IMPLICATIONS OF RELIABILITY ENHANCEMENT ACHIEVED BY FAULT AVOIDANCE ON DYNAMICALLY RECONFIGURABLE ARCHITECTURES

Hiroaki KONOURA[†], *Yukio MITSUYAMA*[‡], *Masanori HASHIMOTO*[†], *and Takao ONOYE*[†]

[†]Dept. Information Systems Engineering, Osaka University, Japan & JST CREST
[‡]School of Systems Engineering, Kochi University of Technology, Japan & JST CREST

ABSTRACT

Fault avoidance methods on dynamically reconfigurable devices have been proposed to extend device life-time, while their quantitative comparison has not been sufficiently presented. This paper shows results of quantitative life-time evaluation by simulating fault avoidance procedures of representative five methods under the same conditions of wear-out scenario, application and device architecture. Experimental results reveal 1) MTTF is highly correlated with the number of avoided faults, 2) there is the efficiency difference of spare usage in five fault avoidance methods, and 3) spares should be prevented from wear-out not to spoil life-time enhancement.

1. INTRODUCTION

Aggressive CMOS technology scaling is threatening device reliability, and all of early life, normal life and wear-out failures are increasing. For early life failures, burn-in and testing are thoughtfully studied to screen faulty chips [1]. Normal life failures are mostly caused by temporal effects, such as soft errors and power supply noise, and error mitigation and recovery are extensively deliberated [2]. On the other hand, wear-out failures lead to permanent errors in field and generally cannot be fixed without special mechanisms. Wear-out failures originate from aging effect such as NBTI (Negative Bias Temperature Instability) and TDDB (Time Dependent Die Breakdown), and result in degradation of device life-time and frequent device replacement.

To enhance life-time of devices by overcoming wear-out failures, fault tolerance techniques are required so that chips with a few faulty modules keep the functionality. Such fault tolerance techniques are widely researched at different levels. For instance, SRAM is often equipped to have redundant rows for replacement [3]. At architecture level, graceful performance degradation instead of sudden device failure is explored [4]. Besides, one of the most credible approaches for the fault tolerance is to exploit the redundancy and replace the faulty modules with spares, and it is highly compatible with reconfigurable devices, because unused basic elements (BEs) are available and can be used for the replacement.

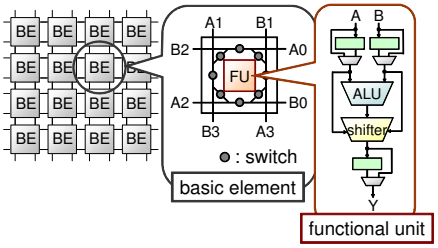


Fig. 1. Basic reconfigurable architecture.

For reconfigurable devices, several methods to replace faulty BEs have been proposed. On FPGA, references [5–7] proposed fault tolerance techniques using spare logic blocks. Similar approaches that utilize spares for replacement are investigated for coarse grained reconfigurable devices [8] and homogeneous many-core systems with network-on-chip [9]. Thus, there are several proposals that swap faulty BEs with spares for life-time enhancement. However, their efficiency has not been quantitatively compared.

In this study, five fault avoidance techniques based on the previous works are applied to a coarse grained reconfigurable devices, and the following implications are obtained.

1. Difference in mean time-to-failure (MTTF) is well characterized with the number of avoided faulty BEs.
2. Efficiency in spare usage, which is defined as the number of avoided faulty BEs divided by the number of available spares, is significantly different in each fault avoidance method. In our test case, more than six-fold difference of efficiency in spare usage was obtained.
3. Spares should be kept fresh by eliminating aging effects, otherwise the life-time enhancement thanks to spare replacement degrades by 34.0% in our test case.

2. ENVIRONMENT OF LIFE-TIME EVALUATION

This section introduces a basic reconfigurable architecture and an environment for life-time evaluation used in this study.

