C H A P T E R 7

HIERARCHICAL DIAGNOSIS

The test planning framework described in Chapter 5 employs the functional testing approach in which the behavior of a model is tested, and aims to validate the correctness of a system model. This chapter presents a structural testing approach in which module decomposition used in the hierarchical design is taken into account, and aims to isolate the faulty module in a hierarchical system.

7.1 Faults in Hierarchical Systems

Modern technologies require that digital systems be designed modularly and hierarchically. When testing a hierarchical system, it is desirable not only to know whether the system contains faults or not, but also to know which module of the system is the one causing this failure if a failure is observed. Figure 7.1 depicts the hierarchical structure of a digital device, which is decomposed into testable modules at different levels of abstraction, e.g., the board-processor-register-gate decomposition. All modules except the lowest level modules are composed of modules at one level below. Modules at the same levels are interconnected with signals.

When a module contains a fault, it is faulty and all the modules comprising it are also considered faulty. For example, if the module M_{121} is faulty, M_{12}, M_1, and M are also said to be faulty. When a module contains a fault, the fault either resides in one of its sub-modules or resides in the interconnect among the sub-modules. In this work, we assume that the interconnect is fault-free and a fault only resides in its sub-modules. Thus, if a module is faulty, at least one of its sub-modules must be faulty. Also, if the topmost module is faulty, at least one module at every level is faulty. A module is pronounced faulty if all its inputs are observed correct and any of its outputs is observed incorrect.

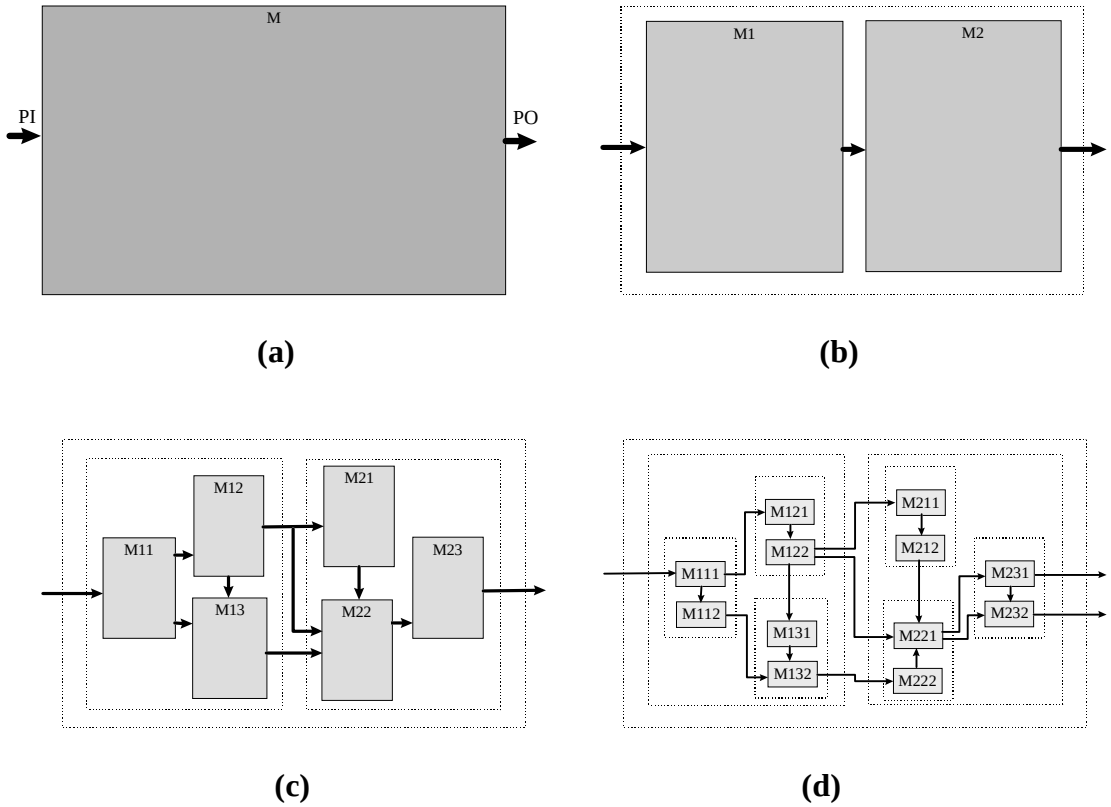


Figure 7.1 A Hierarchical Circuit

7.2 Top-Down Testing and Flat Testing

Isolating faulty modules involves incrementally probing the signals among modules to collect information for diagnosis. By applying the *incremental testing* approach to faulty module isolation, groups of the modules in the system are tested until, eventually, the faulty modules are pinned down. A form of incremental testing is called *top-down testing* [26, 50], in which the highest level modules are tested first [40]. The application of top-down testing to faulty module isolation is discussed as follows.

