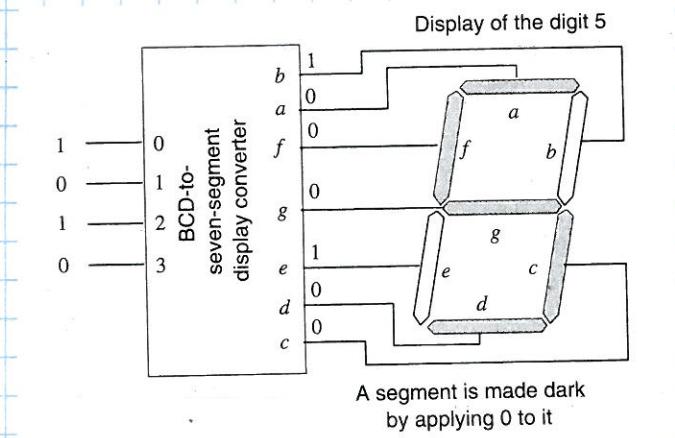


Figure 9.5: NETWORK IN EXAMPLE 9.5

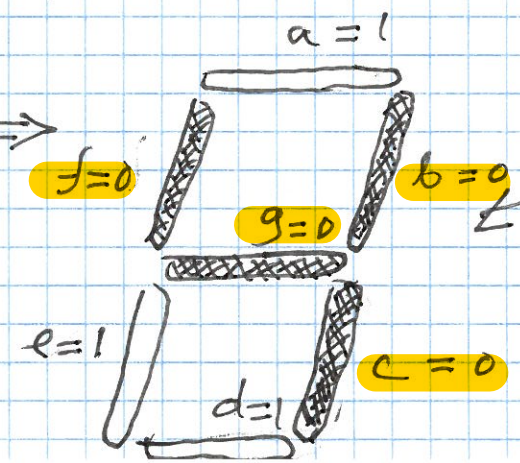
CSM51A BCD TO 7-SEGMENT DECODER AND DISPLAY

0 — SEGMENT DARK



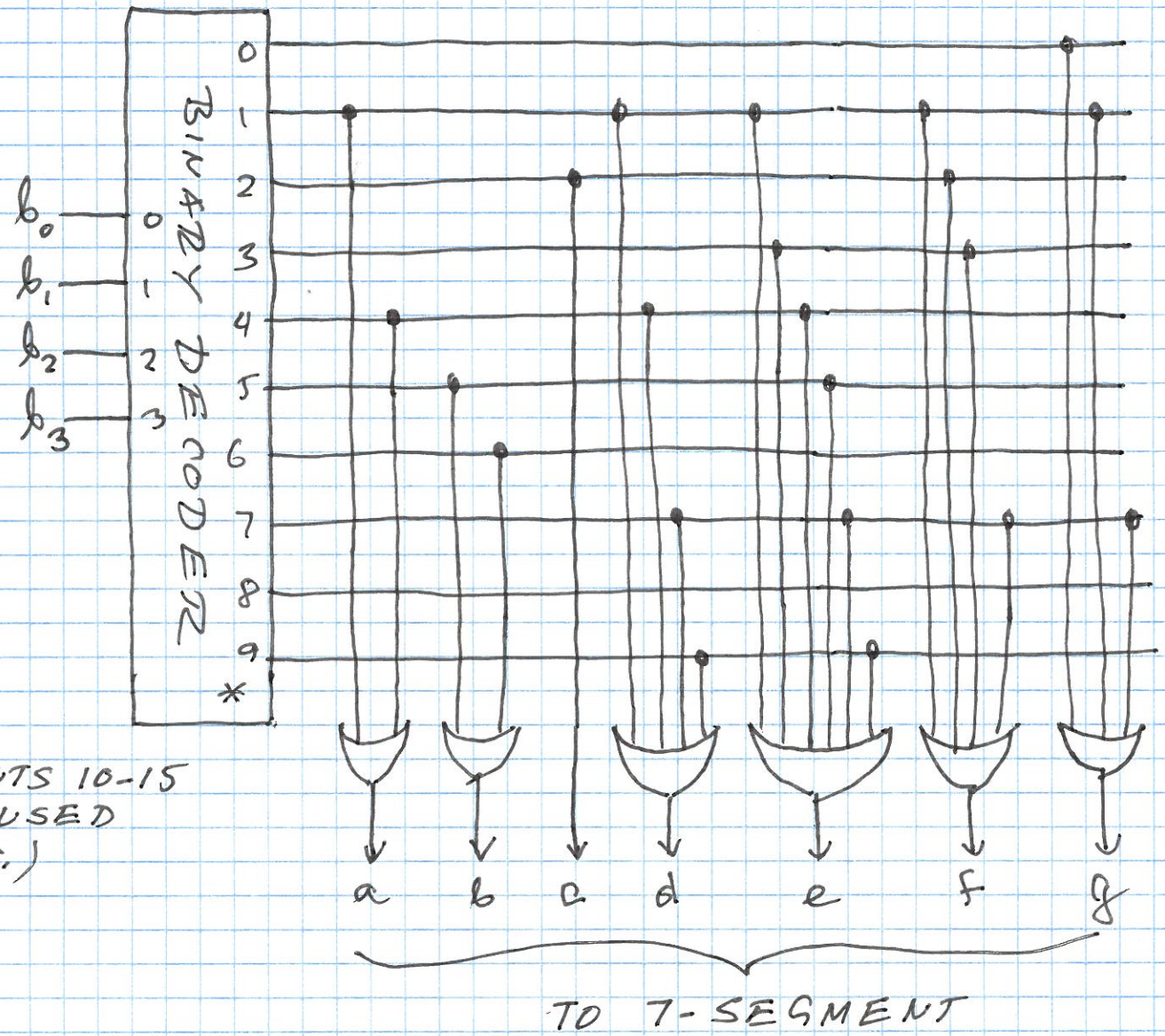
BCD				7-SEGMENT DISPLAY						
b_3	b_2	b_1	b_0	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0
0	0	1	1	0	0	0	0	1	1	0
0	1	0	0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0	1	0	0
0	1	1	0	0	1	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0

"4" ⟹



CS M51A

BCD TO 7-SEGMENT DECODER



* – OUTPUTS 10-15
NOT USED
(D.C.)

CS MSIA

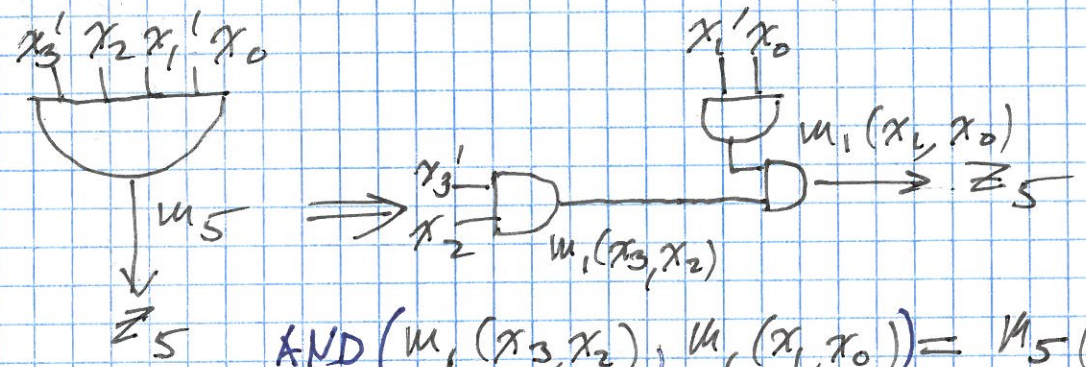
DESIGN OF LARGE DECODERS

$y_i = E \cdot m_i(\underline{x})$ $\quad i = 0 \ldots 2^n - 1 \Rightarrow$ MINTERMS OF n VARS.

FOR LARGE n, FANIN OF AND GATES BECOMES A PROBLEM

SOLUTION: USE ASSOCIATIVITY OF AND TO OBTAIN A SUITABLE DECOMPOSITION

CONSIDER $\quad Z_5 = m_5(x_3, x_2, x_1, x_0)$



$\text{AND}(m_1(x_3, x_2), m_1(x_1, x_0)) = m_5(x_3, x_2, x_1, x_0)$

COINCIDENCE

$x = 5 \Rightarrow Z_5 = 1$

8	4	2	1
0	1	0	1

$m_1 = 1 \quad m_1 = 1$

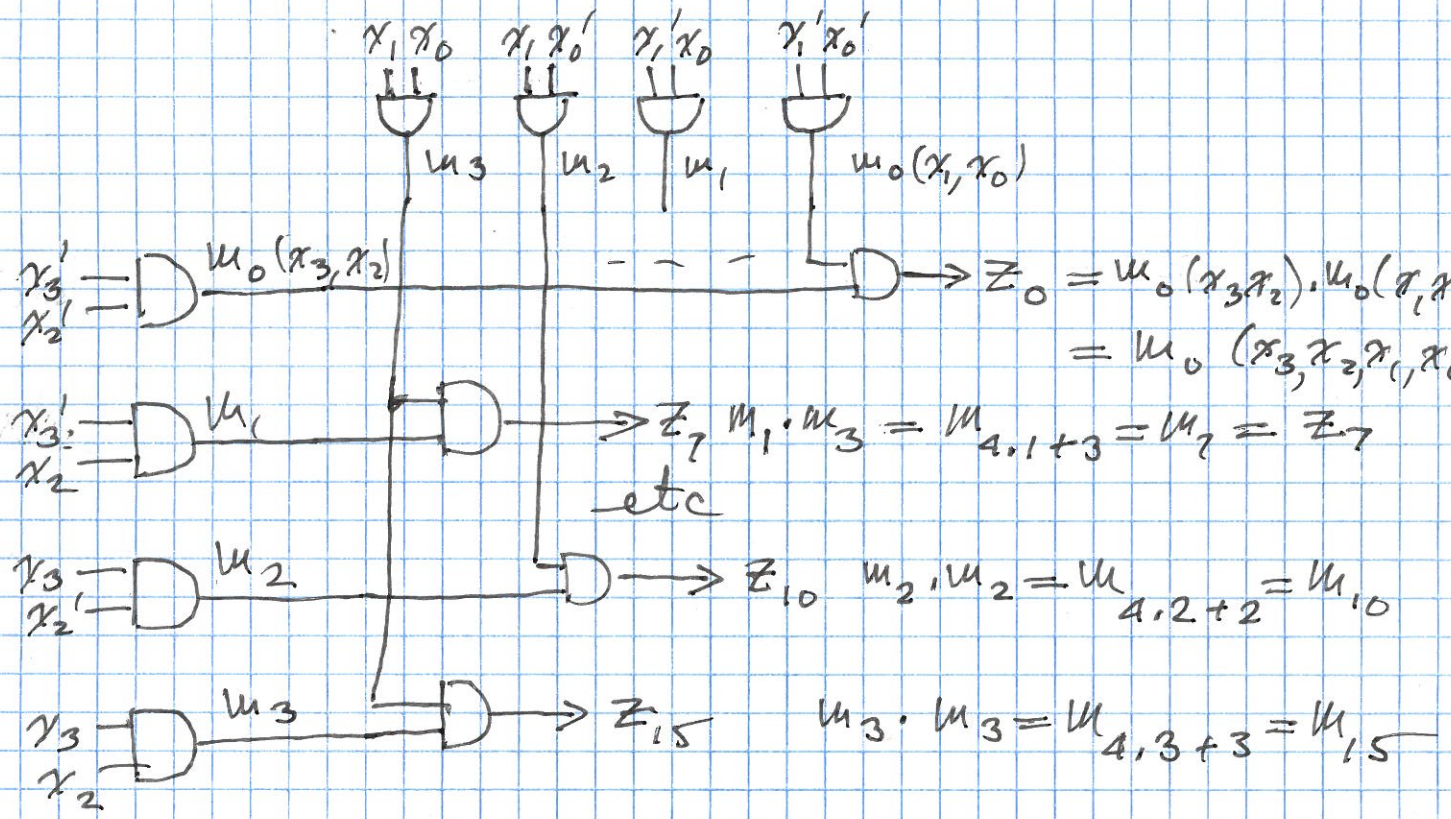
$1 \cdot 4 + 1$

$\Rightarrow m_{4 \cdot 1 + 1} = m_5$

CSM51A

EXAMPLE OF COINCIDENT DECODER

4-TO-16 DECODER USING 2-TO-4 DECODERS AND 2-INPUT AND GATES



CS MSIA

COINCIDENT DECODING

$$x = (x_7\, x_6\, x_5\, x_4\, x_3\, x_2\, x_1\, x_0)$$
$$= ((x_7\, x_6\, x_5\, x_4)(x_3\, x_2\, x_1\, x_0))$$
$$= (x_{LEFT}, x_{RIGHT})$$

$$x = 2^4 \cdot x_{LEFT} + x_{RIGHT}$$

LET $\underline{y} = DEC(\underline{x}_{LEFT})$

$\underline{w} = DEC(\underline{x}_{RIGHT})$

$\underline{z} = DEC(\underline{x})$

$z_i = AND(y_s, w_t)$

$\Rightarrow i = 2^4 s + t$

$x = (\underbrace{0010}\,\underbrace{0100})$ $\quad DEC(\underline{x}) \Rightarrow z_{36} = 1$

$$\left.\begin{array}{l} DEC(\underline{x}_{LEFT}) \Rightarrow y_2 = 1 \\ DEC(\underline{x}_{RIGHT}) \Rightarrow w_4 = 1 \end{array}\right\} \quad i = 2^4 \cdot 2 + 4 = 36$$

$$z_{36} = AND(y_2, w_4)$$

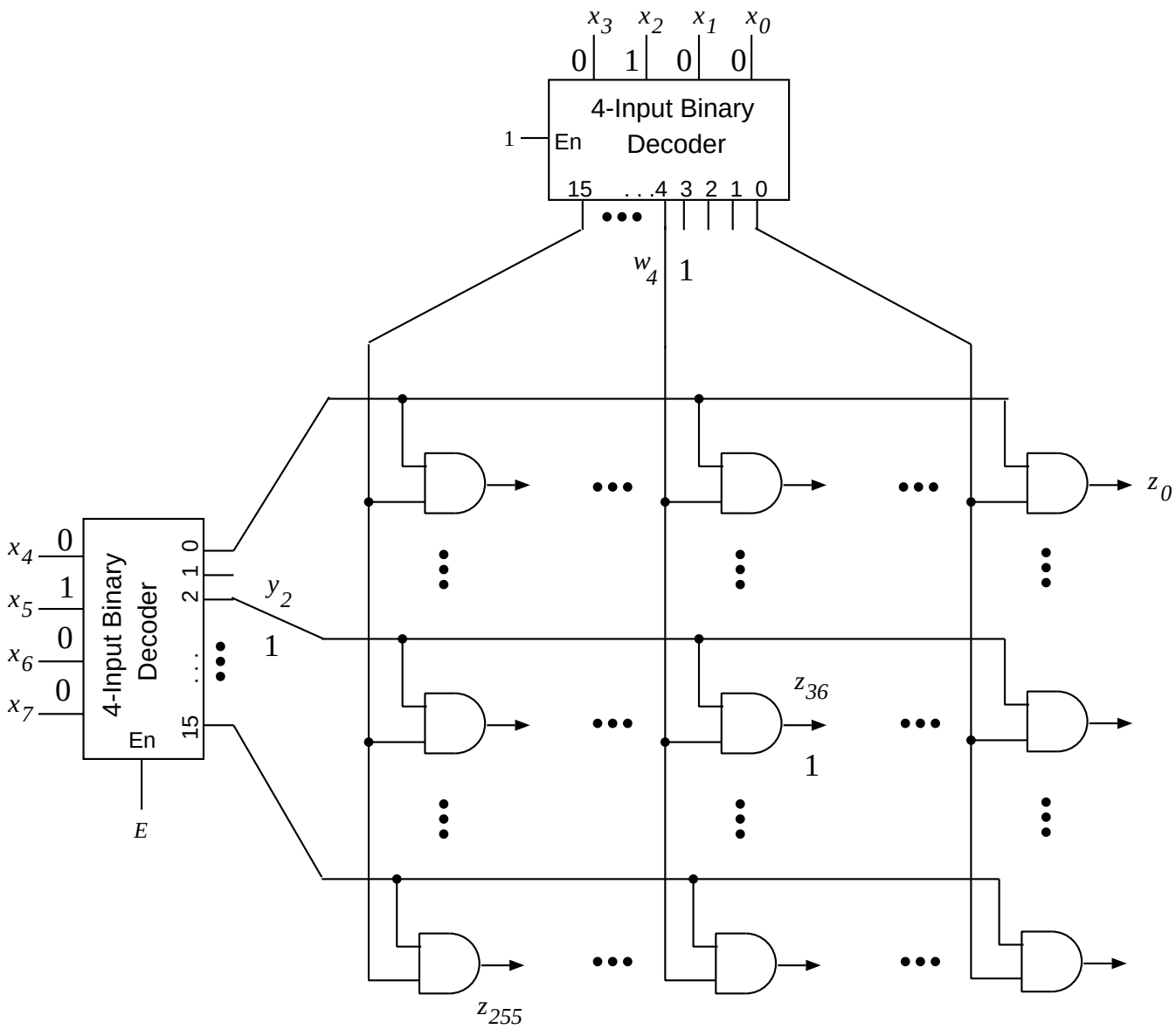


Figure 9.6: 8-INPUT COINCIDENT DECODER.

n-INPUT COINCIDENT DECODER

$$\underline{y} = \mathrm{DEC}(\underline{x}_{\mathrm{left}}, E)$$

$$\underline{w} = \mathrm{DEC}(\underline{x}_{\mathrm{right}}, 1)$$

$$\underline{z} = (\mathrm{AND}(y_{2^{n/2}-1}, w_{2^{n/2}-1}), \ldots, \mathrm{AND}(y_s, w_t), \ldots, \mathrm{AND}(y_0, w_0))$$

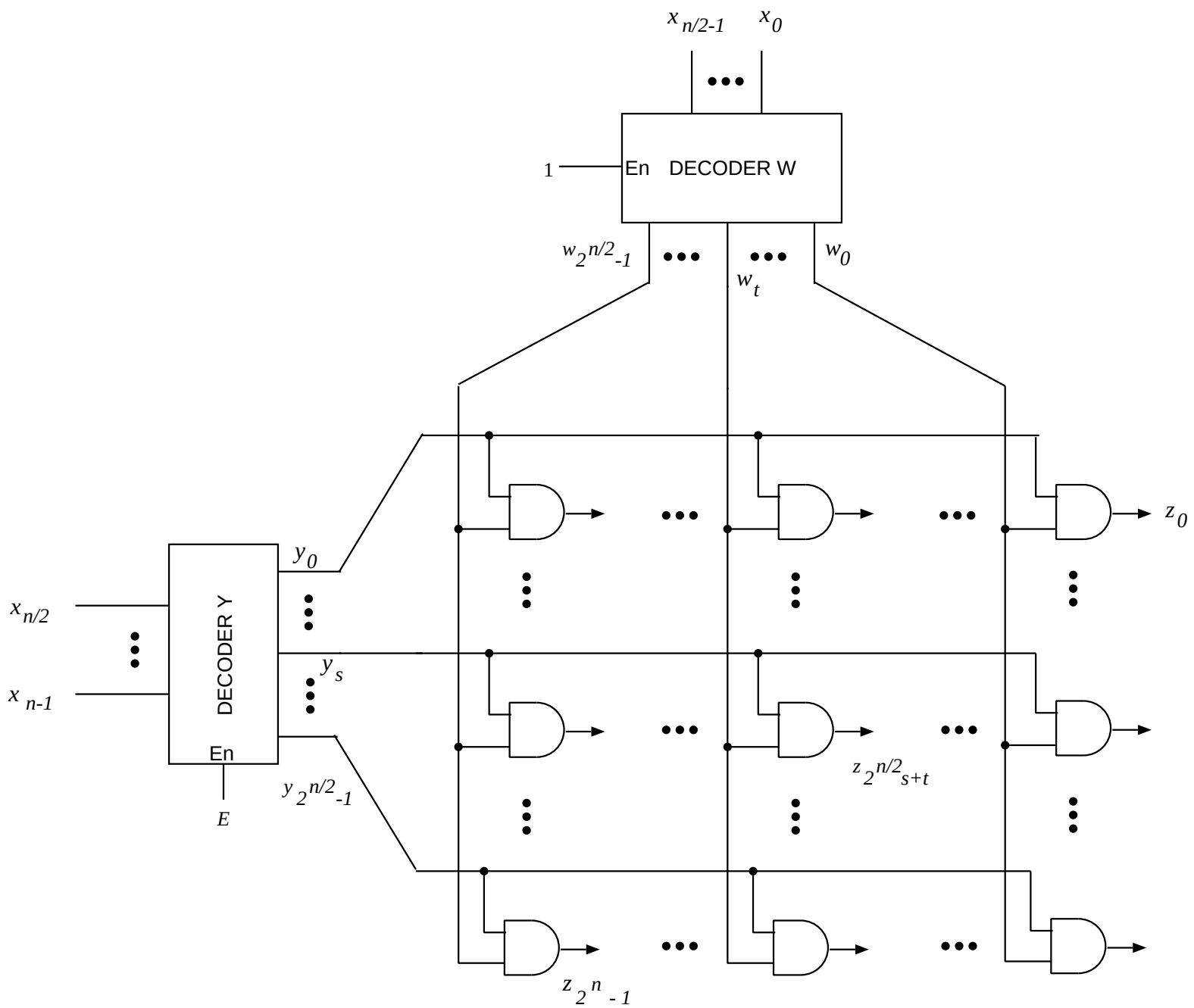


Figure 9.7: n-INPUT COINCIDENT DECODER.