2.1. Basic reconfigurable architecture for swapping

Figure 1 illustrates the target architecture for this study, which is based on a coarse grained dynamically reconfigurable architecture introduced in [10]. In this architecture, identical BEs are aligned repeatedly in a two-dimensional array.

978-0-7695-4529-5/11 $26.00 © 2011 IEEE
DOI 10.1109/FPL.2011.108

IEEE computer society

Each BE is connected to adjacent BEs in four directions with two wires (A0-A3, B0-B3). Interconnections are configured with eight switches, and these switches are included in BEs. Each BE contains a functional unit (FU) having ALU and shifter, and the FU receives 1 or 2 inputs (A, B) and performs both arithmetic and logic operation for them. It is assumed that each BE has a self-testing mechanism and faulty BEs can be identified immediately before their life-times end. This is why we use a term of fault avoidance rather than fault tolerance in this paper.

2.2. Procedure of life-time evaluation

For each device, we evaluate time-to-failure (TTF) by simulating fault avoidance for a wear-out scenario. We randomly generate sets of wear-out scenarios in Monte Carlo manner and apply them to the device on the simulation. Then, a statistical distribution of TTF is obtained. This evaluation is repeated for each fault avoidance method.

Figure 2 shows the procedure of life-time evaluation used in this study. First, initial parameters, such as the number of BEs, the netlist of target circuit and a fault avoidance method for evaluation, are given. A detail of each fault avoidance method will be described in Sect. 3. We then produce an initial mapping tailored for the specified fault avoidance method. We next generate a wear-out scenario randomly according to a fault distribution and assign life-times of used BEs, which defines the temporal sequence of BE faults. The applied wear-out scenario in this work will be briefly explained in Sect. 5.1. After that, BE replacement with the specified fault avoidance method is simulated following the sequence above. When the fault avoidance cannot be completed, the life-time of the device ends, and both the number of avoided faulty BEs and time-to-failure (TTF) are recorded. This fault avoidance simulation for a wear-out scenario is one trial and is repeated to obtain the statistics.

3. FAULT AVOIDANCE METHODS FOR WEAR-OUT

Fault avoidance methods on reconfigurable devices are classified into two groups.

Hardware-oriented A faulty BE is automatically swapped with a spare using additional wires, switches, and controllers, along the replacement policy.

Reconfiguration-oriented Inherent reconfigurability is exploited for fault avoidance with partial mapping modification.

This section explains five fault avoidance methods for evaluation; column swap, row direction swap and neighbor swap in hardware-oriented group, and dynamic partial P&R and pre-compiled reconfiguration in reconfiguration-oriented group.

3.1. Column swap

Reference [11] presented a technique using the column (row) redundancy on the reconfigurable device. Referring to this technique, we implemented (a) column swap as shown in Fig. 3(a). First, an initial mapping is generated so that right columns are kept as spares. When a BE becomes faulty, each column on the right of the faulty BE is shifted to the right by one column.

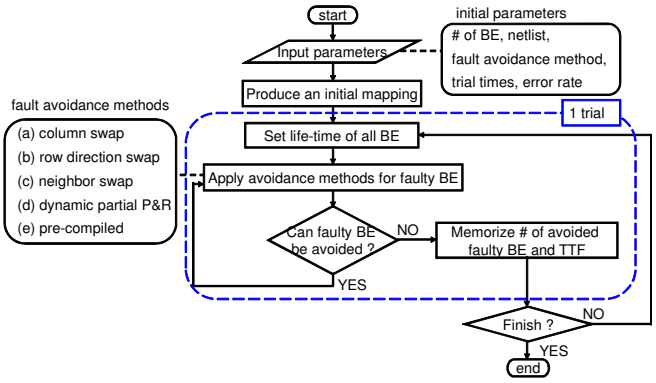


Fig. 2. Procedure of device life-time evaluation.

In this method, the possible number of swapping is often limited to a lower number because one spare column is entirely consumed by one faulty BE. As a hardware support, additional wires and switches are necessary to bypass faulty BE in horizontal direction.

