An ROBDD-Based Combinatorial Method for the Evaluation of Yield of Defect-Tolerant Systems-on-Chip

Juan A. Carrasco, *Senior Member, IEEE*, and Víctor Suñé

***Abstract*—In this paper, we develop a combinatorial method for the evaluation of the functional yield of defect-tolerant systems-on-chip (SoC). The method assumes that random manufacturing defects are produced according to a model in which defects cause the failure of given components of the system following a distribution common to all defects. The distribution of the number of defects is arbitrary. The yield is obtained by conditioning on the number of defects that result in the failure of some component and performing recursive computations over a reduced ordered binary decision diagram (ROBDD) representation of the fault-tree function of the system. The method has excellent error control. Numerical experiments seem to indicate that the method is efficient and, with some exceptions, allows the analysis with affordable computational resources of systems with very large numbers of components.**

***Index Terms*—Combinatorial method, defect-tolerant systems-on-chip (SoC), manufacturing defects, reduced ordered binary decision diagram (ROBDD), yield.**

I. Introduction

SYSTEMS-ON-CHIP (SoCs) represent a rapidly growing field in the electronic and computer industry [1]. Applications include wireless systems [2], 3-D graphics systems [3], reconfigurable systems [4], and others. The high densities and areas of those integrated systems make them very susceptible to random manufacturing defects [5]. In fact, complex SoCs are likely to have a very small functional yield if they are not designed with built-in defect-tolerance [6]. Here, we take the usual definition of functional yield as the probability that a system without parametric faults will not have some random defect preventing the system from functioning properly. Clearly, there is a need for efficient methodologies for estimating the functional yield of complex defect-tolerant SoCs. When the defect-tolerant SoC has a simple structure, it is often possible to make *ad hoc* evaluations (see, for instance, [7]–[9]). However, given the trend towards the use of a sophisticated network-on-chip as communication subsystem among the intellectual property cores (IPs) of the SoC [10]–[12], it is foreseeable that many defect-tolerant designs will not have such a simple structure. Evaluating the yield of those defect-tolerant SoCs is far from being a trivial task, mainly because of the fact that realistic defect models have clustering [13] and, thus, introduce dependencies among the failed states of the components of the system (see, for instance, [9] and [14]). A combinatorial method for the evaluation of the functional yield of defect-tolerant SoCs has already been developed [15]. However, the computational cost of that method is relatively high and the method seems to be able to handle only defect-tolerant SoCs with up to a few tens of components. The aim of this paper is to develop a more efficient combinatorial method for the evaluation of the functional yield of defect-tolerant SoCs which can handle much more complex systems.

We will assume that the defect-tolerant SoC is made up of a set $\{1,2,\ldots,C\}$ of components and that whether the system is functioning or not is determined from the failed states of the components through a fault-tree function $F(x_1,\ldots,x_C)$, where variable x_i takes value 1 if and only if component i is failed and the function takes value 1 if and only if the system is not functioning. We will exclude the trivial cases $F(x_1,\ldots,x_C)=0$ and $F(x_1,\ldots,x_C)=1$. It will be also assumed that a gate-level description of the fault-tree function is available.

The production of defects will be modeled using two sets of probabilities: the probabilities Q_k, $k=0,1,2,\ldots$ that the number of random manufacturing defects in the area occupied by the system is k, and the probabilities $P_i>0$, $1\leq i\leq C$ that a given defect causes the failure of component i. It will be assumed that any given defect will result in the failure of any given component of the system following the probabilities P_i, $1\leq i\leq C$ independently of the number of defects, of whether the remaining defects cause a component failure or not, and of which components affect the remaining defects. That model is useful from the designer's point of view, since the distribution of the number of defects Q_k, $k=0,1,2,\ldots$ could be provided by the manufacturer of the SoC and the probabilities P_i, $1\leq i\leq C$ could be estimated as follows. Let A, A_i, and β_j be, respectively, the area of the system, the area of component i, and the probability that a given defect is of type j. Then, we can take $P_i=(A_i/A)\sum_j \beta_j\theta_{j,i}$, where $\theta_{j,i}$ [16] is the probability that any given defect of type j affecting component i causes the failure of the component and the sum is taken over all possible defect types. Each probability $\theta_{j,i}$ could be estimated from the distribution of the size of defects of type j and the layout of component i using appropriate tools [17], [18].

The assumed model is consistent with all large-area clustering compound Poisson models [9], [19]. Those models result by assuming that: 1) defect clusters are comparable in size

Manuscript received September 03, 2004; revised October 16, 2007. Current version published January 14, 2009. This work was supported by the "Comisión Internacional de Ciencia y Tecnología" (CICYT) of the Ministry of Science and Technology of Spain under the research Grant TAP1999-0443-C05-05 and by the CICYT and FEDER ("Fondo Europeo de Desarrollo Regional") under the research Grant DPI2004-05077.

The authors are with the Universitat Politècnica de Catalunya, Barcelona 08034, Spain (e-mail: carrasco@eel.upc.edu; sunye@eel.upc.edu).

Digital Object Identifier 10.1109/TVLSI.2008.2004479

with the chip; 2) the number of defects in each chip follows a Poisson distribution whose parameter is itself a random variable with some distribution; and 3) defects are uniformly distributed within the area of the chip; and include the widely-used negative binomial distribution model [20], [21].

From a computational point of view, it is convenient to map the previously described model into a model taking into account only *faults*, i.e., defects that cause the failure of some component. That model includes the probabilities

$$Q'_k = P[\text{number of faults is } k], \quad k = 0, 1, 2, \ldots$$
$$P'_i = P[\text{a given fault affects component } i], \quad 1 \le i \le C.$$

The mapping can be performed using

$$Q'_k = \sum_{j=k}^{\infty} Q_j \binom{j}{k} P_L^k (1 - P_L)^{j-k}$$
$$P'_i = \frac{P_i}{P_L}$$

where $P_L = \sum_{i=1}^{C} P_i$ is the probability that a given defect causes a fault.

As previously mentioned, the negative binomial distribution is the most widely used distribution for the number of defects in a chip. That distribution has the form

$$Q_k = \frac{\Gamma(\alpha + k)}{k!\Gamma(\alpha)} \frac{(\lambda/\alpha)^k}{(1 + \lambda/\alpha)^{\alpha+k}} \tag{1}$$

where λ is the expected number of defects and α is the clustering parameter (the clustering increases for decreasing α). It is known (see [22]) that, when the distribution of the number of defects is negative binomial, the distribution of the number of faults is also negative binomial with the same clustering parameter. More precisely, when the distribution of the number of defects is given by (1), the distribution of the number of faults is

$$Q'_k = \frac{\Gamma(\alpha + k)}{k!\Gamma(\alpha)} \frac{(\lambda'/\alpha)^k}{(1 + \lambda'/\alpha)^{\alpha+k}}$$

with $\lambda' = P_L\lambda$. A similar result holds for any large-area clustering compound Poisson model [9].

The rest of the paper is organized as follows. Section II develops and describes the method. The method will require the construction of an reduced ordered binary decision diagram (ROBDD) [23] representation of the fault-tree function of the system. Section III analyzes the computational cost of the method using scalable benchmarks and discusses how the method compares to alternatives. Finally, Section IV presents some conclusions. The Appendix includes the proofs of the results on which the method is based.

II. Method

Let Y denote the functional yield of the system and let Y_k be the functional yield conditioned on the system being affected by k faults. Using the law of total probability

$$Y = \sum_{k=0}^{\infty} Q'_k Y_k.$$

The proposed method estimates Y with bounded from above absolute error by truncating the summatory to $k = K$ and obtaining estimates $\widetilde{Y}_k$ for Y_k with bounded from above absolute error. Let ε be an error control parameter, let

$$K = \min \left\{ l \ge 0 : 1 - \sum_{k=0}^{l} Q'_k \le \varepsilon/2 \right\} \tag{2}$$

and assume that an estimate $\widetilde{Y}_k$ for Y_k is available satisfying $|Y_k - \widetilde{Y}_k| \le \varepsilon/2$. Then, the method estimates Y with absolute error $|Y - \widetilde{Y}| \le \varepsilon$ by

$$\widetilde{Y} = \sum_{k=0}^{K} Q'_k \widetilde{Y}_k.$$

The estimates $\widetilde{Y}_k$ are computed with the help of an ROBDD representation of $F(x_1, \ldots, x_C)$. Let G_F denote such a representation for a given ordering of variables. We will start by expressing the Y_k in terms of some conditional probabilities associated with the root node of G_F and deriving recursive expressions for the conditional probabilities associated with non-terminal nodes of G_F. After that, we will show how the $\widetilde{Y}_k$ used by the method are computed by approximating these recursive expressions with bounded from above absolute error.

A. Exact Y_k

First, we introduce some notation which will be used throughout this paper.

b_0, b_1	"0" and "1" terminal nodes of G_F.
$\mathcal{N}$	Set of non-terminal nodes of G_F.
u	Root node of G_F.
$n(0)$, $n(1)$	0- and 1-successor of node n, $n \in \mathcal{N}$.
$H_n()$	Boolean function represented by node n, $n \in \mathcal{N} \cup \{b_0, b_1\}$.
$c(n)$	Component i such that variable x_i is associated with node n, $n \in \mathcal{N}$.
$\mathcal{S}(n)$	If $n \in \mathcal{N}$, set of components i such that variable x_i is located not before $x_{c(n)}$ in the ordering of variables; $\emptyset$ otherwise.
$\mathcal{S}_b(n)$	$\mathcal{S}(n) - \{c(n)\}$, $n \in \mathcal{N}$.
$\mathcal{A}_0(n)$	$\mathcal{S}_b(n) - \mathcal{S}(n(0))$, $n \in \mathcal{N}$.
$\mathcal{A}_1(n)$	$\mathcal{S}_b(n) - \mathcal{S}(n(1))$, $n \in \mathcal{N}$.
$\mathcal{B}_1(n)$	$\mathcal{S}(n) - \mathcal{S}(n(1))$, $n \in \mathcal{N}$.
X_i	Binary random variable with value 1 if and only if component i, $1 \le i \le C$ is affected by some fault.
$q_{\mathcal{D}\|\mathcal{E}}$	Conditional probability that a fault affects some component in set $\mathcal{D}$ given that it affects some component in set $\mathcal{E} \neq \emptyset$, $\mathcal{D} \subset \mathcal{E}$: $q_{\mathcal{D}\|\mathcal{E}} = (\sum_{i\in\mathcal{D}} P'_i)/(\sum_{i\in\mathcal{E}} P'_i)$. We will use the shorthand $q_{i\|\mathcal{E}}$ for $q_{\{i\}\|\mathcal{E}}$.
$\|\mathcal{A}\|$	Number of elements in set $\mathcal{A}$.

$\mathbf{1}_c$	Indicator function: 1 if condition c is satisfied; 0 otherwise.
$B_p^{k,l}$	Binomial probability mass function: $B_p^{k,l} = \binom{k}{l} p^l (1-p)^{k-l}$.

Let $Z_k(n)$, $k \geq 0$, $n \in \mathcal{N}$ be the probability that, given that the set of components $\mathcal{S}(n)$ is affected by k faults, the function $H_n()$, with x_i replaced by X_i, $1 \leq i \leq C$, has value 1[1]. Clearly

$$Y_k = 1 - Z_k(u).$$

The $Z_k(u)$, $k \geq 0$ can be obtained by processing recursively G_F using recursive expressions for $Z_k(n)$, $k \geq 0$, $n \in \mathcal{N}$. To obtain these recursive expressions we exploit the "structure" of G_F. Several cases will have to be considered. Two of them will be discussed in detail next. The recursive expressions for all possible cases will be given later in the form of a theorem.