TREE DECODING

$$\underline{x} = (\underline{x}_{\text{left}}, \underline{x}_{\text{right}})$$

$$\underline{x}_{\text{left}} = (x_3, x_2)$$

$$\underline{x}_{\text{right}} = (x_1, x_0)$$

4-INPUT TREE DECODER

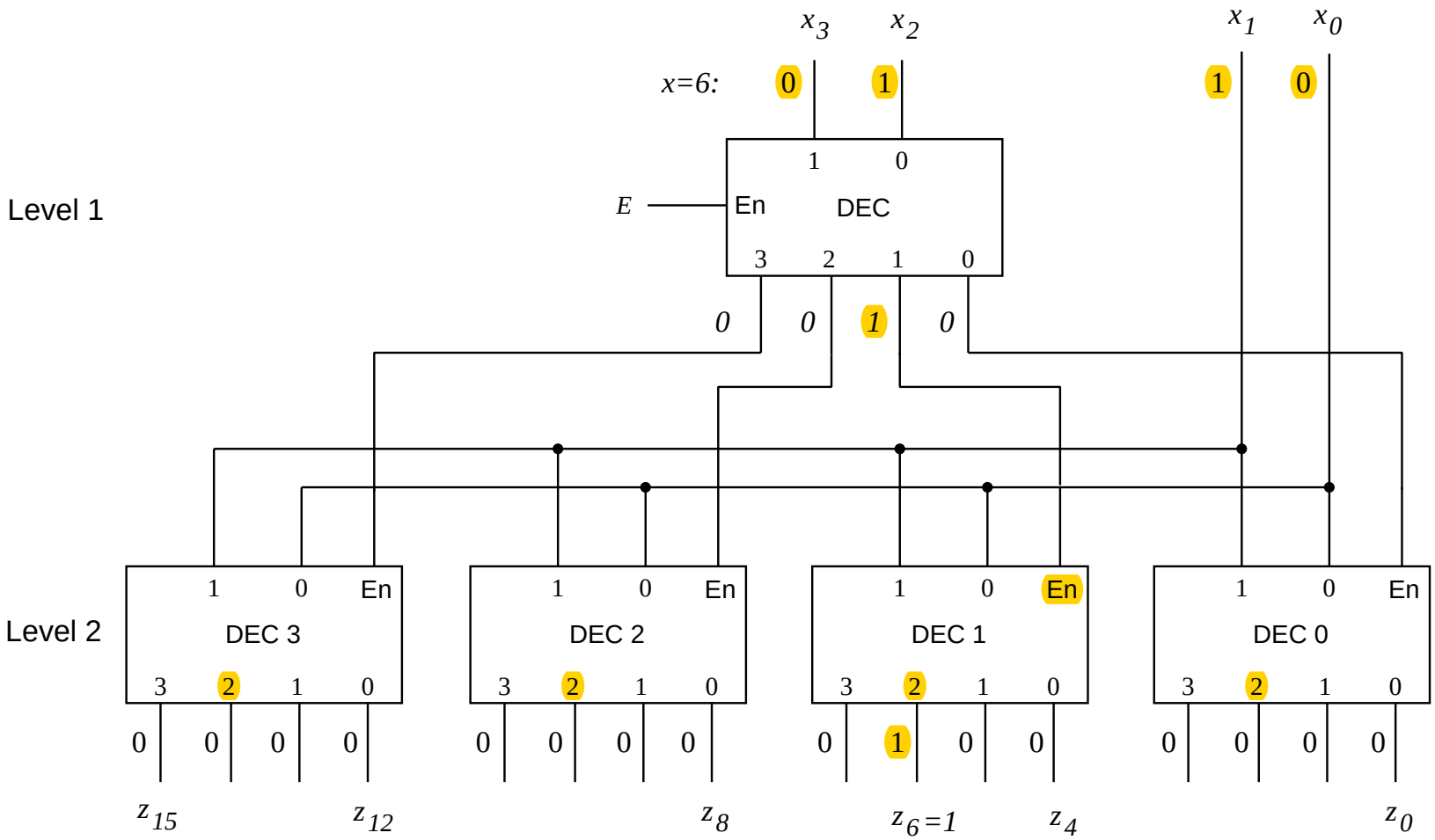


Figure 9.8: 4-INPUT TREE DECODER.

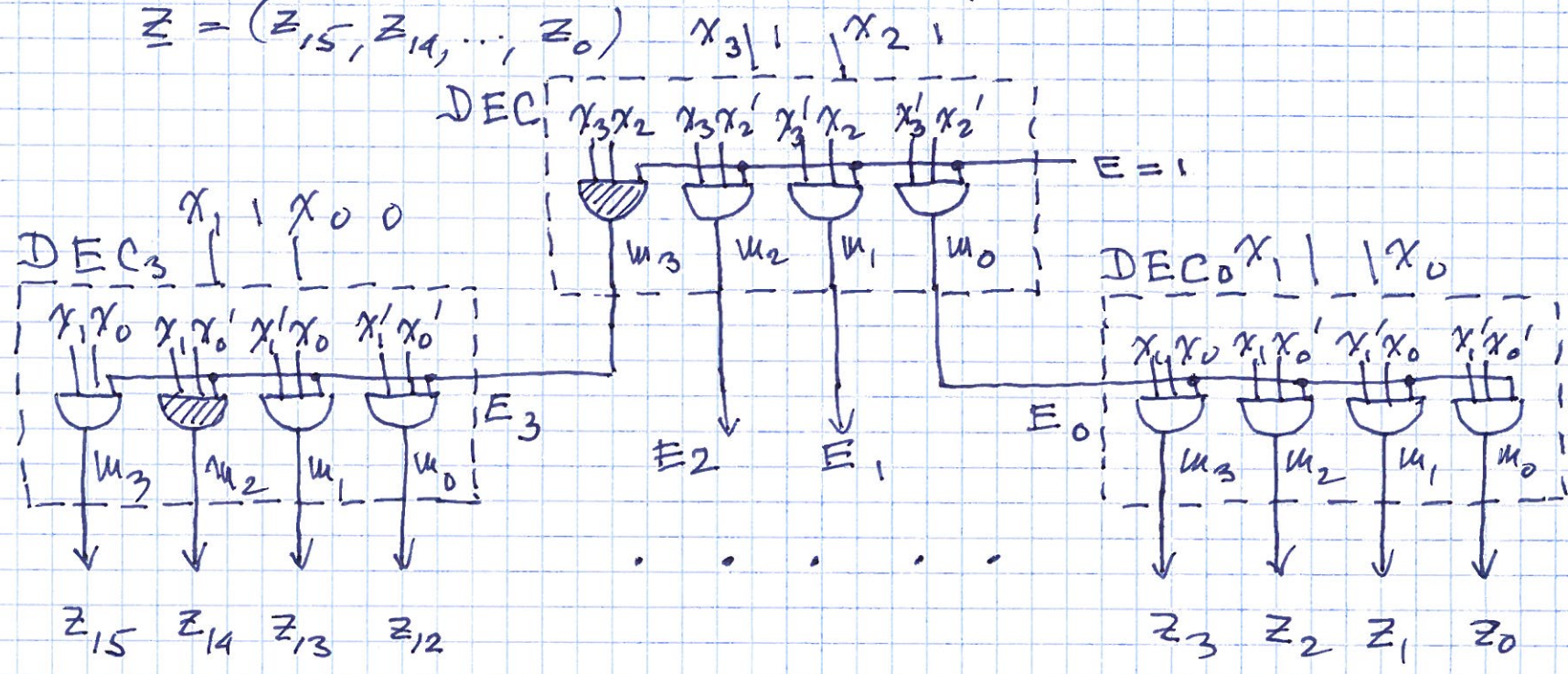
CS M51A

TREE DECODER

4-INPUT, 16-OUTPUT

$\underline{Z} = DEC(\underline{x}, E) \qquad \underline{x} = (x_3, x_2, x_1, x_0)$

$\underline{Z} = (z_{15}, z_{14}, \ldots, z_0)$



$E_i = m_i(x_3, x_2) \qquad i = 0, 1, 2, 3$

$z_k = E_i(x_3, x_2) \cdot m_j(x_1, x_0) = m_i(x_3, x_2) \cdot m_j(x_1, x_0) = m_k(x_3, x_2, x_1, x_0)$

FOR $x = (\underbrace{1\ 1}_{3}\ \underbrace{1\ 0}_{2}) \Rightarrow k = 2^2 \cdot i + j$

$k = 4 \cdot 3 + 2 = 14 \qquad DEC(1110, 1) \Rightarrow z_{14} = 1$

n-INPUT TREE DECODERr

$$\underline{w} = \mathrm{DEC}(\underline{x}_{\mathrm{left}}, E)$$

$$\underline{z} = (\mathrm{DEC}(\underline{x}_{\mathrm{right}}, w_{2^{n/2}-1}), \ldots, \mathrm{DEC}(\underline{x}_{\mathrm{right}}, w_t), \ldots, \mathrm{DEC}(\underline{x}_{\mathrm{right}}, w_0))$$

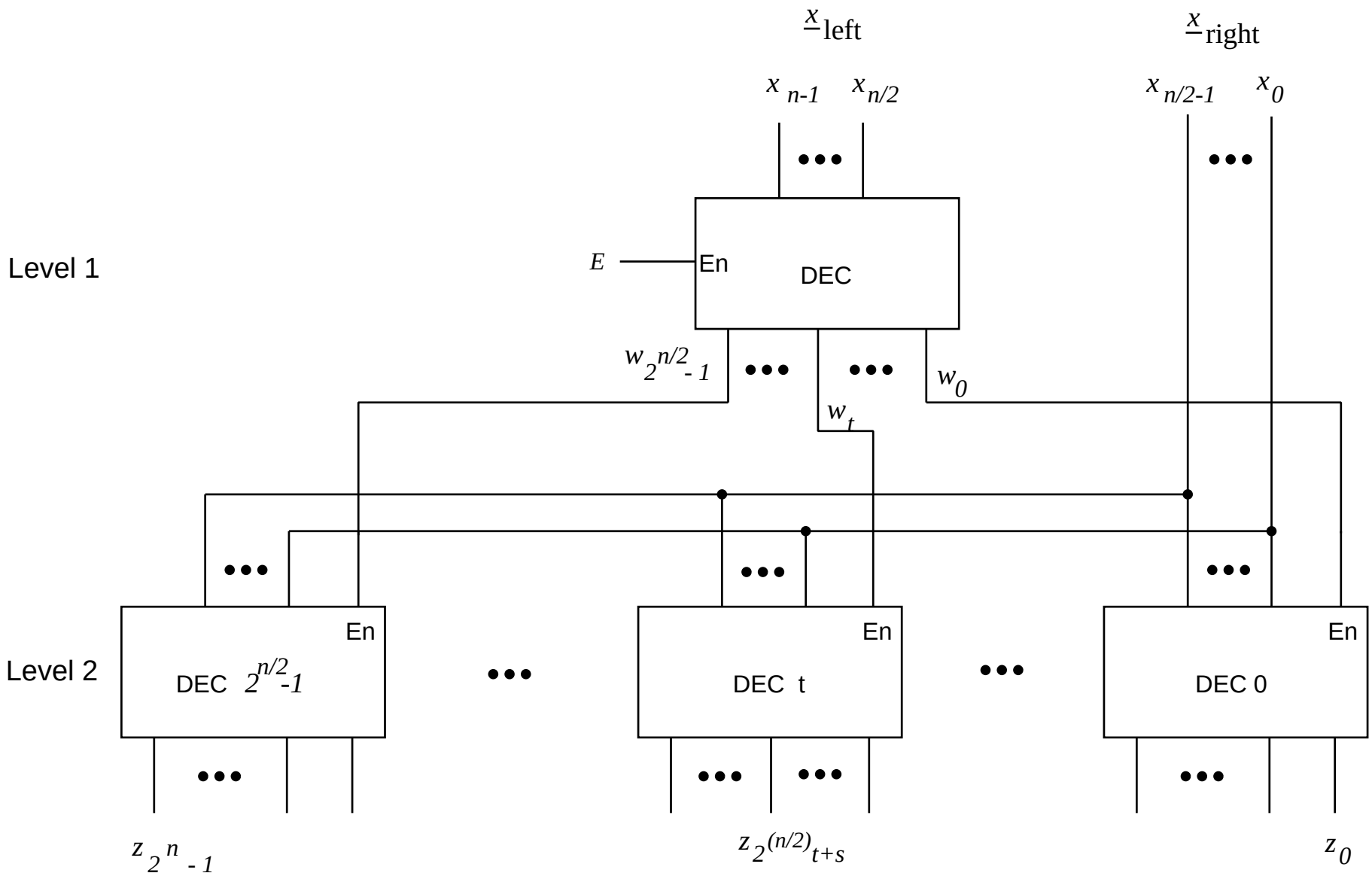


Figure 9.9: n-INPUT TWO-LEVEL TREE DECODER.

COMPARISON OF DECODER NETWORKS

	Coincident	Tree
Decoder modules	2	$2^k + 1$
AND gates	2^{2k}	–
Load per network input	1 decoder input	2^k decoder inputs (max)
Fanout per decoder output	2^k AND inputs	1 enable input
Number of module inputs (related to number of connections)	$2k + 2 + 2^{2k+1}$	$1 + k + 2^k + k2^k$
Delay	$t_{\text{decoder}} + t_{\text{AND}}$	$2t_{\text{decoder}}$

EXAMPLE 9.6: 6-INPUT DECODER

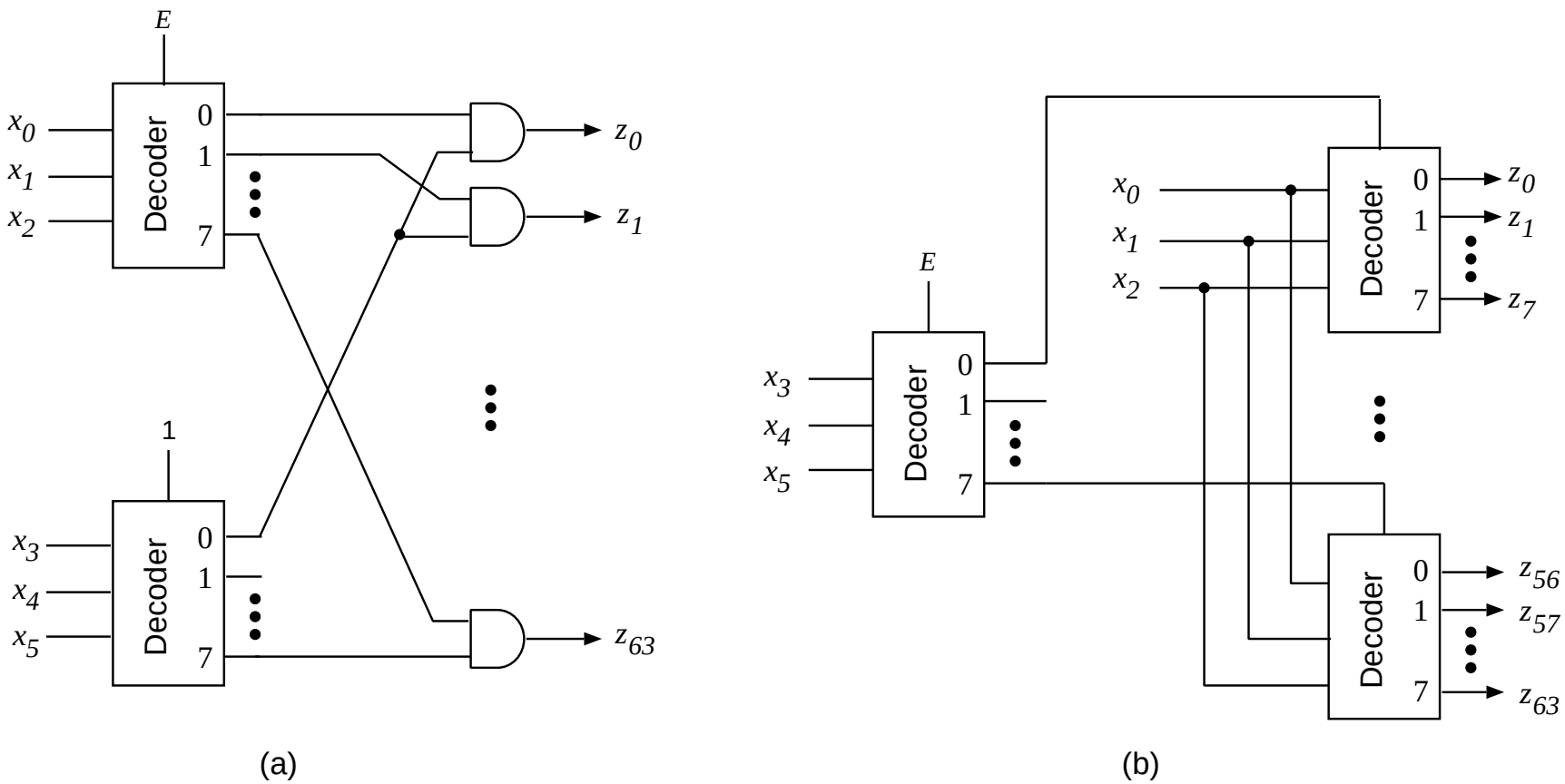
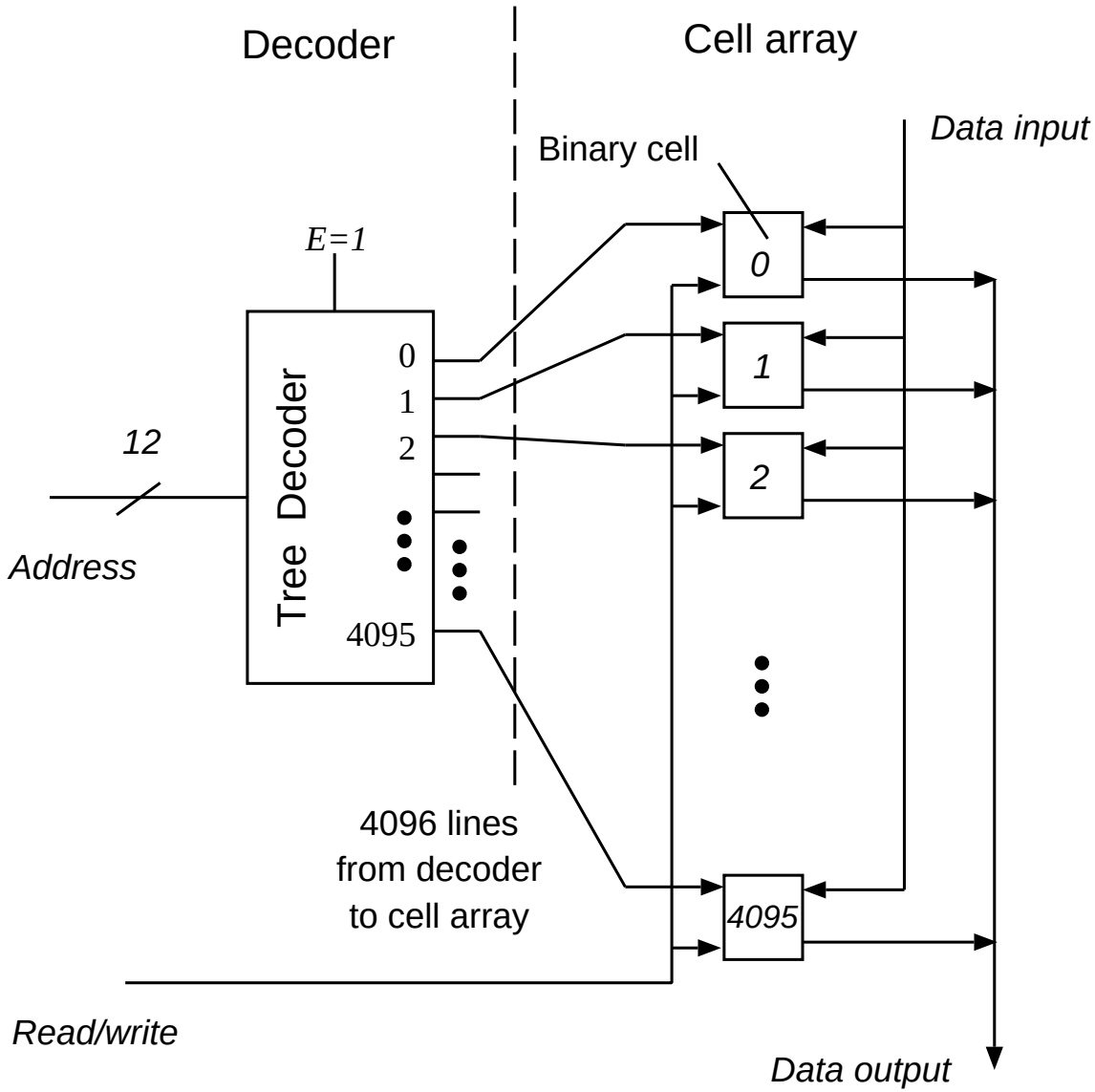


Figure 9.10: IMPLEMENTATION OF 6-INPUT DECODER. a) COINCIDENT DECODER. b) TREE DECODER.

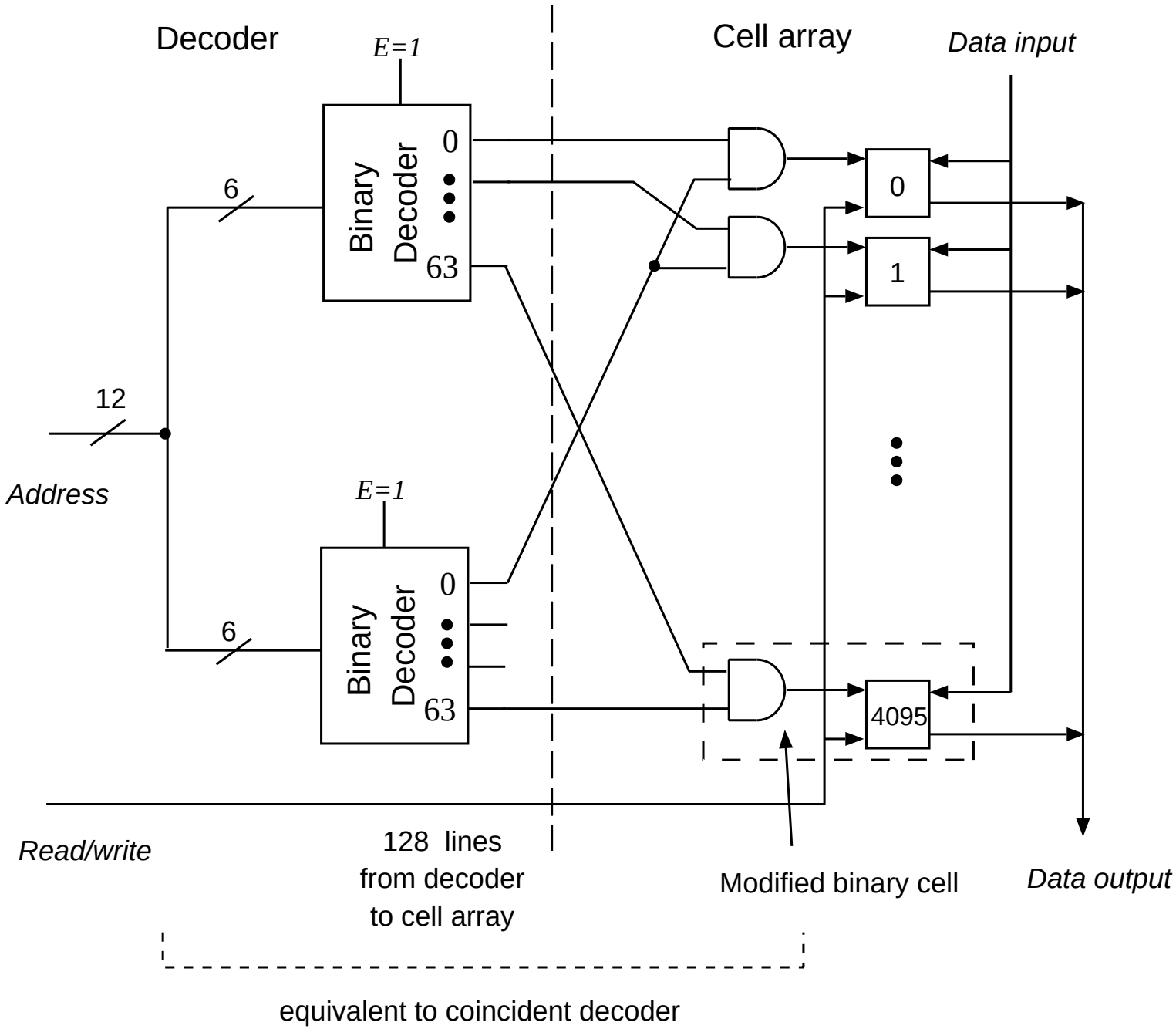
EXAMPLE 9.6 (cont.)

