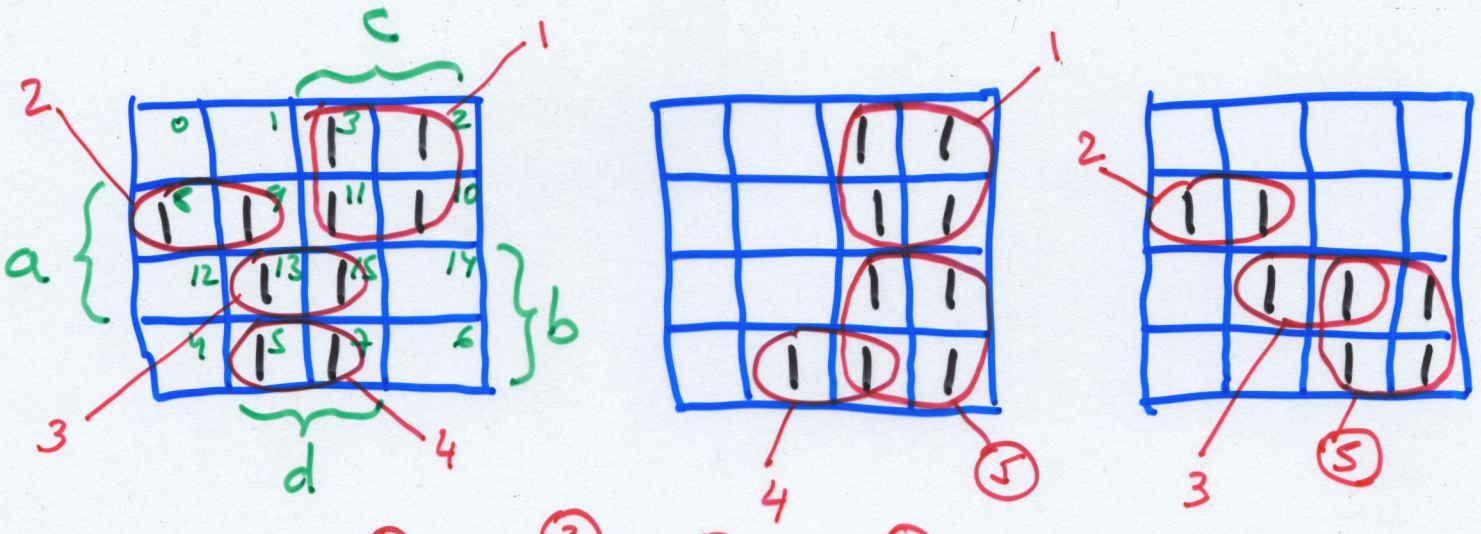
Multiple Output Circuits : Example

$$f_1 = \sum m(2,3,5,7,8,9,10,11,13,15)$$

$$f_2 = \sum m(2,3,5,6,7,10,11,14,15)$$

$$f_3 = \sum m(6,7,8,9,13,14,15)$$

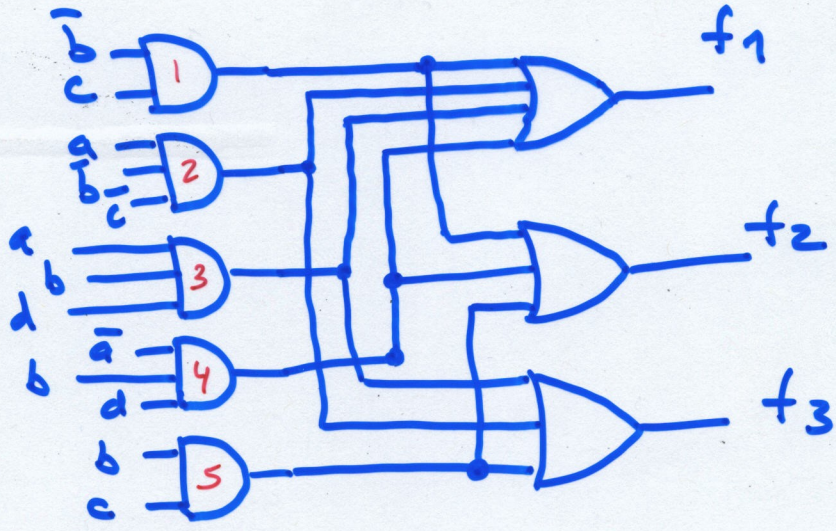


$$f_1 = \overset{①}{\bar{b}c} + \overset{②}{a\bar{b}\bar{c}} + \overset{③}{abd} + \overset{④}{\bar{a}bd}$$

