Fast Fault Detection for a Multiple-plane Packet Switch

Roberto Rojas-Cessa,* *Student Member, IEEE,* Eiji Oki,** *Member, IEEE,* and
H. Jonathan Chao, *Fellow, IEEE*
Department of Electrical Engineering
Polytechnic University
6 Metrotech Center, Brooklyn, New York 11201, USA

Abstract—
In high-speed and high-throughput packet switches, system reliability is critical to avoid the loss of a huge amount of information and to avoid re-transmission of traffic. We propose a series of concurrent fault-detection mechanisms for a multiple-plane crossbar-based packet switch. Our switch model, called the $m + z$ model, has m active planes and z spare planes. This switch has distributed arbiters on each plane. The spare planes, used for substitution of faulty active ones, are also used in the fault detection mechanism, thus providing sufficient data redundancy for fault detection and location. Our detection scheme is able to detect a single fault in one time slot without increasing transmission overhead. The proposed schemes can be used for switches with different numbers of active planes and the number of spare planes needed for fault detection is small.

I. Introduction

In high-speed and high-capacity packet switches, system reliability is critical to avoid the loss of a huge amount of information and to avoid re-transmission of traffic. In a packet switch, shared resources such as the switch fabric and schedulers, must be fault-tolerant to avoid switch collapse due to a fault occurrence. As the demand of traffic with defined Quality of Service (QoS) guarantees increases, it is essential to count on fault-tolerant switches to avoid increased traffic due to re-transmissions that may affect the fragile flow control, congestion control, and scheduling mechanisms, which are already occupied with fault-free traffic flowing through, and may jeopardize the QoS parameters for the guaranteed traffic.

Fault-tolerant systems are mainly concerned with two issues: fault detection and resource redundancy [1]-[12]. One of the components in a switch that should be afforded a high degree of redundancy is the switch fabric since this is the most shared part of a switch.

This research is supported by NSF Grant 9906673.

*Corresponding author, E-mail:rrojas@kings.poly.edu, phone: (718)-260-3496, fax: (718)-260-3906.

**He is a Research Engineer with NTT Network Service Systems Laboratories, Tokyo, Japan. This work was done while he was a Visiting Scholar at Polytechnic University.

Crossbars are very popular because of their non-blocking nature, design simplicity, and market availability. The non-blocking property in a crossbar makes routing more practical; several crossbar-based switches have been proposed in research and commercially as in [13]-[16]. However, fault tolerance in crossbars presents a high complexity because of their large number of crosspoint elements, O(N^2). Redundancy and fault detection on all resources within the crossbars, such as switch elements (SEs) and connection links, become costly. It is important to provide a feasible fault-tolerant system for this popular switching fabric.

A duplex system (a replication of the system) can be considered as a straight-forward solution for a fault tolerant switch. It offers a 100% hardware redundancy. Duplex redundancy is cost-effective for small switches only (i.e., those with a small number of ports, connection cards, or links).

A multiple-plane switch architecture is appealing because of the augmented throughput and relaxed port speed and pin count. Redundancy can be provided to a switch with multiple switching planes. A switch that transmits a data packet through m active planes can have one or more spare planes to replace any faulty active plane. [17] presents a practical spare distribution scheme with less than 100% hardware redundancy with a system of $m + y + z$, where m is the number of active planes. A switch plane can comprise a bi-dimensional array of modules (e.g., boards). y is the number of spare modules in the y-axis. A y module can substitute any module in the corresponding active plane in the y-axis. z is the number of planes (or modules) in the z-axis. This methodology has been shown to be more cost-effective than a duplex system for a large switch since it can provide comparable availability of spares with less hardware redundancy. This sparing scheme allows a fast (i.e., concurrent) system reconfiguration.

In our switch model, we adopt a similar architecture with spares in the z-axis only.

A complete fault detection and location system comprises roughly three parts: a *detection scheme* on the physical layer, a *statistical information pool* for fault occurrences, and a *fault-tolerance manager* that determines when a switch part is considered faulty and when the replacement or recovery of the faulty part is performed. The detection scheme provides information to the manager when a discrepancy is found. The manager performs the determination of a fault according to the collected statistical information.

Fault detection can be performed in a concurrent fashion or during intermittent test-mode periods (sequential fashion) in an on-line system.

In sequential fashion, testing is performed while the switch is intermittently set in a testing mode to run suitable detection procedures. During the testing phase, the regular forwarding of cells is inhibited and the fault models are tested on the switch resources such as arbiters, switch elements, and transmission paths. For a large system, this methodology is impractical for high-speed switches because the testing takes an amount of time proportional to the number of inputs and outputs.

In the concurrent fashion, testing is performed while a cell is traversing the switch. The switching parts (the switch elements and paths) are tested as they are utilized. The detection has to keep up with the switching speed. Some extra hardware can be used for this purpose as in [1]. That paper presents a concurrent detection mechanism for a multiple-plane banyan-based architecture where the scheduler is separated from the switching network. In that scheme, an active plane is tested at a time, and the spare plane is used to transmit a copy of the transmitting cells. If these two plane outputs present differences, both active and spare planes are suspected to be faulty. After a plane is suspected of failure, the switch is placed in a testing mode for fault location. During the testing mode, special test patterns are applied to the active planes under test with the same routing information that triggered the change from the working mode. However, this method needs a second comparison phase (in a second mode) in the following cell slot to define the fault. This makes the detection system slow, and switching from one mode to the other is difficult for a large N switch, where N is the number of input and output ports, because all ports need to be synchronized.