The first case is $\mathcal{S}(n) = \{c(n)\}$. In that case, $x_{c(n)}$ occupies the last position in the ordering of variables and both $n(0)$ and $n(1)$ are terminal nodes. We will obtain $Z_k(n)$ first for $k = 0$ and next for $k > 0$. For $k = 0$ (no fault affects component $c(n)$), variable $X_{c(n)}$ will take the value 0 with probability 1. But, when $X_{c(n)} = 0$, $H_n()$ is reduced to the function $H_{n(0)}()$. It follows that $Z_0(n)$ is equal to the probability that the function $H_{n(0)}()$ takes the value 1. Since $n(0)$ is a terminal node, this function is the constant function $\mathbf{1}_{n(0)=b_1}$ and, then, $Z_0(n) = \mathbf{1}_{n(0)=b_1}$. For $k > 0$, all k faults will affect component $c(n)$. Then, with probability 1 variable $X_{c(n)}$ will take the value 1 and $H_n()$ will be reduced to the constant function $H_{n(1)}() = \mathbf{1}_{n(1)=b_1}$. Therefore, for $k > 0$, $Z_k(n) = \mathbf{1}_{n(1)=b_1}$.

The second case is $\mathcal{S}(n) \neq \{c(n)\}$, $\mathcal{A}_0(n) = \mathcal{A}_1(n) = \emptyset$. In words, neither $n(0)$ nor $n(1)$ are terminal nodes and in the ordering of variables, both $x_{c(n(0))}$ and $x_{c(n(1))}$ are located right after $x_{c(n)}$, implying $c(n(0)) = c(n(1))$. Reasoning as we did in the previously considered case for $k = 0$, $Z_0(n)$ is equal to the probability that the function $H_{n(0)}()$ takes the value 1. But $n(0)$ not being a terminal node, that probability is $Z_0(n(0))$ and, then, $Z_0(n) = Z_0(n(0))$. The expression for $k > 0$ is obtained by conditioning on the number of faults affecting component $c(n)$ from the k faults affecting components in $\mathcal{S}(n)$. The conditional probability that a fault affects component $c(n)$ given that it affects components in $\mathcal{S}(n)$ is $q_{c(n)|\mathcal{S}(n)}$. In addition, in this case, $\mathcal{S}(n(0)) = \mathcal{S}(n(1)) = \mathcal{S}(n) - \{c(n)\}$. Then, for $k > 0$, the k faults will be distributed randomly among $c(n)$ and the components in $\mathcal{S}(n(0)) = \mathcal{S}(n(1))$ following a binomial probability distribution with parameters k and $q_{c(n)|\mathcal{S}(n)}$. Then, the probability that no fault affects component $c(n)$ is $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 0, $H_n()$ will be reduced to $H_{n(0)}$, and the k faults will affect components in $\mathcal{S}(n(0))$, implying that $Z_k(n)$ will include the contribution $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} Z_k(n(0))$. The probability that l, $1 \leq l \leq k$ faults affect component $c(n)$ is $B^{k,l}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 1, $H_n()$ will be reduced to $H_{n(1)}$, and the remaining $k-l$ faults will affect components in $\mathcal{S}(n(1))$, implying that $Z_k(n)$ will include the contributions $B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}(n(1))$, $1 \leq l \leq k$. Putting everything together, in the case $\mathcal{S}(n) \neq \{c(n)\}$, $\mathcal{A}_0(n) = \mathcal{A}_1(n) = \emptyset$, we have, for $k > 0$

$$Z_k(n) = B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} Z_k\left(n(0)\right) + \sum_{l=1}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}\left(n(1)\right).$$

The following theorem gives the complete set of recursive expressions for $Z_k(n)$, $k \geq 0$, $n \in \mathcal{N}$.

Theorem 1: For $n \in \mathcal{N}$, the following expressions for $Z_k(n)$, $k \geq 0$, hold.

Case a) $(\mathcal{S}(n) = \{c(n)\})$:

$$\begin{aligned} Z_0(n) &= \mathbf{1}_{n(0)=b_1} \\ Z_k(n) &= \mathbf{1}_{n(1)=b_1}, \quad k > 0. \end{aligned}$$

Case b) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) = \emptyset, \mathcal{A}_1(n) = \emptyset)$:

$$\begin{aligned} Z_0(n) &= Z_0\left(n(0)\right) \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} Z_k\left(n(0)\right) \\ &\quad + \sum_{l=1}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}\left(n(1)\right), \quad k > 0. \end{aligned}$$

Case c) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) \neq \emptyset, \mathcal{A}_1(n) = \emptyset, \mathcal{S}(n(0)) \neq \emptyset)$:

$$\begin{aligned} Z_0(n) &= Z_0\left(n(0)\right) \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \sum_{l=0}^{k} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} Z_{k-l}\left(n(0)\right) \\ &\quad + \sum_{l=1}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}\left(n(1)\right), \quad k > 0. \end{aligned}$$

Case d) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_1(n) = \emptyset, \mathcal{S}(n(0)) = \emptyset)$:

$$\begin{aligned} Z_0(n) &= \mathbf{1}_{n(0)=b_1} \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \mathbf{1}_{n(0)=b_1} \\ &\quad + \sum_{l=1}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}\left(n(1)\right), \quad k > 0. \end{aligned}$$

Case e) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) = \emptyset, \mathcal{A}_1(n) \neq \emptyset, \mathcal{S}(n(1)) \neq \emptyset)$:

$$\begin{aligned} Z_0(n) &= Z_0\left(n(0)\right) \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} Z_k\left(n(0)\right) \\ &\quad + \sum_{l=1}^{k} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \\ &\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} Z_{k-l}\left(n(1)\right), \quad k > 0. \end{aligned}$$

Case f) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) = \emptyset, \mathcal{S}(n(1)) = \emptyset)$:

$$\begin{aligned} Z_0(n) &= Z_0\left(n(0)\right) \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} Z_k\left(n(0)\right) \\ &\quad + \left(1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\right) \mathbf{1}_{n(1)=b_1}, \quad k > 0. \end{aligned}$$

[1]For the remainder of this paper, we will refer to the probability that function $H_n()$ with binary variables x_i replaced by the corresponding binary random variables X_i has value 1 simply as the probability that function $H_n()$ has value 1.

Case g) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) \neq \emptyset, \mathcal{A}_1(n) \neq \emptyset, \mathcal{S}(n(0)) \neq \emptyset, \mathcal{S}(n(1)) \neq \emptyset)$:

$$\begin{aligned} Z_0(n) &= Z_0\left(n(0)\right) \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \sum_{l=0}^{k} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} Z_{k-l}\left(n(0)\right) \\ &\quad + \sum_{l=1}^{k} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \\ &\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} Z_{k-l}\left(n(1)\right), \quad k > 0. \end{aligned}$$

Case h) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_1(n) \neq \emptyset, \mathcal{S}(n(0)) = \emptyset, \mathcal{S}(n(1)) \neq \emptyset)$:

$$\begin{aligned} Z_0(n) &= \mathbf{1}_{n(0)=b_1} \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \mathbf{1}_{n(0)=b_1} \\ &\quad + \sum_{l=1}^{k} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \\ &\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} Z_{k-l}\left(n(1)\right), \quad k > 0. \end{aligned}$$

Case i) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) \neq \emptyset, \mathcal{S}(n(0)) \neq \emptyset, \mathcal{S}(n(1)) = \emptyset)$:

$$\begin{aligned} Z_0(n) &= Z_0\left(n(0)\right) \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \sum_{l=0}^{k} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} Z_{k-l}\left(n(0)\right) \\ &\quad + \left(1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\right) \mathbf{1}_{n(1)=b_1}, \quad k > 0. \end{aligned}$$

Case j) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{S}(n(0)) = \emptyset, \mathcal{S}(n(1)) = \emptyset)$:

$$\begin{aligned} Z_0(n) &= \mathbf{1}_{n(0)=b_1} \\ Z_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \mathbf{1}_{n(0)=b_1} \\ &\quad + \left(1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\right) \mathbf{1}_{n(1)=b_1}, \quad k > 0. \end{aligned}$$

Cases a and b have already been proven; Cases c–j are proven in the appendix.

B. Computation of $\widetilde{Y}_k$

We could choose the truncation parameter K so that the left part is $\leq \varepsilon$ (i.e., using (2) with $\varepsilon/2$ replaced by ε), obtain $Z_k(u)$, $0 \leq k \leq K$ by processing G_F bottom-up, using the recursive expressions given by Theorem 1, and estimate the functional yield of the system with absolute error bounded from above by ε as $\sum_{k=0}^{K} Q'_k Y_k$, $Y_k = 1 - Z_k(u)$. However, that trivial method has a computational cost $O(K^2)$ per G_F node, and the total cost of the bottom-up processing of G_F can be relatively large if K is large. In order to reduce that computational cost, in the proposed method, as discussed, K is chosen so that the left part is $\leq \varepsilon/2$, estimates for $Z_k(u)$, $0 \leq k \leq K$ with absolute error bounded from above by $\varepsilon/2$ are obtained, and using them, estimates $\widetilde{Y}_k$ for Y_k, $0 \leq k \leq K$ with absolute error bounded from above by $\varepsilon/2$ are obtained, yielding an estimate for the functional yield $\sum_{k=0}^{K} Q'_k \widetilde{Y}_k$ with absolute error bounded from above by ε. The estimates for $Z_k(u)$, $0 \leq k \leq K$ will be obtained by processing G_F bottom-up, using approximate versions of the recursive expressions of Theorem 1, in which the summatories have been truncated. The truncations take advantage of the fact that a Bernoulli random variable with k replications and small "success" probability has, for large k, a number of successes much smaller than k with high probability. For large K, the bottom-up processing of G_F will account for a significant portion of the computational cost of the trivial method, the truncations of the summatories will tend to be significant, and the computational cost of the method will tend to be significantly smaller than in the trivial method.

The following theorem shows how the summatories involved in Theorem 1 can be truncated so that $Z_k(n)$, $n \in \mathcal{N}$ are estimated with controlled absolute error. The theorem does not specify the truncation points; it only gives conditions which the truncation points have to satisfy. Efficient procedures for the selection of truncation points minimizing the number of terms in the summatories will be described after the theorem.

Theorem 2: Let ε', $0 \leq \varepsilon' < 1$. For $n \in \mathcal{N}$, let $\widetilde{Z}_k(n)$, $k \geq 0$ be defined as follows.

