Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 410172, 13 pages
http://dx.doi.org/10.1155/2015/410172

Research Article

A Probabilistic Spatial Distribution Model for Wire Faults in Parallel Network-on-Chip Links

Arseniy Vitkovskiy, Paul Christodoulides, and Vassos Soteriou

Faculty of Engineering and Technology, Cyprus University of Technology, 3603 Limassol, Cyprus

Correspondence should be addressed to Paul Christodoulides; paul.christodoulides@cut.ac.cy

Received 4 October 2014; Accepted 11 January 2015

Academic Editor: Jinhu Lü

High-performance chip multiprocessors contain numerous parallel-processing cores where a fabric devised as a network-on-chip (NoC) efficiently handles their escalating intertile communication demands. Unfortunately, prolonged operational stresses cause accelerated physically induced wearout leading to permanent metal wire faults in links. Where only a subset of wires may malfunction, enduring healthy wires are leveraged to sustain connectivity when a partially faulty link recovery mechanism is utilized, where its data recovery latency overhead is proportional to the number of consecutive faulty wires. With NoC link failure models being ultimately important, albeit being absent from existing literature, the construction of a mathematical model towards the understanding of the distribution of wire faults in parallel on-chip links is very critical. This paper steps in such a direction, where the objective is to find the probability of having a "fault segment" consisting of a certain number of consecutive "faulty" wires in a parallel NoC link. First, it is shown how the given problem can be reduced to an equivalent combinatorial problem through partitions and necklaces. Then the proposed algorithm counts certain classes of necklaces by making a separation between periodic and aperiodic cases. Finally, the resulting analytical model is tested successfully against a far more costly brute-force algorithm.

1. Introduction

Continuous complementary metal-oxide-semiconductor (CMOS) transistor miniaturization, following Moore's law, has sparked the multicore era [1, 2] in which the architectural paradigm dictates that software application execution is handled by numerous processing cores that operate in parallel. This modular design of chips, including general-purpose chip multiprocessors (CMPs), not only ensures ultrahigh performance attainment but also provides a number of advantageous attributes such as those of power and thermal management, reconfigurability, and fault-tolerance, among others [3–5]. Networks-on-chips (NoCs) [6, 7], microscale equivalents of large-scale interconnection networks [8, 9], which also draw similarities to complex networks [10–12], as they are homogenous and exhibit clustering behaviour and short-distance communication between node-pairs, have become the de facto communication backbone in these multicore chips, including CMPs such as the Tilera TILE64 CMP [2] and Intel's 48-core Single-chip Cloud Computer (SCC) [1], hence becoming inherent components in these parallel on-chip systems.

Unfortunately, deep submicron CMOS process technology is marred by increasing susceptibility to wearout, expected to increase by 10x in the next 10 years by ITRS [13], dramatically shortening the useful lifespan of multicore systems. Point-to-point links, comprising a set of parallel metallic wires [14], interconnect neighbouring routers, allowing message transfers on-chip. Prolonged operational stress onto these parallel wires gives rise to accelerated wearout, due to physical failure mechanisms primarily including electromigration (EM) and negative bias temperature instability [15] that cause permanent device faults that can, in turn, quickly lead to architectural-level failures and possible catastrophic NoC operational failure.

Faults induced by these anomalies are widely predicted to become increasingly common in the near future [16]. Research indicates that about 20% of all link errors are caused by permanent failures, occurring both at manufacture-time and at run-time [17, 18]. Moreover, the wire repeaters

(buffers), that is, the link drivers found in each router, the output latches, and the flip-flops of pipelined links are also susceptible and potentially vulnerable [19].

Even an isolated intrarouter or communication link failure in the NoC fabric can turn a static regular topology into an irregular one with subconnected geometry; hence, either physical connectivity among routers may not exist at all, and/or the associated routing protocol may not be able to advance packets to their destinations due to protocol-level violation(s) [20]. In-transit messages cannot traverse faulty links, with back-pressure causing the effects of the fault(s) to spread backwards, quickly causing congestion, and even leading the entire system to stall indefinitely. Further, vital components such as vital input/output (I/O) and various off-chip memory modules may be partitioned away from the CMP as well, making them inaccessible. Indeed, a number of surveys [4, 5, 21, 22], which outline the design challenges and lay the roadmap in future multicore design, have emphasized the need to conduct research and identify the primary challenges in NoC reliability maintenance techniques, including link-level fault diagnosis and tolerance, as a means to safeguard the scalability and performance sustainability of general-purpose CMPs and application-driven systems-on-chips (SoCs).

The facts that high data rate on-chip links are susceptible to increasing failure rates that decelerate the NoC's performance, that the NoC is critical to a CMP's overall functionality, and that no real link failure data are readily available from manufacturers (for obvious reasons) point to the crucial need in constructing a mathematical model to aid in the understanding and exploration of the distribution of wire faults in parallel on-chip links. This model can potentially be coupled to fault-tolerant mechanisms at the chip's architectural-level to realize improvements in intercore communication resiliency [1, 2]. This work takes decisive steps in such a direction.

In this paper, we derive and demonstrate combinatorics-based models that can be used to calculate the spatial probability distribution of individual wire faults in a parallel network-on-chip (NoC) [6] interconnect link given its bit-width (summation of the numbers of single-bit width healthy and unhealthy wires in this parallel link) and a given number of faulty single-bit width wires that reside in this link. Modern NoCs employ interrouter links comprising several unidirectional parallel wires [14] that can transfer an entire data flit in one clock cycle. Since each wire is associated with separate driver circuitry, a particular driver failure only affects its associated wire in a parallel NoC link. (The terms "unhealthy," "corrupted," "nonoperational," and "faulty" are used interchangeably throughout this paper.) (A flit, or flow-control unit, is a logical segment of a packetized message. In wormhole flow-control, often employed in NoCs, a packet containing data, comprised of a series of bits, is often split into several flits to reduce buffering requirements and to achieve efficient communication among router nodes.)

Previous research studies in [23–25], where the first two works constitute our previously published research, target the recovery of partially corrupted packetized data being retransmitted, using a partially faulty link recovery mechanism (PFLRM) that employs a shifting mechanism which leverages the existing healthy links in a partially faulty link, that is, a parallel NoC link in which a subset of its wires are faulty while the remaining wires are operational. These mechanisms retransmit a flit in a bit-shifted scheme from the sender router at every clock cycle, for a given number of cycles, so as to eventually receive all the essential information to enable recovery and reconstruction of the flit data at the receiver router. Under these mechanisms, it has been shown that the consecutiveness or "clustering" of these faulty wires, where each such cluster is separated from its neighboring clusters with at least a healthy link in between them, directly affects the recovery latency required to restore the received partially corrupted flit data at the receiver routers, hence directly impacting negatively the NoC performance. We are, therefore, particularly interested in the number of such consecutive faulty wires in a parallel NoC link as the "maximum wire fault clustering" (i.e., the longest existing consecutiveness of faulty wires in a parallel link, i.e., fault-segment) correlates to the number of overhead clock cycles that are required to retransmit a flit over a partially faulty parallel NoC link, as Section 2 will demonstrate with a detailed example; the wider this fault clustering is, the greater the number of flit retransmissions are needed for flit recovery, hence the lower the performances of the NoC and of the entire CMP. Note that we consider the two edge wires of the parallel link to be virtually consecutive for the functional purposes of the packetized message bit-shifting mechanism that forms part of the recovery in [23–25]; hence, the link arrangement forms a *virtual ring*, with the edge wires "touching" each other, as demonstrated in the example of Figures 1(a) and 1(b) where 5 wires are assumed to exist in a parallel link. We adopt a random spatial distribution of faulty wires in the parallel NoC link and aim to determine the probability distribution of corrupted (and noncorrupted) flit data bits (or associated NoC link wires), as no real data for wire failures in NoC links are published by IC manufacturers. (The terms "consecutive," "clustering," "adjacent," and "segment(s)" are used interchangeably throughout this paper.)

To effectively calculate the levels of these "parallel wire segmentations," we derive (or perfect) a novel algorithm that can be used to determine the segmentation probability for an ordered collection of objects (i.e., parallel wires in a NoC link) of two distinct classes: faulty wires and healthy or "nonfaulty" wires. The algorithm presented here is a more rigorous extension of a preliminary algorithm presented in [26], which heavily depended on a stated (unproven) conjecture that led to non-100%-precise results.

