IEICE TRANS. FUNDAMENTALS, VOL.E97–A, NO.7 JULY 2014

PAPER Special Section on Design Methodologies for System on a Chip

Comparative Evaluation of Lifetime Enhancement with Fault Avoidance on Dynamically Reconfigurable Devices

Hiroaki KONOURA[†,††a)], **Takashi IMAGAWA**[†††,††b)], ***Student Members***, **Yukio MITSUYAMA**[††††,††c)], **Masanori HASHIMOTO**[†,††d)], ***and*** **Takao ONOYE**[†,††e)], ***Members***

SUMMARY Fault tolerant methods using dynamically reconfigurable devices have been studied to overcome wear-out failures. However, quantitative comparisons have not been sufficiently assessed on device lifetime enhancement with these methods, whereas they have mainly been evaluated individually from various viewpoints such as additional hardware overheads, performance, and downtime for fault recovery. This paper presents quantitative lifetime evaluations performed by simulating the fault-avoidance procedures of five representative methods under the same conditions in wear-out scenarios, applications, and device architecture. The simulation results indicated that improvements of up to 70% mean-time-to-failure (MTTF) in comparison with ideal fault avoidance could be achieved by using methods of fault avoidance with 'row direction shift' and 'dynamic partial reconfiguration'. 'Column shift', on the other hand, attained a high degree of stability with moderate improvements in MTTF. The experimental results also revealed that spare basic elements (BEs) should be prevented from aging so that improvements in MTTF would not be adversely affected. Moreover, we found that the selection of initial mappings guided by wire utilization could increase the lifetimes of partial reconfiguration based fault avoidance.

key words: *fault avoidance, lifetime enhancement, mean-time-to-failure (MTTF), partial reconfiguration*

1. Introduction

Aggressive CMOS technology scaling is threatening the reliability of devices and early-life, normal-life, and wear-out failures are all increasing. Burn-in and testing have been extensively studied to screen faulty chips [1] in early-life failures. Normal-life failures are mostly caused by temporal effects, such as soft errors and power supply noise, and error mitigation and recovery have extensively been considered [2]. However, wear-out failures lead to permanent errors in fields and generally cannot be resolved without special mechanisms. Wear-out failures originate from aging effects such as bias temperature instability (BTI) and time dependent dielectric breakdown (TDDB), and these result in degrading the lifetimes of devices and may require frequent device replacement. Fault-tolerance techniques, which sustain chip functionality even when there are some faulty modules, are required to increase device lifetimes by coping with increasing numbers of wear-out failures.

Research on fault tolerance has a long history, and fault tolerant techniques have been widely researched at different levels. For instance, static random-access memories (SRAMs) are often equipped with redundant rows to act as replacements [3]. Gradually, graceful performance degradation instead of sudden failures in devices has also been explored [4] at the architecture level. However, one of the most reliable approaches to fault tolerance is to exploit redundancy and replace faulty modules with spares. These are especially extremely compatible with reconfigurable devices because unused basic elements (BEs) are available and can be used for the replacements.

Here, let us explain the degree of lifetime enhancement. Figure 1 plots the mean-time-to-failure (MTTF) as a function of the number of avoidable faulty BEs on a device. Note that the experimental setup for this evaluation was the same as that in Sect. 5.1 and will be explained later. MTTF with one-time avoidance of a single faulty BE is doubled compared to MTTF with no avoidance. However, when 10 BEs can be avoided, MTTF increases five fold and when 100 faulty BEs can be avoided, it increases twenty fold. This has two implications for designers in that 1) moderate lifetime enhancements, such as up to two times, can be attained by one-time fault avoidance, 2) aggressive lifetime enhancement, such as more than 10 times, requires successive fault avoidance in faulty BEs. Designers who adopt fault-avoidance methods need to explore and select the best one that satisfies the required level of lifetime enhancement depending on target applications and the environment.

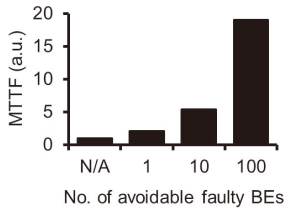


Fig. 1 Relationship between MTTF and number of avoidable faulty BEs (FFT).

Manuscript received September 12, 2013.
Manuscript revised January 9, 2014.

[†]The authors are with the Department of Information Systems Engineering, Osaka University, Suita-shi, 565-0871 Japan.
[††]The authors are with JST, CREST, Tokyo, 102-0075 Japan.
[†††]The author is with the Department of Communications and Computer Engineering, Kyoto University, Kyoto-shi, 606-8501 Japan.
[††††]The author is with the School of Systems Engineering, Kochi University of Technology, Kami-shi, 782-8502 Japan.
a) E-mail: konoura.hiroaki@ist.osaka-u.ac.jp
b) E-mail: imagawa@easter.kuee.kyoto-u.ac.jp
c) E-mail: mitsuyama.yukio@kochi-tech.ac.jp
d) E-mail: hasimoto@ist.osaka-u.ac.jp
e) E-mail: onoye@ist.osaka-u.ac.jp
DOI: 10.1587/transfun.E97.A.1468

Several methods to replace faulty BEs on reconfigurable devices have been proposed to achieve such lifetime enhancements. Doumar et al. [5] proposed a hardware-oriented method of fault recovery on field-programmable gate arrays (FPGAs). Bypassing faulty BEs with this method is guided by rules, and global wires and switches inherently equipped in FPGAs are utilized for bypassing faults. A substitute BE inheriting the function of a faulty BE is uniquely determined corresponding to its spatial location, and thus, the possible number of avoidable faulty BEs is simply estimated by using available bypassing mechanisms and remaining wire resources. Such methods can be reinforced to accommodate more faults by adding extra wires and switches; however, these hardware overheads tend to be significant. To make matters worse, they might cause increases in potential breakdowns. In fact, Koal and Vierhaus [6], who evaluated the reliability of a reconfigurable processor with fault recovery, indicated that a large hardware overhead for fault recovery adversely affected the enhanced lifetime attained by recovery. Dynamical partial reconfiguration on coarse-grained reconfigurable architectures has been studied [7] as an alternative method of solving this problem. When a faulty BE is identified with this method by periodically reconfiguring and running test patterns, a new mapping, which isolates the faulty BE by assigning its operation to unused BEs exploring available freedom both in space and time, is dynamically calculated by an embedded CPU. This method exploits the reconfigurability of devices and does not require any additional switches or wires that could be a new source of faults. However, multiple faulty BEs have not been evaluated and the potentially enhanced lifetime with this method is yet unknown.

Parris et al. [8] surveyed and summarized such fault-tolerance techniques in terms of both overheads, such as physical resources and reduced throughput, and sustainability, such as recovery granularity and fault capacity. Here, fault capacity means that the number of fault-free resource units necessary to sustain system functionality against a single additional fault. Similarly, Doumar and Ito [9] classified fault tolerant methods and calculated the yields of configurable logic blocks with some defects. These surveys are very helpful to understand the features of representative fault-tolerance methods. However, achievable life-time enhancement with hardware-oriented methods and reconfiguration-oriented methods under sequential or multiple faults has not been quantitatively evaluated and compared. Further beneficial guidelines for designers are required to enable them to select appropriate fault-avoidance methods that will provide sufficient lifetime enhancements required for target applications and environments.

Here, the terms fault avoidance and fault tolerance are often used in similar contexts, but Laprie [10] differentiates them as "preventing fault occurrence" for the former and "providing service in spite of faults having occurred or occurring" for the latter. Although we focused on preventing fault occurrences with proactive fault detection and use the term fault avoidance rather than fault tolerance in this paper, the observations in this work are still expected to be helpful for fault tolerance.

In this study, five fault-avoidance methods spanning hardware-oriented to reconfiguration-oriented approaches were selected as being representative and they were applied to coarse-grained reconfigurable devices. Preliminary results from evaluating these five methods have been reported [11]. MTTF increases due to these five fault-avoidance techniques were compared and discussed through evaluations with small applications of a six-tap finite impulse response (FIR) filter and two-point discrete Fourier transform (DFT) mapped on a 6×6 array.

On the other hand, we mapped more than ten times larger applications to acquire more comprehensive observations in this study. The placement and routing algorithm needs to be improved to obtain large-scale initial mappings so that target applications with different scales can be evaluated and new observations can be obtained to deal with larger applications. For instance, we found that efficiency in spare usage derived from previous evaluations with a single application was underestimated, and newly obtained results indicated that the maximum difference in efficiency in spare usage between the five fault-avoidance methods reached a hundred fold. Moreover, other results such as the shapes of time-to-failure (TTF) distributions and the 10th percentile of the TTF are presented to explain the features of each fault-avoidance method in this work. Another contribution of this work is to clarify the importance of initial mappings for fault-avoidance methods using partial reconfigurations and to indicate that lifetime enhancements are dependent on the utilization of wire tracks.

This paper consists of six sections. Section 2 describes the environment for lifetime evaluations assumed in this work. The five fault-avoidance methods are classified and introduced in Sect. 3, followed by a procedure for partial placement and routing in Sect. 4, which is required to evaluate fault-avoidance methods that are based on reconfigurations. Section 5 explains the experimental setup and presents simulation results for device lifetimes, and the features of these five methods are summarized in the last subsection. Finally, Sect. 6 concludes the paper.

2. Environment for Evaluation of Lifetimes

This section first reviews techniques of fault detection to achieve predictive fault avoidance. We then present the basic reconfigurable architecture and environment used for evaluating the lifetimes in this study.