Case a) $(\mathcal{S}(n) = \{c(n)\})$:

$$\begin{aligned} \widetilde{Z}_0(n) &= \mathbf{1}_{n(0)=b_1} \\ \widetilde{Z}_k(n) &= \mathbf{1}_{n(1)=b_1}, \quad k > 0. \end{aligned}$$

Case b) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) = \emptyset, \mathcal{A}_1(n) = \emptyset)$:

$$\begin{aligned} \widetilde{Z}_0(n) &= \widetilde{Z}_0\left(n(0)\right) \\ \widetilde{Z}_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_k\left(n(0)\right) \\ &\quad + \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_{k-l}\left(n(1)\right), \quad k > 0 \end{aligned}$$

where, if $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} + \varepsilon' \geq 1$, then $m = 1$, $M = 0$, and otherwise m, M, $1 \leq m \leq M \leq k$ are integers satisfying

$$\sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \geq 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \varepsilon'.$$

Case c) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) \neq \emptyset, \mathcal{A}_1(n) = \emptyset, \mathcal{S}(n(0)) \neq \emptyset)$:

$$\begin{aligned} \widetilde{Z}_0(n) &= \widetilde{Z}_0\left(n(0)\right), \\ \widetilde{Z}_k(n) &= B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \widetilde{Z}_{k-l}\left(n(0)\right) \\ &\quad + \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_{k-l}\left(n(1)\right), \quad k > 0 \end{aligned}$$

where m, M are as in Case b with ε' replaced by $\varepsilon'/2$, and m', M', $0 \leq m' \leq M' \leq k$ are integers satisfying

$$\sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \geq 1 - \varepsilon'/2.$$

Case d) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_1(n) = \emptyset, \mathcal{S}(n(0)) = \emptyset)$:

$$\widetilde{Z}_0(n) = \mathbf{1}_{n(0)=b_1}$$
$$\widetilde{Z}_k(n) = B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \mathbf{1}_{n(0)=b_1} + \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_{k-l}(n(1)), \quad k>0$$

where m, M are as in Case b.

Case e) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) = \emptyset, \mathcal{A}_1(n) \neq \emptyset, \mathcal{S}(n(1)) \neq \emptyset)$:

$$\widetilde{Z}_0(n) = \widetilde{Z}_0(n(0))$$
$$\widetilde{Z}_k(n) = B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_k(n(0)) + \sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \widetilde{Z}_{k-l}(n(1)), \quad k>0$$

where, if $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} + \varepsilon' \geq 1$, then $m=1$, $M=0$, and otherwise m, M, $1 \leq m \leq M \leq k$ are integers satisfying

$$\sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \geq 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \varepsilon'.$$

Case f) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) = \emptyset, \mathcal{S}(n(1)) = \emptyset)$:

$$\widetilde{Z}_0(n) = \widetilde{Z}_0(n(0)),$$
$$\widetilde{Z}_k(n) = B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_k(n(0)) + \left(1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\right) \mathbf{1}_{n(1)=b_1}, \quad k>0.$$

Case g) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) \neq \emptyset, \mathcal{A}_1(n) \neq \emptyset, \mathcal{S}(n(0)) \neq \emptyset, \mathcal{S}(n(1)) \neq \emptyset)$:

$$\widetilde{Z}_0(n) = \widetilde{Z}_0(n(0)),$$
$$\widetilde{Z}_k(n) = B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \widetilde{Z}_{k-l}(n(0)) + \sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \widetilde{Z}_{k-l}(n(1)), \quad k>0$$

where m, M are as in Case e with ε' replaced by $\varepsilon'/2$ and m', M' are as in Case c.

Case h) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_1(n) \neq \emptyset, \mathcal{S}(n(0)) = \emptyset, \mathcal{S}(n(1)) \neq \emptyset)$:

$$\widetilde{Z}_0(n) = \mathbf{1}_{n(0)=b_1},$$
$$\widetilde{Z}_k(n) = B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \mathbf{1}_{n(0)=b_1} + \sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \widetilde{Z}_{k-l}(n(1)), \quad k>0$$

where m, M are as in Case e.

Case i) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{A}_0(n) \neq \emptyset, \mathcal{S}(n(0)) \neq \emptyset, \mathcal{S}(n(1)) = \emptyset)$:

$$\widetilde{Z}_0(n) = \widetilde{Z}_0(n(0)),$$
$$\widetilde{Z}_k(n) = B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \widetilde{Z}_{k-l}(n(0)) + \left(1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\right) \mathbf{1}_{n(1)=b_1}, \quad k>0$$

where m', M' are as in Case c with $\varepsilon'/2$ replaced by ε'.

Case j) $(\mathcal{S}(n) \neq \{c(n)\}, \mathcal{S}(n(0)) = \emptyset, \mathcal{S}(n(1)) = \emptyset)$:

$$\widetilde{Z}_0(n) = \mathbf{1}_{n(0)=b_1}$$
$$\widetilde{Z}_k(n) = B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \mathbf{1}_{n(0)=b_1} + \left(1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\right) \mathbf{1}_{n(1)=b_1}, \quad k>0.$$

Then

$$\left|Z_k(n) - \widetilde{Z}_k(n)\right| \leq \left(|\mathcal{S}(n)| - 1\right) \varepsilon'.$$

Proof: See the appendix. □

We discuss next procedures for selecting the truncation points of the summatories involved in the previous theorem and computing the truncated summatories. In Cases b, c, and d of the theorem, we have to select m and M and compute

$$S = \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_{k-l}(n(1))$$

with m, M, $1 \leq m \leq M \leq k$ satisfying

$$S' = \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \geq 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \delta$$

for some δ, $0 \leq \delta < 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. In Cases c, g, and i, we have to select m' and M' and compute

$$T = \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \widetilde{Z}_{k-l}(n(0))$$

with m', M', $0 \leq m' \leq M' \leq k$ satisfying

$$T' = \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \geq 1 - \delta$$

for some δ, $0 \leq \delta < 1$. Finally, with $q_1 = q_{\mathcal{B}_1(n)|\mathcal{S}(n)}$, $q_2 = q_{\mathcal{A}_1(n)|\mathcal{S}(n)}$, and $R^{k,l}_{q_1,q_2} = \binom{k}{l}(q_1^l - q_2^l)(1-q_1)^{k-l}$, in Cases e, g, and h, we have to select m and M and compute

$$U = \sum_{l=m}^{M} R^{k,l}_{q_1,q_2} \widetilde{Z}_{k-l}(n(1))$$

with m, M, $1 \leq m \leq M \leq k$ satisfying

$$U' = \sum_{l=m}^{M} R^{k,l}_{q_1,q_2} \geq 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \delta$$

for some δ, $0 \leq \delta < 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. We note that, in the latter case, such integers exist because, since $q_{c(n)|\mathcal{S}(n)} = q_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q_{\mathcal{A}_1(n)|\mathcal{S}(n)} = q_1 - q_2$, $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} + \sum_{l=1}^{k} R^{k,l}_{q_1,q_2} = 1$.

Let $\lfloor x \rfloor$ denote the largest integer $\leq x$. The procedure for selecting m and M and computing S is based on the fact that as l goes from 0 to k, the terms $B^{k,l}_p, 0 < p < 1$ first increase monotonically and next decrease monotonically, reaching their largest value at $l = \lfloor (k+1)p \rfloor$, except when $(k+1)p$ is an integer, in which case the largest value is achieved at both $l = (k+1)p-1$ and $l = (k+1)p$ [24]. The procedure is as follows. Starting with $m = M = \max\{1, \lfloor (k+1)q_{c(n)|\mathcal{S}(n)} \rfloor\}$, we compute S and S' for decreasing m as long as $S' < 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \delta$ and either $M = k$ or, M being $< k$, $m > 1$ and $B^{k,m-1}_{q_{c(n)|\mathcal{S}(n)}} \geq B^{k,M+1}_{q_{c(n)|\mathcal{S}(n)}}$. After that, S and S' are updated for increasing M as long as $S' < 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \delta$ and either $m = 1$, or, m being > 1, $M < k$ and $B^{k,M+1}_{q_{c(n)|\mathcal{S}(n)}} \geq B^{k,m-1}_{q_{c(n)|\mathcal{S}(n)}}$. From that point on, we continue updating S and S' for decreasing m and increasing M alternatively in a similar way until S' becomes $\geq 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \delta$ or $m = 1$, $M = k$ is reached.

The procedure for selecting m' and M' and computing T is similar. Starting with $m' = M' = \lfloor (k+1)q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)} \rfloor$, we compute T and T' for decreasing m' as long as $T' < 1 - \delta$ and either $M' = k$ or, M' being $< k$, $m' > 0$ and $B^{k,m'-1}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \geq B^{k,M'+1}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}}$. After that, T and T' are updated for increasing M' as long as $T' < 1 - \delta$ and either $m' = 0$, or, m' being > 0, $M' < k$ and $B^{k,M'+1}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \geq B^{k,m'-1}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}}$. From that point on, we continue updating T and T' for decreasing m' and increasing M' alternatively in a similar way until T' becomes $\geq 1 - \delta$ or $m' = 0$, $M' = k$ is reached.

Let $\lceil x \rceil$ denote the smallest integer $\geq x$. The procedure for selecting m and M and computing U is based on the following lemma.

Lemma 1: Assume $\mathcal{A}_1(n) \neq \emptyset$ and $\mathcal{S}(n(1)) \neq \emptyset$, and let $q_1 = q_{\mathcal{B}_1(n)|\mathcal{S}(n)}$, $q_2 = q_{\mathcal{A}_1(n)|\mathcal{S}(n)}$, and $R^{k,l}_{q_1,q_2} = \binom{k}{l}(q_1^l - q_2^l)(1-q_1)^{k-l}$, $k \geq 1$, $1 \leq l \leq k$. Then, the terms $R^{k,l}_{q_1,q_2}$ increase monotonically as l goes from 1 to $\lfloor (k+1)q_1 \rfloor$ and decrease monotonically as l goes from $\lceil kq_1 + 1 \rceil - 1$ to k.

Proof: See the appendix. □

The procedure, which uses the fact that, since $0 < q_1 < 1$, $\lfloor (k+1)q_1 \rfloor \leq \lceil kq_1 + 1 \rceil - 1 \leq \lfloor (k+1)q_1 \rfloor + 1$, is as follows. Starting with $m = M = \min\{k, \lfloor (k+1)q_1 \rfloor + 1\}$, we compute U and U' for decreasing m as long as $U' < 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \delta$ and either $M = k$, or, M being $< k$, $m > 1$ and $R^{k,m-1}_{q_1,q_2} \geq R^{k,M+1}_{q_1,q_2}$. After that, U and U' are updated for increasing M as long as $U' < 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \delta$ and either $m = 1$ or, m being > 1, $M < k$ and $R^{k,M+1}_{q_1,q_2} \geq R^{k,m-1}_{q_1,q_2}$. From that point on, we continue updating U and U' for decreasing m and increasing M alternatively in a similar way until U' becomes $\geq 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \delta$ or $m = 1$, $M = k$ is reached.

We can now define the $\widetilde{Y}_k$, $0 \leq k \leq K$ used in the method and how they are computed. The $\widetilde{Y}_k$ are

$$\widetilde{Y}_k = 1 - \widetilde{Z}_k(u).$$

The estimates $\widetilde{Z}_k(n)$, $n \in \mathcal{N}$ are computed by traversing bottom-up G_{F} and using the recursive expressions given in Theorem 2 with $\varepsilon' = \varepsilon/(2\max\{1, C-1\})$. The truncation points and the summatories are computed using the procedures just described. Since $|\mathcal{S}(u)| = C$, we have, by Theorem 2

$$|Y_k - \widetilde{Y}_k| = \left| Z_k(u) - \widetilde{Z}_k(u) \right| \leq (C-1)\varepsilon' = (C-1)\frac{\varepsilon}{2\max\{1, C-1\}} \leq \varepsilon/2$$

as required.