	Coincident	Tree
Decoder modules	2	9
AND gates	64	–
Load per network input	1 decoder input	8 decoder inputs (max)
Fanout per decoder output	8 AND inputs	1 enable input
Number of module inputs	136	36
Delay	$3d$	$4d$



(a)

Figure 9.11: a) SYSTEM WITH TREE DECODER.



(b)

Figure 9.11: b) SYSTEM WITH COINCIDENT DECODER.

BINARY ENCODERS

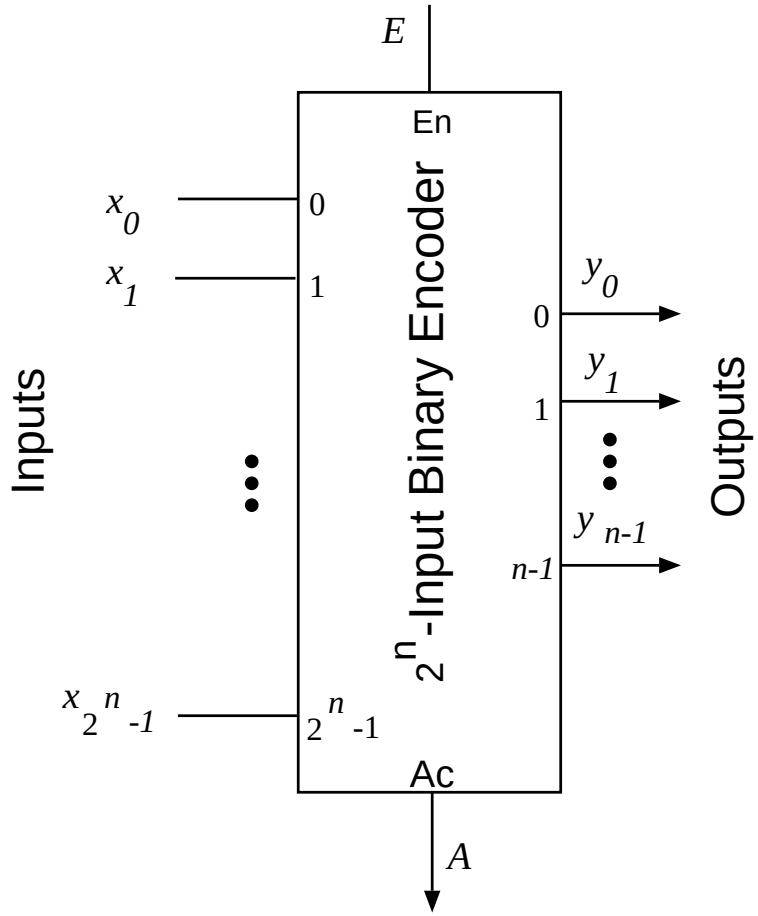


Figure 9.12: 2^n-INPUT BINARY ENCODER.

BINARY ENCODER: HIGH-LEVEL SPECIFICATION

INPUTS: $\underline{x} = (x_{2^n-1}, \ldots, x_0), \quad x_i \in \{0,1\}$, with at most one $x_i = E \in \{0,1\}$

OUTPUTS: $\underline{y} = (y_{n-1}, \ldots, y_0), \quad y_j \in \{0,1\}$
$A \in \{0,1\}$

FUNCTION: $y = \begin{cases} i & \textbf{if} \quad (x_i = 1) \ \textbf{and} \ (E = 1) \\ 0 & \textbf{otherwise} \end{cases}$

$A = \begin{cases} 1 & \textbf{if} \quad (\text{some } x_i = 1) \ \textbf{and} \ (E = 1) \\ 0 & \textbf{otherwise} \end{cases}$

$$y = \sum_{j=0}^{n-1} y_j 2^j, \qquad i = 0, \ldots, 2^n - 1$$

EXAMPLE 9.7: FUNCTION OF AN 8-INPUT BINARY ENCODER

E	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	y	y_2	y_1	y_0	A
1	0	0	0	0	0	0	0	1	0	0	0	0	1
1	0	0	0	0	0	0	1	0	1	0	0	1	1
1	0	0	0	0	0	1	0	0	2	0	1	0	1
1	0	0	0	0	1	0	0	0	3	0	1	1	1
1	0	0	0	1	0	0	0	0	4	1	0	0	1
1	0	0	1	0	0	0	0	0	5	1	0	1	1
1	0	1	0	0	0	0	0	0	6	1	1	0	1
1	1	0	0	0	0	0	0	0	7	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	-	-	-	-	-	-	-	-	0	0	0	0	0

BINARY SPECIFICATION OF ENCODER

INPUTS: $\underline{x} = (x_{2^n-1}, \ldots, x_0), \quad x_i \in \{0, 1\}$

, with at most one $x_i = 1$

$E \in \{0, 1\}$

OUTPUTS: $\underline{y} = (y_{n-1}, \ldots, y_0), \quad y_j \in \{0, 1\}$

$A \in \{0, 1\}$

FUNCTION: $y_j = E \cdot \Sigma(x_k), \quad j = 0, \ldots, n-1$

$A = E \cdot \Sigma(x_i), \quad i = 0, \ldots, 2^n - 1$

EXAMPLE 9.8

$$y_0 = E \cdot (x_1 + x_3 + x_5 + x_7)$$

$$y_1 = E \cdot (x_2 + x_3 + x_6 + x_7)$$

$$y_2 = E \cdot (x_4 + x_5 + x_6 + x_7)$$

$$A = E \cdot (x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7)$$

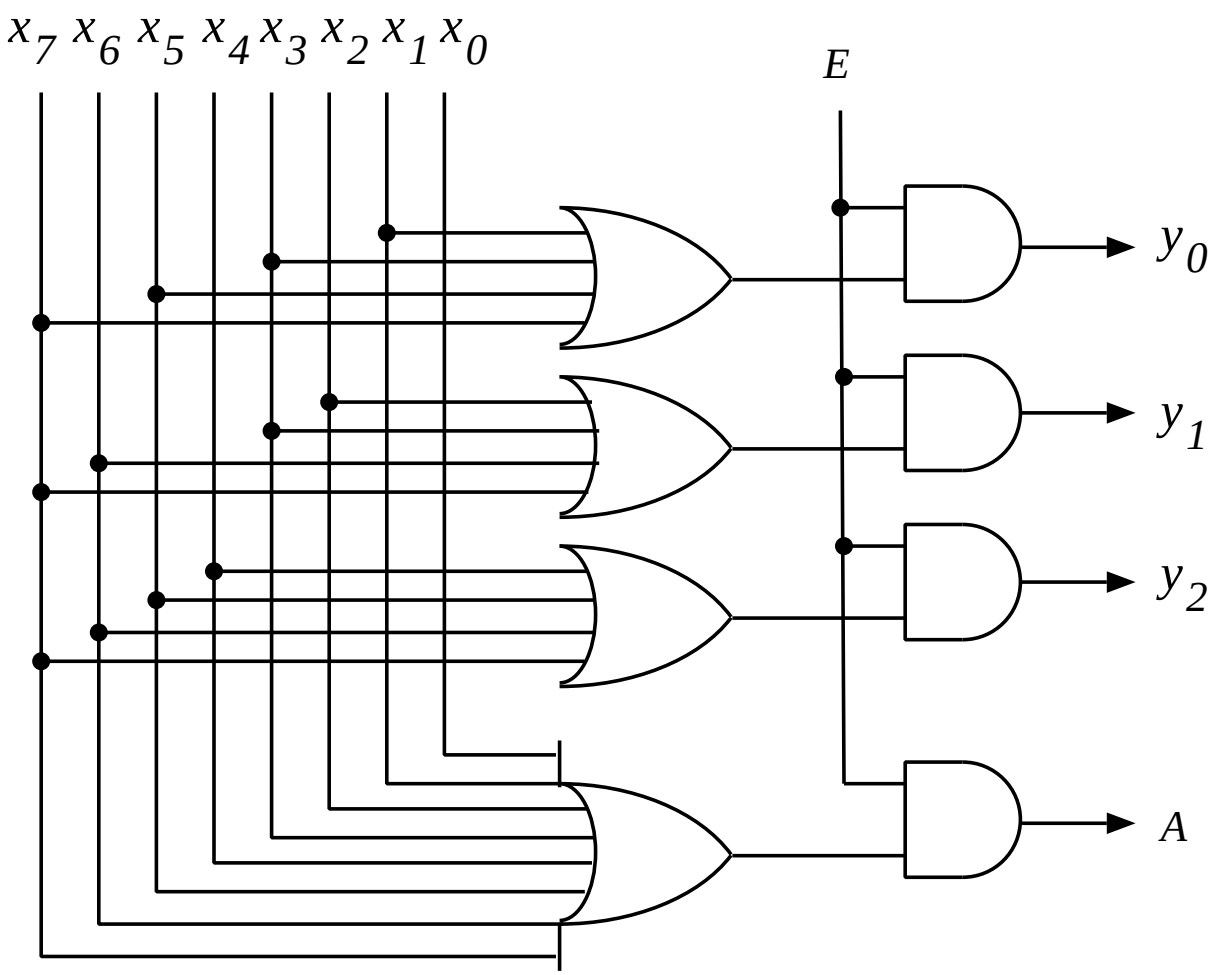


Figure 9.13: IMPLEMENTATION OF AN 8-INPUT BINARY ENCODER.

USES OF BINARY ENCODERS

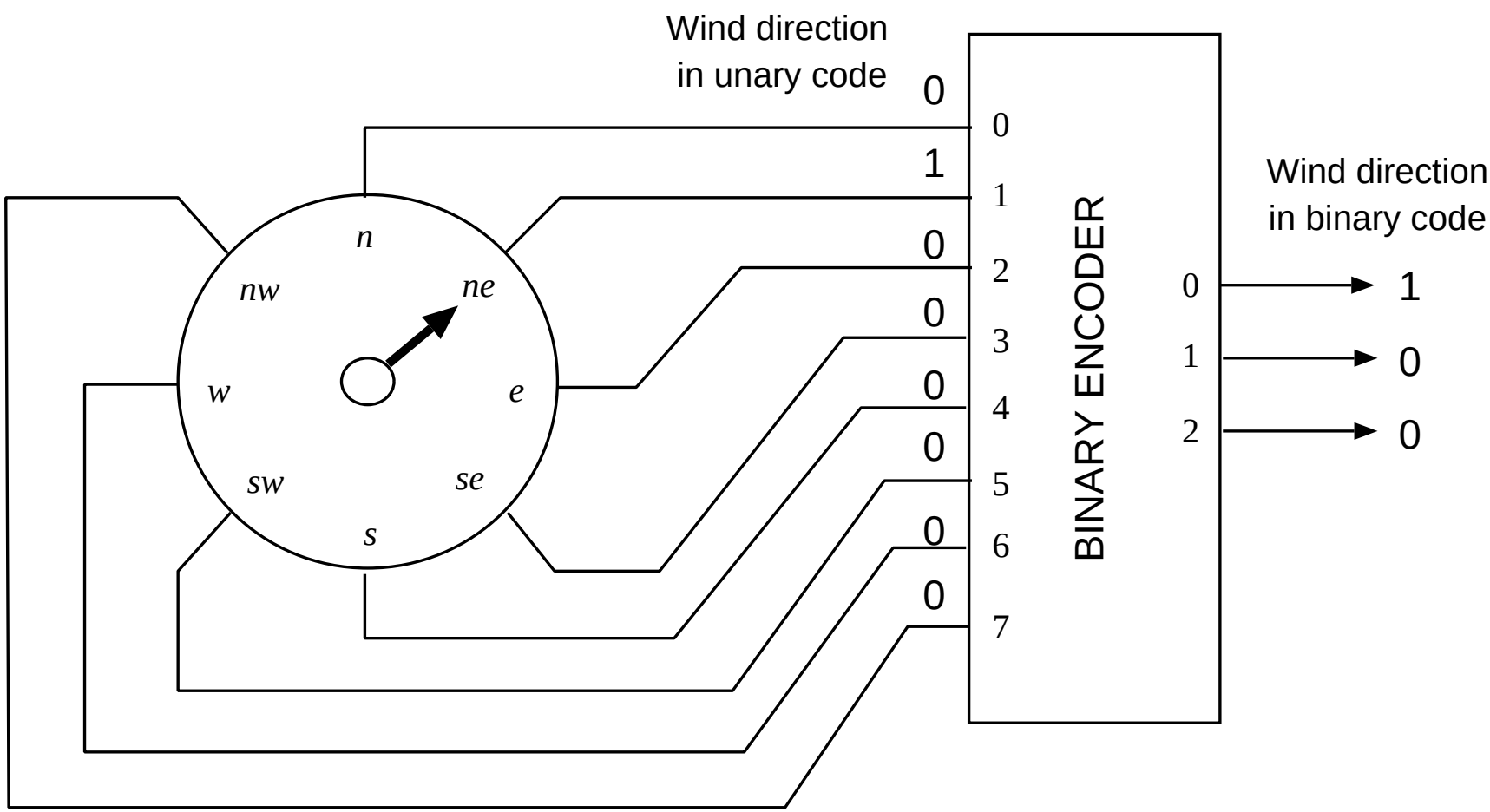


Figure 9.14: WIND DIRECTION ENCODER.

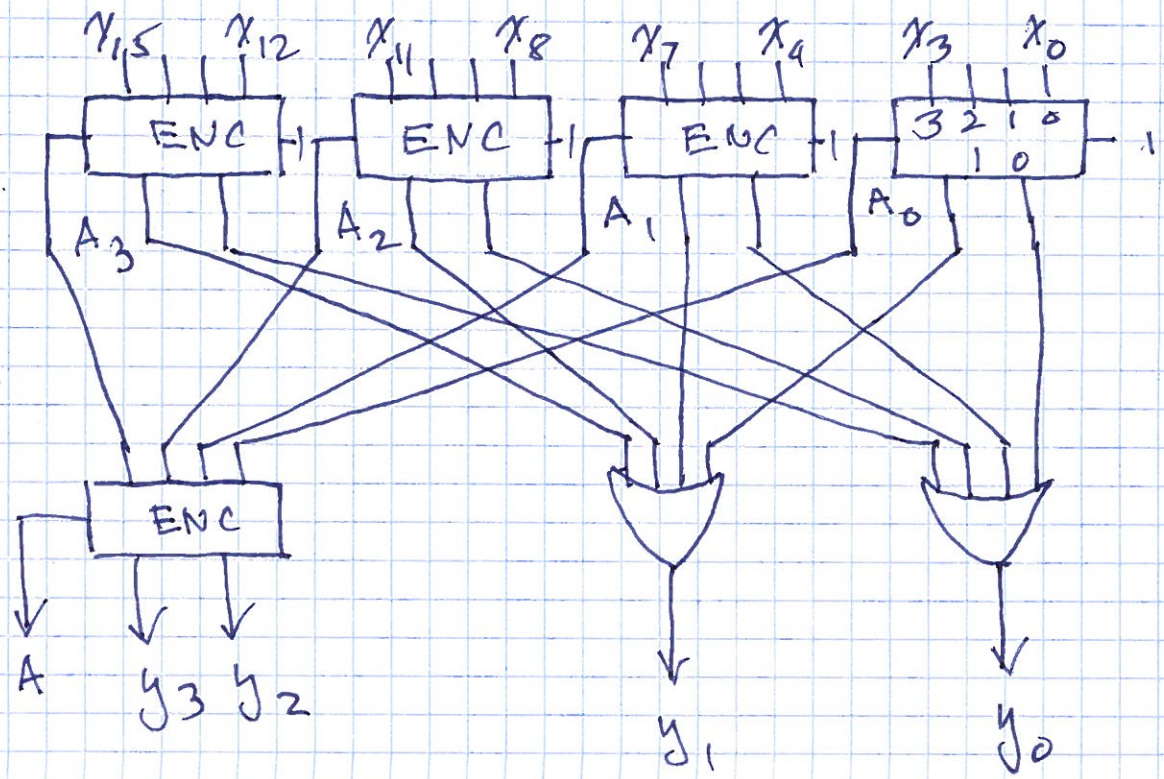
CSM51A

ENCODER NETWORKS

$\underline{x} = (x_{15}, x_{14}, \ldots, x_1, x_0)$ WITH AT MOST ONE $x_i = 1$

$\underline{y} = (y_3, y_2, y_1, y_0)$, A

	y_3	y_2	y_1	y_0
x_0	0	0	0	0
x_1			0	1
x_2			1	0
x_3			1	1
x_4	0	1	0	0
x_5			0	1
x_6			1	0
x_7			1	1
x_8	1	0	0	0
x_9			0	1
x_{10}			1	0
x_{11}			1	1
x_{12}	1	1	0	0
x_{13}			0	1
x_{14}			1	0
x_{15}			1	1



PRIORITY ENCODERS: HIGH-LEVEL DESCRIPTION

INPUTS: $\underline{x} = (x_{2^n-1}, \ldots, x_0), \quad x_i \in \{0, 1\}$

OUTPUTS: $\underline{y} = (y_{n-1}, \ldots, y_0), \quad y_j \in \{0, 1\}$

FUNCTION:

$$y = \begin{cases} i \ \textbf{if} \ \ (x_i = 1) \ \textbf{and} \ (x_k = 0, \ k > i) \ \textbf{and} \ (E = 1) \\ 0 \ \textbf{otherwise} \end{cases}$$

$$A = \begin{cases} 1 \ \textbf{if} \ \ (\text{some } x_i = 1) \ \textbf{and} \ (E = 1) \\ 0 \ \textbf{otherwise} \end{cases}$$

$$y = \sum_{j=0}^{n-1} y_j 2^j, \qquad i, k \in \{0, 1, \ldots, 2^n - 1\}$$

8-INPUT PRIORITY ENCODER

E	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	y_2	y_1	y_0	A
1	0	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	0	0	0	1	-	0	0	1	1
1	0	0	0	0	0	1	-	-	0	1	0	1
1	0	0	0	0	1	-	-	-	0	1	1	1
1	0	0	0	1	-	-	-	-	1	0	0	1
1	0	0	1	-	-	-	-	-	1	0	1	1
1	0	1	-	-	-	-	-	-	1	1	0	1
1	1	-	-	-	-	-	-	-	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0
0	-	-	-	-	-	-	-	-	0	0	0	0

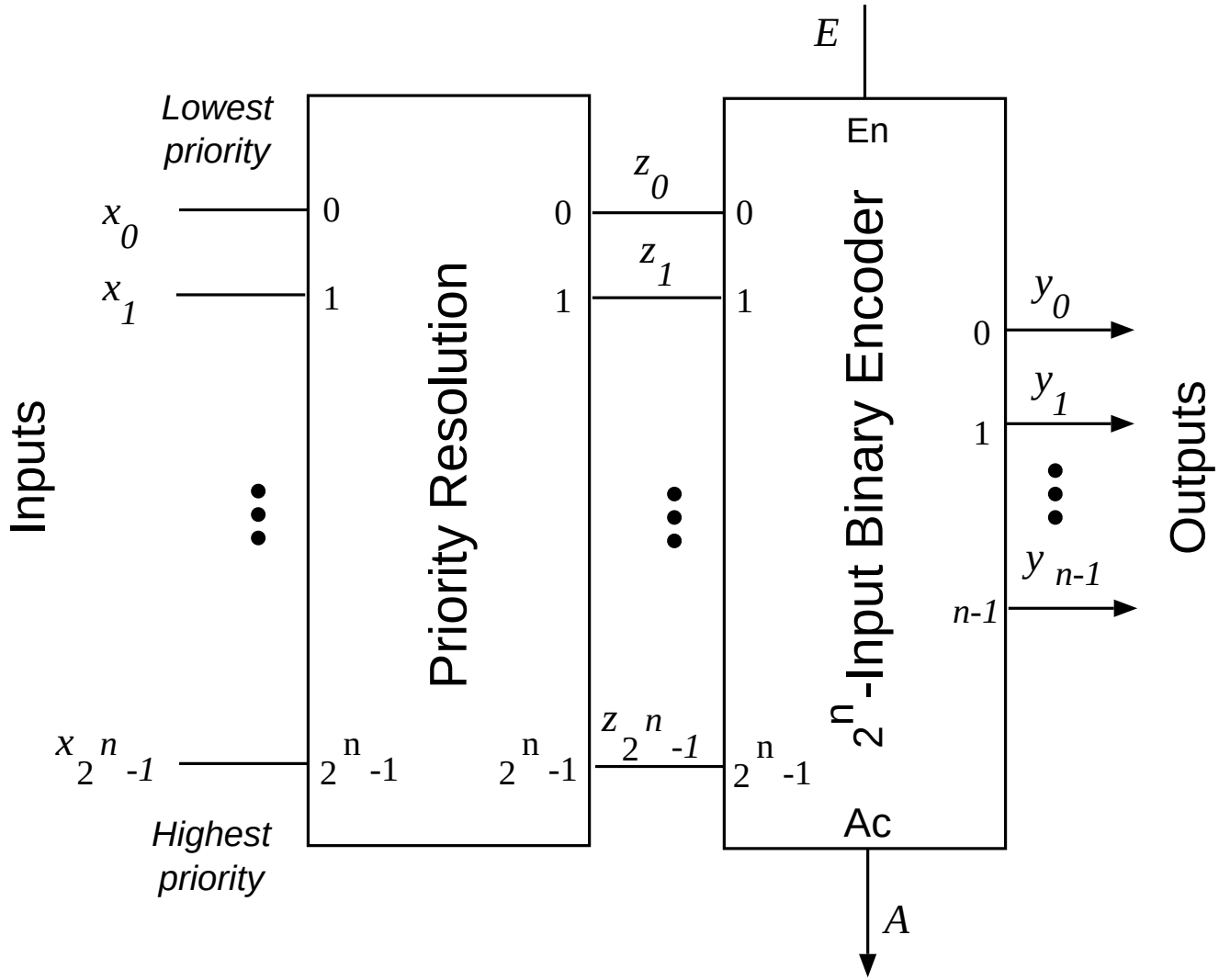


Figure 9.15: PRIORITY ENCODER.