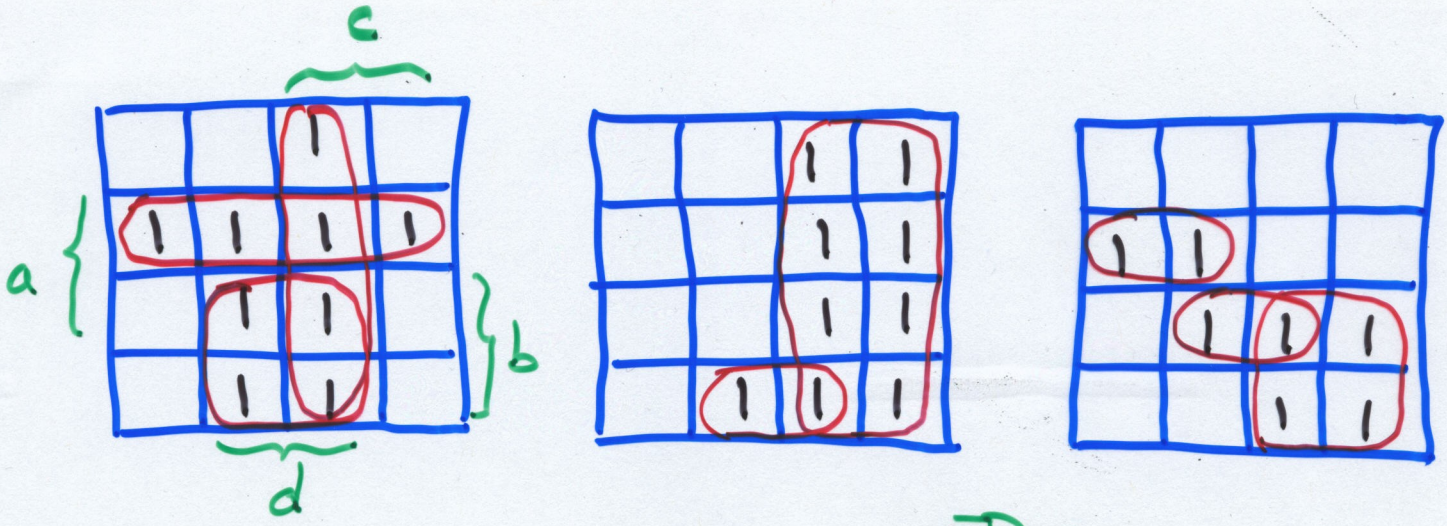
$$f_2 = \overset{④}{\bar{a}bd} + \overset{⑤}{bc} + \overset{①}{\bar{b}c} = \overset{④}{\bar{a}bd} + c$$

$$f_3 = \overset{②}{a\bar{b}\bar{c}} + \overset{③}{abd} + \overset{⑤}{bc}$$

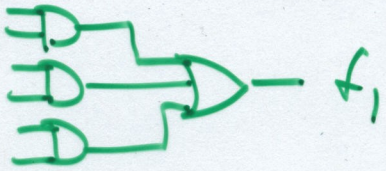
Circuit:



8 GATES
23 INPUTS
(46 TRANSIST)

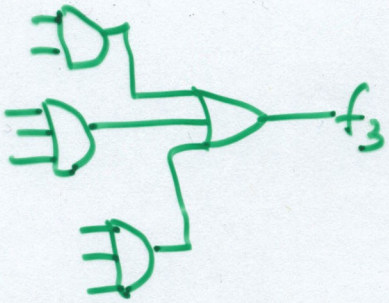


$f_1 = a\bar{b} + bd + cd$



$f_2 = c + \bar{a}bd$

$f_3 = bc + abd + a\bar{b}\bar{c}$

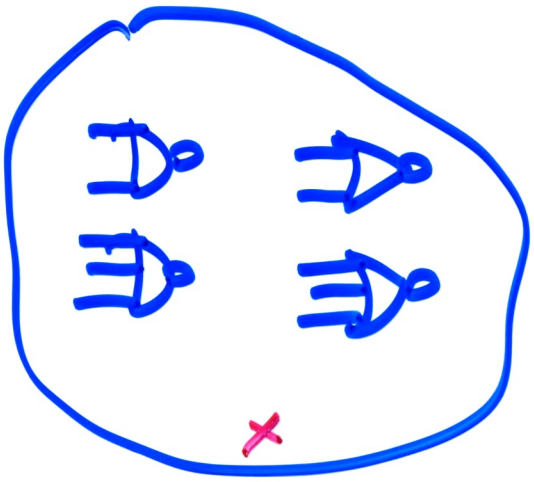


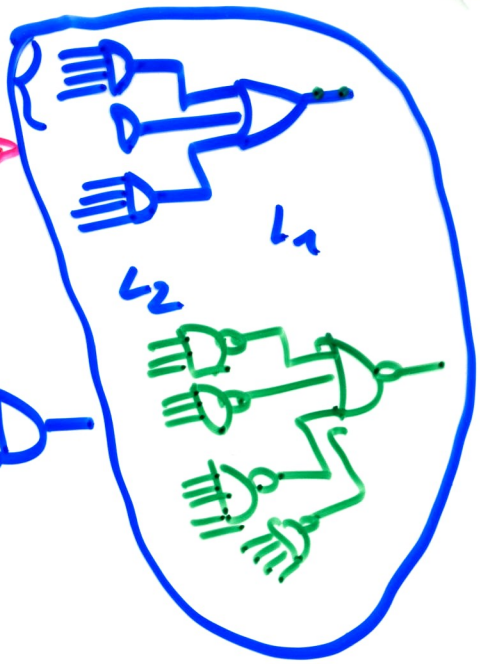
10 : GATES

25 : INPUTS (50 Transistors)

