DESCRIPTION AND ANALYSIS OF GATE NETWORKS

- GATE NETWORKS: DEFINITION
- SETS OF GATES: (AND OR NOT), NAND NOR XOR
- ANALYSIS AND DESCRIPTION OF GATE NETWORKS

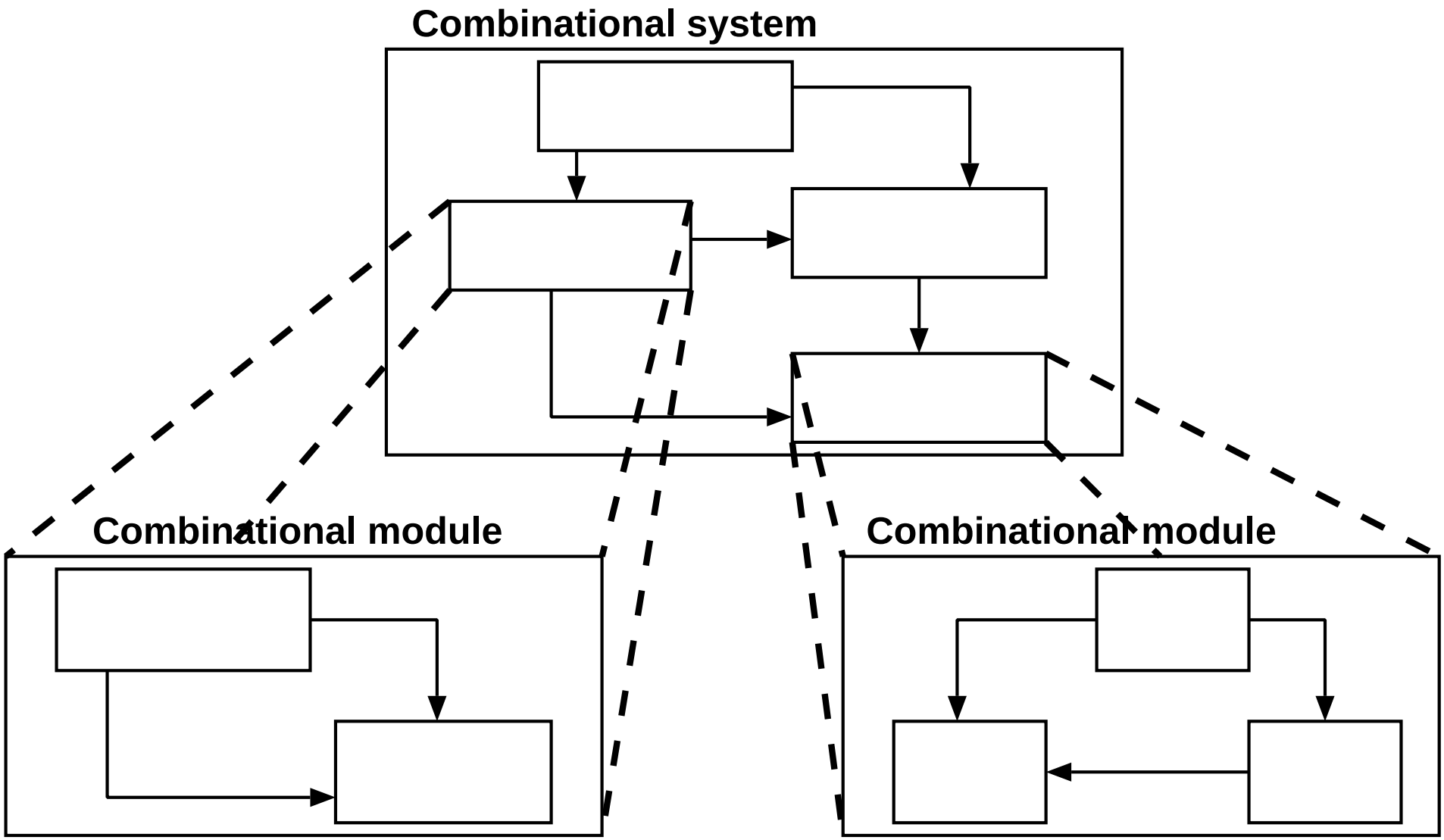


Figure 4.1: HIERARCHICAL IMPLEMENTATION OF A MODULE

GATE NETWORKS

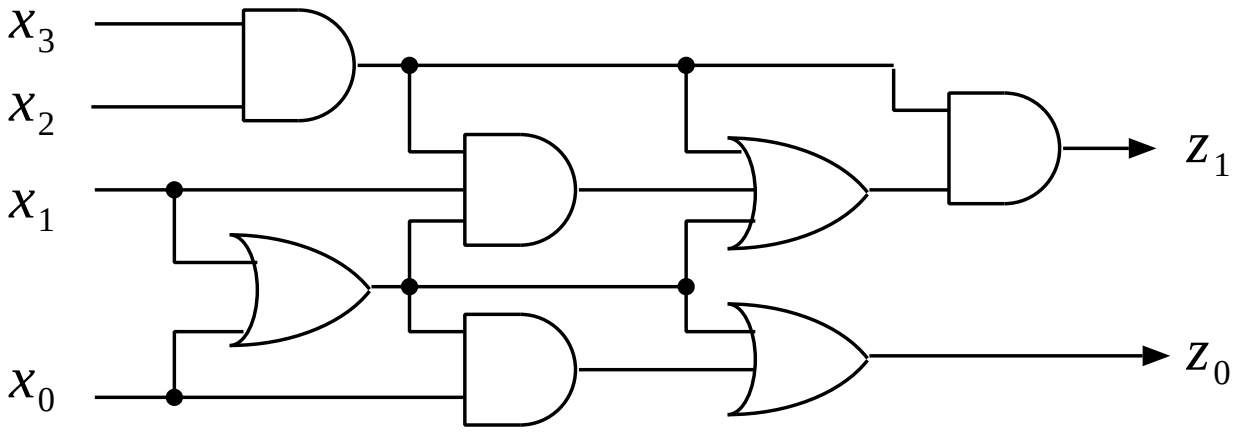


Figure 4.2: A GATE NETWORK

- GATES
- EXTERNAL INPUTS AND OUTPUTS
- CONNECTIONS

GATE NETWORKS (cont.)

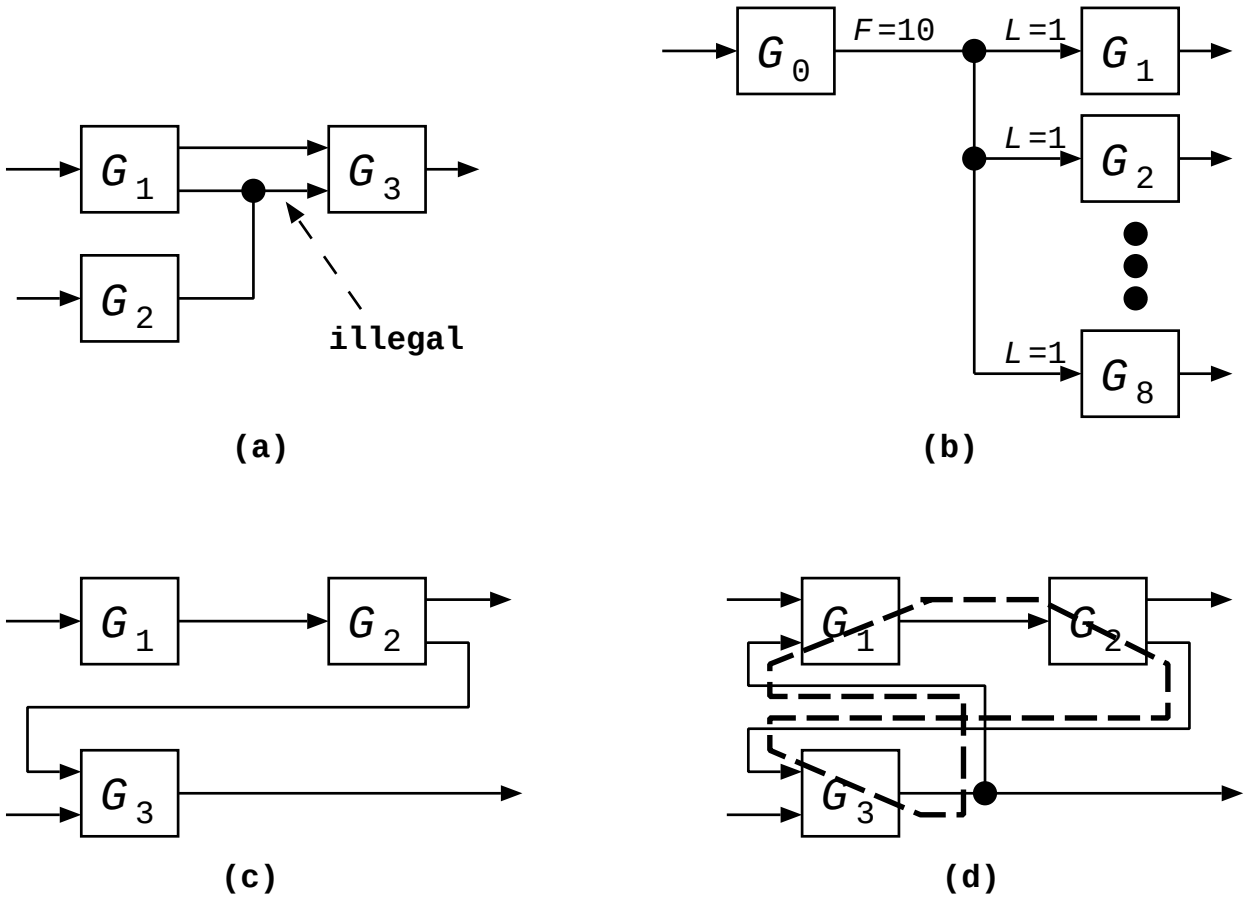


Figure 4.3: a) ILLEGAL NETWORK CONNECTION. b) ACCEPTABLE OUTPUT LOAD. c) LOOP-FREE NETWORK. d) LOOP NETWORK

DESCRIPTION OF GATE NETWORKS

- LOGIC DIAGRAM (GRAPHICAL REPRESENTATION)
- NET LIST (TABULAR REPRESENTATION)
- HDL DESCRIPTION (PROGRAM)