Remark: For efficiency reasons, ROBDD packages typically use ROBDDs with complement edges [25]. In those ROBDDs, a complement edge leads to a node representing the complement of the function obtained by setting the variable associated with the node to the value (0 or 1) associated with the edge. In addition, the top node may represent the complement of the function. The proposed method for computing the yield can be easily adapted to ROBDDs with complement edges. It suffices to set $\widetilde{Y}_k = \widetilde{Z}_k(u)$ in case the top node u represents the complement of the function and, in the recursive expressions for $\widetilde{Z}_k(n)$ of Theorem 2, use the complements to 1 of the values associated with the nodes reached by a complement edge, e.g., for the node $n(0)$, use $1 - \widetilde{Z}_l(n(0))$ instead of $\widetilde{Z}_l(n(0))$ if $n(0)$ is a non-terminal node and $1 - \mathbf{1}_{n(0)=b_1}$ instead of $\mathbf{1}_{n(0)=b_1}$ if $n(0)$ is a terminal node, in case the 0-edge is a complement edge.

III. Analysis

A. Benchmarks

We describe next the benchmarks that were used to analyze the computational cost of the proposed method. The benchmarks are instances of three scalable SoC examples. In all benchmarks, the number of defects is assumed to follow a negative binomial distribution with $\alpha = 3$ and the probabilities P_i are taken so that $P_L = \sum_{i=1}^{C} P_i = 0.5$, implying that the probability of any given defect causing a fault is 0.5 and that the expected number of faults is $\lambda/2$.

As previously said, there exists a trend in the SoC community towards the use of a network-on-chip as a communication subsystem among the IPs. Examples of such networks include Nostrum [26], SoCBUS [27], and SoCIN [28], which use a regular 2-D mesh topology, and SPIN [29], which uses a fat-tree topology. Then, as our first scalable example we have chosen a defect-tolerant SoC with a network-on-chip with a regular 2-D mesh topology. The system, called MESH (k, n), includes k groups of $4n^2/k$ IPs $\mathrm{IP}^{(j)}$, $1 \leq j \leq k$ interconnected by a $2n \times 2n$ mesh made up of $4n^2$ switches S. The architecture of MESH (k, n) is illustrated in Fig. 1 for the case $k = 2$, $n = 2$.

The system is functioning if $4n^2/k - 1$ unfailed IPs of every group can communicate through the mesh with $4n^2/k - 1$ unfailed IPs of every other group. It is assumed that links are not affected by defects. Thus, the system can be conceptualized as made up of only $\mathrm{IP}^{(j)}$'s and S's. The probabilities P_i are taken so that, calling $P_{\mathrm{IP}^{(j)}}$ the P_i probability of an $\mathrm{IP}^{(j)}$ and P_S the P_i probability of an S, all $P_{\mathrm{IP}^{(j)}}$ are equal and $P_S/P_{\mathrm{IP}^{(1)}} = 0.2$.

The second scalable example is the defect-tolerant SoC FAT-TREE (k, n) with the architecture described in Fig. 2 for the case $k = 2$, $n = 3$.

The system includes $k^n/2$ IPs IPA and $k^n/2$ IPs IPB interconnected by a k-ary, n-tree fat-tree [30]. The system is func-

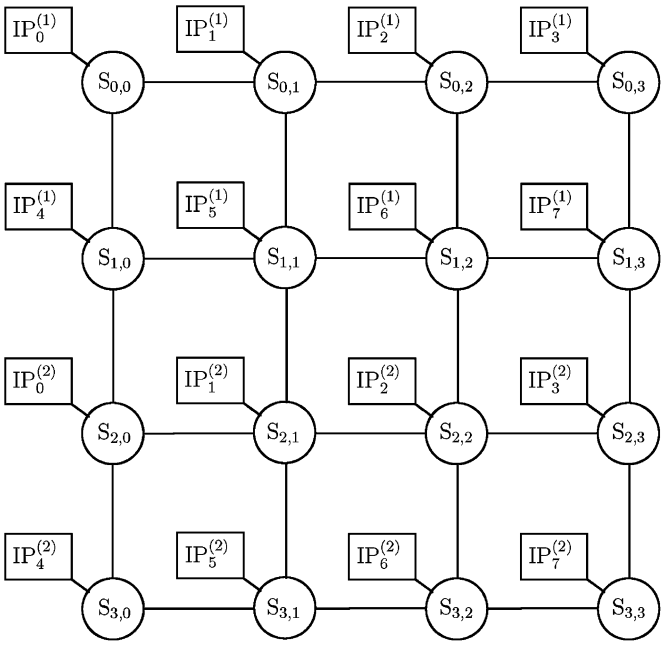


Fig. 1. Architecture of defect-tolerant SoC MESH (2,2).

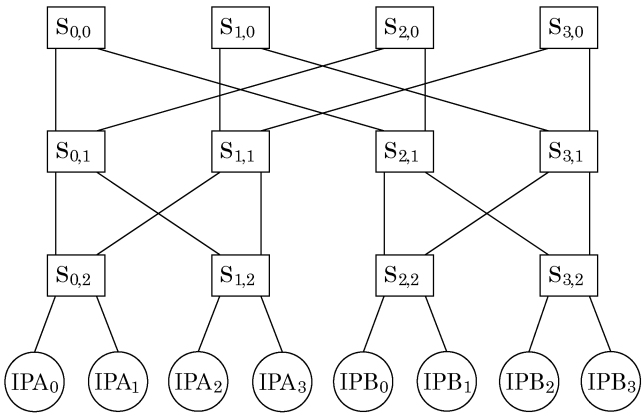


Fig. 2. Architecture of defect-tolerant SoC FAT-TREE (2,3).

tioning if $k^n/2-1$ unfailed IPAs can communicate through the fat-tree with $k^n/2-1$ unfailed IPBs. It is assumed that links are not affected by defects. Thus, the system can be conceptualized as made up of only IPAs, IPBs, and S's. The probabilities P_i are taken so that, calling P_{IPA} the P_i probability of an IPA, P_{IPB} the P_i probability of an IPB, and P_S the P_i probability of an S, $P_{\mathrm{IPB}} = P_{\mathrm{IPA}}$ and $P_S/P_{\mathrm{IPA}} = 0.15$.

The last scalable example is the defect-tolerant SoC ESEN $(k, n \times m)$ with the architecture described in Fig. 3 for the case $k = 8, n = 8, m = 2$.

The system includes k groups of $(n \times m)/k$ IPs $\mathrm{IP}^{(j)}$, $1 \leq j \leq k$ interconnected by an ESEN multiexchange interconnection network with n inputs [31], through $2n$ concentrators C if $m > 1$, in which each switching element (S) of the first and last stage have a redundant copy. The system is functioning if $(n \times m)/k - 1$ unfailed IPs of each group can communicate with $(n \times m)/k - 1$ unfailed IPs of every other group through the ESEN network. It is assumed that links are not affected by

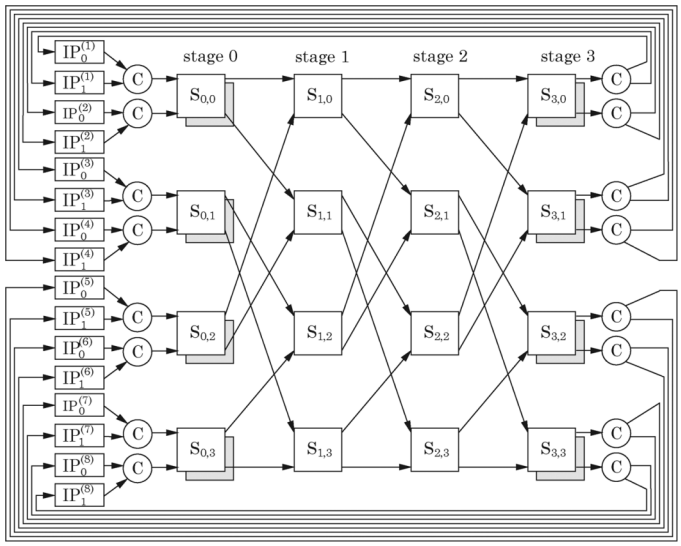


Fig. 3. Architecture of defect-tolerant SoC ESEN (8, 8 × 2).

TABLE I
CHARACTERISTICS OF THE BENCHMARKS

benchmark	components	gates	edges
MESH (8, 32)	8,192	12,337	167,408
MESH (32, 32)	8,192	12,289	153,536
FAT-TREE (4, 3)	112	1,017	3,076
ESEN (128, 32 × 32)	1,216	4,705	13,562
ESEN (512, 32 × 32)	1,216	1,633	4,346

defects. Thus, the system can be conceptualized as made up of only $\mathrm{IP}^{(j)}$'s, S's, and, if $m > 1$, C's. The probabilities P_i are taken so that, calling $P_{\mathrm{IP}^{(j)}}$ the P_i probability of an $\mathrm{IP}^{(j)}$, P_S the P_i probability of an S, and P_C the P_i probability of a C, all $P_{\mathrm{IP}^{(j)}}$ are equal, $P_S/P_{\mathrm{IP}^{(1)}} = 0.5$, and $P_C/P_{\mathrm{IP}^{(1)}} = 0.05$.

Table I gives the number of components of the instances of the scalable examples used as benchmarks in the experiments and the numbers of gates and edges of the fault-trees which were used to specify the fault-tree functions. Those fault-trees were generated automatically using code specific for every scalable example.

B. Results

All results were obtained on a workstation equipped with four Dual-Core AMD Opteron processor chips at 2.2 GHz and 32 GB of main memory, using only one core and limiting memory consumption to 4 GB. To build the ROBDD representations, we used the CUDD package [25], which constructs ROBDDs with complement 0-edges. The order of the variables was chosen prior to building the ROBDDs using the heuristic *weight* described in [32] for the MESH (k, n) and ESEN $(k, n \times m)$ benchmarks, and the heuristic *H4* described in [33] for the FAT-TREE (k, n) benchmarks.

We start by showing that the benchmarks cover a wide range of design scenarios and that the proposed method can be applied to very complex SoCs for a wide range of values of the expected number of defects λ. Fig. 4 plots the functional yield of the benchmarks as a function of λ. We can note that the

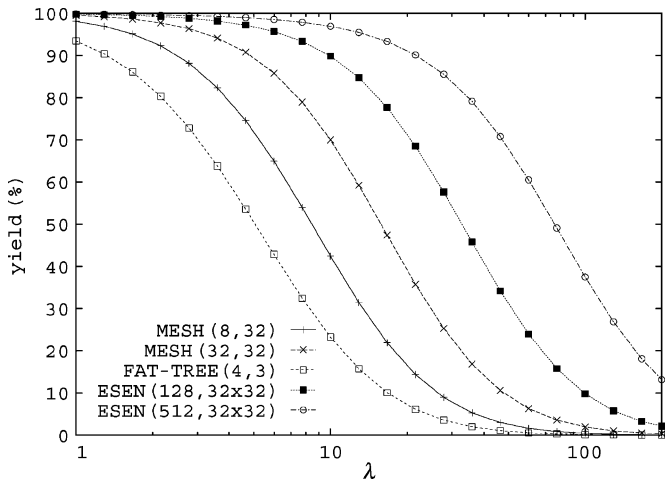


Fig. 4. Functional yield as a function of the expected number of defects.