In a high-speed fault-tolerant switch, it is necessary to provide a fast fault detection scheme that, for a high complexity fabric such as a crossbar, is able to provide high fault detection coverage. It is also important to simultaneously provide fault detection coverage for the spare planes, and that the transmission overhead ratio is minimally increased by the detection mechanism.

In this paper, we propose a series of fast concurrent fault-detection schemes for a multiple-plane crossbar-based packet switch.

We assume a $m + z$ multiple-plane architecture, where m planes are active and z planes are used as spares. These detection schemes detect single fault models and locate the fault at the plane level for active and spare planes in a single cell cycle (i.e., a fixed-length time is used to transmit a fixed-length packet from an input to its destined output port). Also, with these schemes, the cell overhead is not increased, and the checking complexity at the outputs is reduced to single bit comparisons. These schemes are used within a fault analysis system for fault detection and faulty-plane location. The active planes and the spare plane are continuously tested so that the substitution of a faulty plane can be performed quickly. We explain our detection mechanisms with the example of a system with a single spare plane.

This paper is organized as follows. We present a description of our switch model in Section II. In Section III, we define the fault models considered in this architecture. In Section IV, we present our schemes for fault detection. In Section V, the detection coverage and utilization are estimated. In Section VI, we comment about extending these schemes for any number of active planes. In Section VII, we present our conclusions.

II. Multiple-plane Switch Model

As Figure 1 shows, a multiple-plane architecture uses more than one plane to transfer packets from the inputs to their destined outputs. We assume a switch that uses a multiple-plane technique, using m active planes. Variable length packets are divided into fixed-length cells. Furthermore, a cell, with a length of L bits, is segmented into a number of segments equal to the number of planes m; all segments follow the same path between inputs and outputs, producing path redundancy. All segments are transferred simultaneously.

We assume a crossbar as a switching fabric, where each port is w-bit wide. In a crossbar-based switch, a scheduler is in charge of selecting the set of cells that can be sent through the crossbar to resolve input and output port contention. The scheduler can be of a distributed nature as in [18] or of a centralized nature as in [14], [15], and [19]. In our switch model, we assume a distributed scheduler architecture, the input arbiters are considered in the input line cards, and the output arbiters are implemented

in the crossbar fabric as in [18]. Using independent arbitration planes, every plane has a copy of the output arbiter that provides redundant arbitration that is well-utilized for fault-tolerant measures.

In the crossbar logic, every crosspoint element is in either active or idle state (also referred as cross or toggle state, respectively). A crosspoint element can be transmitting data to the output port or disconnecting the data transmission, in the active state or in the idle state, respectively. The bit values in the active state depend on the user data. All inputs are interconnected to an output by an *input joint* logic. We assume that any circuitry working as an input joint can be modeled as an OR gate. Figure 2 shows the input joint logic. In the rest of the paper, we assume that the idle value for a crosspoint is "0."

III. Fault Models

The fault models in [1] and [12] are adapted to our architecture, others are the same models as considered for digital logic switching systems in [20], and the rest are defined according to our switch model.

We separated the fault models arbiter faults and crosspoint faults. We assume that multiple faults have a very low probability of occurrence, so we do not consider them.

Some examples of the arbiter faults are presented below.

Grant Multiplicity. If two or more inputs are matched with a single output, multiple grants are issued. This misbehavior might bring up a collision between two cells.

Grant Loss. If the scheduler receives at least a request, a grant should be issued. If the scheduler does not issue a grant, the grant is considered lost.

Grant Stuck. This fault is interpreted as a situation when a port or a group of ports are receiving a grant consecutively (back to back) while other ports remain starved.

We present some examples of crosspoint fault models below.

Cross or Idle (unconnected) Stuck Crosspoint (CS). This fault occurs in a crosspoint when it remains permanently in cross-stuck regardless of the arbiter result.

Toggle or Active (connected) Stuck Crosspoint. Similarly to CS, Toggle Stuck (TS) occurs when a crosspoint remains in toggle state invariably along the time, even in the presence of a declined grant signal.

1/0 Stuck Crosspoint (OZS). This fault occurs in the data links when an output link remains in either logic zero or logic one when data pass through it.

In general, we assume that any fault may produce a discrepancy among the redundant information. A discrepancy exists whenever a difference on any of the w bits is detected.

IV. Schemes for Fault Detection and Location

In this section, we present three schemes for discrepancy detection and plane-location. The determination and location of a fault is pursued up to the plane level. Once the faulty plane is located, it is replaced with a spare non-faulty plane by the control system.