2.1 Techniques of Fault Detection

Major permanent faults emerging on the user side can roughly be classified into two groups; logic faults and delay faults. Logic faults such as stuck-at faults and bridge faults are caused by physical damage to wires and CMOS gates, which occur due to electro migration (EM) [12], electrostatic discharge (ESD) [13], and TDDB, for example. De-

lay faults such as path delay faults and transition faults are caused by aging effects in CMOS gates such as BTI, hot carrier injection (HCI), and early-stage (soft-breakdown) TDDB [14]–[16]. Techniques of fault detection are necessary to achieve long lifetimes of devices by eliminating these faults.

Redundant circuits such as triple modular redundancy (TMR) [17] can be used to detect logic faults in addition to fault tolerance. Even if some redundant modules become faulty in these circuits, the majority of their outputs is presumed to be correct, and faulty modules are detected and isolated by reconfiguring them. Another popular approach is doing periodical self-tests, and this is attracting more attention due to its lower cost than that of the redundancy-based approach. Inoue et al. [18] proposed a self-test method that enabled all the functions of BEs to be periodically configured and tested by changing test frames on coarse-grained dynamically reconfigurable processors (DRPs).

On the other hand, it is more complex to detect delay faults than logic faults, because it is essential to apply two test vectors to activate gates and paths of interest. Moreover, even after delay faults on some paths have been detected, components that significantly degrade on the paths cannot be easily identified. Kameda et al. [19] proposed a scheme to predict potential path delay faults in the near future on coarse-grained reconfigurable devices. Problematic components were identified through iterative BE replacements and path delay testing.

We supposed that such fault detection mechanisms were embedded in the target architecture in this research and assumed that faults could be predicted before they were exposed as errors, while fault detection mechanisms are still being researched. This paper therefore uses the term fault avoidance, as mentioned in Sect. 1.

2.2 Basic Reconfigurable Architecture for Fault Avoidance

Figure 2 outlines the target architecture used for this study, which is based on the coarse-grained dynamically reconfigurable architecture introduced by Alnajjar et al. [20]. Identical BEs in this architecture are aligned repeatedly in a two-dimensional array. Each BE is connected to adjacent BEs in four directions with two wires. Interconnections are configured with eight switches, and these switches and its configuration information are included in the BEs. Each BE contains a functional unit (FU) having an arithmetic logic unit (ALU) and shifter. FU receives one or two inputs (A and/or B) from {N0, N1, E0, E1, S0, S1, W0, W1}, and performs both arithmetic and logic operations for them. Output direction of the FU is also selected by configuration information in the BE. Each BE is assumed to have a self-testing mechanism and faulty BEs can be identified before their lifetimes end.

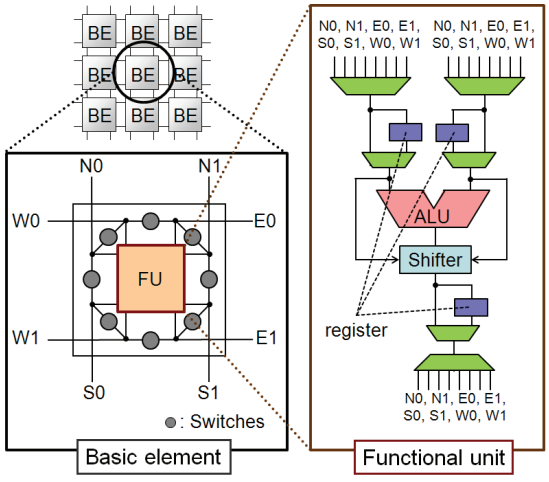


Fig. 2 Basic reconfigurable architecture.

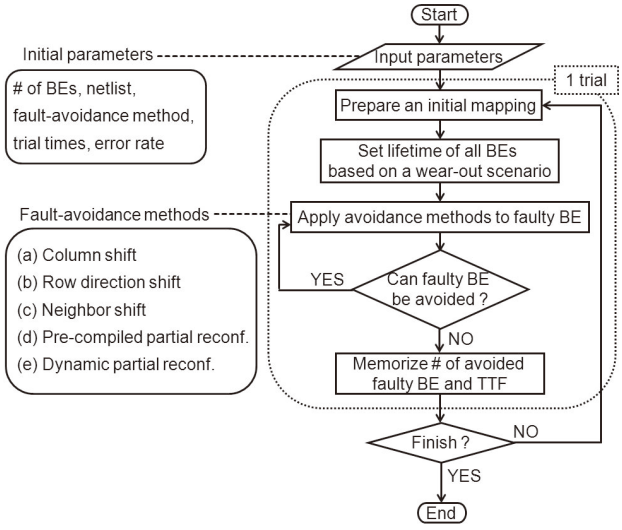


Fig. 3 Procedure to evaluate device lifetimes.

2.3 Procedure to Evaluate Lifetimes

We evaluated the time-to-failure (TTF) for all devices by simulating the fault-avoidance procedure for a wear-out scenario. We randomly generated sets of wear-out scenarios in a Monte Carlo manner and applied them to the devices in the simulations. Then, a statistical distribution of TTF was obtained. This evaluation was repeated for each fault-avoidance method.

Figure 3 is a flow diagram of the procedure we used to evaluate lifetimes in this study. First, initial parameters, such as the number of BEs, a netlist of target circuits, and fault-avoidance methods for evaluation, are given. The details on these fault-avoidance methods will be described in Sect. 3. We prepared initial mapping tailored to the specified fault-avoidance method. We next randomly generated a wear-out scenario according to a fault distribution and assigned the lifetimes of used BEs, which determined the tem-

poral sequence of BE faults. The wear-out scenarios we applied in this work will be briefly explained in Sect. 5.1. BE replacement with the specified fault-avoidance method was simulated after that. When fault avoidance could not be completed, the lifetimes of device ended, and both the number of eliminated faulty BEs and TTF were recorded. This fault-avoidance simulation for a wear-out scenario was one trial and was repeated to obtain statistics.

3. Fault-Avoidance Methods for Wear-Out Scenarios

This section outlines and explains the classification of five fault-avoidance methods we evaluated.

3.1 Classification of Fault-Avoidance Methods

We classified fault-avoidance methods on reconfigurable devices in Table 1. First, the fault-avoidance methods were divided into 'static allocation' and 'dynamic allocation' in terms of spare BE allocation. The spare BEs to eliminate a faulty BE are specified beforehand in 'static allocation' group. Namely, pairs between a faulty BE and spare BEs are fixed. Then, the number of avoidable faults can be roughly predicted and the device lifetime can be stably extended. On the other hand, a new mapping to eliminate the faulty BE is computed on the fly in the 'dynamic allocation' group. The number of avoidable faults and consequent lifetime extensions are less predictive, since they depend on many factors such as the initial mappings, the remapping algorithm, and the allowed downtime for preparing new configuration information.

Second, we examined fault-avoidance methods from the view point of the amount of hardware overhead. The hardware overhead generally tends to be large in such methods where successive fault-avoidance is carried out using dedicated hardware for bypassing faults. In contrast, that is small in such methods where fault avoidance is performed exploiting inherent reconfigurability with the help of embedded CPUs and memories.

The following subsections explain the five fault-avoidance methods we evaluated, which are classified in Table 1 and Fig. 4. These are (a) column shift, (b) row direction shift, (c) neighbor shift, (d) pre-compiled partial reconfiguration, and (e) dynamic partial reconfiguration.

Although usable resources such as wires and switches in methods (a) through (c) to both avoid various faults and replace them with spare BEs are originally limited, we assume that ideal fault avoidance is undertaken ignoring hardware limitations compared with methods (d) and (e) unless specially noted, since a discussion on differences originat-

Table 1 Classification of fault-avoidance methods.

Spare BE allocation	Hardware overhead	Typical methods
Static allocation	Considerable	(a) Column shift
		(b) Row direction shift
		(c) Neighbor shift
	Small	(d) Pre-compiled partial reconf.
Dynamic allocation		(e) Dynamic partial reconf.

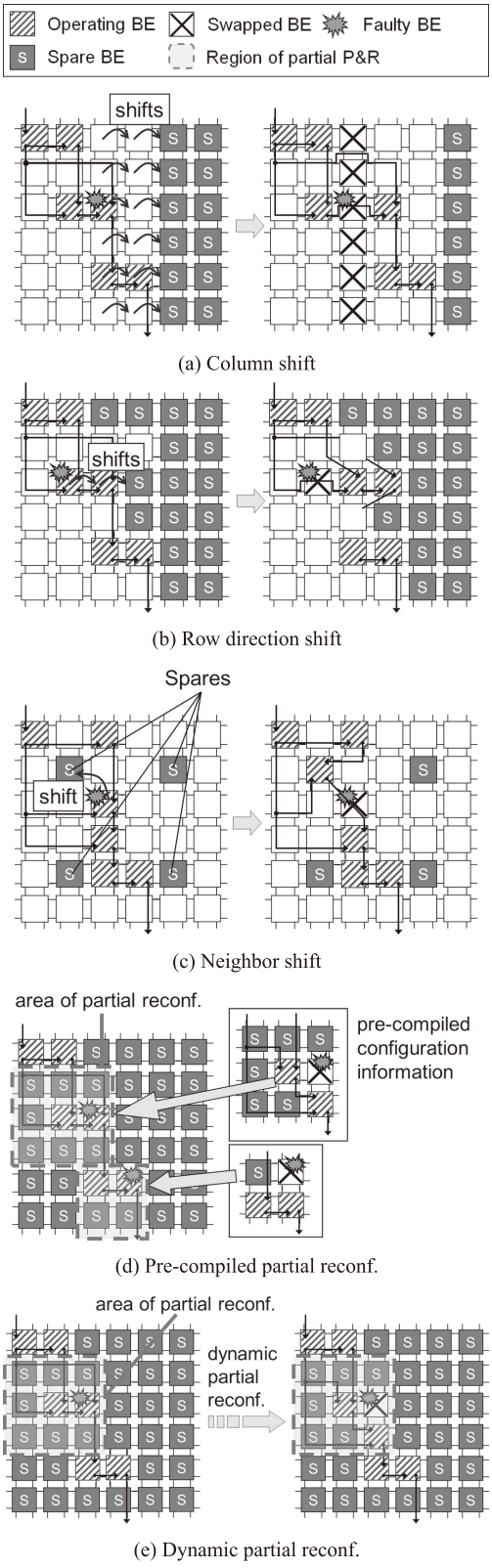


Fig. 4 Five fault-avoidance methods.

ing from detailed implementations is beyond the scope of this paper.

