

Use symbol D represent $1/0$ $\leftarrow$ Normal Circuit
 $\bar{D}$ represent $0/1$ $\leftarrow$ Faulty Circuit

Facilitates multiple path sensitization.

Four valued algebra $\{0, 1, D, \bar{D}\}$

behaves in accordance with rules of Boolean Algebra

NOT

0	1
1	0
D	$\bar{D}$
$\bar{D}$	D

AND

	0	1	D	$\bar{D}$
0	0	0	0	0
1	0	1	D	$\bar{D}$
D	0	D	D	0
$\bar{D}$	0	$\bar{D}$	0	$\bar{D}$

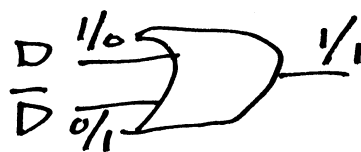
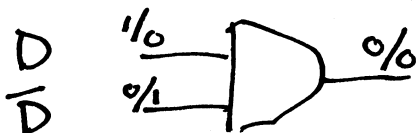
(a)

OR

	0	1	D	$\bar{D}$
0	0	1	D	$\bar{D}$
1	1	1	1	1
D	D	1	D	1
$\bar{D}$	$\bar{D}$	1	1	$\bar{D}$

(b)

Figure 2.15



Path Sensitization procedure ^{DA2} ~~has~~ can be formalized into D-algorithm

- Can be programmed
- Enables test generation for more complex Fault models and propagation through more complex elements

(For s-a faults and gate elements not really necessary)

- Is an algorithm
(Can prove that it will always find a test if one exists).

Cubical Algebra

Three types of cubes used.

- (1) For a circuit element E which realizes the combinational function f , the *primitive* cubes* (pc) of f are a typical representation of the prime implicants of f and $\bar{f}$. These cubes concisely represent the logical behavior of E .

Consider a logic element E , shown in Figure 2.16(a), which realizes the combinational function f shown in Figure 2.16(b). The prime implicants of f are $\bar{x}_1\bar{x}_3$ and x_2 , and the prime implicants of $\bar{f}$ are $\bar{x}_2x_3$ and $x_1\bar{x}_2$.

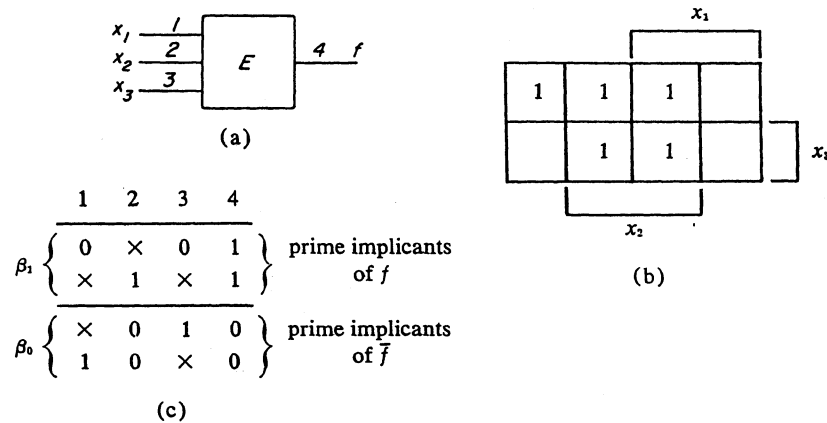


Figure 2.16

*The term *primitive* is used to denote minimal input conditions required to produce a desired output.

X indicates unspecified value
associated with line

Intersection of 2 cubes α_1, β_j

$$\alpha_1 \cap \beta_j$$

= ~~the~~ the value of the two cubes
in each position in which
they have the same value

= the value of the specified cube
in each position where one
cube has a specified value
and the other is unspecified (x)