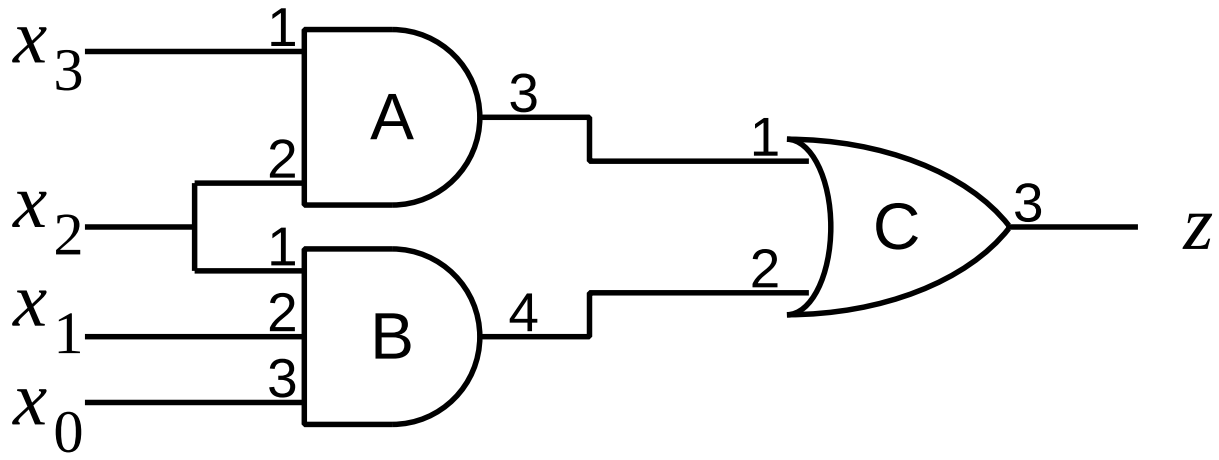


Figure 4.4: a) GRAPHICAL REPRESENTATION (LOGIC DIAGRAM)

Gate	Type	Inputs	Output
A	$\text{AND} - 2$	A_1	A_3
		A_2	
B	$\text{AND} - 3$	B_1	B_4
		B_2	
		B_3	
C	$\text{OR} - 2$	C_1	C_3
		C_2	

Gates

From	To
x_3	A_1
x_2	A_2
x_2	B_1
x_1	B_2
x_0	B_3
A_3	C_1
B_4	C_2
C_3	z

Connections

(b)

```
A_3 <= x3 and x2;
B_4 <= x2 and x1 and x0;
C_3 <= A_3 or B_4;
z   <= C_3;
```

(c)

CHARACTERIZATION OF A GATE NETWORK

- FUNCTIONAL SPECIFICATION
- INPUT LOAD FACTORS OF THE NETWORK INPUTS;
- FAN-OUT FACTOR OF THE NETWORK OUTPUTS (ONLY FOR SOME TECHNOLOGIES); AND
- PROPAGATION DELAYS THROUGH THE NETWORK

UNIVERSAL SETS OF GATES

- Set {AND,OR,NOT}

$$z = (((x_0 + x_1)x_2)' + x_2x_3 + x_4)'$$

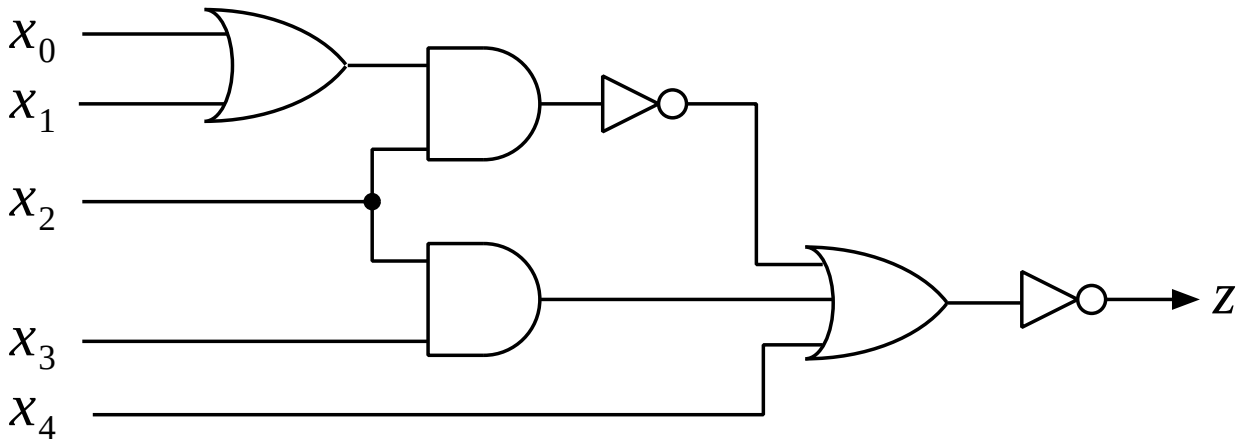


Figure 4.4: CORRESPONDENCE AMONG SWITCHING EXPRESSION ANd AND-OR-NOT NETWORK

UNIVERSAL SETS OF GATES (cont.)

- Sets {AND,NOT} and {OR,NOT}

$$x_{n-1} + x_{n-2} + \ldots x_i + \ldots x_0 = (x'_{n-1}x'_{n-2}\ldots x'_i \ldots x'_0)'$$

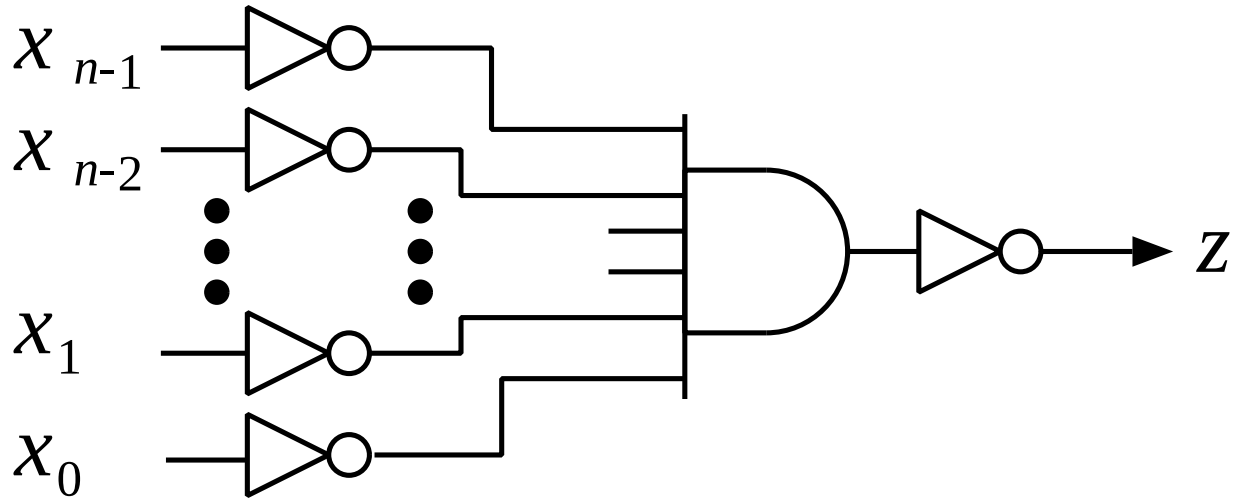


Figure 4.5: AND-NOT IMPLEMENTATION OF AN OR GATE

UNIVERSAL SETS OF GATES (cont.)

- Sets {NAND} and {NOR}

$$x' = (xx)'$$

$$NOT(x) = NAND(x, x)$$

$$x_1x_0 = ((x_1x_0)')' = ((x_1x_0)'(x_1x_0)')'$$

$$AND(x_1, x_0) = NAND(NAND(x_1, x_0), NAND(x_1, x_0))$$

UNIVERSAL SETS OF GATES (cont.)

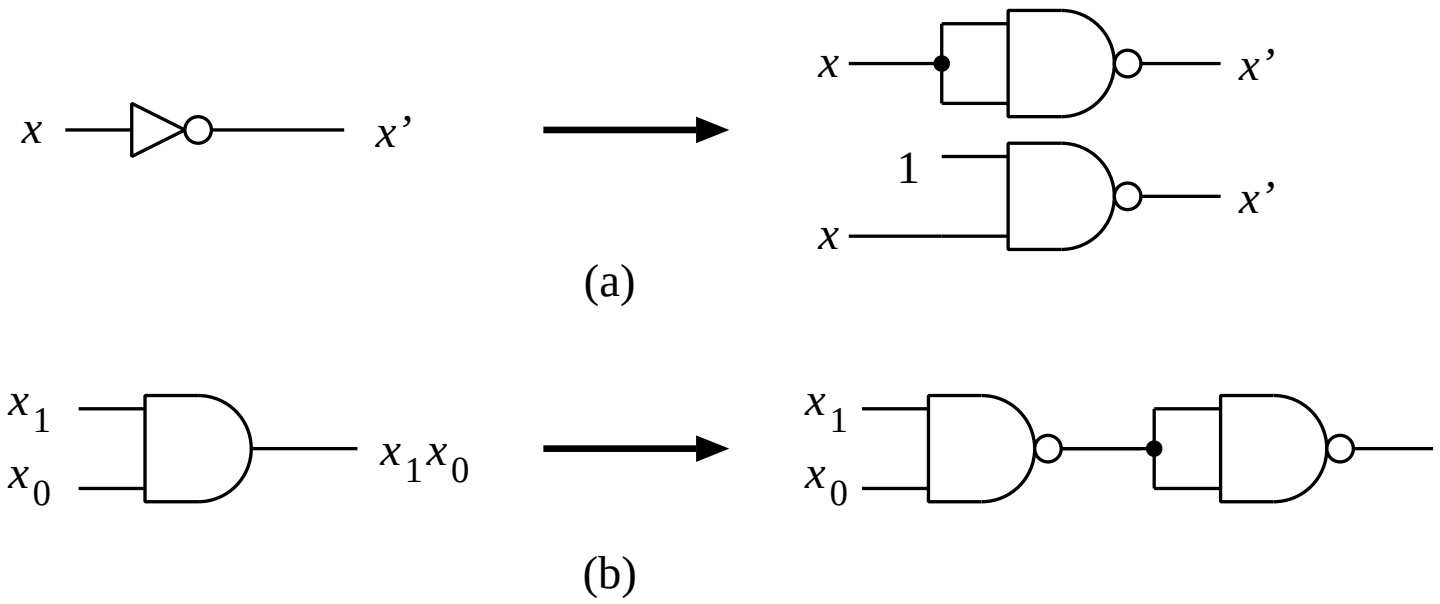


Figure 4.6: IMPLEMENTATIONS WITH NAND GATES: a) NOT; b) AND

EXAMPLE: SHOW THAT $*$, DEFINED AS

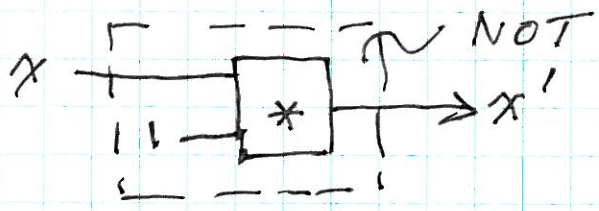
$x * y = x' y$, IS UNIVERSAL

ASSUMING "1" IS AVAILABLE

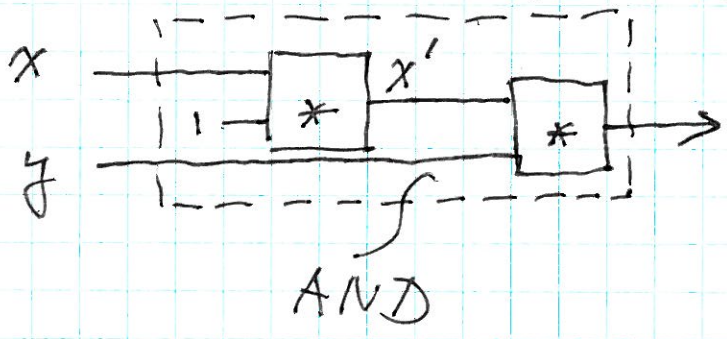
NOTE: $*$ IS <u>NOT</u> COMMUTATIVE, I.E., $x * y \neq y * x$

