DESIGN OF MULTILEVEL NETWORKS

- TRANSFORMATIONS TO SATISFY CONSTRAINTS
  - number of gate inputs
  - network size
  - network delay
- DESIGN OF NETWORKS WITH XOR and XNOR GATES
- DESIGN OF NETWORKS WITH multiplexers (MUXes)

DESIGN MORE COMPLEX THAN FOR TWO-LEVEL NETWORKS

- NO STANDARD FORM
- SEVERAL REQUIREMENTS HAVE TO BE MET SIMULTANEOUSLY
- SEVERAL OUTPUTS HAVE TO BE CONSIDERED
- CAD TOOLS (logic synthesis) USED

A DESIGN PROCEDURE

1. OBTAIN SP or PS EXPRESSIONS FOR THE FUNCTIONS OF THE SYSTEM
2. TRANSFORM THE EXPRESSIONS (or the corresponding two-level networks) so that the requirements are met
3. REPLACE AND and OR GATES BY NAND and NOR WHEN APPROPRIATE

SEVERAL ITERATIONS MIGHT BE NEEDED

TYPICAL TRANSFORMATIONS TO MEET NETWORK REQUIREMENTS

- SIZE OF NETWORK: number of gates and number of gate inputs
- NUMBER OF GATES REDUCED BY
  1. FACTORING
  2. SUBEXPRESSIONS SHARED BY SEVERAL NETWORK OUTPUTS

EXAMPLE 6.1: 1-BIT COMPARATOR

INPUTS: $x, y \in \{0, 1\}$
$c \in \{GREATER, EQUAL, LESS\}$

OUTPUT: $z \in \{GREATER, EQUAL, LESS\}$

FUNCTION: $$z = \begin{cases} GREATER & \textbf{if} \quad x > y \textbf{ or } (x = y \textbf{ and } c = GREATER) \\ EQUAL & \textbf{if} \quad x = y \textbf{ and } c = EQUAL \\ LESS & \textbf{if} \quad x < y \textbf{ or } (x = y \textbf{ and } c = LESS) \end{cases}$$

Example 6.1: Comparator (cont.)

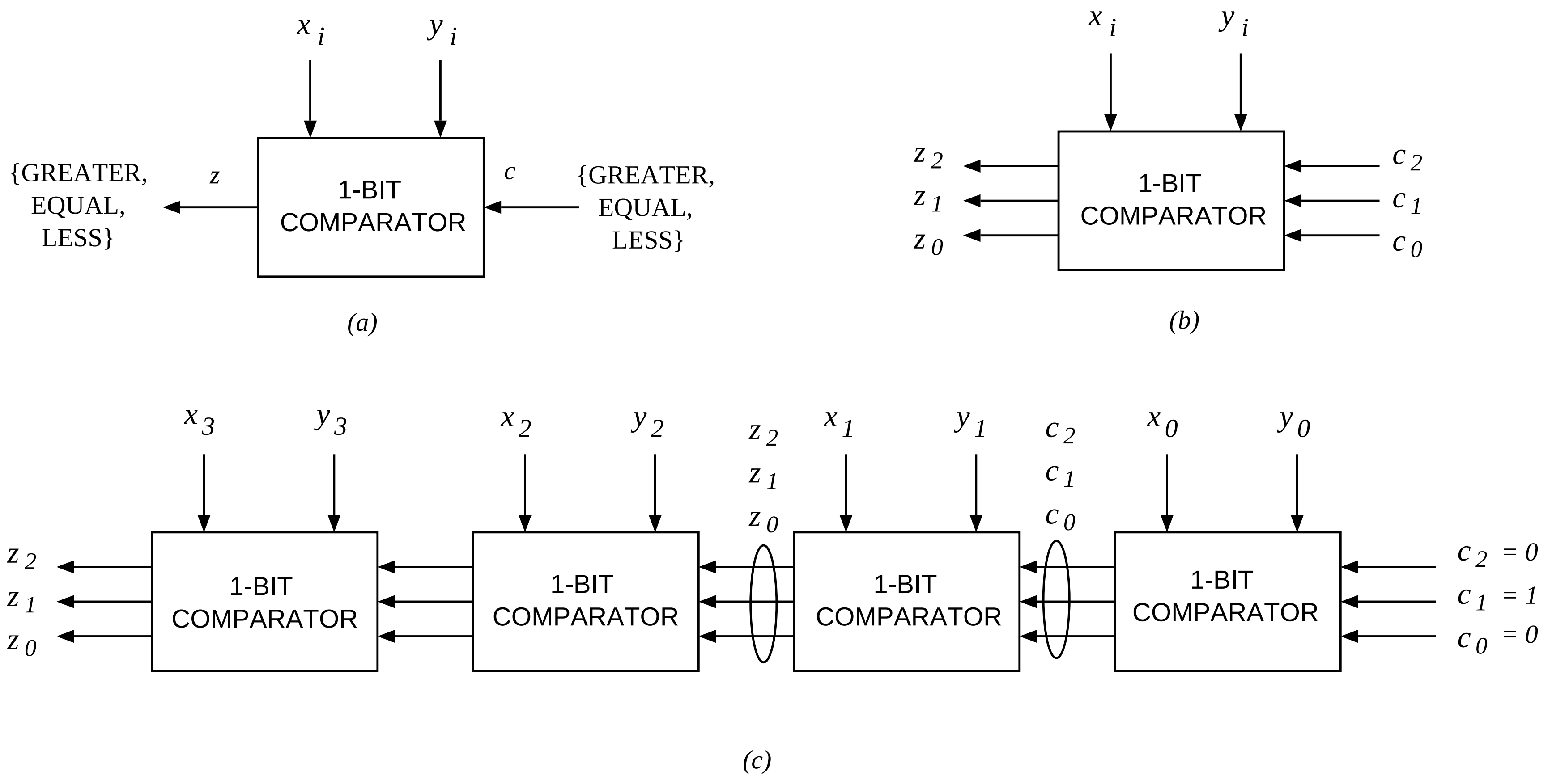


Figure 6.1: COMPARATOR

Example 6.1: Comparator (cont.)