Assume that a failure is observed in system level testing for a certain test case and we want to isolate the faulty modules that cause the system to fail in the same test case. In the top-down approach, the system is first viewed as a network of modules at the second level. The interconnect among these modules is probed and eventually a faulty module at this level is identified. The isolated faulty module is then zoomed in on and viewed as a network of its sub-modules at the third level and diagnosis at this level starts. This process continues until the faulty module at the lowest level is isolated. After the faulty module at the lowest level is fixed, the system is tested again by the same test case to ensure that 1) the fault is indeed eliminated, and 2) no other lowest level modules combine to cause the same failure.

An alternative approach is called flat testing, in which the interconnect among modules at the lowest level is probed. At least one faulty module at the lowest level will be eventually isolated since, otherwise, the system will not fail the test case. Once the faulty module at the lowest level is identified, all modules that comprise the lowest level faulty module can be pronounced faulty.

The faulty module isolation problem is a search problem. The use of the cause-effect relation derived from the network topology can reduce the search space and thus accelerate the search process. For example, if the signal *A* is probed incorrect, the modules along the paths from primary inputs to the signal *A* are suspects since they feed the signal *A*. Therefore, the search time depends on the network topology of the system.

Figure 7.2 shows a system with serial topology at all levels. The search problem can make the most use of the cause-effect relation through binary search for this network topology. Both top-down testing and flat testing have a search complexity of $O(lw \log r)$, where w is the average total width of output signals of a module, r is the average number of immediate sub-modules of a module, and l the number of levels in the system. This is the best case that the search problem can be.

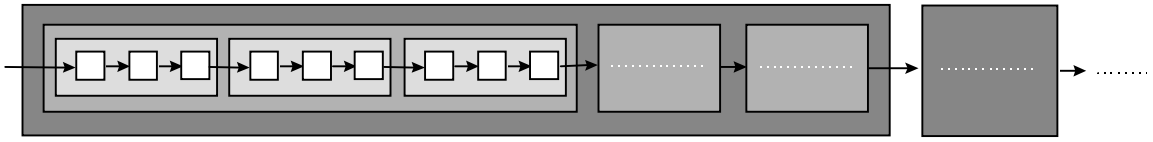


Figure 7.2 A System with Serial Topology at All Levels

Figure 7.3 shows a system with parallel topology at all levels. Since the modules at the same level are not connected with each other, the search problem makes no use of the cause-effect relation and binary search feature. The search complexity in both top-down testing and flat testing are $O(r^{l-1}w)$. This is the worst case that the search problem can be.

The circuits in Figures 7.2 and 7.3 illustrate two extreme cases of the search problem for faulty module isolation. In general, the complexity of the search problem is between these two cases. Although both top-down testing and flat testing have the same search complexity in each of the two extreme cases, top-down testing, in general, is a

better and more natural approach for hierarchical circuits designed modularly and hierarchically.

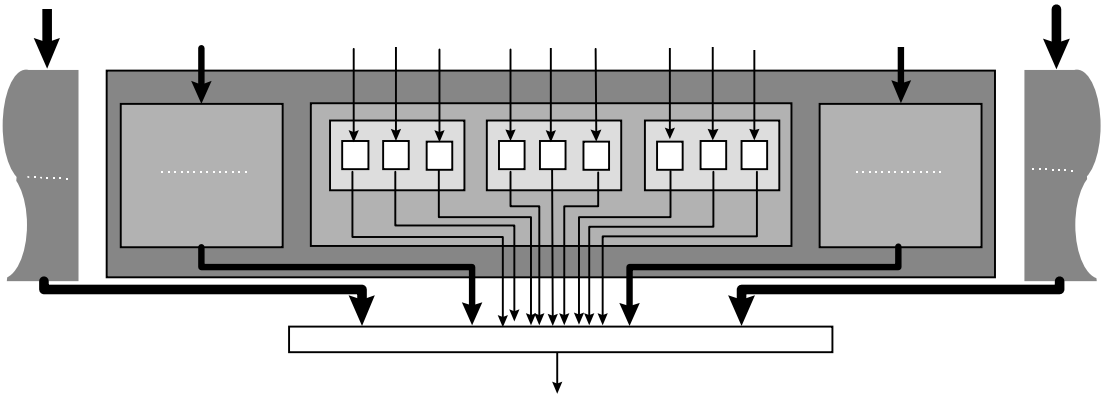


Figure 7.3 A System with Parallel Topology at All Levels

7.3 Exposability and Probe Selection Algorithm

Due to the complexity of the circuit and the limitation imposed by test equipment, it is not possible to probe all the signals at the same level at one time. Therefore, an efficient probe selection algorithm plays a key role in search of faulty modules. Selection criteria use cost functions and aim to reduce the overall cost by selecting check points. Selection criteria differ by the cost functions. Here, we define the *exposability* measure which reflects the extent that signal values are revealed to the tester. The exposability measures can be derived from measures of *controllability*, *observability*, and *testability*

which are used mainly as cost functions for testable design and test generation [4~6, 25, 70, 71]. But some modifications need to be made.