= x in any position in which both are
unspecified

Example

$$\begin{array}{l} 0x10 \cap x110 = 0110 \\ 0x10 \cap 0xxx0 = 0x10 \end{array}$$

= does not exist if 2 cubes
have opposite values in any
position

$$0x10 \cap 1xxx$$

does not exist

Example

- (2) A *primitive D-cube of a logic fault (pdcf)* α in a logic element E specifies the minimal input conditions which must be applied to E in order to produce an error signal (D or $\bar{D}$) at the output of E . These input conditions can be determined from the primitive cubes of f , the normal function realized by E , and f_α , the function realized by the faulty circuit as follows: An input condition produces a faulty output $D(\bar{D})$ if it is contained in a prime implicant of $f(\bar{f})$ and also contained in a prime implicant of $\bar{f}_\alpha(f_\alpha)$. We denote the set of prime implicants of $f(\bar{f})$ as $\beta_1(\beta_0)$ and the prime implicants of $f_\alpha(\bar{f}_\alpha)$ as $\alpha_1(\alpha_0)$. The primitive D -cubes of the fault which result in output $\bar{D}$ can be obtained by intersecting the inputs of each cube in α_1 with those for each cube in β_0 . The primitive D -cubes of a fault α resulting in output D can be obtained by intersecting cubes in α_0 and β_1 .

For the logic element defined in Figure 2.16 and the faulty function f_α defined in Figure 2.17(a),(b), the primitive D cubes of the fault are shown in Figure 2.17(c).

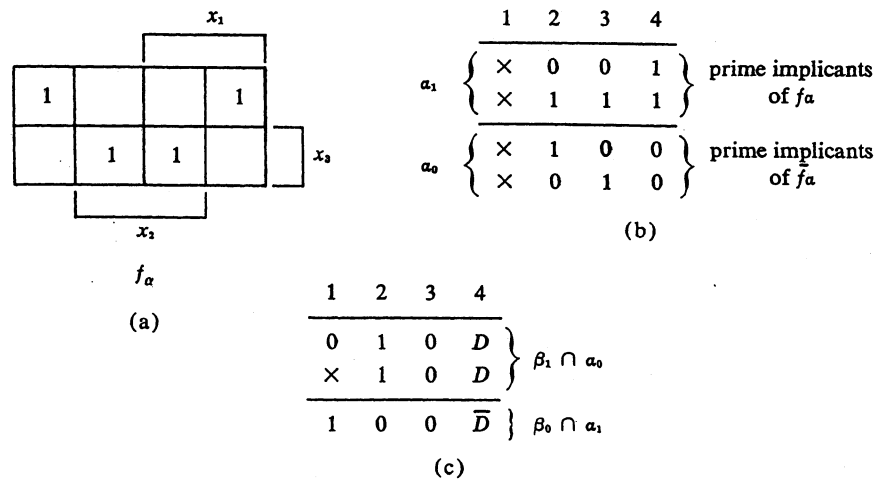


Figure 2.17

Primitive D -cubes of a fault may also be produced for logic blocks with several outputs as illustrated in the following example.

Example 2.6: Consider the circuit of Figure 2.18(a) and the input bridge fault which creates the AND of the shorted signals. Part (b) of the figure specifies the primitive cubes of the normal and faulty circuit. The primitive D cubes of the fault result from the intersection of an α cube and β cube which differ in at least one output. In this case the two D -cubes result from

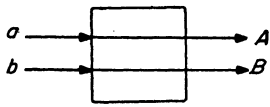
		a	b	A	B	
(a)		0	0	0	0	} Primitive cubes of circuit (β)
		0	1	0	1	
		1	0	1	0	
		1	1	1	1	
(b)		0	$\times$	0	0	} Primitive cubes of faulty circuit (α)
		$\times$	0	0	0	
		1	1	1	1	
		0	1	0	D	} Primitive D cubes of the fault
		1	0	D	0	

Figure 2.18

the intersection of the normal cube 0101 with the faulty cube $\times 000$ and from the intersection of 1010 with 0×00 . $\square$

DA

- (3) The *propagation D-cubes* of a logic element E specify minimal input conditions to the logic element which are required to propagate an error signal on an input (or inputs) to the output of that element. Let the logical behavior of E be defined by two sets of primitive cubes β_0 and β_1 which result in 0 and 1 outputs from E respectively. In order to propagate an error on an input line r , the other inputs must be defined so that if $r = 0$ the resulting cube is in β_0 , and if $r = 1$ the resulting cube is in β_1 , or vice versa. Such cubes can be derived by either (1) intersecting cubes in β_0 with $r = 0$ and β_1 with $r = 1$ (ignoring the r th position), resulting in $\bar{D}$ or D on both r and the output, or by (2) intersecting cubes in β_0 with $r = 1$ and β_1 with $r = 0$, resulting in D on r and $\bar{D}$ on the output or $\bar{D}$ on r and D on the output. Note that propagation D -cubes always occur in pairs which differ only in that all D components have complementary values.