CODING:

c	c_2	c_1	c_0
z	z_2	z_1	z_0
$GREATER$	1	0	0
$EQUAL$	0	1	0
$LESS$	0	0	1

		x, y			
		00	01	10	11
	100	100	001	100	100
c	010	010	001	100	010
	001	001	001	100	001
		z			

Example 6.1: Comparator (cont.)

- SWITCHING EXPRESSIONS:

$$
\begin{aligned}
z_2 &= xy' + xc_2 + y'c_2 \quad G \\
z_1 &= (x' + y)(x + y')c_1 \quad E \\
z_0 &= x'y + x'c_0 + yc_0 \quad S
\end{aligned}
$$

- RESULTING TWO-LEVEL NETWORK:
  - 7 AND and 4 OR gates
  - 22 equivalent gates
  - 25 gate inputs

REDUCING NETWORK SIZE (cont.)

DEFINE:

$$\begin{aligned} t &= (x + y') \\ w &= (x' + y) \\ z_2 &= xy' + tc_2 \\ z_1 &= twc_1 \\ z_0 &= x'y + wc_0 \end{aligned}$$

- SIZE: 18 EQUIVALENT GATES
- FURTHER REDUCTION: NAND NETWORK – 9 EQUIVALENT GATES

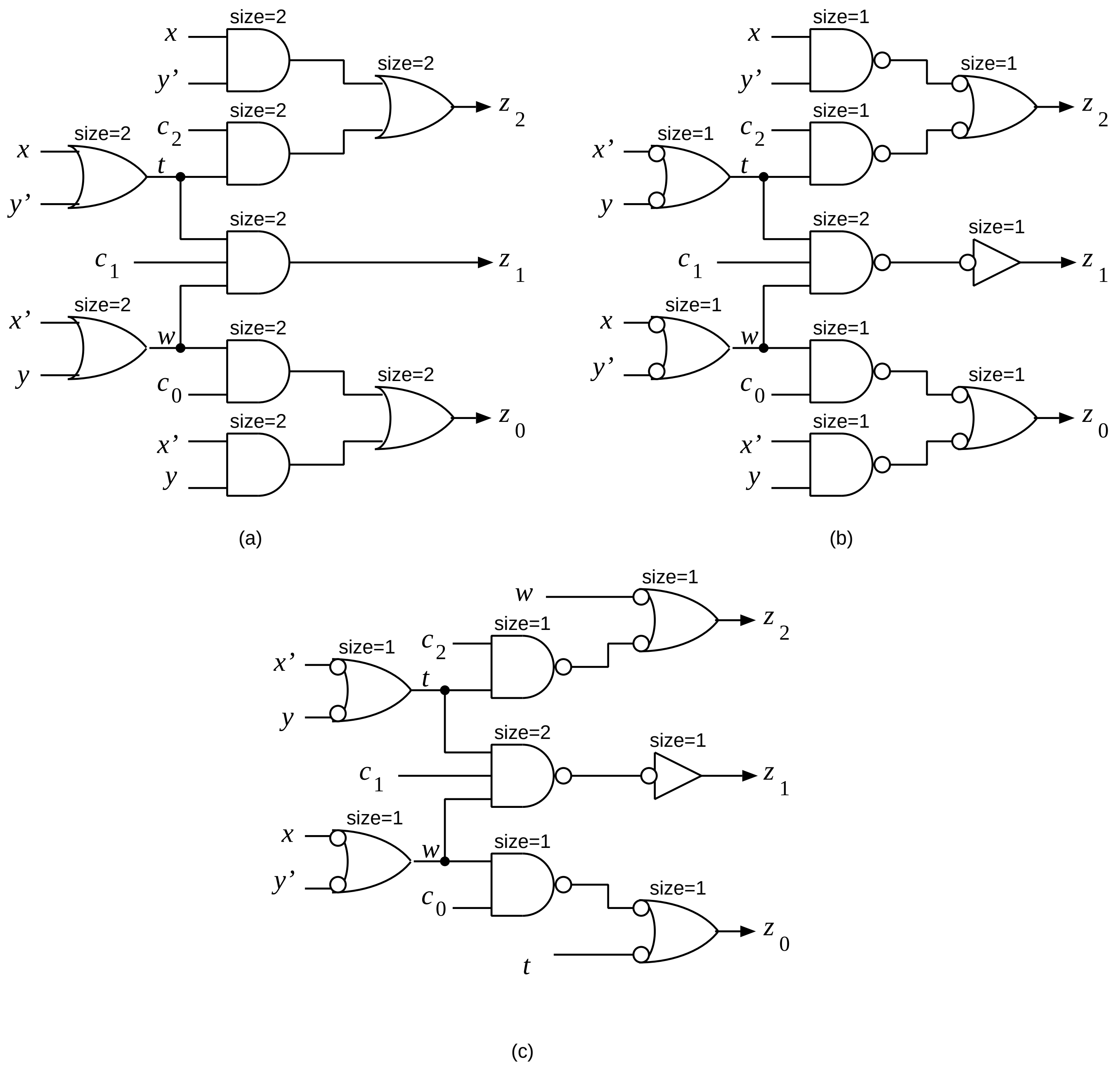


Figure 6.2: 1-BIT COMPARATOR IMPLEMENTATIONS

EXAMPLE 6.2: MODULO-64 INCREMENTER

- A TWO-LEVEL IMPLEMENTATION:

$$\begin{aligned}
z_5 &= x_5x_4' + x_5x_3' + x_5x_2' + x_5x_1' + x_5x_0' + x_5'x_4x_3x_2x_1x_0\\
z_4 &= x_4x_3' + x_4x_2' + x_4x_1' + x_4x_0' + x_4'x_3x_2x_1x_0\\
z_3 &= x_3x_2' + x_3x_1' + x_3x_0' + x_3'x_2x_1x_0\\
z_2 &= x_2x_1' + x_2x_0' + x_2'x_1x_0\\
z_1 &= x_1x_0' + x_1'x_0\\
z_0 &= x_0'
\end{aligned}$$

- TWO-LEVEL NETWORK:

7 NOT 20 AND , 5 OR gates, and 77 gate inputs

FACTORING

$$\begin{aligned}
z_5 &= x_5(x_4' + x_3' + x_2' + x_1' + x_0') + x_5'x_4x_3x_2x_1x_0 \\
z_4 &= x_4(x_3' + x_2' + x_1' + x_0') + x_4'x_3x_2x_1x_0 \\
z_3 &= x_3(x_2' + x_1' + x_0') + x_3'x_2x_1x_0 \\
z_2 &= x_2(x_1' + x_0') + x_2'x_1x_0 \\
z_1 &= x_1x_0' + x_1'x_0 \\
z_0 &= x_0'
\end{aligned}$$

- FOUR-LEVEL NETWORK (NOT-OR-AND-OR):

7 NOT 10 AND and 9 OR gates, and 61 gate inputs

4-LEVEL IMPLEMENTATION OF MODULO-64 INCREMENTER

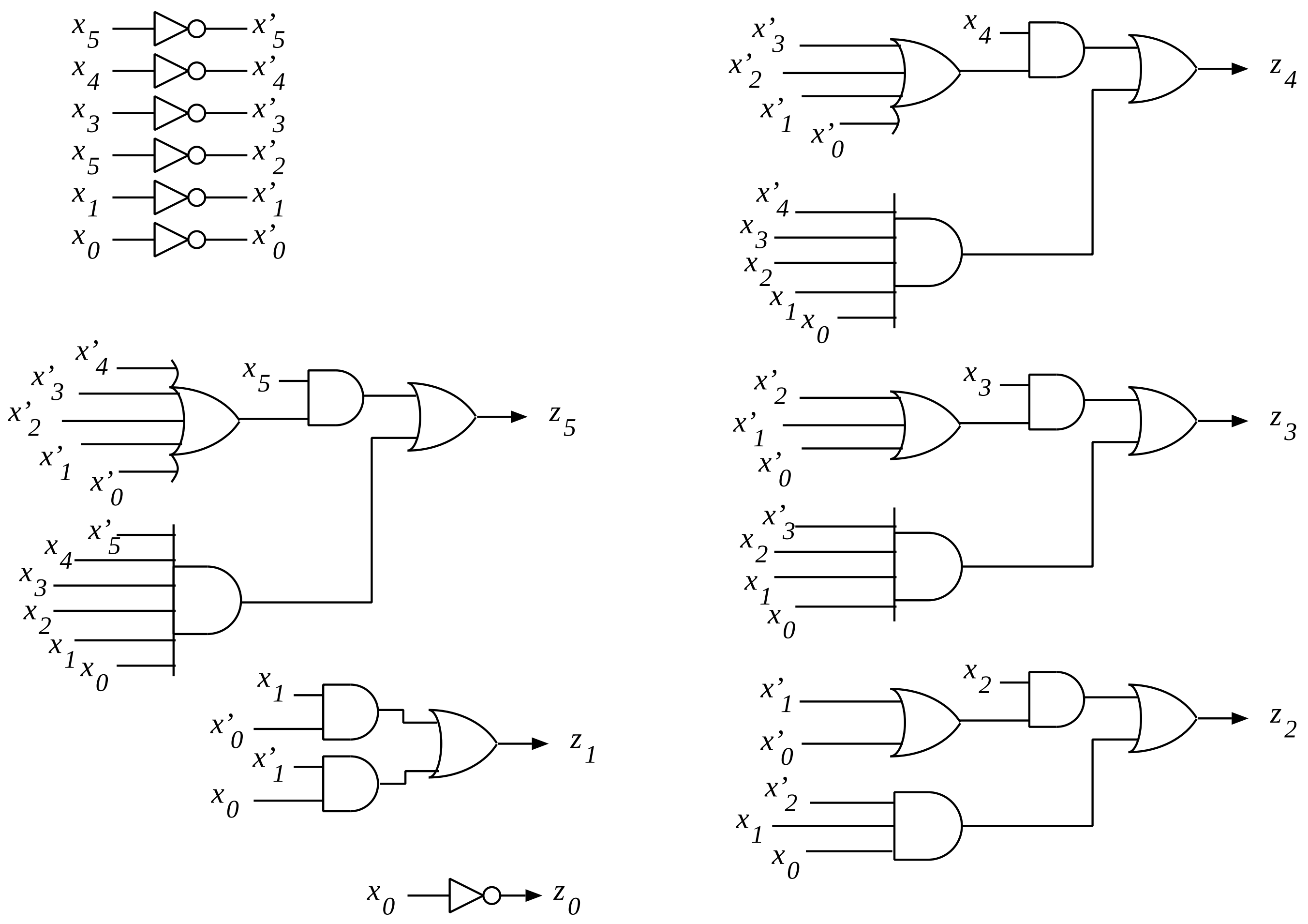


Figure 6.3: FOUR-LEVEL NETWORK FOR MODULO-64 INCREMENTER.

THE FAN-IN OF GATES

- FAN-IN OF GATES $\Leftrightarrow$ NUMBER OF OPERANDS PER OPERATOR
- REDUCED BY DECOMPOSING A LARGE GATE INTO SEVERAL SMALLER GATES
- AND AND OR ARE ASSOCIATIVE,

$$a + b + c + d + e + f = (a + b + c) + (d + e + f)$$

INCREMENTER WITH MAX FAN-IN OF 3

TERMS TO DECOMPOSE:

$$(x_4' + x_3' + x_2' + x_1' + x_0') = (x_4' + x_3' + r_{210})$$
$$(x_5' x_4 x_3 x_2 x_1 x_0) = x_5' a_{43} a_{210}$$
$$(x_3' + x_2' + x_1' + x_0') = x_3' + r_{210}$$
$$(x_4' x_3 x_2 x_1 x_0) = x_4' x_3 a_{210}$$

$$\begin{aligned}
z_5 &= x_5(x_4' + x_3' + r_{210}) + x_5' a_{43} a_{210} \\
z_4 &= x_4(x_3' + r_{210}) + x_4' x_3 a_{210} \\
z_3 &= x_3 r_{210} + x_3' a_{210} \\
z_2 &= x_2(x_1' + x_0') + x_2' x_1 x_0 \\
z_1 &= x_1 x_0' + x_1' x_0 \\
z_0 &= x_0'
\end{aligned}$$