A signal is said to be an exposed signal if its value is known. Thus, primary inputs (PI) and primary outputs (PO) are exposed signals. The probed signals are also exposed signals since once they are probed, their values remain unchanged and known in the course of diagnosis for particular failure. For every signal l, two functions $E_f(l)$ and $E_b(l)$ are first computed. The *forward exposability* $E_f(l)$ indicates the extent that the value of l is exposed to the tester from an exposed point along the signal flow to l from PI's. The *backward exposability* $E_b(l)$ indicates the extent that the value of l is exposed to the tester from an exposed point against the signal flow to l from PO's. The *overall exposability* $E(l)$ is a composite function of $E_f(l)$ and $E_b(l)$, indicating the extent that the value of l is exposed to the tester from any exposed signals.

The cost functions for probe selection can be defined in as many ways as the cost functions that measure the controllability, observability, and testability are defined. As an example, a simple distance-based cost function is illustrated as follows.

Let S be the set of exposed signals. $\forall\ s \in S$, $E_f(s) = E_b(s) = 0$. For any signal $l \notin S$, $E_f(l)$ is defined to be the longest distance from an $s \in S$ along the signal flow to l. $E_b(l)$ is defined to be the shortest distance from l along the signal flow to an $s \in S$. E_f can be computed as follows:

1. Set E_f measures of all the exposed signals to 0.

2. Starting from PI toward PO, for all modules M whose inputs are all assigned the E_f values, mark all M's outputs as the largest E_f measure of M's inputs plus 1.

To compute E_b,

1. Set E_b measures of all the exposed signals to 0.

2. Starting from PO toward PI, for all modules M whose outputs are all assigned E_b values, mark all inputs to M as the smallest E_b measure of M's outputs plus 1.

The examples in Sections 7.5.1 and 7.5.2 illustrate the assignments of both E_f and E_b measures. Note that the algorithm for E_f works only for networks without feedback loops. Deadlocks occur in the presence of feedback loops. Although faulty module isolation in the presence of feedback loops is beyond the scope of this dissertation, the E_f algorithm is revised as follows for deadlock resolution and illustrated in Figure 7.4.