3.2. Row direction swap

Reference [8] proposed a scheme that a faulty BE is eliminated using a spare on the same row. Referring to this technique, (b) row direction swap is implemented as Fig. 3(b). An initial mapping is generated so that right BEs are kept as spares, which is similar to (a) column swap. A faulty BE is eliminated by shifting the BEs on the right of the faulty BE to the right. Unlike (a) column swap, the elimination of the faulty BE is completed within a single row. Although the data flow is limited to one direction in [8], this limitation is relaxed and all directions are allowed in this study.

To realize this swapping and detour signal interconnections, a considerable number of additional wires and switches are necessary to bypass faulty BEs and compensate the vertical mismatch due to the BE shifting.

3.3. Neighbor swap

Reference [5] pointed out that the down-time for faulty BE elimination is an important metric, and the amount of BE shifting should be kept low. For such a purpose, (c) neighbor swap is implemented as illustrated in Fig. 3(c). In this method, spare BEs are uniformly distributed in the BE array. One of neighbor spare BEs around a faulty BE is selected for swapping.

In this method, once a spare BE is consumed, other surrounding BEs have less possibility to be replaced in the future. In the case that each spare is shared by several BEs, multiple faults in neighboring BEs cannot be avoided. Preliminarily distributed spares degrades the routability on the

device. As a hardware support, additional wires and switches are necessary for signal detouring.

3.4. Dynamic P&R

Reference [6] presented a self-repair technique with dynamic placement and routing (P&R) on FPGA. This (d) dynamic partial P&R is applied to the target reconfigurable architecture as illustrated in Fig. 3(d). An initial mapping is generated without dedicated spare BEs. When a BE becomes faulty, partial P&R are performed on the fly.

This method requires less wires and switches than methods (a) through (c), because ordinary reconfigurability is used for fault avoidance. Drawbacks are the downtime during the reconfiguration and necessity of a processor for dynamic partial P&R. Details of partial dynamic P&R will be explained in Sect. 4.

3.5. Pre-compiled reconfiguration

Reference [7] proposed a fault recovery technique by selecting a proper configuration from ones prepared beforehand and reloading it. This idea of (e) pre-compiled reconfiguration is implemented as shown in Fig. 3(e). With method (e), partial P&R are performed in advance for initial mapping assuming one faulty BE, and this pre-compiled configuration information for swapping is stored. When a faulty BE is detected, appropriate pre-compiled configuration information is applied for fault avoidance.

In this method, most of advantages and disadvantages are the same with those of method (d). A difference is the requirement of extra memory for storing spare configuration information in stead of a processor.

4. IMPLEMENTATION OF P&R

To evaluate device life-time with methods (c), (d) and (e), implementations of partial P&R and spare/fault-aware P&R are required. In this study, we extended popular placement and routing algorithms of VPR [12] and PathFinder [13] for enabling partial P&R and spare/fault-aware P&R.

4.1. Procedure for partial P&R

In performing partial P&R, we need to specify a region for partial P&R. A smaller region is desirable, because the downtime involved with the reconfiguration is shorter and the memory usage for storing is small. The procedure adopted in this evaluation is as follows.