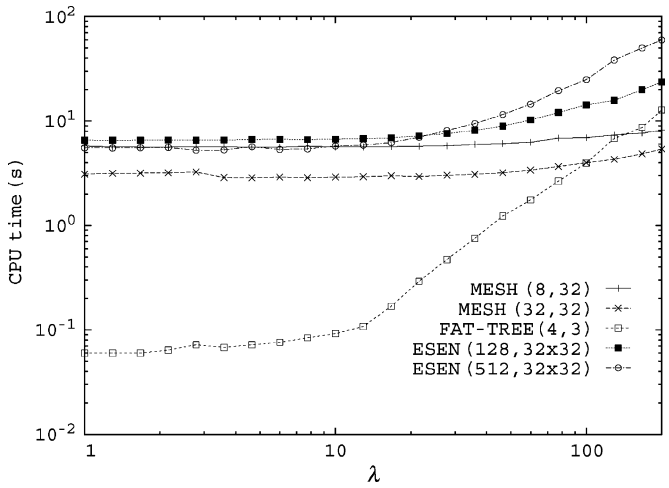


Fig. 5. CPU times with $\varepsilon = 10^{-5}$ as a function of the expected number of defects.

TABLE II
COMPUTATIONAL COST OF THE METHOD WITH $\varepsilon = 10^{-5}$

benchmark	#nodes	memory (MB)	CPU (s) tot	CPU (s) proc
MESH (8, 32), λ = 1	16,354	109	5.75	0.128
MESH (8, 32), λ = 10	16,354	113	5.65	0.152
MESH (8, 32), λ = 200	16,354	180	8.17	2.67
MESH (32, 32), λ = 1	16,258	139	3.10	0.140
MESH (32, 32), λ = 10	16,258	142	2.90	0.152
MESH (32, 32), λ = 200	16,258	209	5.38	2.67
FAT-TREE (4, 3), λ = 1	3,063	13.0	0.060	0.008
FAT-TREE (4, 3), λ = 10	3,063	13.0	0.092	0.040
FAT-TREE (4, 3), λ = 200	3,063	26.5	12.7	12.7
ESEN (128, 32×32), λ = 1	71,853	64.2	6.56	0.108
ESEN (128, 32×32), λ = 10	71,853	78.1	6.71	0.264
ESEN (128, 32×32), λ = 200	71,853	381	23.7	17.3
ESEN (512, 32×32), λ = 1	71,085	54.9	5.53	0.128
ESEN (512, 32×32), λ = 10	71,085	69.6	5.77	0.400
ESEN (512, 32×32), λ = 200	71,085	368	59.7	54.6

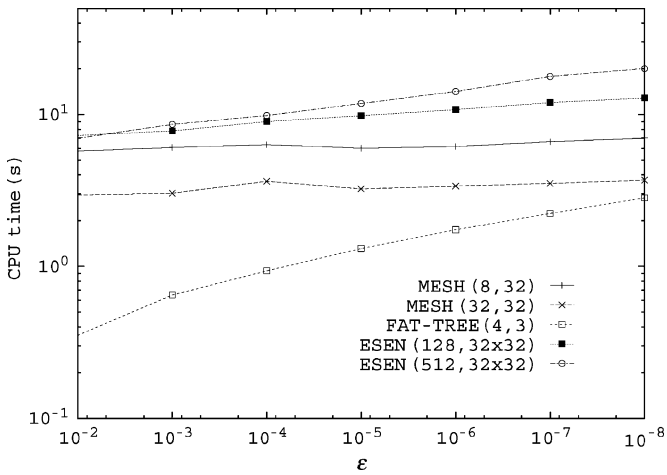


Fig. 6. CPU times for $\lambda = 50$ as a function of the error requirement ε.

benchmarks cover a wide range of dependencies of the functional yield with respect to λ. Fig. 5 gives the CPU times consumed by the method with an error requirement $\varepsilon = 10^{-5}$ as a function of λ. As it can be seen, the method is able to compute the functional yield with a tight error requirement for very complex systems for λ as large as 200 in reasonable CPU times, the only exception being the FAT-TREE (4,3) benchmark which has only a moderate number of components (112). Attempt to apply the method to a FAT-TREE benchmark with a larger number of components resulted in the failure of the method due to excessive memory consumption. A behavior even worse than that of the FAT-TREE example is possible since the size of the ROBDD is in the worst case exponential with the number of variables of the represented function [34].

Table II analyzes the computational cost of the method in more detail. We give the size (number of nodes) of the ROBDD representation of the fault-tree function, the memory consumption, the total CPU time (tot), and the CPU time of the bottom-up processing of the ROBDD (proc). In all cases the computational cost both in terms of memory and CPU time is moderate. In addition, the CPU time of the bottom-up processing of the ROBDD is practically negligible up to $\lambda = 10$ and afterwards seems to increase relatively slowly with λ for almost all the benchmarks.

We analyze next the impact of the error requirement ε on the CPU time. Fig. 6 shows the CPU times for $\lambda = 50$ as a function of ε. As it can be seen, the CPU time increases moderately with the error requirement.

To end, we analyze the extent to which the truncation of the summatories of the recursive expressions for $Z_k(n), n \in \mathcal{N}$ reduces the CPU time of the bottom-up processing of the ROBDD and the CPU time of the method for large values of λ. Table III compares the CPU times of the bottom-up processing of the ROBDD (proc) and the total CPU times (tot) for the proposed method and for the trivial method (without truncation of summatories) for $\lambda = 200$ and an error requirement $\varepsilon = 10^{-5}$. The proposed method has significantly smaller CPU times than the trivial method, with a reduction factor ranging from 2 to 21 for the CPU times of the bottom-up processing of the ROBDD and from 2 to 15 for the total CPU times. The savings are significantly smaller for the FAT-TREE benchmark. This has to

TABLE III
IMPACT OF THE TRUNCATION OF THE SUMMATORIES

benchmark	CPU (s)			
	proposed		trivial	
	tot	proc	tot	proc
MESH (8, 32), λ = 200	8.17	2.67	54.5	48.3
MESH (32, 32), λ = 200	5.38	2.67	50.6	47.7
FAT-TREE (4,3), λ = 200	12.7	12.7	24.9	24.8
ESEN (128, 32×32), λ = 200	23.7	17.3	373	367
ESEN (512, 32×32), λ = 200	59.7	54.6	377	372

do with the fact that the benchmark has a significantly smaller number of components, causing the "success" probability of the Bernoulli distributions appearing in the summatories to be significantly larger, and causing the truncations of those summatories to be less significant. The values of the truncation parameter K was 585 for the proposed method and 559 for the trivial method.

Summarizing, the proposed method seems, with some exceptions, to be able to process with moderate computational cost defect-tolerant systems with very large numbers of components for a wide range of values of λ and ε.

C. Discussion of Alternatives

There seem to be only three immediate alternatives to the proposed method: the combinatorial method described in [15], the so-called "compounding technique" (see, e.g., [13]), and simulation. The compounding technique can only be used with compound Poisson models.

The method described in [15] uses reduced ordered multivalued decision diagrams. However, experiments in [15] indicated that that method has a much larger computational cost than the method proposed in this paper and is able to handle efficiently only SoCs with up to a few tens of components. As a confirmation, the method was not able to analyze any of the five benchmarks considered in this paper with $\lambda = 10$ when run with an error requirement $\varepsilon = 10^{-5}$ and a memory limitation of 4 GB. In all cases, the method failed due to excessive memory consumption.

The compounding technique exploits the fact that any compound Poisson model can be interpreted as a Poisson model in which the parameter is a random variable with some distribution, and that components are statistically independent when the number of faults follows a Poisson distribution. Then, the yield of the system can be computed by combining: 1) a standard ROBDD-based combinatorial method [35] to compute the yield of the system conditioned to the parameter of the Poisson distribution having a specified value and 2) a numerical integration method requiring only values of the function to be integrated to obtain the yield from values of the product of the conditional yield and the probability density function of the parameter of the Poisson distribution. That alternative requires the construction of the ROBDD representation of the fault-tree function of the system exactly as the method proposed in this paper. In addition, it requires to make potentially many traversals of the ROBDD with a constant cost per node in terms of CPU time consumption. However, the method does not provide rigorous error control, as the method proposed in the paper does. On the other hand, the method proposed in the paper requires only one traversal of the ROBDD but with a cost per node that depends on λ and ε, and has a potentially larger memory requirement resulting from the need of storing $K + 1$ floating-point variables per node of the ROBDD. To compare with our method, we implemented the compounding technique using the well-known Quadpack numerical integration package [36] to perform the numerical integration and ran it on the five benchmarks with $\lambda = 10$ and a target absolute error $\varepsilon = 10^{-5}$. In terms of memory consumption, the alternative was cheaper in all cases. Thus, for the MESH (8,32) benchmark it required 81.6 MB versus the 113 MB required by our method; for the MESH (32,32) benchmark, 112 MB versus 142 MB; for the FAT-TREE (4,3) benchmark, 12.7 MB versus 13.0 MB; for the ESEN (128, 32 × 32) benchmark, 57.3 MB versus 78.1 MB; and for the ESEN (512, 32 × 32) benchmark, 48.9 MB versus 69.6 MB. With regard to CPU time, our method was slightly faster: 5.65 s versus 6.24 s for the first benchmark, 2.90 versus 3.96 s for the second, 0.092 versus 0.116 s for the third, 6.71 versus 8.72 s for the fourth, and 5.77 versus 7.46 s for the fifth. In summary, the price paid in the proposed method in terms of computational cost seems to be moderate and justified for having strict error control.

Simulation has clearly a smaller memory consumption than our method. For practical purposes, that consumption is reduced to the memory required to hold the description of the fault-tree. On the other hand, it suffers from poor error control (simulation offers just a confidence interval for the estimate which is known to succeed with a given probability, assuming that the estimation of the variance by the sample variance is accurate) and is potentially very time-consuming if the yield is neither close to 0% nor close to 100% and has to be estimated with high accuracy. To analyze the second issue, we built an efficient simulator and ran it on the five benchmarks with $\lambda = 10$. The simulator used the fault-tree to determine the functioning state of the SoC. Table IV gives the computational cost of the simulation with a target 95% confidence interval of $\pm 10^{-2}$, $\pm 10^{-3}$, and $\pm 10^{-4}$. In comparison with our method (see Table II), we note that, as expected, in all cases simulation has a significantly smaller memory consumption. However, the CPU times are large, particularly when the yield to be estimated is not too close to 100% nor to 0% (see Fig. 4), and increase sharply with the reciprocal of the required half-width of the confidence interval. In summary, due to its poor error control, simulation seems to be an alternative to be used only when our method fails due to excessive memory consumption or excessive CPU time, and in that case with a not too tight accuracy requirement.

IV. CONCLUSION

In this paper, we have developed a new combinatorial method for the evaluation of the functional yield of defect-tolerant SoCs. The inputs of the method are the distribution of the number of random manufacturing defects in the area occupied by the system, which can be provided by the manufacturer of the SoC, and, for each component making up the system, the probability that a given defect causes the failure of the component, which can be estimated from data of the manufacturing process and the layout of the components using existing tools. The method requires the construction of an ROBDD representation of the fault-tree function of the system and provides rigorous error

TABLE IV
COMPUTATIONAL COST OF SIMULATION WITH A TARGET 95% CONFIDENCE INTERVAL OF $\pm 10^{-2}$, $\pm 10^{-3}$, AND $\pm 10^{-4}$

benchmark	memory (MB) (CPU (s)) $\pm 10^{-2}$	$\pm 10^{-3}$	$\pm 10^{-4}$
MESH (8, 32), λ = 10	13.0	13.0	13.0
	(3.79)	(217)	(2.43×10^4)
MESH (32, 32), λ = 10	12.3	12.4	12.4
	(2.36)	(73.0)	(7.78×10^3)
FAT-TREE (4, 3), λ = 10	1.66	1.71	1.71
	(0.048)	(3.60)	(359)
ESEN (128, 32 × 32), λ = 10	3.04	3.09	3.09
	(0.124)	(1.86)	(173)
ESEN (512, 32 × 32), λ = 10	2.06	2.10	2.11
	(0.096)	(0.292)	(28.0)

control. Our experiments seem to indicate that, with some exceptions, the method allows the analysis with affordable computational resources of defect-tolerant SoCs with very large numbers of components for a wide range of values of the expected number of defects and the error requirement.