- MORE GATES AND MORE LEVELS:
  6 NOT, 18 NAND, 3 NOR, size: 31 equivalent gates

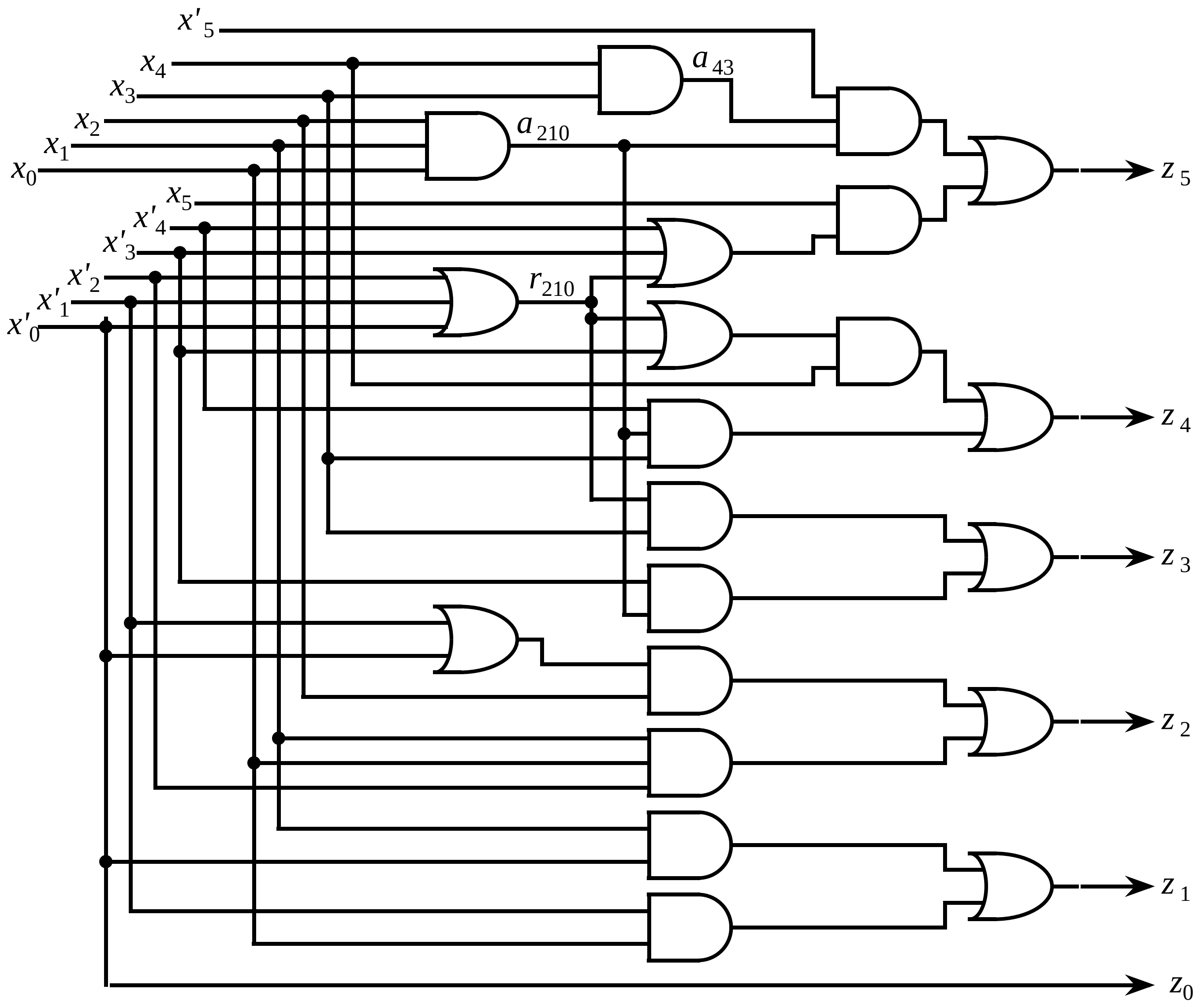


Figure 6.4: REDUCING THE NUMBER OF GATE INPUTS

EXAMPLE 6.4: REDUCING OUTPUT LOAD OF A GATE (Buffering)