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

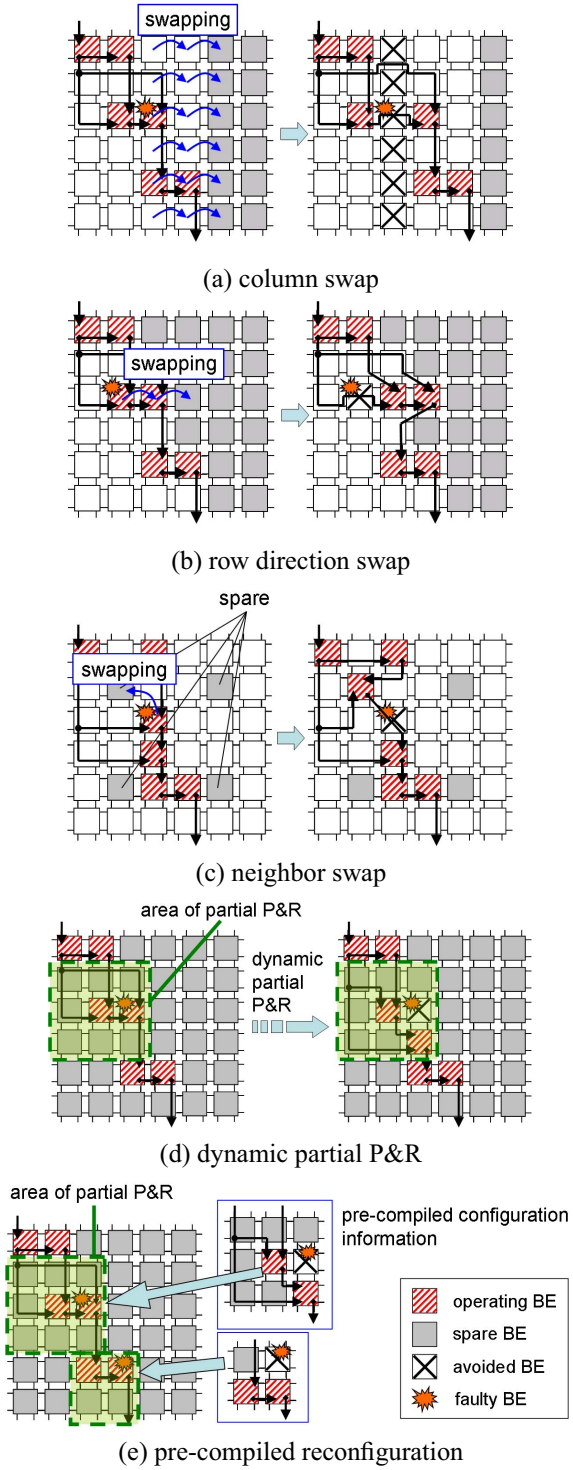


Fig. 3. Five fault avoidance methods.

Fig. 4 shows an example of region specification. Supposing BE(1, 2) becomes faulty, BE(2, 2) and BE(1, 3) are adjacent to and connected with the faulty BE, and thus should be included in 2x2 rectangle. In this case, BE(2, 2) is using only wires, then this 2x2 rectangle is the region for partial

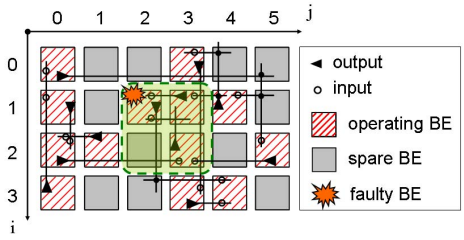


Fig. 4. Example of region specification for partial P&R.

P&R.

Once the region is specified, fault-aware P&R, which will be explained in the next subsection, is carried out to the region. When fault-aware P&R succeeds, partial P&R completes. On the other hand, when fault-aware P&R fails, the rectangle is expanded in the direction including more unused BEs and fault-aware P&R reruns. This expansion and fault-aware P&R repeat until partial P&R completes.

4.2. Spare/fault-aware P&R

In method (c), spare-aware P&R is necessary, since BEs allocated for spare cannot be used for mapping. Partial P&R in methods (d) and (e) requires 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 for 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 wire length, is expressed as [12]

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

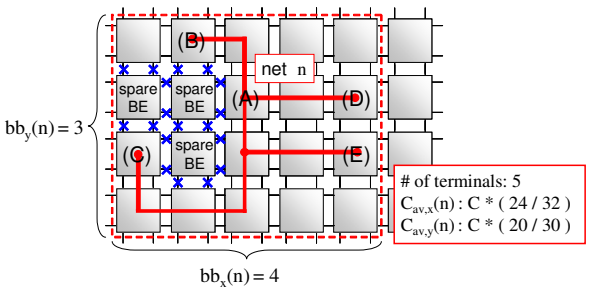
where N_{nets} is the total number of nets. $q(n)$ means a weight of net n, depending on the number of terminals. $bb_x(n)$ and $bb_y(n)$ are width and height of the bounding box for net n. $C_{av,x}(n)$ and $C_{av,y}(n)$ mean the average routing capacity of horizontal and vertical wires, respectively.