1. Set E_f measures of all the exposed signals to 0.

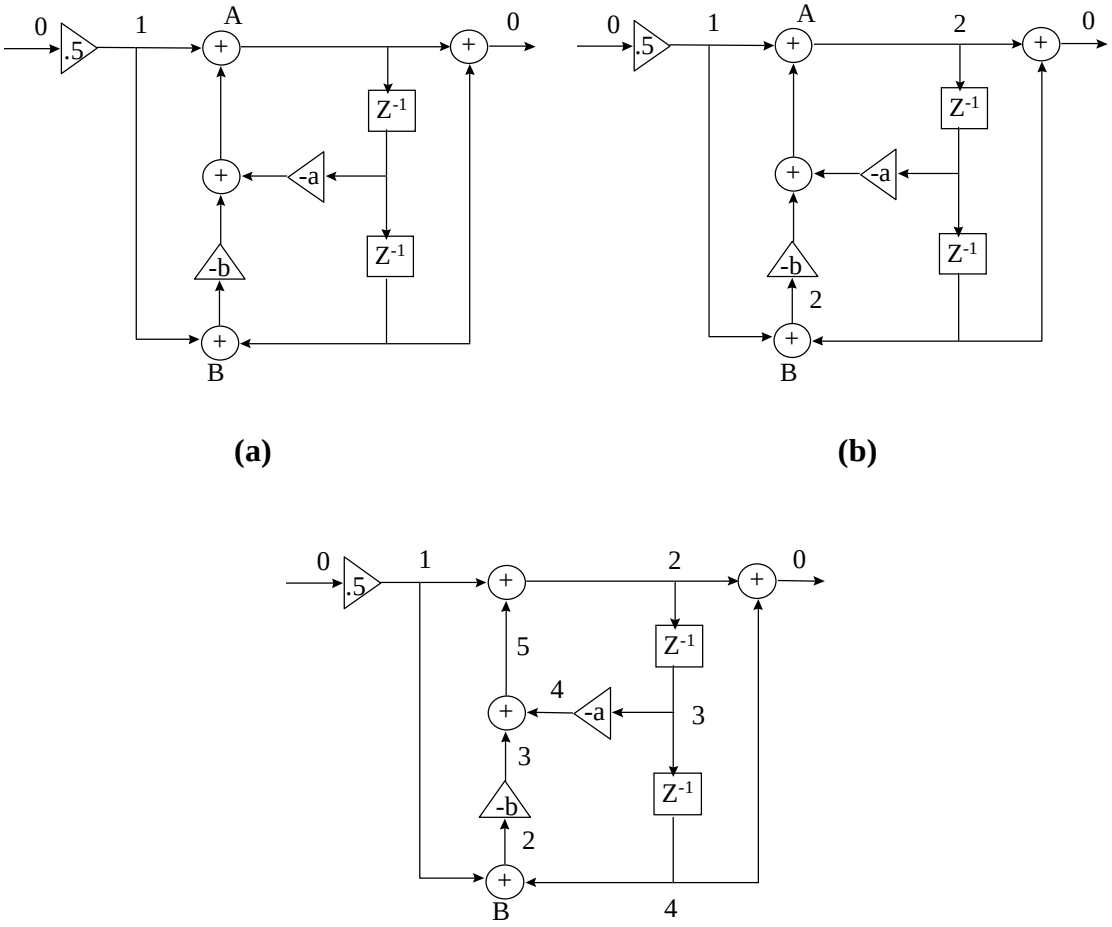


Figure 7.4 A Unity-gain Resonator with Feedback

2. Starting from PI toward PO, for all modules *M* whose inputs are all assigned the E_f values, mark all *M*'s outputs as the largest E_f measure of *M*'s inputs plus 1, until no more assignments can be made due to deadlocks. See Figure 7.4(a).

3. For all modules whose input signals are partially assigned due to deadlocks (i.e., modules *A* and *B* in Figure 7.4(a)), resolve the deadlocks by setting aside

their unassigned inputs and marking all their output signals as the largest E_f values of their assigned input signals plus 1. See Figure 7.4(b).

4. Go back to Step 2 and continue. See Figure 7.4(c).

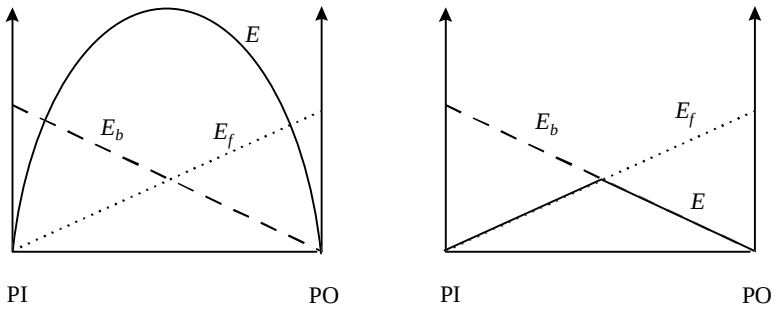
$E(l)$ can be defined in many ways. It can be defined as

$$E(l) = E_f(l)\, E_b(l) \tag{7.1}$$

or

$$E(l) = \min(E_f(l), E_b(l)) \tag{7.2}$$

or in other forms. No matter how $E(l)$ is defined, the distance-based cost functions have to hold the property that the exposed points are most exposable and have the lowest exposability ratings, and the points away from the exposed points are less exposable and have higher exposability ratings. That is, the exposability must be a convex curve. This property ensures that the cost function benefits most from the cause-effect relation through binary search. Typical graphs of E, E_f, and E_b in the definitions in Equations 7.1 and 7.2 are depicted in Figures 7.5(a) and 7.5(b), respectively. Equation 7.1 will be adopted hereafter as the definition of the exposability.



(a) **(b)**

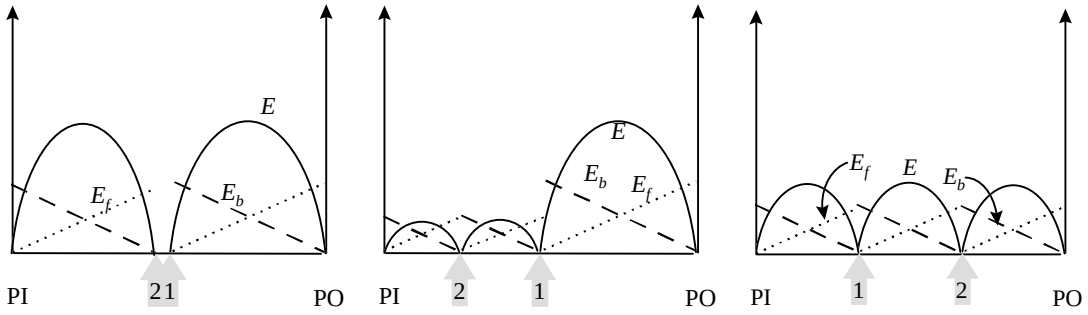
Figure 7.5 Typical Graphs for E_f, E_b, and E

$E(l)$ is used as a cost function for selecting check points. Let n be the number of check points allowed to probed simultaneously. The goal of probe selection is to reduce the overall cost. When n equals 1, selecting the signal with the highest exposability rating in every iteration always reduces the overall cost the most. When n is greater than 1, at least three selection heuristics can be adopted in the probe selection algorithm:

Heuristic 1: Pick n least exposable signals from the candidate probe set. This is a simple heuristic that needs only to compute the cost function once in order to pick n signals, but the performance is not good in term of cost reduction. Figure 7.6(a) illustrates the effects of applying Heuristic 1 when $n = 2$.

Heuristic 2: The algorithm is shown in Figure 7.7. These n selected signals are to be probed simultaneously. This heuristic takes more time since the cost function needs computing n times in order to pick n signals. However, it is much more effective in overall

cost reduction than the first heuristic. Figure 7.6(b) illustrates the effect of applying Heuristic 2 when $n = 2$.



(a) Heuristic 1 **(b) Heuristic 2** **(c) Heuristic 3**

Figure 7.6 Effects of Different Probe Selection Heuristics on Cost Function

---

```
repeat
    {
        select the least exposable signal
        assume it is exposed
        recompute the cost function
        update the candidate probe set
    }
until n signals are selected
probe these n signals simultaneously
```

Figure 7.7 Algorithm for Heuristic 2

Heuristic 3: Exhaustively try all possible combinations of n signals from the candidate probe set and then pick the combination that results in maximal overall cost

reduction. This heuristic is the optimal one in overall cost reduction, but is computation-expensive in searching for the optimal combination. It may not be affordable when the candidate probe set is large. Some restriction must be taken such as limiting the search time or limiting the number of searched combinations. Figure 7.6(b) illustrates the effect of applying Heuristic 3 when $n = 2$.

Figure 7.8 gives the algorithm for isolating the faulty module at a particular level.

---

1. *Compute E_f, E_b,* and *E for all candidate signals*
2. *Apply probe selection heuristic to select n signals from the candidate probe set*
3. *Probe these n signals*
4. *Reduce suspect module set and candidate probe set by cause-effect reasoning*
5. *If suspect module set contains only one module, it is the faulty module. Stop*
6. *Go to Step 1*

Figure 7.8 Algorithm for Isolating Faulty Module at a Particular Level

7.4 Differences between Exposability and Controllability/Observability

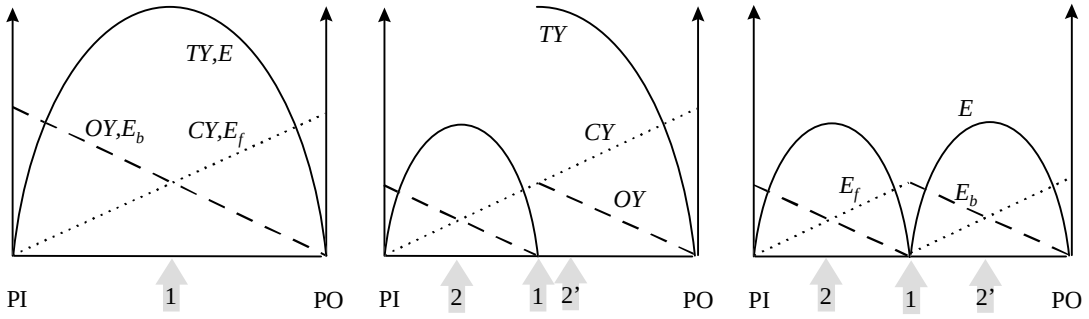
The cost functions differ from those defined for design for testability and test generation algorithms in the following ways.

(1) The techniques of design for testability aim at improving the overall testability (TY) of the circuits. They differentiate the controllability (CY) and the observability (OY), which measure the difficulties of setting and observing signals, respectively. The

controllability is improved by inserting control points as PI and the observability is improved by inserting observation points as PO. All the CY, OY, and TY are used as cost functions. By altering the design, the overall testability is improved according and the circuit becomes more testable. On the other hand, the objective of probing is to expose the current signal values for diagnosis, without altering the circuit design. Although the overall exposability is computed from the forward exposability and backward exposability, only the overall exposability is used as the cost function. When selecting probe points, there is no need to differentiate whether the circuit turns out to be more "forward" exposable or "backward" exposable.

Figure 7.9 explains why the traditional testability measures are not suitable as cost functions for probing. Figure 7.9(a) depicts the initial profile of the cost functions before probing starts. In the initial status, CY is the same as E_f, OY is the same as E_b, and TY is the same as E. Assume that only one signal is to be probed at one time. The first pick is half way between PI and PO. Figure 7.9(b) depicts the profile after first probing when using TY as the cost function. Since the value of the probed point is made known and will remain unchanged through this diagnosis session, we can view the probed point as a virtual PO. Thus, in the first half (between PI and the first pick), the observability is improved and the testability turns out to be a convex curve. So, the binary search feature still holds in this region. However, in the second half (between the first pick and PO), all measures remain the same and the testability decreases monotonically. The binary search feature no longer holds in this region and it ends up with the inefficient linear search.

Figure 7.9(c) depicts the profile after first probing when using the exposability measure as the cost function. E_f in the second half and E_b in the first half are both improved. E becomes convex in both regions which still get the most benefit out of binary search.



(a) Initial Profile **(b) TY after First Probing** **(c) E after First Probing**

Figure 7.9 Impact of Probing on Testability and Exposability

(2) In general, cost functions for test generation are static. They are computed by a preprocessing step and are not modified during test generation process [29]. But cost functions for faulty module isolation have to be updated dynamically to reflect the current status in each step.

(3) In design for testability (test generation), for "every signal" OY measures the difficulty of propagating the data (error) to "any" PO, whichever is easier. In diagnosis, the candidate probe points can be reduced by applying the cause-effect relation. Therefore, E_b needs to be computed for only those signals feeding the incorrect PO. As illustrated in

Figure 7.10, A and B represent the incorrect PO and only the shaded area which feeds A and B needs to be taken into account.

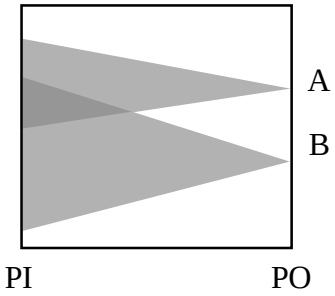
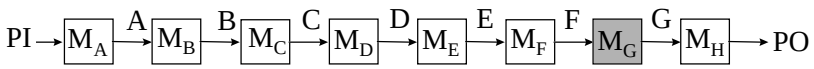


Figure 7.10 The Suspect Area for Incorrect A and B

7.5 Examples

7.5.1 A Linear Array Network

Figure 7.11 illustrates the probe selection algorithm for a linear array network example. Assume that only one probe is allowed at one time. The faulty module is marked with shade on it. In the first pass, the signal *D* has the highest exposability value and is thus first picked, probed and found correct. By applying cause-effect relation, the suspect module set reduces to $\{M_E, M_F, M_G\}$ and the candidate probe set becomes $\{E, F, G\}$. In the second pass, the cost function is recomputed and the signal *F* is probed next and found correct. The suspect module set reduces to $\{M_G\}$, so M_G is the faulty module. The faulty module is isolated in two passes.



(a) Network

l	PI	A	B	C	D	E	F	G	PO
$E_f(l)$	0	1	2	3	4	5	6	7	0
$E_b(l)$	0	7	6	5	4	3	2	1	0
$E(l)$	0	7	12	15	16	15	12	7	0

(b) Before First Probe

l	PI	A	B	C	D	E	F	G	PO
$E_f(l)$	0	1	2	3	0	1	2	3	0
$E_b(l)$	0	3	2	1	0	3	2	1	0
$E(l)$	0	3	4	3	0	3	4	3	0

(c) Before Second Probe

l	PI	A	B	C	D	E	F	G	PO
$E_f(l)$	0	1	2	3	0	1	0	1	0
$E_b(l)$	0	3	2	1	0	1	0	1	0
$E(l)$	0	3	4	3	0	1	0	1	0

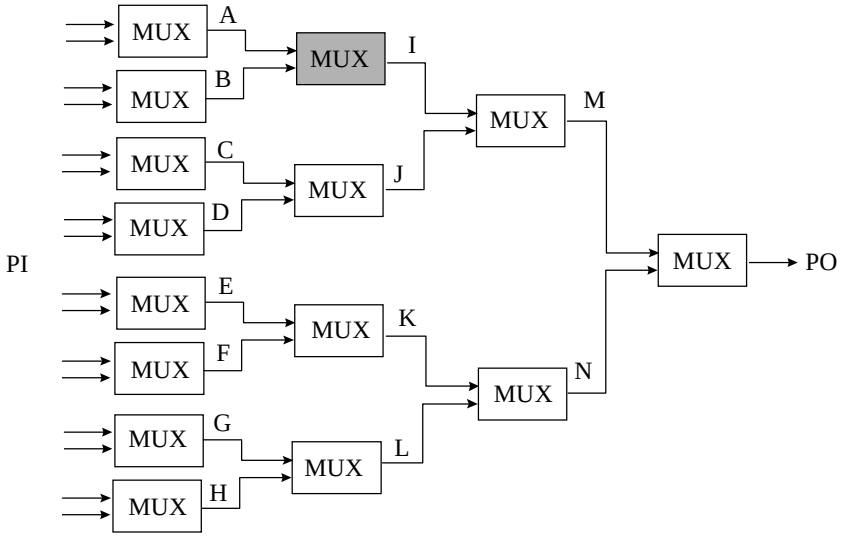
(d) After Second Probe

Figure 7.11 A Linear Array Network

7.5.2 A Tree Network

Figure 7.12 illustrates an example with a tree structure. It is a sixteen-input multiplexer constructed from two-input multiplexers. The *SELECT* and *ENABLE* lines have no influence on the exposability measures in this particular example so are not shown for clarity. Assume that the signal *K* is first picked and found correct. Thus, the signals *E*

and F are removed from the candidate probe set. The cost function is recomputed as shown in Figure 7.12(c). Note that $E_f(N)$ does not improved since, according to the definition, PI-*G*-*L*-*N* and PI-*H*-*L*-*N* are still the longest paths from an exposed point to N. Assume that the signal I is next picked and found incorrect. The candidate probe set is reduced to $\{A, B\}$. Assume that A and B are picked in the third and fourth passes and found correct. This pins down the faulty module to be the one whose output is I. After this, gate level diagnosis focusing on the gates in the faulty multiplexer continues.



(a) The Network

l	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$E_f(l)$	1	1	1	1	1	1	1	1	2	2	2	2	3	3
$E_b(l)$	3	3	3	3	3	3	3	3	2	2	2	2	1	1
$E(l)$	3	3	3	3	3	3	3	3	4	4	4	4	3	3

(b) Before First Probe

l	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$E_f(l)$	1	1	1	1	1	1	1	1	2	2	0	2	3	3
$E_b(l)$	3	3	3	3	1	1	3	3	2	2	0	2	1	1
$E(l)$	3	3	3	3	1	1	3	3	4	4	0	4	3	3

(c) Before Second Probe

l	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$E_f(l)$	1	1	1	1	1	1	1	1	0	2	0	2	3	3
$E_b(l)$	1	1	3	3	1	1	3	3	0	2	0	2	1	1
$E(l)$	1	1	3	3	1	1	3	3	0	4	0	4	3	3

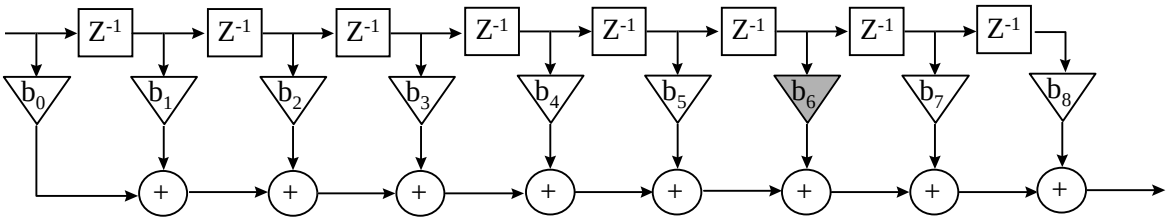
(d) After Second Probe

Figure 7.12 A Tree Structure

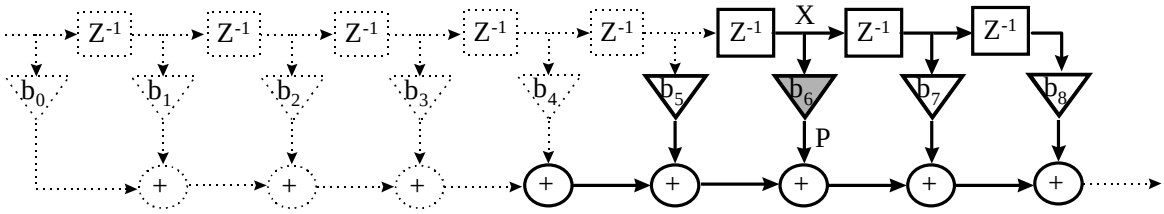
7.5.3 A Ladder Network

Figure 7.13 depicts an example of an eighth-order transversal FIR filter which implements the difference equation $y(n) = \sum_{m=0}^{8} b_m x(n-m)$. The data signals are of word-width, say, 8 bits. Assume that a data word can be probed at once. After the first three iterations, modules in the left part of the circuit (shown as dashed part in Figure 7.13(b)) are removed from the suspect set by cause-effect reasoning. The remaining part takes another 2 to 5 iterations to identify the faulty module which is a multiplier, depending on which signals to choose first among those of the same exposability measures. Figure 7.13(c) shows the faulty multiplier zoomed into the next lower level. The multiplier is constructed by six carry-save adders, gating logic, and a parallel adder. Assume that all

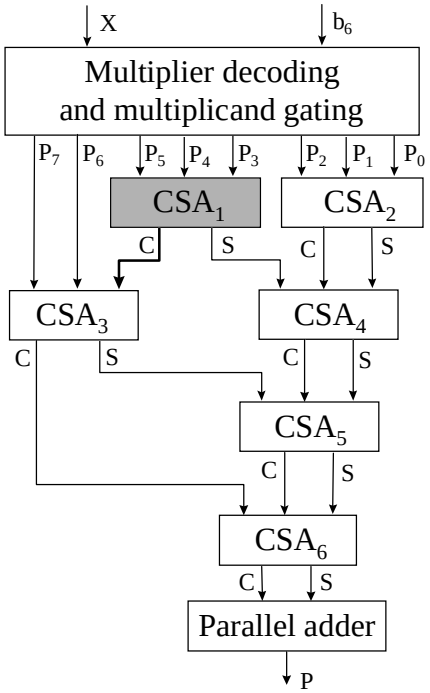
signals at this level can be probed at one time. Also assume that the module CSA_1 is faulty and the fault is propagated through its carry output C which feeds the module CSA_3. Isolating this faulty module takes 5 to 8 iterations. Similar procedure can be applied to isolate the faulty gate of the faulty CSA.