These schemes are based on comparison of redundant user data that is transmitted through the spare plane. The spare plane transmits a copy of data from the active planes at each time slot so that the spare plane is also tested. The comparison is done by pairs of planes on a bit-cycle basis, so with the comparison of three planes separately and concurrently, fault detection can be performed effectively. As the arbiter results are redundant for each plane, the same input and output ports are matched on all planes. As a result, all named faults are detected by this method.

The detection process is summarized as follows: *Phase 1.* A copy of data from the active planes is sent on the spare plane, according to the procedure stated below. *Phase 2.* After data have been transmitted, the data at the output ports of the active planes are compared to the redundant data on the spare plane. Either of the following cases occurs:
(i) If all data from these $m + z$ planes have no discrepancies, the planes are free of faulty parts.
(ii) If a discrepancy is detected, the plane with a discrepancy is located according to the discrepancy type.[1] The scheme details and how data redundancy is done are presented below in an example. For simplicity, we consider a switch where there are four active planes and a single spare plane (i.e., $m = 4$ and $z = 1$), without losing generality. We also assume that each data bus is four bits wide (i.e., $w = 4$) and the cell length is 512 bits ($L = 64 \times 8$).

Data on Active Planes. A bit transmitted through each plane is denoted as $\pi^l_{t_b}$, where the bit set π corresponds to an active switching plane, $\pi = \{W, X, Y, Z\}$, l is one of the data lines, $l = \{A, B, C, D\}$, and t_b is the bit-clock cycle when the pair (I, J), where I is an input port and J is an output port in the crossbar, is set up for cell transmission, numbered as $t_b = \{0, ..., \frac{L}{w \times m} - 1, \frac{L}{w \times m}, ...\}$.

In our example, the number of bit clocks in a cell slot is $t_b = \{0, 1, 2, ..., 31\}$. However, t_b continues in the next time slot where the same pair input-output port are selected for transmission. This procedure provides the ability to accumulate as many comparison bits as needed.

[1] A discrepancy means that a plane shows a difference. Note that when S differs, two or more differences are seen.

Data on the Spare plane. The bit sets sent through the spare plane (S) and the single bits are denoted in the same way as those for the active planes. Each of the bits transmitted through S is denoted by $S^l_{t_b}$, where l and t_b have the same meaning as above.

In the following subsections, three schemes are presented.

A. Scheme 1: Redundant Bits on the Spare Plane

The transmitted bit sets on the spare plane, as a copy of bit sets on the active planes, are chosen as follows:

At time $t_b = 0$ the data lines S^l_0 for $l = \{A^S, B^S, C^S, D^S\}$ (where S stands for the spare plane) transmit a copy of the data in the active plane W of this bit time, W^l_0. This set is denoted as $S(W^l_0)$.

At time $t_b = 1$ the data lines S^l_1 transmit a copy of the data on active plane X, X^l_1. This set is denoted as $S(X^l_1)$; and so on.

In general, at time t_b, the data lines in the set $S^l_{t_b} = S(\pi^l_{t_b})$ transmit a copy of the plane $t_b \bmod m$ ($t_b \bmod 4$), where the remainder $\{0, 1, 2, 3\}$ corresponds to $\pi = \{W, X, Y, Z\}$, respectively.

During a cell slot time, the transmitted bit sets on the S plane are: $S(W^l_0)$, $S(X^l_1)$, ..., $S(Z^l_3)$, $S(W^l_4)$, ..., $S(Z^l_7)$, ..., $S(Z^l_{31})$. In this case, the first element ($S(W^l_0)$) is sent in the first bit cycle $t_b = 0$ and the set ($S(Z^l_{31})$) is sent on the 32^{nd} bit cycle ($t_b = 31$). This data distribution is shown in Figure 3. In the next time slot, with the same input-output pair, the t_b count continues. Figure 4 shows that the information, passed through the S plane, is compared to the corresponding bits on the active planes ($W, X, Y,$ and Z) during a time slot. The results are diagnosed according to this method. A discrepancy can be detected in either the spare plane S or in any of the active π planes. We define that a bit set $P1$, of plane $P1$, is different from the bit set $P2$, of plane $P2$, if any bit of set $P1$ differs from its corresponding bit on plane $P2$.

B. Scheme 2: Addition of Bit Complements

According to the input joint logic, Scheme 1 has a dependence on the distribution of the values of "0" (0-bit ratio) and "1" for each comparison bit. It can be seen that the number of logical "0"s in the comparison bits increases the efficiency of the mechanism. The number of "0"s depends on the user data, so Scheme 1 has unpredictable coverage for an unpredictable 0-bit ratio.

To overcome the problem stated above, Scheme 2 presents a variation on the bit distribution in Scheme 1 among the active and spare planes to balance the number of "0"s and "1"s (or bit-ratio) in the comparison bits, making the detection scheme more independent of the random user data.[2]

In Scheme 2, the active planes transmit some extra bit sets: an original bit set and its complement for every comparison bit set. The spare plane transmits a copy of the original bit set and the bit sets that could not be transmitted through the active planes because of the transmission of the complements bit sets.