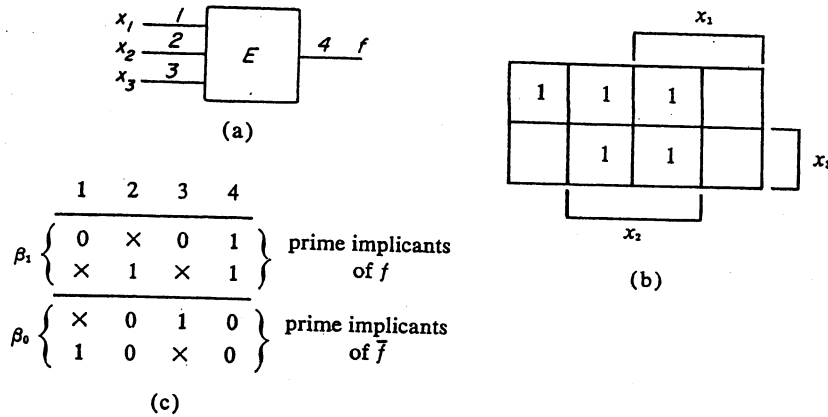


Figure 2.16

Consider the propagation of an error on line 1 through the logic element defined in Figure 2.16. The propagation D -cubes can be derived from the primitive cubes of Figure 2.16(c) by intersecting cubes in β_0 for which line 1 has the value 1 with those cubes in β_1 for which line 1 has the value 0. This results in the following propagation D -cubes:

1	2	3	4
$\bar{D}$	0	0	$\bar{D}$
$\bar{D}$	0	0	D

There are no cubes in β_0 for which line 1 has the value 0. Propagation D -cubes for multiple input faults can be similarly derived as illustrated in the following example.

Example 2.7: Consider the propagation of error signals $\bar{D}$ and D on input lines 1 and 2 respectively for the logic element of Figure 2.16. Intersecting the cubes of β_0 with 0 or $\times$ and 1 or $\times$ in positions 1 and 2 with cubes in β_1 with 1 or $\times$ in position 1 and 0 or $\times$ in position 2 we obtain the following propagation D -cube from the intersection of the second cube in β_1 with the second cube in β_0 .

1	2	3	4
$\bar{D}$	D	0	D

Note that the intersected cubes must not be differently specified in any of the positions corresponding to nonfaulty lines. $\square$

Figure 2.19 shows the primitive cubes, primitive D -cubes of all stuck faults, and propagation D -cubes for a 3-input NAND gate with input lines

1	2	3	4		1	2	3	4	Fault	1	2	3	4
0	$\times$	$\times$	1		0	$\times$	$\times$	D	4 $s-a-0$	D	1	1	$\bar{D}$
$\times$	0	$\times$	1	β_1	$\times$	0	$\times$	D	4 $s-a-0$	1	D	1	$\bar{D}$
$\times$	$\times$	0	1		$\times$	$\times$	0	D	4 $s-a-0$	1	1	D	$\bar{D}$
1	1	1	0	β_0	1	1	1	$\bar{D}$	4 $s-a-1$	D	D	1	$\bar{D}$
				(a)	$\bar{D}$	1	1	D	1 $s-a-1$	D	1	D	$\bar{D}$
					1	$\bar{D}$	1	D	2 $s-a-1$	1	D	D	$\bar{D}$
					1	1	$\bar{D}$	D	3 $s-a-1$	D	D	D	$\bar{D}$
									(b)				(c)

Figure 2.19 (a) Primitive cubes (b) Primitive D -cubes of faults (c) Propagation D -cubes for NAND gate

1, 2, 3 and output line 4. Additional propagation D -cubes can be obtained by complementing the D 's and $\bar{D}$'s in the cubes shown.