$f_1 = \Sigma(\ ) : \underline{SOP}$

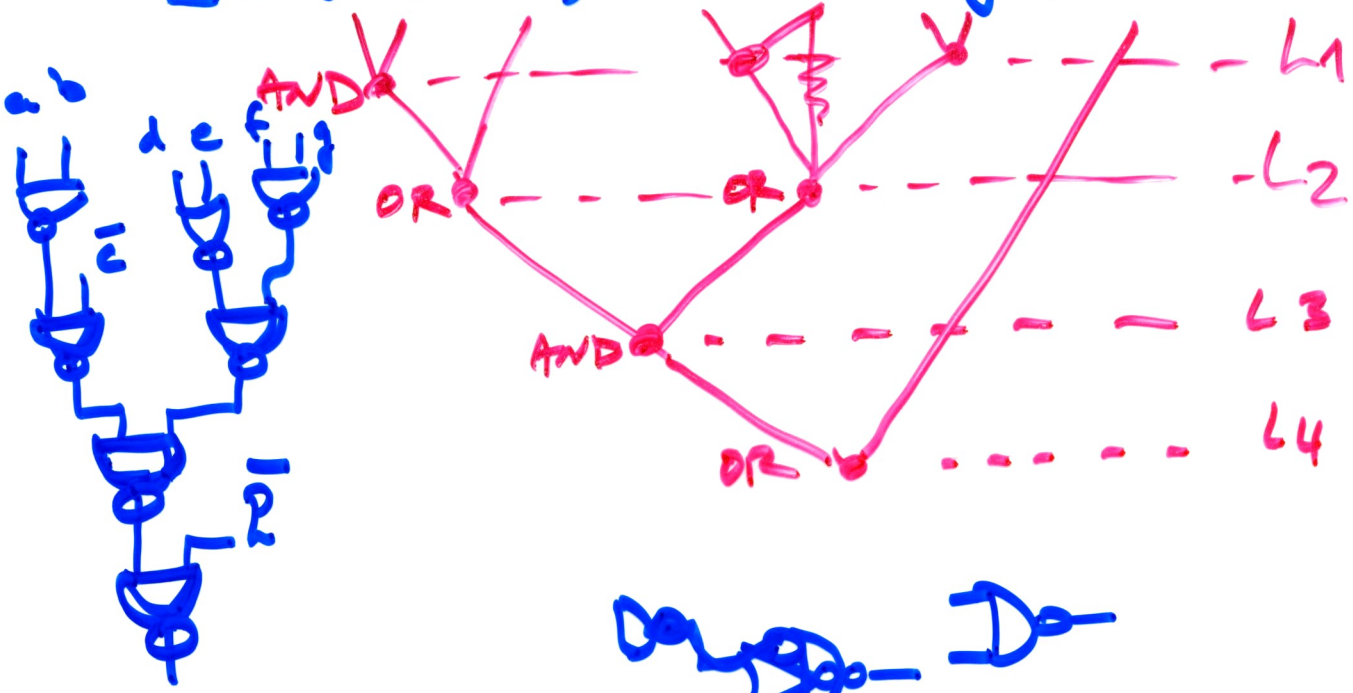
$= \Pi(\ ) : \underline{POS}$