In our example, the distribution in the active plane W is: W_0, $\overline{W_0}$, W_2,..., W_8, $\overline{W_8}$,..., W_{31}, in one cell slot. For the active plane X, the bit sets are: X_0, X_1, X_2, $\overline{X_2}$,..., X_{10}, $\overline{X_{10}}$,..., X_{31}, and in that way for the rest of the active planes. The spare plane S transmits $S(W_0)$, $S(W_1)$, ..., $S(X_4)$, $S(X_5)$, ..., $S(Y_8)$, ..., $S(Z_{12})$, $S(Z_{13})$, ..., $S(X_{16})$, ..., $S(Z_{31})$.

The pictorial description of the bit format is shown in Figure 5. In this figure, we represent a bit set as any W, X, Y, Z, as above. Each letter corresponds to an active plane (i.e., $0, 1, 2, 3$) and S, the spare plane. Each set, as before, encloses four bits from lines $A, B, C,$ and D.

C. Scheme 3: Embedding Small Hardware for Collision Monitoring

Both of the schemes presented above use bit redundancy through the active and the spare planes to detect a fault. Even though Scheme 2 is more independent of the user data, it presents a disadvantage: the number of comparison bits, collected during a given period of time, is reduced when compared to Scheme 1 because of the transmission of the complemented bits. The time to collect a bit set in Scheme 2 is double the time that Scheme 1 takes.

Scheme 3 uses the properties of Scheme 1 and Scheme 2. It produces a discrepancy coverage independent of the user data and a higher coverage. Scheme 3 uses the data distribution as in Scheme 1 and the *input joint* is replaced by an *exclusive OR* function to modify the input joint output when two "1"s collide. This small modification balances the fault collision sensitivity for the 0-bit ratio, without affecting the behavior of the input joint.

V. Coverage of Discrepancy Detection

The detection coverage depends on the deployed scheme. In this section, we evaluate the coverage to detect a difference with all presented schemes.

The redundant data or bit sets are concatenated to form a bit string. These bits are called *bits in comparison* or *testing bits*. The testing bits transmitted through a legitimate (or fault-free) input are denoted as string B_l, and the ones

[2]The scheme does not modify the information of user cells.

transmitted through a faulty input are denoted as string B_c or colliding string.

We estimate the fault detection coverage by analyzing the probabilities of a single bit being "1". We define $P_l(1)$ as the probability of a legitimate bit as being of value "1"; for a value of zero the probability is $P_l(0)$. Since we study the coverage when a bit collision occurs, we define $P_c(1)$ as the probability of a colliding bit (or erroneous bit) of having value "1". For simplicity, we assume that:

$$P_l(b) = P_c(b), \tag{1}$$

where b is either "0" or "1", as shown above, without losing generality. So the following coverage is a function of the probability product between the legitimate and the colliding bits. In the results presented in the coverage figures, we refer to a bit probability as:

$$p(b) = P_l(b) = P_c(b). \tag{2}$$

However, we describe the origin of each bit probability in the following equations. The coverage Cov of a detection scheme is estimated as:

$$Cov = \frac{D}{D+U}, \tag{3}$$

where D is the average probability of having a detectable combination in n testing bits and U the average probability for an undetectable combination.

A. Coverage for Scheme 1

The coverage, using (3), of Scheme 1 is defined by U_{s1} and D_{s1}, which are the average probability of the occurrence for a detectable and an undetectable combination for Scheme 1, respectively. U_{s1} and D_{s1} can be represented as:

$$U_{s1} = \sum_{k=0}^{n} C(n,k) P_l(0)^{n-k} P_l(1)^k \alpha, \tag{4}$$

$$\alpha = \sum_{j=0}^{k} C(k,j) P_c(0)^{n-j} P_c(1)^j \tag{5}$$

and

$$D_{s1} = \sum_{k=0}^{n} C(n,k) P_l(0)^{n-k} P_l(1)^k (\beta - \delta), \tag{6}$$

where

$$\beta = \sum_{j=0}^{n} C(n,j) P_c(0)^{n-j} P_c(1)^j \tag{7}$$

and

$$\delta = \sum_{i=0}^{k} C(k,i) P(0)^{n-i} P_c(1)^i. \tag{8}$$

j and i are variables that depend on a specific combination; they are used to indicate the number of "1"s in a combination. k is the number of "1"s in the combination of the testing field of n bits. $C(r,s)$ is the number of combinations with r "1"s in a combination with s bits. For this scheme, n is defined as:

$$n = \lfloor \frac{zt_b}{m} \rfloor w, \tag{9}$$

where t_b is the number of bit cycles that the pair (I, J) are connected during one or more time slots. (9) provides the number of testing bits by bit cycle. It takes m bit cycles to collect the first comparison bit set or n first bits. So t_b increases by steps of m bit cycles. Figure 6 shows the results obtained from (4) and (6) in function of n (here we assume that n can be collected arbitrarily for demonstration purposes). This scheme performs best when the distribution of "0" and "1" values is uniform with an average of $p(1) = p(0) = \frac{1}{2}$, and it degrades for any other value of $p(b)$.