$$z_i = w\ x\ y_i \qquad 0 \leq i \leq 63$$

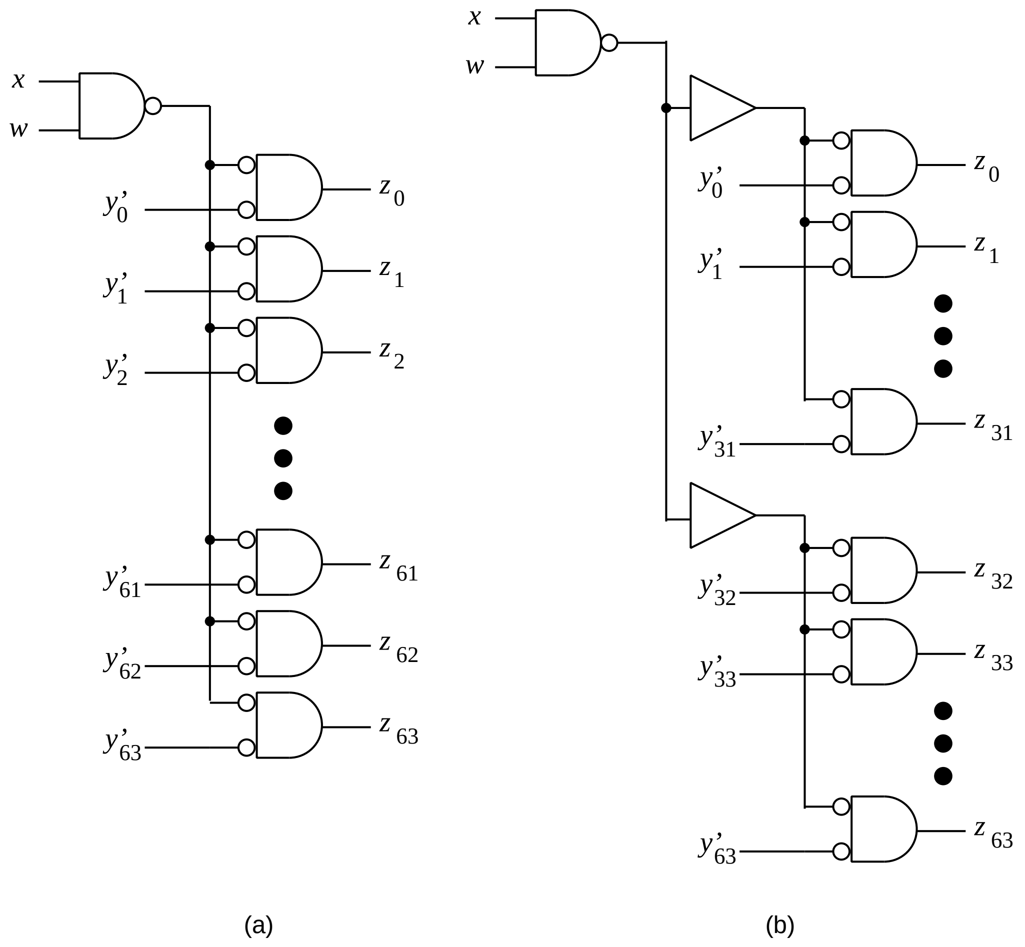


Figure 6.5: REDUCING THE OUTPUT LOAD

- OUTPUT LOAD OF NAND PRODUCING $w \cdot x$: 64I (I is load factor of NOR gate)

- PROPAGATION DELAY (high to low) between x and z_i (load 5 at output):

$$(0.05 + 0.038 \times 64) + (0.07 + 0.016 \times 5) = 2.63ns$$

- USE BUFFERS

Gate type	Fan-in	Propagation delays		Input factor	Size
		t_{pLH} [ns]	t_{pHL} [ns]	[Standard loads]	[equiv. [gates]
Buffer	1	$0.15 + 0.006L$	$0.19 + 0.003L$	2	4
Inv. Buf.	1	$0.04 + 0.006L$	$0.05 + 0.006L$	4.7	3

- DELAY:

$$(0.05 + 0.038 \times 4) + (0.15 + 0.006 \times 32) + (0.07 + 0.016 \times 5) = 0.69ns$$

EXAMPLE 6.5: EVEN PARITY CIRCUIT – alternatives

INPUT: $\underline{x} = (x_7, x_6, \ldots, x_0), \quad x_i \in \{0,1\}$

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

FUNCTION: $$z = \begin{cases} 1 & \textbf{if} \quad \Sigma_{i=0}^{7} x_i \text{ is even} \\ 0 & \textbf{otherwise} \end{cases}$$

IMPLEMENTATION 1: TWO-LEVEL NETWORK.

CSP: 128 MINTERMS – NO REDUCTION POSSIBLE

COST: 128 AND gates and one OR gate

EACH AND GATE 8 INPUTS, OR GATE 128 INPUTS

NOT PRACTICAL: large number of gates, large fan-in

IMPLEMENTATION 2: DIVIDE INTO TWO PARTS

$$P(\underline{x}) = P(\underline{x}_l)P(\underline{x}_r) + P'(\underline{x}_l)P'(\underline{x}_r)$$

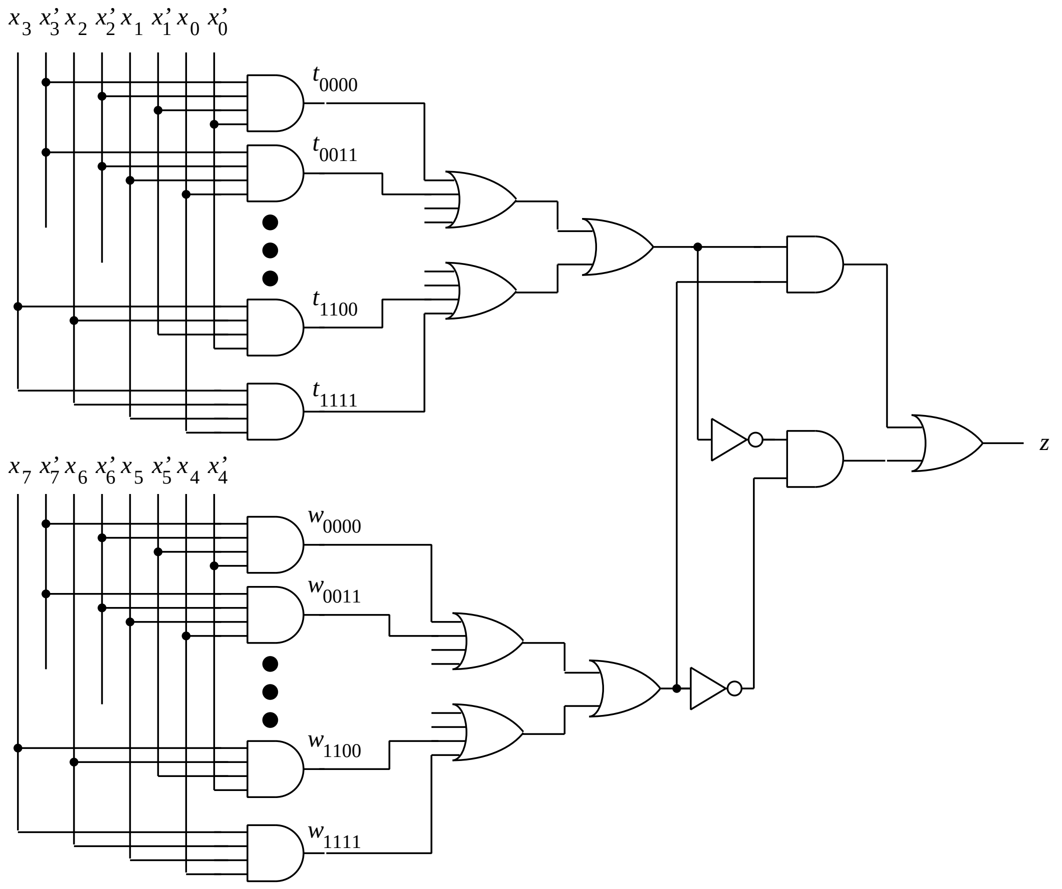

x_3 x_3' x_2 x_2' x_1 x_1' x_0 x_0'
t_{0000}
t_{0011}
t_{1100}
t_{1111}
x_7 x_7' x_6 x_6' x_5 x_5' x_4 x_4'
w_{0000}
w_{0011}
w_{1100}
w_{1111}
z