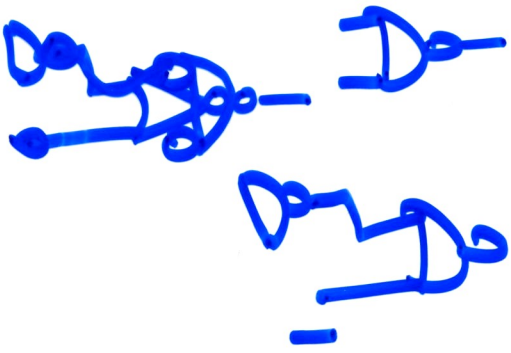




Multi level Gate Network

$Z = (ab + c)(d + e + fg) + h$





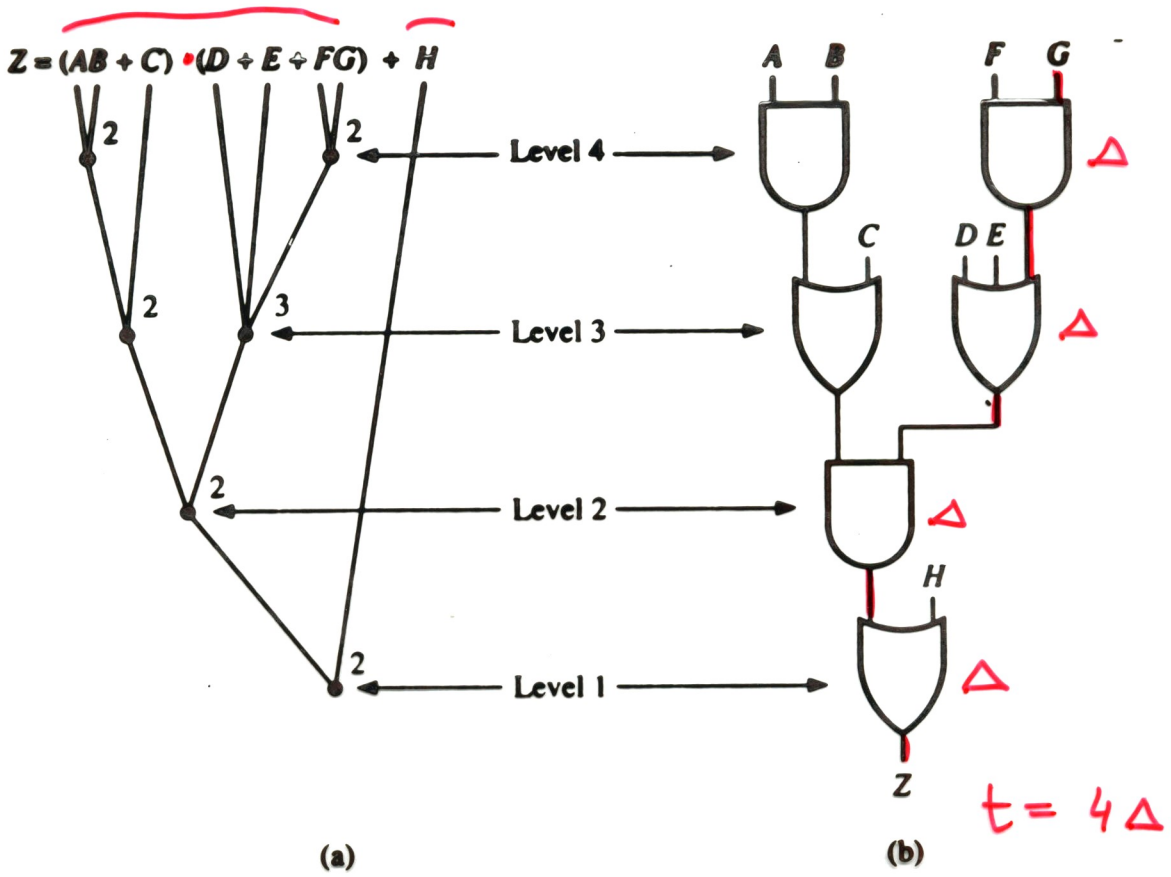