APPENDIX

A. Proof of Cases c–j of Theorem 1

Let k be the number of faults affecting components in $\mathcal{S}(n)$. We will first justify the expressions for $k = 0$ and next for $k > 0$. For $k = 0$, with probability 1 variable $X_{c(n)}$ will take the value 0 and $H_n()$ will be reduced to the function $H_{n(0)}()$. It follows that $Z_0(n)$ is equal to the probability that the function $H_{n(0)}()$ takes the value 1. In Cases c, e, f, g, and i, $n(0)$ is not a terminal node, implying that $Z_0(n) = Z_0(n(0))$. In Cases d, h, and j, $n(0)$ is a terminal node. Then, in those cases, $H_{n(0)}()$ is the constant function $\mathbf{1}_{n(0)=b_1}$ and, therefore, $Z_0(n) = \mathbf{1}_{n(0)=b_1}$. The expressions for $k > 0$ are justified next on a case by case basis.

Case c) In this case, neither $n(0)$ nor $n(1)$ is a terminal node, $\mathcal{S}(n)$ includes components different from $c(n)$ not in $\mathcal{S}(n(0))$ (the components in $\mathcal{A}_0(n)$), and $\mathcal{S}(n)$ does not include components different from $c(n)$ not in $\mathcal{S}(n(1))$. The expression is obtained by conditioning on the number of faults affecting component $c(n)$ from the k faults affecting components in $\mathcal{S}(n)$. The probability that no fault affects component $c(n)$ is $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 0, $H_n()$ will be reduced to $H_{n(0)}$, and the k faults will affect components in $\mathcal{S}_b(n)$. In that case, with probability $B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}}$, l faults, $0 \leq l \leq k$ will affect components in $\mathcal{A}_0(n)$ and $k - l$ faults will affect components in $\mathcal{S}(n(0))$. Let $1 \leq l \leq k$. With probability $B^{k,l}_{q_{c(n)|\mathcal{S}(n)}}$, l faults will affect component $c(n)$, variable $X_{c(n)}$ will take the value 1, $H_n()$ will be reduced to $H_{n(1)}$, and the remaining $k - l$ faults will affect components in $\mathcal{S}(n(1))$. This justifies the recursive expression.

Case d) In this case, $n(0)$ is a terminal node, $n(1)$ is not a terminal node, and $\mathcal{S}(n)$ does not include components different from $c(n)$ not in $\mathcal{S}(n(1))$. The expression is obtained by conditioning on the number of faults affecting component $c(n)$ from the k faults affecting components in $\mathcal{S}(n)$. The probability that no faults affects component $c(n)$ is $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 0 and $H_n()$ will be reduced to the constant function $\mathbf{1}_{n(0)=b_1}$. The probability that l, $1 \leq l \leq k$ faults affect component $c(n)$ is $B^{k,l}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 1, $H_n()$ will be reduced to $H_{n(1)}$, and the remaining $k - l$ faults will affect components in $\mathcal{S}(n(1))$. This justifies the recursive expression.

Case e) In this case, neither $n(0)$ nor $n(1)$ is a terminal node, $\mathcal{S}(n)$ does not include components different from $c(n)$ not in $\mathcal{S}(n(0))$, and $\mathcal{S}(n)$ includes components different from $c(n)$ not in $\mathcal{S}(n(1))$ (the components in $\mathcal{A}_1(n)$). The expression is obtained by conditioning on the number of faults affecting component $c(n)$ from the k faults affecting components in $\mathcal{S}(n)$. The probability that no fault affects component $c(n)$ is $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 0, $H_n()$ will be reduced to $H_{n(0)}$, and all k faults will affect components in $\mathcal{S}(n(0))$. The probability that l, $1 \leq l \leq k$ faults will affect components in $\mathcal{B}_1(n) = \{c(n)\} \cup \mathcal{A}_1(n)$ with some of them affecting component $c(n)$ is $\binom{k}{l} q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} (1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} - \binom{k}{l} q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} (1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} = \binom{k}{l} (q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}) (1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l}$. With that probability, variable $X_{c(n)}$ will take the value 1, $H_n()$ will be reduced to $H_{n(1)}$, and $k - l$ faults will affect components in $\mathcal{S}(n(1)) = \mathcal{S}(n) - \mathcal{B}_1(n)$. This justifies the recursive expression.

Case f) In this case, $n(0)$ is not a terminal node, $n(1)$ is a terminal node, $\mathcal{S}(n)$ does not include components different from $c(n)$ not in $\mathcal{S}(n(0))$, and $\mathcal{S}(n)$ includes components different from $c(n)$. With probability $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$, variable $X_{c(n)}$ will take the value 0, $H_n()$ will be reduced to $H_{n(0)}$, and all k faults will affect components in $\mathcal{S}(n(0))$. With probability $1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$, variable $X_{c(n)}$ will take the value 1 and $H_n()$ will be reduced to the constant function $\mathbf{1}_{n(1)=b_1}$. This justifies the recursive expression.

Case g) In this case, neither $n(0)$ nor $n(1)$ is a terminal node, $\mathcal{S}(n)$ includes components different from $c(n)$ not in $\mathcal{S}(n(0))$ (the components in $\mathcal{A}_0(n)$), and $\mathcal{S}(n)$ includes components different from $c(n)$ not in $\mathcal{S}(n(1))$ (the components in $\mathcal{A}_1(n)$). The expression is obtained by conditioning on the number of faults affecting component $c(n)$ from the k faults affecting components in $\mathcal{S}(n)$. The probability that no fault affects component $c(n)$ is $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 0, $H_n()$ will be reduced to $H_{n(0)}$, and the k faults will affect components in $\mathcal{S}_b(n)$. In that case, with probability $B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}}$, l faults, $0 \leq l \leq k$ will affect components in $\mathcal{A}_0(n)$ and $k - l$ faults will affect components in $\mathcal{S}(n(0))$. It remains to deal with the case in which some fault affects component $c(n)$. The probability that l, $1 \leq l \leq k$ faults will affect components in $\mathcal{B}_1(n) = \{c(n)\} \cup \mathcal{A}_1(n)$ with some of them affecting component $c(n)$ is $\binom{k}{l} q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} (1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} - \binom{k}{l} q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}$

$(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} = \binom{k}{l}(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l}$. With that probability, variable $X_{c(n)}$ will take the value 1, $H_n()$ will be reduced to $H_{n(1)}$, and $k-l$ faults will affect components in $\mathcal{S}(n(1)) = \mathcal{S}(n) - \mathcal{B}_1(n)$. This justifies the recursive expression.

Case h) In this case, $n(0)$ is a terminal node, $n(1)$ is not a terminal node, and $\mathcal{S}(n)$ includes components different from $c(n)$ not in $\mathcal{S}(n(1))$ (the components in $\mathcal{A}_1(n)$). The expression is obtained by conditioning on the number of faults affecting component $c(n)$ from the k faults affecting components in $\mathcal{S}(n)$. The probability that no fault affects component $c(n)$ is $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 0 and $H_n()$ will be reduced to the constant function $\mathbf{1}_{n(0)=b_1}$. The probability that l, $1 \le l \le k$ faults will affect components in $\mathcal{B}_1(n) = \{c(n)\} \cup \mathcal{A}_1(n)$ with some of them affecting component $c(n)$ is $\binom{k}{l} q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} - \binom{k}{l} q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} = \binom{k}{l}(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l}$. With that probability, variable $X_{c(n)}$ will take the value 1, $H_n()$ will be reduced to $H_{n(1)}$, and $k-l$ faults will affect components in $\mathcal{S}(n(1)) = \mathcal{S}(n) - \mathcal{B}_1(n)$. This justifies the recursive expression.

Case i) In this case, $n(0)$ is not a terminal node, $n(1)$ is a terminal node, and $\mathcal{S}(n)$ includes components different from $c(n)$ not in $\mathcal{S}(n(0))$ (the components in $\mathcal{A}_0(n)$). The probability that no fault affects component $c(n)$ is $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 0, $H_n()$ will be reduced to $H_{n(0)}$, and the k faults will affect components in $\mathcal{S}_b(n)$. In that case, with probability $B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}}$, l faults, $0 \le l \le k$ will affect components in $\mathcal{A}_0(n)$ and $k-l$ faults will affect components in $\mathcal{S}(n(0))$. It remains to deal with the case in which some fault affects component $c(n)$. The probability that some fault affects component $c(n)$ is $1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. With that probability, variable $X_{c(n)}$ will take the value 1 and $H_n()$ will be reduced to the constant function $\mathbf{1}_{n(1)=b_1}$. This justifies the recursive expression.

Case j) In this case, both $n(0)$ and $n(1)$ are terminal nodes and $\mathcal{S}(n)$ includes components different from $c(n)$. The probability that no fault affects component $c(n)$ is $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$. Then, with that probability, variable $X_{c(n)}$ will take the value 0 and $H_n()$ will be reduced to the constant function $\mathbf{1}_{n(0)=b_1}$ and, with probability $1-B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}$, variable $X_{c(n)}$ will take the value 1 and $H_n()$ will be reduced to the constant function $\mathbf{1}_{n(1)=b_1}$. This justifies the recursive expression.

□

B. Proof of Theorem 2

The proof is by complete induction on $|\mathcal{S}(n)|$. The result is trivially true for $|\mathcal{S}(n)| = 1$ because this implies $\mathcal{S}(n) = \{c(n)\}$ and, then, from Case a of both Theorem 1 and the theorem, $Z_k(n) = \widetilde{Z}_k(n)$, $k \ge 0$. We will assume now that the result holds for $n \in \mathcal{N}$, $|\mathcal{S}(n)| \le L$, $L \ge 1$, and will show that, for $n \in \mathcal{N}$, $|\mathcal{S}(n)| = L+1$

$$\left|Z_k(n) - \widetilde{Z}_k(n)\right| \le L\varepsilon', \quad k \ge 0. \tag{3}$$

We start by proving (3) for $k = 0$. Since $|\mathcal{S}(n)| > 1$, Case a of the theorem is impossible. In Cases b, c, e, f, g, and i of the theorem, we have, using the corresponding case of Theorem 1 and the induction hypothesis

$$\left|Z_0(n) - \widetilde{Z}_0(n)\right| = \left|Z_0(n(0)) - \widetilde{Z}_0(n(0))\right| \le (L-1)\varepsilon' \le L\varepsilon'.$$

In Cases d, h, and j of the theorem the result is trivial because, using the corresponding cases of Theorem 1, $Z_0(n) = \widetilde{Z}_0(n)$. We prove next (3) for $k > 0$ on a case by case basis, ignoring Case a, which is impossible since $|\mathcal{S}(n)| > 1$.

