Appearing: 1992 IEEE Workshop on Defect and Fault Tolerance in VLSI Systems

# Scan-Based Testability for Fault-Tolerant Architectures

André DeHon
MIT AI Lab
545 Technology Square
Cambridge, MA 02139

**Abstract**

*The acceptance and use of standard scan-based Test Access Ports (TAPs), such as the IEEE-1149.1-1990 standard, have begun to ease the task of system testability and in-circuit diagnostics. The typical singular nature of these TAPs along with the all-or-nothing manner in which test facilities are accessed make such standard TAPs inappropriate for use in fault-tolerant architectures. We propose three simple additions to standard scan practices which allow scan techniques to be effectively utilized in fault-tolerant environments. Specifically, we advocate the incorporation of multiple-TAPs, port-by-port selection control, and partial external scan. Multi-TAP construction offers tolerance to faults in the scan path or circuitry. Port-by-port selection and partial external scan allow fault-diagnostics which are minimally intrusive and in-operation reconfiguration for fault-masking and repair.*

## 1 Introduction

With the standardization of Test Access Ports (TAPs) and boundary-scan techniques in IEEE-1149.1-1990 [4], vendors are beginning to make components with scan-based TAPs readily available. Nonetheless, the facilities offered by TAP interfaces such as the IEEE-1149 standard are not well-suited for fault-tolerant system architectures. The singular and serial nature of the scan path exposes a critical single point of failure in the test system. Architects are forced either to use a few long serial scan chains or to use many short scan chains. The former allows a fault in a scan path to affect a large number of components while the latter requires significant wiring for the control of many scan paths. Furthermore, standard TAPs provide no facilities for bringing small portions of the system into test-mode while leaving the remainder of the system in normal operation. In fault-tolerant architectures where the system can function without all components on-line, these all-or-nothing testing modes can be inconvenient.

In this paper, we present three simple additions to standard scan practices which allow scan techniques to be utilized effectively in a fault-tolerant setting. The basic techniques introduced are:

1. Multi-TAP scan architecture – each component is given multiple Test Access Ports allowing the component to be accessed from any of several scan paths.

2. Port-by-port selection – each *channel* (Section 3.2) on a component can be independently disabled.

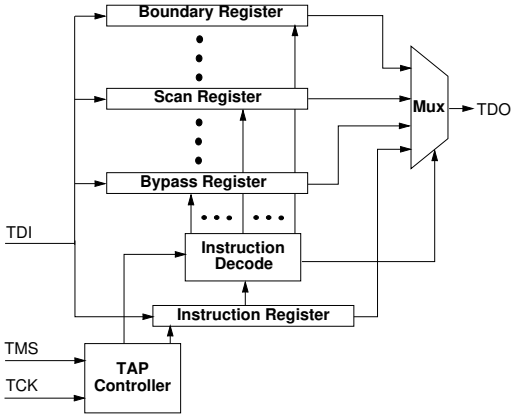



Figure 1: Standard IEEE TAP and Scan Architecture

3. Partial-external-scan – each *channel* can be scanned in boundary-test mode independently of the operation of other channels on the same component.

These additions are developed in Section 4.

We further show how the aforementioned additions combine to provide a scan architecture which is well adapted for the class of fault-tolerant systems described in Section 3. In particular, the additions allow:

1. Minimized impact of scan path faults on system diagnosability (Section 4.1)
2. Minimally intrusive in-operation fault-diagnosis (Section 5)
3. In-operation reconfiguration for:
   - fault-masking (Section 6)
   - repair (Section 7)

The paper opens with a brief review of standard TAPs and sparing based fault-tolerance techniques.

# 2 Background

## 2.1 IEEE-1149.1-1990 TAP

The IEEE Standard TAP [4] defines a serial test interface requiring four dedicated I/O pins on each component. The standard allows components to be daisy-chained so that a single test path can provide access to many or all components in a system. The standard provides facilities for external boundary-scan testing, internal component functional testing, and internal scan testing. Additionally, the TAP provides access to component-specific testing and configuration facilities. Figure 1 shows the basic architecture for an IEEE scan-based TAP.