Fig. 5 illustrates an example to explain how the calculation of objective function changes with some spare BEs. When spare BEs are located as shown in Fig. 5, BE(C) cannot be connected with the shortest path. In this case, a detour is necessary and $bb_y(n)$ increases to 3. Moreover, spare BEs degrade the routing capacity. In this example, owing to the location of spare BEs, the number of usable horizontal and vertical wires are decreased from 32 to 24 and from 30 to 20, respectively. At that time, the horizontal and vertical average capacities decrease from C to $C*24/32$ and $C*20/30$, respectively.

As for spare/fault-aware routing, PathFinder algorithm[13] is also modified so that detouring can be considered.

5. EXPERIMENTAL RESULTS

This section describes the results of life-time evaluation with five fault avoidance methods.



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

Fig. 5. Example of spare aware computation of objective function.

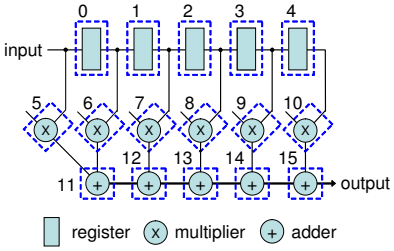


Fig. 6. 6-tap FIR filter.

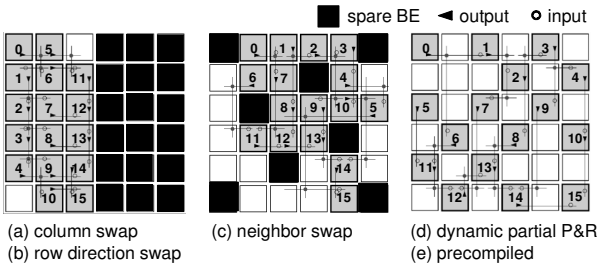


Fig. 7. Initial mappings of 6-tap FIR filter.

5.1. Experimental setup

Life-time of each BE is assumed to follow Weibull distribution which is widely used for evaluating device reliability (e.g. [14]). Error rate λ is set to 1.0×10^{-6} [/h], and shape parameter m is set to 2 assuming aging process. In the evaluation, spare BEs are assumed not to age. Additional wires, switches, and I/O interface for each fault-avoidance method are built-in. The number of TTF evaluation is 1,000.

For evaluation, 6-tap FIR filter (Fig. 6) and 2-point FFT were selected. Initial mappings of 6-tap FIR filter were prepared as shown in Fig. 7, where the available number of BEs are the same and forms a 6x6 array. The number of BEs is equal to 16, as shown in Fig. 6. For methods (a) and (b), right three columns are kept as spares. For method (c), eight spare BEs are located regularly. For methods (d) and (e), the mapping density was set to low for enabling partial P&R. Similarly, initial mappings of 2-point FFT were generated.

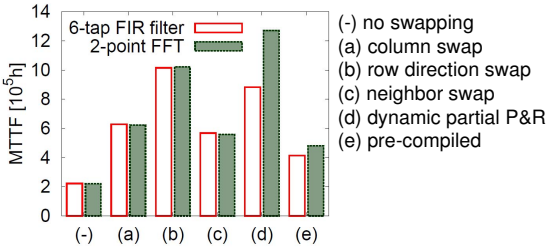


Fig. 8. MTTFs of 6-tap FIR filter and 2-point FFT with five fault avoidance methods.