The goal of a complete mathematical model describing the probability distribution of the length of a fault-segment for a given number of parallel NoC wires and faulty wires is reached through a series of combinatorial arguments with regard to partitions and necklaces. Necklaces, apart from their intrinsic usefulness in the field of combinatorics, have proven to be a powerful tool in other areas of mathematics and other sciences. Some customary notions and theories related to necklaces include the Lyndon word [27], the actual homonym necklace problem (see, e.g., [28]), the necklace

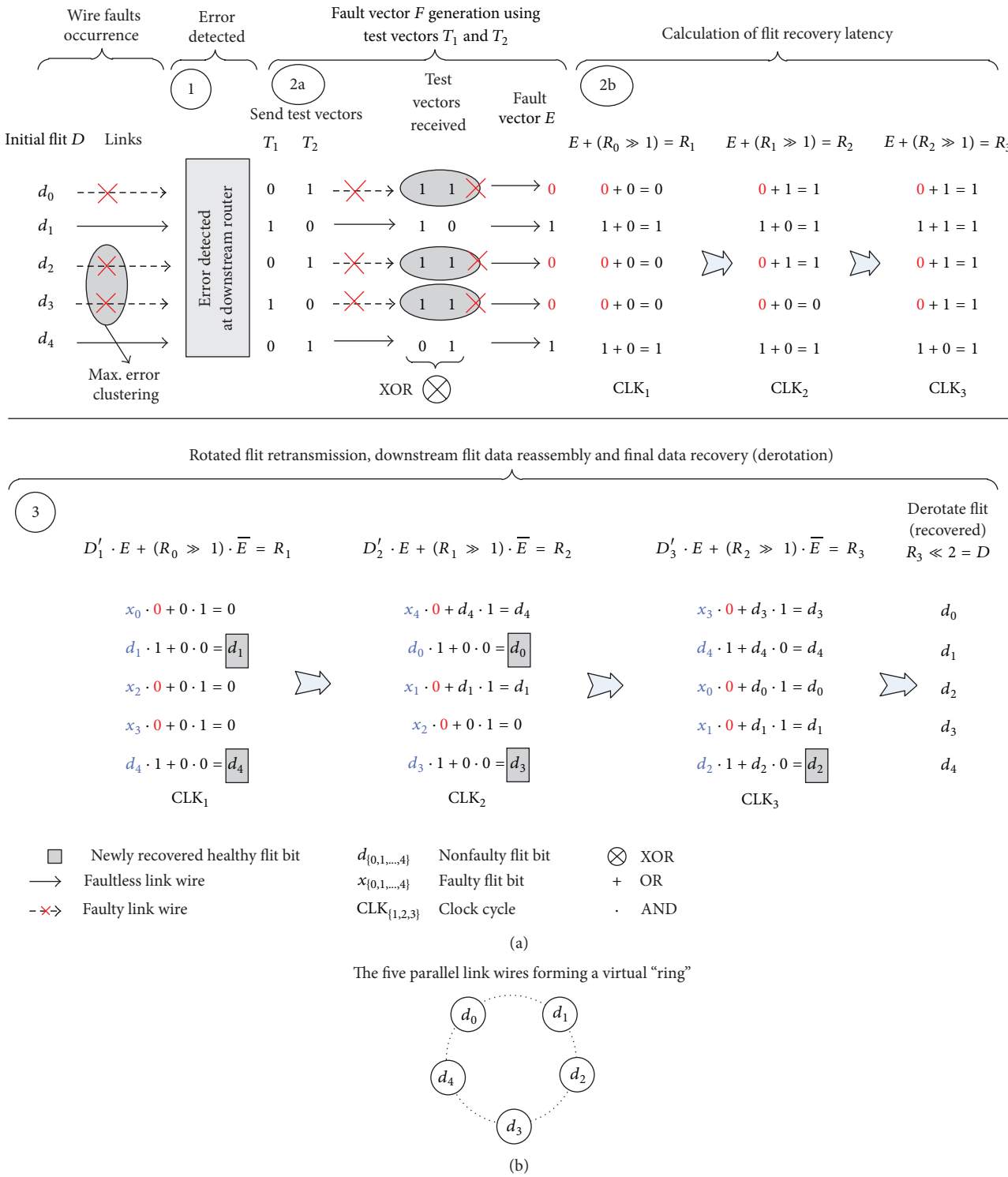

FIGURE 1: (a) Demonstration of the PFLRM functionality under all three phases of recovery, using a 5-bit flit width, a faulty wire clustering of 2, and a total of 3 faulty wires (60% faulty wires). Stuck-at-one permanent faults are assumed. In phase 2-a the fault vector is rotated twice until all bits of vector $\mathbf{R}_3$ equal 1, indicating a maximum fault clustering of 2. The boxed bit numbers under phase 3 indicate the respective newly recovered flit bits from the received and corrupted flit vector $\mathbf{D}'$. The final two-position anticlockwise deshifting at the downstream router recovers the final flit to exactly equal to $n + 1$, the error-free flit being sent from the upstream router; the recovery phase takes 3 clock cycles ($CLK_{\{1,2,3\}}$) to complete (1 base plus 2 recovery cycles). (b) The five wires comprising the same parallel NoC link forming a virtual "ring."

splitting problem [29], and most notably a proof of Fermat's little theorem [30].

The rest of this paper is organized as follows. Section 2 presents an overview of the partially faulty link recovery mechanism, published in our previous works [23, 24], which forms the basis for the proposed faulty wire distribution model presented in the paper. Next, in Section 3 the problem definition is formally given. Section 4 accommodates the algorithm that leads to the determination of the probability distribution of the length of fault-segments for a given number of wires and faulty wires comprising a parallel NoC link. The algorithm is constructed through basic counting principles and probability rules, where appropriate, and through a derivation showing its correspondence to an equivalent necklace problem. In Section 5, an arithmetic example demonstrates the effectiveness of the obtained analytical model, which is also verified by the results of a brute-force algorithm, with a runtime computation comparison of our analytical model versus the brute-force approach demonstrating its advantageous speedup. The further applicability of the presented algorithm is discussed in Section 6. Finally, Section 7 concludes this paper.

2. Demonstration of the Partially Faulty Link Recovery Mechanism

For purposes of completeness, we give an outline of the partially fault link recovery mechanism (PFLRM), which forms the basis on which we build our distribution model presented in this paper. A full description of the mechanism can be found in [23, 24]. The PFLRM scheme can detect bit corruptions in received flit data caused by independent wire failures in a parallel NoC link [14]. This detection initiates a data recovery process, whereby the downstream router instructs the upstream router to retransmit the flit(s) appropriately bit-rotated over a respective number of cycles, so as to bypass the faulty wire(s) that cause(s) the respective flit-bit error(s). Healthy bit fragments are extracted from each of received bit-rotated incarnations of the unhealthy flit and placed in an assembly block. PFLRM reacts dynamically to bypass permanent wire faults. While PFLRM also works for transient faults, for clarity we focus on permanent faults only.

Preliminarily, we denote an initially healthy parallel NoC link as a vector $\mathbf{D} = (d_0, d_1, \ldots, d_n)$, where $n \in \mathbb{Z}^+$, of $n+1$ noncorrupted flit bits sent from an upstream router towards a downstream router. Each such vector member represents the relevant and distinct bit of a flit traversing a relevant wire of a link. When faults occur, some of these wires, or link vector members, become faulty, and as a result a flit will be received at the downstream router with some of its bits being corrupted (while the remaining flit bits remain healthy and contain the correct data), denoted as $\mathbf{D}'$. Individual corrupted flit bits are denoted as x_i, $i \in \{0, 1, \ldots, n\}$. The relevant *positions* (placements or distribution) of faulty wires are assumed to be *random*. In our example of Figure 1(a) we assume that the wires carrying flit bits d_0, d_2, and d_3 between the upstream and the downstream routers in a 5-bit link become faulty simultaneously, respectively, denoted as x_0, x_2, and x_3. The same figure shows how PFLRM reconstructs corrupted flits transmitted over a partially faulty link (PFL) in a 3-phase scheme: (1) dynamic fault occurrence and detection, (2-a) fault vector generation, (2-b) flit recovery latency calculation, and (3) flit retransmission (upstream router), reassembly, and final flit recovery (downstream router). All 3 phases are executed when a wire fault(s) originally occurs; after the fault vector is generated, only the last phase is required, until later a new wire becomes faulty.

In phase 1, the error detection block in the downstream router detects the error (but does not recover or distinguish which bit(s) are erroneous), causing the initiation of phase 2-a. In phase 2-a, the upstream router stops the transmission of subsequent flits without dropping any packets and transmits two consecutive test vectors, $\mathbf{T}_1$ and $\mathbf{T}_2$, to the downstream router containing alternating "zeros" and "ones" with a one-bit shift difference between the two (refer to Figure 1(a) phase 2-a). Stuck-at-zero or stuck-at-one errors in any of the link wires are detected by a bitwise exclusive or (XOR) operation in the downstream router, indicated by a corresponding 0 in the respective generated fault vector $\mathbf{E}$.