Having defined *primitive cubes*, *primitive D -cubes of a fault*, and *propagation D -cubes* we can now specify a test generation procedure which can be shown to be algorithmic in the sense that it will derive a test for any logical fault, if such a test exists. [24]

Procedure 2.2 (*D-Algorithm—Generation of a test for a logic fault*):

(1) Select a primitive D -cube of the fault under consideration. This produces the error signal D or $\bar{D}$ at the site of the fault. (Usually a choice exists. The initial choice is arbitrary but it may be necessary during the execution of the algorithm to return and consider another possible choice. This is called *backtrack*. Backtrack may have to be iterated until all choices have been considered.)

(2) *Implication*: In the execution of (1), some gate inputs or outputs may be specified so as to uniquely *imply* values on other signals in the circuit. The *Implication* procedure traces such signal determination both forwards and backwards through the circuit. Implication proceeds as follows: Whenever a previously unspecified signal value becomes specified, all the elements associated with this signal are placed on a list B and processed one at a time (and removed). For each element processed, based upon the previously specified inputs and outputs, it is determined if new line values of 0, 1 are implied.* These implied line values can be determined by intersecting the *test cube* (tc) (which specifies all previously determined signal values of the circuit) with the primitive cubes for this circuit element. If the line associated with the output of an element E is specified in the test cube, and only one primitive cube C_p of E has the same specified value on this line, C_p is intersected with the test cube. Similarly if *all* the specified inputs of some primitive cube C_p of E' are identical to the values associated with these lines in the test cube, C_p is intersected with the test cube. If any line values are implied, they are specified in the test cube and the associated gate indices are placed on the list B . When a line a is assigned a value due to implications from its inputs (forward implication), then all gates having a as an input are placed on B . If a is assigned a value due to the backward implication on element E , then the gate whose output is a is placed

*Here we have only determined signal values implied by inputs of 0 or 1. It is also possible to have signal values implied by inputs of D or $\bar{D}$. For example, for a 2-input NAND with inputs 1 and D the output is implied to be $\bar{D}$. For basic gate elements (AND, OR, NOT, NAND, NOR) implication of D 's and $\bar{D}$'s is simply determined. For example, for an AND element the output is implied to be $D(\bar{D})$ if and only if some input(s) are $D(\bar{D})$ and all other inputs are 1. Similarly the output is implied to be 0 if one input value is D and another input value is $\bar{D}$. For general complex elements the determination of implication of D and $\bar{D}$ inputs is much more complex. Since implication only affects the *efficiency* of execution of this procedure, we will not consider this problem in detail (See Problem 2.17).

onto list B along with all gates having a as an input, except for element E itself.

An *inconsistency* occurs when the value 0(1) is implied on a line l which has previously been specified to the complementary value 1(0). If an inconsistency occurs, backtrack to the last point a choice existed, reset all lines to their value at this point and begin again with the next choice.

(3) *D-Drive*. The *D-frontier* consists of the set of all elements whose output values are unspecified but whose input has some signal D or $\bar{D}$. (An associated list A consists of all of the elements in the *D-frontier*.) D -drive selects an element in A and attempts to propagate the D or $\bar{D}$ on the inputs of the associated element to an output of the element. This is accomplished by intersecting the current circuit test cube tc describing the circuit signal values with a propagation D -cube of the selected element of A , resulting in a new test cube. (For this intersection to exist all D 's ($\bar{D}$'s) of the propagation D -cube must correspond to D 's ($\bar{D}$'s) or unspecified entries of the test cube tc . If such intersection is impossible a new element in A is selected. If intersection cannot be accomplished for any element in A , backtrack to the last point at which a choice existed, resetting all lines to their values at that point, and begin with the next choice.

(4) *Implication of D-drive*: Perform implication for the new test cube derived in (3).

(5) Repeat (3) and (4) until the faulty signal has been propagated to an output.