PRIORITY RESOLUTION: HIGH-LEVEL AND BINARY-LEVEL DESCRIPTION

Inputs: $\underline{x} = (x_{2^n-1}, \ldots, x_0), \quad x_i \in \{0, 1\}$
Outputs: $\underline{z} = (z_{2^n-1}, \ldots, z_0), \quad z_i \in \{0, 1\}$

Function: $z_i = \begin{cases} 1 \ \textbf{if} \ \ (x_i = 1) \ \textbf{and} \ (x_k = 0, \quad k > i) \\ 0 \ \textbf{otherwise} \end{cases}$
with $i, k = 0, 1, \ldots, 2^n - 1$

- BINARY DESCRIPTION:

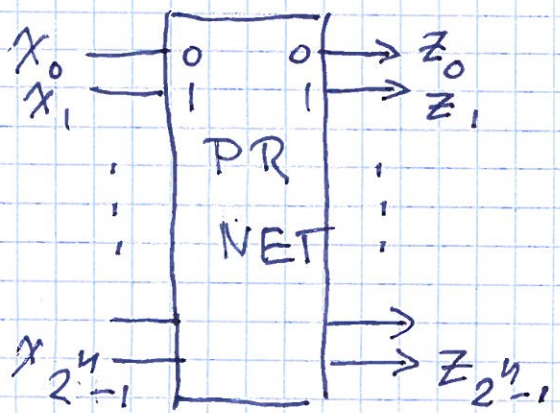
$$z_i = x'_{2^n-1} x'_{2^n-2} \ldots x'_{i+1} x_i \quad , \quad i = 0, 1, \ldots, 2^n - 1$$

OR ITERATIVELY

$$\begin{aligned} c_{i-1} &= c_i + x_i \\ z_i &= c'_i x_i \end{aligned}$$

CS MSIA

ITERATIVE NETWORK FOR PRIORITY RESOLUTION



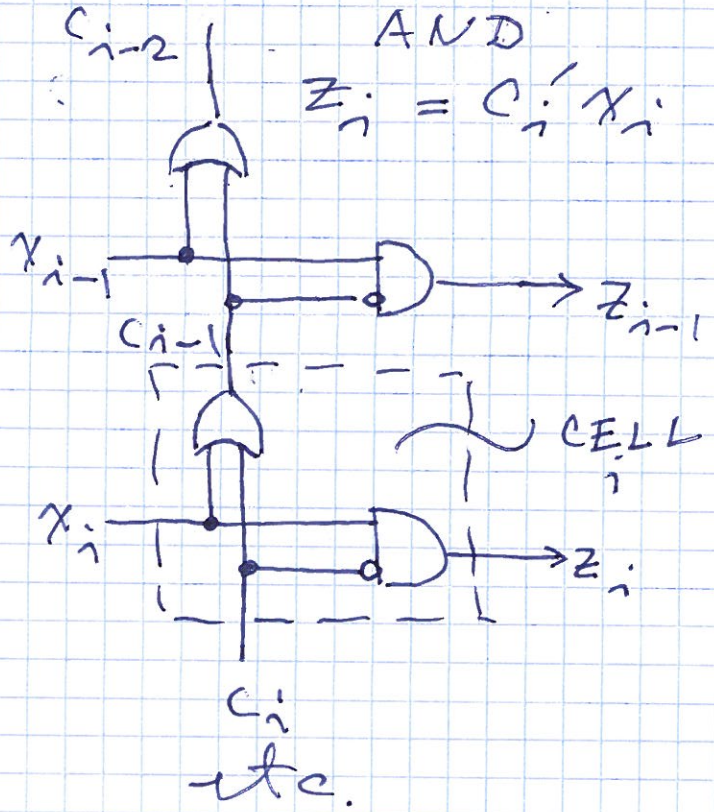
$$z_i = \underbrace{x'_{2^n-1}\, x'_{2^n-2} \cdots x'_{i+1}}_{c'_i}\, x_i = c'_i\, x_i$$

$$z_{i-1} = \underbrace{\overbrace{x'_{2^n-1}\, x'_{2^n-2} \cdots x'_{i+1}}^{c'_i}\, x'_i}_{c'_{i-1}}\, x_{i-1} = c'_{i-1}\, x_{i-1}$$

etc. $c'_{i-1} = c'_i\, x'_i$ OR $c_{i-1} = c_i + x_i$

AND

$z_i = c'_i\, x_i$



DELAY: (WORST CASE)

$$T = (n-1)\, t_{OR} + t_{NOT} + t_{AND}$$

(LINEAR IN n)

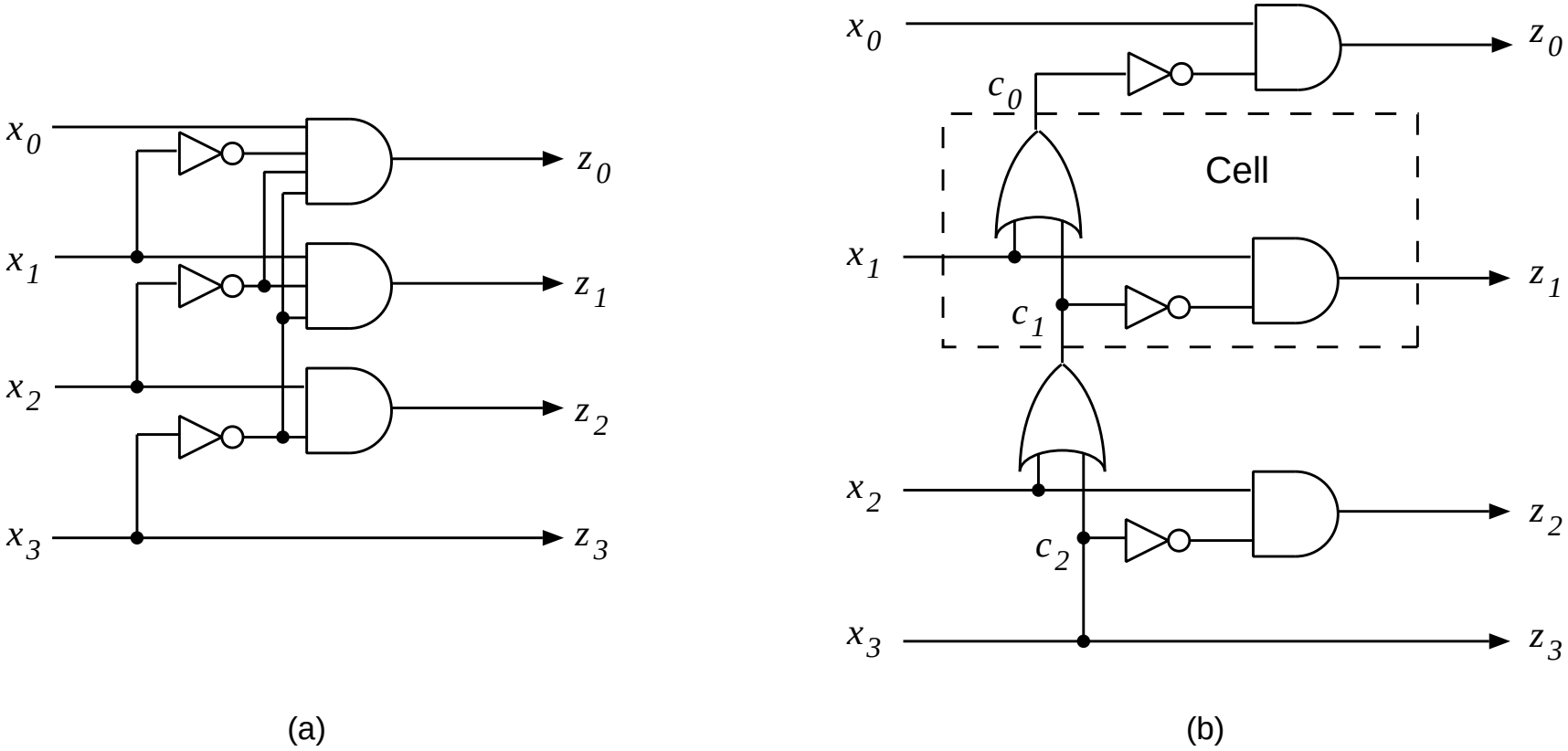


Figure 9.16: 4-BIT PRIORITY RESOLUTION NETWORKS: a) PARALLEL; b) ITERATIVE.

USES OF PRIORITY ENCODERS

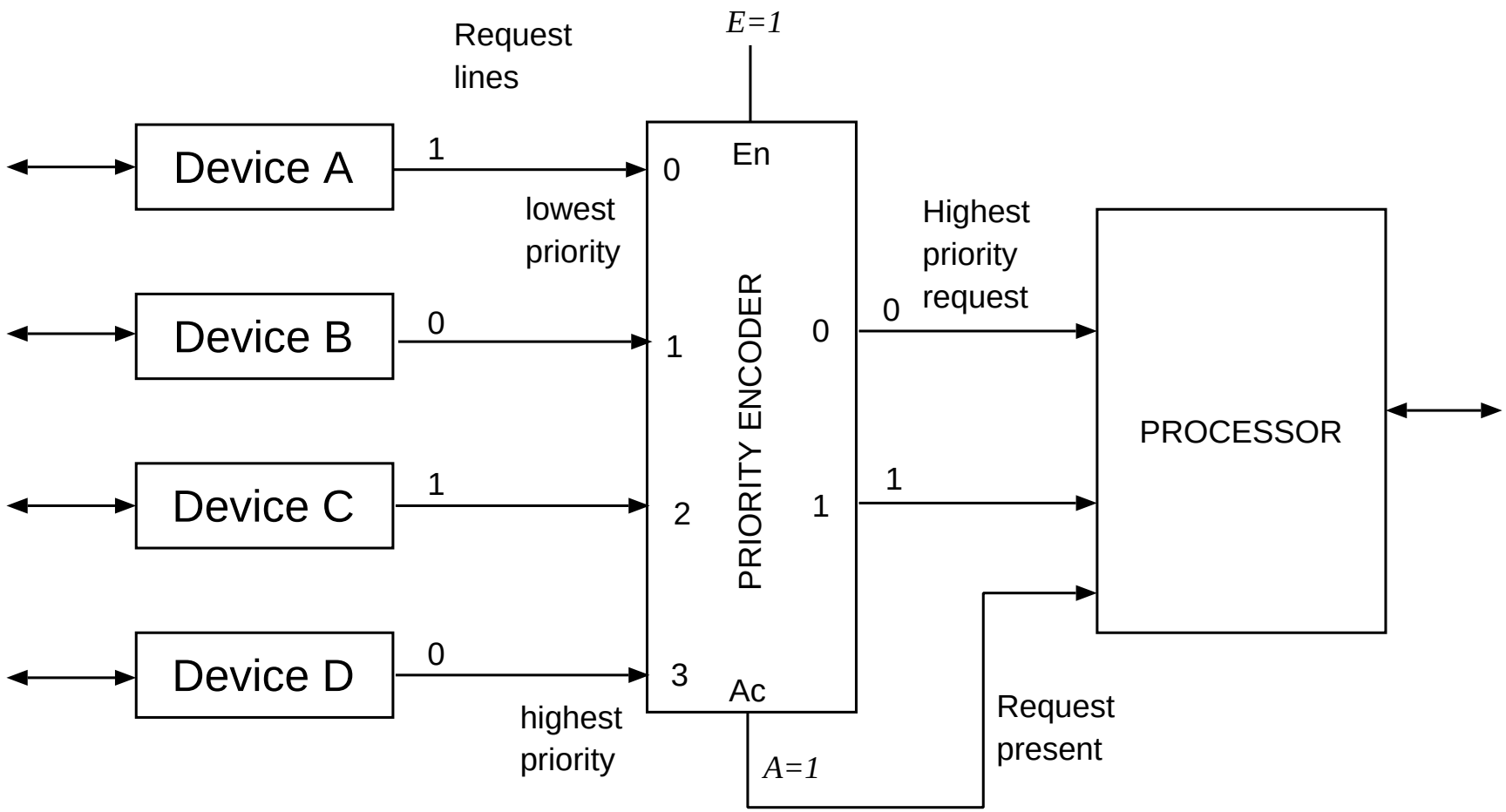


Figure 9.17: RESOLVING INTERRUPT REQUESTS USING A PRIORITY ENCODER.

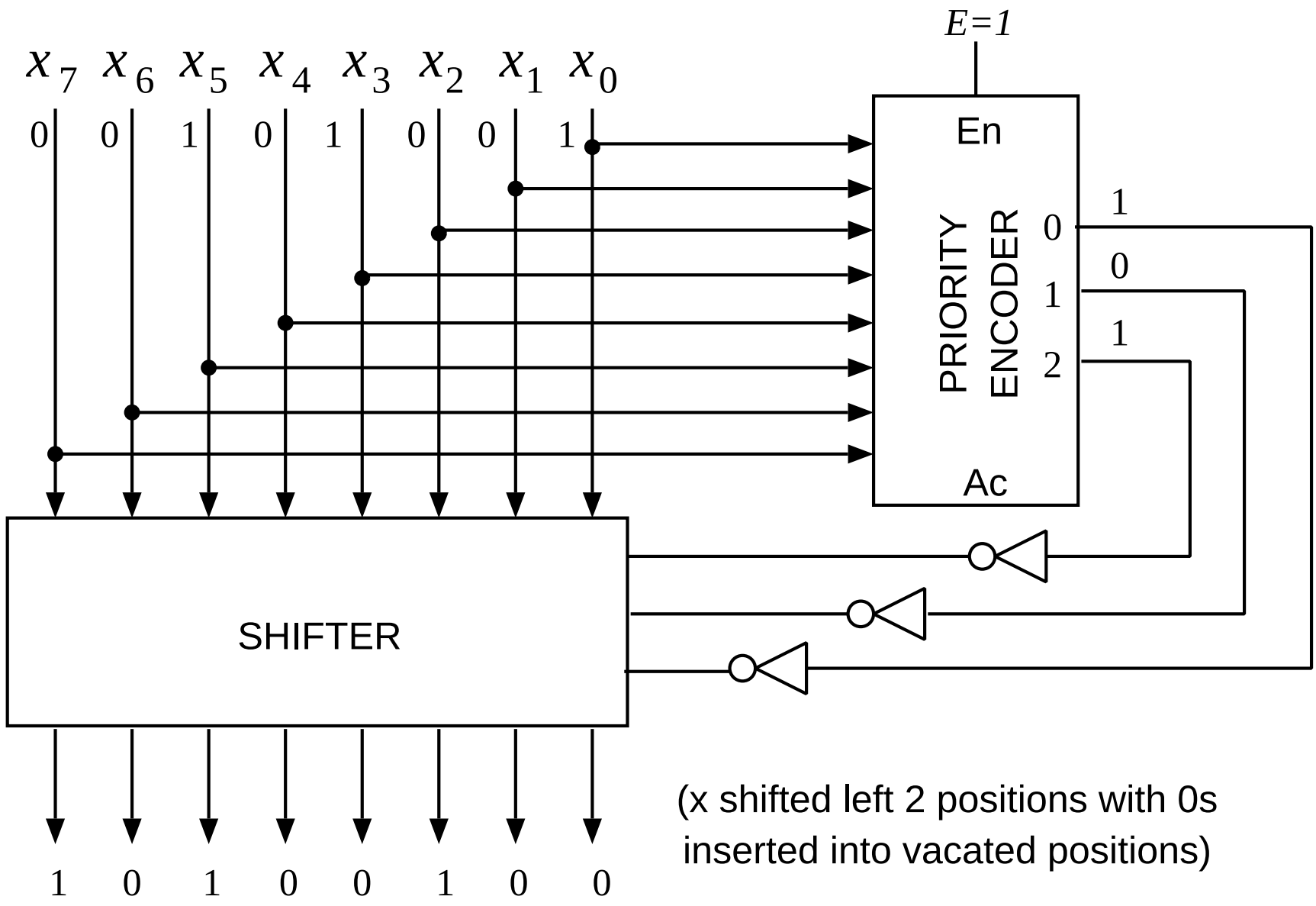
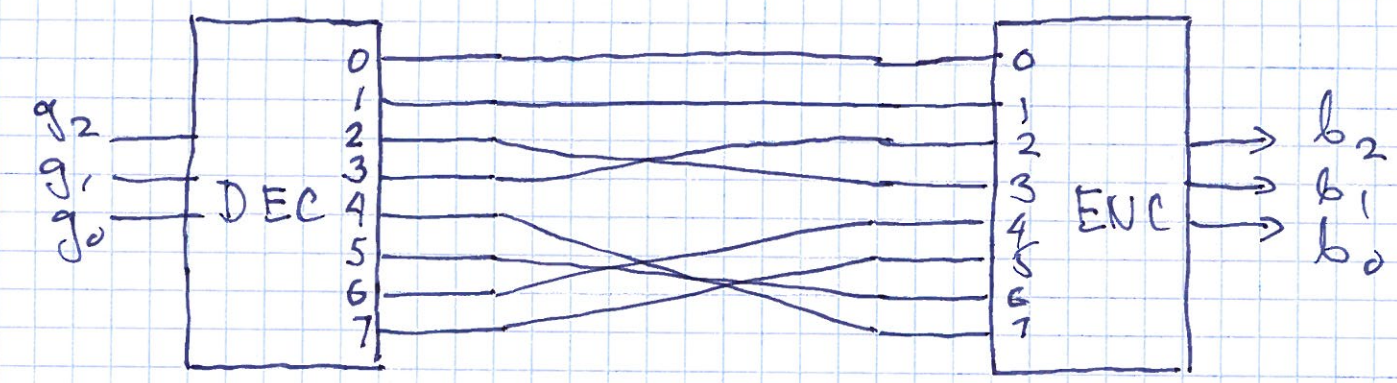


Figure 9.18: DETECTING THE LEFTMOST 1 IN A BIT-VECTOR AND REMOVING LEADING ZEROES.

CS M51A 3-BIT GRAY-TO-BINARY CODE
(EX. 9.9) USING BINARY DECODER AND ENCODER

DECIMAL IN	GRAY CODE g_2	g_1	g_0	DECODER OUT		ENCODER IN	ENCODER OUT b_2	b_1	b_0
0	0	0	0	0	→	0	0	0	0
1	0	0	1	1	→	1	0	0	1
2	0	1	1	3	→	2	0	1	0
3	0	1	0	2	→	3	0	1	1
4	1	1	0	6	→	4	1	0	0
5	1	1	1	7	→	5	1	0	1
6	1	0	1	5	→	6	1	1	0
7	1	0	0	4	→	7	1	1	1



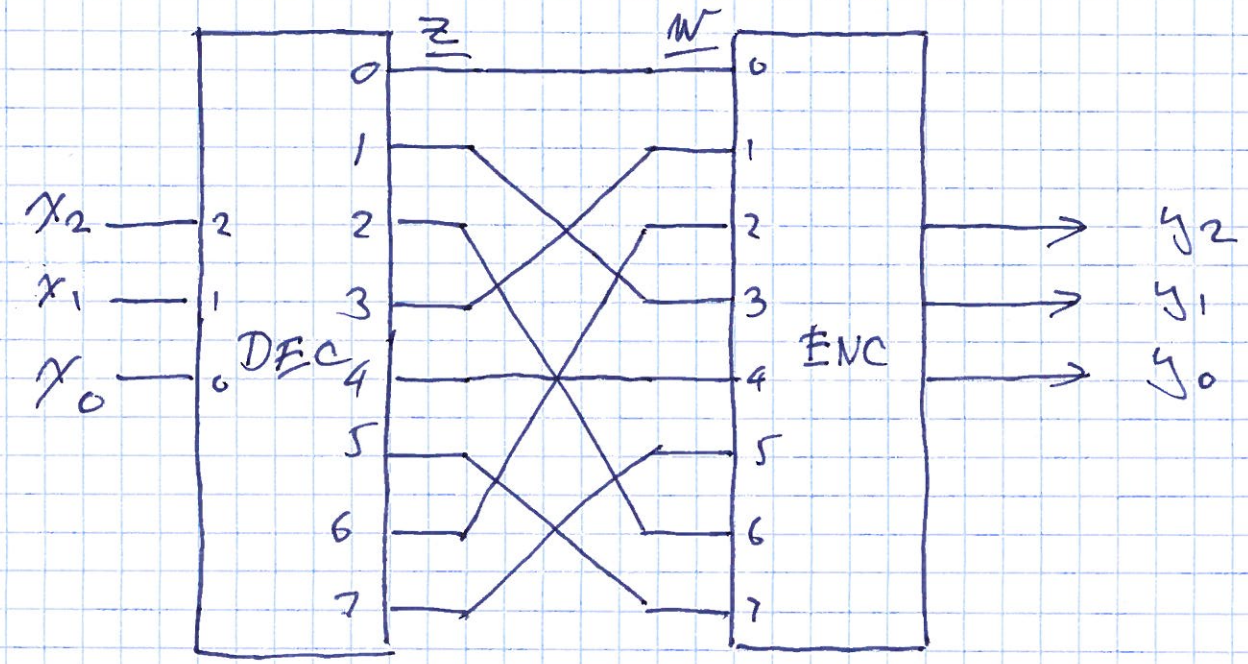
(1 1 0) $\xrightarrow{\text{DEC}}$ 6 → 4 $\xrightarrow{\text{ENC}}$ (1 0 0)

CS M51A

COMPUTE $y = (3x) \bmod 8$ USING BINARY DECODER AND ENCODER

INPUT: $x \in \{0, 1, 2, \ldots, 7\}$ OUTPUT: $y \in \{0, 1, 2, \ldots, 7\}$

x	0	1	2	3	4	5	6	7
y	0	3	6	1	4	7	2	5



IN OUT IN OUT

$\underline{x} = (1\ 1\ 0) \longrightarrow z_6 \longrightarrow w_2 \longrightarrow \underline{y} = (0\ 1\ 0)$