The gist in recovering received flits corrupted during transmission is to utilize this fault vector as many times as required to extract healthy flit bits and use them to reassemble the entire healthy flit at the downstream router; then, repeat for the next flit(s). To do this, each healthy flit $\mathbf{D}$ at the upstream router is rotated clockwise a number of times, one bit position at every clock cycle i, such that $\mathbf{D}_i = \mathbf{D}_{i-1} \gg 1$, where $\gg$ denotes one-bit clockwise rotation, $i \in \mathbb{Z}^+$, and $i < n+1$ (the bit-width of the link) and sent over the parallel PFL a finite number of times (see next) to bypass faulty wires, while recovering flit bits over the remaining healthy wires. Due to this bit-rotational mechanism the wires at the edges of the link are considered to be virtually adjacent to each other, forming a "*ring*." For each rotated version of the received corrupted flit $\mathbf{D}'$, the healthy bits are compared against the fault vector and a flit recovery vector $\mathbf{R}_i$ is generated each time, such that

$$\mathbf{D}'_1 = \mathbf{D}', \qquad \mathbf{R}_0 = \mathbf{0}, \qquad \mathbf{R}_i = \mathbf{D}'_i \cdot \mathbf{E} + (\mathbf{R}_{i-1} \gg 1) \cdot \overline{\mathbf{E}}; \quad i \in \mathbb{Z}^+, \quad i < n+1, \tag{1}$$

where $\mathbf{R}_{i-1}$ is the partially recovered flit vector from the previous clock cycle and $\overline{\mathbf{E}}$ is the bit-wise negation of the fault vector $\mathbf{E}$. In other terms, if a bit from the current received flit vector $\mathbf{D}'$ is healthy (i.e., it utilized a faultless wire to arrive at the downstream router), as denoted by the corresponding bit (logic 1) of the fault vector $\mathbf{E}$, then it is extracted and assembled in the current flit recovery vector $\mathbf{R}_i$. Otherwise, that bit of $\mathbf{R}_i$ is left unconsidered for recovery; instead, the previously recovered corresponding flit bit is retrieved. For instance, in phase 3 of our example in Figure 1(a), in the first cycle of recovery (CLK_1), flit bits d_1 and d_4 are recovered; in CLK_2, these bits are rotated and flit bits d_0 and d_3 are recovered at their relative bit placement, with d_2 being recovered last in CLK_3. The relative rotations of the transmitted unhealthy flit vector $\mathbf{D}'_i$ and the flit recovery vector $\mathbf{R}_i$ in each cycle ensure that the recovered flit vector $\mathbf{R}_i$ is progressively built.

The recovery vector $\mathbf{R}_i$ requires a final $(i-1)$-bit anticlockwise derotation to reproduce the healthy flit vector $\mathbf{D}$ downstream. The number of these derotations is directly related to the number of consecutive faulty link wires; we refer to this as the "maximum wire fault clustering"; this also determines the number of additional clock cycles that are required to transmit a flit over the PFL for recovery purposes, referred to as the "flit recovery latency," with phase 2-b of PFLRM being exactly responsible in determining its size. Since in our example it equals two (with wires carrying flit bits d_0 and d_3 in Figure 1(a) being adjacent), $\mathbf{R}_3$ is finally anticlockwise-rotated two bit positions to recover $\mathbf{D}$. As mentioned above, possible wire faults at the link edges are also considered consecutive (bits d_0 and d_4 in Figure 1(a)), hence forming a "*ring*," as Figure 1(b) shows, due to the bit-rotational nature of the PFLRM algorithm; this is a vital postulation which is considered in our proposed mathematical model in this paper (see Sections 3 to 5). Phase 2-b utilizes the same hardware and recovery principle as those of phase 3, which recovers the actual flit. It basically rotates the initial fault vector $\mathbf{E}$ and compares it with its previous rotated version, assembling the logic-1 fault vector bits, until all bits equal 1, indicating the absence of errors, as vector $\mathbf{R}_3$ of Figure 1(a) shows. Since it uses the same hardware as that of phase 3 (calculation of the max wire clustering), (1) is reutilized with $\mathbf{D}_i$ replaced by $\mathbf{E}$ (the fault vector acts as our "data flit"), such that

$$\begin{aligned} &\mathbf{R}_0 = \mathbf{0}, \qquad \mathbf{R}_i = \mathbf{E}\cdot\mathbf{E} + \left(\mathbf{R}_{i-1} \gg 1\right)\cdot\overline{\mathbf{E}} \quad \text{or} \\ &\mathbf{R}_i = \mathbf{E} + \left(\mathbf{R}_{i-1} \gg 1\right)\cdot\overline{\mathbf{E}}; \\ &i \in \mathbb{Z}^+, \quad i < n+1. \end{aligned} \tag{2}$$

As the same mathematical principles ((1) and (2)), and, thus, hardware, are used for both the calculation of flit recovery latency and actual flit recovery, the PFLRM hardware overhead can be reduced. In theory, PFLRM can tolerate up to n faults (flit bit width minus 1), though in such scenario the recovery latency is prohibitive.

3. Problem Definition

As in Section 2, we assume a parallel NoC link consisting of W wires (W being equal to the size of vector $\mathbf{D}$) that are placed in *parallel*, wrapped around a common axis forming a *ring* shape (refer to the example contained in Figure 1 and outlined in Section 2). Each of these wires may be either healthy or faulty, but not both. The number of faulty wires F (F being equal to the number of d_i's in vector $\mathbf{D}'$), $0 \le F \le W$, $W > 0$, and the *position* (placement) of faulty wires are both *random*.

Consecutively positioned (adjacent) faulty wires form a *fault-segment* (in Figure 1(a) wires 2 and 3 form single fault-segment of size two, while wire 0 forms a separate fault-segment of size one). Let S, $S \le F$ ($S = 0 \Leftrightarrow F = 0$), denote the *size* (or *length*) of the *largest* fault-segment present in the link.

Note that one should not view the representation depicted in Figure 1 in terms of graph theory, let alone random graph theory, as at best the links (possible nodes) can only form a complete cycle ("ring"), or a graph consisting of several connected components, where each node can only have exactly two neighbors (with all clear implications regarding the clustering coefficients) [31, 32]. Exploring the probability distributions of the occurrence of such configurations is beyond the scope of the current paper. What is actually desired here is the following. For given values of W and F, we seek to find the probability distribution of S, $P_W(S \mid F)$.

4. Algorithm Derivation

Hereafter, we present an algorithm in order to find the probability $P_W(S \mid F)$ for each value of S, for given values of W and F. We find it useful to demonstrate the construction of the algorithm through arithmetic examples that clarify all notions involved.

4.1. Number of Possible Wire Arrangements. Let $A(W,F) = \{a_1, a_2, \ldots, a_{|A|}\}$ denote the set of all possible wire arrangements a_i for given W and F values. The *cardinality* (i.e., number of elements) of set $A(W,F)$ is simply equal to the number of combinations in choosing F faulty wires out of W wires, given by

$$|A(W,F)| = C(F,W) = \binom{W}{F} = \frac{W!}{F!\,(W-F)!}. \tag{3}$$

Similarly, let $A(W,F,S) \subseteq A(W,F)$ denote the set of all possible wire arrangements for given W, F, and S values. Then, the problem reduces to finding $|A(W,F,S)|$, which when divided by $|A(W,F)|$ will yield exactly the required probability distribution $P_W(S \mid F)$.

4.2. Size of Fault-Segment. Let H denote the number of healthy wires in a parallel link; that is, $H = W - F$ (H being equal to the number of x_i's in vector $\mathbf{D}'$). From the problem definition, the size of the largest fault-segment S has a lower bound which is equal to zero. However, it is possible to define the greatest lower bound of S more precisely (refer to Example 1 for demonstration) as

$$\left\lceil \frac{F}{H} \right\rceil \le S \le F. \tag{4}$$

Example 1. Let $W = 14$ and $F = 8$. Then, $H = W - F = 6$ and the greatest lower bound of S is $\lceil F/H \rceil = \lceil 8/6 \rceil = 2 \le S$. Consequently, for this case S can never be equal to 0 or 1. An illustration of such a wire arrangement is

$$\times \times \bigcirc \times \bigcirc \times \bigcirc \times \times \times \bigcirc \times \bigcirc \times \bigcirc, \tag{5}$$

with $S = 2$ (clustering of faulty wires 1 and 2, as well as of 8 and 9), where the link is shown as an "unwrapped" transverse section, with $\bigcirc$ and $\times$ denoting a healthy and faulty wires, respectively. The same link/wire representation is adopted throughout the remaining length of this paper.