(6) *Line justification (Consistency)*. Execution of (1)–(5) may result in specifying the output value of an element E but leaving the inputs to the element unspecified. The inputs to such elements are now specified to produce the specified output value by intersection of the test cube with the primitive cubes of the element. The test cube may be intersected with any primitive cube of the element E which has no specified signal values which differ from those of the test cube. Implication is then performed on the new test cube and the process is repeated until all specified element outputs have been justified. Backtracking may again be required. $\square$

The flowchart of the D -algorithm is shown in Figure 2.20. The boxes indicated with *** represent operations involving possible exhaustive consideration of several choices.

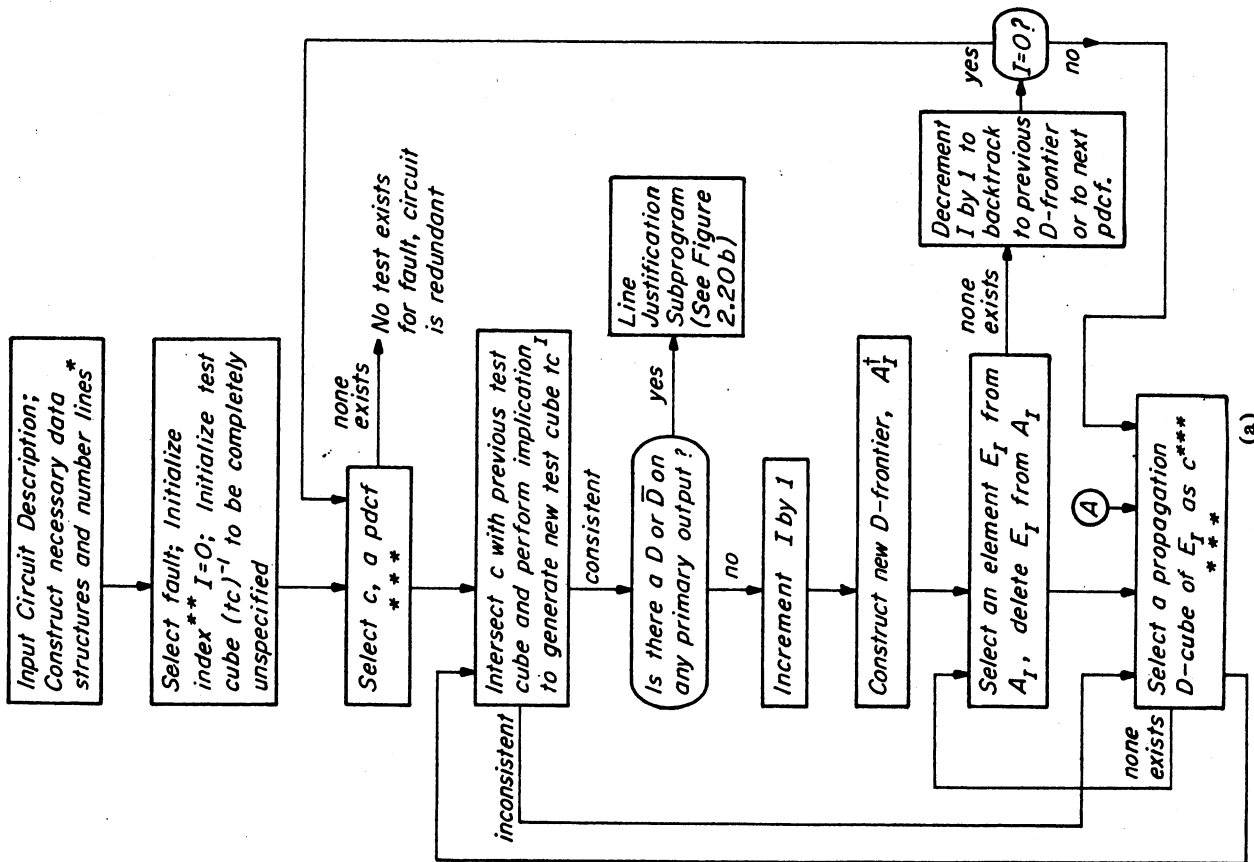


Figure 2.20 (a) D-algorithm Flowchart

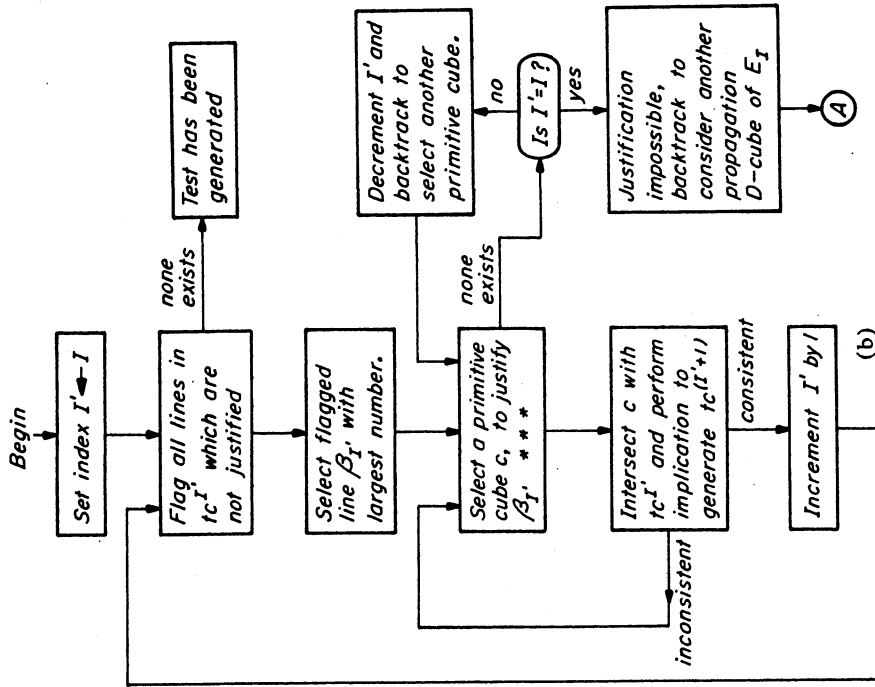


Figure 2.20 (b) Line Justification Subprogram Flowchart

*Lines are numbered (assigned integer values) in such a way that the number associated with the output of a gate is larger than the number associated with any of the input lines of that gate. This is done to facilitate line justification.

**The index I is used to keep track of the different test cubes and D-frontiers generated during execution and to enable backtracking.

***Indicates sequential consideration of a set of possible choices which can be represented as a list or stack but typically are generated dynamically.

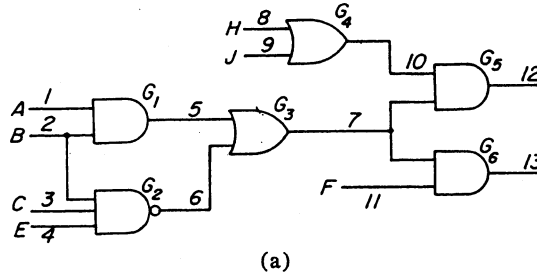