Case b) Let $S = \sum_{l=1}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}(n(1))$, $\widetilde{S} = \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_{k-l}(n(1))$. Then, using: 1) the induction hypothesis; 2) $0 \le Z_{k-l}() \le 1$, $0 \le l \le k$; 3) $\sum_{l=0}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} = 1$; and 4) $\sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \ge 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \varepsilon'$, which, we note, also holds when $m = 1, M = 0$ because in that case $1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \varepsilon' \le 0$,

$$\begin{aligned}
|S - \widetilde{S}| &\le \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \left|Z_{k-l}(n(1)) - \widetilde{Z}_{k-l}(n(1))\right| \\
&\quad + \sum_{l=1}^{m-1} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}(n(1)) \\
&\quad + \sum_{l=M+1}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}(n(1)) \\
&\le (L-1)\varepsilon' \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} + \sum_{l=1}^{m-1} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \\
&\quad + \sum_{l=M+1}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \\
&= (L-1)\varepsilon' \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} + 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \\
&\quad - \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \\
&\le (L-1)\varepsilon' \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} + \varepsilon'.
\end{aligned} \tag{4}$$

Finally, using: 1) Case b of Theorem 1, 2) the induction hypothesis, 3) inequality (4), and 4) the fact that, with $m \ge 1$ and $M \le k$, $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} + \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \le 1$, we have, for $k > 0$,

$$\begin{aligned}
\left|Z_k(n) - \widetilde{Z}_k(n)\right| &\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \left|Z_k(n(0)) - \widetilde{Z}_k(n(0))\right| \\
&\quad + |S - \widetilde{S}| \\
&\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} (L-1)\varepsilon' + (L-1)\varepsilon' \\
&\quad \times \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} + \varepsilon' \\
&\le (L-1)\varepsilon' + \varepsilon' = L\varepsilon'
\end{aligned}$$

Case c) Let $T = \sum_{l=0}^{k} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} Z_{k-l}(n(0))$, $\widetilde{T} = \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \widetilde{Z}_{k-l}(n(0))$. Then, using: 1) the induction hypothesis, 2) $0 \leq Z_{k-l}() \leq 1$, $0 \leq l \leq k$, 3) $\sum_{l=0}^{k} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} = 1$, which, with $0 \leq m' \leq M' \leq k$, implies $\sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \leq 1$, and 4) $\sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \geq 1 - \varepsilon'/2$,

$$\begin{aligned}
|T - \widetilde{T}| &\leq \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \left| Z_{k-l}(n(0)) - \widetilde{Z}_{k-l}(n(0)) \right| \\
&\quad + \sum_{l=0}^{m'-1} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} Z_{k-l}(n(0)) \\
&\quad + \sum_{l=M'+1}^{k} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} Z_{k-l}(n(0)) \\
&\leq (L-1)\varepsilon' \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \\
&\quad + \sum_{l=0}^{m'-1} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} + \sum_{l=M'+1}^{k} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \\
&\leq (L-1)\varepsilon' + 1 - \sum_{l=m'}^{M'} B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}} \\
&\leq (L-1)\varepsilon' + \varepsilon'/2. \qquad (5)
\end{aligned}$$

Let $S = \sum_{l=1}^{k} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} Z_{k-l}(n(1))$, $\widetilde{S} = \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \widetilde{Z}_{k-l}(n(1))$. Reasoning analogously as we did to obtain (4), we have

$$|S - \widetilde{S}| \leq (L-1)\varepsilon' \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} + \varepsilon'/2. \qquad (6)$$

Finally, using: 1) Case c of Theorem 1, 2) inequalities (5) and (6), and 3) the fact that, with $m \geq 1$ and $M \leq k$, $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} + \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \leq 1$, we have, for $k > 0$,

$$\begin{aligned}
\left| Z_k(n) - \widetilde{Z}_k(n) \right| &\leq B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} |T - \widetilde{T}| + |S - \widetilde{S}| \\
&\leq B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \left((L-1)\varepsilon' + \varepsilon'/2 \right) \\
&\quad + (L-1)\varepsilon' \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} + \varepsilon'/2 \\
&\leq (L-1)\varepsilon' + (\varepsilon'/2)\left(B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} + 1 \right) \\
&\leq (L-1)\varepsilon' + \varepsilon' = L\varepsilon'.
\end{aligned}$$

Case d) Let $S, \widetilde{S}$ be exactly as in Case b (the integers m, M satisfy the same conditions). Then, using: 1) Case d of Theorem 1, 2) inequality (4), and 3) the fact that, with $m \geq 1$ and $M \leq k$, $\sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} \leq 1$, we have, for $k > 0$,

$$\begin{aligned}
\left| Z_k(n) - \widetilde{Z}_k(n) \right| &\leq |S - \widetilde{S}| \leq (L-1)\varepsilon' \sum_{l=m}^{M} B^{k,l}_{q_{c(n)|\mathcal{S}(n)}} + \varepsilon' \\
&\leq (L-1)\varepsilon' + \varepsilon' = L\varepsilon'.
\end{aligned}$$

Case e) Let $U = \sum_{l=1}^{k} \binom{k}{l} (q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} Z_{k-l}(n(1))$, $\widetilde{U} = \sum_{l=m}^{M} \binom{k}{l} (q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} \widetilde{Z}_{k-l}(n(1))$. Then, using: 1) the induction hypothesis, 2) $0 \leq Z_{k-l}() \leq 1$, $0 \leq l \leq k$, 3) that, since $q_{c(n)|\mathcal{S}(n)} = q_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q_{\mathcal{A}_1(n)|\mathcal{S}(n)}$, $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} + \sum_{l=1}^{k} \binom{k}{l} (q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} = 1$, and 4) $\sum_{l=m}^{M} \binom{k}{l} (q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} \geq 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \varepsilon'$, which, we note, also holds when $m = 1$, $M = 0$ because in that case $1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} - \varepsilon' \leq 0$,

$$\begin{aligned}
&|U - \widetilde{U}| \\
&\leq \sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \left| Z_{k-l}(n(1)) - \widetilde{Z}_{k-l}(n(1)) \right| \\
&\quad + \sum_{l=1}^{m-1} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} Z_{k-l}(n(1)) \\
&\quad + \sum_{l=M+1}^{k} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} Z_{k-l}(n(1)) \\
&\leq (L-1)\varepsilon' \sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \\
&\quad + \sum_{l=1}^{m-1} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \\
&\quad + \sum_{l=M+1}^{k} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \\
&= (L-1)\varepsilon' \sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} + 1 - B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} \\
&\quad - \sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} \\
&\leq (L-1)\varepsilon' \sum_{l=m}^{M} \binom{k}{l} \left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)} \right) \\
&\quad \times \left(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l} + \varepsilon'. \qquad (7)
\end{aligned}$$

Finally, using: 1) Case e of Theorem 1, 2) the induction hypothesis, 3) inequality (7), and 4) the fact that, with $m \geq 1$ and $M \leq k$, $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}} + \sum_{l=m}^{M} \binom{k}{l} (q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)} - q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1 - q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l} \leq 1$, we have, for $k > 0$,

$$\begin{aligned}\left|Z_k(n)-\widetilde{Z}_k(n)\right| &\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\left|Z_k\left(n(0)\right)-\widetilde{Z}_k\left(n(0)\right)\right| \\ &\quad + |U-\widetilde{U}| \\ &\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}(L-1)\varepsilon' \\ &\quad + (L-1)\varepsilon'\sum_{l=m}^{M}\binom{k}{l} \\ &\quad \times\left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}-q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \\ &\quad \times\left(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l}+\varepsilon' \\ &\le (L-1)\varepsilon'+\varepsilon'=L\varepsilon'.\end{aligned}$$

Case f) Using Case f of Theorem 1 and the induction hypothesis, we have, for $k>0$,

$$\begin{aligned}\left|Z_k(n)-\widetilde{Z}_k(n)\right| &\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\left|Z_k\left(n(0)\right)-\widetilde{Z}_k\left(n(0)\right)\right| \\ &\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}(L-1)\varepsilon'\le(L-1)\varepsilon'\le L\varepsilon'.\end{aligned}$$

Case g) Let T, $\widetilde{T}$ be exactly as in Case c (the integers m', M' satisfy the same conditions). Furthermore, let $U=\sum_{l=1}^{k}\binom{k}{l}(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}-q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l}Z_{k-l}(n(1))$, $\widetilde{U}=\sum_{l=m}^{M}\binom{k}{l}(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}-q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l}\widetilde{Z}_{k-l}(n(1))$. Reasoning analogously as we did to obtain (7), we have

$$\begin{aligned}|U-\widetilde{U}|\le(L-1)\varepsilon'\sum_{l=m}^{M}\binom{k}{l}\left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}-q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \\ \times\left(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l}+\varepsilon'/2. \quad (8)\end{aligned}$$

Then, using: 1) Case g of Theorem 1, 2) inequalities (5) and (8), and 3) the fact that, with $m\ge 1$ and $M\le k$, $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}+\sum_{l=m}^{M}\binom{k}{l}(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}-q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l}\le 1$, we have, for $k>0$

$$\begin{aligned}\left|Z_k(n)-\widetilde{Z}_k(n)\right| &\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}|T-\widetilde{T}|+|U-\widetilde{U}| \\ &\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\left((L-1)\varepsilon'+\varepsilon'/2\right) \\ &\quad + (L-1)\varepsilon'\sum_{l=m}^{M}\binom{k}{l} \\ &\quad \times\left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}-q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \\ &\quad \times\left(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l}+\varepsilon'/2 \\ &\le (L-1)\varepsilon'+(\varepsilon'/2)\left(B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}+1\right) \\ &\le (L-1)\varepsilon'+\varepsilon'=L\varepsilon'.\end{aligned}$$

Case h) Let $U,\widetilde{U}$ be exactly as in Case e (the integers m, M satisfy the same conditions). Then, using: 1) Case h of Theorem 1; 2) inequality (7); and 3) the fact that, with $m\ge 1$ and $M\le k$, $\sum_{l=m}^{M}\binom{k}{l}(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}-q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)})(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)})^{k-l}\le 1$, we have, for $k>0$

$$\begin{aligned}\left|Z_k(n)-\widetilde{Z}_k(n)\right| &= |U-\widetilde{U}| \\ &\le (L-1)\varepsilon'\sum_{l=m}^{M}\binom{k}{l} \\ &\quad \times\left(q^l_{\mathcal{B}_1(n)|\mathcal{S}(n)}-q^l_{\mathcal{A}_1(n)|\mathcal{S}(n)}\right) \\ &\quad \times\left(1-q_{\mathcal{B}_1(n)|\mathcal{S}(n)}\right)^{k-l}+\varepsilon' \\ &\le (L-1)\varepsilon'+\varepsilon'=L\varepsilon'.\end{aligned}$$

Case i) Let $T=\sum_{l=0}^{k}B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}}Z_{k-l}(n(0))$, $\widetilde{T}=\sum_{l=m'}^{M'}B^{k,l}_{q_{\mathcal{A}_0(n)|\mathcal{S}_b(n)}}\widetilde{Z}_{k-l}(n(0))$. Reasoning analogously as we did to obtain (5), we have

$$|T-\widetilde{T}|\le(L-1)\varepsilon'+\varepsilon'=L\varepsilon'.$$

Then, using: 1) Case i of Theorem 1, 2) the previous inequality, and 3) $B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}\le 1$, we have, for $k>0$

$$\left|Z_k(n)-\widetilde{Z}_k(n)\right|\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}|T-\widetilde{T}|\le B^{k,0}_{q_{c(n)|\mathcal{S}(n)}}L\varepsilon'\le L\varepsilon'.$$

Case j) The result is immediate because, using Case j of Theorem 1, $Z_k(n)=\widetilde{Z}_k(n)$, $k>0$. □