4.3. Number of Fault-Segments. Let σ denote the number of fault-segments in a parallel link. It is not difficult to see (refer to Example 2) that

$$\left\lceil \frac{F}{S} \right\rceil \le \sigma \le \min\left(F - S + 1, H\right), \quad S > 0. \tag{6}$$

Example 2. Let $W = 14$, $F = 8$, and $S = 4$. Then, $\sigma_{\min} = \lceil F/S \rceil = \lceil 8/4 \rceil = 2$. Moreover, $\sigma_{\max} = \min(F - S + 1, H) = \min(5, 6) = 5$. Such wire arrangements for $\sigma = 2$, 3, 4, and 5 are, respectively, the following:

$$\begin{gathered} \times\times\times\times\bigcirc\bigcirc\times\times\times\times\bigcirc\bigcirc\bigcirc\bigcirc, \\ \times\times\times\times\bigcirc\times\times\bigcirc\bigcirc\times\times\bigcirc\bigcirc\bigcirc, \\ \times\times\times\times\bigcirc\times\bigcirc\bigcirc\times\times\bigcirc\times\bigcirc\bigcirc, \\ \times\times\times\times\bigcirc\times\bigcirc\times\bigcirc\times\bigcirc\times\bigcirc\bigcirc. \end{gathered} \tag{7}$$

Now let $S = 2$ for the same W and F values. Then, $\sigma_{\min} = \lceil F/S \rceil = \lceil 8/2 \rceil = 4$. Moreover, $\sigma_{\max} = \min(F - S + 1, H) = \min(7, 6) = 6$. Such wire arrangement for $\sigma = 4$, 5, and 6 are, respectively, the following:

$$\begin{gathered} \times\times\bigcirc\times\times\bigcirc\times\times\bigcirc\times\times\bigcirc\bigcirc\bigcirc, \\ \times\times\bigcirc\times\times\bigcirc\times\times\bigcirc\bigcirc\times\bigcirc\times\bigcirc, \\ \times\times\bigcirc\times\times\bigcirc\times\bigcirc\times\bigcirc\times\bigcirc\times\bigcirc. \end{gathered} \tag{8}$$

4.4. Initial Computations. Using basic counting and probability principles, it was noticed that for certain value choices of S and F, $P_W(S \mid F)$ can be obtained as follows:

$$\begin{gathered} P_W(0 \mid 0) = 1, \qquad P_W(1 \mid 1) = 1, \\ P_W(S = W \mid F = W) = 1, \\ P_W(S = W - 1 \mid F = W - 1) = 1, \\ P_W(S \mid F) = \frac{(S+1)}{\binom{W}{F}} \quad \left(\text{when } F = \frac{WS}{S+1}\right), \\ P_W(S \mid F) = \frac{W\binom{W-S-2}{F-S}}{\binom{W}{F}} \\ \left(\text{when } \left\lceil \frac{(F+1)}{2} \right\rceil \le S \le F \le W - 2\right). \end{gathered} \tag{9}$$

The number of wire arrangements for $W = 16$ are shown in Table 1.

The next step is to find a general algorithm for all possible (including the nonboldface in Table 1) cases.

4.5. String Representation of Wire Arrangements. The set A_i of t_i (equivalent) rotations (or circular shifts) $\{a_i, a_{i+1}, \ldots, a_{i+t_i-1}\}$ of a wire arrangement $a_i \in A(W, F, S)$ can equivalently be represented by the *string* $r_i = s_{i1}s_{i2}\cdots s_{iH}$, where s_{ij} is the size of the jth fault-segment followed by a single healthy wire. Making the convention that $s_1 = S$, we have

$$\sum_{j=1}^{H} s_{ij} = F \quad \left(0 \le s_{ij} \le S\right). \tag{10}$$

Note that (10) allows for $s_{i,j\neq 1} = 0$, denoting an empty fault-segment followed by a single healthy wire (refer to Example 3 below).

Example 3. Let $W = 14$, $F = 8$, and $S = 4$. Then, $H = W - F = 6$. Clearly, one of the respective wire arrangements, namely, $a = \times\times\times\times\bigcirc\bigcirc\times\times\bigcirc\times\bigcirc\bigcirc\times\bigcirc$, can be expressed by the string $r = s_1s_2s_3s_4s_5s_6 = 402101$. At the same time the wire arrangement $\bigcirc\times\times\times\times\bigcirc\bigcirc\times\times\bigcirc\times\bigcirc\bigcirc\times$ (arising by the *unary circular right shift* of a) is equivalent to the initial wire arrangement a and, thus, can also be denoted by the string r. Similarly, all circular shifts of a form an equivalence class represented by the string r.

Clearly, the set of strings r_i has a one-to-one correspondence to the set of nonintersecting subsets $A_i = \{a_i, a_{i+1}, \ldots, a_{i+t_i-1}\} \subseteq A(W, F, S)$ (with *cardinality* $|A_i| = t_i$). Thus, $\bigcup_{i=1}^{n} A_i = \{a_1, a_2, \ldots, a_{|A(W,F,S)|}\} = A(F, W, S)$, where n is the number of all subsets A_i and the actual number of wire arrangements can be found as follows:

$$|A(W, F, S)| = \sum_{i=1}^{n} |A_i| = \sum_{i=1}^{n} t_i. \tag{11}$$

The string representation for a wire arrangement, as described above, will then allow us to find all subsets A_i. We now introduce some terminology that will help us to reach this goal.

Definition 4. (a) The string $r_i = s_{i1}s_{i2}\cdots s_{iH}$ is said to be *periodic if and only if* there exists positive integer $1 \le \tau_i < H$ such that $s_{i,j+\tau} = s_{ij}$, for all $j = 1, 2, \ldots, H - \tau_i$. We call τ_i the *period of string* r_i.

(b) If there is more than one τ_i satisfying the condition (a) above, then the string is said to have *multiple* periods that are all divisors of H.

(c) If condition (a) is not satisfied, although the string r_i is *nonperiodic*, for the sake of generality, the period is considered to be $\tau_i = H$.

(d) The *period* t_i *of any wire arrangement* $a_i \in A(W, F, S)$ from subset A_i, represented by the respective string r_i of period τ_i, can be obtained as follows:

$$t_i = \sum_{j=1}^{\tau_i} \left(s_{ij} + 1\right) = \sum_{j=1}^{\tau_i} s_{ij} + \tau_i. \tag{12}$$

Definition 5. (a) The *frequency* ϕ_i *of string* $r_i = s_{i1}s_{i2}\cdots s_{iH}$ is the number of occurrences of a repeating substring $s_{i1}s_{i2}\cdots s_{i\tau_i}$ within r_i, where τ_i is the period of r_i and is given by

$$\phi_i = \frac{H}{\tau_i}. \tag{13}$$

(b) If string r_i has multiple periods, then it also has *multiple* frequencies.

(c) If string r_i is *nonperiodic*, that is, $\tau_i = H$, then its frequency $\phi_i = 1$.

Table 1: The number of wire arrangements, $|A(W, F, S)|$, for $W = 16$ and all possible F and S. The results obtained using the set of (9) are shown in boldface. They exactly coincide with the results from a *brute-force algorithm* implemented in the MATLAB computing environment (Section 5.1).

	$S=0$	$S=1$	$S=2$	$S=3$	$S=4$	$S=5$	$S=6$	$S=7$	$S=8$	$S=9$	$S=10$	$S=11$	$S=12$	$S=13$	$S=14$	$S=15$	$S=16$
$F=0$	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$F=1$	0	**16**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$F=2$	0	104	**16**	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$F=3$	0	352	**192**	**16**	0	0	0	0	0	0	0	0	0	0	0	0	0
$F=4$	0	660	968	**176**	**16**	0	0	0	0	0	0	0	0	0	0	0	0
$F=5$	0	672	2640	**880**	**160**	**16**	0	0	0	0	0	0	0	0	0	0	0
$F=6$	0	336	4224	2568	**720**	**144**	**16**	0	0	0	0	0	0	0	0	0	0
$F=7$	0	64	4032	4704	**1920**	**576**	**128**	**16**	0	0	0	0	0	0	0	0	0
$F=8$	0	**2**	2212	5432	3304	**1344**	**448**	**112**	**16**	0	0	0	0	0	0	0	0
$F=9$	0	0	608	3776	3696	**2016**	**896**	**336**	**96**	**16**	0	0	0	0	0	0	0
$F=10$	0	0	56	1400	2560	1976	**1120**	**560**	**240**	**80**	**16**	0	0	0	0	0	0
$F=11$	0	0	0	208	960	1184	**896**	**560**	**320**	**160**	**64**	**16**	0	0	0	0	0
$F=12$	0	0	0	**4**	136	360	424	**336**	**240**	**160**	**96**	**48**	**16**	0	0	0	0
$F=13$	0	0	0	0	0	32	80	**112**	**96**	**80**	**64**	**48**	**32**	**16**	0	0	0
$F=14$	0	0	0	0	0	0	0	**8**	**16**	**16**	**16**	**16**	**16**	**16**	**16**	0	0
$F=15$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**16**	0
$F=16$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

(d) The *frequency* f_i *of the wire arrangement* $a_i \in A(W, F, S)$ is the number of occurrences of a repeating subarrangement within a_i and is given by

$$f_i = \frac{W}{t_i}. \tag{14}$$

Clearly, using (12)–(14), the respective frequencies f_i and ϕ_i of the wire arrangement $a_i \in A(W, F, S)$ and the corresponding string r_i are equal; that is, $f_i = \phi_i$.