Figure 8–1
Four-Level Realization of Z

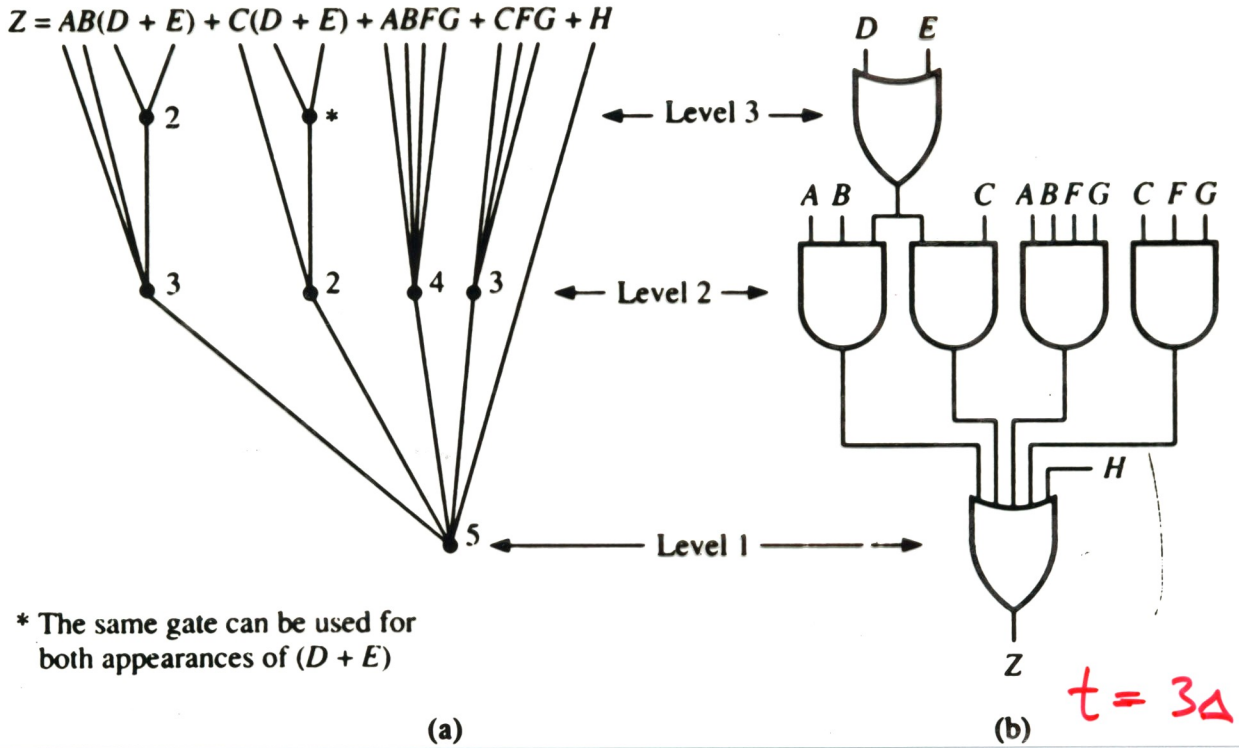
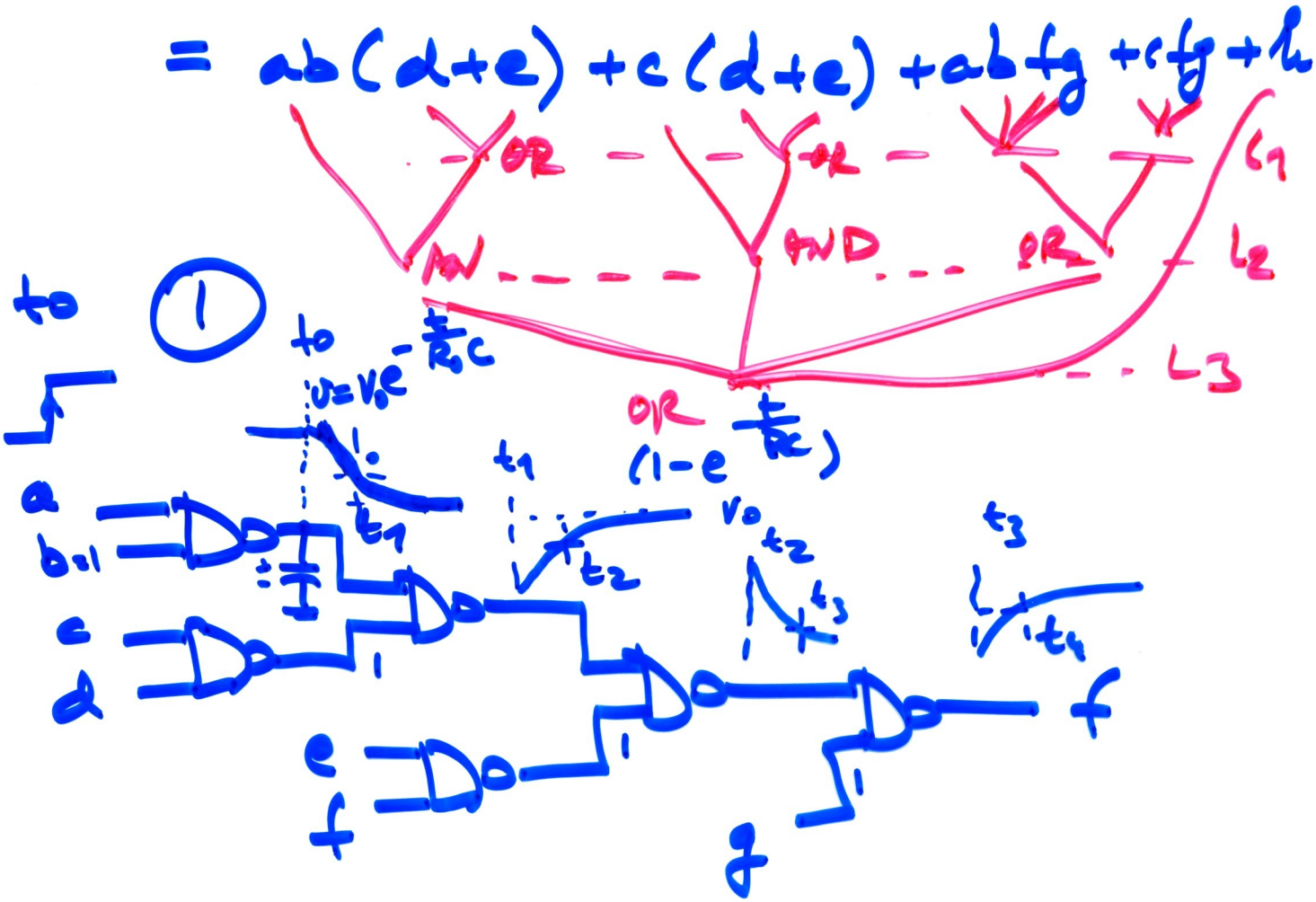