MULTIPLEXERS (selectors)

- HIGH-LEVEL AND BINARY-LEVEL DESCRIPTION

INPUTS: $\underline{x} = (x_{2^n-1}, \ldots, x_0), \quad x_i \in \{0,1\}$

$\underline{s} = (s_{n-1}, \ldots, s_0), \quad s_j \in \{0,1\}$

$E \in \{0,1\}$

OUTPUTS: $z \in \{0,1\}$

FUNCTION: $z = \begin{cases} x_s & \textbf{if} \quad E = 1 \\ 0 & \textbf{if} \quad E = 0 \end{cases}$

$$s = \sum_{j=0}^{n-1} s_j 2^j$$

$$z = E \cdot \left[\sum_{i=0}^{2^n-1} x_i \cdot m_i(\underline{s}) \right]$$

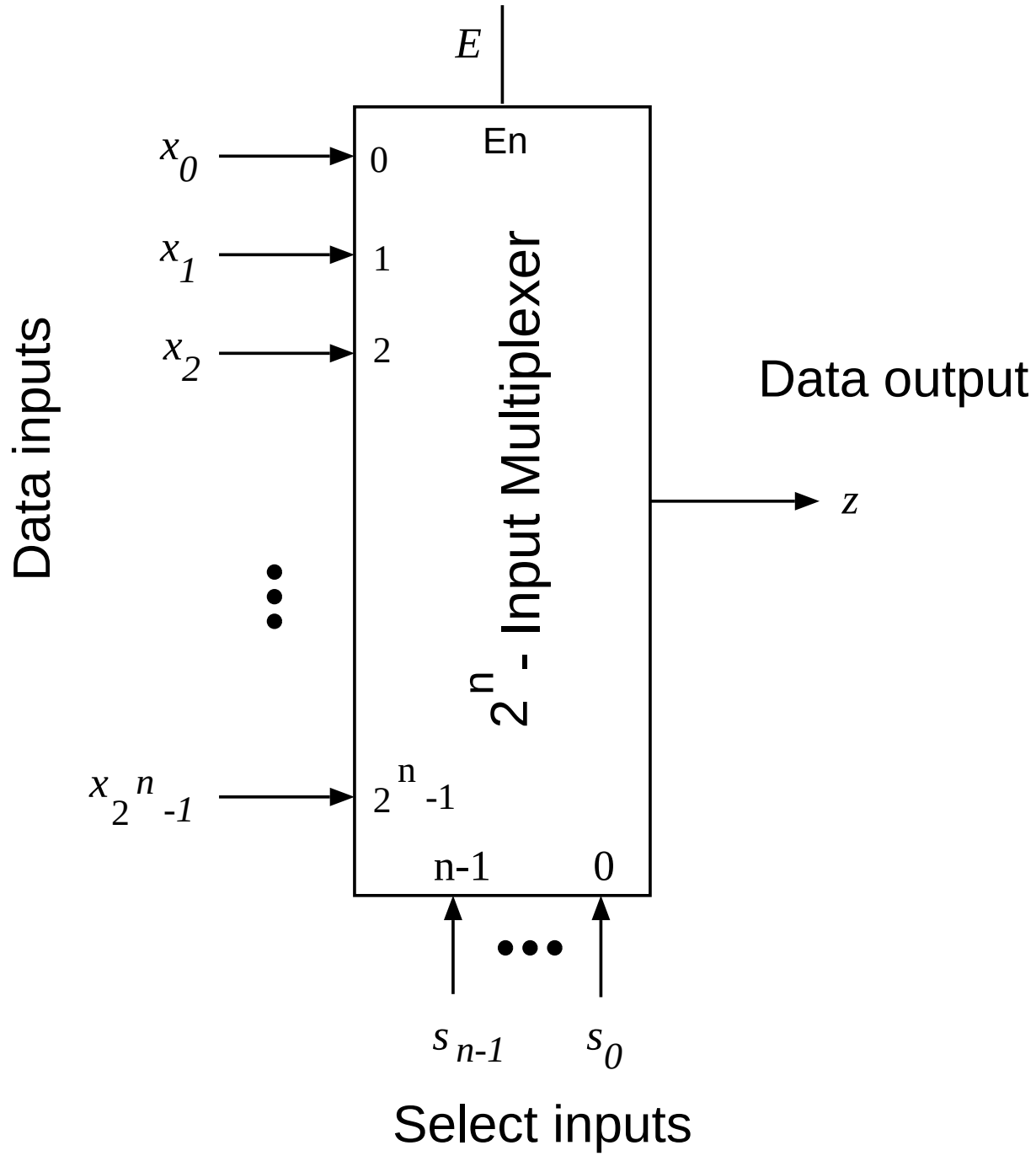


Figure 9.19: 2^n-INPUT MULTIPLEXER.

EXAMPLE 9.11: 4-INPUT MULTIPLEXER

E	s_1	s_0	z
1	0	0	x_0
1	0	1	x_1
1	1	0	x_2
1	1	1	x_3
0	-	-	0

$$
\begin{aligned}
z &= E \cdot (x_0 m_0(s_1, s_0) + x_1 m_1(s_1, s_0) + x_2 m_2(s_1, s_0) + x_3 m_3(s_1, s_0)) \\
&= E \cdot (x_0 s_1' s_0' + x_1 s_1' s_0 + x_2 s_1 s_0' + x_3 s_1 s_0)
\end{aligned}
$$

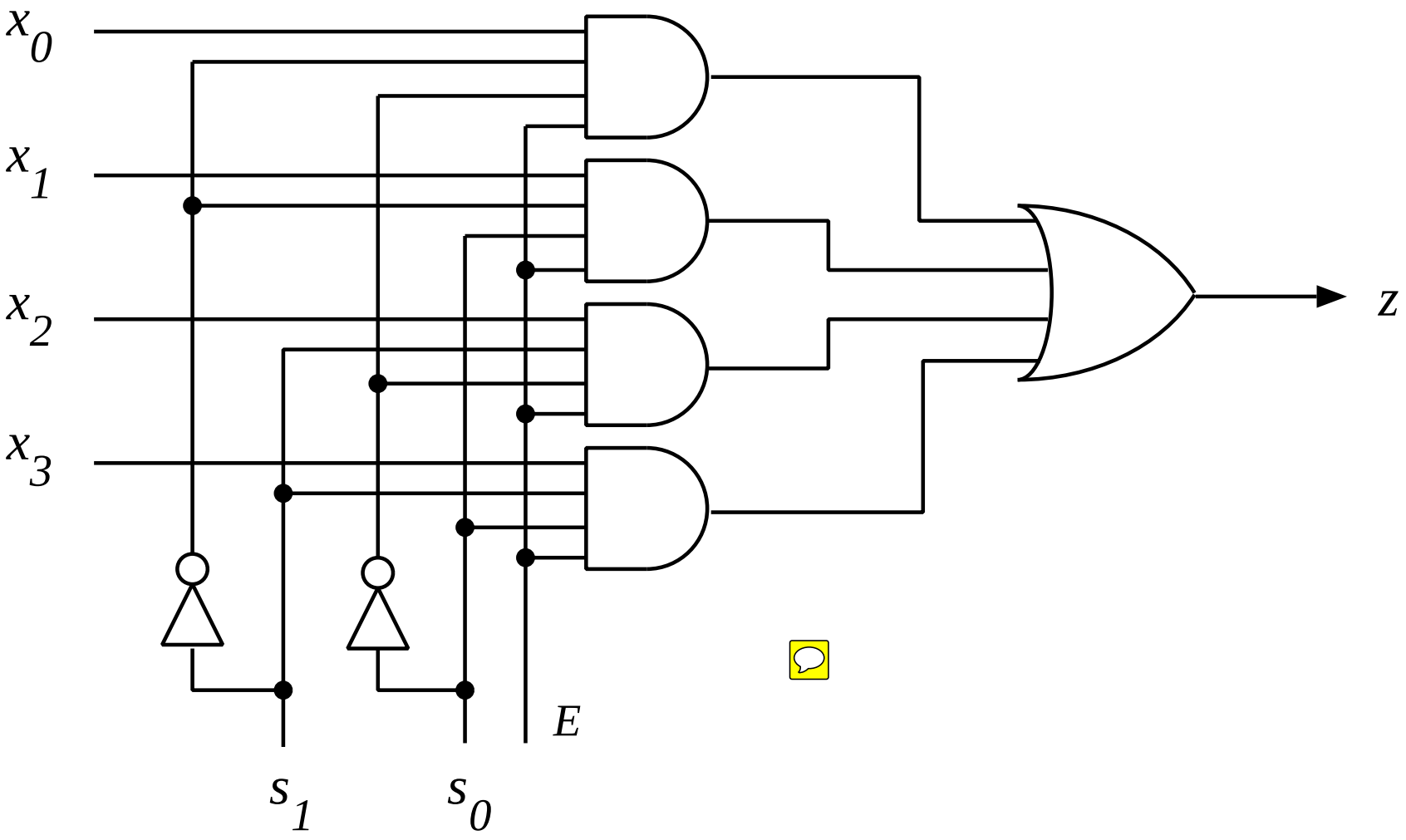
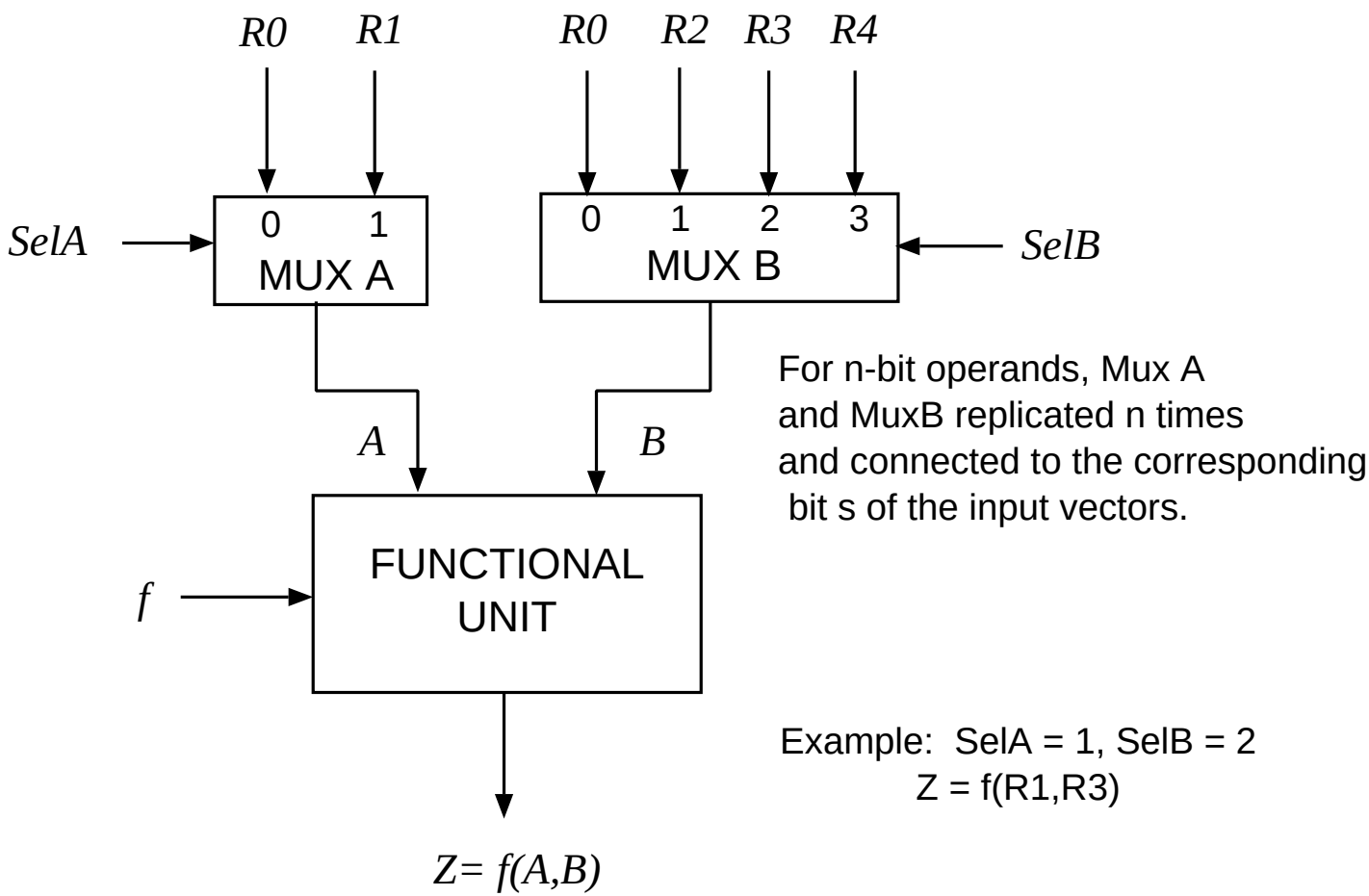


Figure 9.20: GATE IMPLEMENTATION OF 4-INPUT MULTIPLEXER

TYPICAL USES



For n-bit operands, Mux A and MuxB replicated n times and connected to the corresponding bit s of the input vectors.

Example: SelA = 1, SelB = 2
Z = f(R1,R3)

Figure 9.21: MULTIPLEXER: EXAMPLE OF USE.

MULTIPLEXER AS UNIVERSAL COMBINATIONAL MODULE

- connect input variables $\underline{x}$ to select inputs of multiplexer $\underline{s}$
- set data inputs to multiplexer equal to values of function for corresponding assignment of select variables
- using a variable at data inputs reduces size of the multiplexer

EXAMPLE

$$
\begin{aligned}
E(x_2, x_1, x_0) &= \Sigma\, m(1, 2, 4, 6, 7) \\
&= x_2'(x_1'x_0) + x_2'(x_1x_0') + x_2(x_1'x_0') + x_2(x_1x_0') + x_2(x_1x_0) \\
&= x_2'm_1(x_1, x_0) + x_2'm_2(x_1, x_0) \\
&\quad + x_2m_0(x_1, x_0) + x_2m_2(x_1, x_0) + x_2m_3(x_1, x_0) \\
&= x_2m_0(x_1, x_0) + x_2'm_1(x_1, x_0) \\
&\quad + 1 \cdot m_2(x_1, x_0) + x_2m_3(x_1, x_0)
\end{aligned}
$$

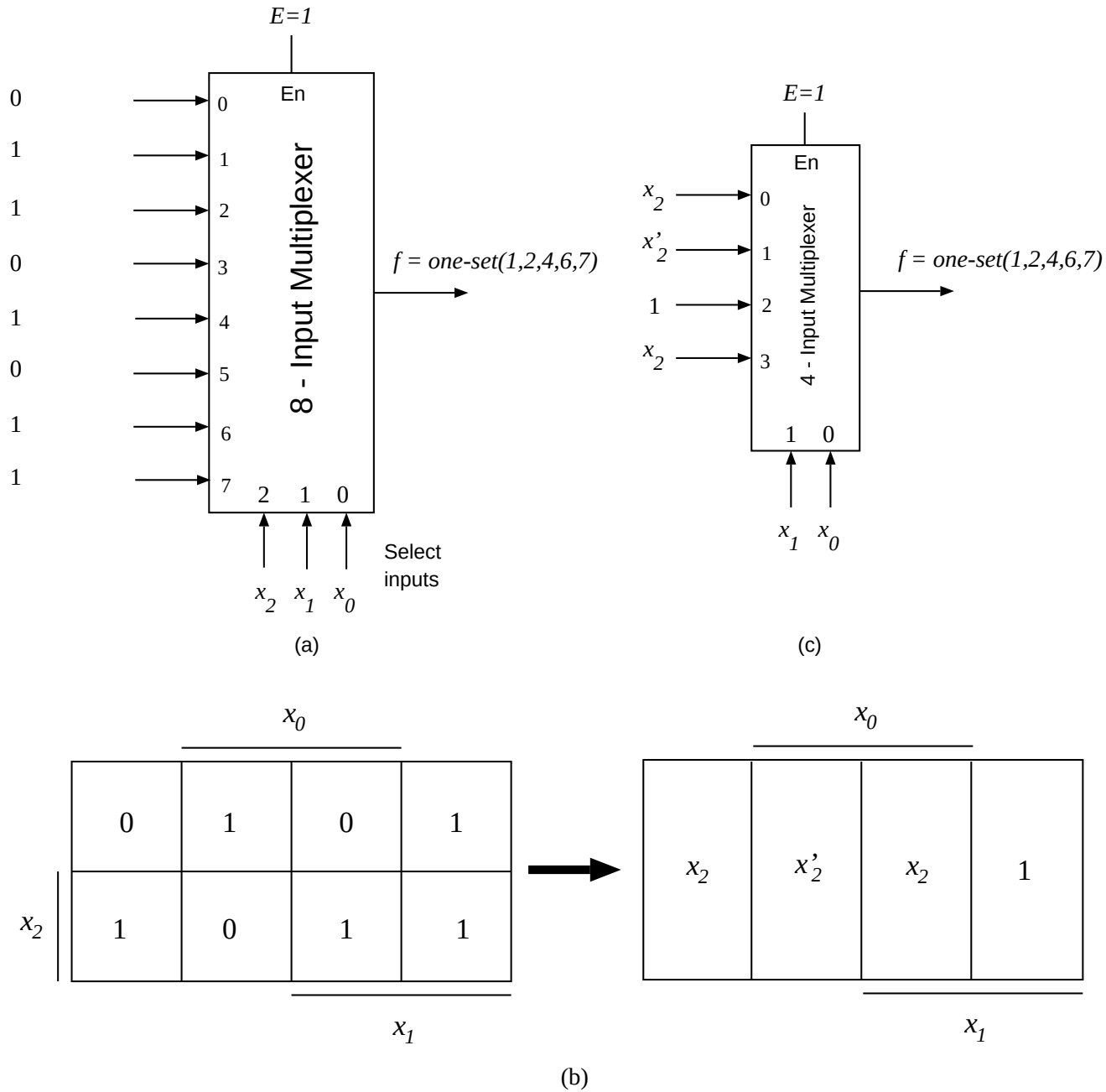


Figure 9.22: IMPLEMENTATION OF $f(x_2, x_1, x_0) = \textit{one-set}(1,2,4,6,7)$: a) 8-INPUT MULTIPLEXER; b) K-map; c) 4-INPUT MULTIPLEXER.

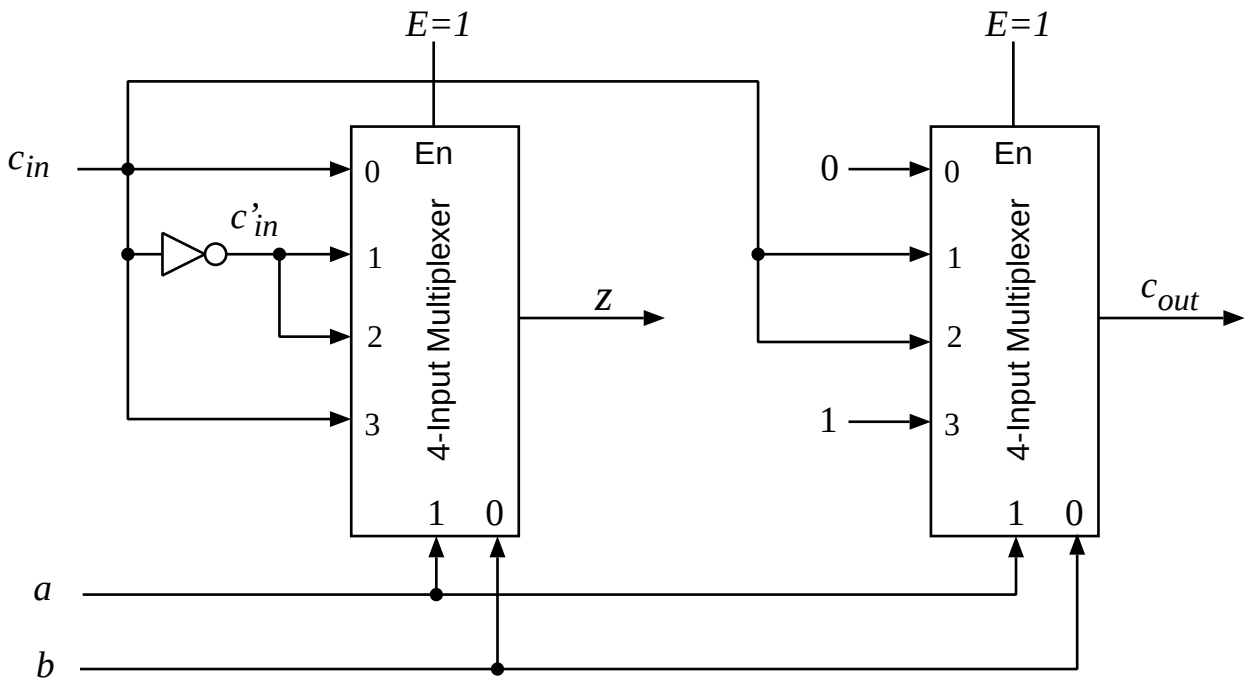
EXAMPLE 9.12: ONE-BIT ADDER

INPUTS: $a, b, c_{in} \in \{0, 1\}$
OUTPUTS: $z, c_{out} \in \{0, 1\}$

a	b	c_{in}	z	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\begin{aligned} z &= (a'b') \cdot c_{in} + (a'b) \cdot c'_{in} + (ab') \cdot c'_{in} + (ab) \cdot c_{in} \\ &= c_{in} m_0(a, b) + c'_{in} m_1(a, b) + c'_{in} m_2(a, b) + c_{in} m_3(a, b) \end{aligned}$$

$$c_{out} = 0 \cdot m_0(a,b) + c_{in} m_1(a,b) + c_{in} m_2(a,b) + 1 \cdot m_3(a,b)$$



z:

a \ c_{in}, b			
	0	1	0	1
a	1	0	1	0

c_{out} :

a \ c_{in}, b			
	0	0	1	0
a	0	1	1	1

a \ b		
	c_{in}	c'_{in}
a	c'_{in}	c_{in}

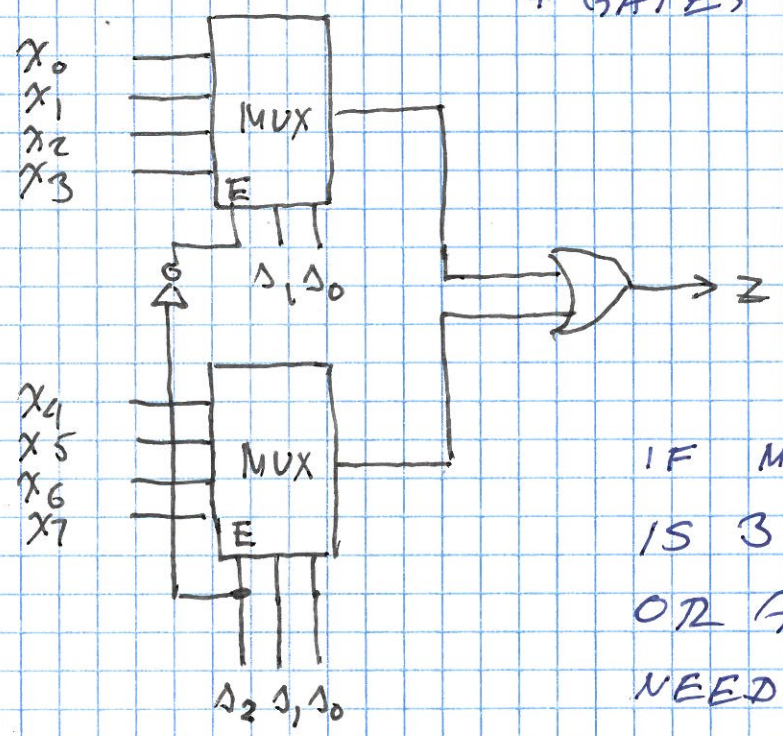
a \ b		
	0	c_{in}
a	c_{in}	1

Figure 9.23: IMPLEMENTATION OF ONE-BIT ADDER WITH 4-INPUT MULTIPLEXERS.