3.2 Column Shift

[21], [22] presented an approach that used column (row) redundancy on a reconfigurable device. We implemented (a) the column shift in Fig. 4(a), by referring to this approach. First, an initial mapping is generated so that the right columns are kept as spare BEs. When a BE becomes faulty, each column on the right of the faulty BE is shifted to the right by one column.

The possible number of fault avoidances in this approach is often limited due to the fact that one spare column is entirely consumed by one faulty BE. Additional wires and switches are necessary as a hardware support to bypass faulty BEs horizontally.

3.3 Row Direction Shift

[23], [24] proposed a scheme where a faulty BE was eliminated using a spare on the same row. Row direction shift was implemented according to Fig. 4(b) by referring to their scheme.

An initial mapping is generated so that the BEs at right were retained as spare BEs, which is similar to (a) column shift. A faulty BE is eliminated by shifting the BEs on the right of the faulty BE to the right. Unlike (a) column shift, the elimination of the faulty BE is completed within a single row. Although their data flow is limited to one direction (from north to south) [23], this limitation is relaxed and all directions are allowed in our study.

A considerable number of additional wires and switches are necessary to bypass faulty BEs and to compensate for the vertical mismatch originating from BE shifting in order to achieve this fault avoidance and detour signal interconnections.

3.4 Neighbor Shift

Doumar et al. [5] pointed out that the downtime to eliminate faulty BEs is an important metric, and the amount of BE shifting should be kept low. Neighbor shift was implemented according to Fig. 4(c) to achieve this purpose. Spare BEs are uniformly distributed in the BE array in this method. One of the neighboring spare BEs around a faulty BE is selected for avoidance.

Once a spare BE is consumed in this method, other surrounding BEs have less possibility of being replaced in the future. Multiple faults in neighboring BEs cannot be avoided where each spare is shared by several BEs. Further, preliminarily distributed spare BEs are obstacles and degrade routability on devices. Additional wires and switches are necessary to detour signals through hardware support.

3.5 Pre-Compiled Partial Reconfiguration

[25], [26] proposed a method of fault recovery by selecting a proper configuration from ones prepared beforehand and reloading it. This idea of a pre-compiled partial reconfiguration was implemented according to Fig. 4(d). Partial placement and routing (P&R) are simulated in advance by assuming that one of BEs became faulty, and the result is stored as pre-compiled partial configuration information for remapping was stored. When a faulty BE is detected, appropriate pre-compiled configuration information is applied to avoid the fault.

In this method, as multiple sets of configuration information to avoid not only a first faulty BE but also an n-th faulty BE based on the mapping after n times avoidance are pre-compiled and stored in memory, longer MTTF can be expected. On the other hand, the appropriate sets to load for avoiding a new faulty BE depends on the history of faulty BEs avoided so far, more precisely the sequence of faulty BEs avoided. The number of permutation of faulty BEs explodes exponentially as the number of avoided BEs increases, and it is difficult to store the pre-compiled configuration sets for every permutation. We thus supposed that this method stored the configuration information at least one faulty BE can be avoided.

This method requires fewer wires and switches than methods (a) through (c) because ordinary reconfigurability is used for fault avoidance. Instead, extra memory for storing spare configuration information is required. The details on partial P&R for reconfiguration will be explained in Sect. 4.

3.6 Dynamic Partial Reconfiguration

[7], [27]–[29] presented a self-repair approach with dynamic reconfiguration on FPGAs. This (e) dynamic partial reconfiguration is applied to the target reconfigurable architecture in Fig. 4(e). An initial mapping is generated without a deliberate allocation of spare BEs. When a BE becomes faulty, partial reconfiguration is performed on the fly using a processor. The device was down until a new configuration is generated and loaded.

Most of advantages and disadvantages are the same as those with method (d). The main difference is the requirement of a processor for dynamic partial reconfiguration instead of extra memory. The major drawback is the downtime that occurred when both calculating dynamic P&R and reconfiguring.

4. Implementation of Placement and Routing

Partial P&R and spare/fault-aware P&R need to be implemented to evaluate device lifetime with methods (c), (d) and (e). We extended the popular P&R algorithms of VPR [30] and PathFinder [31] to enable partial P&R and spare/fault-aware P&R in this study. These extensions are explained in the following.

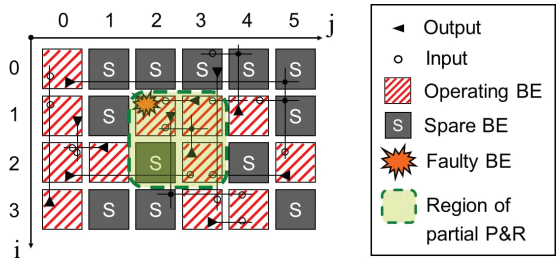
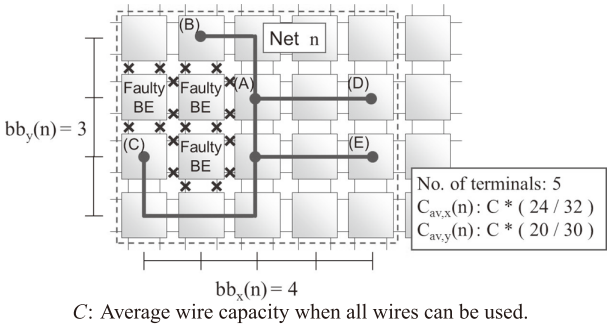


Fig. 5 Example of region selection for partial P&R.



C: Average wire capacity when all wires can be used.

Fig. 6 Example of spare/fault-aware computation of objective function.

4.1 Procedure for Partial Placement and Routing

We need to specify the region for partial P&R in performing it. A smaller region is preferable, because the downtime involved with reconfiguration is shorter and less memory for storage is required. The procedure adopted in this evaluation involves two steps.

1. Find the minimum rectangle including BEs that are adjacent to and connected to the faulty one.
2. If the rectangle includes at least one unused BE, partial P&R are carried out in this rectangle. Otherwise, the rectangle is expanded toward a direction in which more neighboring unused BEs are included. BEs only used for wires are regarded as unused BEs.

Figure 5 shows an example of region selection. Supposing BE(1, 2) becomes faulty, BE(2, 2) and BE(1, 3) are adjacent to and connected to the faulty BE, and thus should be included in the 2×2 rectangle. Here, as BE(2, 2) using only wires is included, this 2×2 rectangle surrounded by a broken line is then the region for partial P&R.

Once the region is selected, fault-aware P&R, which will be explained in the next subsection, is carried out in the region. When fault-aware P&R are successful, it is completed. Otherwise, the rectangle is expanded in a direction that includes more unused BEs and fault-aware P&R are rerun. This expansion and fault-aware P&R are repeated until partial P&R succeed.

4.2 Spare/Fault-Aware Placement and Routing

Spare-aware P&R are necessary in method (c), since BEs allocated for spare in advance cannot be used for mapping. The partial P&R in methods (d) and (e) require fault-aware P&R not to use faulty BEs. This means that spare-aware and fault-aware P&Rs are identical. The following explains the extension of VPR to spare/fault awareness. Here, both FU and wires in a spare/faulty BE are assumed to be unusable.

The objective function for placement in VPR, which corresponds to prospective wire length, is expressed as [30]

$$Cost = \sum_{n=1}^{N_{nets}} q(n) \left[\frac{bb_x(n)}{C_{av,x}(n)} + \frac{bb_y(n)}{C_{av,y}(n)} \right], \quad (1)$$

where N_{nets} is the total number of nets, $q(n)$ means the weight of net n, depending on the number of terminals, and $bb_x(n)$ and $bb_y(n)$ are the width and height of the bounding box for net n. Here, $C_{av,x}(n)$ means the average routing capacity of horizontal wires and $C_{av,y}(n)$ means that of vertical wires.

Figure 6 has an example that explains how the calculation of the objective function changes with some faulty BEs. When faulty BEs are located as shown in the figure, BE(C) cannot be connected to BE(A), BE(B), BE(D), and BE(E) with the shortest path. A detour is necessary in this case and $bb_y(n)$ increases from two to three. Moreover, spare BEs degrade routing capacity. Due to the location of spare BEs in this example, the number of usable horizontal wires is decreased from 32 to 24 and that of vertical wires is decreased from 30 to 20. Consequently, the horizontal and vertical average capacities decrease from C to $C * 24/32$ for the former and $C * 20/30$ for the latter.