Figure 8–2
Three-Level Realization of Z

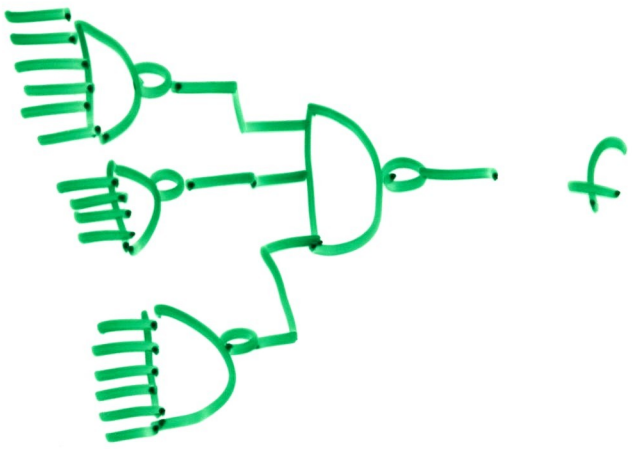
$$Z = (ab+c)(d+e+fg)+h$$

$$= ab(d+e)+c(d+e)+abfg+cfg+h$$

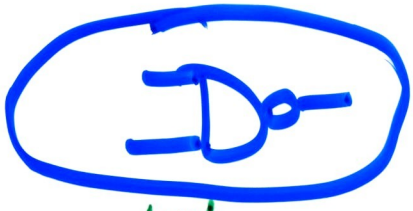


$t_4 - t_0 = \Delta$ delay

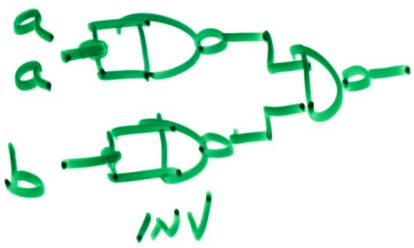
(2)



NOT $\bar{}$; AND ; OR complete set



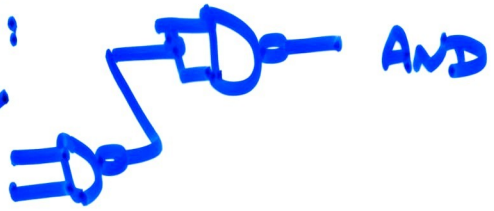
INV



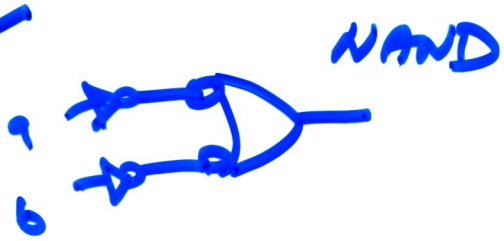
INV

NOT ✓ :

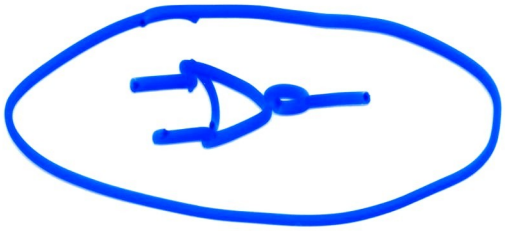
AND ✓

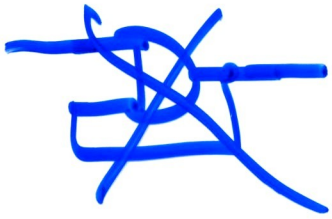


OR: ✓



Simplify





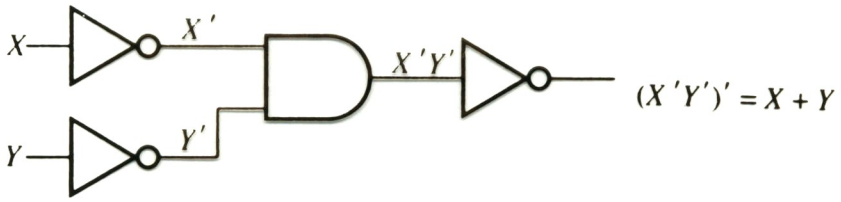
NOT X

INV

contained in function

8.3 Functionally Complete Sets of Logic Gates

A set of logic operations is said to be *functionally complete* if any Boolean function can be expressed in terms of this set of operations. The set AND, OR, and NOT is obviously functionally complete since any function can be expressed in sum-of-products form, and a sum-of-products expression uses only the AND, OR, and NOT operations. Similarly, a set of logic gates is functionally complete if all switching functions can be realized using this set of gates. Since the set of operations AND, OR, and NOT is functionally complete, any set of logic gates which can realize AND, OR, and NOT is also functionally complete. AND and NOT are a functionally complete set of gates since OR can also be realized using AND and NOT:



If a single gate forms a functionally complete set by itself, then any switching function can be realized using only gates of that type. The NAND gate is an example of such a gate. Since the NAND gate performs the AND operation followed by an inversion, NOT, AND, and OR can be realized using only NAND gates as shown in Fig. 8–11.