C. Proof of Lemma 1

From the assumption $\mathcal{A}_1(n)$, $\mathcal{S}(n(1))\neq\emptyset$ it is clear that both $q_1=q_{\mathcal{B}_1(n)|\mathcal{S}(n)}$ and $q_2=q_{\mathcal{A}_1(n)|\mathcal{S}(n)}$ are >0 and <1. Further, since $\mathcal{A}_1(n)=\mathcal{B}_1(n)-\{c(n)\}$, $q_2<q_1$. Therefore, for $1<l\le k$ we can write

$$\begin{aligned}\frac{R^{k,l}_{q_1,q_2}}{R^{k,l-1}_{q_1,q_2}} &= \frac{k-l+1}{l}\frac{q_1^l-q_2^l}{q_1^{l-1}-q_2^{l-1}}\frac{1}{1-q_1} \\ &= \frac{k-l+1}{l}\frac{q_1}{1-q_1}\frac{1-(q_2/q_1)^l}{1-(q_2/q_1)^{l-1}}.\end{aligned}$$

Using elementary analysis techniques it is straightforward to show that for l integer >1 and $0<x<1$, the function $f(x)=(1-x^l)/(1-x^{l-1})$ increases on x and tends to $l/(l-1)$ as x approaches 1. Then, since $0<q_2/q_1<1$

$$1<\frac{1-(q_2/q_1)^l}{1-(q_2/q_1)^{l-1}}<\frac{l}{l-1}$$

implying that, for $1<l\le k$,

$$\begin{aligned}\frac{R^{k,l}_{q_1,q_2}}{R^{k,l-1}_{q_1,q_2}} &> \frac{k-l+1}{l}\frac{q_1}{1-q_1}=1+\frac{(k+1)q_1-l}{l(1-q_1)} \\ \frac{R^{k,l}_{q_1,q_2}}{R^{k,l-1}_{q_1,q_2}} &< \frac{k-l+1}{l}\frac{q_1}{1-q_1}\frac{l}{l-1}=1+\frac{kq_1+1-l}{(l-1)(1-q_1)}.\end{aligned}$$

Accordingly, the term $R^{k,l}_{q_1,q_2}$, $1<l\le k$ is larger than the preceding one if $l\le(k+1)q_1$ and smaller if $l\ge kq_1+1$. This proves that the terms increase monotonically as l goes from 1 to $\lfloor(k+1)q_1\rfloor$ and decrease monotonically as l goes from $\lceil kq_1+1\rceil-1$ to k. □

REFERENCES

[1] J.-C. Lo, C. Metra, and F. Lombardi, "Guest editors' introduction: Special section on design and test of systems-on-chip (SoC)," *IEEE Trans. Computers*, vol. 55, no. 2, pp. 97–98, Feb. 2006.

[2] J.-F. Frigon, A. M. Eltawil, E. Grayver, A. Tarighat, and H. Zou, "Design and implementation of a baseband WCDMA dual-antenna mobile terminal," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 3, pp. 518–529, Mar. 2007.

[3] D. Kim, K. Chung, C.-H. Yu, C.-H. Kim, I. Lee, J. Bae, Y.-J. Kim, J.-H. Park, S. Kim, Y.-H. Park, N.-H. Seong, J.-A. Lee, J. Park, S. Oh, S.-W. Jeong, and L.-S. Kim, "An SoC with 1.3 Gtexels/s 3-D graphics full pipeline for consumer applications," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 71–84, Jan. 2006.

[4] A. Lodi, A. Cappelli, M. Bocchi, C. Mucci, M. Innocenti, C. De Bartolomeis, L. Ciccarelli, R. Giansante, A. Deledda, F. Campi, M. Toma, and R. Guerrieri, "XiSystem: A XiRisc-based SoC with reconfigurable IO module," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 85–96, Jan. 2006.

[5] Y. Zorian and D. Gizopoulos, "Guest editors' introduction: Design for yield and reliability," *IEEE Des. Test. Comput.*, vol. 21, no. 3, pp. 177–182, Mar. 2004.

[6] F. J. Meyer and N. Park, "Predicting defect-tolerant yield in the embedded core context," *IEEE Trans. Comput.*, vol. 52, no. 11, pp. 1470–1479, Nov. 2003.

[7] Y.-Y. Chen and S. J. Upadhyaya, "Yield analysis of reconfigurable array processors based on multiple-level redundancy," *IEEE Trans. Computers*, vol. 42, no. 9, pp. 1136–1141, Sep. 1993.

[8] I. Koren and Z. Koren, "Analysis of a hybrid defect-tolerance scheme for high-density memory ICs," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 1997, pp. 166–174.

[9] I. Koren and Z. Koren, "Defect tolerance in VLSI circuits: Techniques and yield analysis," *Proc. IEEE*, vol. 86, no. 9, pp. 1819–1838, Sep. 1998.

[10] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[11] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surveys*, vol. 38, no. 1, 2006, Article 1.

[12] F. Angiolini, P. Meloni, S. M. Carta, L. Raffo, and L. Benini, "A layout-aware analysis of networks-on-chip and traditional interconnects for MPSoCs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 3, pp. 421–434, Mar. 2007.

[13] C. H. Stapper, "On yield, fault distributions, and clustering of particles," *IBM J. Res. Develop.*, vol. 30, no. 3, pp. 326–338, 1986.

[14] D. Nikolos and H. T. Vergos, "On the yield of VLSI processors with on-chip CPU cache," *IEEE Trans. Comput.*, vol. 48, no. 10, pp. 1138–1144, Oct. 1999.

[15] J. A. Carrasco and V. Suñé, "Combinatorial methods for the evaluation of yield and operational reliability of fault-tolerant systems-on-chip," *Microelectron. Reliab.*, vol. 44, pp. 339–350, 2004.

[16] C. H. Stapper and R. J. Rosner, "Integrated circuit yield management and yield analysis: Development and implementation," *IEEE Trans. Semicond. Manuf.*, vol. 8, no. 2, pp. 95–102, Feb. 1995.

[17] I. A. Wagner and I. Koren, "An interactive VLSI CAD tool for yield estimation," *IEEE Trans. Semicond. Manuf.*, vol. 8, no. 2, pp. 130–138, Feb. 1995.

[18] G. A. Allan, "Yield prediction by sampling IC layout," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 19, no. 3, pp. 359–371, Mar. 2000.

[19] J. Cunningham, "The use and evaluation of yield models in integrated circuit manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 3, no. 2, pp. 60–71, Feb. 1990.

[20] T. Okabe, M. Nagata, and S. Shimada, "Analysis on yield of integrated circuits and a new expression for the yield," *Elec. Eng. Japan*, vol. 92, no. 6, pp. 135–141, 1972.

[21] C. H. Stapper, "Defect density distribution for LSI yield calculations," *IEEE Trans. Electron Devices*, vol. 20, no. 7, pp. 655–657, Jul. 1973.

[22] I. Koren, Z. Koren, and C. H. Stapper, "A unified negative-binomial distribution for yield analysis of defect-tolerant circuits," *IEEE Trans. Comput.*, vol. 42, no. 6, pp. 724–734, Jun. 1993.

[23] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Computers*, vol. C-35, no. 8, pp. 677–691, Aug. 1986.

[24] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed. New York: Wiley, 1968, vol. I.

[25] Univ. Colorado, Boulder, "CUDD: CU Decision Diagram Package, Release 2.4.1," 2007. [Online]. Available: vlsi.colorado.edu/~fabio/CUDD

[26] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The Nostrum backbone—A communication protocol stack for networks on chip," in *Proc. 17th Int. Conf. VLSI Des.*, 2004, pp. 693–696.

[27] D. Wiklund and D. Liu, "SoCBUS: Switched network on chip for hard real time embedded systems," in *Proc. Int. Parallel Distrib. Process. Symp.*, 2003.

[28] C. A. Zeferino and A. A. Susin, "SoCIN: A parametric and scalable network-on-chip," in *Proc. 16th Symp. Integr. Circuits Syst. Des. (SBCCI)*, 2003, pp. 169–174.

[29] A. Andriahantenaia and A. Greiner, "Micro-network for SoC: Implementation of a 32-port SPIN network," in *Proc. Conf. Des., Autom. Test Europe (DATE)*, 2003, p. 1128.

[30] F. Petrini and M. Vanneschi, "Performance analysis of wormhole routed k-ary n-trees," *Int. J. Foundations Comput. Sci.*, vol. 9, no. 2, pp. 157–177, 1998.

[31] S. Rai and Y. C. Oh, "Tighter bounds on full access probability in fault-tolerant multistage interconnection networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 3, pp. 328–335, Mar. 1999.

[32] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," in *Proc. 27th ACM/IEEE Des. Autom. Conf.*, 1990, pp. 52–57.

[33] M. Bouissou, F. Bruyére, and A. Rauzy, "BDD based fault-tree processing: A comparison of variable ordering heuristics," in *Proc. Europ. Safety Reliab. Assoc. Conf. (ESREL)*, C. G. Soares, Ed., 1997, vol. 3, pp. 2045–2052.

[34] I. Wegener, "The size of reduced OBDD's and optimal read-once branching programs for almost all Boolean functions," *IEEE Trans. Computers*, vol. 43, no. 11, pp. 1262–1269, Nov. 1994.

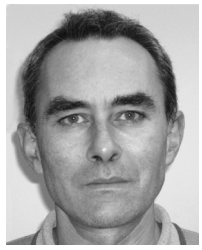
[35] S. A. Doyle and J. B. Dugan, "Dependability assessment using binary decision diagrams (BDDS)," in *Proc. 25th Int. Symp. Fault-Tolerant Comput. (FTCS-25)*, 1995, pp. 249–258.

[36] R. Piessens, E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner, *Quadpack: A Subroutine Package for Automatic Integration*. New York: Springer Verlag, 1983, vol. 1, Series in Computational Mathematics.

Juan A. Carrasco (SM'02) received the Engineer degree in industrial engineering and the Ph.D. degree in industrial engineering from the Polytechnical University of Catalonia (UPC), Barcelona, Spain, in 1982 and 1987, respectively, and the M.Sc. degree in computer science from Stanford University, Stanford, CA, in 1987.

He is currently an Associate Professor with the Electronics Engineering Department, UPC. He visited INRIA (CNRS), Rennes, France, twice, in 1996 and 1998. His research is focused on the development of methodologies for the modeling and evaluation of fault-tolerant systems, a topic in which he has published 60 papers in refereed journals and conference proceedings. He has directed the design and implementation of METFAC-2, a Markovian modeling tool (see http://www.dit.upc.es/qine/tools/metfac/). He has been the principal investigator of several research projects funded by both public and private institutions, has been in the program committees of 11 international conferences, is a research project evaluator of the Spanish and Catalonian research project evaluation agencies, and is a reviewer of *ACM Computer Reviews*.

Prof. Carrasco is a senior member of the IEEE Computer Society.



Víctor Suñé received the engineer degree in industrial engineering and the Ph.D. degree in industrial engineering from the Polytechnical University of Catalonia (UPC), Barcelona, Spain, in 1995 and 2000, respectively.

He is currently a Lecturer with the Electronics Engineering Department, UPC. He visited Duke University in 2006. His research is focused on the development of methodologies for the modeling and evaluation of fault-tolerant systems, a topic in which he has published 13 papers in refereed journals and conference proceedings. He is co-designer and co-implementor of METFAC-2, a Markovian modeling tool (see http://www.dit.upc.es/qine/tools/metfac/). He has participated in a number of research projects funded by public institutions.