The above modification works well for small applications. However, routable placements cannot be obtained for larger applications, because wire congestion has not been considered in Eq. (1). In fact, for large applications such as 1,024-point FFT, initial routing was mostly aborted due to BEs being so compactly placed that wire resources were insufficient. We added terms to solve these problems that were relevant to wire congestion to Eq. (1) by referring to Lou et al. [32];

$$Cost = (1 + ov \cdot w) \cdot \sum_{n=1}^{N_{nets}} q(n) \left[\frac{bb_x(n)}{C_{av,x}(n)} + \frac{bb_y(n)}{C_{av,y}(n)} \right], \quad (2)$$

where ov is the summation of wire utilization overflows from available wire resources. To be concrete, each basic element has two horizontal tracks and two vertical tracks as shown in Fig. 2. The overflow of the i-th basic element ov_i is expressed as $max(0, NH_i - 2) + max(0, NV_i - 2)$, and ov is $\Sigma_i ov_i$, where NH_i (NV_i) is the number of horizontally (vertically) routed wires in the i-th basic element. Note that in the placement phase, each net has not been routed, and then NH_i and NV_i are probabilistically estimated referring to [32]. Parameter w means the weight of wire congestion. By adding $ov \cdot w$ to the cost function, placement results with higher congestion (i.e. larger $ov \cdot w$) are less likely selected, and we can expect to have a better placement result which is easy to route in the following routing phase. Let us demon-

Table 2 Improved success rate in initial generation of layout due to additional terms in Eq. (2) (FFT).

w	0	1	10	100	1,000
Success rate	2.2%	70.1%	96.2%	98.1%	95.9%

strate the contribution of the modified objective function. Table 2 lists the success rate for generating routable initial mappings where it is clear that larger w ($\geq$10) helps generate initial mappings successfully with an over 95% possibility. The w was experimentally set to 100 throughout this work.

The PathFinder algorithm [31] based on maze routing was improved for spare/fault-aware routing so that wires passing through spare/faulty BEs could not be used.

5. Experimental Results

This section presents experimental results obtained from evaluating lifetimes and compares the five fault-avoidance methods. Section 5.1 first describes the experimental setup. Section 5.2 explains our evaluation and comparison of statistics such as MTTF, the standard deviation of the TTF distribution, and the 10th percentile of TTF, which we used to figure out the features of these methods. Section 5.3 points out that preventing BEs from aging is important to achieve MTTF enhancements. In addition, Sect. 5.4 associates achievable MTTFs in 'neighbor shift', 'pre-compiled partial reconfiguration', and 'dynamic partial reconfiguration' with their initial mappings through case studies. Finally, Sect. 5.5 summarizes the features of the five fault-avoidance methods.

5.1 Experimental Setup

The lifetimes of BEs were supposed to follow a Weibull distribution, which is widely used for evaluating device reliability (e.g. [33]). The scale parameter, λ, was set to 1.0×10^{-6} [/h], and the shape parameter, m, was set to two assuming aging process. Figure 7 shows an example of the lifetime distribution of 10,000 BEs resulting from the Weibull distribution, where the number of faulty BEs reaches a peak at $t = 8.0 \times 10^5$ [h] and then gradually decreases.

We assumed in this evaluation that spare BEs would not age owing to selective power gating or other mechanisms except for those in Sect. 5.3. The additional wires, switches, and I/O interface for each fault-avoidance method were built-in. Also, connections between a BE array and outside components were not considered to simplify the evaluation. There were 100 TTF evaluations for each fault-avoidance method.

Table 3 summarizes target applications that have different scales. Here, mapping from a data flow graph and netlist generation were carried out by Imagawa et al. [34]. A square BE array, whose utilization was nearly 50%, was prepared for each application. Let us take the 14-tap FIR filter in Fig. 8 as an example to explain the way in which we prepared the initial mappings using Fig. 9. First, P&R were performed within the left side of the BE array for methods (a) and (b), and the rightward columns were retained as spare BEs to avoid faulty BEs as much as possible. Nine spare BEs were regularly located at the center of each 3×3 region for method (c) to ensure that at least one faulty BE could be reliably avoided. Evaluation was carried out for method (d) with an initial mapping that could avoid any first faulty BE regardless of its location and could carry out such avoidance with the least information on mapping modifications

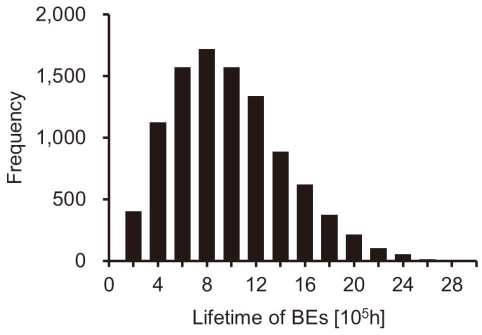


Fig. 7 Lifetime distribution of 10,000 BEs resulting from Weibull distribution.

Table 3 Scale of target applications.

Target application	# of used BEs	# of rows and cols. of BE array	BE utilization
Nega-posi	12	5×5	48.0%
x differ	20	6×6	55.6%
14-tap FIR filter	40	9×9	49.4%
Gaussian filter	73	12×12	50.7%
FFT (1,024-p, radix-2)	111	15×15	49.3%

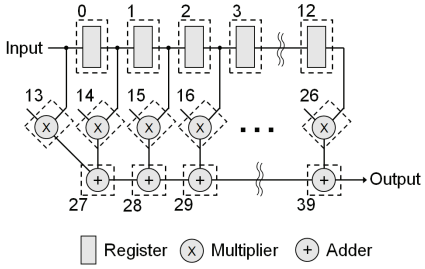


Fig. 8 14-tap FIR filter.

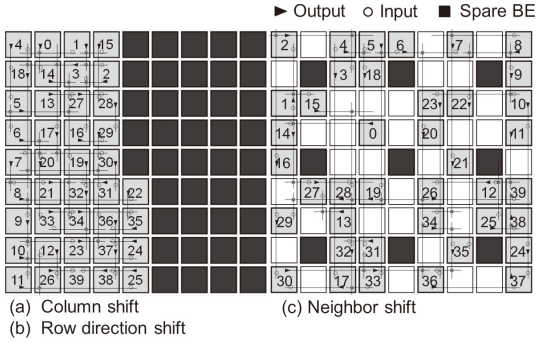


Fig. 9 Initial mappings of 14-tap FIR filter with methods (a) through (c).

to store. Finally, different initial mappings were generated for every trial in Fig. 3 for method (e). A scenario for BEs to wear out was also randomly generated for each trial.

In this paper, the run-time of partial P&R in method (e) is not explicitly considered in MTTF evaluation, since this work assumes that a faulty BE is predicted in advance before faults start to emerge in that BE, as explained in Sect. 2.1. When the time interval between fault prediction and fault emergence is longer than the run-time of dynamic P&R, the fault-avoidance succeeds. In this paper, we supposed that the time interval was always longer than the run-time of P&R. On the other hand, the CPU time needed for dynamic P&R heavily depends on the performance of CPU available and how much computation time can be allotted for dynamic P&R. In addition, the time interval depends on test methods and aging processes. Thus, evaluating MTTF degradation due to the run-time of partial P&R is not straightforward, and hence this is not considered.

5.2 Enhanced Lifetimes with Fault Avoidance

This section discusses the characteristics of the five fault-avoidance methods by comparing statistics of lifetime enhancements.

5.2.1 MTTF Enhancement

Figure 10 shows the MTTFs of target applications attained with the five fault-avoidance methods. We can see that they achieved different MTTFs. Compared with no avoidance, method (b) most decidedly extended MTTFs to a range of 4.0 to 5.7 times. It owed such long extensions to a large number of additional switches and wires to bypass faulty BEs. To clarify this, Fig. 11 shows MTTFs when the number of avoidable BEs in each row is limited, whereas Fig. 10 assumes that it is unlimited. As the number of avoidable BEs in each row is incremented involving increased hardware overhead, both the number of avoided faulty BEs and MTTFs increase linearly. If the number of avoidable faulty BEs in each row is limited to one, the MTTF becomes 3.14×10^5 h, which is less than that of method (a) and comparable to that of method (c). On the other hand, method (e) also extends the MTTFs of 'nega-posi', 'x differ', and '14-tap FIR filter' significantly to over 7.74×10^5 h exploiting the reconfigurability of devices.

Next, a metric called the MTTF enhancement ratio, which is defined as the MTTF increase with each method divided by the MTTF increase with ideal avoidance by utilizing all spare BEs, was computed in Fig. 12 to further compare MTTF enhancements. The MTTF enhancement ratio expressed as a percentage becomes 100% when all the spares are used for fault avoidance, i.e. the number of avoided faulty BEs is equal to the number of spares. The MTTF enhancement ratio decreases as the efficiency of spare usage for MTTF improvements degrades. The MTTF enhancement ratio in methods (a) and (b) was stable and less dependent on target applications because the ratios of available spare BEs to the total BEs were the same. On the other hand, the MTTF enhancement ratio in methods (d) and (e) varied depending on applications. To understand these variations, we examined wire utilization that were directly related to the difficulty of implementing P&R. Figures 13–15 plot the relationships between MTTF and the worst wire utilization ratio in whole, 5×5, and 3×3 regions of the initial mapping, respectively. Here, the average for the worst-wire utilization ratio over 100 trials was used, and each dot corresponds to each application. The correlation coefficients between the wire utilization ratio and MTTF in Figs. 13–15 correspond to −0.752, −0.980, and −0.984. These results indicate that the wire utilization ratio, particularly in the most congested small region, is well correlated with MTTF for the fault-avoidance methods using partial reconfiguration. Further analysis of the dependence of MTTF enhancement

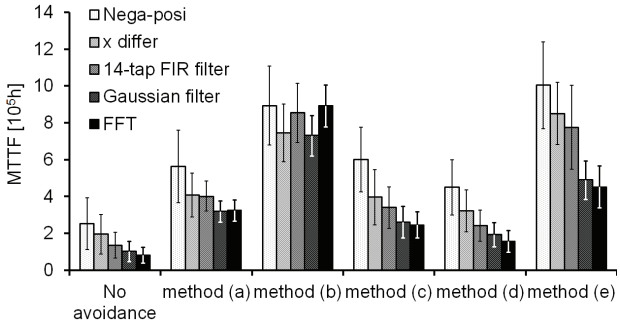


Fig. 10 MTTFs of target applications with five fault-avoidance methods (Error bars represent standard deviations).