CS MS1A

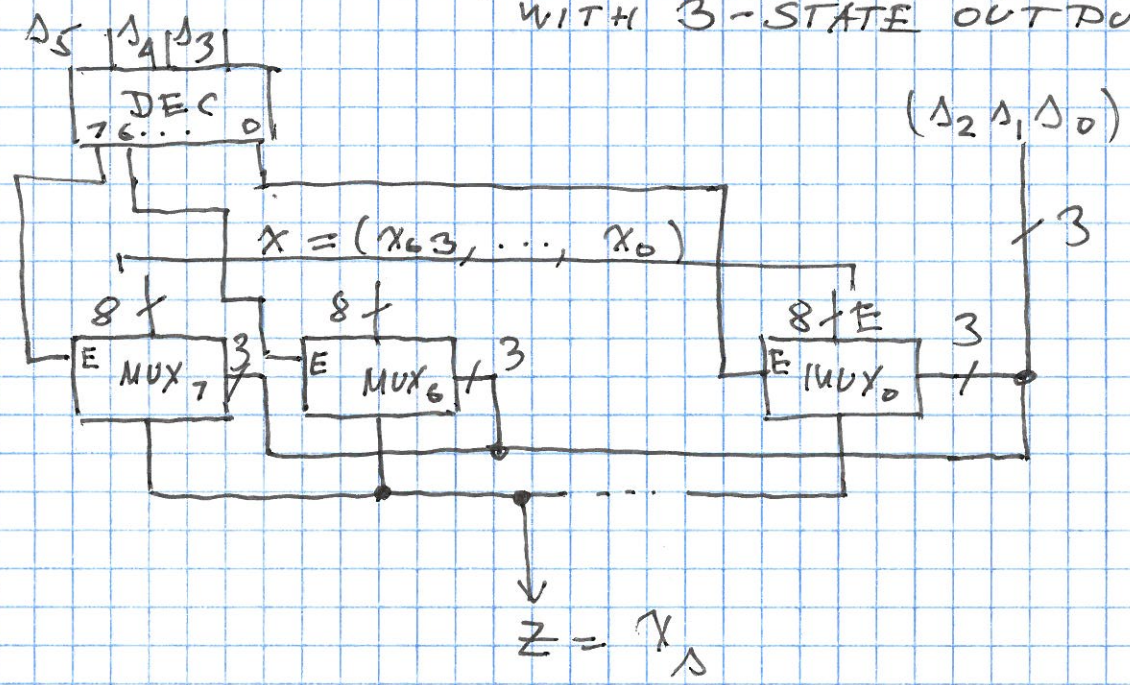
MULTIPLEXER NETWORKS : EXAMPLES

8-TO-1 MUX USING 4-TO-1 MUXES + GATES



IF MUX OUTPUT IS 3-STATE, OR GATE NOT NEEDED.

64-TO-1 MUX (WITH DECODER + 8-1 MUXES WITH 3-STATE OUTPUTS)



CS MSIA

VECTOR MULTIPLEXERS

MUX CAN BE GENERALIZED

$$Z = MUX(x_1, x_0, s) = x_1 s + x_0 s' = \begin{cases} x_1 & \text{IF } s \\ x_0 & \text{IF } s' \end{cases}$$

REPLACE x_1 AND x_0 WITH k-BIT VECTORS $\underline{x1} = (x1_{k-1}, \ldots, x1_1, x1_0)$ AND

$$\underline{x0} = (x0_{k-1}, \ldots, x0_1, x0_0)$$

AND Z WITH

$$\underline{Z} = (Z_{k-1}, \ldots, Z_0)$$

NOW

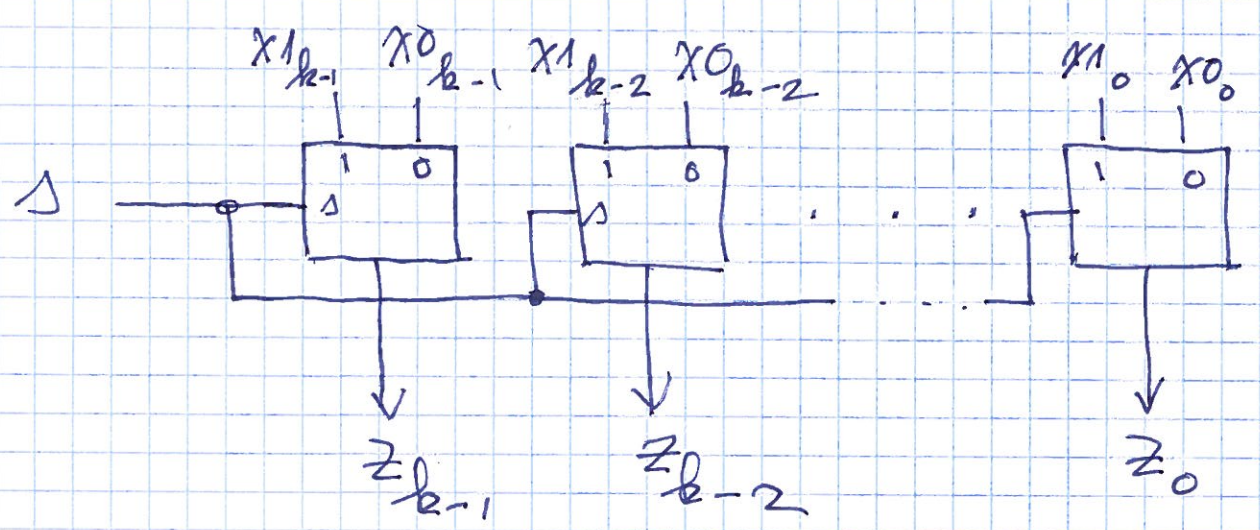
$$\underline{Z} = \begin{cases} \underline{x1} & \text{if } s \\ \underline{x0} & \text{IF } s' \end{cases}$$

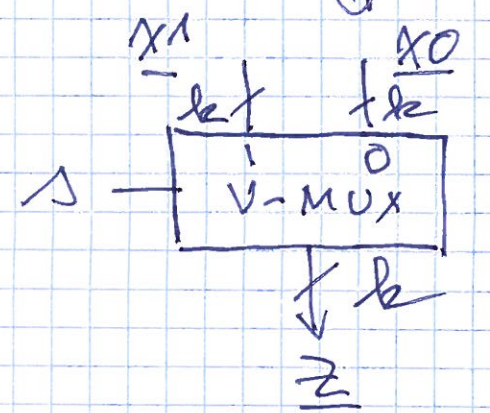
AT BIT-LEVEL

$$Z_i = \begin{cases} x1_i & \text{IF } s \\ x0_i & \text{IF } s' \end{cases} \qquad i = 0, \ldots, k-1$$

CS M51A

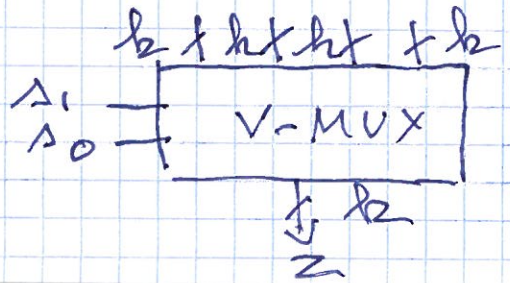
VECTOR MULTIPLEXER





(2-TO-1 V-MUX MADE OF k 2-TO-1 MUXES)

CAN BE EXTENDED TO MORE INPUTS



MULTIPLEXER TREES

$$\underline{s}_{\text{left}} = (s_3, s_2)$$

$$\underline{s}_{\text{right}} = (s_1, s_0)$$

$$w_j = x_{(4j+s_{\text{right}})} \quad , \quad 0 \leq j \leq 3$$

$$z = w_{s_{\text{left}}}$$

$$s = 4s_{\text{left}} + s_{\text{right}}$$

$$z = x_{4s_{\text{left}}+s_{\text{right}}} = x_s$$

16-INPUT TREE MULTIPLEXER

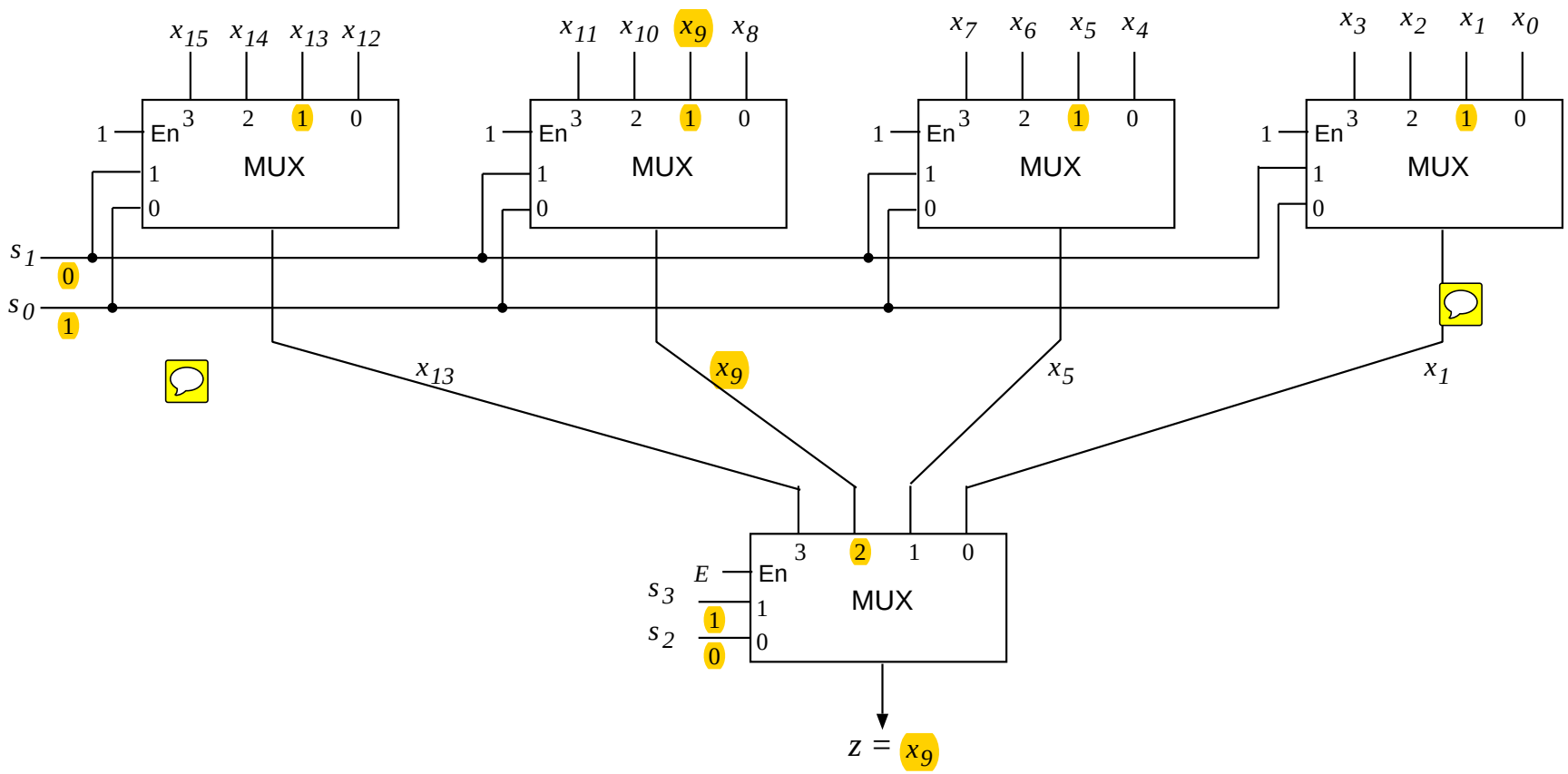


Figure 9.24: TREE IMPLEMENTATION OF A 16-INPUT MULTIPLEXER.

CSMSIA DESIGN THE CHARACTER REPLACEMENT SYSTEM (EXAMPLE 2.2)

INPUT: $\underline{x} = (x_3, x_2, x_1, x_0)$, $\underline{x}_i \in \{A, B, \ldots, Z, a, b, \ldots, z\}$

$y \in \{A, \ldots, z\}$

$k \in \{0, 1, 2, 3\}$

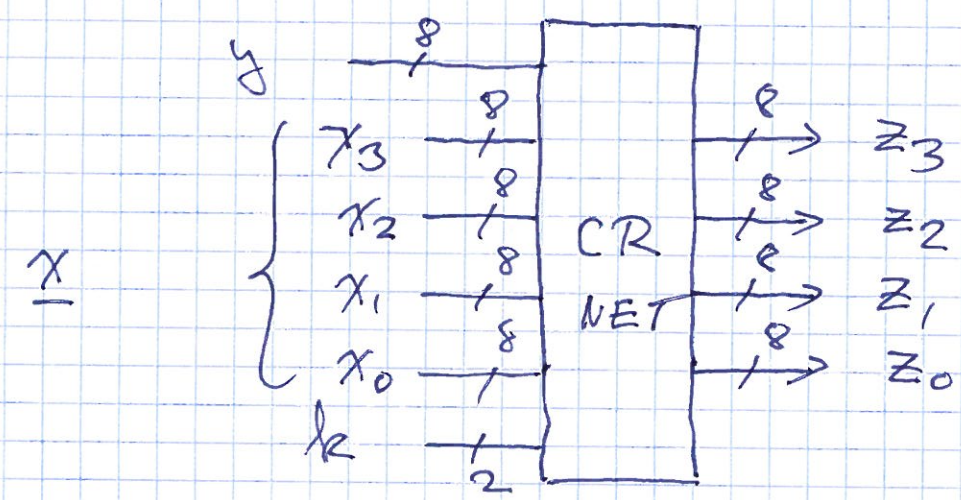
OUTPUT: $\underline{z} = (z_3, z_2, z_1, z_0)$, $z_i \in \{A, B, \ldots Z, a, b, \ldots, z\}$

FUNCTION:

$$z_j = \begin{cases} x_j & \text{if } j \neq k \\ y & \text{if } j = k \end{cases}$$

EXAMPLE: $\underline{x} = (C\ A\ S\ E)$ $y = R$ $k = 1$

$\underline{z} = (C\ A\ R\ E)$



CHAR

$\underline{x}_j = (x_{j7}, x_{j6}, \ldots, x_{j0})$

$\underline{y} = (y_7, y_6, \ldots, y_0)$

$x_{jk} \in \{0, 1\}$

$y_j \in \{0, 1\}$

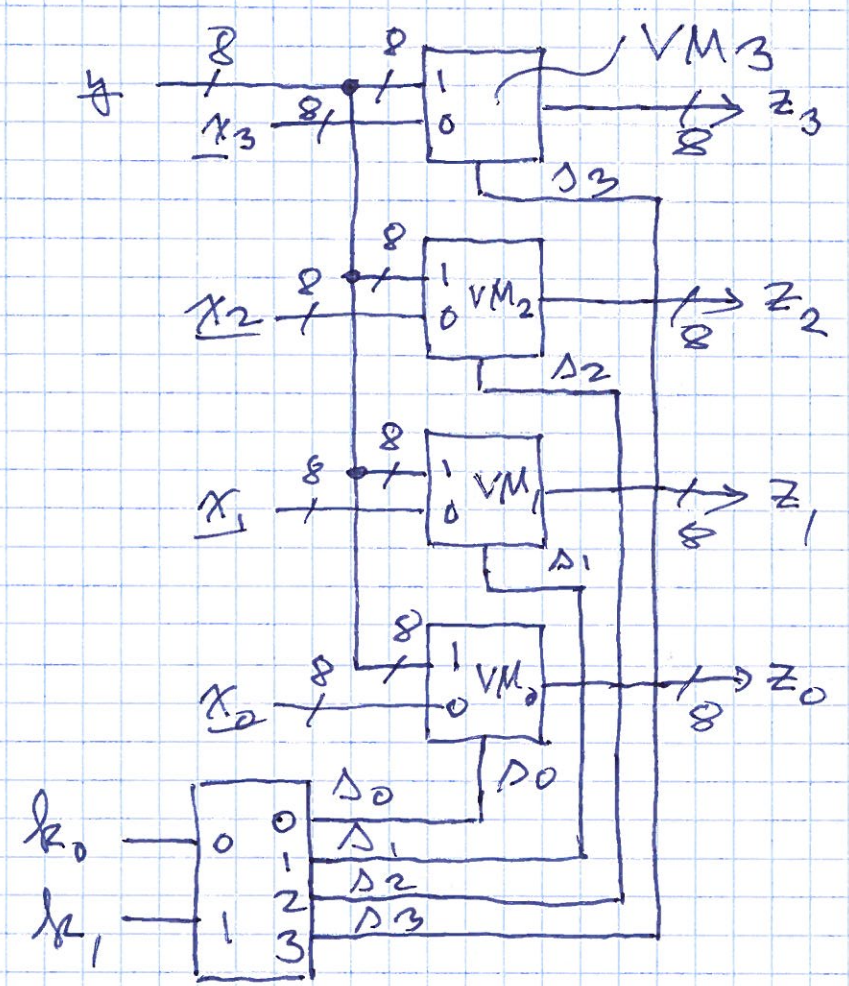
$k_1, k_0 \in \{0, 1\}$

CS MSIA DESIGN (CONT.)

MODULES: VM — 8-BIT 2-TO-1 MUX (VECTOR MUX OF WIDTH 8)

$$\underline{z}_j = \underline{x}_j \text{ IF } s_j = 0;\ \underline{y} \text{ IF } s_j = 1$$

DECODER: D $\quad \underline{s} = (s_3, s_2, s_1, s_0) = DEC(k_1, k_0)$



COST:

1 2-TO-4 BINARY DECODER

4 x (8 2-TO-1) MULTIPLEXERS

= 32 2-TO-1 MUXES

DELAY:

$$T = t_{DEC} + t_{2\text{-}1\,MUX}$$

~ 5 LOGIC LEVELS

DEMULTIPLEXERS (distributors)

- A HIGH-LEVEL DESCRIPTION:

INPUTS: $x, E \in \{0,1\}$

$\underline{s} = (s_{n-1}, \ldots, s_0) \quad , \quad s_j \in \{0,1\}$

OUTPUTS: $\underline{y} = (y_{2^n-1}, \ldots, y_0) \quad , \quad y_i \in \{0,1\}$

FUNCTION: $y_i = \begin{cases} x \ \textbf{if} \ \ (i = s) \ \textbf{and} \ (E = 1) \\ 0 \ \textbf{if} \ \ (i \neq s) \ \textbf{or} \ (E = 0) \end{cases}$

$$s = \sum_{j=0}^{n-1} s_j 2^j, \ 0 \leq i \leq 2^n - 1$$

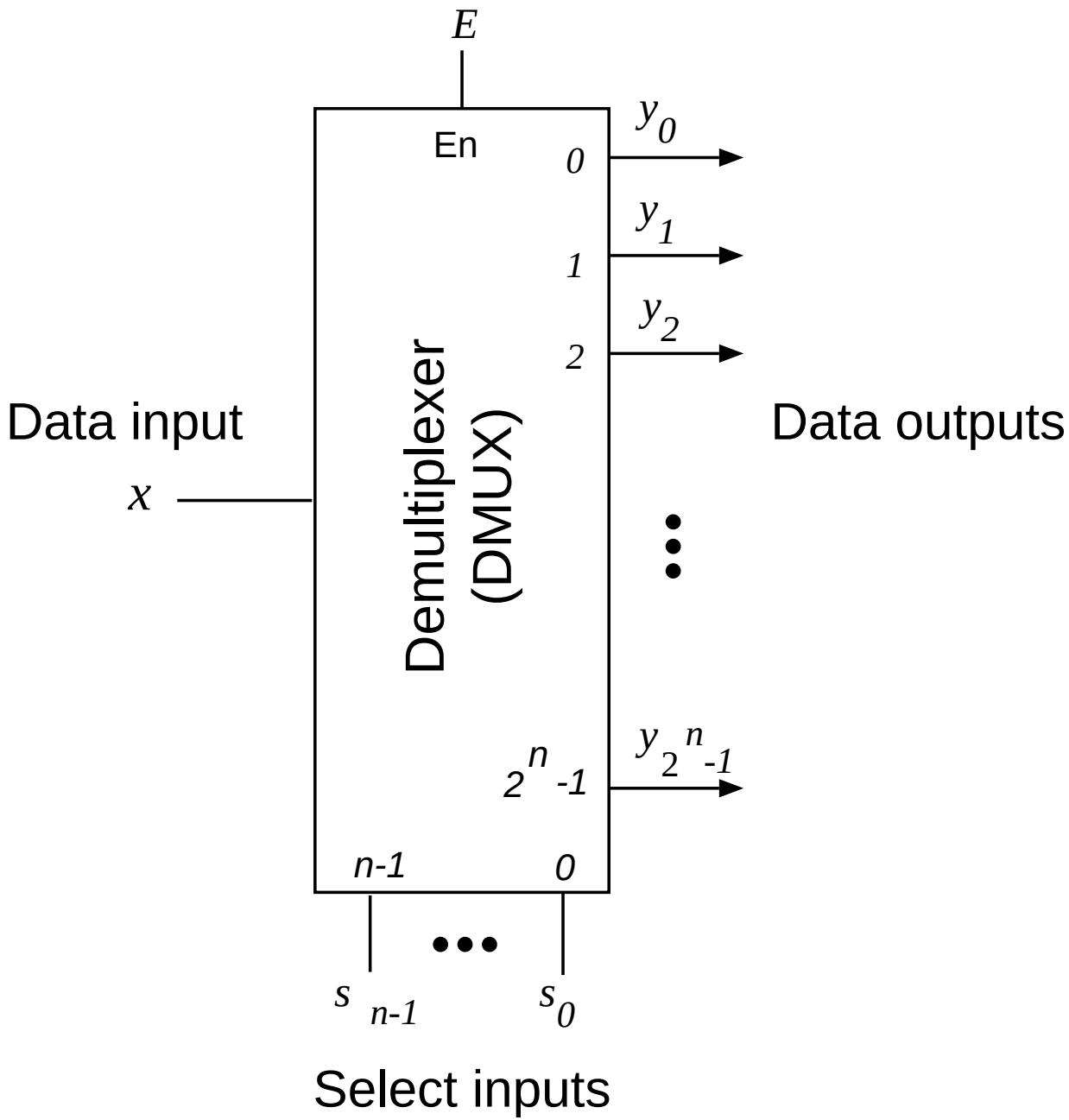


Figure 9.25: 2^n-OUTPUT DEMULTIPLEXER.