(a) The Whole Network



(b) After 3 Iterations



(c) The Faulty Multiplier Viewed at Next Level

Figure 7.13 A Ladder Structure

7.6 An Expanded Goal Tree

Chapter 5 has discussed the functional goal tree concept in the test planning framework where test goals are defined based on the functionality of circuits. This section presents a goal tree that explores both the structural and functional aspects of a structural circuit.

Assume that the FIR filter is a board that is composed of 25 chips as shown in Figure 7.13(a). Figure 7.14 illustrates an expanded goal tree for the FIR filter. It has two

major subgoals: the testing goal and the diagnosis goal. The testing goal explores the goals of test generation, evaluation, and test set compaction. The test generation goal is divided into then functional goal where the behavior of the FIR filter is tested at board level, and the structural goal where structural testing approaches are applied to chips of different types. The diagnosis goal specifies how the filter should be diagnosed when it is tested faulty. It consists of two goals: the wire connection fault goal and the faulty module isolation goal. Test groups are the leaves of the tree. The test group of the faulty module isolation goal provides the information needed to configure the diagnosis process, such as the approach to be used (top-down or flat), test files that cause the filter to fail, cost functions, number of probes, probe selection heuristics, the cause-effect reasoning algorithms, and so on.

The expanded goal tree explores both functional and structural aspects of a circuit. It also unifies various phases of testing into the same test plan. Chapter 8 will discuss some future areas that expand the application of the goal tree concept.

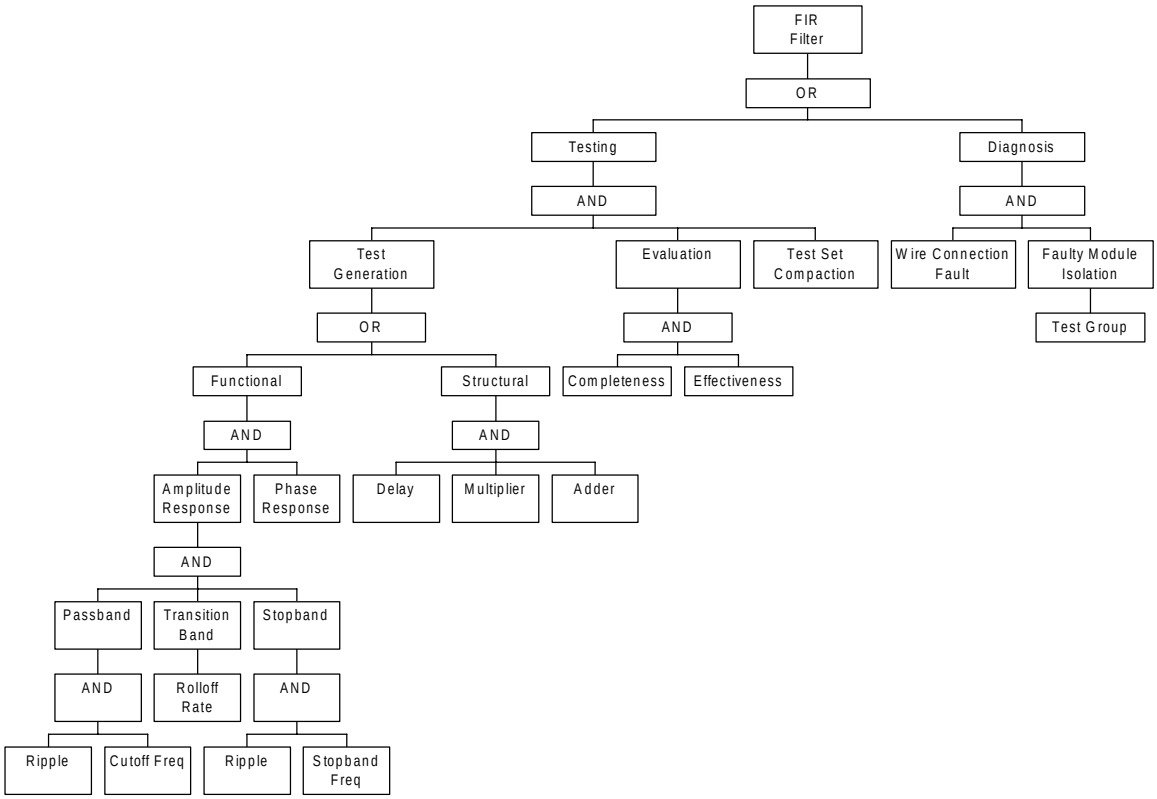


Figure 7.14 Expanded Goal Tree