Figure 6.6: NETWORK WITH FAN-IN=4

SUMMARY OF ALTERNATIVE IMPLEMENTATIONS OF PARITY FUNCTION

Table 6.2: CHARACTERISTICS OF ALTERNATIVE IMPLEMENTATIONS FOR THE PARITY FUNCTION

Impl.	Network input load	Gates				No. levels
		Type	Fan-in	Fan-out	Number	
1	64	AND	8	1	128	2
		OR	128	-	1	
2	4	AND	4	1	16	6
		OR	4	1	4	
		OR	2	1	3	
		AND	2	1	2	
		NOT	1	1	2	

NETWORKS WITH XOR AND XNOR GATES

EXAMPLE 6.6: 8-INPUT ODD-PARITY CHECKER

INPUT: $\underline{x} = (x_7, \ldots, x_0), x_i \in \{0,1\}$
OUTPUT: $z \in \{0,1\}$

FUNCTION: $z = \begin{cases} 0 & \text{if number of } 1'\text{s in } \underline{x} \text{ is even} \\ 1 & \text{if number of } 1'\text{s in } \underline{x} \text{ is odd} \end{cases}$

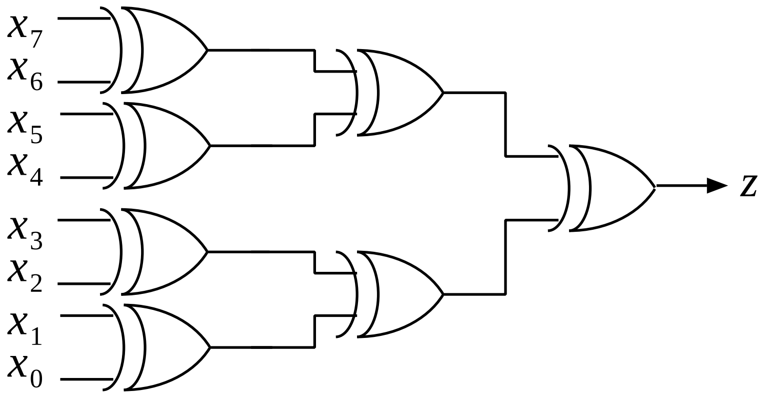


Figure 6.7: ODD-PARITY CHECKER

$$z = x_7 \oplus x_6 \oplus x_5 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0$$

EXAMPLE: 32-BIT EQUALITY COMPARATOR

INPUT: $\underline{x} = (x_{31}, \ldots, x_0), x_i \in \{0,1\}$

$\underline{y} = (y_{31}, \ldots, y_0), y_i \in \{0,1\}$

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

FUNCTION: $z = \begin{cases} 1 \ \textbf{if} \ \ x_i = y_i \ \ \textbf{for} \ \ 0 \le i \le 31 \\ 0 \ \textbf{otherwise} \end{cases}$

$$z = AND(XNOR(x_{31}, y_{31}), \ldots, \ XNOR(x_i, y_i), \ldots, XNOR(x_0, y_0))$$

