Combinatorial Methods for the Evaluation of Yield and Operational Reliability of Fault-Tolerant Systems-on-Chip*

Juan A. Carrasco and Víctor Suñé
Departament d'Enginyeria Electrònica
Universitat Politècnica de Catalunya
Diagonal 647, plta. 9
08028 Barcelona, Spain

Except for formatting details and the correction of some errata, this version matches exactly the version published with the same title and authors in *Microelectronics Reliability*, vol. 44, no. 2, 2004, pp. 339–350

Abstract

In this paper we develop combinatorial methods for the evaluation of yield and operational reliability of fault-tolerant systems-on-chip. The method for yield computation assumes that defects are produced according to a model in which defects are lethal and affect given components of the system following a distribution common to all defects; the method for the computation of operational reliability also assumes that the fault-tree function of the system is increasing. The distribution of the number of defects is arbitrary. The methods are based on the formulation of, respectively, the yield and the operational reliability as the probability that a given boolean function with multiple-valued variables has value 1. That probability is computed by analyzing a ROMDD (reduced ordered multiple-value decision diagram) representation of the function. For efficiency reasons, a coded ROBDD (reduced ordered binary decision diagram) representation of the function is built first and, then, that coded ROBDD is transformed into the ROMDD required by the methods. We present numerical experiments showing that the methods are able to cope with quite large systems in moderate CPU times.

*This paper is an extended version of D. P. Munteanu, V. Suñé, R. Rodríguex-Montañés, and J. A. Carrasco, "A Combinatorial Method for the Evaluation of Yield of Fault-Tolerant Systems-on-Chip," in Proc. IEEE Int. Conf. on Dependable Systems and Networks (DSN2003), San Francisco, June 2003, pp. 563–572.

1 Introduction

Systems-on-chip are becoming popular. The high densities and areas of those integrated systems make them very susceptible to manufacturing defects. In fact, complex systems-on-chip are likely to have a very small yield and may have a small operational reliability if they are not designed with built-in fault-tolerance. Then, there is a need for efficient methodologies for estimating the yield and operational reliability of complex fault-tolerant systems-on-chip. When the fault-tolerant system-on-chip has a regular structure, it is often possible to make "ad-hoc" evaluations (see, for instance, [12, 13, 18, 19]). However, many fault-tolerant designs do not have a regular structure, particularly those using a sophisticated network-on-chip as a communication subsystem among the intellectual property cores (IPs) [4]. Computing the yield and operational reliability of such systems-on-chip is difficult, mainly because the fact that realistic defect distributions have clustering [8, 14, 15, 16, 17, 19] and, thus, introduce dependencies among the initial failed states of the components of the system (see, for instance, [19, 28]). Simulation is an approach which is not severely limited by the complexity of the system, but tends to be expensive and does not provide strict error control. The aim of this paper is to develop combinatorial methods for the evaluation of the yield and operational reliability of fault-tolerant systems-on-chip with precise error control which can cope with quite complex systems using currently affordable computational resources.

We assume that the fault-tolerant system-on-chip 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 the value 1 if and only if component i is failed and the function takes the value 1 if and only if the system is not functioning. In the combinatorial method for yield evaluation, no restriction will be imposed on $F(x_1, \ldots, x_C)$. In the combinatorial method for the evaluation of the operational reliability, it will be assumed that $F()$ is increasing. It will be assumed that a gate-level description of the function is available. A way to ensure that $F()$ is increasing is to allow only the use of AND and OR gates in the gate-level description.

The production of manufacturing defects will be modeled using the following probabilities:

$$Q_k = P[\text{number of manufacturing defects is } k], \quad k = 0, 1, 2, \ldots\ ,$$

$$P_i = P[\text{any given defect affects component } i \text{ and is lethal}]\,,$$

where a defect is lethal if it causes the component not to function properly. We emphasize that it is assumed that all defects will be distributed over the components making up the system and will be lethal following the probabilities P_i, $1 \leq i \leq C$, independently of the number of defects, of which components affect the remaining defects and of whether those defects are lethal or not. 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 easily provided by the manufacturer of the system-on-chip and the probabilities P_i, $1 \leq i \leq C$ could be estimated from the final layout of the system-on-chip using appropriate tools [20, 22, 32, 33] or from IP layouts and routing estimates [31]. Thus, the methodologies could be used at several design stages. The assumed model is consistent with all

compound Poisson yield models [19], which include the widely used negative binomial distribution for the number of defects. The assumed model will not be consistent however with yield models accounting for spatial clustering[1] such as the one proposed in [23].

From a computational point of view, it is convenient to map the previously described model into a model taking into account only lethal manufacturing defects, i.e. defects which effectively make some component of the system to be defective (not to work properly). That model includes the probabilities:

$$Q'_k = P[\text{number of lethal manufacturing defects is } k], \quad k = 0, 1, 2, \dots ,$$

$$P'_i = P[\text{any given lethal defect affects component } i] .$$

The reason why the last model is computationally more convenient is basically because, since not all defects will be lethal, the distribution Q'_k, $k = 0, 1, 2, \dots$ will be shifted to lower values of k in relation to the distribution Q_k, $k = 0, 1, 2, \dots$ and, then, if only up to M defects are analyzed (the computational cost of the methods will increase with M), higher accuracy will be obtained if the distribution Q'_k, $k = 0, 1, 2, \dots$ is used instead of Q_k, $k = 0, 1, 2, \dots$. The mapping can be performed using:

$$Q'_k = \sum_{m=k}^{\infty} Q_m \binom{m}{k} P_L^k (1 - P_L)^{m-k} , \tag{1}$$

$$P'_i = \frac{P_i}{P_L} ,$$

where $P_L = \sum_{i=1}^{C} P_i$ is the probability that any given defect is lethal. As previously commented, the negative binomial distribution is the most widely used distribution for the number of defects affecting a chip. That distribution has the form:

$$Q_k = \frac{\Gamma(\alpha + k)}{k!\Gamma(\alpha)} \frac{(\lambda/\alpha)^k}{(1 + \lambda/\alpha)^{\alpha+k}} , \tag{2}$$

where λ is the expected number of defects and α is the clustering parameter (the clustering increases for decreasing α). It is known (see [16]) that, when the distribution of the number of defects is negative binomial, the distribution of the number of lethal defects is also negative binomial with the same clustering parameter. More precisely, when the distribution of the number of defects is given by (2), the distribution of the number of lethal defects is:

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

with $\lambda' = P_L \lambda$. Similar results hold for all compound Poisson distributions [19].

The production of operational faults will be modeled using operational reliability functions $R_i(t)$, $1 \leq i \leq C$, where $R_i(t)$ is the probability that, assuming that component i is not affected by any lethal defect (and, therefore, it is unfailed at time 0), it will not have failed by time t. We allow

[1] Spatial clustering refers to the fact that irrespectively of the expected number of defects on the system-on-chip, defects tend to cluster spatially.

arbitrary decreasing operational reliability functions with $R_i(0) = 1$. In addition, we will model coverage failures by introducing coverage parameters C_i, $1 \leq i \leq C$. Upon failure of component i, the system fails with probability $1 - C_i$, irrespectively of the subset of components which were failed before.

2 Method for Yield Computation

In the method the yield loss, L, is computed analyzing whether the system is initially (right after manufacturing) functioning or not assuming $0, 1, 2, \ldots, M$ lethal defects. Let

$$L_k = P[\text{system is finitially not unctioning} \mid \text{there are } k \text{ lethal defects}] \,.$$

We have

$$L = \sum_{k=0}^{\infty} Q'_k L_k \,.$$

Analyzing up to M defects we can pessimistically estimate L by

$$L^M = \sum_{k=0}^{M} Q'_k L_k + \sum_{k=M+1}^{\infty} Q'_k$$

with error bounded from above by $\sum_{k=M+1}^{\infty} Q'_k = 1 - \sum_{k=0}^{M} Q'_k$. Then, given a suitable error control parameter ε, we can select

$$M = \min \left\{ m \geq 0 \ : \ 1 - \sum_{k=0}^{m} Q'_k \leq \varepsilon \right\} ,$$

guaranteeing and absolute error in the yield loss estimation $\leq \varepsilon$.

The yield loss estimate L^M can be formalized as the probability that a Boolean function of certain independent integer-valued random variables is equal to 1. Assume that the defects are numbered in some arbitrary order. Those random variables are:

$$W = \begin{cases} k, 0 \leq k \leq M & \text{if there are } k \text{ lethal defects} \\ M+1 & \text{if there are more than } M \text{ lethal defects} \end{cases}$$

and, for $1 \leq k \leq M$,

$$V_k = i \quad \text{if the } k\text{th lethal defect affects component } i \,.$$

Note that the random variable W takes values in $\{0, 1, \ldots, M+1\}$ and each random variable V_k takes values in $\{1, 2, \ldots, C\}$. The random variable W has probability distribution $P[W = k] = Q'_k$, $0 \leq k \leq M$, $P[W = M+1] = 1 - \sum_{k=0}^{M} Q'_k$. The random variables V_k have probability distributions $P[V_k = i] = P'_i$, $1 \leq k \leq M$, $1 \leq i \leq C$.

Let $I_k(x)$ denote the boolean function with integer-valued variable x returning the value 1 if $x = k$ and the value 0 otherwise and let $I_{\geq l}(x)$ denote the boolean function with integer-valued variable x returning the value 1 if $x \geq l$ and the value 0 otherwise. Let the boolean function

$$G(w, v_1, \ldots, v_M) = I_{M+1}(w) \vee \;\; F\left(\bigvee_{l=1}^{M} I_{\geq l}(w) \wedge I_1(v_l), \ldots, \bigvee_{l=1}^{M} I_{\geq l}(w) \wedge I_C(v_l)\right). \quad (3)$$

Then, we have the following result.

Theorem 1. $L^M = P[G(W, V_1, V_2, \ldots, V_M) = 1]$.

Intuitively, the reason why Theorem 1 holds is that $I_{M+1}(W)$ "tells" whether the number of lethal defects is $> M$, $I_{\geq l}(W)$ "tells" whether there is a lth lethal defect, $I_i(V_l)$ "tells" whether the lth lethal defect affects component i and, then, $\bigvee_{l=1}^{M} I_{\geq l}(W) \wedge I_i(V_l)$ "tells" whether component i is affected by some of the first M lethal defects. A formal proof follows.

Proof of Theorem 1 The quantity L_k is the probability that given there are k lethal defects the system is not functioning. Since, assuming there are k lethal defects, component i is failed if and only if $\bigvee_{l=1}^{k} I_i(V_l) = 1$, we have

$$L_k = P\left[F\left(\bigvee_{l=1}^{k} I_1(V_l), \ldots, \bigvee_{l=1}^{k} I_C(V_l)\right) = 1\right]. \quad (4)$$

Using the theorem of total probability and the independence of the random variables $W, V_1, \ldots, V_M$:

$$\begin{aligned} P[G(W, V_1, \ldots, V_M) = 1] &= \sum_{k=0}^{M+1} P[W = k] P[G(W, V_1, \ldots, V_M) = 1 \mid W = k] \\ &= \sum_{k=0}^{M} Q'_k P[G(k, V_1, \ldots, V_M) = 1] \\ &\quad + \left(1 - \sum_{k=0}^{M} Q'_k\right) P[G(M+1, V_1, \ldots, V_M) = 1]. \quad (5) \end{aligned}$$

But, from the definition of G (3), for $0 \leq k \leq M$:

$$G(k, v_1, \ldots, v_M) = F\left(\bigvee_{l=1}^{k} I_1(v_l), \ldots, \bigvee_{l=1}^{k} I_C(v_l)\right) \quad (6)$$

and

$$G(M+1, v_1, \ldots, v_M) = 1. \quad (7)$$

Then, using (4)–(7)

$$P[G(W, V_1, \ldots, V_M) = 1] = \sum_{k=0}^{M} Q'_k L_k + \sum_{k=M+1}^{\infty} Q'_k = L^M. \quad \square$$

In the method, the probability $P[G(W, V_1, \ldots, V_M) = 1]$ is computed building an ROMDD (reduced ordered multiple-valued decision diagram) of the function $G(w, v_1, \ldots, v_M)$. ROMDDs

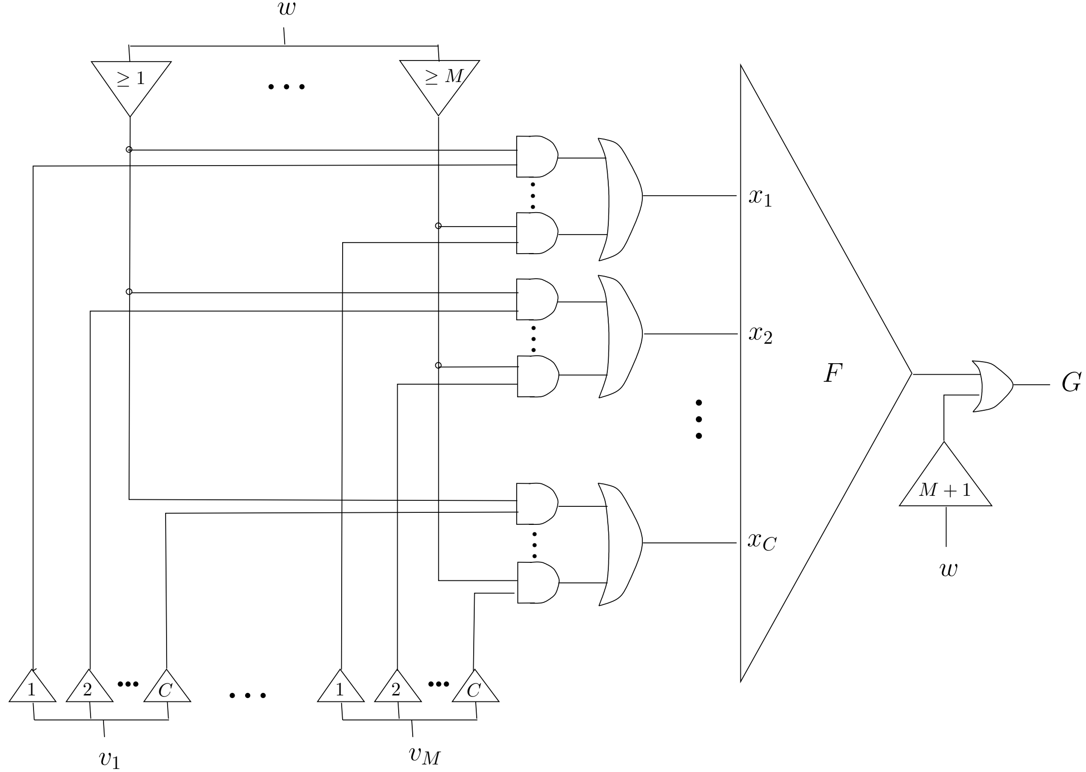

Figure 1: Gate-level description of the function $G(w, v_1, \ldots, v_M)$.

are a natural extension of the well-known ROBDDs (reduced ordered binary decision diagrams) [6] in which both the variables and the function are allowed to be multiple-valued. A gate-level representation of the function $G(w, v_1, \ldots, v_M)$ can be obtained from a gate-level representation of $F(x_1, \ldots, x_C)$ as shown in Fig. 1, where the gate labeled i inside is a "filter" gate returning the value 1 if its integer-valued input has value i and returning the value 0 otherwise and the gate labeled $\geq i$ inside is a "filter" gate returning the value 1 if its integer-valued input has value $\geq i$ and returning the value 0 otherwise. As ROBDDs, ROMDDs are canonical representations which can be built and manipulated in a similar way as ROBDDs can. A ROMDD representing a function H, which can take values in the set S_H, of variables x_i, $i = 1, 2, \ldots, n$, which can take values in the sets S_i is a directed acyclic graph with up to $|S_H|$ terminal nodes each labeled with a distinct value of the set S_H. Every non-terminal node is labeled by an input variable x_i and has as many as $|S_i|$ edges, each labeled by a subset of S_i, with subsets associated with different edges being non-intersecting. The ROMDD has a unique non-terminal node without incoming edges, representing the function $H(x_1, \ldots, x_n)$, called the top node. The input variables encountered in every path from the top node to a terminal node form a sequence of non-repeating input variables consistent with an ordering $x_{p(1)}, \ldots, x_{p(n)}$ of the input variables of the function. Every non-terminal node of the ROMDD represents a unique function of the set of input variables not before the input variable associated with the node in the ordering $x_{p(1)}, \ldots, x_{p(n)}$. That a ROMDD is a canonical representation means that, given H, the ROMDD only depends on the selected ordering $x_{p(1)}, \ldots, x_{p(n)}$ for the multiple-valued variables.

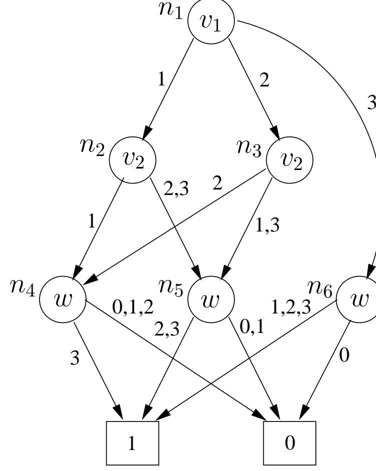

Figure 2: Small ROMDD to illustrate the computation of $P[G(W, V_1, \ldots, V_M) = 1]$.

Using the fact that the random variables $W, V_1, \ldots, V_M$ are independent and that the function represented by a non-terminal node only depends on the set of variables not before the variable associated with the node in the considered ordering for the inpute variables, it is possible to compute $P[G(W, V_1, \ldots, V_M) = 1]$ from an ROMDD representation of the function $G(w, v_1, \ldots, v_M)$. This can be achieved by assigning the value 1 to the terminal node labeled "1" and the value 0 to the terminal node labeled "0", making a depth-first, left-most traversal [1] of the ROMDD, and computing the probability that the function represented by a non-terminal node has value 1 when returning from each non-terminal node. Assume that node n has associated with it the variable w, that $M = 4$, and that n has edges to nodes n_1, n_2 and n_3 with subsets of values of w $\{0, 1\}$, $\{3\}$ and $\{2, 4, 5\}$, respectively. Then, denoting by $\text{value}(x)$ the "value" variable associated with node x, when returning from node n, $\text{value}(n)$ would be computed as $(P[W = 0] + P[W = 1]) \times \text{value}(n_1) + P[W = 3] \times \text{value}(n_2) + (P[W = 2] + P[W = 4] + P[W = 5]) \times \text{value}(n_3)$. At the end of the traversal, the "value" variable of the top node will hold $P[G(W, V_1, \ldots, V_M) = 1]$. We illustrate the computational procedure with the small ROMDD shown in Fig. 2 which corresponds to a fault-tolerant system having fault-tree function $F(x_1, x_2, x_3) = x_1x_2 + x_3$ and $M = 2$ under the multiple-valued variable ordering v_1, v_2, w. This implies that the random variable W will take values in the set $\{0, 1, 2, 3\}$ and the random variables V_1 and V_2 will take values in the set $\{1, 2, 3\}$. Using a depth-first, left-most traversal of the ROMDD, $P[G(W, V_1, V_2) = 1] = \text{value}(n_1)$ would be computed following the sequence:

$$
\begin{aligned}
&\text{value}(n_4) = Q'_3\,,\\
&\text{value}(n_5) = Q'_2 + Q'_3\,,\\
&\text{value}(n_2) = P'_1 \times \text{value}(n_4) + (P'_2 + P'_3) \times \text{value}(n_5)\,,\\
&\text{value}(n_3) = P'_2 \times \text{value}(n_4) + (P'_1 + P'_3) \times \text{value}(n_5)\,,\\
&\text{value}(n_6) = Q'_1 + Q'_2 + Q'_3\,,\\
&P[G(W, V_1, V_2) = 1] = \text{value}(n_1) = P'_1 \times \text{value}(n_2) + P'_2 \times \text{value}(n_3) + P'_3 \times \text{value}(n_6)\,.
\end{aligned}
$$

Although there are algorithms and packages for ROMDD manipulation [24, 30], there is currently consensus in the ROMDD community that the most efficient way for analyzing

multiple-valued functions of multiple-valued variables is by using coded ROBDDs [24, 25]. A coded ROBDD of a multiple-valued function $H(x_1, x_2, \ldots, x_n)$ of multiple-valued variables x_i is the ROBDD of any function $H'(x_{1,1}, \ldots, x_{1,k_1}, x_{2,1}, \ldots, x_{2,k_2}, \ldots, x_{n,1}, \ldots, x_{n,k_n})$ which represents $H(x_1, x_2, \ldots, x_n)$ in terms of groups $x_{i,1}, x_{i,2}, \ldots, x_{i,k_i}$ of binary variables encoding the multiple-valued variables x_i. Formally, denoting by S_i the domain of x_i and by $x_{i,1}(j), \ldots, x_{i,k_i}(j)$ the codeword representing value $j \in S_i$ in the code used for x_i, H' has to satisfy $H'(x_{1,1}(j_1), \ldots, x_{1,k_1}(j_1), x_{2,1}(j_2), \ldots, x_{2,k_2}(j_2), \ldots, x_{n,1}(j_n), \ldots, x_{n,k_n}(j_n)) = H(j_1, \ldots, j_n)$ for every $(j_1, \ldots, j_n) \in S_1 \times S_2 \times \cdots \times S_n$.

Coded ROBDDs can be used directly in many applications such as formal verification. However, the combinatorial method for yield computation requires the availability of the ROMDD. Assume that the function represented by the coded ROBSS is Boolean. Given an ordering $x_{p(1)}, \ldots, x_{p(n)}$ of the multiple-valued variables, the ROMDD can be efficiently obtained from a coded ROBDD if the coded ROBDD is obtained using an ordering for the binary variables in which the variables encoding each multiple-valued variable are kept grouped and the groups are ordered according to the ordering $x_{p(1)}, \ldots, x_{p(n)}$. The conversion procedure is based on viewing the coded ROMDD as made up of non-terminal layers and a terminal layer, where each non-terminal layer contains the nodes with binary variables encoding a given multiple-valued variable. The terminal layer includes the terminal node b_0 with value 0 and the terminal node b_1 with value 1. Some of the nodes in each layer are entry nodes (they have incoming arcs from other layers). The procedure builds incrementally the ROMDD by processing bottom-up each layer of the coded ROBDD. Processed entry nodes of the coded ROBDD are associated with nodes of the constructed ROMDD. Let $\mathrm{mapping}(n)$ be the node of the ROMDD associated with the entry node n of the coded ROBDD. The bottom (terminal) layer of the coded ROBDD is processed by creating a copy b'_0 in the ROMDD of the terminal node b_0 of the coded ROBDD and creating a copy b'_1 in the ROMDD of the terminal node b_1 and making $\mathrm{mapping}(b_0) = b'_0$ and $\mathrm{mapping}(b_1) = b'_1$. The remaining layers of the coded ROBDD are processed by processing each entry node n of the layer as follows. For each possible value i of the multiple-valued variable x associated with the layer, it is determined which entry node of a different (down) layer is reached from n when the values of the group of binary variables associated with the layer encoding value i are followed. Let $n_{s(i)}$ be the node of the coded ROBDD reached when the value i is “simulated”. If all $\mathrm{mapping}(n_{s(i)})$ are equal to some node n' of the ROMDD, then $\mathrm{mapping}(n)$ must be made equal to n' and no node has to be added to the ROMDD. Otherwise, the ROMDD must have a node associated with n with multiple-valued variable x. That node must have successor $\mathrm{mapping}(n_{s(i)})$ for each value i of x. If there exists in the ROMDD some node n' with multiple-valued variable x and successor $\mathrm{mapping}(n_{s(i)})$ for each value i of x, then, $\mathrm{mapping}(n)$ is set to n' and no node is added to the ROMDD. Otherwise, a node n' with multiple-valued variable x and successor $\mathrm{mapping}(n_{s(i)})$ for each value i of x is added to the ROMDD and $\mathrm{mapping}(n)$ is set to n'. When not all combinations of values of the groups of binary variables encode values in the domain of the associated multiple-valued variable, the ROMDD built in that way may have nodes which are unreachable from the top node. Such nodes are identified and deleted by making a depth-first, left-most traversal of the ROMDD starting from the top node. Fig. 3 illustrates the processing of a layer of a coded ROBDD associated with a multiple-valued variable x which

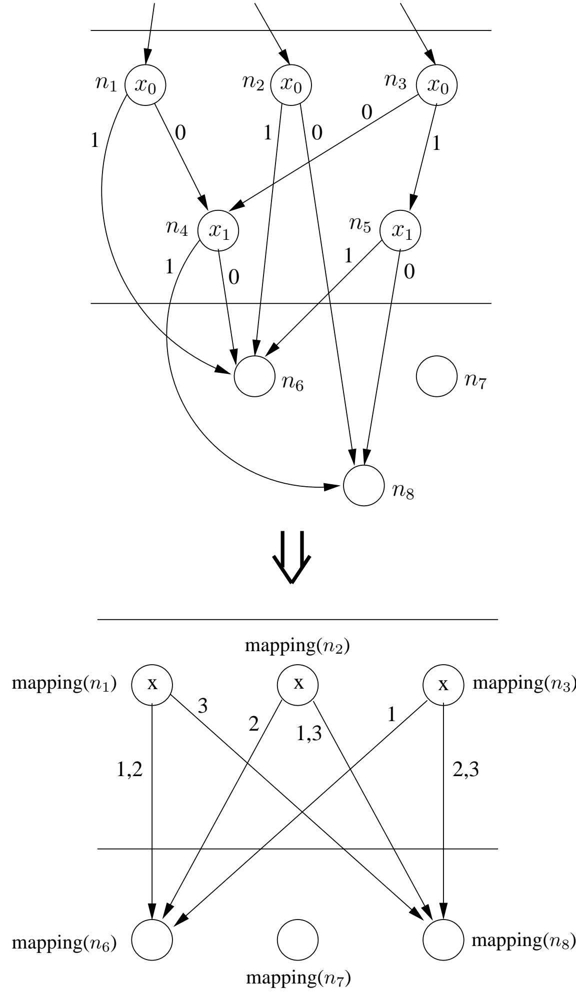

Figure 3: Illustration of the procedure for obtaining the ROMDD from the coded ROBDD

takes values in the domain $\{1, 2, 3\}$ and in which two binary variables x_1, x_0 have been used to encode variable x using the code $1 = 00$, $2 = 01$ and $3 = 10$.

The coded ROBDD used in the method for yield computation is built by processing an implementation of the function $G(w, v_1, \ldots, v_M)$ in binary logic obtained by encoding the variable w in binary using a minimum number of bits. For the variables v_i, since they have values in the domain $\{1, 2, \ldots, C\}$, a binary code of minimum number of bits encoding $v_i - 1$ is used. Such strategy keeps minimum the number of binary variables and tends to result in coded ROBDDs of minimum size. Filter gates are substituted by binary logic expressed in terms of the binary variables $w_{l'}, \ldots w_0$ encoding the multiple-valued variable w and the binary variables $v_i^l \ldots v_i^0$ encoding each multiple-valued variable v_i. Although, with given orderings of the variables, the coded ROBDD and the ROMDD will be independent on the particular implementation of that logic, that implementation may affect the heuristic-based orderings. This makes convenient to report which logic is used. Calling z_k, $1 \leq k \leq M$, the output of the "filter" gate labeled $\geq k$ having as input w, and calling z_{M+1} the output of the "filter" gate labeled $M + 1$ having as input w, the binary logic used for generating

z_k, $1 \leq k \leq M+1$ is

$$z_{M+1} = \text{lit}(w_{l'}, M+1) \wedge \text{lit}(w_{l'-1}, M+1) \wedge \cdots \wedge \text{lit}(w_0, M+1)\,,$$

$$z_k = z_{k+1} \vee \text{lit}(w_{l'}, k) \wedge \text{lit}(w_{l'-1}, k) \wedge \cdots \wedge \text{lit}(w_0, k)\,, \quad 1 \leq k \leq M\,,$$

where $\text{lit}(w_i, m) = w_i$ if the ith bit of the binary code representing m is 1 and $\text{lit}(w_i, m) = \overline{w}_i$ if it is 0, where $\overline{x}$ denotes the complement of the binary variable x. Calling z_i^k the output of the "filter" gate labeled k having as input v_i, the binary logic used for generating z_i^k is

$$z_i^k = \text{lit}(v_i^l, k-1) \wedge \text{lit}(v_i^{l-1}, k-1) \wedge \cdots \wedge \text{lit}(v_i^0, k-1)\,,$$

where $\text{lit}(v_i^j, m) = v_i^j$ if the jth bit of the binary code representing m is 1 and $\text{lit}(v_i^j, m) = \overline{v_i^j}$ if it is 0. For building the coded ROBDD our implementation of the method uses the well-known BDD Library developed at Carnegie-Mellon University [3].

It is well-known that the size of the ROBDD of a Boolean function of binary variables depends on the ordering of the binary variables. Similarly, the size of the ROMDD of a multiple-valued function of multiple-valued variables depends on the ordering of the multiple-valued variables. The variables are most often sorted using heuristics and an abundant literature is available about heuristics for ordering the variables of Boolean functions of binary variables using gate-level representations of the functions [5, 7, 9, 10, 11, 21, 26, 27]. Those heuristics can be classified into static and dynamic depending on whether the ordering is computed before the ROBDD is built or the ordering may be changed during the ROBDD construction. Three heuristics which are relatively simple to implement and which have good performance are the *topology* heuristic described in [27], the *weight* heuristic described in [26] and the *H4* heuristic described in [5].

3 Method for Computation of the Operational Reliability

3.1 Perfect coverages

We will find convenient to compute the operational unreliability $\text{ur}(t)$ instead of the operational reliability. The operational unreliability $\text{ur}(t)$ is formally defined as the probability that the system has been failed at some time $\tau \in [0, t]$. Since $F(x_1, \ldots, x_n)$ is assumed to be increasing, the system will be failed at time t if and only if the system has been failed at some time $\tau \in [0, t]$ and, then, $\text{ur}(t)$ is simply the probability that the system is failed at time t. Let

$$\text{ur}_k(t) = P[\text{system is failed at time } t \mid \text{there are } k \text{ lethal defects}]\,.$$

We have

$$\text{ur}(t) = \sum_{k=0}^{\infty} Q'_k \text{ur}_k(t)\,.$$

Analyzing up to M defects we can pessimistically estimate $\text{ur}(t)$ by

$$\text{ur}^M(t) = \sum_{k=0}^{M} Q'_k \text{ur}_k(t) + \sum_{k=M+1}^{\infty} Q'_k$$

with error bounded from above by $\sum_{k=M+1}^{\infty} Q'_k = 1 - \sum_{k=0}^{M} Q'_k$. Then, given a suitable error control parameter ε, we can select

$$M = \min \left\{ m \geq 0 \;:\; 1 - \sum_{k=0}^{m} Q'_k \leq \varepsilon \right\} , \tag{8}$$

guaranteeing and absolute error in the unreliability estimation $\leq \varepsilon$.

In this section we will describe the method for the particular case $C_i = 1, 1 \leq i \leq C$, i.e. when all components have perfect fault coverage. The more general case $0 \leq C_i \leq 1$, $1 \leq i \leq C$ will be dealt with in Section 3.2.

The unreliability estimate $\mathrm{ur}^M(t)$ can be formalized as the probability that a boolean function of certain independent integer-valued variables is equal to 1. The function is

$$\begin{aligned} G(w, v_1, \ldots, v_M, u_1, \ldots, u_C) &= I_{M+1}(w) \\ \vee\; F &\left(u_1 \vee \sum_{l=1}^{M} I_{\geq l}(w) \wedge I_1(v_l), \ldots, u_C \vee \sum_{l=1}^{M} I_{\geq l}(w) \wedge I_C(v_l) \right) . \end{aligned} \tag{9}$$

Then, with the random variables W and V_i, $1 \leq i \leq M$ defined in Section 2 and the random variables U_i, $1 \leq i \leq C$ taking values in $\{0, 1\}$ with probability distributions $P[U_i = 0] = R_i(t)$ and $P[U_i = 1] = 1 - R_i(t)$, we have the following result.

Theorem 2. $\mathrm{ur}^M(t) = P[G(W, V_1, V_2, \ldots, V_M, U_1, U_2, \ldots, U_C) = 1]$.

Intuitively, the reason why Theorem 2 holds is that $I_{M+1}(W)$ "tells" whether the number of lethal defects is $> M$, $I_{\geq l}(W)$ "tells" whether there is a lth lethal defect, $I_i(V_l)$ "tells" whether the lth lethal defect affects component i, then, $\bigvee_{l=1}^{M} I_{\geq l}(W) \wedge I_i(V_l)$ "tells" whether component i is affected by some of the first M lethal defects, and, then, since U_i "tells" whether, assuming component i is initially unfailed, component i has failed by time t, $U_i \vee \bigvee_{l=1}^{M} I_{\geq l}(W) \wedge I_i(V_l)$ "tells" whether at time t component i is failed. A formal proof can be developed using the following proposition.

Proposition 1. $\mathrm{ur}_k(t) = P[F(U_1 \vee \bigvee_{l=1}^{k} I_1(V_l), U_2 \vee \bigvee_{l=1}^{k} I_2(V_l), \ldots, U_C \vee \bigvee_{l=1}^{k} I_C(V_l)) = 1]$.

Proof. Consider a system S' differing from the true system S only in that each component i of S has been replaced by two components i', i''. Component i' has the initial state of component i and does not fail with time. Component i'' is initially unfailed and has reliability $R_i(t)$. Components i'' behave independently among them and independently of components i'. The fault-tree function of system S' is $F'(x'_1, \ldots, x'_C, x''_1, \ldots, x''_C) = F(x'_1 \vee x''_1, \ldots, x'_C \vee x''_C)$, where x'_i is equal to 1 if and only if component i' is failed and x''_i is equal to 1 if and only if component i'' is failed. By construction, the reliabilities of S' and S are identical. Then, $\mathrm{ur}_k(t)$ is the unreliability of S' conditioned to the set of components $\{1', \ldots, C'\}$ being affected by exactly k lethal defects. But, given there are k lethal defects, component i' is affected by some of them if and only if $\bigvee_{l=1}^{k} I_i(V_l) = 1$ and the result follows. $\square$

Proof of Theorem 2 Using the theorem of total probability and the independence of the random variables $W, V_1, \ldots, V_M, U_1, \ldots, U_C$:

$$\begin{aligned} & P[G(W, V_1, \ldots, V_M, U_1, \ldots, U_C) = 1] \\ &= \sum_{k=0}^{M+1} P[W = k] P[G(W, V_1, \ldots, V_M, U_1, \ldots, U_C) = 1 \mid W = k] \\ &= \sum_{k=0}^{M+1} P[W = k] P[G(k, V_1, \ldots, V_M, U_1, \ldots, U_C) = 1] \,. \end{aligned}$$

But, from the definition of G (9), for $0 \leq k \leq M$:

$$G(k, v_1, \ldots, v_M, u_1, \ldots, u_C) = F\left(u_1 \vee \bigvee_{l=1}^{k} I_1(v_l), \ldots, u_C \vee \bigvee_{l=1}^{k} I_C(v_l)\right)$$

and

$$G(M+1, v_1, \ldots, v_M, u_1, \ldots, u_C) = 1 \,.$$

Using, then, Proposition 1:

$$\begin{aligned} & P[G(W, V_1, \ldots, V_M, U_1, \ldots, U_C) = 1] \\ &= \sum_{k=0}^{M} P[W = k] \mathrm{ur}_k(t) + P[W = M+1] = \sum_{k=0}^{M} Q'_k \mathrm{ur}_k(t) + \sum_{k=M+1}^{\infty} Q'_k = \mathrm{ur}^M(t) \,. \quad \square \end{aligned}$$

The probability $P[G(W, V_1, \ldots, V_M, U_1, \ldots, U_C) = 1]$ can be computed from a ROMDD of the function $G(w, v_1, \ldots, v_M, u_1, \ldots, u_C)$ in a similar way as $P[G(W, V_1, \ldots, V_M) = 1]$ is computed from a ROMDD of the function $G(w, v_1, \ldots, v_M)$ in the combinatorial method for evaluation of yield. Such a ROMDD can be obtained from the gate-level description of the function $G(w, v_1, \ldots, v_M, u_1, \ldots, u_C)$ shown in Fig. 4 in terms of the available gate-level description of the function $F(x_1, \ldots, x_C)$. As in the combinatorial method for evaluation of yield, we obtain first a coded ROBDD, using the same codes for the multiple-valued variables $w, v_1, \ldots, v_M$ and the same binary logic for implementing the filter gates as in that method and, then, obtain the ROMDD. Our implementation of the method for the computation of operational reliability also uses the BDD Library [3] for building the coded ROBDD.

3.2 Extension to take into account imperfect coverages

The method for evaluation of reliability can be easily extended to take into account imperfect component coverages to operational faults using the SEA approach [2]. Using that approach, the unreliability $\mathrm{ur}(t)$ for the system S' introduced in the proof of Proposition 1 with coverage faults (equal to the reliability of the true system S) can be formalized in terms of the unreliability, $\mathrm{ur}^*(t)$ of a fictitious system, S'^*, differing from S' only in that the reliabilities of the components subject to coverage faults (components i'', $1 \leq i \leq C$) have values

$$R_i^*(t) = \frac{R_i(t)}{R_i(t) + C_i(1 - R_i(t))} \tag{10}$$

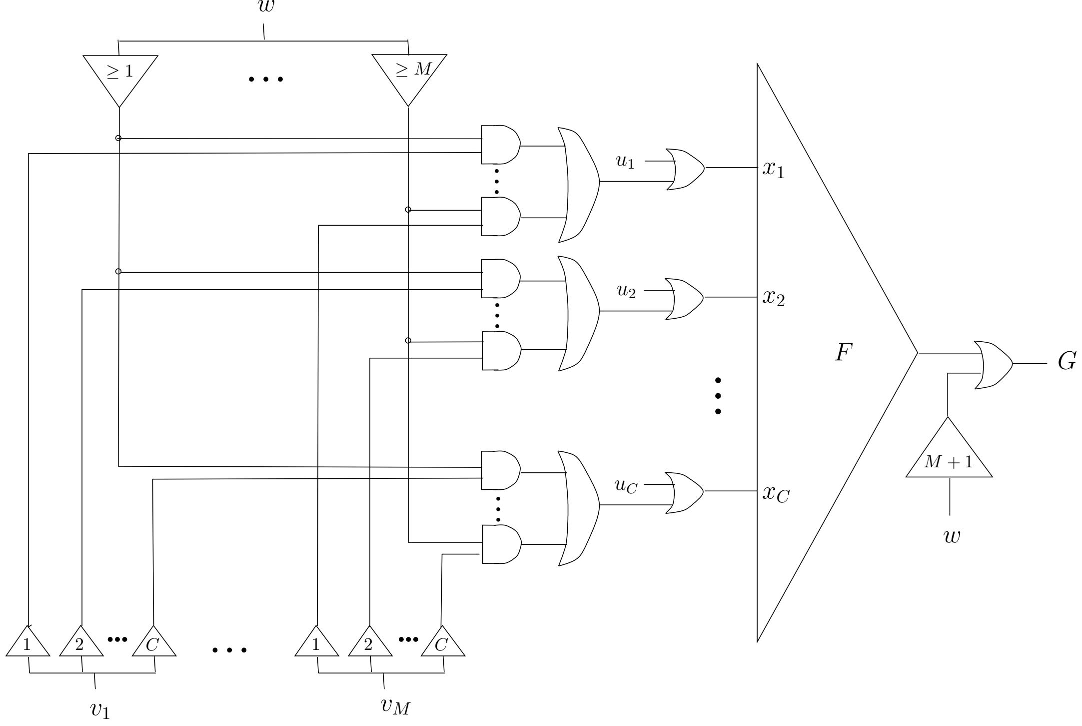

Figure 4: Gate-level description of the function $G(w, v_1, \ldots, v_M, u_1, \ldots, u_C)$.

as

$$\mathrm{ur}(t) = 1 - P_U(1 - \mathrm{ur}^*(t)),$$

with

$$P_U = \prod_{i=1}^{C}(1 - (1 - C_i)(1 - R_i(t))). \quad (11)$$

If the combinatorial method is applied to the system S^* differing from S only in that the operational reliabilities of the components are $R_i^*(t)$ and an approximation $\mathrm{ur}^{*M}(t)$ for the unreliability $\mathrm{ur}^*(t)$ of S^* is obtained with M selected using (8), it will hold $\mathrm{ur}^{*M}(t) - \mathrm{ur}^*(t) \leq \varepsilon$ and, if $ur(t)$ is estimated by

$$\mathrm{ur}^M(t) = 1 - P_U(1 - \mathrm{ur}^{*M}(t)), \quad (12)$$

it will hold

$$\mathrm{ur}^M(t) - \mathrm{ur}(t) = P_U(1 - \mathrm{ur}^*(t)) - P_U(1 - \mathrm{ur}^{*M}(t)) = P_U\mathrm{ur}^{*M}(t) - P_U\mathrm{ur}^*(t) \leq P_U\varepsilon \leq \varepsilon.$$

Then, M can be selected using (8), the pessimistic unreliability approximation $\mathrm{ur}^{*M}(t)$ can be computed using the combinatorial method with the operational reliabilities $R_i^*(t)$ given by (10) instead of $R_i(t)$, and using (12) with P_U given by (11) a pessimistic approximation $\mathrm{ur}^M(t)$ for $\mathrm{ur}(t)$ will be obtained with error bounded from above by ε.

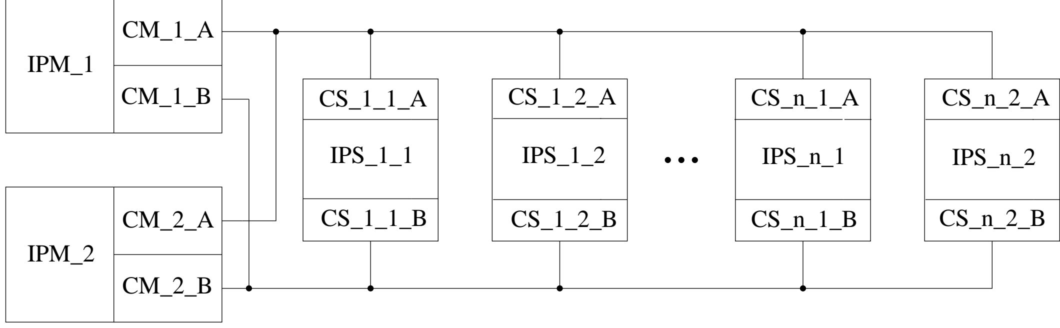

Figure 5: Architecture of system-on-chip MSn.

4 Benchmarks description

In this section we describe the benchmarks which will be used to evaluate the performance of the combinatorial methods. The benchmarks are two scalable examples which instantiate systems-on-chip of increasing numbers of components. The first scalable example, called MSn, is the system-on-chip with the architecture illustrated in Fig. 5. The system includes a cluster of two "master" Intellectual Property cores IPM and n clusters including two "slave" Intellectual Property cores IPS. Those Intellectual Property cores are interconnected using communication modules CM and CS and two buses. Buses are assumed to be not affected by manufacturing defects. This implies that the system can be conceptualized as made up of only IPMs, IPSs and communication modules. The system is operational if at least an unfailed IPM can communicate with at least an unfailed IPS of each cluster using unfailed communication modules. The communication between the IPM and each IPS has to be direct, i.e. it can only involve a bus and two communication modules. The numbetr of manufacturing defects is assumed to have a negative binomial distribution with clustering parameter $\alpha = 3$; for the expected number of defects two values will be considered: $\lambda = 2$ and $\lambda = 4$. Furthermore, the probabilities P_i will be taken so that $P_L = \sum_{i=1}^{C} P_i = 0.5$ (and, then, λ' will take the values 1 and 2) and, calling, P_{IPM} the P_i probability of an IPM, P_{IPS} the P_i probability of an IPS, and P_{C} the P_i probability of a communication module, the following relationships are satisfied: $P_{\mathrm{IPS}}/P_{\mathrm{IPM}} = 0.5$, $P_{\mathrm{C}}/P_{\mathrm{IPM}} = 0.1$. The two values for λ are intended to reflect scenarios in which the fault-tolerant system-on-chip is affected by defects with moderate and high severity. The required value of M and, thus, the computational cost of the method increases with λ. The performance of the method for the evaluation of the operational reliability is independent on the component reliability functions $R_i(t)$ and the values of the coverages C_i. Thus, we will not specify particular values for those functions and parameters.

The second scalable example is the system-on-chip ESEN $n \times m$ with the architecture described in Fig. 6 for the case $n = 8$, $m = 2$. The system includes $(n \times m)/2$ Intellectual Property cores IPA and $(n \times m)/2$ Intellectual Property cores IPB interconnected by a ESEN multiexchange interconnection network with n inputs [29], through $m \times 1$ concentrators (C) in case $m > 1$, in which each switching element (SE) of the first and last stage have a redundant copy. The system is operational if $(n \times m)/2 - 1$ unfailed IPAs and $(n \times m)/2 - 1$ unfailed IPBs can communicate through the

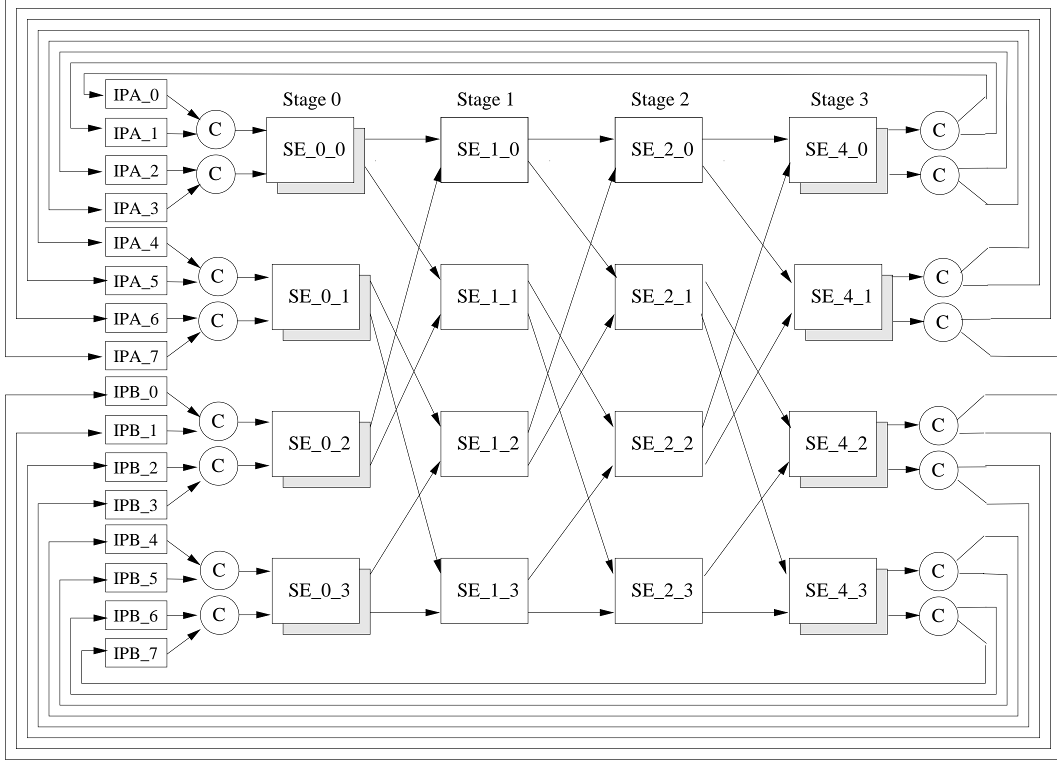

Figure 6: Architecture of system-on-chip ESEN8x2.

interconnection network. It is assumed that links are not affected by manufacturing defects. Thus, the system can be conceptualized as made up of only IPAs, IPBs, SEs and, in case $m > 1$, Cs. As in the first scalable example, the number of manufacturing defects is assumed to have a negative binomial distribution with clustering parameter $\alpha = 3$ and for the expected number of defects two values will be considered: $\lambda = 2$ and $\lambda = 4$. Furthermore, the probabilities P_i will be taken so that $P_L = \sum_{i=1}^{C} P_i = 0.5$ (and, then, λ' will take the values 1 and 2) and, calling, P_{IPA} the P_i probability of an IPA, P_{IPB} the P_i probability of an IPB, P_{SE} the P_i probability of a SE, and P_{C} the P_i probability of a C, the following relationships are satisfied: $P_{\text{IPB}}/P_{\text{IPA}} = 0.5$, $P_{\text{SE}}/P_{\text{IPA}} = 0.05$ and $P_{\text{C}}/P_{\text{IPA}} = 0.02$.

Table 1 gives the number of components C of the benchmarks which will be used to evaluate the performance of the combinatorial methods and the number of gates of the gate-level descriptions of the corresponding fault-tree functions used in the experiments. That information will allow to asses the complexity (number of components) of the systems which the methods can handle.

5 Results

All experiments reported in this section were performed in a workstation with a Sun-Blade-1000 processor and 4 GB of memory. The size of the ROMDD is affected by the ordering of the multiple-values input variables ($w, v_1, \ldots, v_M$ in the method for yield evaluation and

Table 1: Number of components (C) of the benchmarks and number of gates of the gate-level descriptions of the corresponding fault-tree functions.

benchmark	C	gates
MS2	18	27
MS4	30	51
MS6	42	75
MS8	54	99
MS10	66	123
ESEN4x1	14	13
ESEN4x2	26	26
ESEN4x4	34	74
ESEN8x1	32	73
ESEN8x2	56	122

$w, v_1, \ldots, v_M, u_1, \ldots, u_C$ in the method for the evaluation of operational reliability). For the method for yield evaluation we experimented with several orderings: (1) $w, v_1, \ldots, v_M$, (2) $w, v_M, \ldots, v_1$, (3) $v_1, \ldots, v_M, w$, (4) $v_M, \ldots, v_1, w$, (5) ordering which results when the heuristic *topology* is applied to the gate-level description of $G(w, v_1, \ldots, v_M)$ in binary logic and the multiple-valued variables are sorted in increasing oeder of the average indices over the groups of binary variables encoding the multiple-valued variables, (6) idem but using the *weight* heuristic, and (7) idem but using the *H4* heuristic. We found that the ordering based on the *weight* heuristic gave the best results. The size of the coded ROBDD is also affected by the ordering of the group of binary variables encoding each multiple-valued variable. The ordering most to least significant bit gave consistently best results. Similar experiments for the method for the evaluation of operational reliability showed the same orderings to be the best ones. We will assess the computational cost of both methods using those best orderings.

We analyze first the performance of the method for yield computation. Table 2 gives, for the benchmarks in which the method succeeded, the CPU times, peak number of ROBDD nodes (maximum sum of the nodes of the ROBDDs which had to be held simultaneously in memory when processing the generalized fault-tree), size of the coded ROBDD, size of the ROMDD, and computed yield loss when the method was run with an error requirement $\varepsilon = 2 \times 10^{-3}$. Several comments are in order. First, CPU times are reasonable in all cases, since in the worst case (ESEN8x2, $\lambda = 2$) the CPU time is about 18 minutes. Second, the peak number of ROBDD nodes can be or not much larger than the size of the final coded ROBDD. In practice, the application of the method is limited by that peak, since it is that peak which determines the peak memory consumption of the method. Third, the size (number of nodes) of the coded ROBDD is about 10 times the size of the ROMDD. With that factor, even an efficient implementation of ROMDDs is likely to consume more memory than the coded ROBDD, which has a much simpler structure. Thus, the approach of working with coded ROBDDs and translate the final coded ROBDD to the ROMDD required to perform the yield

Table 2: Performance of the method for the evaluation of yield for $\varepsilon = 2 \times 10^{-3}$.

benchmark	CPU time (s)	ROBDD peak	ROBDD	ROMDD	L^M
MS2, $\lambda = 2$	0.98	30,987	24,237	2,034	0.056
MS4, $\lambda = 2$	6.23	427,130	243,154	22,760	0.035
MS6, $\lambda = 2$	66.4	2,564,600	1,120,255	103,228	0.025
MS8, $\lambda = 2$	262	7,518,549	3,154,056	309,136	0.020
MS10, $\lambda = 2$	862	20,344,432	7,954,261	731,748	0.016
MS2, $\lambda = 4$	3.59	124,067	116,960	7534	0.170
MS4, $\lambda = 4$	828	14,175,238	11,885,214	635,530	0.115
ESEN4x1, $\lambda = 2$	0.86	37,231	19,338	3046	0.090
ESEN4x2, $\lambda = 2$	2.72	200,272	54,705	6995	0.152
ESEN4x4, $\lambda = 2$	14.6	368,815	184,332	19,547	0.171
ESEN8x1, $\lambda = 2$	173	6,544,206	904,777	134,512	0.119
ESEN8x2, $\lambda = 2$	1061	29,926,091	2,244,340	303,657	0.165
ESEN4x1, $\lambda = 4$	3.47	143,633	105,511	11,666	0.244
ESEN4x2, $\lambda = 4$	18.3	757,529	378,686	30,783	0.368
ESEN4x4, $\lambda = 4$	109	3,027,309	1,513,441	96,231	0.395

computations seems to be a good approach. This is consistent with the conclusion reached by researchers in the ROMDD community that coded ROBDDs is probably the most efficient way of handling ROMDDs [24, 25]. Putting all results together, it seems that the method can efficiently compute the yield of systems with up to about 60 components when the average number of lethal defects is moderate ($\lambda = 2$) and up to about 30 components when the average number of lethal defects is large ($\lambda = 4$). The number of components which the method can handle depends, of course, on the value of the truncation parameter M. That parameter had value 6 for the examples with $\lambda = 2$ and value 10 for the examples with $\lambda = 4$.

We analyze next the performance of the method for the evaluation of operational reliability. Table 3 gives, for the benchmarks in which the method succeeded, the CPU times, peak number of ROBDD nodes, size of the coded ROBDD, and size of the ROMDD for an error requirement $\varepsilon = 10^{-4}$. As for the method for yield evaluation, CPU times are reasonable in all cases. The method seems to be able to handle efficiently systems with up to about 30 components.