Example 6. The wire arrangement $\times\times\times\bigcirc\bigcirc\bigcirc\times\times\times\bigcirc\bigcirc\bigcirc \in A(12, 6, 3)$ has a period of $t = 6$ and a frequency of $f = W/t = 2$, while the corresponding string $r = s_1s_2s_3s_4s_5s_6 = 300300$ is of period $\tau = 3$ and of frequency $\phi = H/\tau = 2$.

Note that due to the convention in the definition of string r_i (refer to (11)), one string r_i corresponds to t_i equivalent rotations of wire arrangements (refer to Example 7). If string r_i is nonperiodic, that is, $\tau_i = H$, then by (10) and (12) the number of equivalent rotations of wire arrangements is

$$t_i = \sum_{j=1}^{\tau_i} s_{ij} + \tau_i = F + H = W. \tag{15}$$

Moreover, substituting (12) into (11) yields

$$|A(W, F, S)| = \sum_{i=1}^{n} t_i = \sum_{i=1}^{n} \left(\sum_{j=1}^{\tau_i} s_{ij} + \tau_i \right) = \sum_{i=1}^{n} \sum_{j=1}^{\tau_i} s_{ij} + n\tau_i. \tag{16}$$

Example 7. Let $W = 10$, $F = 6$, and $S = 3$. Then, $H = W - F = 4$. A nonperiodic string $r = 3300$ corresponds to $t = W = 10$ equivalent rotations of wire arrangements, demonstrated as follows:

$$\begin{gathered}
\times\times\times\bigcirc\times\times\times\bigcirc\bigcirc\bigcirc, \\
\bigcirc\times\times\times\bigcirc\times\times\times\bigcirc\bigcirc, \\
\bigcirc\bigcirc\times\times\times\bigcirc\times\times\times\bigcirc, \\
\bigcirc\bigcirc\bigcirc\times\times\times\bigcirc\times\times\times, \\
\times\bigcirc\bigcirc\bigcirc\times\times\times\bigcirc\times\times, \\
\times\times\bigcirc\bigcirc\bigcirc\times\times\times\bigcirc\times, \\
\times\times\times\bigcirc\bigcirc\bigcirc\times\times\times\bigcirc, \\
\bigcirc\times\times\times\bigcirc\bigcirc\bigcirc\times\times\times, \\
\times\bigcirc\times\times\times\bigcirc\bigcirc\bigcirc\times\times, \\
\times\times\bigcirc\times\times\times\bigcirc\bigcirc\bigcirc\times.
\end{gathered} \tag{17}$$

However, the periodic string $r = 3030$ with $\tau = 2$ corresponds to only $t = s_1 + s_2 + \tau = 3 + 0 + 2 = 5$ equivalent rotations of wire arrangements:

$$\begin{gathered}
\times\times\times\bigcirc\bigcirc\times\times\times\bigcirc\bigcirc, \\
\bigcirc\times\times\times\bigcirc\bigcirc\times\times\times\bigcirc, \\
\bigcirc\bigcirc\times\times\times\bigcirc\bigcirc\times\times\times, \\
\times\bigcirc\bigcirc\times\times\times\bigcirc\bigcirc\times\times, \\
\times\times\bigcirc\bigcirc\times\times\times\bigcirc\bigcirc\times.
\end{gathered} \tag{18}$$

Clearly, the introduction of the notion of the string r_i, with its one-to-one correspondence to subset A_i, as explained

above, has reduced the current problem to finding all possible such sets A_i, for all $i = 1, 2, \ldots, n$, and their cardinalities, which are nothing else than the periods t_i (related to periods τ_i of the strings r_i through (13) and (14)) of wire arrangements a_i in A_i.

4.6. Partitioning of the Number of Faulty Wires and Corresponding Necklaces. We use *integer partitions* in order to find all string representations of all wire arrangements.

Definition 8. A *k-partition* p of a positive integer n is a partition consisting of exactly k terms, adding zeros whenever necessary.

Returning to the presented problem for a parallel link arrangement of W wires, with F faulty wires and the largest fault-segment S, an *H-partition* p of (integer) F consists of H (number of healthy wires) terms, with the largest term being equal to S (refer to Example 9).

Example 9. Let $W = 15$, $F = 9$, and $S = 3$. Then, $H = W - F = 6$. The 6-partitions of $F = 9$, with the largest term being equal to $S = 3$, are given as follows:

$$\begin{aligned} &3+3+3+0+0+0,\\ &3+3+2+1+0+0,\\ &3+3+1+1+1+0,\\ &3+2+2+2+0+0,\\ &3+2+2+1+1+0,\\ &3+2+1+1+1+1. \end{aligned} \tag{19}$$

Each partition p defines a set of strings, which are given by specific permutations of p's characters. For instance, $p = 3+3+3+0+0+0$ corresponds to a set of strings, namely, $r_1 = 333000$, $r_2 = 330030$, $r_3 = 303030$, and $r_4 = 300330$. Hence, still, knowing the actual H-partitions corresponding to given W, F, and S does not solve the problem, as the number of strings per partition must be found. This can be achieved by noting that the number of all possible strings r_i with nonintersecting sets of equivalent rotations of wire arrangements can be represented by the number of *necklaces* for each partition p (refer to Example 11 for illustration). We recall the definition of a necklace as follows.

Definition 10. A *K-ary necklace* of length n is an equivalence class of n-character strings over an alphabet of size K, taking all rotations as equivalent [33].

Example 11. Let $W = 8$, $F = 5$, and $S = 4$. Then, $H = 3$. It turns out that there is only one 3-partition of $F = 5$, with the largest term being equal to $S = 4$; namely,

$$p = 4 + 1 + 0. \tag{20}$$

All necklaces for the 3-partition above, with the corresponding equivalent rotations of wire arrangements, are

$$\begin{aligned} 4+1+0:\ &\times\times\times\times\bigcirc\times\bigcirc\bigcirc, \quad \bigcirc\times\times\times\times\bigcirc\times\bigcirc,\\ &\bigcirc\bigcirc\times\times\times\times\bigcirc\times, \quad \times\bigcirc\bigcirc\times\times\times\times\bigcirc,\\ &\bigcirc\times\bigcirc\bigcirc\times\times\times\times, \quad \times\bigcirc\times\bigcirc\bigcirc\times\times\times,\\ &\times\times\bigcirc\times\bigcirc\bigcirc\times\times, \quad \times\times\times\bigcirc\times\bigcirc\bigcirc\times,\\ 4+0+1:\ &\times\times\times\times\bigcirc\bigcirc\times\bigcirc, \quad \bigcirc\times\times\times\times\bigcirc\bigcirc\times,\\ &\times\bigcirc\times\times\times\times\bigcirc\bigcirc, \quad \bigcirc\times\bigcirc\times\times\times\times\bigcirc,\\ &\bigcirc\bigcirc\times\bigcirc\times\times\times\times, \quad \times\bigcirc\bigcirc\times\bigcirc\times\times\times,\\ &\times\times\bigcirc\bigcirc\times\bigcirc\times\times, \quad \times\times\times\bigcirc\bigcirc\times\bigcirc\times. \end{aligned} \tag{21}$$

Note that there is a one-to-one correspondence between the necklaces above and (all possible, for this case) strings r_i, whose corresponding sets of equivalent rotations of wire arrangements do not intersect.

For each H-partition one can compute the corresponding number of necklaces (refer to (23)), which in turn can be used to compute the number of wire arrangements. Hence, the problem reduces to finding (a) all such partitions p, as described above, and, subsequently, (b) their corresponding number of necklaces.

There are a number of known algorithms that can actually generate such a list of H-partitions in a *constant amortized time* [34]. Note here that a partition can be extended to an H-partition by simply adding the necessary number of zeros.

An alternative way to approach the problem of finding the required partitions is by defining a string $\eta_i = n_0 n_1 n_2 \cdots n_S$, where n_k is equal to the number of occurrences of integer k in a string r_i. Then, from the way the strings r_i and η_i are constructed and from (6) and (10), we set the following constrained system of equations:

$$\begin{aligned} &\sum_{k=1}^{S} (S+1-j)\, n_k = F,\\ &n_0 + \sum_{k=1}^{S} n_k = H,\\ &\sum_{k=1}^{S} n_k \in \left[\left\lceil \frac{F}{S} \right\rceil, \min(F-S+1, H) \right],\\ &\qquad n_k \in Z_0^+, \quad n_1 \neq 0. \end{aligned} \tag{22}$$

Let us now introduce a new string $\eta_i^* = n_l n_m \cdots n_S$ that arises by excluding any zero terms from string η_i. Let $K \leq S$ be the number of nonzero terms in string η_i; we can find

the number of K-ary necklaces for the corresponding *original* string r_i as follows:

$$N_i(n_l, n_m, \ldots, n_S) = \frac{1}{n} \sum_{j \mid \gcd(n_l, n_m, \ldots, n_S)} \varphi(j) \left(\frac{(n/j)!}{(n_l/j)!\,(n_m/j)! \cdots (n_S/j)!} \right), \tag{23}$$

where $N_i(n_l, n_m, \ldots, n_S)$ denotes the number of K-ary necklaces composed of n_j occurrences of $j = l, m, \ldots, S$, $n = \sum_{j=l}^{S} n_j = H$, and $\varphi(j) = j\prod_{q|j}(1 - 1/q)$ is *Euler's totient function*, defined as a number of positive integers less than or equal to j that are coprime to j [35, 36].