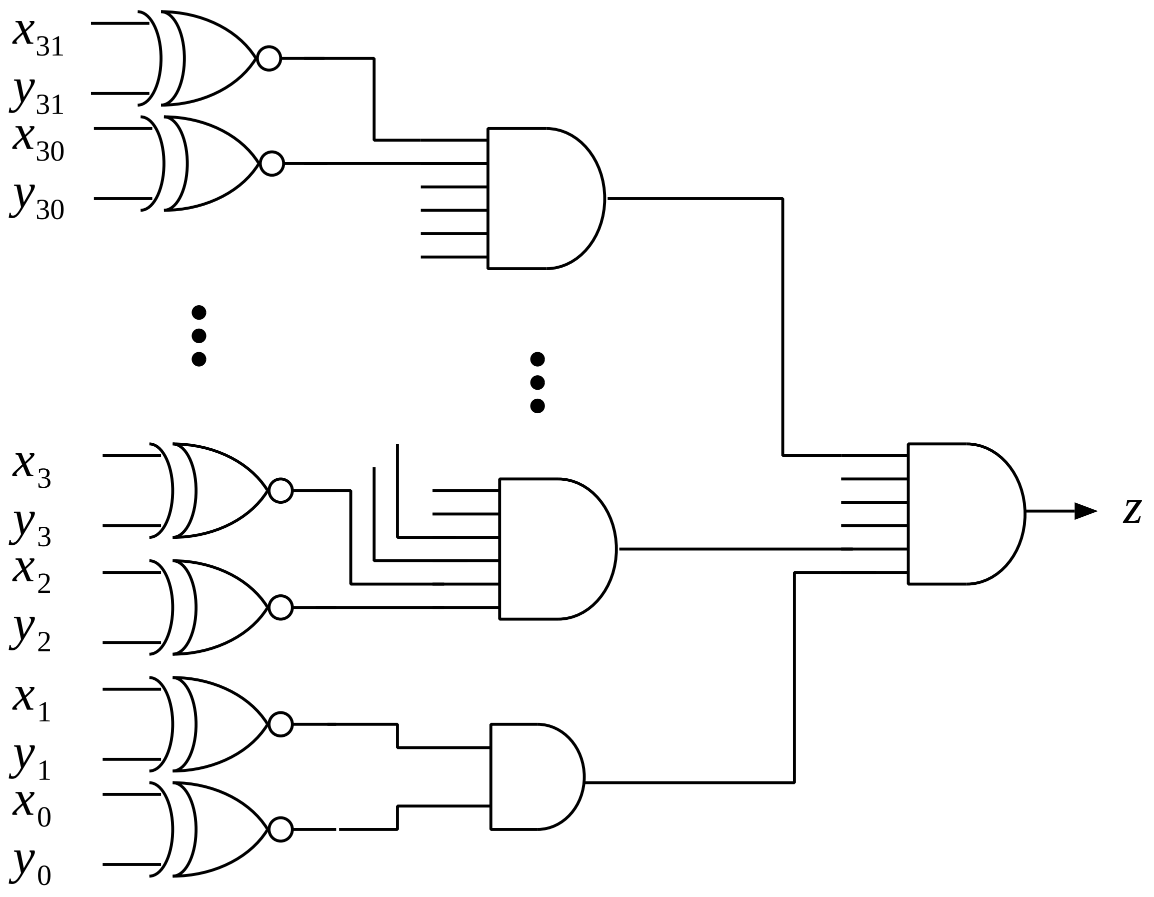


Figure 6.8: 32-BIT EQUALITY COMPARATOR

NETWORKS WITH 2-INPUT MULTIPLEXERS

- 2-INPUT *multiplexer* (MUX): $z = MUX[x_1, x_0, s] = x_1 \cdot s + x_0 s'$

- SET $\{$MUX $\}$ IS UNIVERSAL (constants 0 and 1 available)

$$\begin{aligned} NOT(x) &= MUX[0,1,x] = 0 \cdot x + 1 \cdot x' = x' \\ AND(x_1, x_0) &= MUX[x_1, 0, x_0] = x_1 x_0 + 0 \cdot x_0' = x_1 x_0 \end{aligned}$$

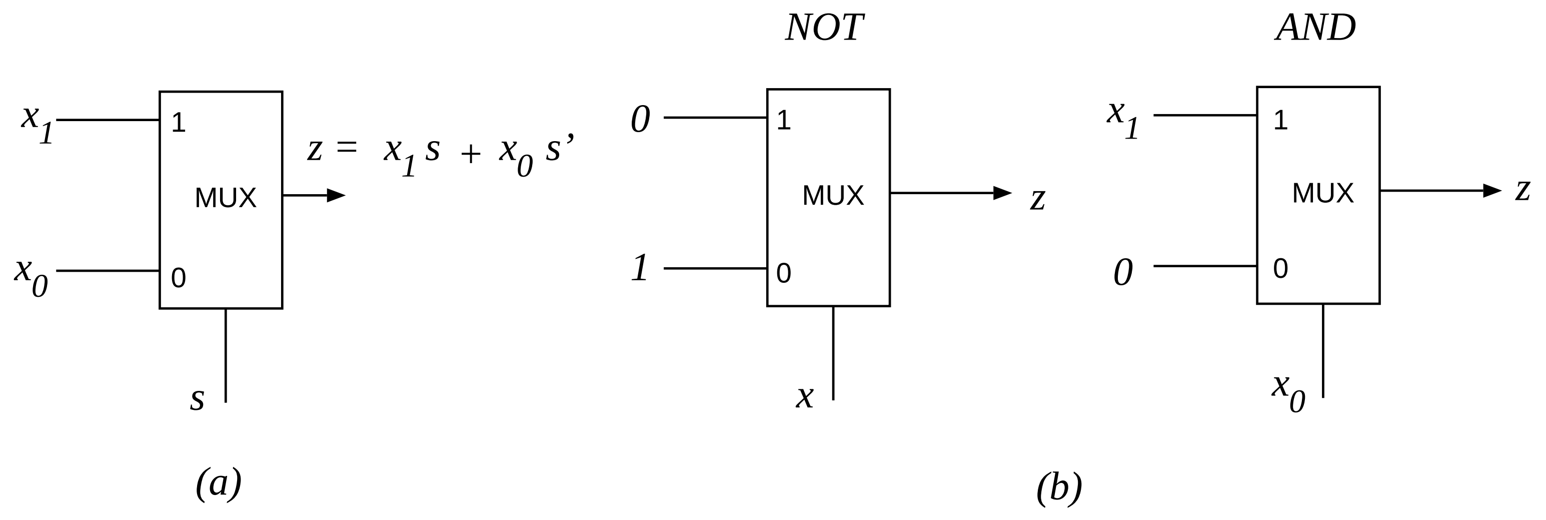


Figure 6.9: 2-INPUT MULTIPLEXER AND NOT and AND GATES

IMPLEMENTATION OF SFs WITH NETWORK OF MUXes

- *SHANNON'S DECOMPOSITION* (SD)

$$\begin{aligned} f(x_{n-1},x_{n-2},\ldots,x_0) &= f(x_{n-1},x_{n-2},\ldots,1)\cdot x_0 \\ &+ f(x_{n-1},x_{n-2},\ldots,0)\cdot x_0' \end{aligned}$$