6 Conclusions

Systems-on-chip have reached a complexity degree that make them very susceptible to both manufacturing defects and operational faults so that reasonable yields and operational reliabilities can only be achieved with the use of fault-tolerance techniques. That application of fault-tolerance calls for

Table 3: Performance of the method for the evaluation of operational reliability for $\varepsilon = 1 \times 10^{-4}$.

benchmark	CPU time (s)	ROBDD peak	ROBDD	ROMDD
MS2, $\lambda = 2$	3.83	139,713	136,108	11,895
MS4, $\lambda = 2$	3031	19,467,751	18,269,514	1,420,505
MS2, $\lambda = 4$	7.8	257,644	254,006	18,785
ESEN4x1, $\lambda = 2$	3.14	141,604	107,612	12,646
ESEN4x2, $\lambda = 2$	19.7	1,037,767	434,943	87,409
ESEN4x4, $\lambda = 2$	113	3,846,387	1,735,787	325,387
ESEN4x1, $\lambda = 4$	6.51	277,375	221,660	23,586
ESEN4x2, $\lambda = 4$	44.1	1,690,942	876,401	117,789
ESEN4x4, $\lambda = 4$	249	6,920,780	3,584,315	423,807

efficient methodologies for evaluation of yield and operational reliability of fault-tolerant systems-on-chip. Such evaluation is difficult because realistic models for manufacturing defects production have clustering and, thus, introduce dependencies among the failed states of the components making up the system. In this paper, we have developed combinatorial methods for the evaluation of yield and operational reliability of fault-tolerant systems-on-chip supporting realistic models with clustering for manufacturing defects. The methods build a ROMDD of a Boolean function with multiple-valued variables which allows to compute with a predefined accuracy the yield or the operational reliability. The ROMDD is built automatically from a gate-level description of the fault-tree specifying the structure function of the system. The computational complexity of the methods increases with the expected number of lethal defects in the fault-tolerant system. We have shown, however, that the methods are able to deal, using currently affordable computational resources, with systems having tens of components.

Acknowledgements

This work was supported by the “Comisión Interministerial de Ciencia y Tecnología” (CICYT) of the Ministry of Science and Technology of Spain under the research grant TAP1999-0443-C05-05.

References

[1] Aho AV, Hopcroft JE, Ullman JD. Data structures and algorithms. Addison-Wesley: 1983.

[2] Amari SV, Dugan JB, Misra RB. A separable method for incorporating imperfect fault-coverage into combinatorial models. IEEE Trans. on Reliab 1999;48(3):267–274.

[3] The BDD Library. Available at http : //www − 2.cs.cmu.edu/ modelcheck/bdd.html.

[4] Benini L, De Micheli G. Networks on chip: A new SoC paradigm. IEEE Comp 2002;35(1):70–78.

[5] Bouissou M, Bruyère F, Rauzy A. BDD based fault-tree Processing: a comparison of variable ordering heuristics. In: Guedes Soares C, editor. Proc. of Euro Safety Reliab Ass Conf (ESREC'97), 3, 1997, p. 2045–52.

[6] Bryant RE. Graph-based algorithms for Boolean function manipulation. IEEE Trans Comput 1986;C-35(8):677–91.

[7] Butler KM, Ross DE, Kapurand R, Ray Mercer M. Heuristics to compute variable orderings for efficient manipulation of ordered binary decision diagrams. In: Proc 28th ACM/IEEE Design Automat Conf, 1991. p. 417–20.

[8] Cunningham J. The use and evaluation of yield models in integrated circuit manufacturing. IEEE Trans Semicond Manufact 1990;3(2):60–71.

[9] Fujita M, Fujisawa H, awato N. Evaluation and improvements of Boolean comparison method based on binary decision diagrams. In: Proc IEEE Int Conf Comput Aided Design (ICCAD'88), 1988. p. 2–5.

[10] Fujita M, Matsunaga Y, Kakuda T. On variable ordering of binary decision diagrams for the application of multi-level logic synthesis. In: Proc IEEE Euro Conf Design Automat (EDAC'91), 1991. pp. 50–4.

[11] Fujita M, Fujisawa H, Matsunaga Y. Variable ordering algorithms for ordered binary decision diagrams and their evaluation. IEEE Trans Comput-Aided Design Integ Circuits Sys 1993;12(1):6–12.

[12] Koren I, Pradhan DK. Yield and performance enhancement through redundancy in VLSI and WSI multiprocessor systems. Proc IEEE 1986;74(5):699-711.

[13] Koren I, Pradhan DK. Modeling the effect of redundancy on yield and performance of VLSI systems. IEEE Trans Comput 1987;C-36(3):344-55.

[14] Koren I, Koren Z, Pradhan DK. Designing interconnection buses in VLSI and WSI for maximum yield and minimum delay. IEEE J Solid-State Circuits 1988;23(3):859–65.

[15] Koren I, Stapper CH. Yield models for defect-tolerant VLSI circuits: a review. In: Koren I, editor, Defect and Fault Tolerance in VLSI Systems. vol. I. Plenum; 1989. p. 1–21.

[16] Koren I, Koren Z, Stapper CA. A unified negative-binomial distribution for yield analysis of defect-tolerant circuits. IEEE Trans Comput 1993;42(6):724–34.

[17] Koren I, Koren Z, Stapper C. A statistical study of defect maps of large area VLSI IC's. IEEE Trans Very Large Scale Integ Sys 1994;2(2):249–56.

[18] Koren I, Koren Z. Analysis of a hybrid defect-tolerance scheme for high-density memory ICs. In: Proc. IEEE Int Symp Defect and Fault Tolerance in VLSI Syst, 1997. p. 38–42.

[19] Koren I, Koren Z. Defect tolerance in VLSI circuits: techniques and yield Analysis. Proc IEEE 1999;86(9):1819–38.

[20] Mak TM, Bhattacharya D, Prunty C, Roeder B, Ramadan N, Ferguson J, Jianlin Y. Cache RAM inductive fault analysis with fab defect modeling. In: Proc IEEE Int Test Conf, 1998. p. 862–71.

[21] Malik S, Wang AR, Brayton RK, Sangiovanni-Vincentelli A. Logic verification using binary decision diagrams in a logic synthesis environment. In: Proc IEEE Int Conf Comput-Aided Design (ICCAD'88), 1988, p. 6–9.

[22] Metra C, Di Francescantonio S, Mak TM, Ricco B. Evaluation of clock distribution networks' most likely faults and produced defects. In: Proc IEEE Int Symp Defect and Fault Tolerance in VLSI Syst, 2001, p. 357–65.

[23] Meyer FJ, Pradhan DK. Modeling defect spatial distribution. IEEE Trans Comput 1989;38(4):538–46.

[24] Miller DM, Drechsler R. Implementing a multiple-valued decision diagram package. In: Proc. 28th IEEE Int Symp Multiple-Valued Logic, 1998, p. 52–7.

[25] Miller DM. Private communication collecting the conclusions of the 30th IEEE Int Symp Multiple-Valued Logic, 2000.

[26] Minato S, Ishiura N, Yajima S. Shared binary decision diagram with attributed edges for efficient Boolean function manipulation. In: Proc 27th ACM/IEEE Design Automat Conf, 1990, p. 52–7.

[27] Nikolskaïa M, Rauzy A, Sherman DJ. Almana: A BDD minimization tool integrating heuristic and rewriting methods. In: Proc Int Conf Formal Meth in Comput-Aided Design (FMCAD), 1998, p. 100–14.

[28] Nikolos D, Vergos HT. On the yield of VLSI processors with on-chip CPU cache. IEEE Trans Comput1999;48(10):1138–44.

[29] Rai S, Oh YC. Tighter bounds on full access probability in fault-tolerant multistage interconnection networks. IEEE Trans Parallel Distributed Syst 1999;10(3):328–35.

[30] Srinivasan A, Kam T, Malik S, Brayton RK. Algorithms for discrete function manipulation. In: Proc IEEE Int Conf Comput-Aided Design (ICCAD-90), 1990, p. 92–5.

[31] Venkataraman A, Koren I. Determination of yield bounds prior to routing. In: Proc IEEE Int Symp on Defect and Fault-Tolerance in VLSI Syst, 1999, p. 4–13.

[32] Wagner IA, Koren I. An interactive VLSI CAD tool for yield estimation. IEEE Trans Semicond Manufact 1995;8(2):130–8.

[33] Yu J, Ferguson FJ. Maximum likelihood estimation for failure analysis. IEEE Trans Semicond Manufact 1998;11(4):681–91.