However, (23) counts all necklaces corresponding to strings r_i (refer to Example 7 for explanation), whether the latter are periodic or not. Clearly one should distinguish between periodic and nonperiodic strings. In order to do so, we simply consider all periods t_i (and corresponding frequencies f_i) of the set $A(W, F, S)$, such that $F/f_i \geq S$, and compute all aperiodic necklaces (or *Lyndon words*) of sets $A(W/f_i, F/f_i, S)$ for each f_i (including $f_1 = 1$) separately, according to the following formula [35]:

$$L_i(n_l, n_m, \ldots, n_S) = \frac{1}{n} \sum_{j \mid \gcd(n_l, n_m, \ldots, n_S)} \mu(j) \left(\frac{(n/j)!}{(n_l/j)!\,(n_m/j)! \cdots (n_S/j)!} \right), \tag{24}$$

where

$$n = \sum_{j=l}^{S} n_j = H,$$

$$\mu(j) = \begin{cases} 0, & \text{if } j \text{ has one or more repeated prime factors,} \\ 1, & \text{if } j = 1, \\ (-1)^k, & \text{if } j \text{ is a product of } k \text{ distinct primes} \end{cases} \tag{25}$$

is the *Moebius function*.

4.7. Full Model for the Probability Distribution. Following the analysis above, it is not difficult to derive the following equation from (11) and (24) that yields the desired number of wire arrangements (once the number of H-partitions, strings, i.e., necklaces, and frequencies are known):

$$|A(W, F, S)| = \sum_i t_i \sum_j L_{ij}, \tag{26}$$

where i denotes the index for each of the Lyndon words corresponding to the H-partitions of F and j the index of the Lyndon word that corresponds to the H-partition. The probability distribution $P_W(S \mid F)$ for all S (in the appropriate domain as given in (4)) is simply given by

$$P_W(S \mid F) = \frac{|A(W, F, S)|}{|A(W, F)|} = \frac{\sum_i t_i \sum_j L_{ij}}{\binom{W}{F}}. \tag{27}$$

5. Demonstration of the Effectiveness of the Derived Model and Results

The derived algorithm in Section 4 can be summarized as follows.

Analytical Algorithm to Measure the Number of Clustering Fault-Segments Based on Combinatorial Arguments

Scope. Generate all F-combinations $f_{F-1}, \ldots, f_1, f_0$ of W numbers $\{0, 1, \ldots, W - 1\}$, given that $W \geq F \geq 0$. Measure the number of S fault-segment sizes such as $0 < S \leq W$ and $S \leq W$.

Label 1 (input). Set W, F, and S.

Label 2 (compute periods and frequencies). Find all common factors of W and F (frequencies f_i, $i = 1 : n$), $F/f_i \geq S$ (t_i period, $t_i f_i = W$).

Label 3 (obtain i reduced problems). Set $W_i = W/f_i$, $F_i = F/f_i$, S, and $H_i = W_i - F_i$.

Label 4 (obtain full lists of the partitions of F_i with S being their maximum element present). Call available routines.

Label 5 (construct η_j-string for each H_i-partition). $\eta_j = n_0 n_1 n_2 \cdots n_S$, where n_k ($k = 0 : S$) is the number of occurrences of digit $S - k$ in the partition.

Label 6 (construct η_j^*-string and compute its Lyndon words). $\eta_j^* = n_l n_m \cdots n_S$, where n_k ($k = l : S$) are the nonzero elements of corresponding string η_j and then $L(\eta_j^*)$ the Lyndon words are computed by calling (24).

Label 7 (compute the number of wire arrangements) $|A(W, F, S)|$ is computed by calling (26).

We demonstrate the applicability and, consequently, the effectiveness of the derived model using the following parameters that were chosen at random (relatively large numbers have been picked to show both the efficiency and the accuracy of the derived model).

Let $W = 20$, let $F = 12$, and let $S = 3$. Hence, $H = 8$.

The wire arrangements will have the following periods and corresponding frequencies (all common factors of W and F, such that $F/f \geq S$):

$(20 = W = t_i f_i)$: $f_1 = 1, t_1 = 20$; $f_2 = 2, t_2 = 10$; $f_3 = 4, t_3 = 5$.

Let us consider all three pairs of f_i and t_i one after another.

(i) $A(20/1, 12/1, 3) = A(20, 12, 3)$ with $H_1 = 20 - 12 = 8$ (see Table 2).

Table 2

H_1-partitions of F_1	η_1-string	$L_1(n_i^*)$
3 + 3 + 3 + 3 + 0 + 0 + 0 + 0	4004 (i.e., four 3s, zero 2s, zero 1s, four 0s)	$L(4,4) = 8$
3 + 3 + 3 + 2 + 1 + 0 + 0 + 0	3113	$L(3,1,1,3) = 140$
3 + 3 + 3 + 1 + 1 + 1 + 0 + 0	3032	$L(3,3,2) = 70$
3 + 3 + 2 + 2 + 2 + 0 + 0 + 0	2303	$L(2,3,3) = 70$
3 + 3 + 2 + 2 + 1 + 1 + 0 + 0	2222	$L(2,2,2,2) = 312$
3 + 3 + 2 + 1 + 1 + 1 + 1 + 0	2141	$L(2,1,4,1) = 105$
3 + 3 + 1 + 1 + 1 + 1 + 1 + 1	2060	$L(2,6) = 3$
3 + 2 + 2 + 2 + 2 + 1 + 0 + 0	1412	$L(1,4,1,2) = 105$
3 + 2 + 2 + 2 + 1 + 1 + 1 + 0	1331	$L(1,3,3,1) = 140$
3 + 2 + 2 + 1 + 1 + 1 + 1 + 1	1250	$L(1,2,5) = 21$
		$\sum_j L_{1j}(n_1^*) = 974$

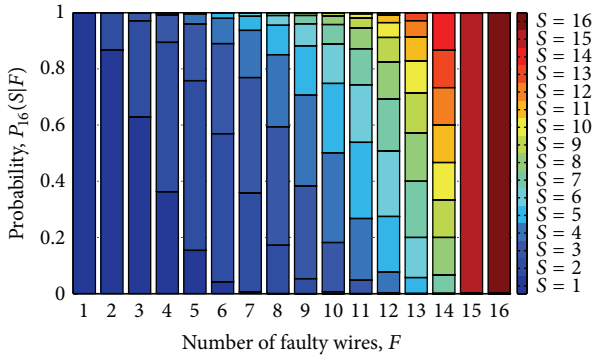

Figure 2: Probability distribution $P_W(S \mid F)$ with a total of $W = 16$ parallel wires including F faulty wires, where S is the size of the largest fault-segment.

(ii) $A(20/2, 12/2, 3) = A(10, 6, 3)$ with $H_2 = 10 - 6 = 4$ (see Table 3).

(iii) $A(20/4, 12/4, 3) = A(5, 3, 3)$ with $H_3 = 5 - 3 = 2$ (see Table 4).

As a result, the total number of wire arrangements, according to (26), is

$$|A(20, 12, 3)| = 20(974) + 10(8) + 5(1) = 19565 \quad (28)$$

and the desired probability (from (27)) is

$$P_{20}(3 \mid 12) = \frac{|A(20, 12, 3)|}{\binom{20}{12}} = \frac{19565}{125970}. \quad (29)$$

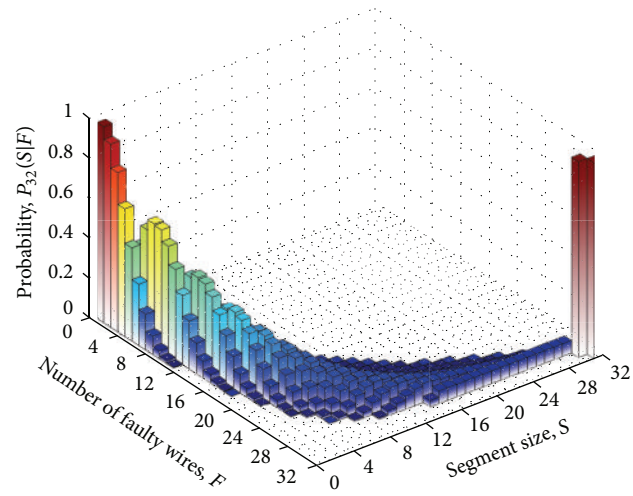

Figure 3: Three-dimensional view of probability distribution $P_W(S \mid F)$ with a total of $W = 32$ parallel wires including F faulty wires, where S is the size of the largest fault-segment.