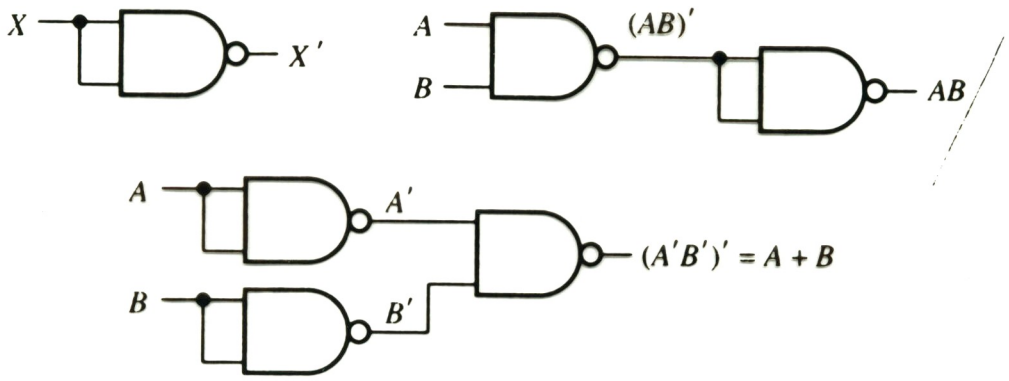
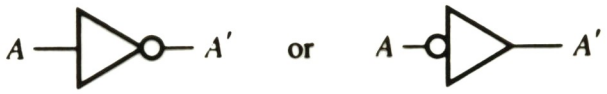


Figure 8–11
NAND Gate Realization of NOT, AND, and OR

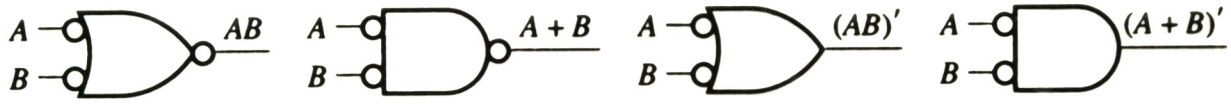
8.6 Network Conversion Using Alternative Gate Symbols

Logic designers who design complex digital systems often find it convenient to use more than one representation for a given type of gate. For example an inverter can be represented by



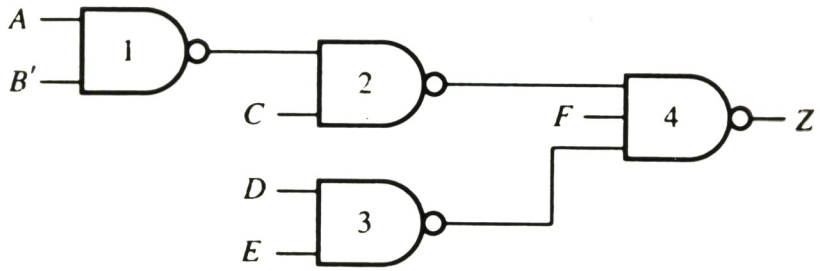
In the second case, the inversion "bubble" is at the input instead of the output. Figure 8–17 shows some alternative representations for AND, OR, NAND, and NOR gates. These equivalent gate symbols are based on the identities

$$AB = (A' + B')', \quad A + B = (A'B')', \quad (AB)' = A' + B', \quad (A + B)' = A'B'$$