TO SHOW THAT $\{*, 1\}$ IS UNIVERSAL, SHOW HOW TO OBTAIN <u>NOT</u> AND <u>AND</u>. $\{$AND, NOT$\}$ IS UNIVERSAL

NOT: $x' = x \cdot 1 = x * 1$



AND: $x \cdot y = (x')' \cdot y = x' * y = (x * 1) * y$



SINCE $\{$AND, NOT$\}$ IS UNIVERSAL, $\{*, 1\}$ IS UNIVERSAL

MIXED-LOGIC NOTATION

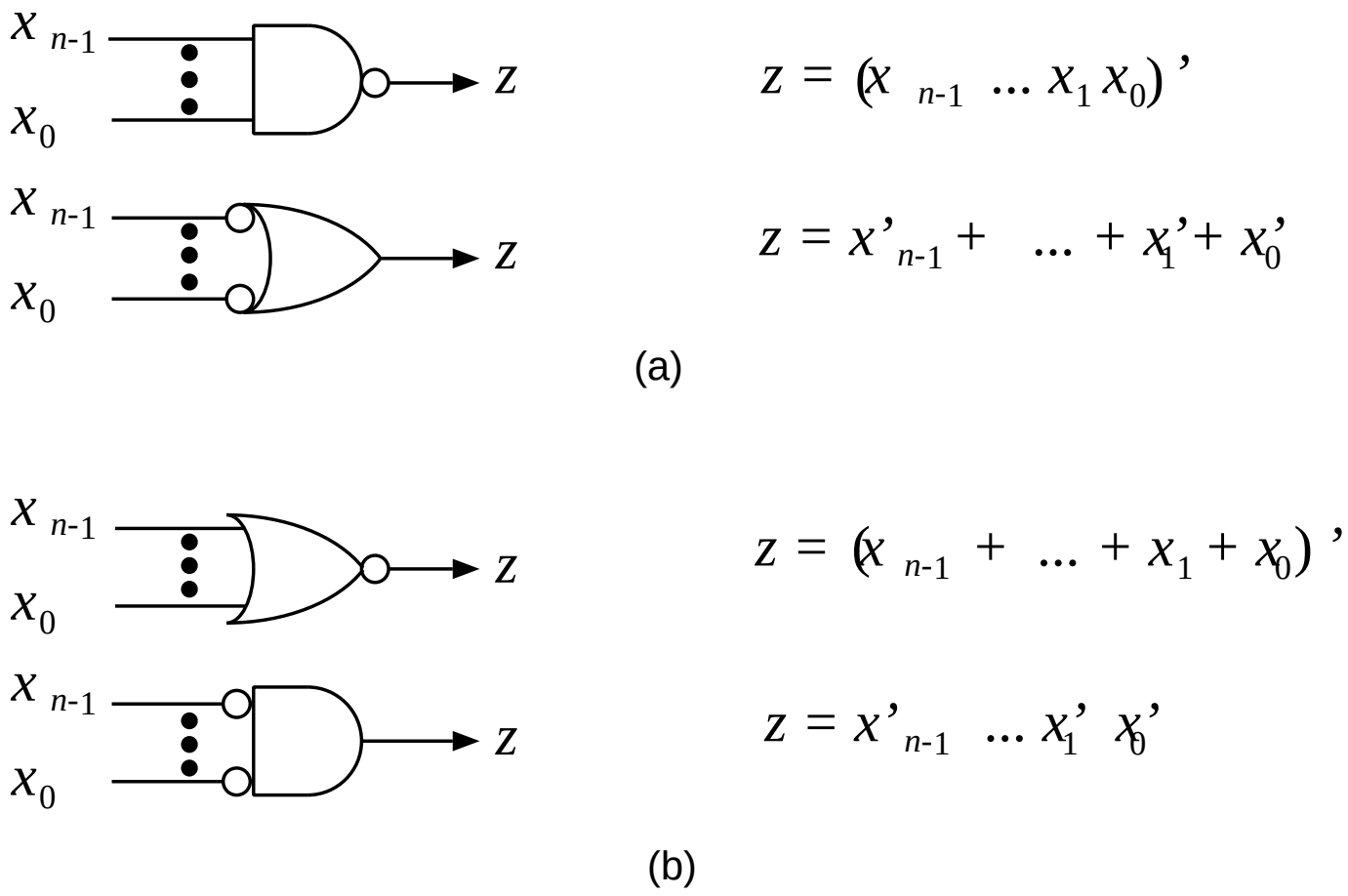


Figure 4.7: MIXED-LOGIC NOTATION: a) NAND GATE b) NOR GATE

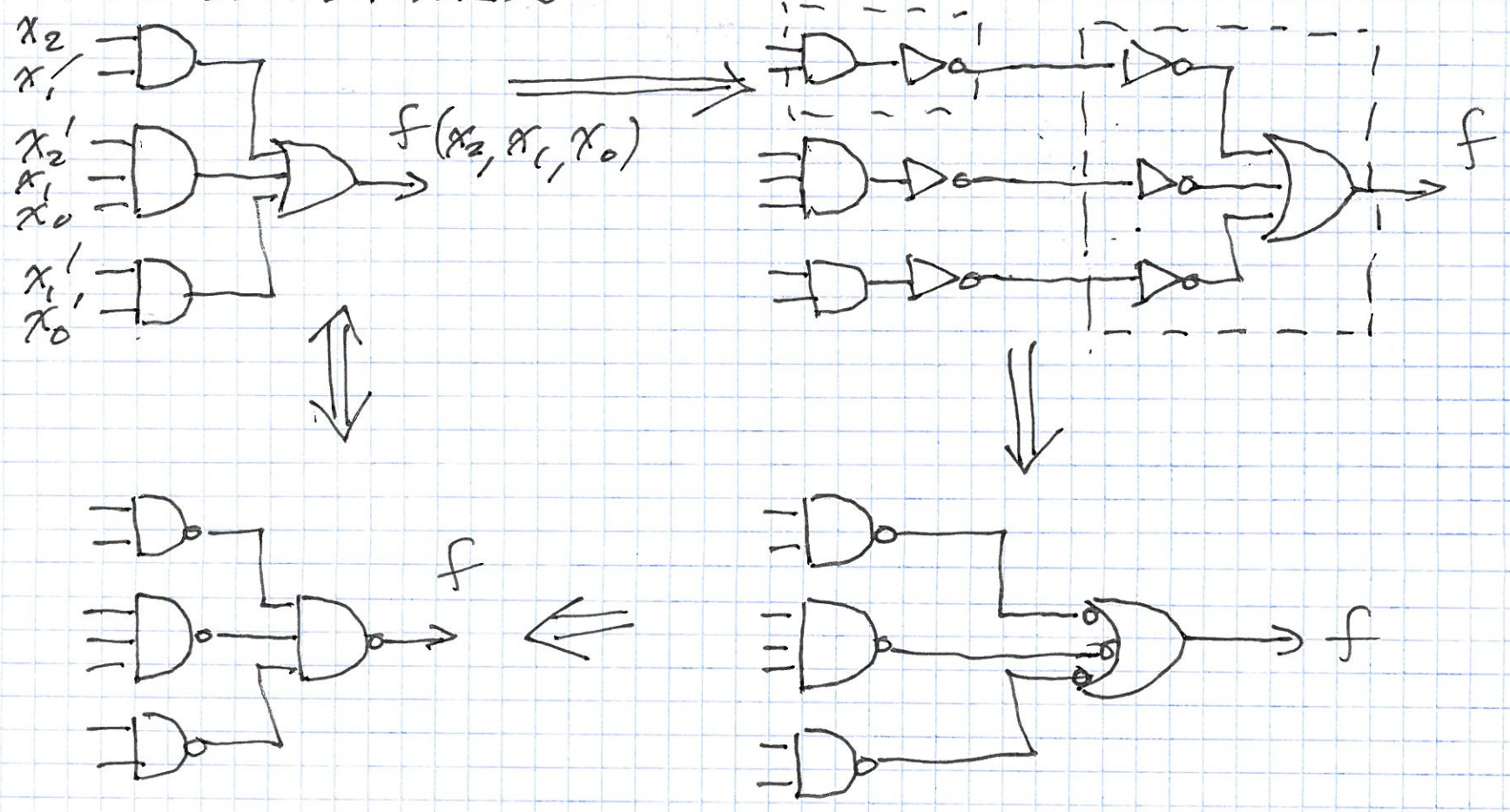
CS MSI A

{NAND} IS UNIVERSAL

— ANY SF HAS A SUM OF PRODUCTS EXPRESSION

EXAMPLE: $f(x_2, x_1, x_0) = x_2 x_1' + x_2' x_1 x_0 + x_1' x_0'$

AND-OR NETWORK



NAND NETWORK

MIXED-LOGIC NOTATION

COMPLEX GATE STRUCTURES

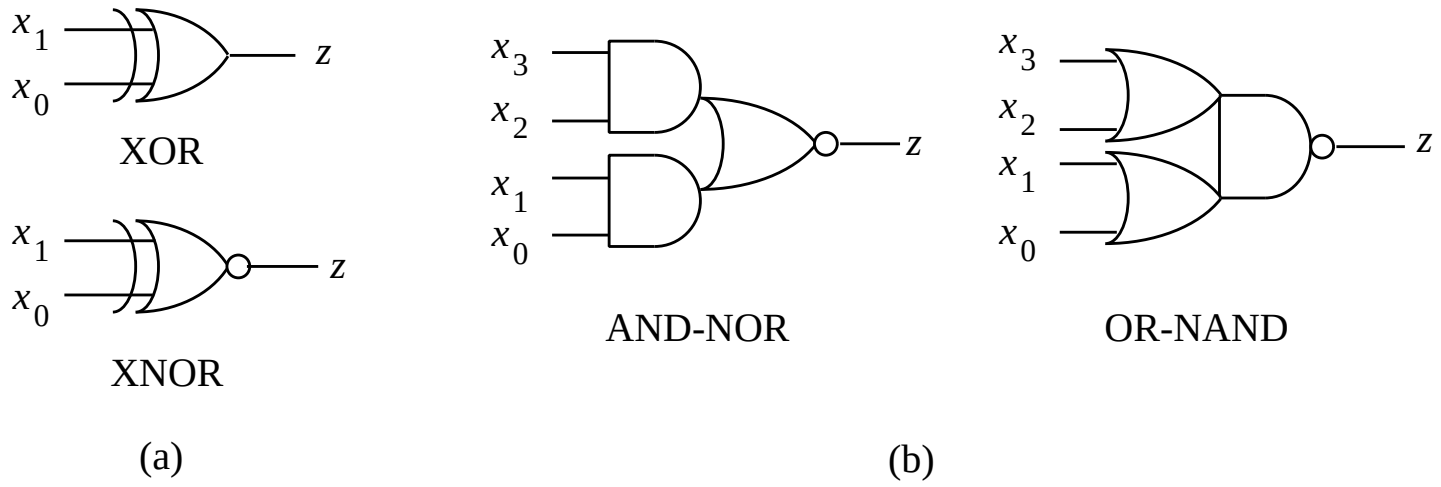


Figure 4.8: ADDITIONAL GATES IN CMOS a) XOR and XNOR, b) COMPLEX GATE STRUCTURES: AND-OR and OR-AND

XOR — A MOST INTERESTING GATE

$$x_1 \oplus x_0 = x_1' x_0 + x_1 x_0' = (x_1 + x_0) \bmod 2$$

↑ ARITH. SUM

SOME PROPERTIES:

(1) IF $x \oplus y = 0$ THEN $x = y$ ⇒ [CAN BE USED TO CHECK IF TWO BIT-VECTORS ARE THE SAME]

(2) IF $x \oplus y = z \oplus y$ THEN $x = z$

(3) $x' \oplus y = (x \oplus y)' = x \odot y$ (EQUIVALENCE)

(4) $x \oplus y = x' \oplus y'$

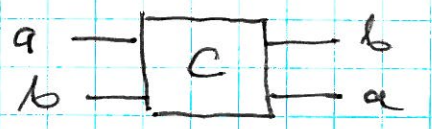
(5) $x \oplus y = y \oplus x$; $x \oplus y \oplus z = (x \oplus y) \oplus z = x \oplus (y \oplus z)$ [COMMUTATIVITY, ASSOCIATIVITY]

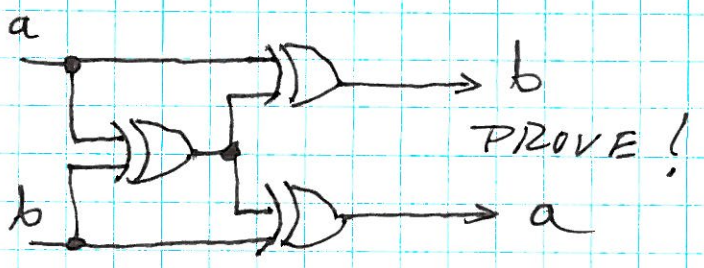
(6) $x \oplus 0 = x$; $x \oplus 1 = x'$; $x \oplus x = 0$; $x \oplus x \oplus x = x$

(7) $\underbrace{x \oplus x \oplus \ldots \oplus x}_{k \text{ TIMES}} = \begin{cases} 0 & \text{IF } k \text{ IS EVEN} \\ x & \text{IF } k \text{ IS ODD} \end{cases}$

(8) $x_{n-1} \oplus x_{n-2} \oplus \cdots \oplus x_0 = \begin{cases} 0 & \text{IF } \#(x_i = 1) \text{ IS EVEN} \\ 1 & \text{IF } \#(x_i = 1) \text{ IS ODD} \end{cases}$

PROBLEM: DESIGN A GATE NETWORK TO IMPLEMENT C SO THAT THERE ARE NO WIRE CROSSOVERS





PROVE!

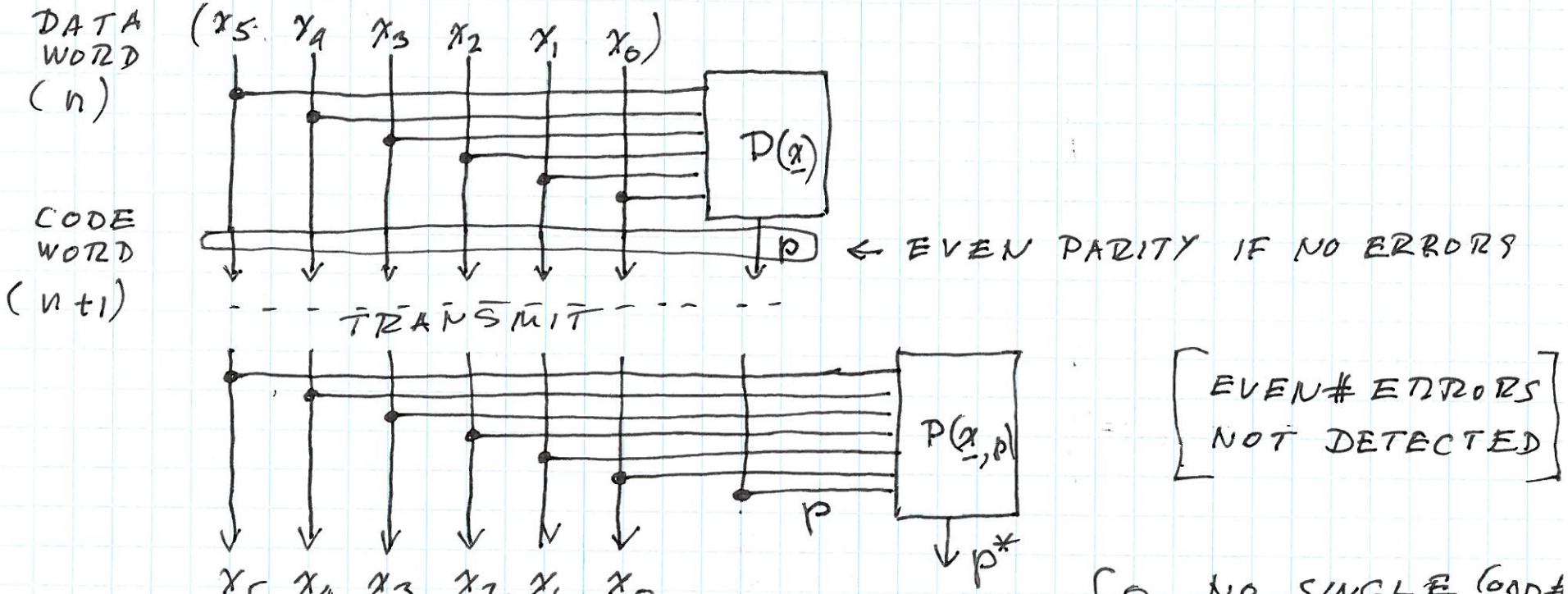
PARITY GENERATORS & DETECTORS

$$P(\underline{x}) = \begin{cases} 0 & \text{IF } \#(x_i = 1) \text{ IS EVEN} \\ 1 & \text{IF } \#(x_i = 1) \text{ IS ODD} \end{cases}$$

EXAMPLE: $\underline{x} = (x_5, x_4, x_3, x_2, x_1, x_0)$

$$P(\underline{x}) = x_5 \oplus x_4 \oplus x_3 \oplus x_2 \oplus x_1 \oplus x_0$$

PARITY CHECKING: DETECTS ODD NUMBER OF BIT ERRORS



[EVEN # ERRORS NOT DETECTED]

$$p^* = P(\underline{x}, p) = \begin{cases} 0 & \text{NO SINGLE (ODD #) ERRORS} \\ 1 & \text{ERROR} \end{cases}$$

ANALYSIS OF GATE NETWORKS

- FUNCTIONAL ANALYSIS:
  1. OBTAIN I/O SWITCHING EXPRESSIONS
  2. OBTAIN A TABULAR REPRESENTATION OF THE (BINARY) FUNCTION (IF FEW VARIABLES)
  3. DEFINE HIGH-LEVEL INPUT AND OUTPUT VARIABLES; USE CODES TO RELATE THESE VARIABLES WITH THE BIT-VECTORS
  4. OBTAIN A HIGH-LEVEL SPECIFICATION OF THE SYSTEM 

- NETWORK CHARACTERISTICS:
  INPUT LOAD FACTORS, FAN-OUT FACTORS, AND DELAYS

OBTAIN SWITCHING EXPRESSIONS

- ASSIGN NAMES TO EACH CONNECTION IN THE NETWORK
- WRITE SWITCHING EXPRESSIONS FOR EACH GATE OUTPUT
- SUBSTITUTE ALL INTERNAL NAMES TO OBTAIN EXTERNAL OUTPUTS IN TERMS OF EXTERNAL INPUTS

Example

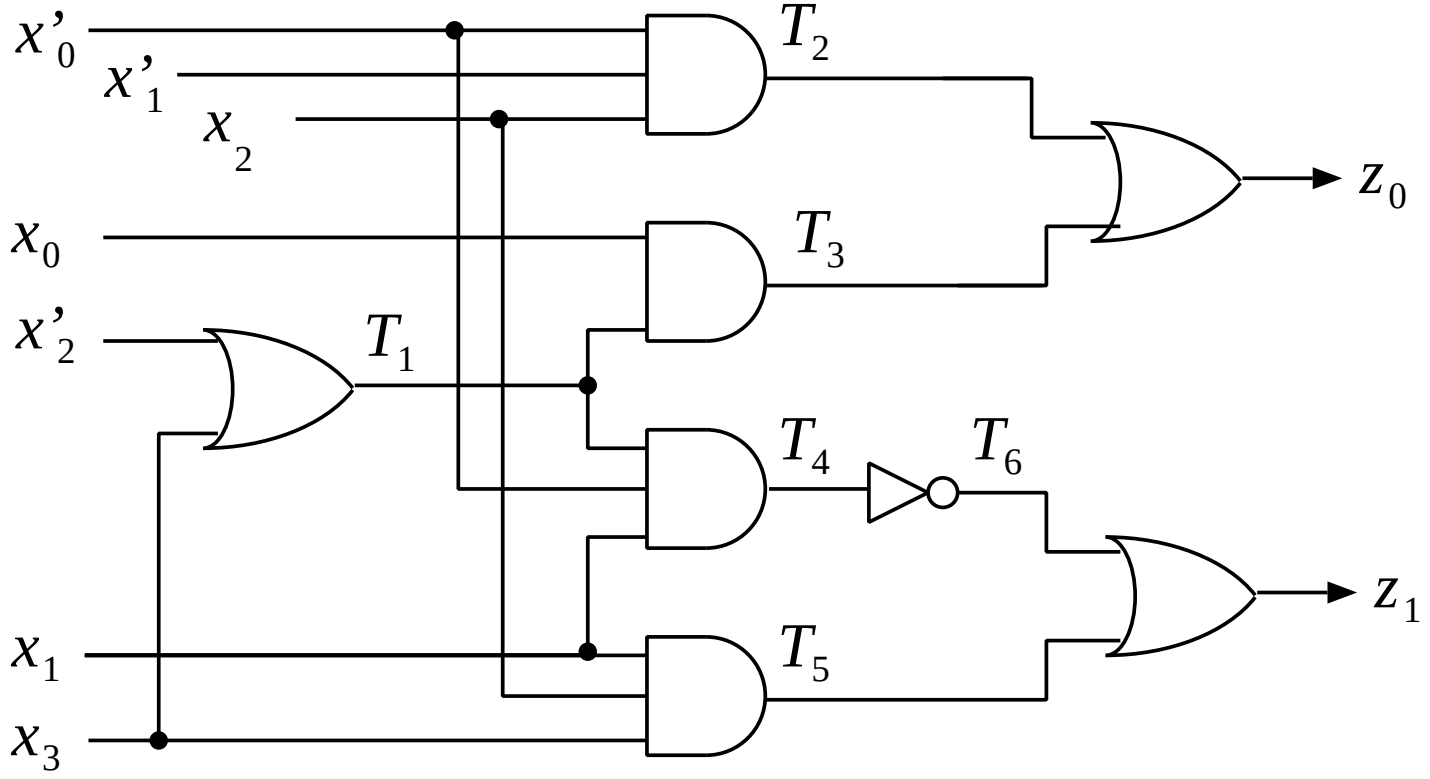


Figure 4.9: GATE NETWORK FOR ANALYSIS

EXAMPLE (cont.)