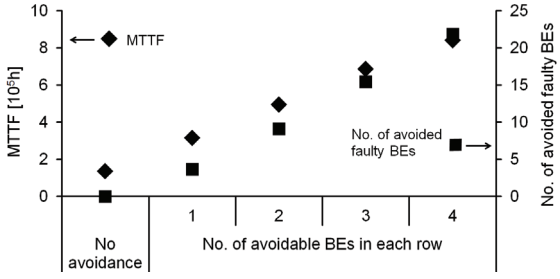
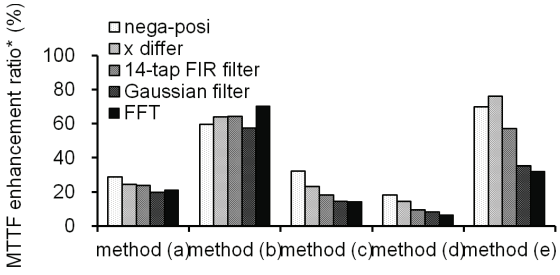


Fig. 11 MTTF and avoided faulty BEs of method (b) in which number of avoidable BEs in each row is limited (14-tap FIR filter).



*MTTF enhancement ratio = $\Delta MTTF_{method(n)}/\Delta MTTF_{ideal}$

Fig. 12 MTTF enhancement ratio for target applications with five fault-avoidance methods (Error bars represent standard deviations).

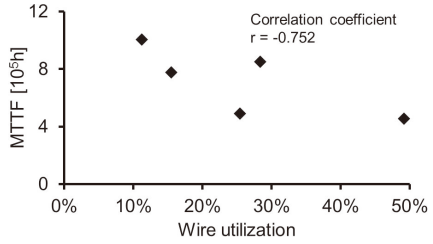


Fig. 13 Relationship between wire utilization ratio and MTTF in method (e).

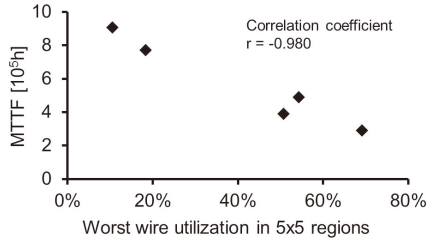


Fig. 14 Relationship between wire utilization ratio in most congested 5×5 region and MTTF in method (e).

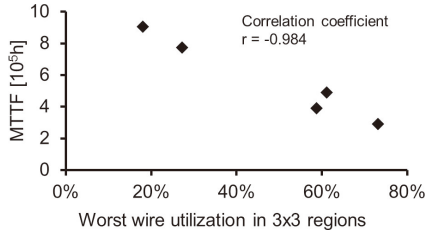


Fig. 15 Relationship between wire utilization ratio in most congested 3×3 region and MTTF in method (e).

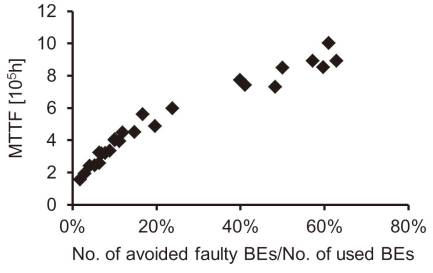


Fig. 16 Relationship between MTTFs and ratio of avoided faulty BEs to used BEs found with five fault-avoidance methods.

on the initial mapping will be discussed in Sect. 5.4.

Figure 16 plots the relationship between the ratio of the number of avoided faulty BEs divided by the number of used BEs and the MTTFs of the five fault-avoidance methods, which intends to simplify the discussion on the number of avoided faulty BEs required to achieve the target MTTF taking into account the scales of applications. This figure indicates that the ratio uniquely corresponds to MTTF. In addition, the improvement in MTTF gradually saturates as the ratio increases. An interesting observation is that the MTTF at 60% is close to that of a single BE (8.78×10^5 hours). This

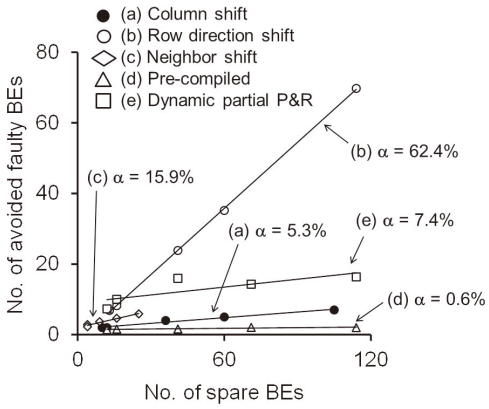


Fig. 17 Relationship between number of spare BEs and average number of avoided faulty BEs (α is slope of each regression line).

means that by using spare BEs, we can extend the MTTF of an application to that of an BE. However, further extensions are not easy to obtain, since the number of necessary spare BEs rapidly increases. Thus, the number of fault avoidances needs to be scaled up according to the number of used BEs to maintain longer MTTFs for large-scale applications.

5.2.2 Efficiency in Spare Usage

The discussion in the previous section clarified that the MTTF was well characterized by the ratio of avoided faulty BEs to used BEs regardless of fault-avoidance methods. The difference in the MTTFs results from the efficiency in spare BE usage and hence we here compare fault-avoidance methods from this point of view. Figure 17 plots the relationship between the number of spare BEs and the average number of avoided faulty BEs with the five fault-avoidance methods, where each dot corresponds to each application. Note that all BEs not assigned to operations in methods (d) and (e) are available for fault avoidance and regarded as spare BEs. We defined the efficiency in spare usage as the slope (α) of regression lines. Larger α means that spare BEs are well exploited to avoid faults.

Method (b) attains the largest α (= 62.4%) when the αs of the five fault-avoidance methods are compared. On the other hand, the α of method (d) is the smallest (= 0.6%); in other words, there are many unused spare BEs still remaining. In method (d), once a BE becomes faulty, corresponding information on the partial reconfiguration, which was preliminarily generated and stored in memory, is loaded and faults are avoided. Let us assume a second faulty BE. If the region of partial reconfiguration for the second faulty BE does not overlap that of the first, the second fault avoidance succeeds. However, if there is an overlapping region, it fails. This is why method (d) attained such low efficiency in spare usage. The difference in efficiency of spare usage becomes more than a hundred-fold by comparing method (d) and method (b).

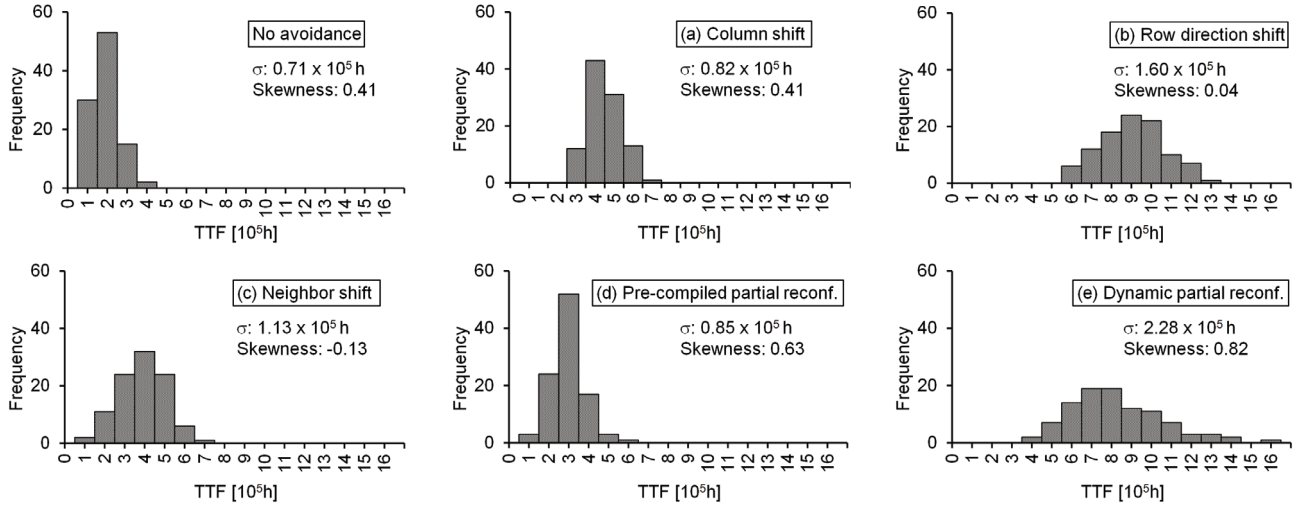


Fig. 18 TTF distributions for 14-tap FIR filter with five fault-avoidance methods.

5.2.3 TTF Distribution

Our evaluation of MTTF enhancements with the five fault-avoidance methods was discussed in the previous subsection. Designers, on the other hand, generally pay attention not only to MTTFs but also to TTF distributions. Figure 18 shows TTF distributions of the 14-tap FIR filter with the five fault-avoidance methods that were used to investigate the predictability of MTTF. Here, σ means the standard deviation of the distribution. Skewness is a statistic indicating the asymmetry of the distribution. A positive skew means that the tail on the right of the distribution is longer than that on the left, and a negative skew means the opposite.

The σs of the TTF distributions in methods (b) and (e), are much larger than that without fault avoidance, and the TTF distributions are widely spread compared with the others. There are two reasons for this. The first is the number of avoided faulty BEs has spread more than the others. More concretely, the σs of the number of avoided faulty BEs with methods (b) and (e) correspond to 5.31 and 6.29, whereas the σs of that with methods (a), (c), and (d) are 0, 1.72, and 0.73. The second reason is that the interval between successive faults was changing and it became longer as the number of avoided BEs increased due to the Weibull distribution in Fig. 7, which made the TTFs become more diverse. On the other hand, compared with no fault avoidance, σ and skew remain almost unchanged for method (a), which indicates the shape of the TTF distribution is roughly maintained. The enhanced lifetime attained with method (a) is stable and predictable.

Table 4 lists the statistics for the TTF distributions of the 14-tap FIR filter. The medians and 10th percentiles of TTFs correspond to lifetimes that 50% and 90% of chips can attain. The 10th percentiles of TTFs with methods (a) through (e) are at least three times larger than those with no fault avoidance, and this improvement in 10th percentiles is larger than those for MTTFs and medians, which is an advantageous property for applications that demand reliability. Thus, fault avoidance effectively improves not only the average lifetime but also gains smaller percentiles for the lifetimes.

Table 4 Statistics for TTF distributions (14-tap FIR filter).

Fault-avoidance methods	MTTF [10^5h]	Median [10^5h]	10th percentile [10^5h]
No fault avoidance	1.35	1.29	0.43
method (a)	4.01	3.88	2.95
method (b)	8.54	8.48	6.05
method (c)	3.40	3.51	1.84
method (d)	2.42	2.34	1.28
method (e)	7.75	7.51	5.01

Table 5 Different scales of BE array (14-tap FIR filter).

# of rows & columns of BE array	BE utilization
8×8	62.5%
9×9	49.4%
12×12	27.8%

5.2.4 BE Utilization Ratio and MTTF Scalability

The BE utilization ratio, which is also associated with the wire utilization ratio, is one of major factors determining the difficulty of P&R. We then evaluated MTTF scalability in terms of the BE utilization ratio; in other words, how MTTFs could be improved as the number of spare BEs increased.

Table 5 lists the setup in the array size for the 14-tap FIR filter. Here, the initial mappings for 'BE 8×8' and 'BE 12×12' were generated in the same way as that explained in Sect. 5.1. As for method (c), we first assumed that a BE at the center was designated as a spare for each 3×3 BEs similarly to the previous experiments, even when the array size was changed. This assumption considered a situation that the array having the same functionality yet a different size was used.

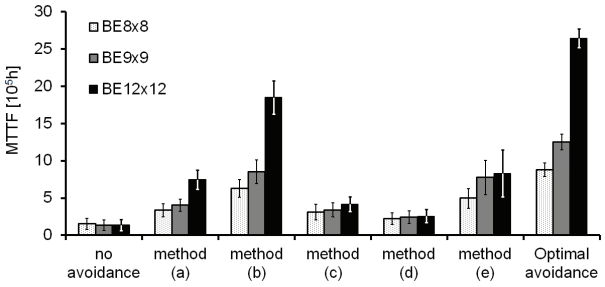


Fig. 19 MTTFs of 14-tap FIR filter with different scales of BE arrays (Error bars represent standard deviations).

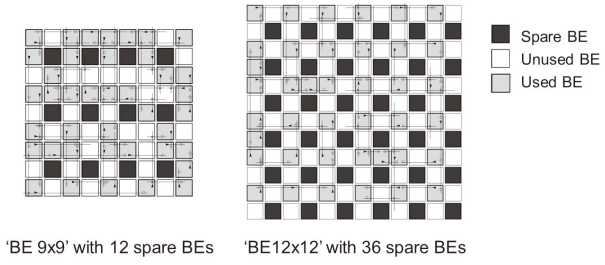


Fig. 20 Additional initial mappings of 14-tap FIR filter for method (c).

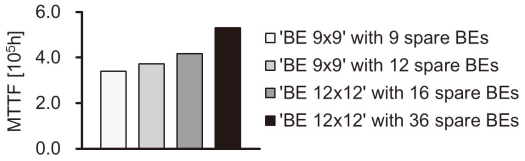


Fig. 21 MTTFs of 14-tap FIR filter with different scales of BE arrays for method (c).

Figure 19 shows the MTTFs. The MTTFs in methods (a) and (b) became longer as the array became larger, i.e. as BE utilization ratio decreases. An increase in columns fully consisting of spare BEs contributed to longer MTTFs in method (a), and method (b) steadily converted any additional spare BEs into improved MTTFs. On the other hand, while a similar tendency was observed in method (e), the improved MTTF from 'BE 9×9' to 'BE 12×12' was quite small and MTTF scalability with method (e) was worse than those with methods (a) and (b). The efficiency in spare usage degraded in method (e) as more spare BEs became available. Method (d) basically prepares a set of mapping modifications for every single faulty BE, and in some cases multiple modifications can fortunately be applied. However, this did not occur too often. To make matters worse, regions of mapping modifications were occasionally very large, which prevented method (d) from avoiding multiple faulty BEs even when the array was large. Moreover, MTTF enhancement in method (c) with respect to increase in array size was quite limited compared to methods (a) and (b). It was due to that spare BEs were just regularly allocated at the center of each 3×3 region and other unused BEs could not be used for fault avoidance in method (c).

For method (c), on the other hand, another situation in

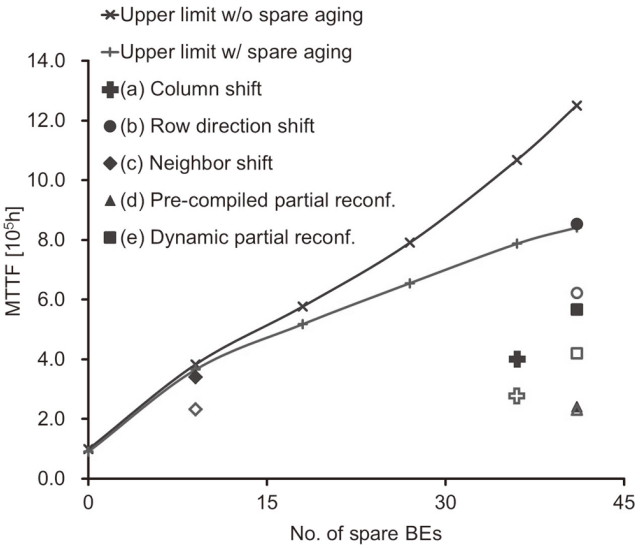


Fig. 22 Relationship of 14-tap FIR filters between number of spare BEs and MTTFs with and without spare aging. (Markers filled by black are without spare aging, and ones filled by white are with spare aging.)

which more than one-ninth BEs are designated as spares can be reasonably considered while the additional hardware for fault avoidance increases. Thus, we permitted the overlapping of 3×3 regions while keeping the same assumption that the center of 3×3 regions is the spare and can substitute one of adjacent eight BEs. If a BE is adjacent to two spare BEs and becomes faulty, either of the two can be used as the substitute. Under this situation, we performed further evaluation of MTTFs in method (c) with additional initial mappings in Fig. 20. The ratio of the spare BEs was scaled by changing the portion of the overlapping. Figure 21 shows the MTTFs. The MTTF of 'BE 9×9' with 12 spare BEs is 1.09 times as large as that of 'BE 9×9' with nine spare BEs. Similarly, the MTTF of 'BE 12×12' with 36 spare BEs is 1.27 times as large as that of 'BE 9×9' with 16 spare BEs. These results demonstrate that method (c) can scale the MTTF by allocating more spares with increase in hardware overhead. However, the MTTF of 'BE 12×12' with 36 spare BEs attained only 71.2% (= 5.31/7.46) of that of 'BE 12×12' in method (a). This can be explained as follows. When a BE becomes faulty in method (a), not only the faulty BE but also other aged but not faulty BEs belonging to the same column are swapped with fresh spare BEs. The swapped BEs are less likely to become faulty in the near future, which helps improve the MTTF.

5.3 Aging Effect on Spare BEs

The evaluations presented thus far assumed that spare BEs would not age. In actual circuits, on the other hand, BEs not assigned to any operations can still suffer from the aging effect. We evaluated how the aging of spare BEs affected lifetime enhancement.

Figure 22 plots the relationships of 14-tap FIR filters between the number of spare BEs and MTTFs with and

without spare aging. Here, spare aging means that the lifetimes of spare BEs are also supposed to follow Weibull distributions as well as the lifetimes of used BEs. The figure has an upper limits where available spare BEs can necessarily be used for fault avoidance, i.e. the number of avoided faulty BEs is equal to the number of spare BEs. Figure 22 indicates that MTTFs significantly decrease due to spare aging. The MTTF for method (b) decreases to 72.8% (= 6.22/8.54). Similarly, the MTTF for method (e) decreases to 74.0% (= 4.20/5.67). Thus, when spare BEs also degrade, the advantage of fault avoidance to enhance lifetimes considerably decreases. Therefore, mechanisms to keep spare BEs fresh, such as power cutoffs, are required to make the best possible use of fault-avoidance methods. On the other hand, some aging effects, such as total doze effect in space environment, may not be eliminated from spare BEs. More specific evaluation in terms of operating environment and aging processes is one of the future works.

5.4 Initial Mapping for Fault Avoidance

Initial mappings affect the number of avoidable faulty BEs for methods (c) to (e). MTTF was evaluated for different mappings. The initial mappings for method (c) in which the numbers of used BEs around each spare BE are averaged out lengthen lifetimes. The two different mappings in Fig. 23 were prepared for evaluation to demonstrate this tendency. Map A was automatically generated as explained in Sect. 5.1, and map B was manually generated such that there were four or five used BEs around spare BE. Table 6 listing the MTTFs with initial mappings of Fig. 23 indicates that the MTTF of map B is slightly but certainly larger than that of map A as we expected. Thus, the enhanced lifetime of method (c) depends on the initial mapping.

The initial mappings for methods (d) and (e) that have a high degree of compatibility with partial reconfiguration are expected to better contribute to extending lifetimes. The initial mappings for methods (d) and (e) are generated by P&R explained in Sect. 4 in a comparative experiment, according to the rule that all BEs assigned to functional operations were regularly placed as on a checker-board. Figure 24 shows different initial mappings from the ten generated for evaluation. The density of all 3×3 regions in these mappings is almost the same. Figure 25 describes the enhanced MTTF for ten initial mappings. The maximum gaps of the enhanced MTTF reach 4.0% (map #1 vs. map #8) for method (d) and 10.1% (map #6 vs. map #7) for method (e). Moreover, the discussion in Sect. 5.2.1 suggests that these gaps originated from the differences in wire utilization.

Figures 26 and 27 plot the relationships between the wire utilization of each initial mapping and the number of avoided faulty BEs with methods (d) and (e). Only fewer faulty BEs were avoided in map #1 and map #6 due to higher wire utilization, and in contrast, more faulty BEs were successfully avoided in map #8 and map #7 due to lower wire utilization. We concluded that even when spares were similarly spread out, MTTFs could differ depending on the initial P&R, especially wire utilization. From another point of

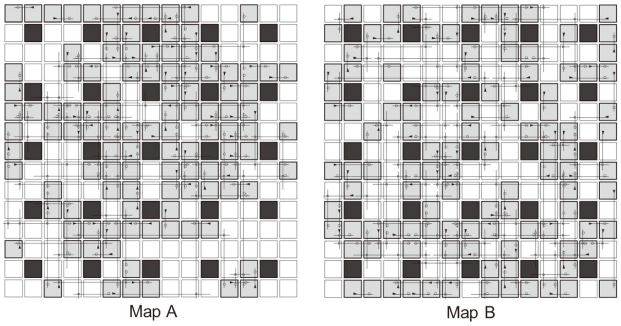


Fig. 23 Initial mappings of FFT for method (c).

Table 6 MTTFs of FFT with initial mappings shown in Fig. 23.

	MTTF [10^5 h]	
	Map A	Map B
No fault avoidance	0.85	
(c) Neighbor shift	2.46	2.55

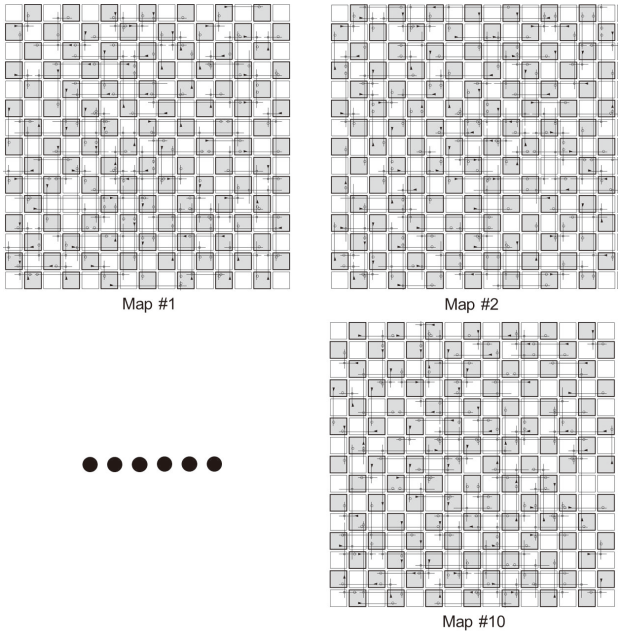


Fig. 24 Initial mappings of FFT for methods (d) and (e).

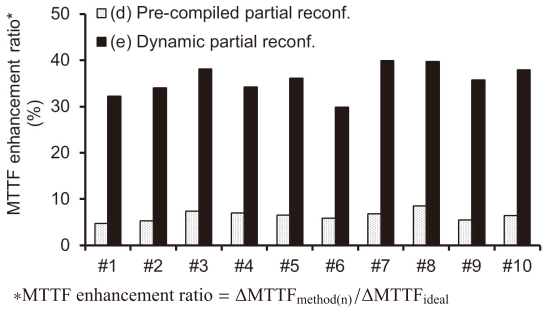


Fig. 25 MTTF enhancement ratio of FFT for initial mappings in Fig. 24.

view, there is room to maximize the enhanced lifetimes even with the same number of spare BEs by obtaining a better initial mapping. We intend to drive such a problem formulation as an optimization problem and include its algorithmic solution in our future work.

5.5 Summary & Discussion

Table 7 summarizes the performance of the five fault-avoidance methods. First, method (b) in a comparison of MTTFs attained the largest MTTF improvement of (3.5 times–10.8 times), and method (e) followed as the second highest (4.0 times–5.7 times). An MTTF enhancement ratio close to 70% was obtained for methods (b) and (e) in comparison with optimal fault avoidance. Method (b) achieved the highest value for the number of avoided faulty BEs per spare BE (62.4%), demonstrating the most efficient spare usage. Methods (a) and (d) provided predictive lifetimes with smaller TTF variations in respect to standard deviation. Methods (b) and (e) achieved higher 10th percentiles of TTF in common with MTTF. We also confirmed through evaluation of five target applications that at least 2.4 times enhancement of 10th percentiles of TTF could undoubtedly be obtained with all the methods. The MTTF enhancements with methods (d) and (e) using partial reconfiguration were more dependent on the scale of applications than those of methods (a) through (c). Method (b) had the highest capability of controlling MTTFs by changing the number of spare BEs in terms of MTTF scalability, followed by methods (a) and (e).

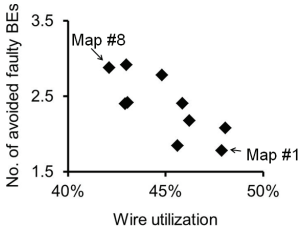


Fig. 26 Relationship between wire utilization and number of avoided faulty BEs with method (d) (FFT).

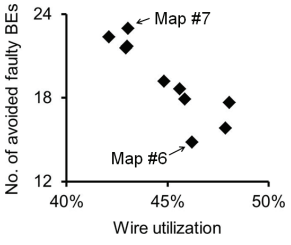


Fig. 27 Relationship between wire utilization and number of avoided faulty BEs with method (e) (FFT).

The characteristics of the five fault-avoidance methods can be summarized as follows. The main advantage of method (a) is its stable and reasonable enhancement of MTTFs with less dependence on applications. One of the main drawbacks of method (a) is that its efficiency with spare usage was low. Method (b) attained the highest MTTF enhancements at the cost of larger hardware overhead. Although both enhancements and scalability of MTTFs are low for method (c), at least one fault avoidance with a minimum number of spare BEs was guaranteed. Method (d) was not good at sequential fault avoidance and therefore its enhancement of lifetimes was the smallest. Method (e) achieved higher MTTF enhancements that were comparable with those of method (b) in some applications. The number of avoided faulty BEs in methods (d) and (e) was notably dependent on mappings and P&R results, which suggests MTTFs could be enhanced by improving initial and partial P&R.

In this paper, methods (d) and (e) are classified in terms of available resources, i.e. memory and CPU. Then, method (d) has a difficulty to avoid multiple faults, and method (e) needs long down time to preform partial P&R. However, assuming both memory and CPU are available, these two drawbacks can be eliminated. In this case, a possible method of fault avoidance is performed as follows.

1. Prepare configuration information to avoid every BE

Table 7 Summary of evaluations of fault-avoidance methods with five target applications. (Bold font means better performance in each row.)

	(a) Column shift	(b) Row direction shift	(c) Neighbor shift	(d) Pre-compiled partial reconf.	(e) Dynamic partial reconf.
MTTF	2.1–3.9×	**3.5–10.8**×	2.0–3.0×	1.7–1.9×	**4.0–5.7**×
MTTF enhancement ratio*	20–29%	**57–70%**	14–32%	7–18%	**32–76%**
Efficiency in spare usage	5.3%	**62.4%**	15.9%	0.6%	7.4%
Standard deviation of TTFs	**1.1–1.4**×	1.5–2.6×	1.3–1.6×	**1.1–1.3**×	1.6–3.2×
10th percentile of TTFs	4.2–8.7×	**8.3–25.4**×	2.9–5.4×	2.4–3.5×	**9.4–11.7**×
Dependence of applications	**low**	**low**	**low**	medium	high
Scalability	**medium**	**high**	low	low	**medium**
Description	Lifetime enhancement is stable, but efficiency in spare usage is low	Quite a high lifetime is obtained by large extra hardware overhead	At least one faulty BE is guaranteed to be avoided with a minimum # of spare BEs	# of avoidable faulty BEs is too small to enhance lifetime substantially	Efficiency of lifetime enhancement depends on mappings and P&R results

∗MTTF enhancement ratio = $\Delta MTTF_{method(n)}/\Delta MTTF_{ideal}$

and store the information in memory beforehand, which is the same with method (d).
2. Apply the prepared configuration information for the faulty BE, which is also the same with method (d).
3. Update configuration information to avoid every BE by using CPU and store the information in memory before the next fault occurs.
4. Repeat the steps of #3 to #4.

In method (e), dynamic P&R is performed after a fault BE is found. On the other hand, in this procedure, dynamic P&R is performed proactively at step 3, and its result is stored in memory. Note that all the information is not necessary to update, and dynamic P&R usually needs to be performed to avoid prospective faulty BEs in proximity to the previous faulty BE.

With this procedure, the downtime is the same with that of method (d), since the configuration information is loaded from the memory. In addition, dynamic P&R is performed for the mapping that has already avoided faulty BEs, and hence multiple faults can be avoided more probably, which is similar to method (e). This fault avoidance method is worth for further evaluation.

Thus far, we summarized and discussed lifetime enhancement achieved by fault-avoidance methods. The evaluation of life-time extension in this work is helpful to identify a few promising fault-avoidance methods in terms of required life-time extension, since the importance of lifetime depends on applications and environment. On the other hand, performance, such as delay, area and power dissipation, is another important factor to select the most appropriate fault-avoidance method. However, these performance metrics cannot be precisely evaluated without implementing the architecture, and their evaluation is time-consuming. An efficient way is to identify and select a few promising fault-avoidance methods referring the results in this paper and evaluate their performance with more concrete architecture implementations supposing specific target applications and operating environment.