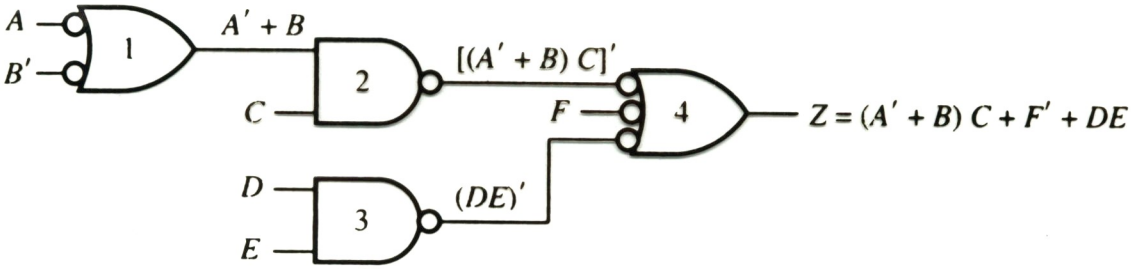


(a) AND (b) OR (c) NAND (d) NOR

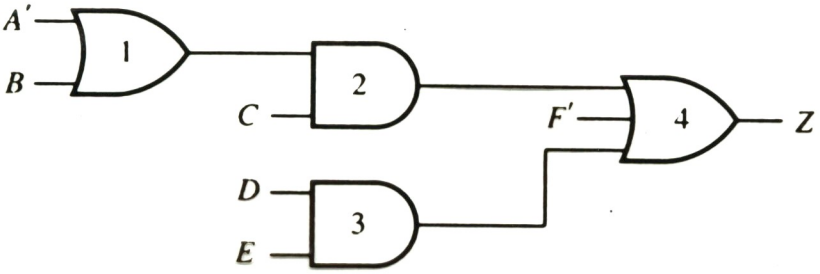
Figure 8–17
Alternative Gate Symbols



(a) NAND-gate network

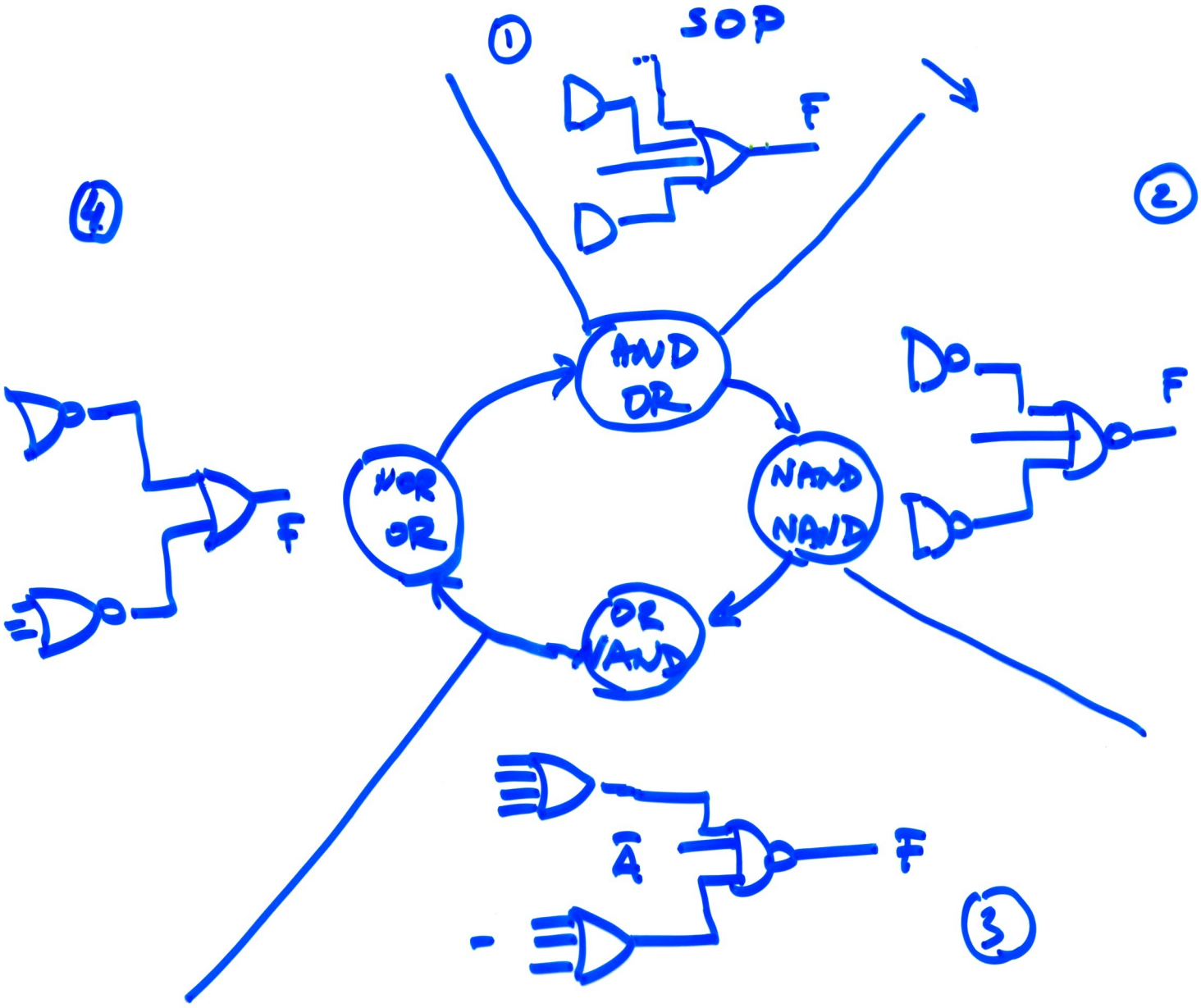


(b) Alternate form for NAND-gate network



(c) Equivalent AND-OR network

Figure 8–18
NAND-Gate Network Conversion



$$F = a + b\bar{c} + \bar{b}cd \quad \cdots ①$$

$$F = \overline{\bar{a} \cdot (\overline{b\bar{c}}) \cdot (\overline{\bar{b}cd})} \quad \cdots ②$$

$$F = \overline{\bar{a} \cdot (\bar{b} + c) \cdot (b + \bar{c} + \bar{d})} \cdots ③$$

$$F = a + \overline{(\bar{b} + c)} + \overline{(b + \bar{c} + \bar{d})} \cdots ④$$