†The new D-frontier A_{I+1} can be constructed from the previous D-frontier A_I as follows: if we have just propagated through a_i , then

$$A_{I+1} = A_I - a_i \cup \{ \text{all elements fed by } a_i \text{ which have unspecified outputs} \}.$$

This is only valid if implication of D and $\bar{D}$ values is not performed.

PA11

Example 2.8: For the circuit of Figure 2.21(a) and the fault $G_1 s-a-0$, the table of Figure 2.21(b) lists the primitive D -cube of the fault and the primitive cubes of gates G_2, G_3, G_4, G_5, G_6 . Checking implication we determine that the primitive D -cube of the fault does not imply any other circuit signal values since only G_2 has a primitive cube with line 1 or 2 specified to value



(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13
Primitive D -cube of fault	1	1			D								
Primitive cubes of G_2		0	$\times$	$\times$		1							
		$\times$	0	$\times$		1							
		$\times$	$\times$	0		1							
		1	1	1		0							
Primitive cubes of G_3					1	$\times$	1						
					$\times$	1	1						
					0	0	0						
Primitive cubes of G_5							$\times$		0	0			
							0		$\times$	0			
							1		1	1			
Primitive cubes of G_1							0			$\times$	0		
							$\times$			0	0		
							1			1	1		
Primitive cubes of G_4								1	$\times$	1			
								$\times$	1	1			
								0	0	0			