6. Conclusion

Quantitative lifetime evaluations with representative five fault-avoidance methods on dynamically reconfigurable devices were undertaken. Our evaluation results indicated that lifetime enhancement ratio, expressed as a percentage, of devices was improved by up to 70% with 'row direction shift' and 'dynamic partial reconfiguration'. On the other hand, 'column shift' was suitable to obtain a certain level of MTTF enhancement. We also found that to make the maximum use of fault-avoidance methods, spare BEs should be prevented from aging. Moreover, we clarify that fault-avoidance methods using partial reconfiguration have the potential to enhance MTTFs by generating mappings that have higher compatibility with partial P&R.

Acknowledgment

The authors would like to thank the project members of JST CREST of NEC Corp., the Kyoto University, Kyoto Institute of Technology, Nara Institute of Science and Technology, and ASTEM RI for the discussions we had with them.

References

[1] S.A. Sundberg, "High-throughput and ultra-high-throughput screening: Solution-and cell-based approaches," Current Opinion in Biotechnology, vol.11, no.1, pp.47–53, 2000.
[2] D. Ernst, N.S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," Proc. MICRO, pp.7–18, 2003.
[3] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.24, no.12, pp.1859–1880, 2005.
[4] O. Khan and S. Kundu, "A self-adaptive system architecture to address transistor aging," Proc. DATE, pp.81–86, 2009.
[5] A. Doumar, S. Kaneko, and H. Ito, "Defect and fault tolerance FPGAs by shifting the configuration data," Proc. DFT, pp.377–385, 1999.
[6] T. Koal and H.T. Vierhaus, "Optimal spare utilization for reliability and mean lifetime improvement of logic built-in self-repair," Proc. DDECS, pp.219–224, 2011.
[7] S. Eisenhardt, A. Küster, T. Schweizer, T. Kuhn, and W. Rosenstiel, "Spatial and temporal data path remapping for fault-tolerant coarse-grained reconfigurable architectures," Proc. DFT, pp.382–388, 2011.
[8] M. Parris, C.A. Sharma, and R.F. DeMara, "Progress in autonomous fault recovery of field programmable gate arrays," ACM Computing Surveys, vol.43, no.4, pp.31:1–31:30, 2010.
[9] A. Doumar and H. Ito, "Detecting, diagnosing, and tolerating faults in SRAM-based field programmable gate arrays: A survey," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.11, no.3, pp.386–405, 2003.
[10] J.C. Laprie, "Dependable computing and fault-tolerance: Concepts and terminology," Proc. FTCS, pp.2–11, 1985.
[11] H. Konoura, Y. Mitsuyama, M. Hashimoto, and T. Onoye, "Implications of reliability enhancement achieved by fault avoidance on dynamically reconfigurable architectures," Proc. FPL, pp.189–194, 2011.
[12] J.M. Soden, R.K. Treece, M.R. Taylor, and C.F. Hawkins, "CMOS IC stuck-open fault electrical effects and design considerations," Proc. ITC, pp.423–430, 1989.
[13] J.E. Vinson and J.J. Liou, "Electrostatic discharge in semiconductor devices: An overview," Proc. IEEE, vol.86, no.2, pp.399–420, 1998.
[14] B.C. Paul, K. Kang, H. Kufluoglu, M.A. Alam, and K. Roy, "Impact of NBTI on the temporal performance degradation of digital circuits," IEEE Electron Device Lett., vol.26, no.8, pp.560–562, 2005.
[15] K.L. Chen, S.A. Saller, I.A. Groves, and D.B. Scott, "Reliability effects on mos transistors due to hot-carrier injection," IEEE J. Solid-State Circuits, vol.20, no.1, pp.306–313, 1985.
[16] J. Noguchi, T. Saito, N. Ohashi, H. Ashihara, H. Maruyama, M. Kubo, H. Yamaguchi, D. Ryuzaki, K.I. Takeda, and K. Hinode, "Impact of low-k dielectrics and barrier metals on TDDB lifetime of Cu interconnects," Proc. IRPS, pp.355–359, 2001.
[17] R. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," IBM Journal of Research and Development, vol.6, no.2, pp.200–209, 1962.
[18] T. Inoue, T. Fujii, and H. Ichihara, "A self-test of dynamically reconfigurable processors with test frames," IEICE Trans. Inf.&Syst., vol.E91-D, no.3, pp.756–762, March 2008.

[19] T. Kameda, H. Konoura, D. Alnajjar, Y. Mitsuyama, M. Hashimoto, and T. Onoye, "A predictive delay fault avoidance scheme for coarse-grained reconfigurable architectures," Proc. FPL, pp.615–618, 2012.
[20] D. Alnajjar, Y. Ko, T. Imagawa, H. Konoura, M. Hiromoto, Y. Mitsuyama, M. Hashimoto, H. Ochi, and T. Onoye, "Coarse-grained dynamically reconfigurable architecture with flexible reliability," Proc. FPL, pp.186–192, 2009.
[21] A. Shibayama, H. Igura, M. Mizuno, and M. Yamashina, "An autonomous reconfigurable cell array for fault-tolerant LSIs," Proc. ISSCC, pp.230–231, 1997.
[22] F. Hatori, T. Sakurai, K. Nogami, K. Sawada, M. Takahashi, M. Ichida, M. Uchida, I. Yoshii, Y. Kawahara, T. Hibi, Y. Saeki, H. Muroga, A. Tanaka, and K. Kanzaki, "Introducing redundancy in field programmable gate arrays," Proc. CICC, pp.1–7, 1993.
[23] Z.E. Rakosi, M. Hiromoto, H. Ochi, and Y. Nakamura, "Hot-swapping architecture extension for mitigation of permanent functional unit faults," Proc. FPL, pp.578–581, 2009.
[24] F. Hanchek and S. Dutt, "Methodologies for tolerating cell and interconnect faults in FPGAs," IEEE Trans. Comput., vol.47, no.1, pp.15–33, 1998.
[25] L. Shang, M. Zhou, Y. Hu, and E. Yang, "A domain partition model approach to the online fault recovery of FPGA-based reconfigurable systems," IEICE Trans. Fundamentals, vol.E94-A, no.1, pp.290–299, Jan. 2011.
[26] J. Lach and W.H. Mangione-Smith, "Low overhead fault-tolerant FPGA systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.6, no.2, pp.212–221, 1998.
[27] N.J. Macias and L.J.K. Durbeck, "Adaptive methods for growing electronic circuits on an imperfect synthetic matrix," Biosystems, vol.73, no.3, pp.173–204, 2004.
[28] R.S. Oreifej, C.A. Sharma, and R.F. DeMara, "Expediting GA-based evolution using group testing techniques for reconfigurable hardware," Proc. ReConFig, pp.1–8, 2006.
[29] J. Emmert, C. Stroud, B. Skaggs, and M. Abramovici, "Dynamic fault tolerance in FPGAs via partial reconfiguration," Proc. FPCCM, pp.165–174, 2000.
[30] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," Proc. FPL, pp.213–222, 1997.
[31] L. McMurchie and C. Ebeling, "PathFinder: A negotiation-based performance-driven router for FPGAs," Proc. FPGA, pp.111–117, 1995.
[32] J. Lou, S. Thakur, S. Krishnamoorthy, and H.S. Sheng, "Estimating routing congestion using probabilistic analysis," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.21, no.1, pp.32–41, 2002.
[33] Y.H. Lee, N. Mielke, M. Agostinelli, S. Gupta, R. Lu, and W. McMahon, "Prediction of logic product failure due to thin-gate oxide breakdown," Proc. IRPS, pp.18–28, 2006.
[34] T. Imagawa, M. Hiromoto, H. Ochi, and T. Sato, "Reliability evaluation environment for exploring design space of coarse-grained reconfigurable architectures," IEICE Trans. Fundamentals, vol.E93-A, no.12, pp.2524–2532, Dec. 2010.

Hiroaki Konoura received B.E. and M.E. degrees in Information Systems Engineering from Osaka University, Japan, in 2009 and 2011, respectively. He is currently a doctoral student in the Department of Information Systems Engineering at Osaka University. His research interest is development of high reliable reconfigurable architecture. He is a student member of IEEE.

Takashi Imagawa received his B.E. degree in Electrical and Electronic Engineering, his master degree in Communications and Computer Engineering, from Kyoto University in 2008 and 2010. Presently, he is a doctor course student at Department of Communications and Computer Engineering, Kyoto University. He is a student member of IPSJ, and IEEE.

Yukio Mitsuyama received B.E., M.E., and Ph.D. degrees in Information Systems Engineering from Osaka University, Japan, in 1998, 2000, and 2010, respectively. He is currently an Associate Professor in School of Engineering, Kochi University of Technology. His research interests include reconfigurable architecture and its VLSI design. He is a member of IEEE and IPSJ.

Masanori Hashimoto received the B.E., M.E. and Ph.D. degrees in Communications and Computer Engineering from Kyoto University, Kyoto, Japan, in 1997, 1999, and 2001, respectively. Since 2004, he has been an Associate Professor in Department of Information Systems Engineering, Graduate School of Information Science and Technology, Osaka University. His research interest includes computer-aided-design for digital integrated circuits, and high-speed circuit design. Dr. Hashimoto served on the technical program committees for international conferences including DAC, ICCAD, ITC, ASP-DAC, DATE, ISPD, and Symposium on VLSI Circuits. He is a member of IEEE, ACM, and IPSJ.

Takao Onoye received the B.E. and M.E. degrees in Electronic Engineering, and Dr.Eng. degree in Information Systems Engineering all from Osaka University, Japan, in 1991, 1993, and 1997, respectively. He is currently a professor in the Department of Information Systems Engineering, Osaka University. His research interests include media-centric low-power architecture and its SoC implementation. He is a member of IEEE, IPSJ, and ITE-J.