In a system in which all components comply with the standard, boundary-scan testing allows complete structural testing. Using the serial scan path, every I/O pin in the system can be configured to drive a logic value or act as a receiver. Using the same serial scan path, the value of every receiver can be sampled and recovered. This mechanism allows the TAP to verify the complete connectivity of the components in the system. All connectivity faults, shorted wires, stuck drivers or receivers, or open-circuits can be identified in this manner [8] [17].

The scan path allows data to be driven into a component independent of the values present on the component's external I/O pins. The resultant values generated by the component in response to the driven data can similarly be sampled and recovered via the serial scan path. This facility permits functional in-circuit verification of the component.

The standard allows additional instructions which may function in a component-specific manner. These instructions provide standard access to internal-component scan-paths. Such internal paths are commonly used to allow a small number of test-patterns to achieve high-fault coverage in components with significant internal state. Other common additions are configuration registers and Built-In-Self-Test (BIST) facilities [10] [12] [11].

### 2.2 Fault tolerance from sparing

Fault-tolerant architectures can take advantage of system reconfiguration to *mask*, or hide, the effects of failures of components or subsystems. As long as the system has functional units available to assume all required tasks, operation can continue unaffected by the presence of masked faulty components. Faulty components must be identified in a timely manner and masked in order for the benefits of reconfigurability to be realized. System performance will, of course, generally degrade as components fail.

Figure 2 shows an abstract system composed of three different kinds of functional units and I/O connections. As faults occur, the system can be reconfigured to avoid the faulty components or links. As long as the system has the minimal configuration shown with the example as a non-faulty sub-graph, it is still functionally complete.

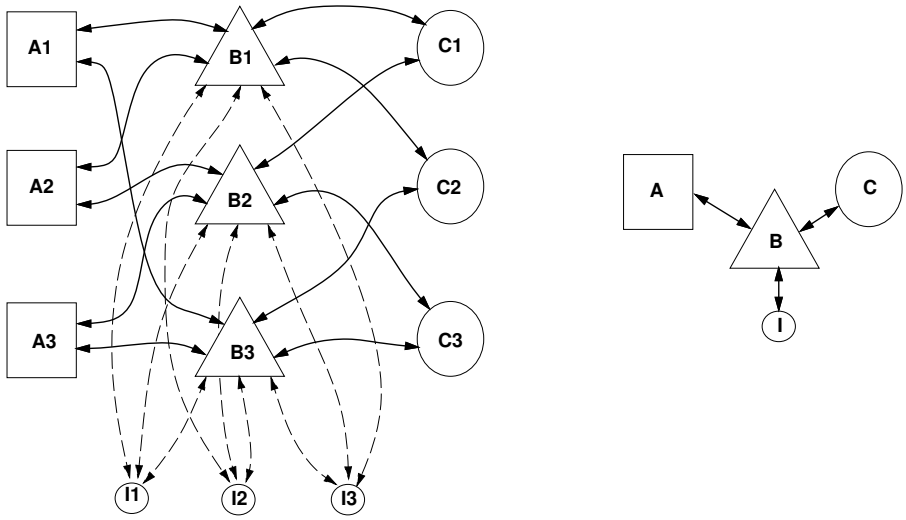
## 3 Operational models

### 3.1 Fault model

For the sake of discussion, we assume a simple structural fault model. Basic functional units can fail in some manner which can be reliably identified with a finite number of static test patterns. These test patterns may involve the use of internal scan paths inside the basic functional units. Connections between functional units are made with wires. The wires and component input/output interfaces may have transient faults due to crosstalk, or noise. Similarly, wires and component i/o structures may develop permanent faults in the form of shorted wires, open connections, or stuck-at wires.