OUTPUT EXPRESSIONS:

$$
\begin{aligned}
z_0 &= T_2 + T_3 \\
&= x_0' x_1' x_2 + x_0 T_1 \\
&= x_0' x_1' x_2 + x_0 (x_2' + x_3) \\
&= x_0' x_1' x_2 + x_0 x_2' + x_0 x_3
\end{aligned}
$$

$$
\begin{aligned}
z_1 &= T_5 + T_6 \\
&= x_1 x_2 x_3 + T_4' \\
&= x_1 x_2 x_3 + (T_1 x_0' x_1)' \\
&= x_1 x_2 x_3 + T_1' + x_0 + x_1' \\
&= x_1 x_2 x_3 + x_2 x_3' + x_0 + x_1'
\end{aligned}
$$

REDUCED EXPRESSIONS:

$$z_0 = x_0' x_1' x_2 + x_0 x_2' + x_0 x_3 \qquad \text{(no reduction possible)}$$
$$z_1 = x_0 + x_1' + x_2$$

HIERARCHICAL APPROACH

- DECOMPOSE THE NETWORK INTO SUBNETWORKS (MODULES)
- ANALYZE EACH SUBNETWORK SEPARATELY
- USE SUBSTITUTION TO OBTAIN THE NETWORK FUNCTION

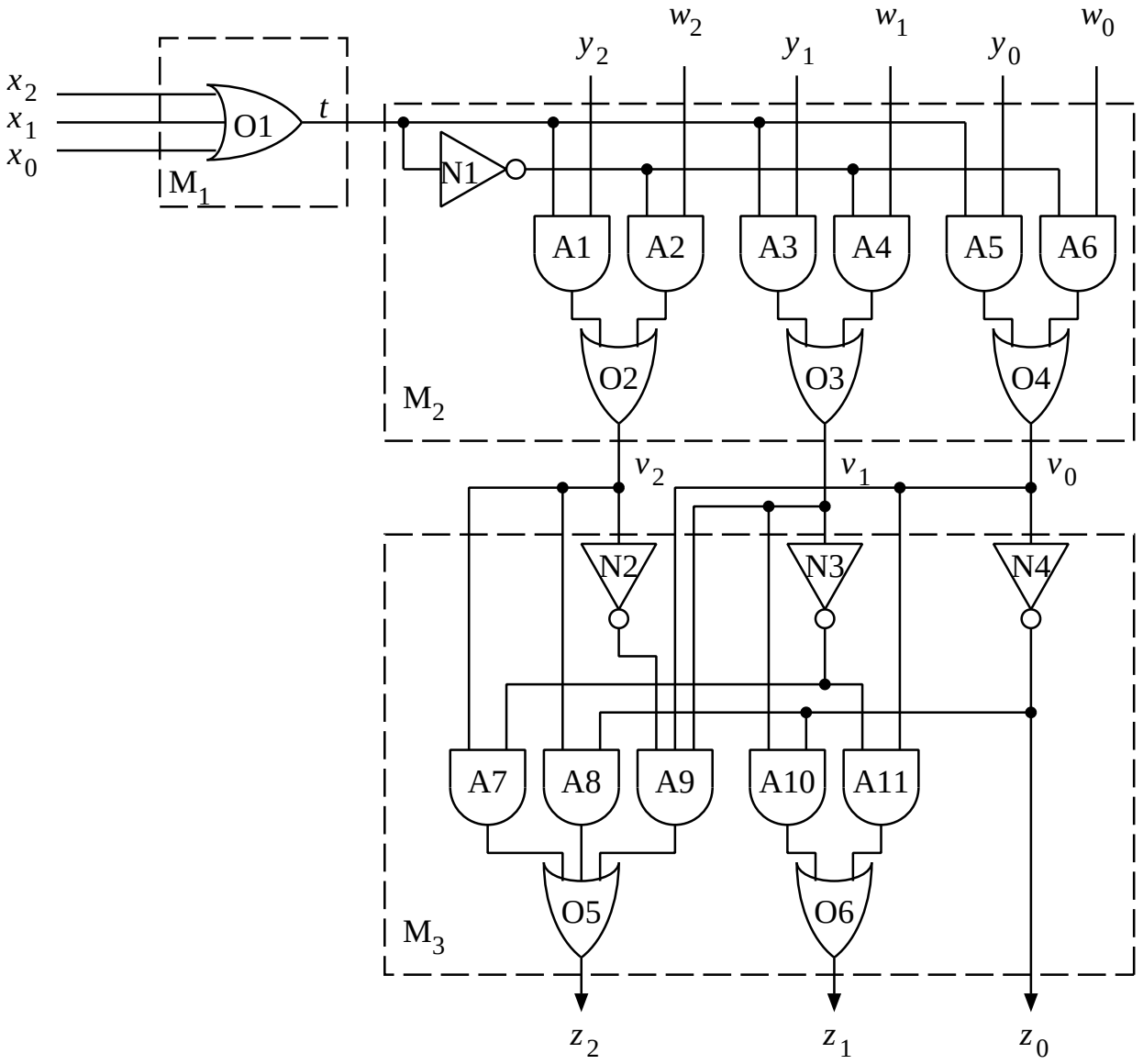


Figure 4.10: NETWORK FOR HIERARCHICAL ANALYSIS

EXAMPLE cont.

VERIFY THAT THE NETWORK SATISFIES THE SPECIFICATION:

Inputs: $x, y, w \in \{0, 1, \ldots, 7\}$

Output: $z \in \{0, 1, \ldots, 7\}$

Function: $$z = \begin{cases} (y+1) \bmod 8 & \textbf{if} \quad x \neq 0 \\ (w+1) \bmod 8 & \textbf{if} \quad x = 0 \end{cases}$$

- SUBNETWORKS

M_1:

$$t = x_2 + x_1 + x_0$$

$$t = \begin{cases} 1 \ \textbf{if} \ \ x \neq 0 \\ 0 \ \textbf{otherwise} \end{cases}$$

M_2:

$$v_i = y_i t + w_i t' \quad (i = 0, 1, 2)$$

$$\underline{v} = \begin{cases} \underline{y} \ \ \textbf{if} \quad t = 1 \\ \underline{w} \ \ \textbf{if} \quad t = 0 \end{cases}$$

$$v = \begin{cases} y \ \ \textbf{if} \ \ t = 1 \\ w \ \ \textbf{if} \ \ t = 0 \end{cases}$$

M_3:

$$\begin{aligned} z_2 &= v_2' v_1 v_0 + v_2 v_1' + v_2 v_0' \\ z_1 &= v_1 v_0' + v_1' v_0 \\ z_0 &= v_0' \end{aligned}$$

EXAMPLE (cont.)

- HIGH-LEVEL SPECIFICATION:

v_2	v_1	v_0	z_2	z_1	z_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0→
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

v	z
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	0

FROM TABLE, WE GET

$$z = (v + 1) \bmod 8$$

SECOND LEVEL OF ANALYSIS:

$$z = \begin{cases} (y+1) \bmod 8 & \textbf{if} \quad x \neq 0 \\ (w+1) \bmod 8 & \textbf{if} \quad x = 0 \end{cases}$$

THIS CORRESPONDS TO THE ORIGINAL SPECIFICATION OF THE FUNCTION

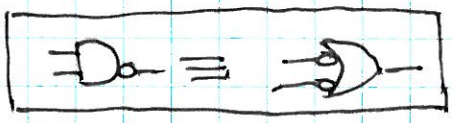
ANALYSIS OF NETWORKS WITH NAND & NOR

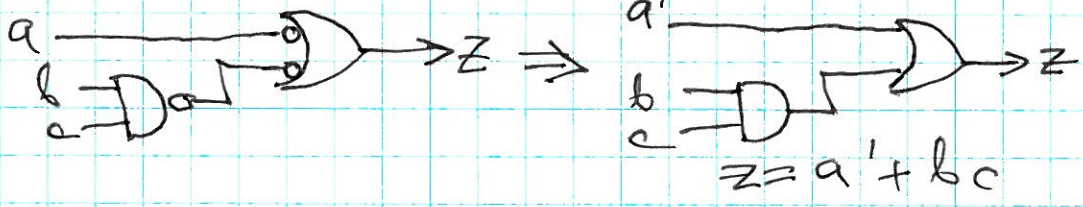
STANDARD WAY:

NAND NETWORK

$Z = (a.(bc)')' = a' + bc$

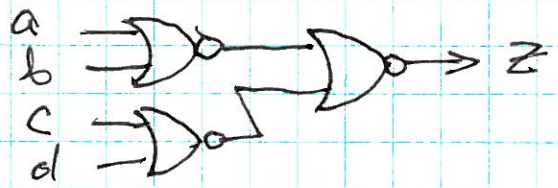
WITH MIXED NOTATION:





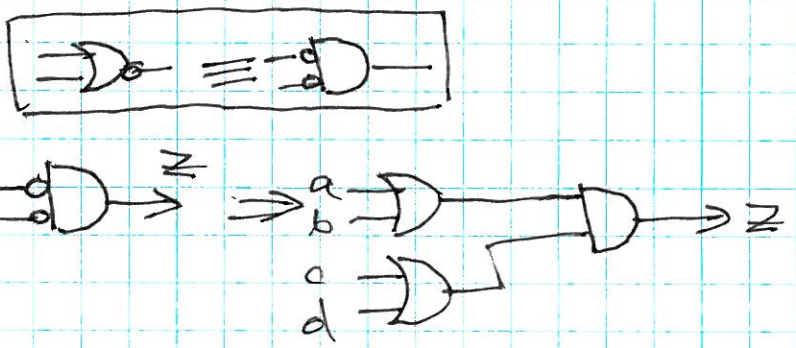
$Z = a' + bc$

NOR NETWORK



$Z = ((a+b)' + (c+d)')'$

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



$Z = (a+b)(c+d)$

EFFICIENT ANALYSIS APPROACH FOR MULTILEVEL NAND (NOR) NETWORKS

ANALYSIS OF NETWORKS WITH NOT, NAND and NOR

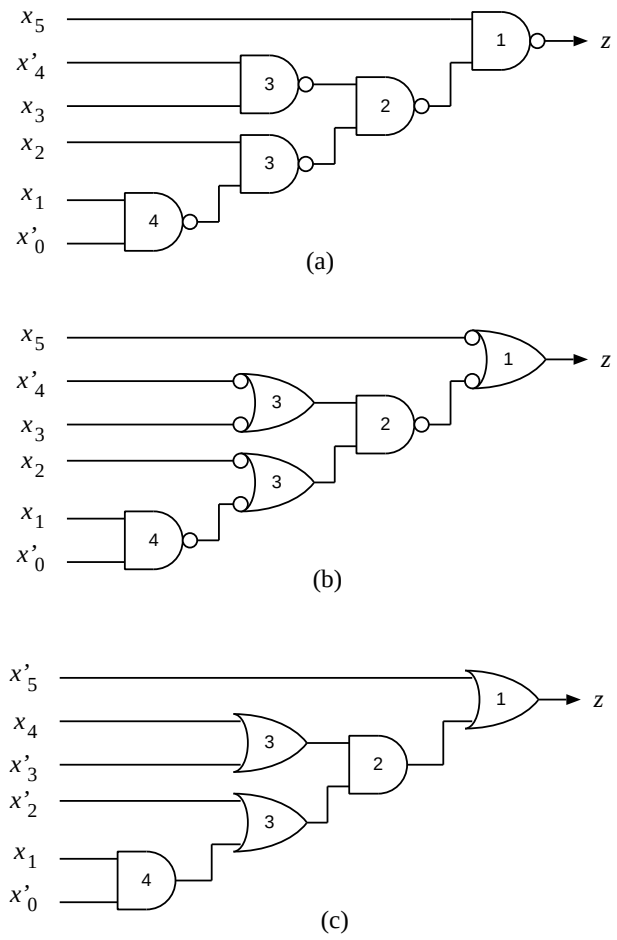
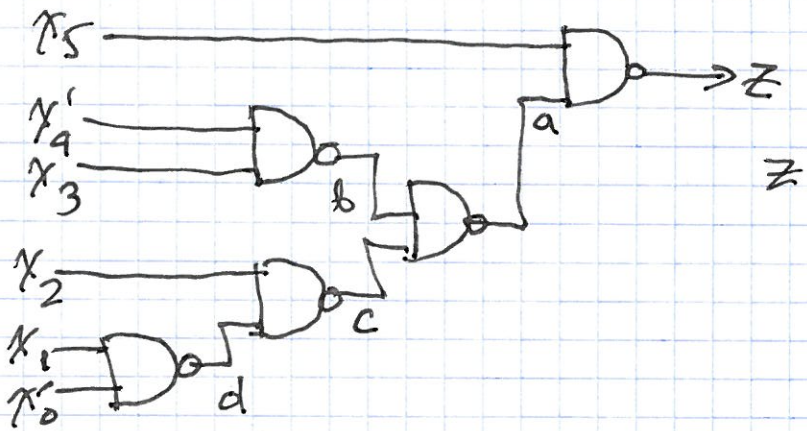


Figure 4.11: a) NAND NETWORK; b) NETWORK REDRAWN IN MIXED-LOGIC NOTATION

CS MSIA ANALYSIS OF NAND NETWORKS



DIRECT APPROACH:

$$
\begin{aligned}
Z &= (x_5 a)' \\
&= (x_5 (b c)')' \\
&= (x_5 ((x_4' x_3)' (x_2 d)')')' \\
&= (x_5 ((x_4' x_3)' (x_2 (x_1 x_0')')')')' \\
&= (x_5 ((x_4' x_3)' (x_2 (x_1' + x_0))')')' \\
&= x_5' + (x_4' x_3)' (x_2 (x_1' + x_0))' \\
&= x_5' + (x_4 + x_3')(x_2' + x_1 x_0') \\
&= x_5' + x_4 x_2' + x_4 x_1 x_0' + x_3' x_2' + x_1 x_0' x_3'
\end{aligned}
$$

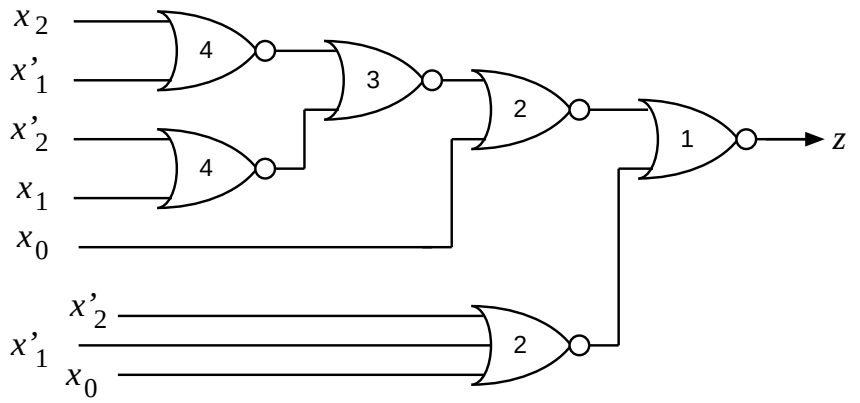
→ INSANITY AND THIS WAS A SMALL NETWORK

→ GO MIXED LOGIC!

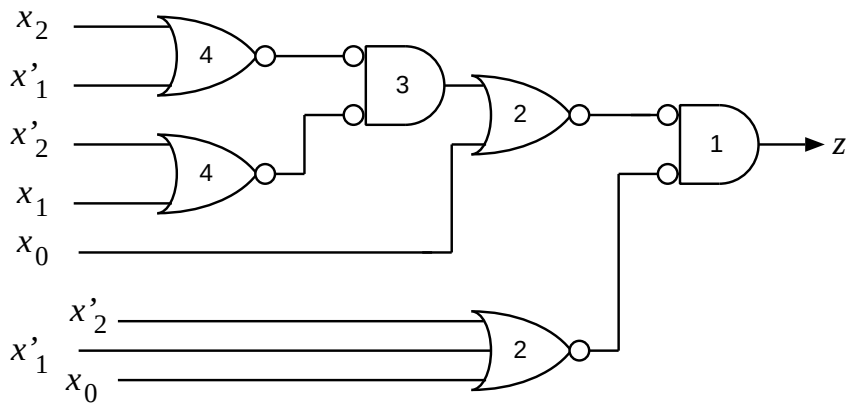
ANALYSIS (cont.)

- USE MIXED-LOGIC TRANSFORMATIONS

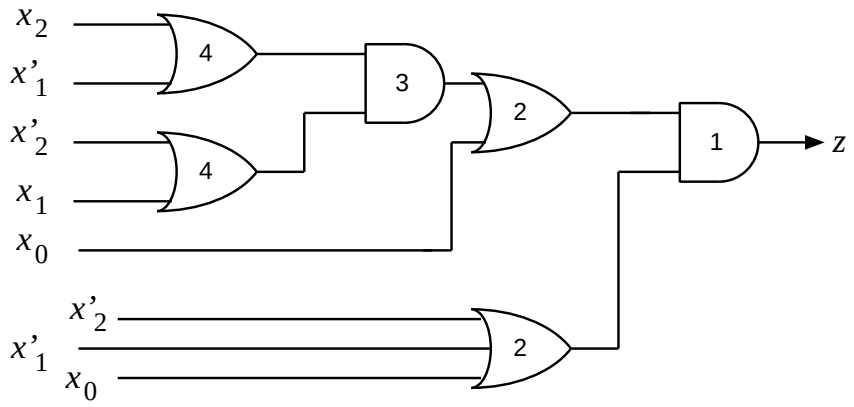
$$
\begin{aligned}
z &= x_5' + (x_4 + x_3')(x_2' + x_1 x_0') \\
&= x_5' + x_4 x_2' + x_3 x_2' + x_4 x_1 x_0' + x_3' x_1 x_0'
\end{aligned}
$$



(a)



(b)



(c)

Figure 4.12: a) NOR NETWORK; b) NETWORK REDRAWN IN MIXED-LOGIC NOTATION

ANALYSIS (cont.)

$$
\begin{aligned}
z &= ((x_2 + x_1')(x_2' + x_1) + x_0)(x_2' + x_1' + x_0') \\
&= (x_2 + x_1' + x_0)(x_2' + x_1 + x_0)(x_2' + x_1' + x_0') \\
&= (x_2 x_1 + x_2' x_1' + x_0)(x_2' + x_1' + x_0) \\
&= x_2' x_1' + x_0
\end{aligned}
$$

ANALYSIS OF CHARACTERISTICS

- LOAD FACTOR OF A NETWORK INPUT
- FAN-OUT FACTOR OF A NETWORK OUTPUT
- SIZE OF THE NETWORK
- NETWORK (PROPAGATION) DELAY
- NUMBER OF LEVELS OF A NETWORK
- DYNAMIC CHARACTERISTICS

Table 4.3: Characteristics of a family of CMOS gates (partial)

Gate type	Fan-in	Propagation delays		Load factor	Size
		t_{pLH} [ns]	t_{pHL} [ns]	[standard loads]	[equiv. gates]
AND	2	$0.15 + 0.037L$	$0.16 + 0.017L$	1.0	2
AND	3	$0.20 + 0.038L$	$0.18 + 0.018L$	1.0	2
OR	2	$0.12 + 0.037L$	$0.20 + 0.019L$	1.0	2
OR	3	$0.12 + 0.038L$	$0.34 + 0.022L$	1.0	2
NOT	1	$0.02 + 0.038L$	$0.05 + 0.017L$	1.0	1