B. Coverage for Scheme 2

The coverage for Scheme 2, according to (3), in function of the average probability of the occurrence for a detectable and an undetectable combination U_{s2} and D_{s2}, respectively. They are represented as:

$$U_{s2} = P_c(0)^n \tag{10}$$

and

$$D_{s2} = 1 - P_c(0)^n, \tag{11}$$

where n, for this scheme, is defined as:

$$n = \lfloor \frac{zt_b}{2m} \rfloor w. \tag{12}$$

Figure 7 shows the improved detection coverage for Scheme 2 with different bit probabilities.

The coverage in Scheme 2 depends on the product of $p(1)$; as $p(1)$ goes from 0 to 1, the coverage increases. For a $p(1) = \frac{1}{2}$, the coverage of this scheme (50%) doubles the coverage of Scheme 1. The number of logical bits in Scheme 2 is half of the bits used for comparison in Scheme 1 for the same period of time.

C. Coverage for Scheme 3

The coverage for Scheme 3 is estimated by using the following equations. Similarly from (3), where, in this case, the equations for U_{s3} and D_{s3} are the same as for Scheme 2:

$$U_{s3} = P_c(0)^n \tag{13}$$

and

$$D_{s3} = 1 - P_c(0)^n. \quad (14)$$

In these equations, n is defined by (9). The time in what n can be collected in this scheme is the same as in Scheme 1. The detection coverage, in terms of n, for Scheme 2, is the same for Scheme 3, as shown in Figure 7.

D. Comparison of Schemes

Figure 8 shows a comparison among the three presented schemes. In this figure, S1, S2, and S3 stand for Scheme 1, Scheme 2, and Scheme 3, respectively. Schemes 2 and 3 have a higher coverage than Scheme 1 in terms of the number of testing bits.

Figure 9 shows the detection coverage for all three schemes as a function of the number of bit cycles. As shown in this figure, the most effective scheme is Scheme 3. 12 bits are sufficient (three bit cycles, 4×3) to achieve an acceptable level of discrepancy detection, or 99.9%, which can be achieved in a single cell slot. If the cell length is increased, more bits for comparison can be used if the detection is made by time slot. Otherwise, the accumulation of bits continues at the next time slots in which the same pair (I, J) are used, providing a large number of testing bits.

Schemes 1 and 3 have almost the same hardware complexity. The number of redundant bits is the same. The only difference is the *exclusive OR* function as the input joint in Scheme 3; however, this is a negligible part within the total hardware complexity in a switch fabric. Scheme 2 has a higher complexity than the other two schemes since all planes transmit not-redundant user data (the spare plane becomes an active plane). The accommodation of bits is more complex and so is the sorting of bits at the outputs. Also, the number of bits that can be used for comparison is half of that used in Schemes 1 and 3 during the same period of time. Scheme 3 provides the fastest detection.

E. Rotation of Planes

We also observe the utilization of planes in these schemes. We define the *utilization ratio* of a plane for detection as the number of bits that are compared for discrepancy detection over the total number of transmitted bits during a cell cycle. The higher the utilization ratio, the faster the testing bits (or n) are collected or the faster the detection. In our example, Scheme 1 uses 25% of each active plane to send detection data while 100% of the data in the spare plane is used for fault detection. This gives better coverage to the spare plane. In Scheme 2, the utilization of active planes is 12.5%, while in the spare plane it is 50%, since the other 50% is not-redundant user data. Scheme 3 has the same utilization as Scheme 1. This unbalanced utilization provides a better coverage (for the same period of time) for the spare plane. In order to balance the utilization and to improve the detection time for the active planes, rotation of planes (of the role of the spare plane) can be performed. The average utilization and coverage are increased by rotating the assignment of the role of the spare plane in a cyclic manner among all planes. In this case, the utilization for Schemes 1 and 3 is 40% for any plane, and for Scheme 2, 20% for any plane. Rotation of planes modifies n as follows. For Schemes 1 and 3:

$$n = \lfloor \frac{2zt_b}{m+z} \rfloor w, \quad (15)$$

for Scheme 2:

$$n = \lfloor \frac{zt_b}{m+z} \rfloor w. \quad (16)$$

In (15) and (16), the ratio $\frac{2z}{m+z}$ is the average utilization of a plane that takes the role as a spare plane during z time units and as an active plane during m time units. Figure 10 shows the coverage in function of the bit cycles t_b when using (15) and (16). Rotation of planes can be used to reduce the time to achieve an acceptable detection coverage (e.g., 99.9%) or to improve the coverage in a limited time period. In this figure, our example shows an improvement to Scheme 1. Scheme 2 does not show an improvement because of the high coverage of the scheme without rotation. Scheme 3 also shows an improvement when rotation is used.

VI. Spare Resources

With a larger number of spare planes, the larger the detection coverage; however, the cost increases proportionally. Our objective is to keep the cost low.

The detection coverage with a larger z can be estimated according to (9) and (12), without rotation of planes. In our example, if the number of spare planes increases to two (i.e., an $m + 2$ switch), the number of comparison bits or utilization increases in proportion to the number of spare planes z, so the coverage is higher (and so is the cost). Figure 11 shows the detection coverage as a function of the number of spare planes. It is seen that for our example, the coverage does not improve significantly when $z = 2$. These detection mechanisms produce an acceptable detection coverage with a small number of hardware resources.