### 3.2 Fault-tolerant system model

Abstractly, a system is composed of many subsystems, each of which performs some function necessary for the composite system to perform properly. In a reconfigurable, fault-tolerant system, any of a number of physically distinct components can perform any given function which is required by the system. During normal operation a subset,




Shown above (left) is an abstract system with redundant functional units that can take advantage of reconfiguration to remain operational when interconnection channels or components fail. A, B, and C represent three different kinds of functional units, each of which is replicated. I represents I/O connections which are also replicated. The arrows represent channels connecting functional units. As long as the system has a non-faulty subgraph as shown on the right, the system may continue to operate.

Figure 2: Fault-Tolerant Example

perhaps even all, of the functional components will perform the necessary tasks. When faults arise, the system can be reconfigured such that the faulty portion is not used. Operation is redirected to non-faulty components and the faulty components are ignored. Hayes develops this kind of fault-tolerant system in detail in [9].

For simplicity, let us think of a functional unit as a single integrated circuit component. Functional units are interconnected in order to realize the overall behavior of the system. Units are connected to each other via bundles of wires, referred to as *channels*. We aim to construct a sparing architecture where faulty components can be avoided. Each functional unit must be interconnected to multiple functional units capable of performing each task needed by the functional unit. When an adjacent functional unit, or its interconnection channel, is identified as faulty, the non-faulty functional unit can be reconfigured to avoid the faulty unit. As long as at least one adjacent functional unit capable of performing each different task remains connected to each non-faulty component via non-faulty channels, functional operation may continue.

It is easiest to think of each IC component in the system as a separate such functional unit interconnected by channels composed of wires. However, in general, the boundaries of functional units may be placed elsewhere. A single IC may contain multiple functional units, or a collection of ICs may serve as a single functional unit. Consequently, channels may be composed of traces on printed circuit boards, silicon or metal inside ICs, cables between boards, optical connections over fiber or free-space, or some combinations thereof.

# 4 Fault-tolerant testability additions

## 4.1 Multi-TAP

Supporting multiple test access ports on a single component is a simple extension of the redundant resource and interconnect ideas. With multiple test access ports, a component's scan capabilities can be accessed through any of multiple serial scan paths. This allows the component to be tested and reconfigured even when there are faults along one of its scan paths. Further, with multiple TAPs on a single component, scan paths can be arranged so that a minimum number of components are severed from the scan test system by multiple scan-path faults. For instance, we can arrange the scan paths in a system with dual-TAP components such that no two components are on the same pair of scan paths. This guarantees that two faulty scan paths will make at most one component inaccessible. Figure 3 shows a gridded topology which has this property.

When adding redundant scan access to a component, there are several issues which must be addressed to assure us that we can realize the potential benefits of having multiple TAPs. We must address the issue of resource contention between the scan paths, *e.g.* two scan paths cannot both perform a boundary scan through the same component at the same time. We must always have the ability to control a component's scan paths from a non-faulty path. This means we must be able to minimize or eliminate any potential for interference from any faulty path(s). We can achieve these goals using two simple techniques:

1. Resource conflict resolution in favor of the most recent requester

2. Sparse encoding of scan instructions

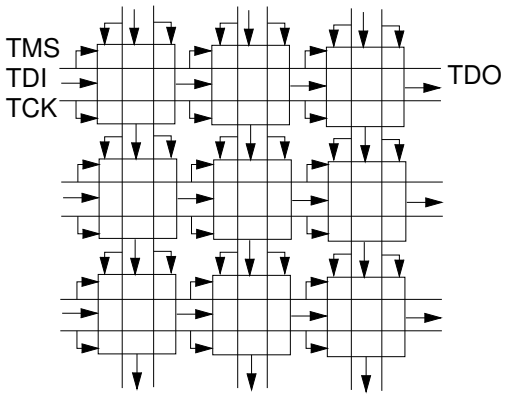



Figure 3: Mesh of Gridded Scan Paths

#### 4.1.1 Conflict resolution

Presumably, access to the scan paths is being coordinated at some level in the system. If everything is working properly, there should never be a resource conflict within a component. However, we are concerned with assuring that reasonable behavior will result even when parts of the system are not behaving properly. We give each TAP its own instruction register and bypass register. These registers behave exactly as in a standard TAP [4]. Differences in TAP behavior arise when multiple TAPs attempt to access the same scan registers. This would occur whenever the different TAPs attempted to load in instructions that referenced the same scan paths on chip. The simple conflict resolution scheme we propose is to give the TAP loading an instruction most recently access to the path. When the new instruction is loaded, the instruction in any conflicting TAP is reset to the bypass instruction. Since each TAP has its own bypass register, there will be no conflict for access to the bypass register. Assuming we can sufficiently minimize the chances that a faulty scan path can successfully load a non-bypass instruction into its instruction register, this scheme satisfies our fault-tolerance criterion. The scheme allows a non-faulty scan path to wrest a component's scan resources away from a faulty scan path. Figure 4 shows a possible architecture for a component with two test access ports.

#### 4.1.2 Sparse scan instruction encoding

The boundary-scan protocol for loading instructions is sufficiently involved as to prevent a faulty scan path from successfully loading an instruction in most cases. However, we would like a stronger guarantee that faulty behavior will not interfere with non-faulty access to a component. Simple faults, such as stuck-at faults on the clock (TCK) or mode (TMS) lines will prevent a path from being able to load an instruction. A stuck-at fault in the data lines or data-path of a component (TDI, TDO) will force the downstream component TAPs to see all zeros or ones, making it possible for faults in the data lines to cause instructions with all zeros or ones to be loaded. Of course, stuck-at faults are not the only kind of fault our system must contend with. Sparse instruction encoding is a simple way to make the chance that a faulty path can load a valid instruction arbitrarily small.

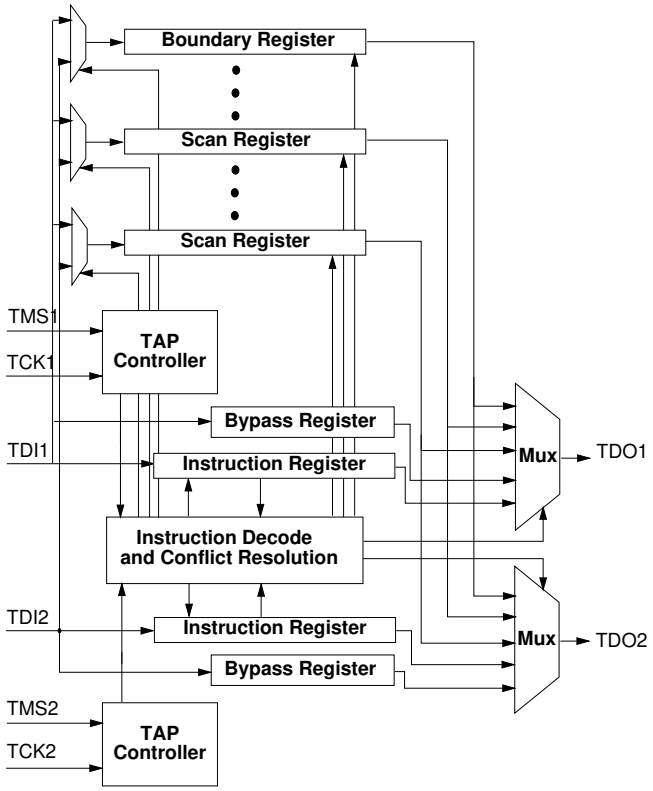



Figure 4: Scan Architecture for Dual-TAP Component

The basic idea in sparse encoding is to make the number of legal encodings small in comparison to the number of possible encodings. The non-legal instruction encodings all get treated as bypass instructions so that they cannot interfere with the normal operation of the component. Error correcting and detecting codes in common use for data storage and transmission [7] [15] are common examples of sparse encodings. In this application, we are concerned with detecting errors and preventing them from corrupting non-faulty operation, not correcting errors. If, for example, we used a simple instruction encoding scheme which computes an $m$-bit checksum on an $n$-bit data word, the space of possible instruction words is $2^{n+m}$ whereas the space of legal instruction codes is $2^n$. If we assume that the clock and mode bits behaved in exactly the correct manner to load in an instruction, but that the data lines held random data, the chances of a legal code word getting loaded are:

$$P_{random_load} = \frac{\text{number of legal codes}}{\text{number of possible codes}} = \frac{2^n}{2^{n+m}} = 2^{-m}$$

Of course, when choosing a checksum, one should make sure that the all zero and all one code words are not legal, checksummed instruction encodings.

McHugh and Whetsel propose adding parity to instruction encodings [13] to identify corrupted instruction words. Sparse encoding is a more general encoding scheme which allows stronger protection against data corruption.

#### 4.1.3 Costs

Reviewing the dual-TAP example shown in Figure 4, we see that the additional costs associated with a multi-TAP component are:

- Four additional I/O pins per additional TAP
- One additional instruction and bypass register per additional TAP
- One additional output MUX per additional TAP
- One conflict resolution unit
- Additional input MUXes for each shared register path

For most modern components, the limited resource is I/O pins rather than silicon area. As such, the additional I/O pins will generally be the first order cost associated with a Multi-TAP controller. Note that the size of the conflict resolution unit is proportional to the product of the number of potentially shared resources and the number of TAPs.

#### 4.1.4 Compatibility

As noted above, in the fault-free case, if both scan paths through a component do not attempt to access the same component register, the multi-TAP component will behave identically to a standard single-TAP component. Multi-TAP components place an additional burden on the software to assure that the scan paths through a given component never attempt to load conflicting instructions. In the faulty case, as long as there is a non-faulty path through a component, the faulty-free path can be used as a standard TAP as long as the faulty path does not manage to load a conflicting instruction. A standard single-TAP component may be used in a system or scan path with multi-TAP components, but the single-TAP component is susceptible to any faults in its single TAP or TAP control lines.

### 4.2 Port-by-port selection

Adding the ability to disable each channel into a component on a port-by-port basis allows us to mask faulty channels and components from the system. The semantics of *disabling* a channel in this manner imply that the component will ignore the channel throughout the time in which the channel is disabled. This means the component will not acknowledge any activity on the disabled channel, and the component will always choose to avoid the disabled channel when seeking service. Sections 5, 6, and 7 go into further detail on the utility of this addition. From the scan path, port selection/deselection is accessed as an internal component configuration register.

### 4.3 Partial external scan

Once we have a way to selectively remove some ports on a component from normal operation, it makes sense to be able to perform scan testing on each component on a port-by-port basis. This capability gives us a finer granularity control over the scan paths allowing us to perform scan tests on subsets of the system while the rest of the system remains in operation.

To support partial external scan, the component needs to handle additional instructions aimed at selecting the appropriate subset of the normal boundary-scan path. Additional MUXes in the boundary-scan path will be necessary to bypass the portions of the normal boundary path which are not being scanned during a particular partial scan operation.

## 5 Fault identification

Assuming we have some initial warning that faults may exist in the system, the component TAP and scan path provide the facility for localizing faults and determining with higher accuracy the nature of the fault. The initial theory can come from warning signs such as bad checksums on data, protocol violations, unusually poor performance, or periodic testing. The facilities existing for formulating fault theories are seldom sufficient to pin-point the source or extent of the error. They often cannot distinguish which component is at fault or even whether the problem is in a component or in the interconnection. Further, the existence of transient errors on the wires makes it necessary to distinguish between physical faults and a noisy environment.