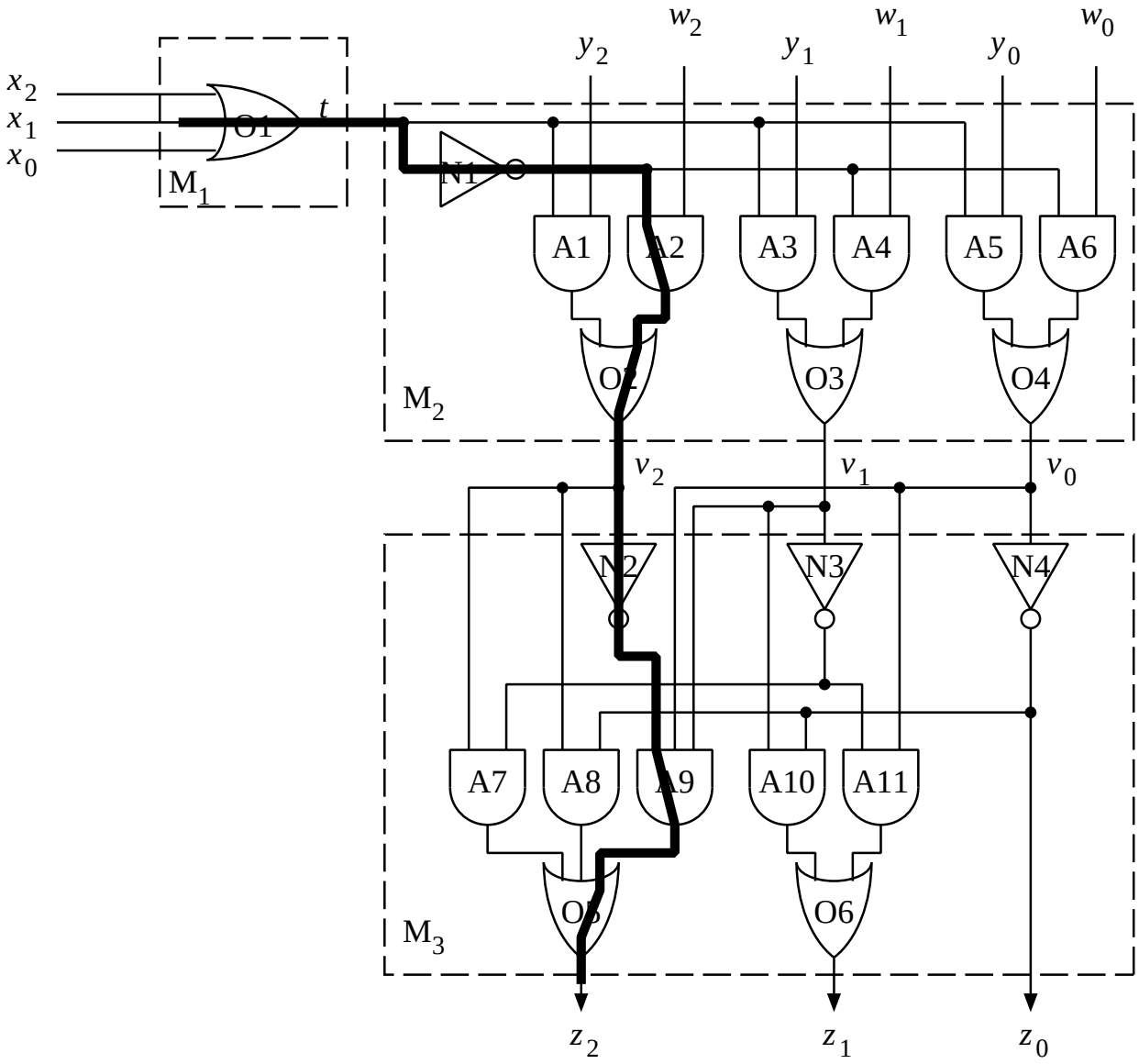


Figure 4.13: NETWORK FOR HIERARCHICAL ANALYSIS

EXAMPLE (cont.)

TYPES OF GATES USED: 2-input AND, 3-input AND, etc.

LOAD FACTORS: NETWORK INPUTS: 1; GATE INPUTS: 1

FANOUT FACTORS: $F = 12$ (assumed)

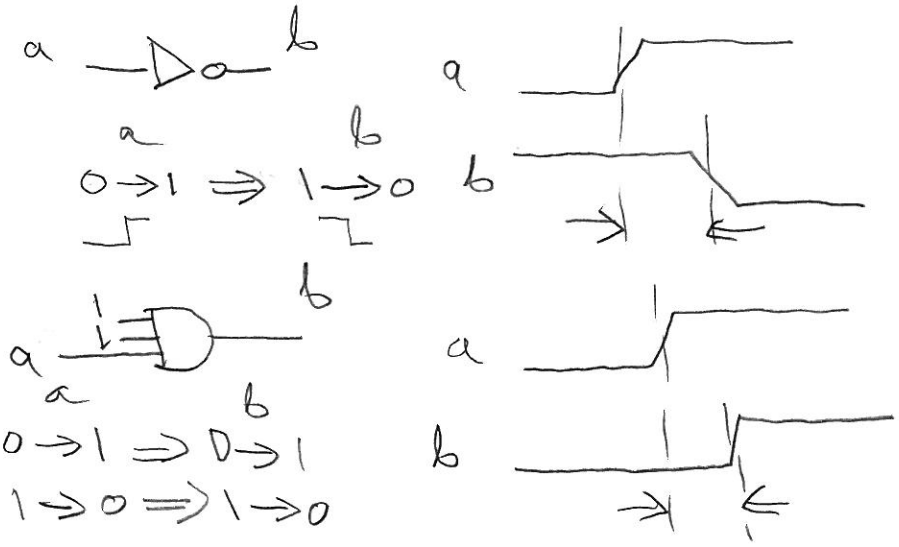
$$F(z_2) = F(z_1) = 12, \quad F(z_0) = 12 - 2 = 10$$

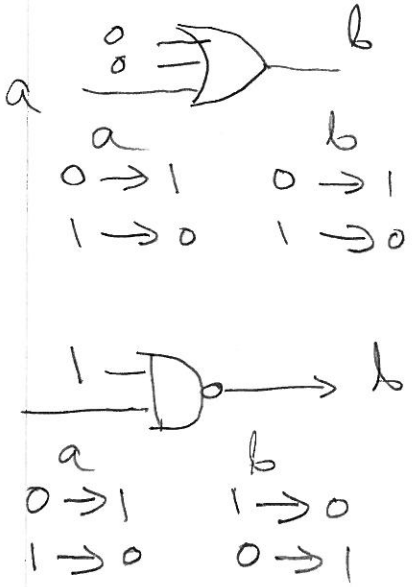
NETWORK SIZE : 38 [equiv. gates] 21 [actual]

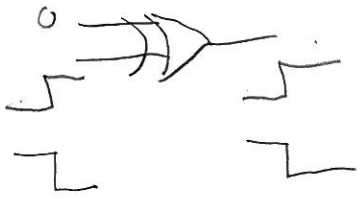
NUMBER OF LEVELS: 7

DELAY ANALYSIS

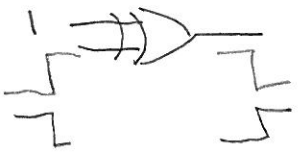
REQUIRED CHANGES IN INPUTS TO CAUSE (0→1) (1→0) IN OUTPUT
L→H H→L







$X \oplus 0 = X$

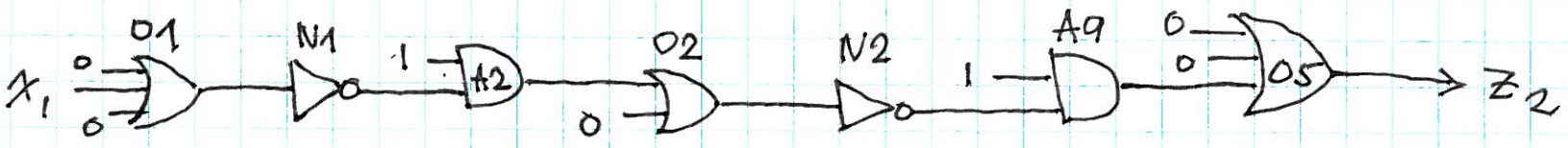


$X \oplus 1 = X'$

* PROCEDURE TO DETERMINE DELAY OF A PATH

E.G. $T_{pLH}(X_1, Z_2)$ (FIG. 4.13)

1. IDENTIFY THE PATH TO ANALYZE



2. SET OTHER GATE INPUTS NOT TO BLOCK TRANSITIONS

3. STARTING FROM THE DESIRED CHANGE OF THE PATH OUTPUT DETERMINE PRECEEDING CHANGES

O1 N1 A2 O2 N2 A9 O5

4. DETERMINE INDIVIDUAL PROPAGATION DELAYS

GATE	ID	L	t_{pLH}	t_{pHL}
OR3	O1	4	$0.12+0.038\times4=0.27_{ns}$	$0.34+0.022\times4=0.43_{us}$

5. ADD THEM UP

$$t_{pLH}(X_1,Z_2)=\overset{0.27}{t_{pLH}}(O1)+\overset{0.1}{t_{pHL}}(N1)+\overset{0.18}{t_{pHL}}(A2)+\overset{0.26}{t_{pHL}}(O2)+\overset{0.06}{t_{pLH}}(N2)$$
$$+\overset{0.24}{t_{pLH}}(A9)+\overset{0.12+0.038L}{t_{pLH}}(O5)=1.23+0.038L\ [ns]$$

NETWORK DELAY Example of path delay calculation:

$$O_1 \rightarrow N_1 \rightarrow A_2 \rightarrow O_2 \rightarrow N_2 \rightarrow A_9 \rightarrow O_5$$

$$\begin{aligned} T_{pLH}(x_1, z_2) &= t_{pLH}(O_1) + t_{pHL}(N_1) + t_{pHL}(A_2) + t_{pHL}(O_2) \\ &\quad + t_{pLH}(N_2) + t_{pLH}(A_9) + t_{pLH}(O_5) \\ T_{pHL}(x_1, z_2) &= t_{pHL}(O_1) + t_{pLH}(N_1) + t_{pLH}(A_2) + t_{pLH}(O_2) \\ &\quad + t_{pHL}(N_2) + t_{pHL}(A_9) + t_{pHL}(O_5) \end{aligned}$$

Gate	Identifier	Output load	t_{pLH} [ns]	t_{pHL} [ns]
OR3	O_1	4	0.27	0.43
NOT	N_1	3	0.13	0.10
AND2	A_2	1	0.19	0.18
OR2	O_2	3	0.23	0.26
NOT	N_2	1	0.06	0.07
AND3	A_9	1	0.24	0.20
OR3	O_5	L	$0.12 + 0.038L$	$0.34 + 0.022L$

$$\begin{aligned} T_{pLH}(x_1, z_2) &= 0.27 + 0.10 + 0.18 + 0.26 + 0.06 \\ &\quad + 0.24 + 0.12 + 0.038L = 1.23 + 0.038L \text{ [ns]} \\ T_{pHL}(x_1, z_2) &= 0.43 + 0.13 + 0.19 + 0.23 + 0.07 \\ &\quad + 0.20 + 0.34 + 0.022L = 1.59 + 0.022L \text{ [ns]} \end{aligned}$$