For a switch with a large m, the spare planes can be distributed in the z-axis (e.g., with a granularity at the board level). The number of spare planes needed for an acceptable detection coverage can be obtained from (9) and (12), given n, m, and w.

In [17], it has been shown that the replacement of a faulty active plane can be done in a single cell cycle. The replacement process can be implemented in our switch model to provide a fast recovery.

VII. Conclusions

We presented a series of concurrent fault detection mechanisms, which use spare resources as an important part in the fault detection process. The multiple-plane architecture uses at least one spare plane for fault detection while all active and spare planes are tested on-line. While using a crossbar switching fabric with a high number of SEs (N^2), the paths and SEs tested are the ones that are being used for transmission. These schemes do not increase the cell overhead. Of all three schemes presented, Scheme 3 has the best coverage. Although Scheme 2 presents a lower coverage than Scheme 3, it can be considered a solution for the case when the input joint or the crossbar cannot be modified. We provide very fast and simple fault detection mechanisms that can be well-utilized for very high-speed switches. We also showed, by means of an example, that the rotation of the role of the spare plane improves the detection coverage or speeds it up. The proposed detection schemes can be used with the transmission check mechanisms to distinguish a faulty plane from a faulty transmission, and they can be adapted to a switch with any number of active planes providing a number of spare planes accordingly. We also showed that a small number of planes are enough to provide an acceptable detection coverage, with a lower cost than a duplex system.

References

[1] Y-H. Choi and P-G. Lee, "Concurrent Error Detection and Fault Location in a Fast ATM Switch," *IEEE/Proceedings of ATS'96*, pp. 113-118, 1996.

[2] S-C. Yang and J. A. Silvester,"A Fault Tolerant Reconfigurable ATM Switch Fabric," *IEEE INFOCOM '91*,vol. 3, pp. 1237-1244, 1991.

[3] V. P. Kumar, J. G. Kneuer, D. Pal, and B. Brunner, "PHOENIX: A Building Block for Fault Tolerant Broadband Packet Switches," *Globecom '91*, pp. 0228-0233, 1991.

[4] Y-H. Choi, "Concurrent Error Detection in Priority Queue Managers for ATM Networks," *IEEE Pacific Rim International Symposium Proceedings*, pp. 59-64, 1997.

[5] B. Iyer, R. Karri, and I. Koren, "Phantom Redundancy: A High-Level Synthesis Approach for Manufacturability," *ICCAD-95, IEEE/ACM Intl Conference '95*, pp. 658-661, 1995.

[6] A. Itoh, "A Fault-Tolerant Switching Network for B-ISDN," *IEEE JSAC*, vol. 9, pp. 1218-1226, October 1991.

[7] A. K. Somani, T. Zhang, *IEEE Transactions on Reliability*, vol 47, pp. 19-29, March 1998.

[8] P. U. Tagle, N. K. Sharma, "Performance of Fault Tolerant ATM Switches," *IEEE Proc. Commun*, vol. 143, No. 5, pp. 317-324, October 1996.

[9] A. Varma and S. Chalasani, "Fault-tolerance Analysis of One-Sided Crosspoint Switching Netowrks," *IEEE Transactions on Computers*, vol. 41, pp. 143-158, February 1992.

[10] J. Ghosh and N. Krishnamurthy, *IEEE Proc. 6th Intl Conference on VLSI Design*, pp. 351-356, January 1993.

[11] T-H. Lee and J-J. Chou, "Fault Tolerance of Banyan Using Multiple Pass," INFOCOM '92, pp. 0867-0875, 1992.

[12] F. Lombardi, C. Feng, and W.-K. Huang, "Detection and Location of Multiple Faults in Baseline Interconnection Networks," *IEEE Transactions on Computers*, vol. 41, No. 10, pp. 1340-1344, October 1992.

[13] H. J. Chao and J-S. Park, "Centralized Contention Resolution Schemes for a Large-capacity Optical ATM Switch," *Proc. IEEE ATM Workshop*, Fairfax, VA, May 1998.

[14] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, "Tiny-Tera: A Packet Switch Core," *IEEE Micro*, pp. 26-33, January-February 1997.

[15] P. Krishna, N. S. Patel, A. Charny, and R. Simcoe, "On the Speedup Required for Work-Conserving Crossbar Switches," *IEEE Selected Areas in Communications*, vol. 27, no. 6, pp. 1052-1066, June 1999.

[16] Cisco Systems, 12016 Multigigabit Switch Router, http://www.cisco.com/univercd/cc/td/doc/product /core/cis12016/hfricg/hfricgpo.html, 1999.

[17] K. Padmanabhan, "An Efficient Architecture for Fault-Tolerant ATM Switches," *IEEE/ACM Transactions on Networking*, p527-537, 1995.

[18] H. J. Chao, "Saturn: A Terabit Packet Switch Using Dual Round-Robin," *IEEE Comm. Mag.*, vol. 38, no. 12, pp. 78-84, December 2000.