In the most naive case, we could move the entire system into test mode and use the standard boundary and internal scan facilities to test the integrity of every connection and every component. In this manner, all structural faults in the interconnection can be identified and all functional component faults matching our model (Section 3.1) can be determined. Real faulty wires and components can be differentiated from transient faults and overloaded system operation which can trigger false fault theories.

However, if the system is large, the impact of removing the entire system from normal operation for testing can be significant. The larger the system, the higher the rate of single component faults and the larger the amount of hardware that must be removed from service for diagnosis. For sufficiently large systems, it is often neither economical nor practical to remove the entire system from service.

With the additions described in Section 4, we can make the testing significantly less intrusive. The addition of port-by-port selection and partial external scan provides fine-grain control of scan testing. At a given time, we can isolate a minimal subset of the

system that is suspected faulty and perform functional and scan testing. By disabling the channels of all components connected to a physical set of wires and performing scan tests on just those channels on those components, we can quickly determine the integrity of the interconnection in question. Similarly, by disabling all channels on components connected to a given component, we can isolate the single component in question from the network to perform functional testing on that single component. In both cases, the rest of the system may continue normal operation while testing occurs.

This scheme provides a capability for fault-identification and localization which is minimally intrusive. The information gained from this scan testing provides detailed information about the nature and extent of suspected faults. With this information, the system is in a much better position to diagnose the extent of faults, perform reconfiguration to avoid faults, and assess the risks associated with continued operation.

# 6 Reconfiguration

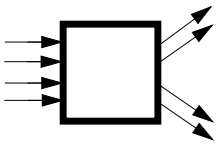
When faulty functional units or interconnections are identified, the fault can be masked by reconfiguring the system to avoid the faulty component. Again, the scan-based TAP provides an effective interface to this reconfiguration. The ability to disable a component's usage of a channel, described previously, provides one effective means of fault avoidance. If an entire unit is faulty, leaving every channel on every component connected to the faulty component in a disabled state will remove the unit from the functional portion of the system so that it cannot interfere with correct operation. Similarly, if faults occur in the wires, drivers, or receivers of an interconnection channel, disabling the channel on all affected components will effectively excise the faulty connection from the system.

This mechanism of disabling individual channels works effectively for reconfiguration for exactly the same reasons it was necessary for fine-grained diagnosis. The fault-tolerant model assumes that other channels remain enabled and connected to functional units which will provide functionally equivalent service to the ones whose channels are disabled. The semantics of disabling a channel imply that the component will ignore the channel throughout the time in which the channel is disabled.

Further, if a functional unit provides sparing within itself, the scan mechanism can be used to reconfigure the unit to swap spares. For some I/O limited components, there is plenty of additional room for function inside a component whose size is dictated by the pin-limited I/O. In these cases, it may make sense to provide redundant structures on the component. Faults in a structure can then be masked by reconfiguring the component to use an alternate, functional structure on the component.

# 7 Repair

The combination of accurate fault-localization coupled with the ability to perform reconfiguration, allows us to realize systems where the fault-repair loop can be closed without human or mechanical intervention, at least up to the fault-level provided by the sparing architecture. Programs monitoring the system integrity are empowered to test theories about faults and reconfigure the system to best mask the effects of failures. Further, with a knowledge of the minimal requirements necessary for complete system operation along with an accurate idea of the fault status of the machine, the overall system integrity can be assessed.



The dilated routing component shown here can route connections in one of two logical directions from any of its four equivalent input ports. Two physical output ports serve each logical direction.

Figure 5: Dilated Routing Component

When outside intervention is necessary to repair the system, these same facilities of channel disabling and channel based scan allow for in-operation replacement. If all the channels on all components into a physically replaceable subsystem are disabled, it is possible to replace the physical subsystem without any further interruption of system operation. Of course, the electrical and mechanical design of the system must also be suitable for live replacement (*e.g.* Tandem Non-Stop computer systems [1], Stratus fault-tolerant computer systems [18], Thinking Machines CM5 [16]). Once replaced, scan testing can determine the interconnection and functional integrity of the replaced component. When the replacement is properly installed and identified as functional, the disabled channels into the replaced subsystem can be re-enabled allowing the subsystem to return to full-service.