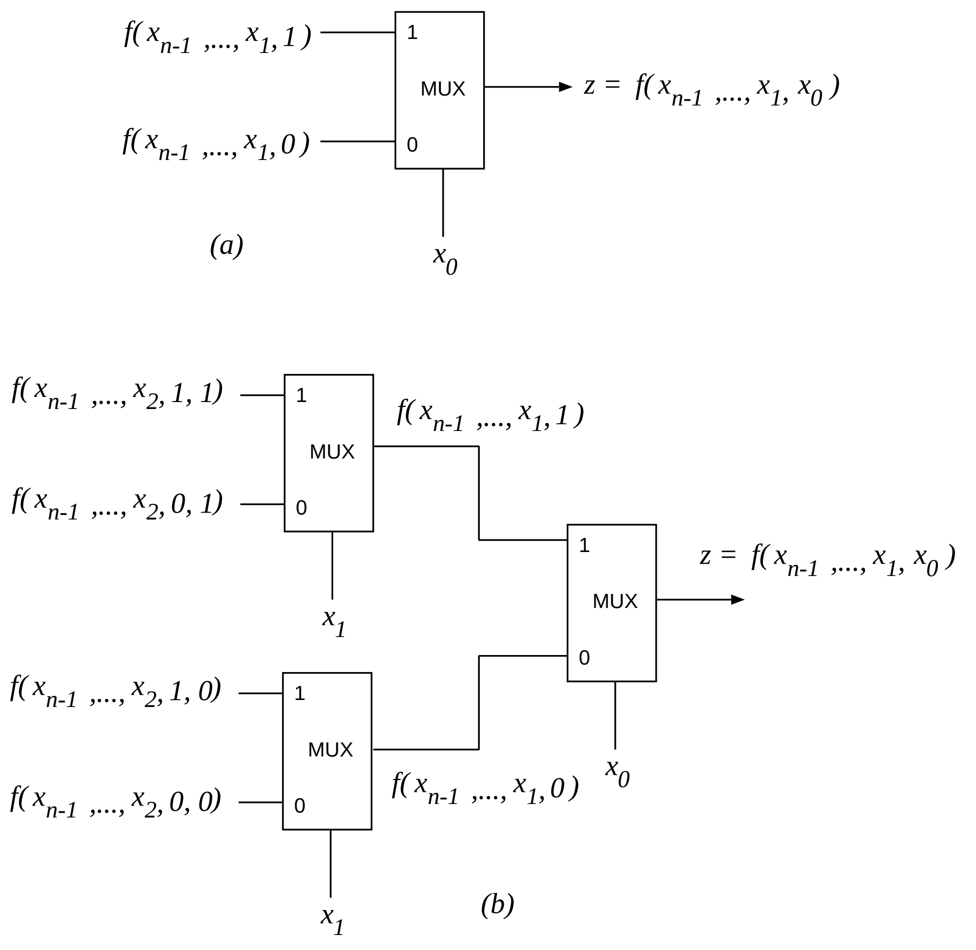
$$\begin{aligned} z &= f(x_{n-1},x_{n-2},\ldots,x_0) \\ &= MUX[f(x_{n-1},x_{n-2},\ldots,x_1,1),\ f(x_{n-1},x_{n-2},\ldots,x_1,0),\ x_0] \end{aligned}$$

EXAMPLE:

$$z = x_3(x_2+x_0)x_1 = MUX[x_3x_1,\ x_3x_2x_1,\ x_0]$$

DESIGN OF NETWORKS WITH MUXes

- OBTAIN A TREE OF MULTIPLEXERS BY REPEATED USE OF SD



$f(x_{n-1},\ldots,x_1,1)$
$f(x_{n-1},\ldots,x_1,0)$
1
MUX
0
$z = f(x_{n-1},\ldots,x_1,x_0)$
x_0
(a)
$f(x_{n-1},\ldots,x_2,1,1)$
$f(x_{n-1},\ldots,x_2,0,1)$
$f(x_{n-1},\ldots,x_2,1,0)$
$f(x_{n-1},\ldots,x_2,0,0)$
$f(x_{n-1},\ldots,x_1,1)$
$f(x_{n-1},\ldots,x_1,0)$
x_1
x_0
$z = f(x_{n-1},\ldots,x_1,x_0)$
(b)

Figure 6.10: a) REALIZATION OF SHANNON'S DECOMPOSITION WITH MULTIPLEXER; b) REPEATED DECOMPOSITION.

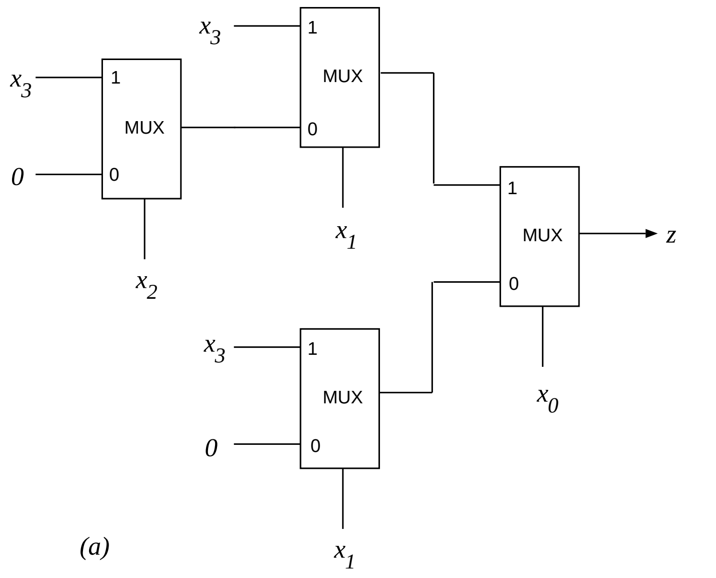
Example 6.8

- IMPLEMENT $f(x_3, x_2, x_1, x_0) = z = x_3(x_1 + x_2x_0)$ WITH MUX TREE
  - DECOMPOSE WITH RESPECT TO x_2, x_1, x_0

$$\begin{array}{ll}
f(x_3, 0, 0, 0) = 0 & f(x_3, 0, 0, 1) = 0 \\
f_{(}x_3, 0, 1, 0) = x_3 & f(x_3, 0, 1, 1) = x_3 \\
f_{(}x_3, 1, 0, 0) = 0 & f(x_3, 1, 0, 1) = x_3 \\
f_{(}x_3, 1, 1, 0) = x_3 & f(x_3, 1, 1, 1) = x_3
\end{array}$$

  - ELIMINATE REDUNDANT MUXes

ORDERING OF VARIABLES IN SUBTREES AFFECTS THE NUMBER OF MUXes



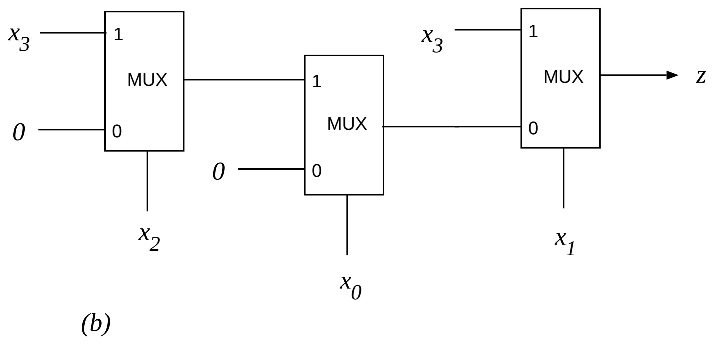


Figure 6.11: