Concurrent Fault Detection for a Multiple-plane Packet Switch

Roberto Rojas-Cessa, *Member, IEEE*, Eiji Oki, *Member, IEEE*, and

H. Jonathan Chao, *Fellow, IEEE*

***Abstract*— In high-speed and high-capacity packet switches, system reliability is critical to avoid loss of huge amounts of information and 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 fault detection and fault location for all switching planes. Our detection schemes are able to quickly detect a single fault without increasing transmission overhead. The proposed schemes can be used for switches with different numbers of active planes and a small number of spare planes.**

I. Introduction

In high-speed and high-capacity packet switches, system reliability is critical to avoid loss of huge amounts of information and re-transmission of traffic (or triggering some other means of data recovery that consumes switch resources and time). In a packet switch, the shared resources such as the switch fabric and arbiters, 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 traffic from re-transmissions. Re-transmission 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.

One of the components in a switch that should be afforded a high concern for redundancy is the switch fabric since this is the most shared part of a switch. Multiple stage switch fabrics, such as expanded banyan networks, provide redundant connection paths in the fabric [1], [2]. However, because banyan networks are blocking, scheduling of packets to enter the fabric is complex. Crossbars are very popular switch fabrics in packet switches and routers because of their non-blocking nature, design simplicity, market availability, and maturity in their scheduling processes. Several crossbar-based switches have been proposed in research and commercial switches [3]-[15]. A multiple-plane switch architecture is appealing because of the augmented switch capacity and port speed [9], [10]. In packet switches, it is a common practice to divide variable-length packets into fixed-length packets, called cells, for transmission in the switch fabric and to re-assemble the packets before they leave the switch. The time to transmit a cell from an input to an output of a switch fabric is called a cell slot.

Fault tolerance in crossbars presents a high complexity because a large number of crosspoint or switch elements (SEs), of order O(N^2), needs to be covered. Redundancy and fault detection on all resources within a crossbar, such as SEs and connection links, become costly. It is necessary to provide a feasible fault-tolerant system for this popular switch fabric.

Fault-tolerant systems are mainly concerned with component fault detection, component failure probability, and resource redundancy [16], [17], [18]-[26]. To make a switch fault-tolerant, the failure probabilities associated with its various components must be designed to be low. Once a fault occurs, the switch should recover rapidly by using redundant resources.

A duplex switch (i.e., a switch with full redundancy of its components) is a straight-forward solution as a fault-tolerant switch. A duplex switch offers high hardware availability and it may be able to recover up to 100% of its original performance. Moreover, managing duplex switches is simple at the expense of redundant hardware resources. However, a duplex switch can be costly when the switch fabric is a large percentage of the overall switch, since the redundant hardware resources impact the total cost. In this case, we should consider reducing the redundant hardware resources by having an intelligent switch controller while keeping the switch fault-tolerant.

Redundancy can be easily provided to a switch with multiple switching planes. [27] showed that it is possible to achieve a comparable availability in a duplex system of a multiple-plane switch with less than 100% hardware redundancy for a large switch size. Therefore, it is cost-effective to consider a multiple-plane switch with the number of spare planes smaller than the number of active planes.

A fault-detection and location system roughly comprises three parts: a *detection scheme*, a *statistical database* to

This work was supported in part by the New York State Center for Advanced Technology in Telecommunications (CATT), and in part by the National Science Foundation (NSF) under grants 9814856 and 9906673.

Roberto Rojas-Cessa is with the the Department of Electrical and Computer Engineering at New Jersey Institute of Technology, University Heights, Newark NJ 07102 USA Email: rojasces@njit.edu

Eiji Oki is with NTT Network Innovation Laboratories, 3-9-11 Midori-cho Musashino-shi, Tokyo 180-8585 Japan. Tel: +81-422-59-3441, Fax: +81-422-59-6387. Email: oki.eiji@lab.ntt.co.jp. This work was done while he was a Visiting Scholar at Polytechnic University.

Jonathan Chao is with the Department of Electrical and Computer Engineering at Polytechnic University, Brooklyn, NY 11201 Email: chao@poly.edu

register fault occurrences, and a *fault-tolerance manager* that determines when a switch part is considered faulty and the time when the replacement or recovery of the faulty part is performed. The detection scheme allows testing data fields for correct data and provides the testing results to the fault tolerance manager. The manager determines the fault according to collected statistical information.

Fault detection can be performed in a sequential or concurrent fashion. In sequential fashion, testing is performed while the switch is intermittently set into a testing mode to run suitable detection procedures. During the testing mode, the regular forwarding of cells is inhibited and the fault models are tested on the switch resources such as arbiters, SEs, and transmission paths. For a large system, this sequential detection is impractical for high-speed and high-capacity switches because the testing may take an amount of time proportional to the number of inputs and outputs. In concurrent detection, testing is performed while the switch is normally switching data traffic so the switch parts (the switch elements and paths) are tested as they are used. The difficulty with concurrent detection is to keep up with the switching speed, determined by the duration of a cell slot.

In [16], a detection scheme for a multiple-plane, banyan-based architecture where the arbiter plane is separate from the switching network was presented. In this scheme, an active plane is tested while the spare plane is used to transmit a copy of the transmitted cells. If these outputs from two planes present differences, both active and spare planes are suspected to be faulty. After a plane is suspected of failure, the switch is placed into 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, targeted for banyan-based fabrics, is not totally concurrent. It needs a second comparison phase (in a second mode) in the following cell slot to define which plane is faulty. This makes the detection system slow and switching from one mode to the other is difficult for a large N switch.

It is necessary to provide concurrent fault-detection schemes for high-speed and high-capacity switches that, for high complexity fabrics such as popular crossbars, are able to provide high fault-detection coverage. Moreover, it is required that the fault-detection schemes simultaneously provide fault-detection coverage for the spare planes so that a faulty plane can be substituted with a healthy plane.[1] It is also desirable that the transmission overhead is not increased by the detection scheme to avoid increasing the implementation complexity of a high-speed switch.

In this paper, we propose a series of concurrent fault-detection schemes for a multiple-plane crossbar-based packet switch. We called this multiple-plane switch the $m+z$ model, where m planes are active and z planes are spares. Our detection schemes quickly detect single faults and locate them at the plane level covering active and spare planes. We only considered single faults because multiple faults have a very low probability of occurrence.[2] Also, with these schemes, the cell overhead is not increased, and the checking complexity at the outputs is reduced to single bit comparisons. The active planes and the spare plane are tested so that the substitution of a faulty plane can be performed with a healthy plane.

This paper is organized as follows. In Section II, we describe our switch model. In Section III, we define the fault models considered in this architecture. In Section IV, we present the detection procedure. In Section V, we present our schemes for fault detection. In Section VI, we evaluate the performance for each proposed scheme by exploring their failure probabilities. In Section VII, we examine these schemes for different numbers of active planes. In Section VIII, we discuss the applicability of these schemes to other switch fabrics. In Section IX, we present our conclusions.

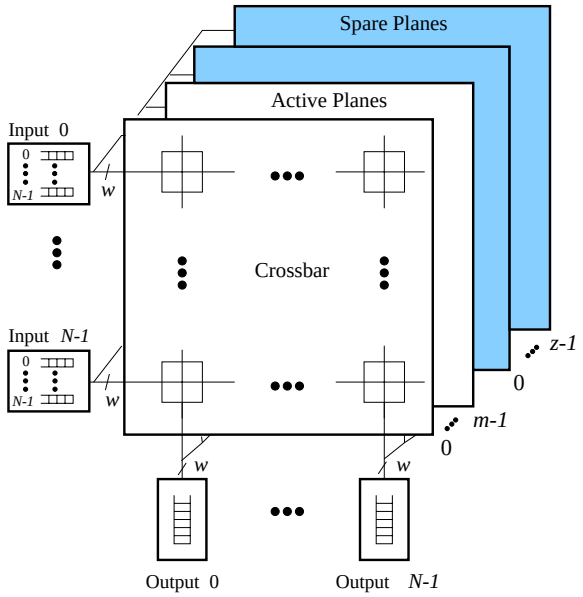

Fig. 1. Multiple-plane switch model with spare planes.

II. Multiple-plane Switch Model

A multiple-plane switch uses m planes to transfer cells from the inputs to their destined outputs, where each plane is a crossbar fabric, and has z spare planes that can be used to substitute active planes when these are determined faulty. Figure 1 shows an example of an $m+z$ switch, where the colored planes are the spare planes.

In an $N \times N$ crossbar, an SE connects input I, where $0 \leq I \leq N-1$, to output J, where $0 \leq J \leq N-1$, under request of an arbiter. In addition, 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, as shown in Figure 2.

A cell, with a length of L bits, is partitioned into a number of segments equal to the number of planes m at the input port. All segments of a particular cell follow the same path

[1]Note that instead of using spare planes, re-routing cells through the active healthy planes cannot be considered an alternative since this requires changing the duration of the cell slot.

[2]The failure probabilities associated with components in the switch fabric are considered when the system is designed so that this assumption can be satisfied.

between inputs and outputs. If L is not proportional to m, bit stuffing can be used, such that the segment length is $\lceil \frac{L}{m} \rceil$. Each segment is transferred to its destined output through a single switch plane. We are not concerned about how a cell is partitioned, but we require that each segment has the same length. Also, each of these m segments can be sent through a w-bit wide data bus. We use two time units: a cell slot and a bit-clock cycle. A bit-clock cycle is the time to transmit a bit-set or w bits from an input to an output in the switch. A cell slot comprises $\lceil \frac{L}{w \times m} \rceil$ bit-clock cycles. Figure 2 shows a w-bit wide data bus between input-SE and SE-output.

In a crossbar-based switch, an arbitration process selects the set of cells that can be sent through the crossbar to resolve input and output port contention. The arbitration process can have a distributed nature [10] or have a centralized nature [4], [11], [14].

We consider a distributed arbitration where the input arbiters are in the input ports, and the output arbiters are in the crossbar fabric [10].[3] By using redundant arbitration planes, every plane has a copy of the output arbiter. The arbitration result configures the SE that interconnects the selected pair (I, J) to transmit a segment next cell slot. An SE is in either

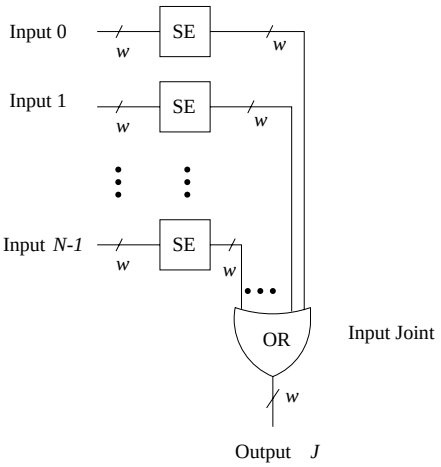

Fig. 2. Input joint: logic of a connection of all inputs to one output.

the active or idle state. An SE in the active state transmits data from an input to the output. An SE in the idle state disconnects the data transmission from an input to the output. The bit values in the active state depend on user data while the bit values in the idle state depend on the logic design. The idle state can be implemented by a transmission of either "0"s or "1"s at the SE output. We assume the idle value for an SE is "0" for the rest of this paper.

III. Fault Models

We consider single fault models for SEs and arbiters [16], [26], [28]. We do not consider multiple faults because they have a very low probability of occurrence. We divided the single-fault models into two categories: arbiter faults and crossbar faults.

[3]A centralized arbitration can also be considered, as discussed in Section VIII.

We define a data or bit *collision* as when two bits come from two different inputs to the input joint, which is equivalent to an OR function, so the result is the product of the logic function. The input joint is the only place where collisions may occur.

A. Arbiter Fault Models

The following is a list of arbiter fault models. Figure 3 shows an example of the arbiter faults. In this figure the switch fabric and the inputs are related to a single output (i.e., a single output arbiter) to simplify our description. In this figure, all requests from different inputs go to the same output arbiter.

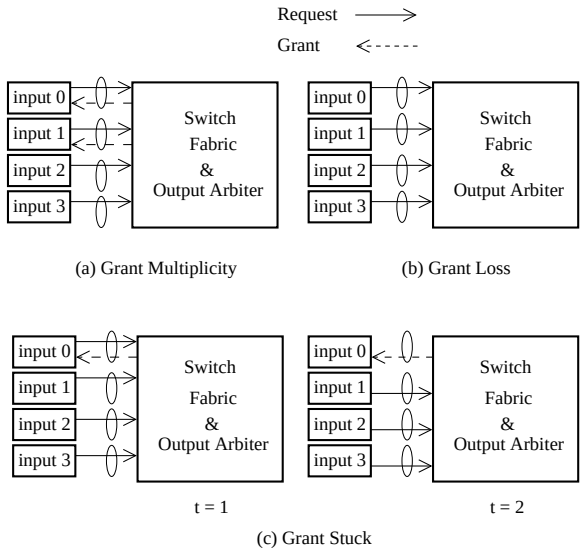

Fig. 3. Examples of arbiter faults.

Grant multiplicity. If two or more inputs are matched with a single output, multiple grants are issued. This misbehavior might cause a collision between two data segments. Figure 3(a) shows an example of this fault, where two or more grant signals are issued.

Grant loss. If the arbiter receives at least a request, a grant should be issued. If the arbiter does not issue a grant, the grant is considered lost. Figure 3(b) shows an example of grant loss.

Grant stuck. This fault occurs when a port or a group of ports receive a grant consecutively (back to back) while other ports remain starved. Figure 3(c) shows an example of this fault.

B. Crossbar Fault Models

These faults are related to SEs. We divide these faults into two types, the control and the data-path fabric faults. The control type is related to the SE state and the data type is related to the link status where the data bits pass through.

Idle-stuck fault. This fault occurs in an SE when the SE permanently remains in idle state regardless of the arbiter result. The transmitting segment is lost as a result. Figure 4 (a) shows an example of this fault. In this case, the segment is not transmitted to the destined output.

Active-stuck fault. Similar to a cross stuck fault, a toggle stuck fault occurs when an SE permanently remains in active

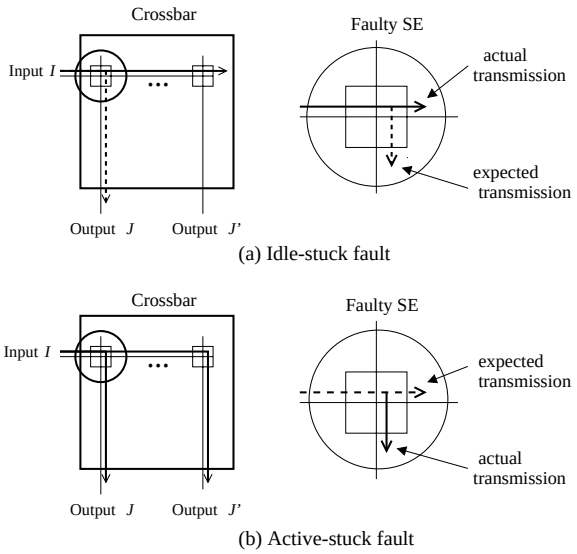

Fig. 4. Idle-stuck and active-stuck faults.

state, regardless of the arbiter result. In occurrence of this fault, a transmitting segment may be misdirected (or duplicated) to a wrong output. Figure 4(b) shows an example of this fault. In this case, an output receives a copy of the wrong segment.

1/0-stuck fault. A 1/0 stuck fault occurs in the data links when an output link remains in either logic zero or logic one instead of passing the user data when an SE transmits a segment. As a result, the segment would be corrupted.

IV. Detection System

The detection system concurrently operates with the switch on-line. The selection and duplication of the data, which is called the comparison data, of a cell to be sent on the spare plane is performed at the input port. When a cell is granted by arbitration, the cell and comparison data are sent through the active and spare planes. At the output port, the comparison data is compared to the user cell data. A discrepancy exists whenever a difference on one or more bits is detected on the comparison data.

The proposed fault-detection schemes are based on comparisons of redundant data transmitted through the spare planes. The spare planes transmit a copy of some data from the active planes so the spare planes are also tested. The comparison is done by pairs of planes on a bit-clock basis. Since the comparison of three planes is done separately and concurrently, plane discrepancy detection can be performed effectively.

The detection process is summarized as follows:

1 A copy of the data from the active planes between the input-output pair (I, J) of the switch fabric is sent on the spare plane.

2 After the data has been transmitted, the result of the comparison at the output port can have either of the following outcomes:

(i) If all data from these $m + z$ planes has no discrepancies, the planes are free of faulty parts.

(ii) If a discrepancy is detected, the plane with the discrepancy is located according to the discrepancy type.[4]

V. Schemes for Discrepancy Detection and Location

In this section, we present three data redundancy schemes for discrepancy detection and plane location. The details of the schemes and how data redundancy is produced are presented below in an example. For description 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 assume that each data bus is four bits wide (e.g., $w = 4$) and the cell length is 64 bytes ($L = 512$ bits).

Data on Active Planes. A bit $\pi^l_{t_b}$ transmitted through each plane corresponds to an active switching plane π, where $\pi = \{W, X, Y, Z\}$ in a bit-clock cycle t_b. l is one of the data lines, $l \in \{A, B, C, D\}$. In a cell slot where the pair (I, J) are connected by arbitration for cell transmission, t_b is numbered as $0, ..., \frac{L}{w \times m} - 1, \frac{L}{w \times m}, ...$ A bit-set π_{t_b} is then formed by w bits.

In our example, the number of bit clocks in the first cell slot is $t_b = 0, 1, 2, ..., 31$. The collection of t_b can continue in the next cell slot where the same input-output pair are selected for transmission to accumulate a large number of comparison bits. For example, in the second cell slot to which this input-output pair are connected, the bit cycles are $t_b = 32, ..., 63$, and this is done in similar way for future cell slots. This is what we call continuous testing.

Data on the S plane. In the plane S, each of the bits transmitted through the lines $\{A^S, B^S, C^S, D^S\}$, where the superindex S indicates that a line belongs to plane S, is denoted as $S^l_{t_b}$ and a bit set through the S plane as S_{t_b}.

Figure 5 shows the location of the parts in the fault detection system. The input ports have transit queues that hold the segments to be sent through planes W to Z. The input ports produce the redundant information by copying the redundant bits into the transit queue for the spare plane S. The comparators are located in the output ports. The fault tolerant manager receives the discrepancy information from the output ports and re-configures the switch by sending the replacement control signal when a plane is faulty.

A. Scheme 1: Redundant Data on the Spare Plane

Continuing with our example, the comparison bit sets to be copied and transmitted on the spare plane are chosen as follows:

- At time $t_b = 0$, the bit set S_0 in S is a copy of the data transmitted on the active plane W at time $t_b = 0$, W_0.
- At time $t_b = 1$ the bit set S_1 in S is a copy of the data transmitted on the active plane X at time $t_b = 1$, X_1.
- In general, at time t_b, the bit set S_{t_b} in S is a copy of the plane $t_b \bmod m$ ($t_b \bmod 4$), where the resulting numbers $0, 1, 2, 3$ correspond to active planes W, X, Y, Z, respectively.

[4]Note that when a spare plane shows a discrepancy, two or more differences are observed.

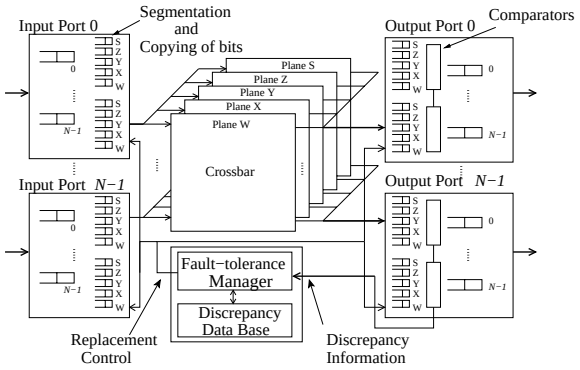

Fig. 5. Example of a 4-plane switch with a single spare plane.

During the first cell slot where the input-output pair (I, J) are connected, the transmitted bit sets on the S plane are: W_0, X_1, ..., Z_3, W_4, ..., Z_7, ..., Z_{31}. In this case, the first set, W_0, is sent in the first bit cycle $t_b = 0$ and the last set, Z_{31}, is sent on the 32^{nd} bit cycle ($t_b = 31$). Figure 6 shows the data distribution in all planes. In the second cell slot, in which the same input-output pair are connected, the t_b count continues, (e.g., 32, ..., 63), and so on.

The selection (or order) of the data bit sets to be copied on the S plane is arbitrary, and we have chosen the way as described above to simplify the description.

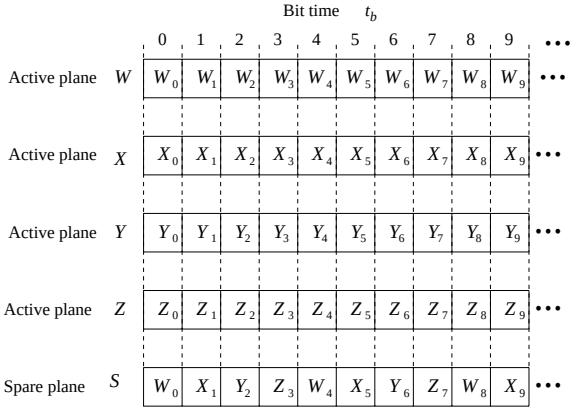

Fig. 6. Scheme 1: bit distribution on active and spare planes.

Once the data is received by the output port J, the redundant data is compared as shown in Figure 7. The comparison results between the spare plane and the others are quickly obtained because the spare plane has transmitted redundant data from all active planes. In this way, a discrepancy can be quickly detected and located. A discrepancy can be detected in the spare plane S or in any of the active planes. We define a bit set $\beta 1$ of plane $P1$ as discrepant from the bit set $\beta 2$ of plane $P2$ if any bit of $\beta 1$ differs from its corresponding bit in $\beta 2$, where $\beta 1$ and $\beta 2$ are any comparison bit sets that have the same bit values at the input ports.

At the output port, the data transmitted on the active planes is compared to the data transmitted on the S plane to search for a discrepancy. We divide the location of the plane with a discrepancy into two cases:

I Discrepancy in the spare plane S. If the data from the S plane differs from two or more active planes, plane S is diagnosed to have a discrepancy.

II Discrepancy in an active plane F. If the data from an active plane differs from the data on the S plane but, any other active plane data does not differ from that in the S plane, the active plane is declared to have a discrepancy.

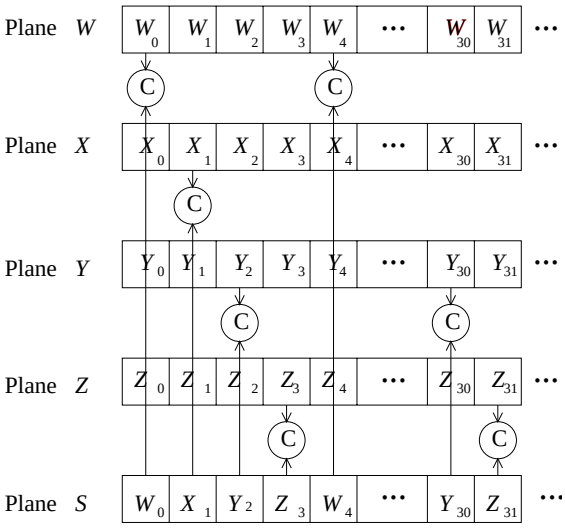

Fig. 7. Comparison of bit sets in Scheme 1.

B. Scheme 2: Data complements on the spare plane

Some bit combinations produced by a fault may be undetectable because some user data combinations may mask the fault. The discrepancy coverage depends on the combination of values sent through by the user data. This coverage is estimated in Section VI.

The efficiency of Scheme 1 to detect discrepancies depends on the distribution of “0”s (0-bit ratio) and “1”s for each comparison bit. It can be easily 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, which is unpredictable.

As an example of the dependence on the “0” value, let us consider a comparison of the following 4-bit strings. If the legitimately transmitted string “0001” collides with “0000” or “0001” at the input joint, it produces an undetectable collision result (the output looks like the legitimate input “0001”). The string “0001” has only two combinations that a collision may produce a masked fault or an undetectable discrepancy result. In another example, if the bit string “1100” collides with any bit string: “0000”, “0100”, “1000”, or “1100”, the result is an undetectable collision result. Therefore, the number of “0”s in the legitimate string makes the detection more collision-sensitive.

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

In Scheme 2, the active planes transmit comparison bit sets and their complements. Therefore, the spare plane transmits

a copy of the comparison bit sets and the bit sets that could not be transmitted through the active planes because of the transmission of the complements bit sets. In other words, the user data is distributed between an active plane and the spare plane.[5]

Using the switch in our example, the bit sets in the active plane W during the first cell slot are: W_0, $\overline{W_0}$, W_2, ..., W_7, W_8, $\overline{W_8}$, W_{10}, ..., W_{30}, 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 similar way for the rest of the active planes. During the same cell slot, the spare plane S transmits: W_0, W_1, X_2, X_3, ..., Y_{12}, Y_{13}, ..., Z_{30}, Z_{31}. Figure 8 shows our example of this scheme. In this figure, we also represent a bit set from planes W, X, Y, Z, as above. Each letter corresponding to an active plane and S to the spare plane.

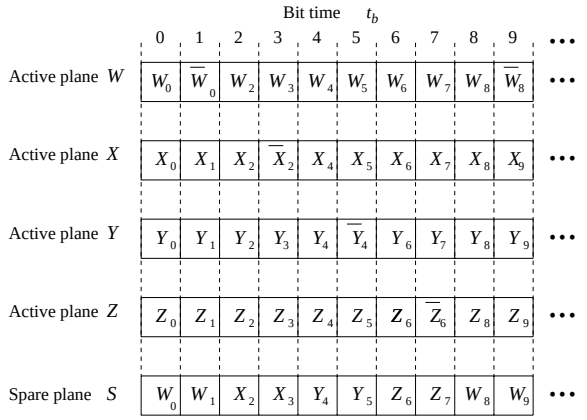

Fig. 8. Scheme 2: bit distribution on planes with bit complements.

C. Scheme 3: Modifying Hardware for Collision Monitoring

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 smaller than the number of comparison bits used by Scheme 1 because of the transmission of the complemented bits in Scheme 2. As a result, the time to collect a required number of comparison bits in Scheme 2 is twice the time in Scheme 1. As an example, we can see in Figure 6 that in eight bit cycles, an active plane sends one comparison bit set and its complement (e.g., W_0 and $\overline{W_0}$). In Scheme 1, an active plane sends two comparison bit sets (e.g., W_0 and W_5) in the same period.

Scheme 3 uses the properties of Schemes 1 and 2 and overcomes the disadvantage of Scheme 2. Scheme 3 also produces a discrepancy coverage independent of the user data, thereby providing a high coverage. Scheme 3 uses the data distribution as in Scheme 1. In addition, Scheme 3 includes a very small modification to the logic function of the *input joint*, the *OR* function used as *input joint* is replaced by an *exclusive OR* function. This small modification balances the fault collision sensitivity for the 0-bit ratio, without affecting the behavior of the input joint. With an *OR* function as the input joint, when a collision occurs, a legitimate "1"-bit produces an output that is not sensitive to a collision

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

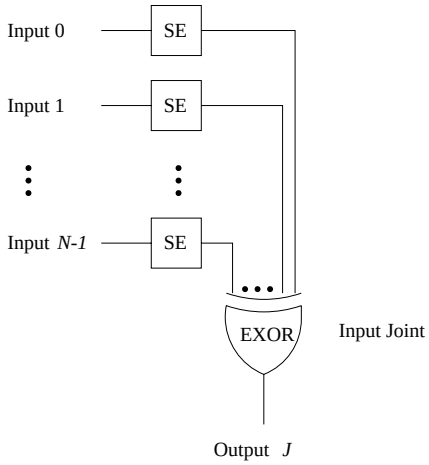

Fig. 9. Scheme 3: modification of the input joint.

occurrence with either "0" or "1" at the input joint (insensitive to the two values). Using an *exclusive OR*, a "1-1" collision is detected since the joint produces an erroneous result in this occurrence. If an original bit "1" collides with a colliding bit "1", the resulting output is bit "0". In this way,the "1"s in a string no longer masks a collision with other "1"-bits in the other string. As for the case when a legitimate "0"-bit is not sensitive to a "0"-bit colliding value (as shown in Section V-A), this situation cannot be better since "0" is the idle output value.

VI. Evaluation of Discrepancy Detection

As described in the previous section, a fault model can be handled as a cell collision. The redundant data or bit sets are concatenated to form a bit string, the comparison bits. The comparison bits transmitted legitimate (or fault-free) through an SE are denoted as string B_l, and the faulty ones transmitted through an SE are denoted as string B_c or the colliding string.

We estimate the fault detection coverage and the failure probability for each scheme by using the probabilities of a single bit to be "1". We define $P_l(1)$ as the probability of a legitimate bit as being of value "1" and $P_l(0)$ as the probability as being of value of "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" and $P_c(0)$ for a value "0." For simplicity, we assume that:

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

where b is either "0" or "1," as described above. We evaluate the performance of these detection mechanisms by looking at the coverage or detection failure probability. These are functions of the probability product between the legitimate and the colliding bits. $p(b)$ is used for simplicity to represent either bit probability in the following graphs.

However, we make a distinction between P_l and P_c in the following equations. The coverage Cov of a detection scheme is estimated as:

$$Cov = \frac{D}{D+U}, \tag{2}$$

where D is the average probability of having a detectable combination of a collision and U the average probability for an undetectable combination of a collision in n bits, both for a given scheme. To make the coverage approach 1, the undetectable probability needs to tend to 0. One desirable feature of a given detection scheme is a small number of undetectable collisions.

The indication of effectiveness for each scheme is shown by plotting the failure probability, Fp, which is:

$$Fp = 1 - Cov. \tag{3}$$

A. Analysis of Scheme 1

The coverage of Scheme 1 is defined by Eq. (2):

$$Cov_{s1} = \frac{D_{s1}}{D_{s1} + U_{s1}},$$

and U_{s1} and D_{s1} are:

$$U_{s1} = \sum_{k=0}^{n} C(n,k) P_l(0)^{n-k} P_l(1)^k \epsilon_1 \tag{4}$$

where $\epsilon_1 = \sum_{j=0}^{k} C(k,j) P_c(0)^{n-j} P_c(1)^j$, and

$$D_{s1} = \sum_{k=0}^{n} C(n,k) P_l(0)^{n-k} P_l(1)^k \epsilon_2, \tag{5}$$

where $\epsilon_2 = \sum_{j=0}^{n} C(n,j) P_c(0)^{n-j} P_c(1)^j - \sum_{i=0}^{k} C(k,i) P_c(0)^{n-i} P_c(1)^i$. U_{s1} and D_{s1} are the average probabilities of the occurrence for a detectable and an undetectable combination for Scheme 1, respectively. Here, j and i are variables that depend on a specific combination; they are used to indicate the number of "1"s in a colliding combination, k is the number of "1"s in the legitimate combination of the n comparison bits. $C(r,s)$ is the number of combinations with r "1"s in a combination with s bits. These equations are derived in Appendix A. For this scheme, n is:

$$n = \lfloor \frac{zt_b}{m} \rfloor w, \tag{6}$$

where t_b is the number of bit cycles to which a the pair (I, J) are connected during one or more cell slots. Eq. (6) provides the number of comparison bits. At each bit cycle, one active plane and the spare planes are compared. 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 10 shows the failure probability of Scheme 1, $Fp_{s1} = 1 - Cov_{s1}$, obtained from Eqs. (2)-(5) in function of n. This scheme performs best when the distribution of "0" and "1" values are $p(1) = p(0) = \frac{1}{2}$, and the performance degrades for any other value of $p(b)$. This graph also shows that the number of comparison bits to get a $Fp = 10^{-6}$ with this scheme is 42.

B. Analysis of Scheme 2

In Scheme 2, more combinations of B_c are detectable because of the complements provided for each legitimate combination. The coverage for Scheme 2, according to Eq. (2) is:

$$Cov_{s2} = \frac{D_{s2}}{U_{s2} + D_{s2}}.$$

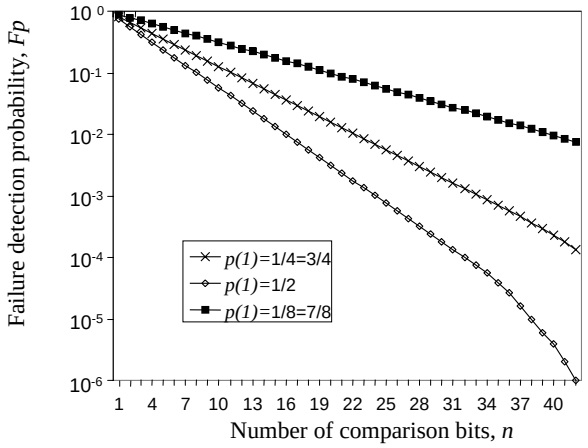

Fig. 10. Failure probability for Scheme 1 in function of n and $p(1)$.

The average probability of the occurrence for a detectable, D_{s2}, and an undetectable combination of a collision, U_{s2}, for Scheme 2 are:

$$U_{s2} = P_c(0)^n \tag{7}$$

and

$$D_{s2} = 1 - P_c(0)^n, \tag{8}$$

where n, for this scheme, is:

$$n = \lfloor \frac{zt_b}{2m} \rfloor w. \tag{9}$$

The derivations of Eqs. (7) and (8) are shown in Appendix B. Figure 11 shows the failure probability of Scheme 2, $Fp_{s2} = 1 - Cov_{s2}$, for 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 and the failure probability decreases. For a $p(1) = \frac{1}{2}$, the failure probability of this scheme is half of that of Scheme 1. The number of comparison bits in Scheme 2 is half of the number bits used for comparison for Scheme 1 collected in the same period of time. As shown in the figure, to achieve $Fp = 10^{-6}$, only 20 bits are needed (about half of those needed for Scheme 1).

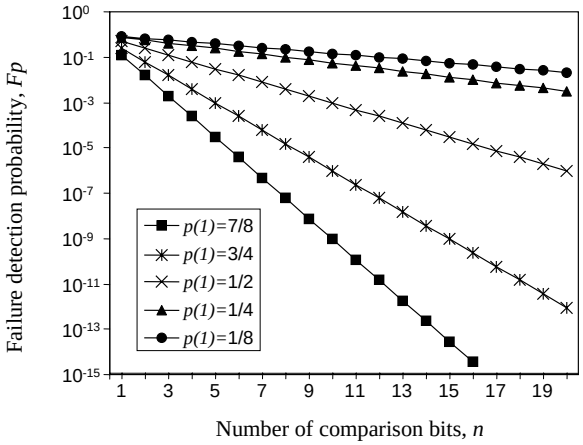

Fig. 11. Failure probability of Schemes 2 and 3 in function of n and $p(1)$.

C. Analysis of Scheme 3

Similarly, the coverage of this scheme is defined by Eq. (2):

$$Cov_{s3} = \frac{D_{s3}}{U_{s3} + D_{s3}},$$

In this scheme, U_{s3} and D_{s3} are the same as those for Scheme 2:

$$U_{s3} = P_c(0)^n \tag{10}$$

and

$$D_{s3} = 1 - P_c(0)^n. \tag{11}$$

In these equations, n is defined by Eq. (6). The time in what n can be collected in this scheme is the same as in Scheme 1. The derivations of Eqs. (10) and (11) are shown in Appendix C.

The failure probability for Scheme 3, $Fp_{s3} = 1 - Cov_{s3}$, is the same as for Scheme 2 in terms of n, as shown in Figure 11. However, since Scheme 3 compares more bits than Scheme 2 in the same period of time, it provides a faster detection than Scheme 2.

D. Comparison of Schemes

Figure 12 shows a comparison of the Fp, in function of n, of 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 n. Figure 13 shows the detection coverage for all three schemes as a function of the number of bit cycles.

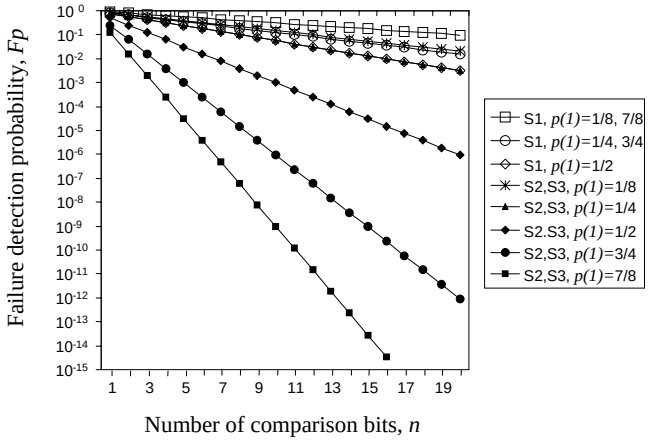

Fig. 12. Comparison of schemes 1, 2, and 3.

As this figure shows, the most effective scheme is Scheme 3. Twenty bits (collected in 5 comparison bit sets) are sufficient to achieve a small failure probability, for example $Fp = 10^{-6}$, for $p(1) = \frac{1}{2}$, which can be achieved in a single cell slot in our example (e.g., $m = 4$ and $z = 1$). Scheme 1 would need 42 bits collected by 11 comparison bit sets or two cell slots to achieve a similar result in our example.

Schemes 1 and 3 have almost the same hardware complexity. The number of redundant bits at each time slot is the same in these two schemes. 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 active and spare planes transmit user data (the spare plane becomes an active plane). More importantly, the number of bits that can be used for comparison is half that used in Schemes 1 and 3. However, Scheme 2 can be used when the input joint logic cannot be modified.

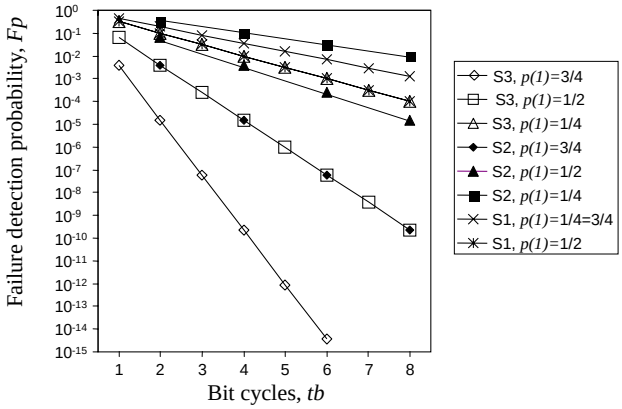

Fig. 13. Comparison of schemes 1, 2, 3 in terms of bit cycles.

E. Rotation of Planes

We define the *utilization ratio* of a plane for detection as the number of comparison bits over the total number of transmitted bits during a cell slot. The higher the utilization ratio, the faster the comparison bits, n, are collected. In our example, Scheme 1 provides a utilization ratio of 0.25 for each active plane and 1.0 for the spare plane. In Scheme 2, the utilization of active planes is 0.125, while in the spare plane is 0.50 (the other 0.50 is non-redundant user data). Scheme 3 has the same utilization as Scheme 1. This unbalanced utilization between the active and spare planes favors the coverage of the spare plane. To balance the utilization ratio and improve the detection time for the active planes, rotation of planes (i.e., rotation 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 cyclical manner among all planes so that the failure probability is decreased. In this case, the utilization ratio for Schemes 1 and 3 is 0.40 for any plane, and for Scheme 2, 0.20 for any plane. Rotation of planes modifies n as follows. For Schemes 1 and 3:

$$n = \lfloor \frac{2zt_b}{m+z} \rfloor w, \tag{12}$$

and for Scheme 2:

$$n = \lfloor \frac{zt_b}{m+z} \rfloor w. \tag{13}$$

In Eqs. (12) and (13), the ratios $\frac{2z}{m+z}$ and $\frac{z}{m+z}$ are the average utilization of a plane for schemes 1, 3, and 2, respectively. Rotation of planes can be used to reduce the time

	Scheme 1				Scheme 2				Scheme 3			
m	z=1	z=2	z=4	z=8	z=1	z=2	z=4	z=8	z=1	z=2	z=4	z=8
2	1	1	1	1	1	1	1	1	1	1	1	1
4	2	1	1	1	2	1	1	1	1	1	1	1
8	6	3	2	1	5	3	2	1	3	2	1	1
16	21	11	6	3	20	10	5	3	10	5	3	2
32	84	42	21	11	80	40	20	10	40	20	10	5
64	336	168	84	42	320	160	80	40	160	80	40	20

TABLE I
NUMBER OF TIME SLOTS TO ACHIEVE $Fp = 10^{-6}$ FOR $p(1) = \frac{1}{2}$.

to achieve a low failure probability or to improve the coverage in a time period.

VII. INCREASING THE NUMBER OF SPARE PLANES

The discrepancy detection can be improved by using more than one spare plane. More bit sets from the user data can be selected to increase the number of bits compared per active plane per bit cycle. However, the cost increases proportionally to the number of spare planes. The detection coverage for any z can be estimated according to Eqs. (6) and (9), without rotation of planes, and to Eqs. (12) and (13) with rotation for Schemes 1, 3, and 2, respectively.

Table I shows the number of time slots to achieve $Fp = 10^{-6}$ with $p(1) = \frac{1}{2}$ is a function of the number of spare planes, z, and the used scheme. In this table, we used $L = 512$ and $w = 4$ for all cases. In this table, Scheme 3 is the most cost-effective in number of spare planes needed since it would need fewer spares than the other schemes. Note that for switches with a large number of active planes, say 32, the number of time slots to provide such $Fp = 10^{-6}$ is 5 time slots for $z = 8$, with Scheme 3. Furthermore, for a switch with a large m, the spare planes can be distributed among groups of active planes, such that a sufficient number of bits can be compared in a small period of time. The number of spare planes needed for an acceptable detection coverage can be obtained from Eqs. (6) and (9), given n, m, and w.

In [27], it was shown that the replacement of a faulty active plane can be done in a single cell cycle, such that the replacement process can be implemented in our switch model to provide a fast recovery.

VIII. APPLICATION TO OTHER SWITCH FABRICS

In this section we discuss the use of these schemes with different switch fabrics. In the case of a crossbar-based switch with a centralized arbiter, these schemes can be used to detect crossbar faults. However, the arbiter faults may not be covered unless arbiter redundancy is used. For instance, arbiter redundancy can be achieved by the following architecture. The m planes in the switch can be divided into r groups with g switch planes each, where $r \geq 3$. An arbiter can be assigned to each group of planes. Thus, the redundant arbiters obtain the same matching result at each time slot. Another choice is having redundant arbiters, an active and two spare arbiters. The active arbiter controls all planes and the two spare arbiters are used to check arbitration results from the active arbiter and from each other. However, the arbiter discrepancies may need to be detected separately from the data transmitted through the switch planes.

Another switch architecture is one with multiple-plane crossbars, where each active plane transmits a complete cell and each plane uses independent arbitration. Each plane decides whether to send a cell depending on its own arbitration result.[6] As a result, one input may send two or up to m cells to different outputs at a given cell slot. Fault detection for these switches has higher complexity since the crossbars on the different planes have different configurations and no redundancy is used. Our schemes do not directly apply to such architectures. However, this switch may not achieve an average high port speed so more complex detection mechanisms can be implemented.

In the case of a shared memory switch, the implementation of the detection schemes requires the switch fabric to provide enough memory to allocate the redundant data used for comparison. The transmission of cells would be as we have described in Section II. This increases the implementation complexity for a large switch and needs to be evaluated for a specific design.

These schemes can also be applied to multiple-stage switch fabrics, such as banyan-based networks in a multiple-plane switch. This requires a cell segmentation similar to the one described in Section II. In this way, using these schemes is possible since the transmission of cells follows redundant data paths. However, using these detection schemes will only determine which switch plane is faulty. On the other hand, the detection of a faulty SE in extended-banyan fabrics has been of interest because of the availability of multiple paths between an input and an output and the possibility of making the switch gracefully degrading. This requires a more complex detection scheme. Although the gracefully degrading characteristic of banyan fabrics is interesting for fault tolerance, input scheduling is more complex than that used for crossbars.

IX. CONCLUSIONS

We presented a series of concurrent fault-detection mechanisms that use the spare resources in the fault-detection process. We assumed a crossbar switch fabric because of its popularity. While using a crossbar switch fabric, with a large number of SEs (N^2), the paths and SEs to be tested are the ones that are used for transmission. One advantage of these schemes is that they do not increase cell header overhead. These schemes provide a fast detection and are simple to implement so they can be used in high-speed switches. Among all three schemes presented, Scheme 3 is the most effective, providing a very low failure probability. Although Scheme 2 presents a slower detection than Scheme 3, it can be considered a solution for the case when the input joint or the crossbar cannot be modified. We also showed that the rotation of the spare plane's role improves the performance in all these schemes. The proposed detection schemes can be used with transmission checking mechanisms to distinguish a faulty plane from a faulty transmission. These schemes can be adapted to a switch with different number of active planes

[6] This is a different transmission method from the one we assumed initially.

and a small number of spare planes. Moreover, these detection schemes can be applied to other switching fabrics, such as banyan networks.

REFERENCES

[1] A. Itoh, "A Fault-Tolerant Switching Network for B-ISDN," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1218-1226, October 1991.

[2] A. K. Somani and T. Zhang, "DIRSMIN: a fault-tolerant switch for B-ISDN applications using dilated reduced-stage MIN," *IEEE Trans. Rel.*, vol 47, pp. 19-29, March 1998.

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

[4] 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.

[5] M. Karol and M. Hluchyj, "Queuing in High-performance Packet-switching," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1587-1597, December 1988.

[6] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319-352, November 1993.

[7] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-queued Switch," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260-1267, August 1999.

[8] A. Mekkittikul and N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% throughput in input-queued switches," *IEEE INFOCOM '98*, vol. 2 , pp. 792 -799, March 1998.

[9] N. McKeown, "The *i*SLIP Scheduling Algorithm for Input-Queue Switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188-200, April 1999.

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

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

[12] G. Nong, J.K. Muppala, and M. Hamdi, "Analysis of nonblocking ATM switches with multiple input queues," *IEEE/ACM Trans. Networking*, vol. 7, issue 1 , pp. 60-74, February 1999.

[13] G. Nong and M. Hamdi, J.K. Muppala, "Performance evaluation of multiple input-queued ATM switches with PIM scheduling under bursty traffic," *IEEE Trans. Commun.* , Vol. 49, issue 8, pp. 1329 -1333, August 2001.

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

[15] Cisco Systems, 12016 Multigigabit Switch Router, http://www.cisco.com, 1999.

[16] 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.

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

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

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

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

[21] 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.

[22] M. Anan and M. Guizani, "A fault tolerant ATM switching architecture," *Performance, Computing, and Communications Conference, 2000. IPCCC '00. Conference Proceeding of the IEEE International , 2000*, pp. 295 -301, 2000.

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

[24] A. Varma and S. Chalasani, "Fault-tolerance Analysis of One-Sided Crosspoint Switching Networks," *IEEE Trans. Comput.*, vol. 41, pp. 143-158, February 1992.

[25] J. Ghosh and N. Krishnamurthy, "Fault-Tolerant Arbitration in Multichip Crossbar Switches," *IEEE Proc. 6th Intl Conference on VLSI Design*, pp. 351-356, January 1993.

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

[27] K. Padmanabhan, "An Efficient Architecture for Fault-Tolerant ATM Switches," *IEEE/ACM Trans. Networking*, pp. 527-537, 1995.

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

APPENDIX

The equations presented in this appendix are derived by assuming a comparison of n bits. Let us take a bit string of n bits. There are two kinds of strings, a legitimate string B_l and a colliding string B_c.[7] A legitimate string is one that belongs to the original user data. The colliding string is a string that for some fault, collides with the original string at the input joint. Our objective is to point out the combinations of the legitimate and the colliding strings to estimate the fault detection coverage. The output data obtained from the input joint is classified into two categories, a detectable collided string and non-detectable collided string. A non-detectable collided string is one that has the same "1-0" permutation in the B_l string as the legitimate string, so that even in the case of a collision, this is not detectable. These cases may occur since not all collisions produce a detectable string. However, the fault is present. The number of detectable combinations in a colliding string depends on the scheme used and the values of the legitimate strings (i.e., user data).

A. Derivation of equations for Scheme 1

In this section we derive Eqs. (4) and (5) for Scheme 1. Table II shows examples of combinations that are detectable and that are not for the case of two comparison bits, $n = 2$. This table shows all possible combinations for B_l and B_c, and according to Scheme 1, the status U or D as undetectable or detectable combination, respectively. We see that any string B_c with all bits "0" produces an undetectable output for any bit values of B_l. When the bit values of the legitimate string B_l are "0"s, the collisions with colliding bits with a "1" value are the only collisions detectable. When the legitimate bit is "1", any bit value of the colliding bit, corresponding the legitimate bit position, is masked or undetectable. Another colliding string combination B_c that masks the output is the one that has the same permutation of B_l values as those of the legitimate string (e.g., $B_l = 01$ and $B_c = 01$).

In summary, to estimate the average probability of the detectable combinations D_{s1} with Eq. (5), we count the probabilities of the combinations that are undetectable, which are related to the number of "1"s in B_l, and these (probabilities) undetectable combinations are subtracted from all the possible combinations with n bits. In general, we observe that the number of detectable combinations for this scheme, is related to the number of "1"s in B_l. As in Table 5, the number of combinations that are detectable for a combinations of B_l with k "1"s is 2^k.

[7] How to create these strings is explained in Section V.

B_l	B_c	Status
0 0	0 0	U
0 0	0 1	D
0 0	1 0	D
0 0	1 1	D
0 1	0 0	U
0 1	0 1	U
0 1	1 0	D
0 1	1 1	D
1 0	0 0	U
1 0	0 1	D
1 0	1 0	U
1 0	1 1	D
1 1	0 0	U
1 1	0 1	U
1 1	1 0	U
1 1	1 1	U

TABLE II
EXAMPLE OF DETECTABLE AND NON-DETECTABLE COMBINATIONS FOR SCHEME 1 WITH $n = 2$.

Composed string B_l &$\overline{B_l}$	Non-detectable B_c
0 0 1 1	0 0 0 0
0 0 1 1	0 0 0 1
0 0 1 1	0 0 1 0
0 0 1 1	0 0 1 1
0 1 1 0	0 0 0 0
0 1 1 0	0 0 0 1
0 1 1 0	0 0 1 0
0 1 1 0	0 1 1 0
1 0 0 1	0 0 0 0
1 0 0 1	0 1 0 0
1 0 0 1	1 0 0 0
1 0 0 1	0 0 1 1
1 1 0 0	0 0 0 0
1 1 0 0	0 1 0 0
1 1 0 0	1 0 0 0
1 1 0 0	1 1 0 0

TABLE III
EXAMPLE OF DETECTABLE AND NON-DETECTABLE COMBINATIONS FOR SCHEME 2 WITH $n = 2$.

In Eq. (4), for estimation of the average coverage of undetectable cases U_{s1}. In the first factor, the sum counts the number of combinations that B_l has from zero to n "1"s in an n-bit string. The number of combinations with a number of k "1"s in an n-bit string is $C(n,k)$. The number of "1"s provides the combinations that are non-detectable. The probability of this combination is given by the product $P_l(0)^{n-k}P_l(1)^k$, where k is the number of "1"s in the related combination. The second factor in the equation estimates the number of non-detectable combinations of B_c. The number of non-detectable combinations are related to k: for k "1"s in B_l, the number of undetectable combinations in B_c is the number of accumulated combinations from zero to k "1"s in a given k. For example, when $k = 0$, there is one combination that is undetectable only. When $k = 1$, there are two non-detectable combinations. When $k = 2$, all four combinations of B_c are undetectable. The product $P_c(0)^{n-j}P_c(1)^j$ estimates the probability of the combination that corresponds to j "1"s.

This section explains the different terms in Eq. (5). The second factor (inside parentheses) estimates the probability of the detectable B_c combinations. This is the number of all cases minus the number of undetectable cases. The subtraction means that the number of detectable combinations of B_c is related to k "1"s in B_l. For instance, a combination where all are "0"s in B_c is always subtracted from the detectable combinations. The number of cases that are masked, as shown in Table II, are as follows: for $n = 2$, there is one combination with zero "1"s, two combinations with one "1", and one combination with two "1"s. Then, $C(k,i)$ determines the number of undetectable combinations of i "1"s in k bits.

$$U_{s1} = \sum_{k=0}^{n} C(n,k)P_l(0)^{n-k}P_l(1)^k\epsilon_1$$

where $\epsilon_1 = \sum_{i=0}^{k} C(k,i)P_c(0)^{n-i}P_c(1)^i$, and

$$D_{s1} = \sum_{k=0}^{n} C(n,k)P_l(0)^{n-k}P_l(1)^k\epsilon_2$$

where $\epsilon_2 = \sum_{j=0}^{n} C(n,j)P_c(0)^{n-j}P_c(1)^j - \sum_{i=0}^{k} C(k,i)P_c(0)^{n-i}P_c(1)^i$.

B. Derivation of equations for Scheme 2

This section explains the terms of Eqs. (7) and (8), for U_{s2} and D_{s2}, respectively. In this scheme, the complement bits $\overline{B_l}$ are also considered since the legitimate string is a composite B_l&$\overline{B_l}$. Note that there are some combinations that are not ever used in the composite string B_l&$\overline{B_l}$.

For brevity, Table III shows only the undetectable combinations for all the possible composite string combinations B_l&$\overline{B_l}$ for $n = 2$. The number of "1"s in B_l is at most n bits in this $2n$-bit string. We follow the same approach to estimate the average probability of the detectable and undetectable combinations. U_{s2} is explained first. Since the composite legitimate string B_l&$\overline{B_l}$ might collide with a similar string, the length of B_c is considered $2n$ bits. With this scheme, the number of undetectable combinations is always the sum from zero to n "1"s (and never more than n "1"s) in a string of $2n$ bits. Eq. (7) adds the probability of having these undetectable combinations. The term $C(n,j)$ is the number of combinations of j "1"s in n bits. The product $P_c(0)^{2n-j}P_c(1)^j$ is the probability of having this combination of "1"s in B_c. Eq. (8) is $1-U_{s2}$, all combinations minus the undetectable combinations.

$$U_{s2} = \sum_{j=0}^{n} C(n,j)P_c(0)^{2n-j}P_c(1)^j$$

or

$$U_{s2} = P_c(0)^n,$$

and

$$D_{s2} = 1 - \sum_{j=0}^{n} C(n,j)P_c(0)^{2n-j}P_c(1)^j$$

or

$$D_{s2} = 1 - P_c(0)^n.$$

B_l	B_c	Status
0 0	0 0	U
0 0	0 1	D
0 0	1 0	D
0 0	1 1	D
0 1	0 0	U
0 1	0 1	D
0 1	1 0	D
0 1	1 1	D
1 0	0 0	U
1 0	0 1	D
1 0	1 0	D
1 0	1 1	D
1 1	0 0	U
1 1	0 1	D
1 1	1 0	D
1 1	1 1	D

TABLE IV
EXAMPLE OF DETECTABLE AND NON-DETECTABLE COMBINATIONS FOR SCHEME 3 WITH $n = 2$.

C. Derivation of equations for Scheme 3

In Scheme 3, we use the same number of bits as in Scheme 1, n, for B_l and B_c. Table IV shows an example of a B_l with $n = 2$ bits. In this case, the number of detectable combinations increases. The only combinations that are non-detectable are the "0"-string B_c. So it is easy to start from U_{s3}, Eq. (10). So Eq. (10) only considers those combinations where all bits in B_c are "0"s. $P_c(0)^n$ is the probability of all bits in B_c being "0"s for n bits. Eq. (11) is obtained by considering all the combinations in an n-bit string minus the undetectable combinations, or $1 - U_{s3}$.

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

and

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