Table 3

H_2-partitions of F_2	η_2-string	$L_2(n_2^*)$
3 + 3 + 0 + 0	2002	$L(2,2) = 1$
3 + 2 + 1 + 0	1111	$L(1,1,1,1) = 6$
3 + 1 + 1 + 1	1030	$L(1,3) = 1$
		$\sum_j L_{2j}(n_2^*) = 8$

Table 4

H_3-partitions of F_3	η_3-string	$L_3(\eta(n_i \neq 0))$
3 + 0	1001	$L(1,1) = 1$
		$\sum_j L_{3j}(n_3^*) = 1$

Figures 2 and 3 show the distribution of the corresponding probabilities $P_W(S \mid F)$ for various values of W obtained by (27). Note that since in these two figures we use different link widths, with 16 wires in Figure 2 and 32 wires in Figure 3, the calculated faulty wire distributions are quite different for each other; even so, the applicability of our analytical model is effectively demonstrated here. Although a 2D simplified figure is in general desirable, Figure 3 is constructed in 3D so as to provide better visual resolution in demonstrating the range of results. All the demonstrated cases have been numerically tested and verified by the results obtained from a brute-force algorithm implemented in MATLAB (Section 5.1).

5.1. Verification of Mathematical Model for Correctness Using a Brute-Force Algorithm. A brute-force algorithm, implemented using the MATLAB computing environment, is utilized to confirm and verify our presented mathematical model for any possible NoC parallel link length encompassing any number of erroneous (unhealthy) wires. This brute-force algorithm generates all combinations of F number of faulty wires given a W-length parallel link (composed of a W number of parallel wires), where $F \leq W$ is based on a simple

sequential lexicographical ascending order algorithm, which is a convenient way to generate combinatorial combinations [34].

Essentially, the algorithm generates the $C(F, W)$ combinations of F objects, denoted by the set $\{f_{F-1}, \ldots, f_1, f_0\}$, chosen from the set of $\{w_{W-1}, \ldots, w_1, w_0\}$ wire objects such that $\{w_{F-1}, \ldots, w_1, w_0\} \subseteq \{w_{W-1}, \ldots, w_1, w_0\}$. Note that there is no need to follow an ascending order, as the C combinations need not be ordered (since we are not interested in sorting the C number of combinations); the aim is to cover all combinations, and the ascending order of the lexicographical algorithm used in our brute-force approach ensures that all combinations are accounted for and covered and also that no combination cases are double-generated (due to symmetry) or repeated. For every combination generated, the number of consecutive objects that represent faulty wire elements is then accounted for; with all sizes of faulty wire clustering cases accumulated, the brute-force results are then compared to the results calculated using our mathematical model to determine its systematic accuracy, under any respective W and F values.

Each of the $C(F, W) = W!/(F!(W - F)!)$ combinations requires $O(F)$ time to be produced as an output, and hence the sequential algorithm runs in $O(\sum_{i=1}^{i=W}(F_i \times C(F_i, W)))$ time to produce, and not just generate, all possible combinations. When F is half the size of W, the worst-case scenario of producing all combinations of $C(F, W)$ is met. Additionally, as W grows linearly, an exponentially increasing number of iterations, and hence time, are required to output all lexicographical combinations, as presented in Section 5.2. Note, though, that there exist some optimized lexicographical algorithms (beyond the scope of the current work), which can generate these combinations using smaller data structures and hence require reduced memory space in computing, to hold combinations such as [37, 38].

To generate $C(F, W)$ combinations, we use binary strings with F objects having a binary value of one to denote the positions of the faulty wires in the W-length wire (while $W-F$ are zeros), while the remaining objects of the set W have a binary value of zero. These zero-value objects, denoted by the set $\{h_{W-F-1}, \ldots, h_1, h_0\}$ represent the healthy objects (or wires) in W, such that $W = F + H$ or $\{w_{W-1}, \ldots, w_1, w_0\} = \{f_{F-1}, \ldots, f_1, f_0\} \cup \{h_{W-F-1}, \ldots, h_1, h_0\}$. As mentioned above, the F-combination brute-force approach is based on the well-known lexicographical order algorithm presented in Knuth's seminal book [34], with the addition of inserting a subprocedure which measures the number of faulty wires clustering fault-segments, under each generated combination, required to compare against our mathematical model. Without loss of generality with regard to other options, the brute-force algorithm is presented as follows.

Brute-Force Algorithm to Measure the Number of Clustering Fault-Segments Based on a Sequential Lexicographical Ascending Order Algorithm Adopted from Knuth's Seminal Book [34] with Suitable and Relevant Alterations

Scope. Generate all F-combinations $f_{F-1}, \ldots, f_1, f_0$ of W numbers $\{0, 1, \ldots, W - 1\}$, given that $W \geq F \geq 0$. Measure the number of S fault-segment sizes such as $0 < S \leq W$ and $S \leq W$, where any $f_F, \ldots, f_2, f_1$ may have a consecutive index with its neighbor objects(s). Additional f_{F+1} and f_{F+2} are used as sentinels.

Label 1 (initialize). Set $f_j \leftarrow j - 1$ for $1 \leq j \leq W$; also set $f_{F+1} \leftarrow W$ and $f_{F+2} \leftarrow 0$.

Label 2 (visit combination). Visit the combination $f_F, \ldots, f_2, f_1$.

Label 3 (count fault-segments). Measure and bookkeep the size and number of consecutive object members (i.e., faulty wires).

Label 4 (find j). Set $j \leftarrow 1$. Then while $f_j + 1 = f_{j+1}$, set $f_j \leftarrow j-1$ and $j \leftarrow j+1$; eventually the condition $f_j+1 \neq f_{j+1}$ will occur.

Label 5 (done?). Terminate the algorithm if $j > W$.

Label 6 (increase f_j). Set $f_j \leftarrow f_j + 1$ and return to Label 2.

5.2. Brute-Force Algorithm versus Analytical Method Compute Costs. To demonstrate the effectiveness of our proposed analytical method, we have run a complete set of experiments for NoC parallel links containing up to 128 links, which is a typical wire bit-width in today's chip multiprocessors [1, 2]. Both methodologies (analytical and brute-force) exhibit an almost perfect exponential relationship of compute time versus the number of parallel NoC wires; however, the analytical model is several orders of magnitude faster and hence more efficient as the number of wires in a link increases, which makes it a desirable choice when designers need to compute the distribution of faults for wider links. In particular, the brute-force is about $e^{0.5W-8}$ slower than its analytical counterpart. This means that for $W = 16$ the two algorithms spend comparable times, while, for example, for $W = 32$, the brute force is three orders of magnitude slower than the analytical algorithm, and for $W = 48$ it is eight orders of magnitude slower, and so forth.

6. Applicability of the Combinatorial Algorithm

Our combinatorial algorithm presented in this paper which calculates the distribution of faults clustering is also relevant to other studies or applications where fault clustering, that is, consecutive faults that may lie in a circular or ring topological arrangement, need to be estimated and calculated for in order to asses risk and reliability and other parameters of interest pertaining to system or object resilience. Related applications of our mathematical model include the reliability and wearout assessment of adjoining parallel high-strength wires of suspension bridge cables [39, 40], where high axial tensile stresses in tandem with the surrounding corrosive environment accelerate corrosion and embrittlement causing cables to deteriorate and eventually fail over time. Another application of our model is to aid the assessment of

the psychological/death chance cost of Russian roulette, both as a glorified game of ultimate risk, where a person spins the cylinder of a revolver that contains a single bullet and aims it at its head, and a tool for suicide [41]. Our model can further be used to calculate the chance of drawing consecutive numbers in a standard roulette game and to derive the probability of having consecutive passenger cabins in a Ferris wheel being occupied (or failing). Next, the model can be used to calculate the recovery hardware cost of adjacent channel/link/node failures in optical ring networks [42] and to assess the structural reliability of consecutive gear teeth due to their exposure to continuous stresses which reduce their fatigue strength in a spur gear [27]. Finally, another application of our combinatorial model is to estimate the chance of consecutive spokes in a wheel, regarded as a disk of uniform stiffness per length of circumference, failing due to their exposure to relentless stresses caused by high radial loads experienced in real-world conditions [43], among numerous other applications.