# 8 Example

As an example, let us consider a fault-tolerant multistage routing network built using dilated routing switches such as the RN1 routing component [14]. Consider one constructed from $4 \times 2$ dilation 2 routing components (See Figure 5). Each routing component has four equivalent input channels and four output channels which are divided into two logical output directions. Messages are routed from any of the four input ports to one of the two output ports in the desired logical direction. Each of the 8 ports in and out of the dilated routing component defines a separate channel which can be independently enabled, disabled, and scanned.

These components can be configured into a network with multiple paths between all endpoints as shown in Figure 6. In Figure 6 all of the paths between a pair of endpoints are highlighted; in a similar manner, there are many paths between every pair of network endpoints. At each stage, each routing component involved in making the connection can utilize either of its equivalent outputs to route the connection. This network has the desired structure described in Section 3.2. If a channel or component becomes faulty, it can be avoided by disabling the ports connecting to the faulty channel or component. If multiple faults exist, the system can continue normal operation as long as there is at least one path between every pair of endpoints. This network and its design issues are described further in [5] [2] [3].

In this network, the first sign of faults would come from failed message checksums or network protocol violations [6]. If these errors persisted, monitoring software would formulate a theory about possible faults in the network. However, if a checksum comes

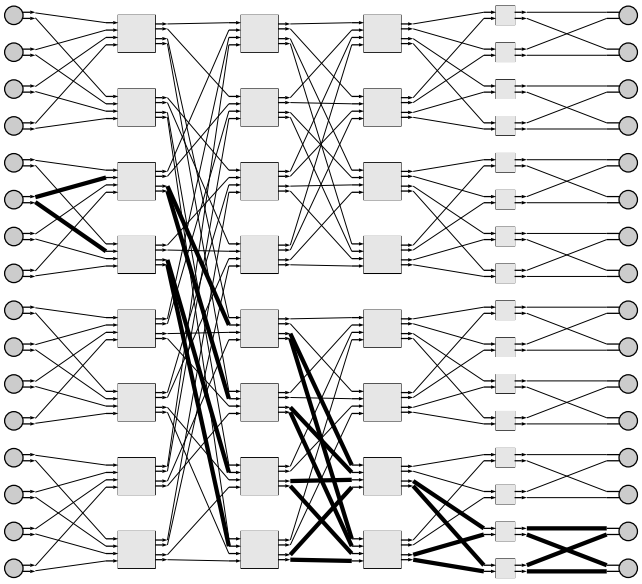


Figure 6: Fault-Tolerant Multipath Network

back corrupted, it is often unclear where it is being corrupted. Any of the wires or components associated with a connection through the network could be at fault for the bad checksum. Since there is a pair of outputs in each logical output direction, one output of each pair may be disabled at any time for testing or fault avoidance without sacrificing the functional correctness of the routing network. We can use the independent scan ability to check the integrity of each interconnection channel along the path suspected to be faulty. If this turns up a structural fault in the interconnection, the faulty channel can be left disabled and the fault noted. However, if this fails to turn up a possible source of corruption, each component in the path can be separately isolated from the network and tested. If the dilated routing components have redundant on-chip switching crossbars and the fault is determined to lie in a component's crossbar, the spare can be switched in to replace the faulty crossbar before returning the routing switch to active operation.

# 9 Conclusions

We have described some simple additions to the IEEE standard boundary-scan and test access port practices which result in a scan methodology appropriate for fault-tolerant systems. In addition to robust degradation of scan-paths in the presence of faults, these additions allow fault localization and system reconfiguration. Fault localization may proceed in parallel with normal operation in a minimally intrusive manner. We have further shown how the same basic mechanisms necessary for in-operation fault isolation can be used for fault avoidance and on-line physical repair. To show how these facilities come together in a representative system, we gave an example from our work with fault-tolerant networks.

## Acknowledgments

This material is based upon work supported under a National Science Foundation Graduate Fellowship. The research is also supported in part by the Defense Advanced Research Projects Agency under contract N00014-91-J-1698.

This work was developed in the MIT Transit Project under the supervision of Principal Research Scientist Thomas Knight, Jr. The work presented here has been refined based on discussions with members of the group, most notably Thomas Knight, Samuel Peretz, and Neil Brock.

## References

[1] T. Anderson, editor. *Resilient Computing Systems*, volume I, chapter 10, pages 178–196. Wiley-Interscience, 1985. Chapter by: C. I. Dimmer.

[2] Frederic Chong, Eran Egozy, and André DeHon. Fault Tolerance and Performance of Multipath Multistage Interconnection Networks. In Thomas F. Knight Jr. and John Savage, editors, *Advanced Research in VLSI and Parallel Systems 1992*, pages 227–242. MIT Press, March 1992.

[3] Frederic T. Chong and Thomas F. Knight, Jr. Design and Performance of Multipath MIN Architectures. In *Symposium on Parallel Architectures and Algorithms*, San Diego, California, June 1992. ACM. To appear.

[4] IEEE Standards Committee. *IEEE Standard Test Access Port and Boundary-Scan Architecture.* IEEE, 345 East 47th Street, New York, NY 10017-2394, July 1990. IEEE Std 1149.1-1990.

[5] André DeHon, Thomas F. Knight Jr., and Henry Minsky. Fault-Tolerant Design for Multistage Routing Networks. In *International Symposium on Shared Memory Multiprocessing.* Information Processing Society of Japan, April 1991.

[6] André DeHon, Thomas F. Knight Jr., and Henry Minsky. RNP: Fault Tolerant Routing Protocol. Transit Note 41, MIT Artificial Intelligence Laboratory, March 1991.

[7] C. Clark Jr. George and J. B. Cain. *Error-Correction Coding for Digital Communications.* Plenum Press, New York, 1982.

[8] P. Goel and M. T. McMahon. Electronic Chip-In-Place Test. In *Proceedings 1982 International Test Conference*, pages 83–90. IEEE, 1982.

[9] John P. Hayes. A Graph Model for Fault-Tolerant Computing Systems. *IEEE Transactions on Computers*, C-25(9):875–884, September 1976.

[10] Bernd Könemann, Joachim Mucha, and Günther Zwiehoff. Built-In Logic Block Observation Techniques. In *Proceedings 1979 International Test Conference*, pages 37–41. IEEE, 1979.

[11] Ron Lake. A Fast 20K Gate Array with On-Chip Test System. *VLSI Systems Design*, 7(6):46–55, June 1986.

[12] Johnny J. LeBlanc. LOCST: A Built-In Self-Test Technique. *IEEE Design and Test*, pages 45–52, November 1984.

[13] Colin M. Maunder and Rodham E. Tulloss, editors. *The Test Access Port and Boundary-Scan Architecture*, chapter 20, pages 205–213. IEEE, 1990. Chapter by: Patrick F. McHugh and Lee Whetsel.

[14] Henry Minsky, André DeHon, and Thomas F. Knight Jr. RN1: Low-Latency, Dilated, Crossbar Router. In *Hot Chips Symposium III*, 1991.

[15] W. Wesley Peterson and E.J. Weldon Jr. *Error-Correcting Codes.* MIT Press, Cambridge, MA, 1972.

[16] Thinking Machines Corporation, Cambridge, MA. *CM5 Technical Summary*, October 1991.

[17] Paul T. Wagner. Interconnect Testing with Boundary Scan. In *Proceedings 1987 International Test Conference*, pages 52–57. IEEE, 1987.

[18] Steve Webber. The Stratus Architecture. Stratus Technical Report TR-1, Status Computer, Inc., 55 Fairbanks Blvd., Marlboro, Massachusetts 01752, 1990.