(b)

Figure 2.21

	1	2	3	4	5	6	7	8	9	10	11	12	13
Initial test cube tc^0	1	1			D								
Propagation D -cube of G_3					D	0	D						
Test cube after D -drive through G_3 tc^1	1	1			D	0	D						
Test cube after implication	1	1	1	1	D	0	D						
Propagation D -cube of G_5							D			1		D	
Test cube after D -drive through G_5 tc^2	1	1	1	1	D	0	D			1		D	

Figure 2.22

1 and that cube also has on lines 3 and 4 signals specified to value 1. At this stage the D -frontier, A_1 , consists only of gate G_3 . The relevant propagation D -cube of G_3 is shown in Figure 2.22 (row 2), and row 4 of this table shows the test cube after the D -drive through G_3 .

Performing implication at this stage results in specifying lines 4 and 5 to have value 1 since line 6 has value 0. At this stage the D -frontier is $A_2 = \{G_5, G_6\}$. The relevant propagation D -cube of G_5 and the test cube after the D -drive through G_5 are shown in Figure 2.22, rows 5 and 6. Line justification of the 1 signal in position 10 results in the specification of line 8 (or 9) to have value 1. $\square$

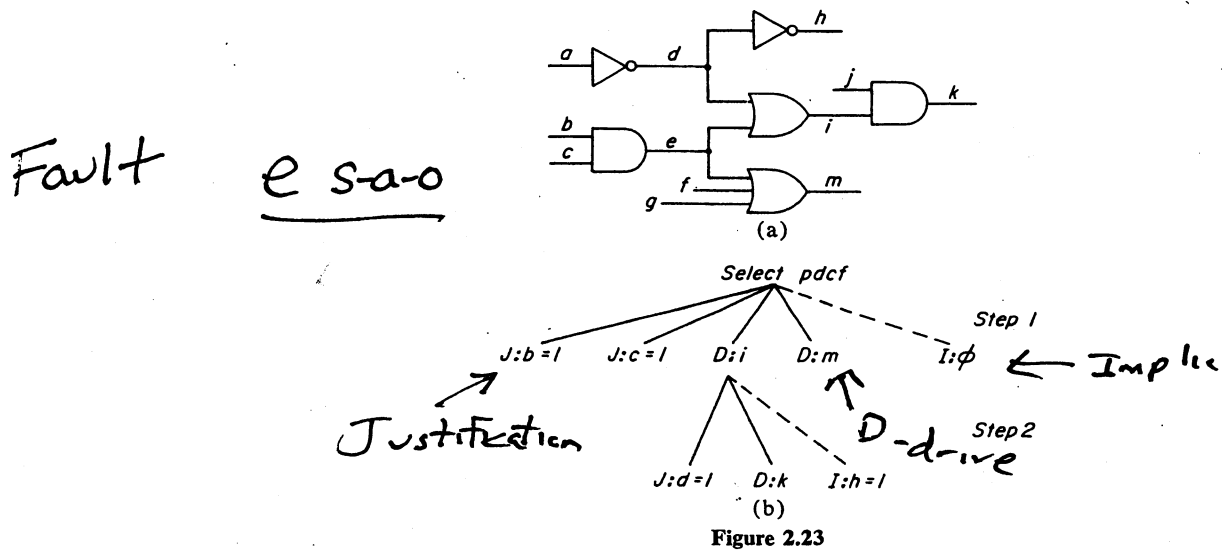
Note that backtracking may be required in line justification. If line justification leads to an inconsistency, we backtrack and consider the next alternative in the current D -frontier A_i . After exhausting all alternatives in this set we would backtrack to the previous D -frontier A_{i-1} and perhaps eventually to consideration of another primitive D -cube of the fault.

Backtracking

DA 1

D - algorithm represented as a tree structure.

Consider the circuit of Figure 2.23 and the fault e s-a-0. We first select the pdf $b = 1, c = 1$ which results in $e = D$. We now have a choice of



several subproblems to solve which can be represented by the tree of Figure 2.23(b), where J refers to line justification, D to D-Drive and I to implication. Implications are not considered to be problems, and are indicated by dotted lines.