7. Conclusions

Networks-on-chips (NoCs) are critical on-chip communication subsystems that transfer packetized messages among the various computational tiles in today's ultrahigh performance general-purpose multicore chips such as chip multiprocessors (CMPs). These have been realized due to the continuous miniaturization of CMOS transistors which have enabled the massive integration of transistors on a single chip, exceeding the billion-transistor mark in today's CMOS process technologies. This progress, unfortunately, has also come at a cost of increased susceptibility to wearout and permanent failures. Parallel on-chip links are particularly prone to the effects of electromigration which can cause eventual permanent breakdown in links, manifesting to protocol-level deadlocks and indefinite CMP stalls, rendering the chip inoperable. Realizing the importance of link failure models in NoCs, this paper derived and demonstrated a combinatorial algorithm that can be used to calculate the spatial probability distribution of wire faults in a parallel NoC interconnect link given its width and a given number of faulty wires, which can appear in this link. Particular emphasis was paid upon the adjacency of the faulty wires that form fault-segments separated by at least one healthy wire, as the size of the largest segment determines the additional delay required by partially faulty link recovery mechanisms, such those of [23–25], to recover corrupted flit data at the receiver router. The developed nearly full analytical model constitutes an application of partitions and necklaces through a systematic approach that derives the correspondence between the presented problem and necklaces, where periodicity plays a crucial role. The model is completely verified by and is far superior with regard to extensive brute-force numerical simulations.

Finally, it is worth mentioning that the resulting formula of the presented algorithm can, *mutatis mutandis*, serve as a prototype for applications in the various "isomorphic" problems from other disciplines, areas, or frameworks, as discussed in Section 6 [39–44].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the Cyprus Research Promotion Foundation's Framework Programme for Research, Technological Development and Innovation 2009-10 (ΔΕΣΜΗ 2009-10), cofunded by the Republic of Cyprus and the European Regional Development Fund, and specifically under Grant no. ΔΙΕΘΝΗΣ/ΣΤΟΧΟΣ/0311/06.

References

[1] S. R. Vangal, J. Howard, G. Ruhl et al., "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, 2008.

[2] S. Bell, B. Edwards, J. Amann et al., "TILE64 processor: a 64-core SoC with mesh interconnect," in *Proceedings of the IEEE International Solid State Circuits Conference*, pp. 588–598, February 2008.

[3] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, pp. 71–121, 2006.

[4] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, 2009.

[5] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayashima, S. W. Keckler, and L.-S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, 2007.

[6] W. J. Dally and B. Towles, "Route packets not wires: on-chip interconnection networks," in *Proceedings of the IEEE Design Automation Conference*, pp. 684–689, May 2001.

[7] D. Bertozzi and L. Benini, "Xpipes: a network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits and Systems Magazine*, vol. 4, no. 2, pp. 18–31, 2004.

[8] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann, Boston, Mass, USA, 2002.

[9] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2004.

[10] J. Lü, X. Yu, G. Chen, and D. Cheng, "Characterizing the synchronizability of small-world dynamical networks," *IEEE Transactions on Circuits and Systems. I. Regular Papers*, vol. 51, no. 4, pp. 787–796, 2004.

[11] J. Zhou, J.-A. Lu, and J. Lü, "Pinning adaptive synchronization of a general complex dynamical network," *Automatica*, vol. 44, no. 4, pp. 996–1003, 2008.

[12] J. Lü and G. Chen, "A time-varying complex dynamical network model and its controlled synchronization criteria," *IEEE Transactions on Automatic Control*, vol. 50, no. 6, pp. 841–846, 2005.

[13] ITRS International Technology Roadmap for Semiconductors, *Process Integration, Devices, and Structures (PIDS)*, 2009.

[14] A. Morgenshtein, I. Cidon, A. Kolodny, and R. Ginosar, "Comparative analysis of serial vs parallel links in NOC," in *Proceedings of the International Symposium on System-on-Chip*, pp. 185–188, November 2004.

[15] K. Constantinides, S. Plaza, J. Blome et al., "BulletProof: a defect-tolerant CMP switch architecture," in *Proceedings of the International Symposium on High-Performance Computer Architecture*, pp. 5–16, February 2006.

[16] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, 2005.

[17] G. de Micheli and L. Benini, *Networks on Chips: Technology and Tools (Systems on Silicon)*, Morgan Kaufmann, Boston, Mass, USA, 2006.

[18] T. Lehtonen, D. Wolpert, P. Liljeberg, J. Plosila, and P. Ampadu, "Self-adaptive system for addressing permanent errors in on-chip interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 4, pp. 527–540, 2010.

[19] S. R. Nassif, N. Mehta, and C. Yu, "A resilience roadmap," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '10)*, pp. 1011–1016, March 2010.

[20] J. Duato, "A theory of fault-tolerant routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 8, pp. 790–802, 1997.

[21] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Computing Surveys*, vol. 46, no. 1, article 8, 2013.

[22] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, article 51, 2006.

[23] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos, "A fine-grained link-level fault-tolerant mechanism for networks-on-chip," in *Proceedings of the 28th IEEE International Conference on Computer Design (ICCD '10)*, pp. 447–454, Amsterdam, The Netherlands, October 2010.

[24] A. Vitkovskiy, V. Soteriou, and C. Nicopoulos, "A dynamically adjusting gracefully degrading link-level fault-tolerant mechanism for NoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 8, pp. 1235–1248, 2012.

[25] M. Palesi, S. Kumar, and V. Catania, "Leveraging partially faulty links usage for enhancing yield and performance in networks-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 3, pp. 426–440, 2010.

[26] A. Vitkovskiy, P. Christodoulides, and V. Soteriou, "A combinatorial application of necklaces: modeling individual link failures in parallel network-on-chip interconnect links," in *Proceedings of the World Congress on Engineering, London, UK, July 2012*, Lecture Notes in Engineering and Computer Science, pp. 125–130, Cyprus University of Technology, 2012.

[27] R. C. Lyndon, "On Burnside's problem," *Transactions of the American Mathematical Society*, vol. 77, pp. 202–215, 1954.

[28] L. Pebody, "Reconstructing odd necklaces," *Combinatorics, Probability and Computing*, vol. 16, no. 4, pp. 503–514, 2007.

[29] N. Alon, "Splitting necklaces," *Advances in Mathematics*, vol. 63, no. 3, pp. 247–253, 1987.

[30] S. W. Golomb, "Combinatorial proof of Fermat's 'little' theorem," *The American Mathematical Monthly*, vol. 63, no. 10, p. 718, 1956.

[31] P. Erdos and A. Renyi, "On random graphs," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.

[32] M. E. J. Newman, "Random graphs as models of networks," in *Handbook of Graphs and Networks*, S. Bornholdt and H. G. Schuster, Eds., pp. 35–68, Wiley-VCH, Berlin, Germany, 2003.

[33] E. W. Weisstein, "Necklace," MathWorld—A Wolfram Web Resource, http://mathworld.wolfram.com/Necklace.html.

[34] D. E. Knuth, *The Art of Computer Programming: Vol. 4: A Combinatorial Algorithms, Part 1*, Addison-Wesley, Upper Saddle River, NJ, USA, 2011.

[35] The Object Server, "Necklaces, Unlabelled Necklaces, Lyndon Words, De Bruijn Sequences," http://www.theory.cs.uvic.ca/~cos/inf/neck/NecklaceInfo.html.

[36] E. W. Weisstein, "Totient function," MathWorld—A Wolfram Web Resource, http://mathworld.wolfram.com/TotientFunction.html.

[37] J. Castro-Gutierrez, D. Landa-Silva, and J. Moreno Perez, "Improved dynamic lexicographic ordering for multi-objective optimisation," in *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, pp. 31–40, September 2010.

[38] J.-E. Martínez-Legaz, "Lexicographical order, inequality systems and optimization," in *System Modelling and Optimization*, vol. 59 of *Lecture Notes in Control and Information Sciences*, pp. 203–212, Springer, Berlin, Germany, 1984.

[39] R. Betti, M. Asce, A. C. West, G. Vermaas, and Y. Cao, "Corrosion and embrittlement in high-strength wires of suspension bridge cables," *Journal of Bridge Engineering*, vol. 10, no. 2, pp. 151–162, 2005.

[40] R. M. Mayrbaurl and S. Camo, "Cracking and fracture of suspension bridge wire," *Journal of Bridge Engineering*, vol. 6, no. 6, pp. 645–650, 2001.

[41] D. A. Fishbain, J. R. Fletcher, T. E. Aldrich, and J. H. Davis, "Relationship between Russian roulette deaths and risk-taking behavior: a controlled study," *American Journal of Psychiatry*, vol. 144, no. 5, pp. 563–567, 1987.

[42] X. Q. Peng, L. Geng, W. Liyan, G. R. Liu, and K. Y. Lam, "A stochastic finite element method for fatigue reliability analysis of gear teeth subjected to bending," *Computational Mechanics*, vol. 21, no. 3, pp. 253–261, 1998.

[43] H. P. Gavin, "Bicycle-wheel spoke patterns and spoke fatigue," *Journal of Engineering Mechanics*, vol. 122, no. 8, pp. 736–742, 1996.

[44] O. Gerstel, R. Ramaswami, and G. H. Sasaki, "Fault tolerant multiwavelength optical rings with limited wavelength conversion," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 7, pp. 1166–1178, 1998.