[19] N. McKeown, V. Anatharam, and J. Walrand, "Achieving 100% Throuhgput in an Input-queued Switch," *IEEE INFOCOM, 1996*, pp. 296-302, 1996.

[20] M. Abramovici, M. A. Breuer, and A. D. Friedman, "Digital Systems Testing and Testable Design," IEEE Press, 1990.

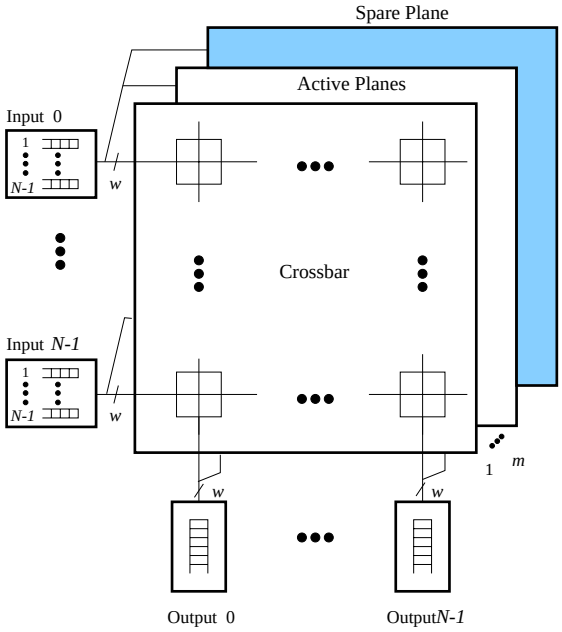

Fig. 1. Multiple-plane switch model with a spare plane

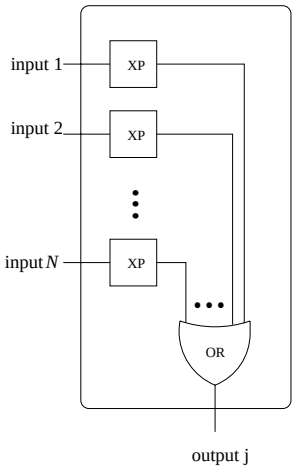

Fig. 2. Input joint: logic of a connection of all inputs to one output

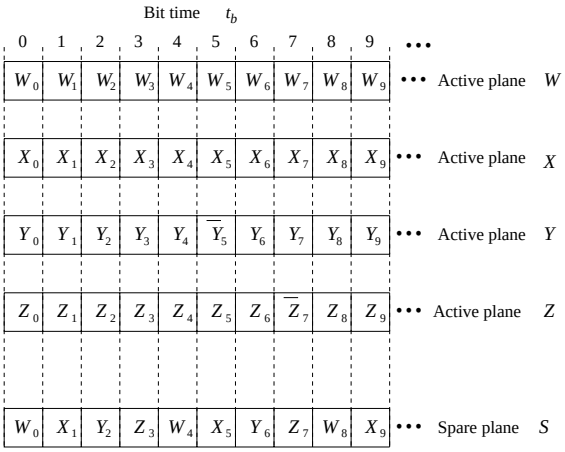

Fig. 3. Scheme 1: bit distribution on active and spare planes

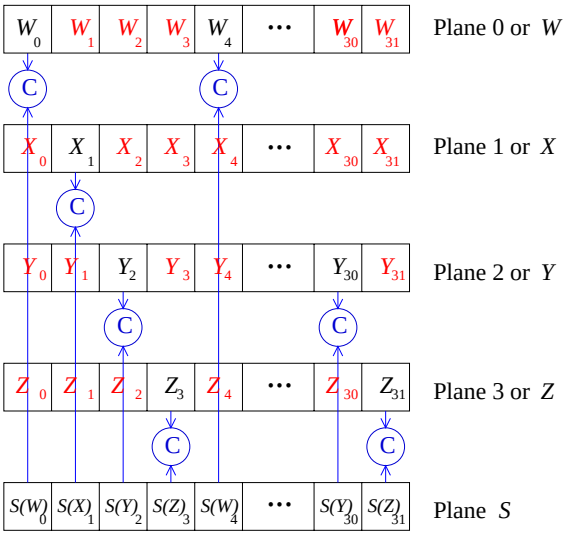

Fig. 4. Comparison of bit sets in Scheme 1

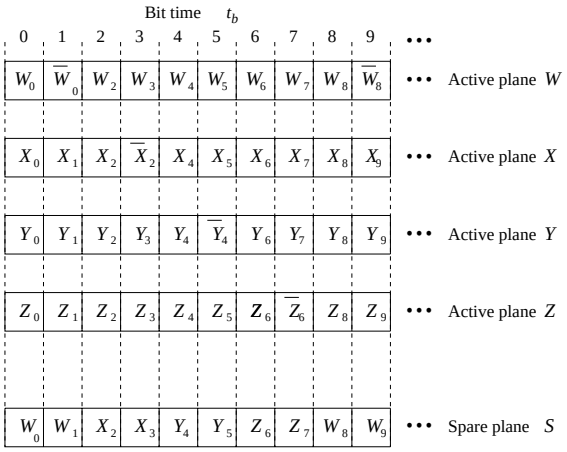