TIMING DIAGRAM

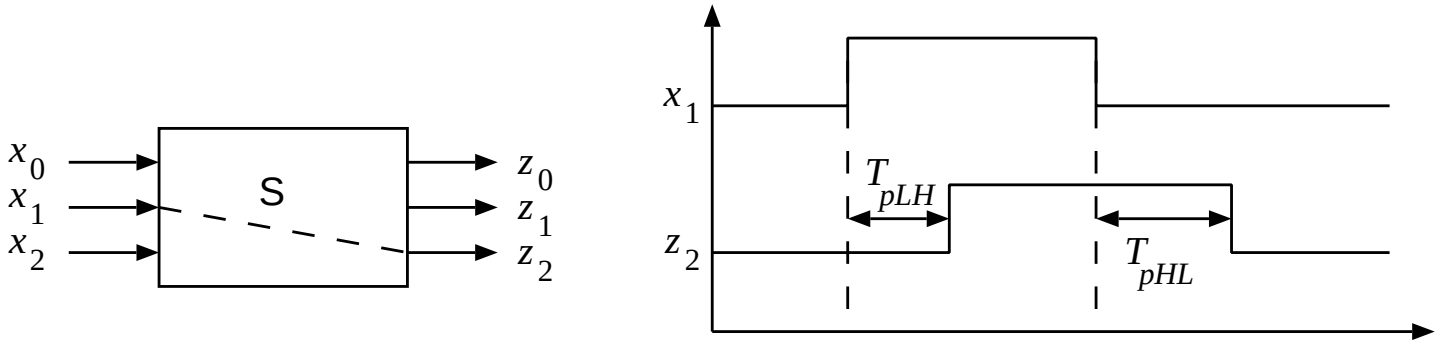
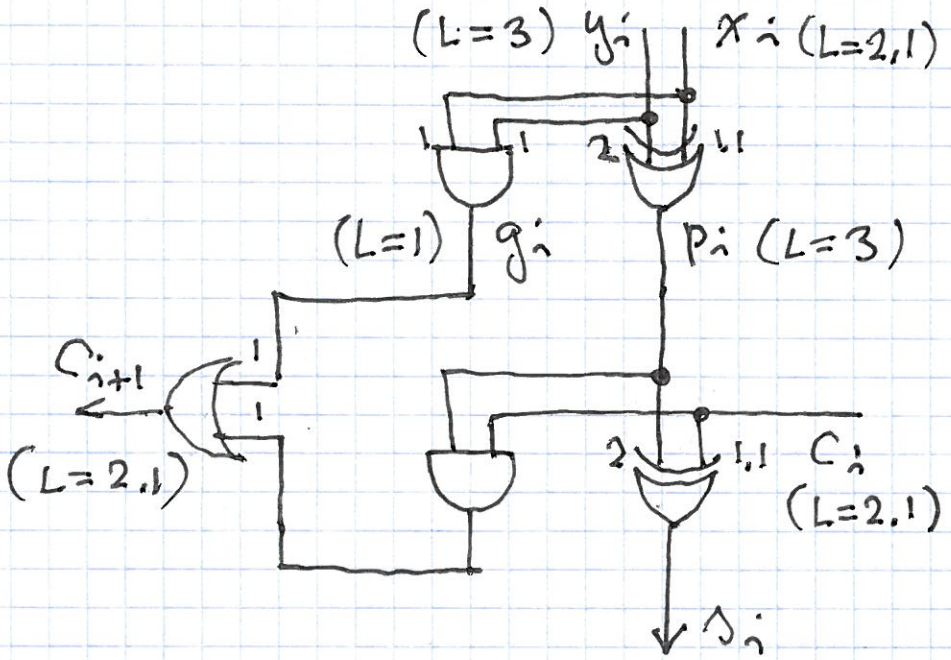


Figure 4.14: TIMING DIAGRAM FROM NETWORK ANALYSIS

CS MSIA EXAMPLE: ANALYSIS

a) VERIFY THAT THE FOLLOWING CIRCUIT IMPLEMENTS A FULL ADDER



x_i	y_i	c_i	g_i	p_i	c_{i+1}	s_i
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	1	0	1
0	1	1	0	1	1	0
1	0	0	0	1	0	1
1	0	1	0	1	1	0
1	1	0	1	0	1	0
1	1	1	1	0	1	1

b) DETERMINE DELAYS $t(x_i, s_i), t(c_i, s_i), t(c_i, c_{i+1})$ (YOU)

$p_i = x_i \oplus y_i$ PROPAGATE CARRY

$g_i = x_i y_i$ GENERATE CARRY

$$c_{i+1} = g_i + p_i c_i$$
$$= x_i y_i + x_i' y_i c_i + x_i y_i' c_i$$
$$= x_i y_i + x_i c_i + y_i c_i \text{ (PROVE!)}$$

(MAJORITY FUNCTION)

$$s_i = p_i \oplus c_i = x_i \oplus y_i \oplus c_i$$
$$\Leftrightarrow (x_i + y_i + c_i) \bmod 2$$

(ARITH. +)

$\Rightarrow x_i + y_i + c_i \in \{0, 1, 2, 3\}$

$= 2c_{i+1} + s_i \in \{0, 1, 2, 3\}$

$\Rightarrow$ FULL ADDER

b) DETERMINE DELAYS

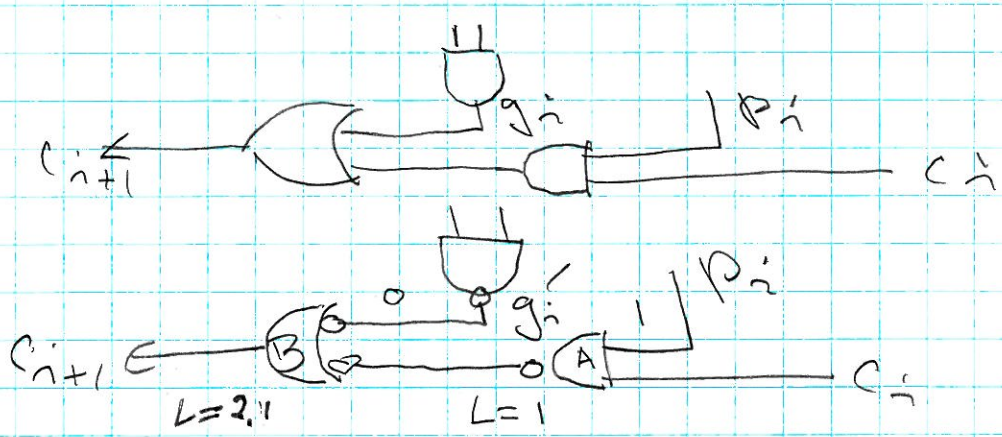
$t_{pLH}(XOR) = 0,30 + 0,036 \cdot L$
$= 0,16 + 0,036 \cdot L$
$t_{pHL}(XOR) = 0,30 + 0,021 \cdot L$
$= 0,15 + 0,0201 L$ etc

FIND:

$t(x_i, s_i)$ | $t_{pLH}(NAND) = 0,05 + 0,038 \cdot L$
$t(c_i, s_i)$ | $t_{pHL}(NAND) = 0,08 + 0,027 \cdot L$
→ $t(c_i, c_{i+1})$ CRITICAL PATH

A FASTER & SMALLER DESIGN

→ REPLACE AND-OR WITH NAND-NAND



IN CRITICAL PATH

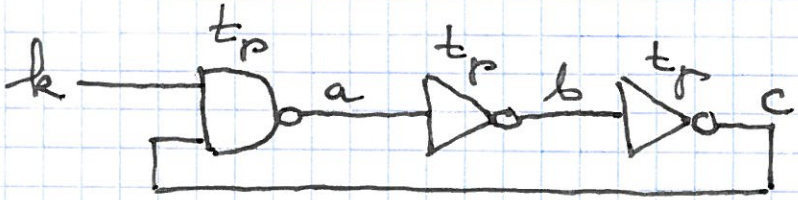
$t_{pLH}(c_i, c_{i+1}) = t_{LH}(A) + t_{LH}(B) = 0,05 + 0,038 \times 1$
$+ 0,05 + 0,038 \times 2,1$
$= 0,2187$ ns

$t_{pHL}(c_i, c_{i+1}) = t_{pHL}(A) + t_{pHL}(B) = 0,08 + 0,027 \times 1$
$+ 0,08 + 0,027 \times 2,1$
$= 0,2437$ ns

CS M51A

DIGITAL OSCILLATOR

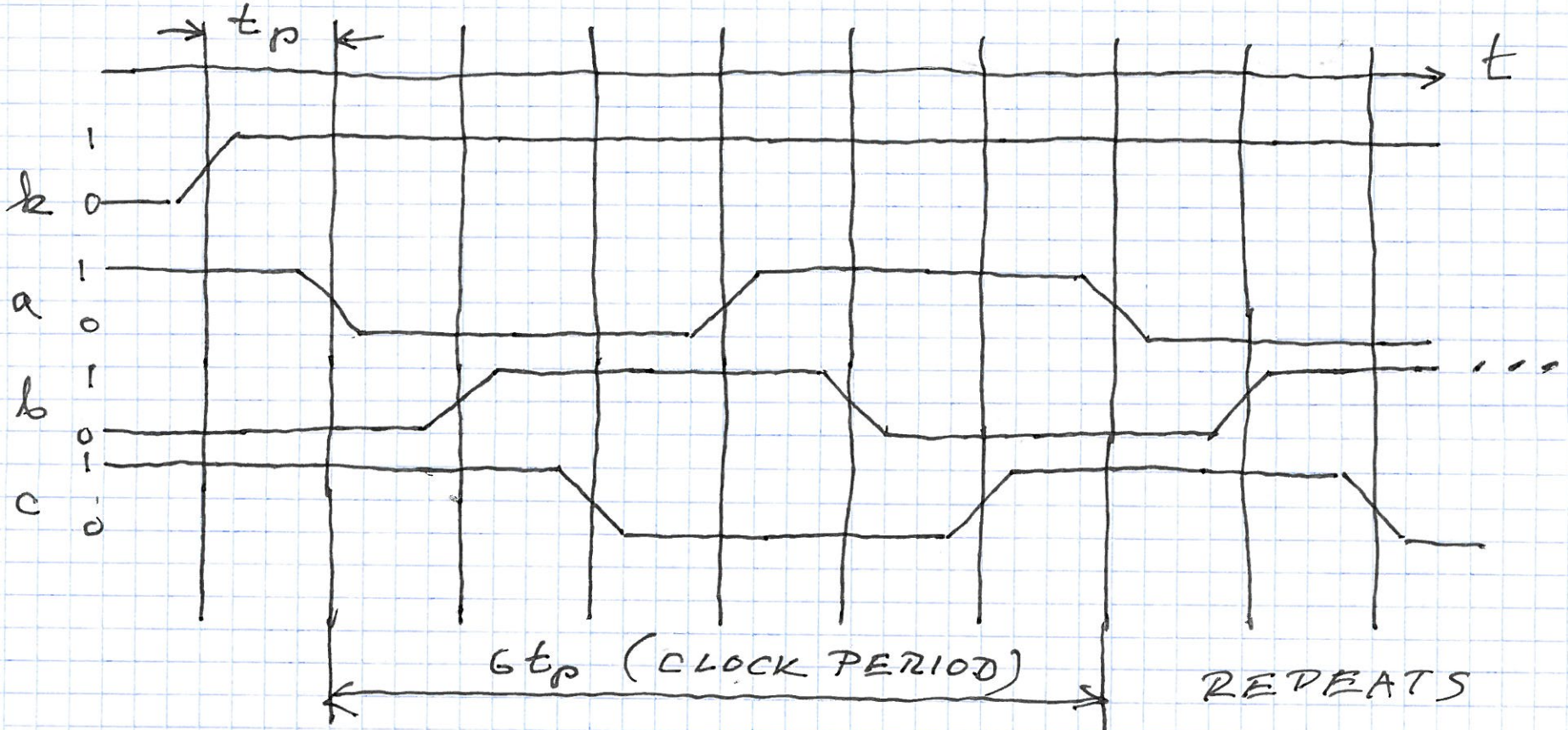
(CLOCK GENERATOR)



- ASSUME ALL PROP. DELAYS ARE t_p

$$\left(t_p = \frac{t_{LH} + t_{HL}}{2} \right)$$

TIMING DIAGRAM:



CLOCK FREQUENCY: $f = \frac{1}{6t_p}$

$t_p = 1\,ns\ (10^{-9}\,sec)$ $\quad f = \frac{10^9}{6} \sim 167\,MHz$

$t_p = 0.1\,ns$ $\quad f = 1.67\,GHz$