EXAMPLE 9.13: 4-OUTPUT DEMULTIPLEXER

E	s_1	s_0	s	y_3	y_2	y_1	y_0
1	0	0	0	0	0	0	x
1	0	1	1	0	0	x	0
1	1	0	2	0	x	0	0
1	1	1	3	x	0	0	0
0	-	-	-	0	0	0	0

$$y_i = E \cdot x \cdot m_i(\underline{s}), \qquad 0 \leq i \leq 2^n - 1$$

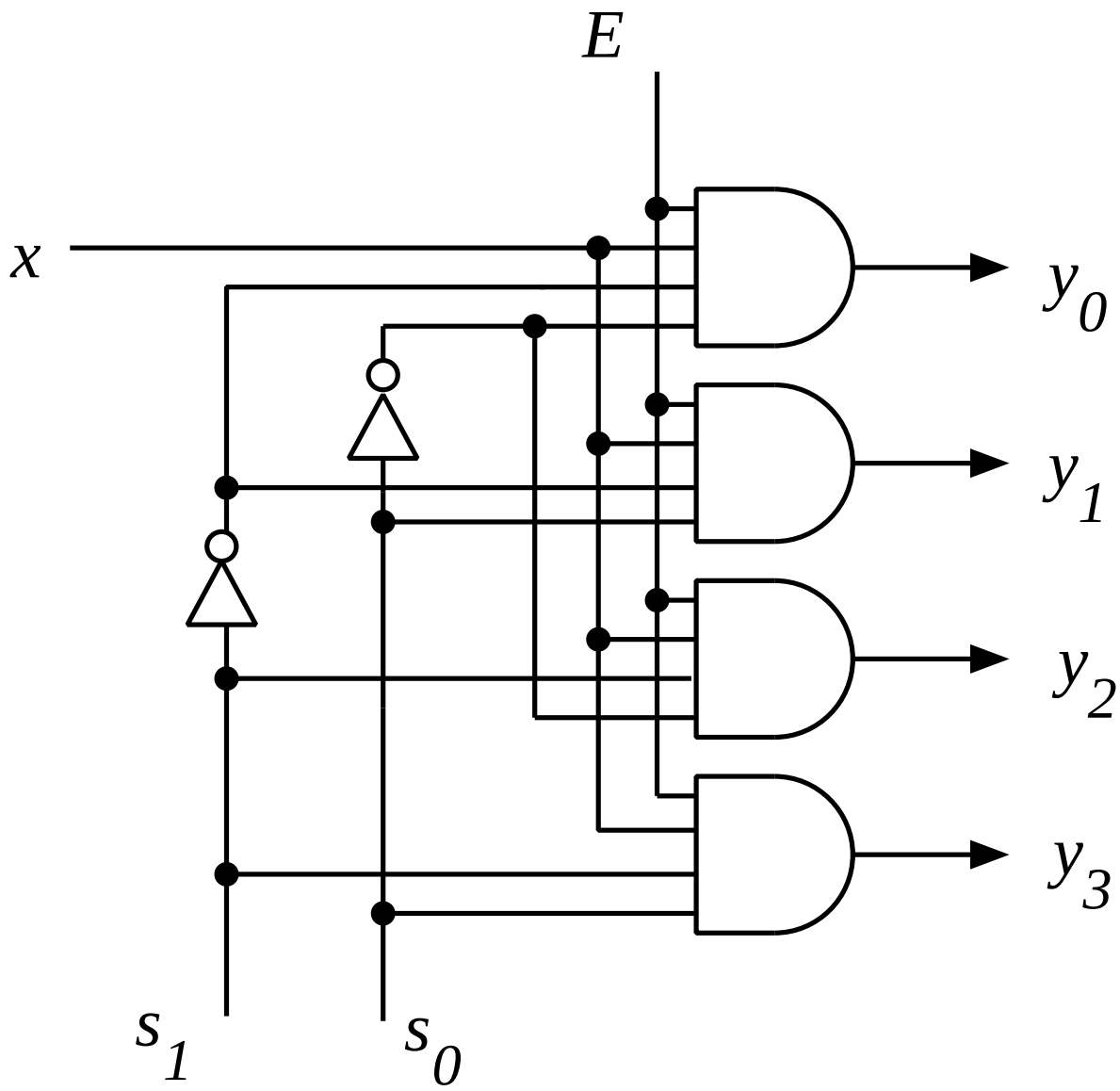


Figure 9.26: GATE NETWORK IMPLEMENTATION OF A 4-OUTPUT DEMULTIPLEXER.

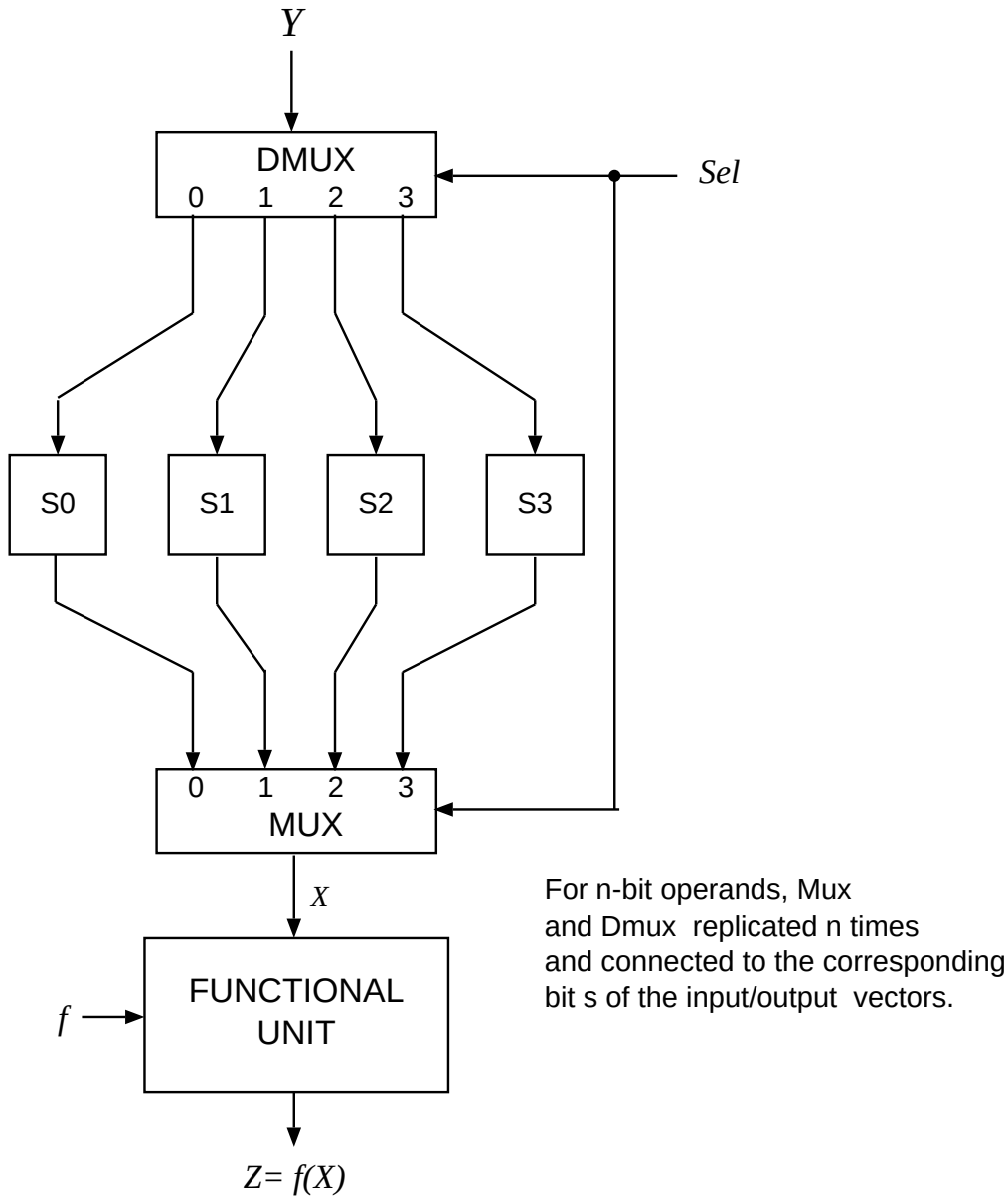


Figure 9.27: DEMULTIPLEXER: example of use.

SIMPLE SHIFTER

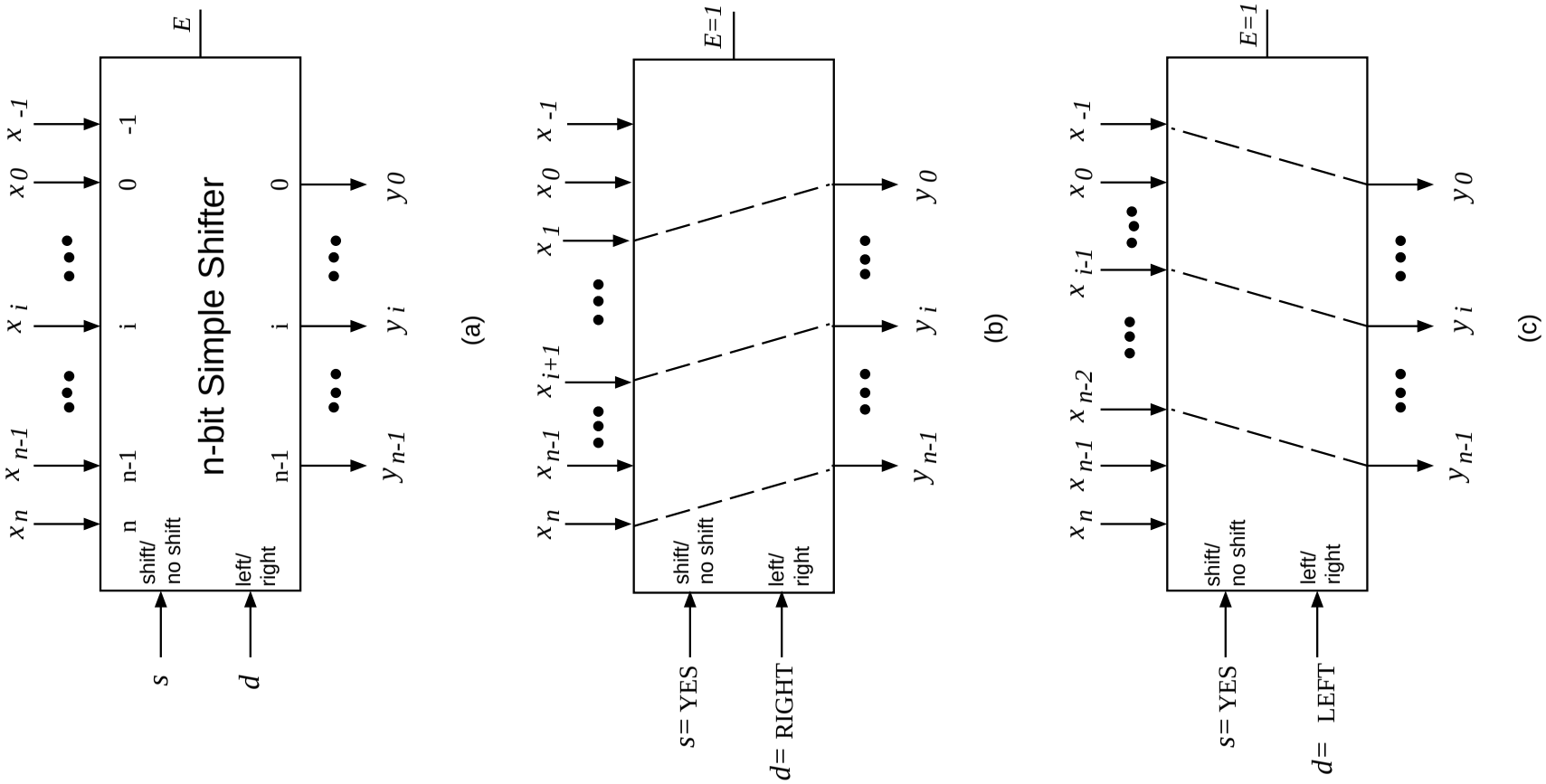


Figure 9.28: n-BIT SIMPLE SHIFTER: a) BLOCK DIAGRAM; b) RIGHT SHIFT; c) LEFT SHIFT.

SIMPLE SHIFTER: HIGH-LEVEL DESCRIPTION

INPUTS: $\underline{x} = (x_n, x_{n-1}, \ldots, x_0, x_{-1}) \quad , \quad x_j \in \{0, 1\}$

$d \in \{RIGHT,\ LEFT\}$

$s \in \{YES,\ NO\}$

$E \in \{0, 1\}$

OUTPUTS: $\underline{y} = (y_{n-1}, \ldots, y_0) \quad , \quad y_j \in \{0, 1\}$

FUNCTION:

$$y_i = \begin{cases} x_{i-1} & \textbf{if} \ \ (d = LEFT) \ \textbf{and} \ (s = YES) \ \textbf{and} \ (E = \\ x_{i+1} & \textbf{if} \ \ (d = RIGHT) \ \textbf{and} \ (s = YES) \ \textbf{and} \ (E \\ x_i & \textbf{if} \ \ (s = NO) \ \textbf{and} \ (E = 1) \\ 0 & \textbf{if} \ \ (E = 0) \end{cases}$$

for $0 \leq i \leq n - 1$.

$$x_{-1} = \begin{cases} 0 & \text{left shift with 0 insert} \\ 1 & \text{left shift with 1 insert} \\ x_{n-1} & \text{left rotate} \end{cases}$$

$$x_n = \begin{cases} 0 & \text{right shift with 0 insert} \\ 1 & \text{right shift with 1 insert} \\ x_0 & \text{right rotate} \end{cases}$$

EXAMPLE 9.14: 4-INPUT SHIFTER

	Control		Data					
	s	d	x_4	x_3	x_2	x_1	x_0	x_{-1}
			1	0	0	1	1	0
No shift	NO	–		0	0	1	1	
Right shift	YES	RIGHT		1	0	0	1	
Left shift	YES	LEFT		0	1	1	0	
				y_3	y_2	y_1	y_0	

Coding:

s	
0	NO
1	YES

d	
0	RIGHT
1	LEFT

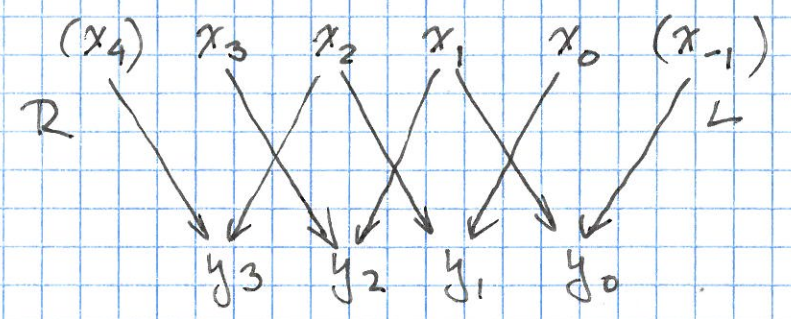
CSMSIA

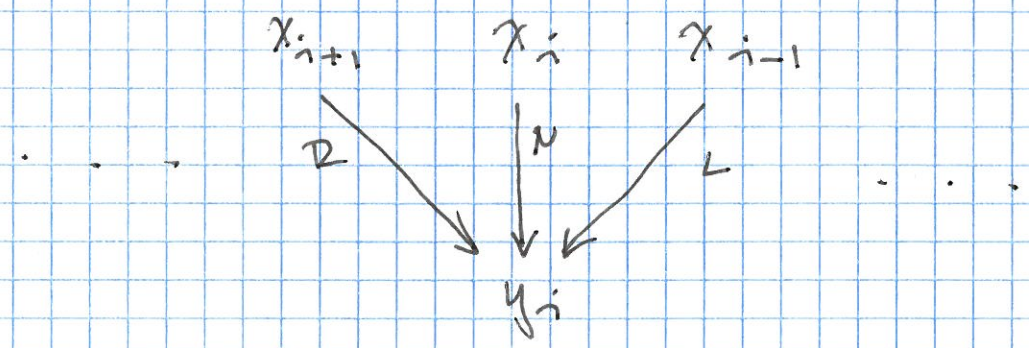
NOTES ON SHIFTERS

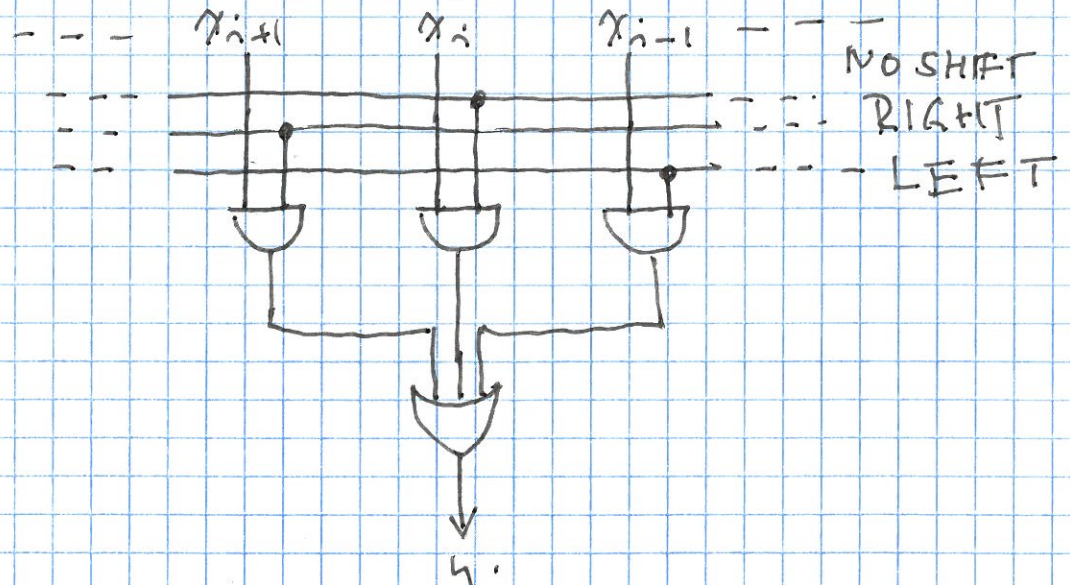
— FOR UNIFORM BIT-VECTOR MOVEMENTS:

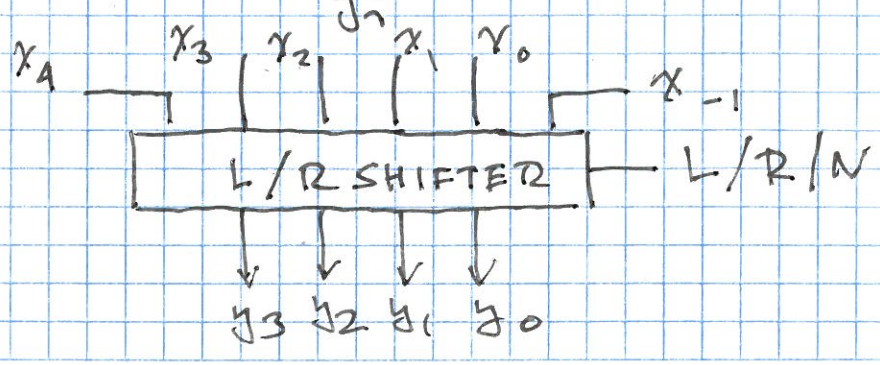
LEFT / RIGHT; 1, 2, 3 ..., p POSITIONS

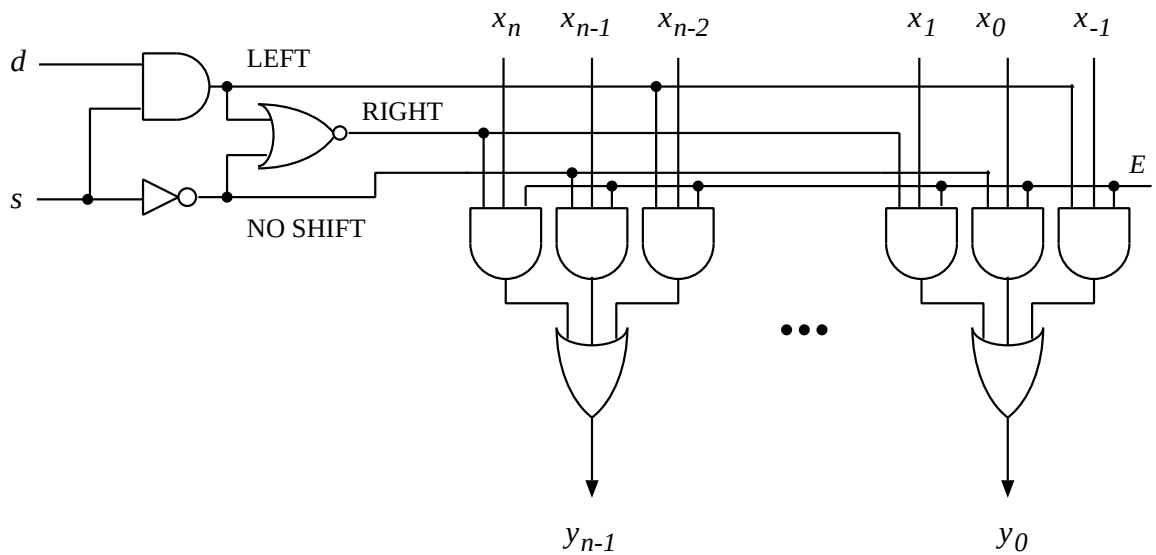
INPUT:



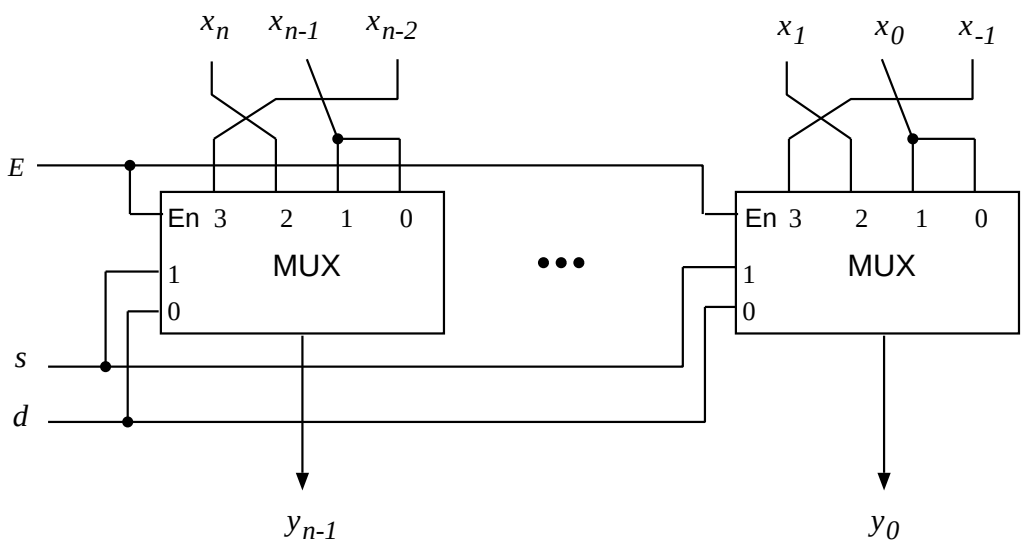








(a)

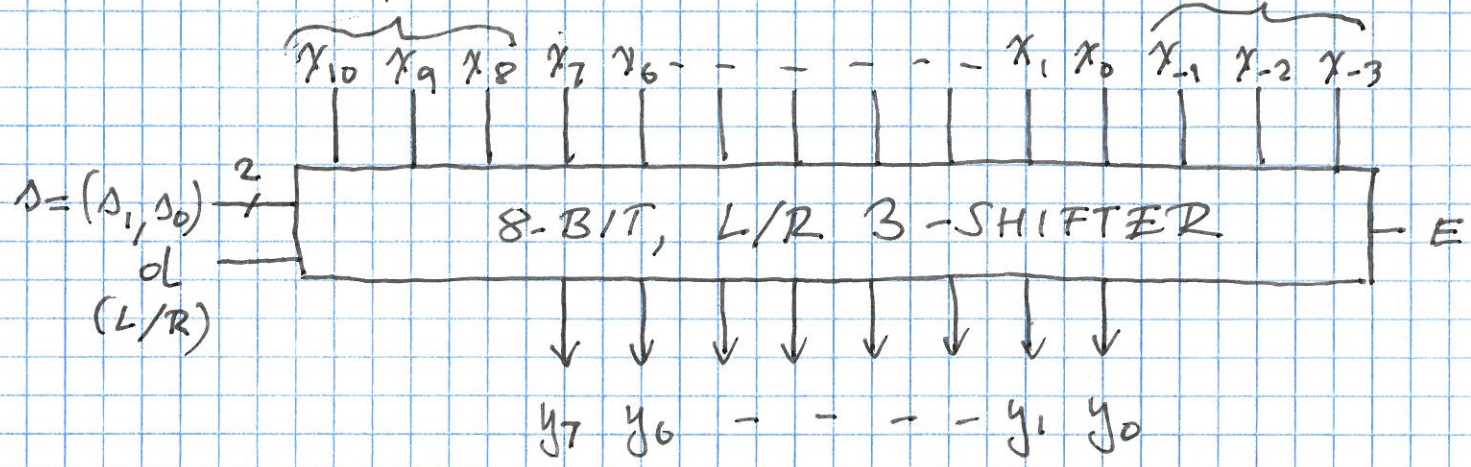


(b)

Figure 9.29: IMPLEMENTATION OF A SIMPLE SHIFTER: a) WITH GATES; b) WITH MULTIPLEXERS.

P-SHIFTER, BIDIRECTIONAL (UNIDIRECTIONAL)

8-BIT, 3-SHIFTER:



EXAMPLE:

x: 1 0 1 0 0 1 1 1 0 1 1 0 1 1

$s=2$

$d=R$

y: 0 1 0 0 1 1 1 0

ROTATOR (LEFT/RIGHT)

$n=4$

1 0 1 1

0 1 1 1 L1 (LEFT ROTATE 1)

1 0 1 1

1 1 0 1

1 1 1 0

0 1 1 1 R3 (RIGHT ROTATE 3)

$$La = R(n-a)$$

p-SHIFTER: HIGH-LEVEL DESCRIPTION

INPUTS: $\underline{x} = (x_{n+p-1}, \ldots, x_n, x_{n-1}, \ldots, x_0, x_{-1}, \ldots, x_{-p})$, x_j

$s \in \{0, 1, ..., p\}$

$d \in \{LEFT,\ RIGHT\}$

$E \in \{0, 1\}$

OUTPUTS: $\underline{y} = (y_{n-1}, \ldots, y_0)$, $y_j \in \{0, 1\}$

FUNCTION:

$$y_i = \begin{cases} x_{i-s} & \textbf{if} \ \ (d = LEFT) \ \textbf{and} \ (E = 1) \\ x_{i+s} & \textbf{if} \ \ (d = RIGHT) \ \textbf{and} \ (E = 1) \\ 0 & \textbf{if} \ \ (E = 0) \end{cases}$$

$$0 \leq i \leq n - 1$$

p-SHIFTER

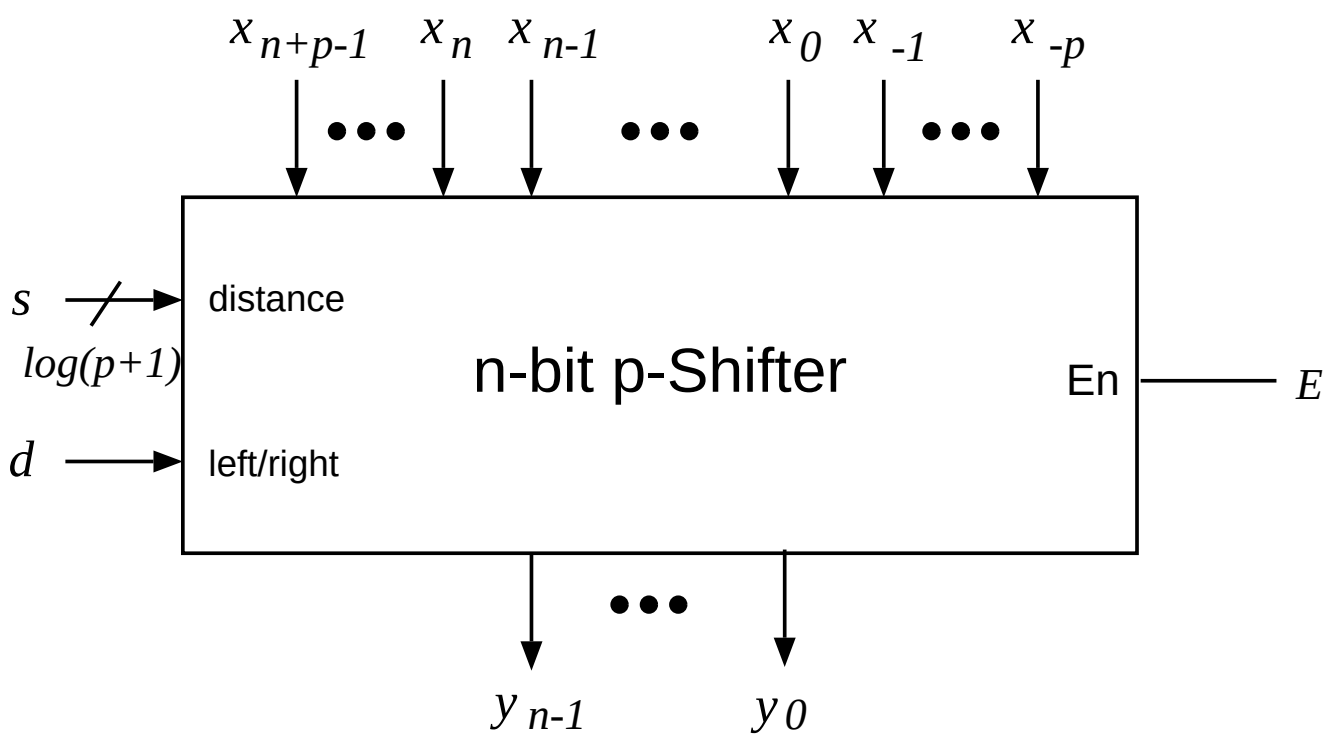


Figure 9.30: n-BIT p-SHIFTER.

BARREL SHIFTER

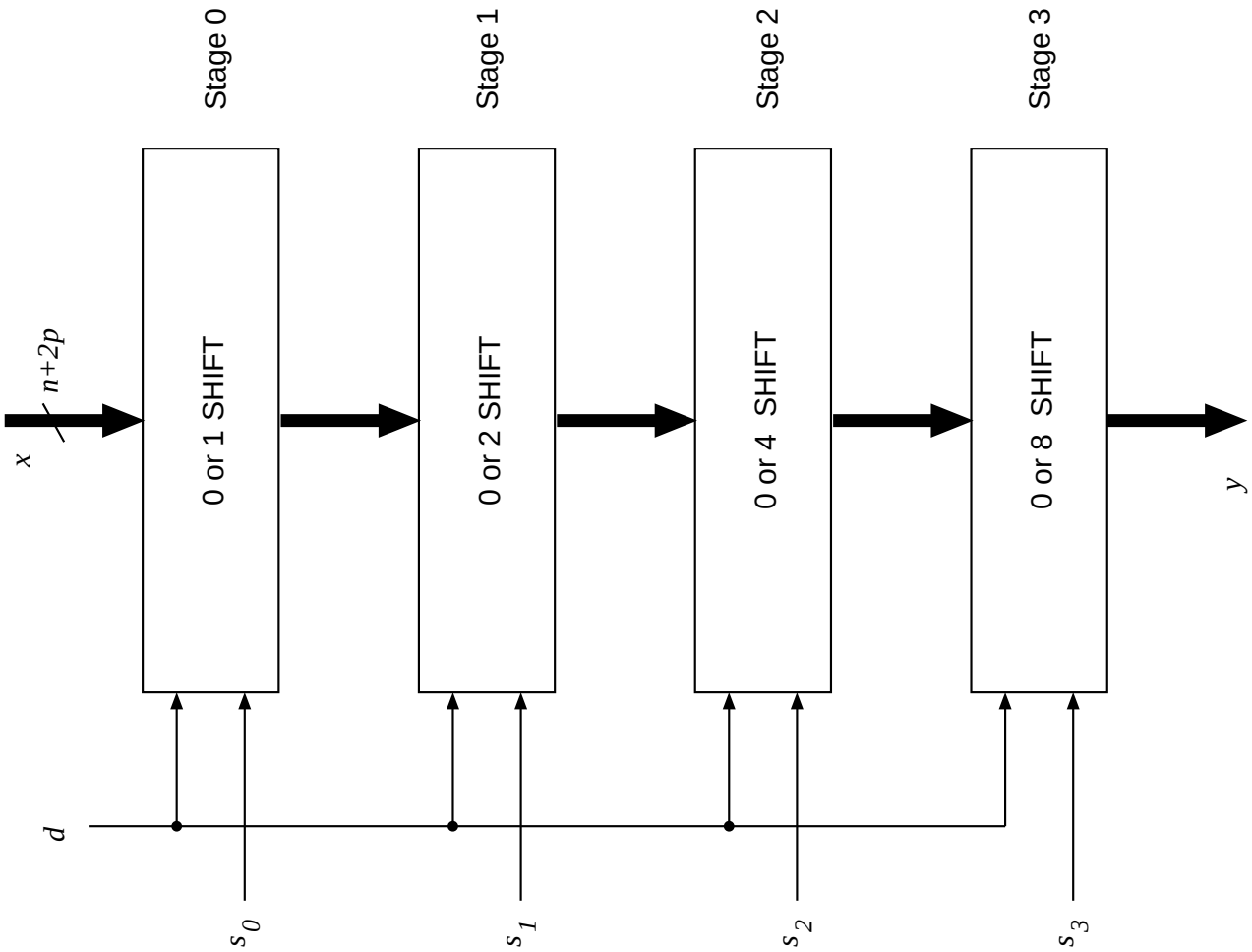


Figure 9.31: BARREL SHIFTER FOR $p = 15$.

LEFT ROTATE BARREL SHIFTER

ROTATE DISTANCE $s = \sum_{i=0} s_i 2^i$

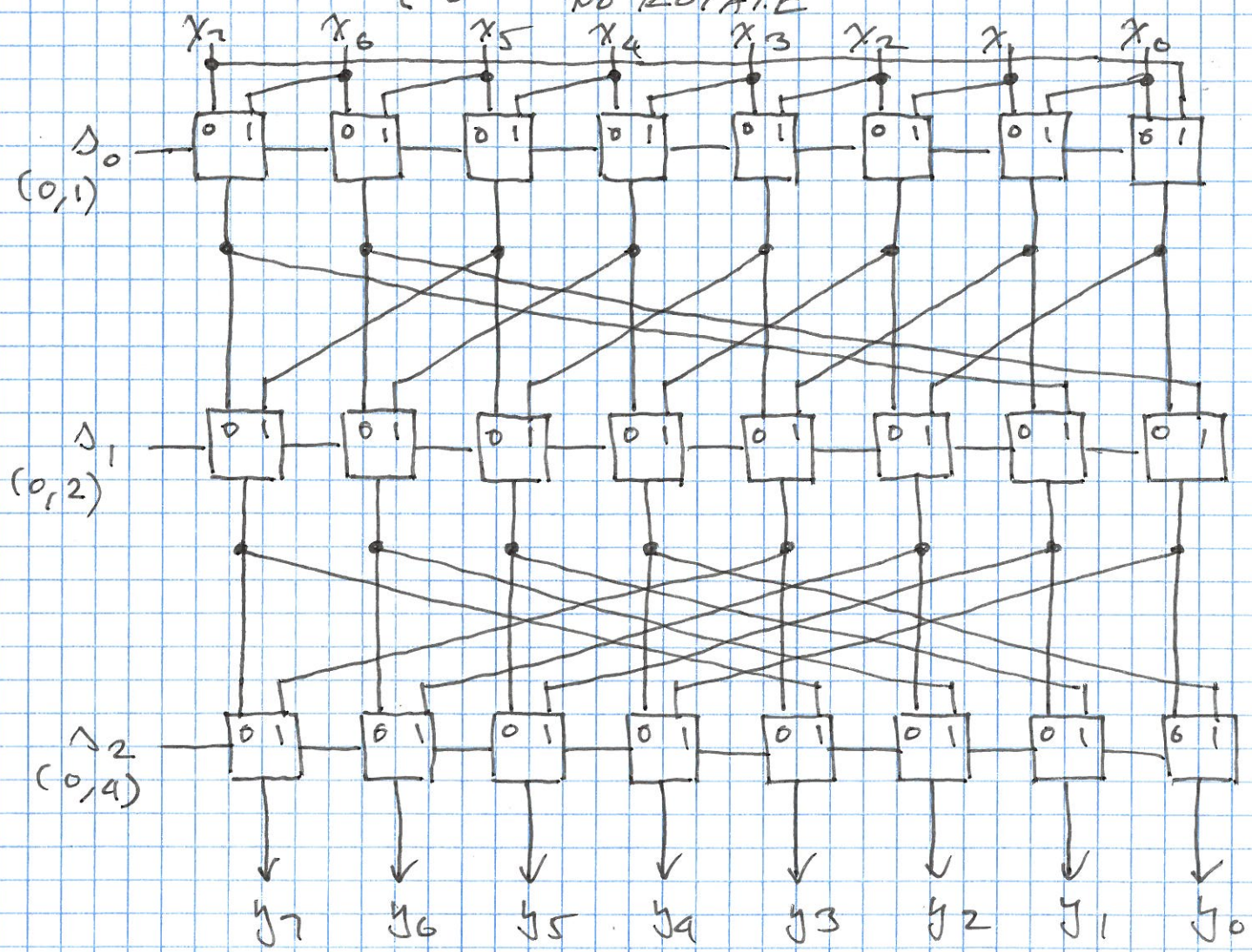
$s_0 = \begin{cases} 1 & \text{ROTATE 1} \\ 0 & \text{NO ROTATE} \end{cases}$

$s_1 = \begin{cases} 1 & \text{ROTATE 2} \\ 0 & \text{NO ROTATE} \end{cases}$

$s_2 = \begin{cases} 1 & \text{ROTATE 4} \\ 0 & \text{NO ROTATE} \end{cases}$

$s = 5$

$= 1\ 0\ 1$ ROTATE



RL2 ≡ RR6	x_5	x_4	x_3	x_2	x_1	x_0	x_7	x_6
RR1 ≡ RL7	x_0	x_7	x_6	x_5	x_4	x_3	x_2	x_1

EXAMPLES OF ROTATIONS:

IN	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	

RR6
≡ RLR

$s = (010)$

									ROTATE
STAGE 1	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	0
STAGE 2	x_5	x_4	x_3	x_2	x_1	x_0	x_7	x_6	2
STEGE 3 (OUT)	x_5	x_4	x_3	x_2	x_1	x_0	x_7	x_6	0

RL7

$s = (111)$

									ROTATE
STAGE 1	x_6	x_5	x_4	x_3	x_2	x_1	x_0	x_7	1
STAGE 2	x_4	x_3	x_2	x_1	x_0	x_7	x_6	x_5	2
STAGE 3 (OUT)	x_0	x_7	x_6	x_5	x_4	x_3	x_2	x_1	4

UNIDIRECTIONAL SHIFTERS

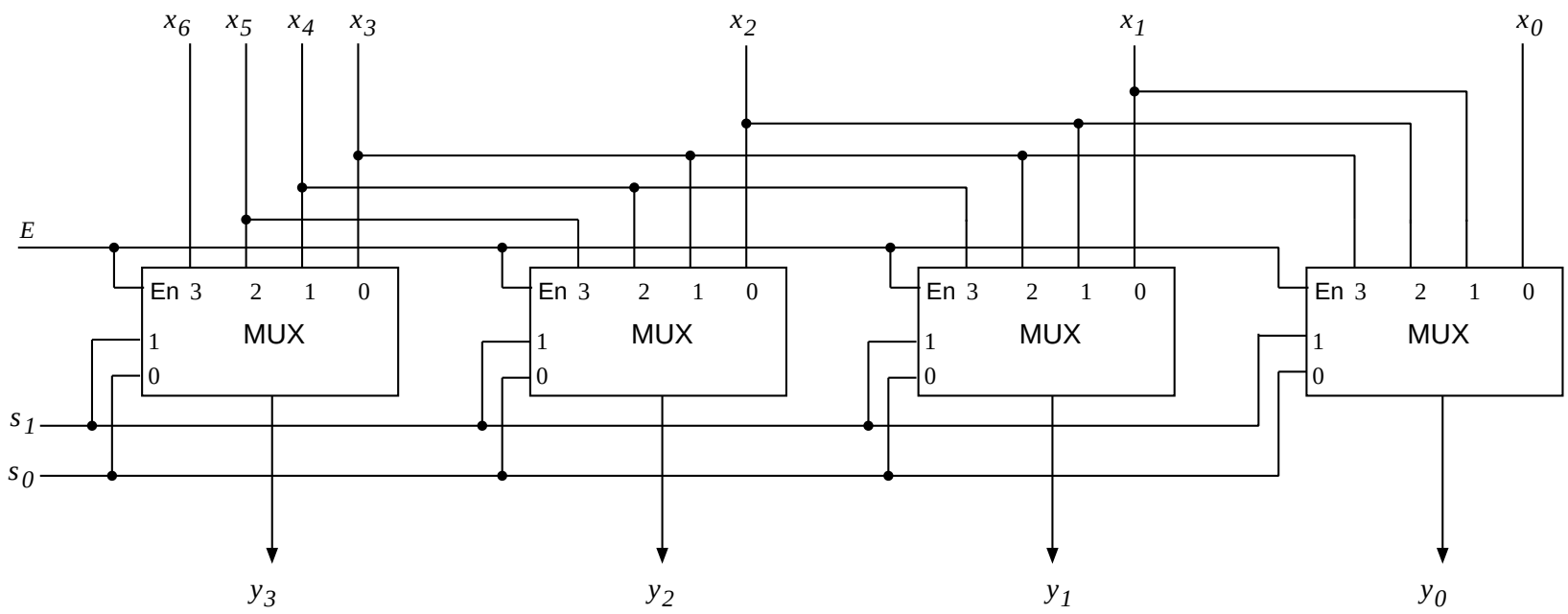


Figure 9.32: MULTIPLEXER IMPLEMENTATION OF A 4-BIT RIGHT 3-SHIFTER.

TYPICAL USES OF SHIFTERS

- ALIGNMENT OF A BIT-VECTOR
- REMOVAL OF THE LEADING (or trailing) BITS OF A VECTOR
- PERFORMING MULTIPLICATION OR DIVISION BY A POWER OF TWO
- EXTRACTING A SUBVECTOR from a bit-vector, using a shifter instead of a selector

CS M51A

P-SHIFTER CAN BE USED TO MULTIPLY AND DIVIDE BY POWERS OF 2.

(L) $y = x \cdot 2^{s}$ $\quad s = \{0, 1, 2, \ldots, p\}$

(R) $z = \left\lfloor \frac{x}{2^{s}} \right\rfloor$

$x = 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1 = 5$

$L3(x) = 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0 = 40 = 5 \times 2^{3}$

$x = 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1 = 13$

$R2(x) = 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1 = 3 = \lfloor 13 \cdot 2^{-2} \rfloor$