Fig. 5. Scheme 2: bit distribution on planes with bit complements

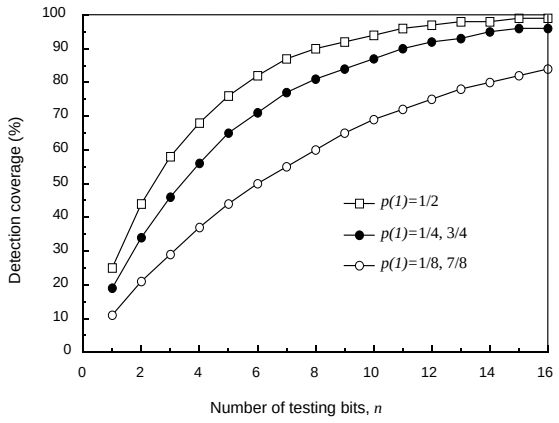

Fig. 6. Coverage for Scheme 1 with different numbers of testing bits and bit probabilities

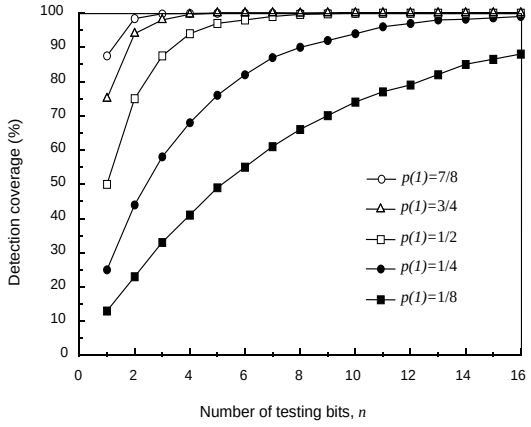

Fig. 7. Coverage of Schemes 2 and 3 with different numbers of testing bits and bit probabilities

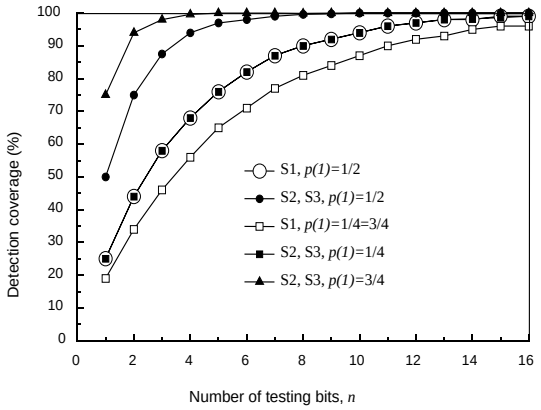

Fig. 8. Comparison of schemes 1, 2, and 3

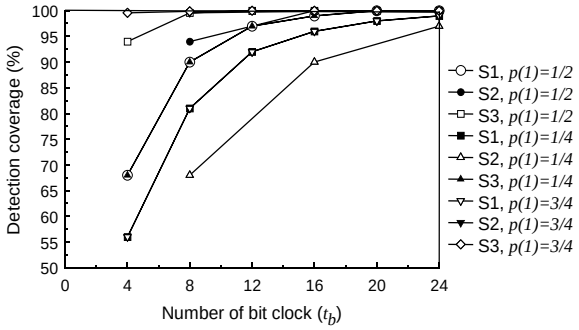

Fig. 9. Coverage in terms of number of bit cycles

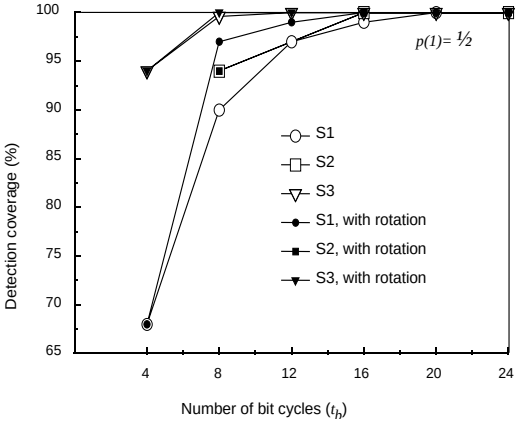

Fig. 10. Coverage considering rotation of the spare plane

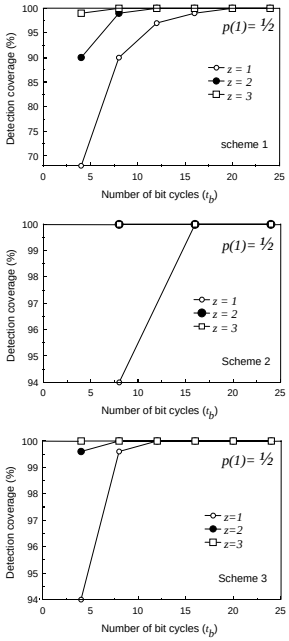

Fig. 11. Coverage in function of z spare planes