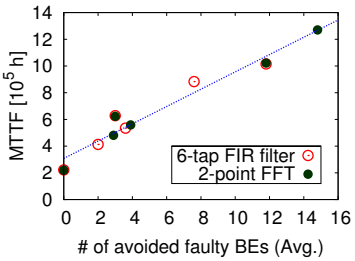


Fig. 9. The relationship between the average number of avoided faulty BEs and MTTF.

5.2. Life-time enhancement with fault avoidance

Fig. 8 shows MTTFs of 6-tap FIR filter and 2-point FFT with five fault avoidance methods. We can see that five fault avoidance methods achieve different MTTFs. To understand the MTTF difference, the relationship between the average number of avoided faulty BEs and MTTF is investigated (Fig. 9). In the figure, each dot corresponds to each fault avoidance method. This result indicates that MTTF is well correlated with the average number of avoided faulty BEs.

We next evaluate how efficiently each fault avoidance method uses spare/unused BEs. For methods (a) through (c), as shown in Fig. 10, additional mappings, in which the number and the location of spare/unused BEs were changed from Fig. 7, are also prepared and evaluated. Between the mappings in Fig. 7 and Fig. 10, the number of usable BEs is fixed to 6x6 in common. Table 1 shows the relationship of the number of spare/unused BEs and the average number of avoided faulty BEs. Here, we define efficiency in spare usage as (the number of avoided faulty BEs) / (the number of spare/unused BEs). Comparing the efficiency in spare usage among five fault avoidance methods, method (b) attained better efficiency for both mappings compared to other methods. On the other hand, in method (e), the efficiency in spare usage is the smallest, in other words, a lot of unused spare BEs still remain. Method (e) prepared, for each faulty BE, configuration information modified within the region of partial P&R. Once a BE becomes faulty, partial P&R is performed. Let us suppose another BE becomes faulty. In this case, if the region of partial P&R is overlapped with that of the previous one, partial P&R cannot be carried out, which means this fault cannot be avoided. This is why method (e) attained such low efficiency in spare usage in this evaluation. Compared method (e) to methods (b) through (d), the efficiency difference of spare usage becomes more than six-fold.

Table 1. The relationship of the number of spare/unused BEs and the average number of avoided faulty BEs with five fault avoidance methods (6-tap FIR filter).

	mapping	# of spare/ unused BEs	# of avoided faulty BEs	efficiency in spare usage
(a) column swap	Fig. 7	18	3.00	0.17
	Fig. 10	8	2.00	0.25
(b) row direction swap	Fig. 7	18	11.77	0.66
	Fig. 10	8	5.23	0.65
(c) neighbor swap	Fig. 7	8	4.09	0.51
	Fig. 10	4	2.32	0.58
(d) dynamic partial P&R	Fig. 7	20	7.63	0.38
(e) pre-compiled	Fig. 7	20	2.00	0.10

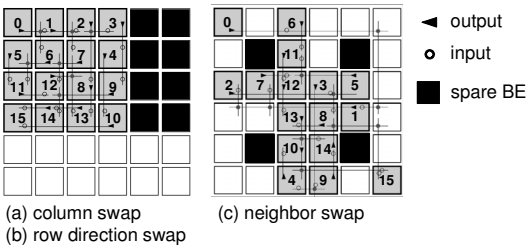


Fig. 10. Additional mappings of 6-tap FIR filter for methods (a) through (c).

We also evaluate the difference of MTTF depending on mapped applications, targeting 6-tap FIR filter and 2-point FFT. In Fig. 8, for methods (a) through (c), MTTF difference in terms of the target circuits is very small, because available spares are the same. On the other hand, for methods (d) and (e), MTTFs of 2-point FFT are higher. This is explained as follows. The total number of used wires in the initial mapping of 6-tap FIR filter is 1.4 ($= 42/30$) times larger than that of 2-point FFT. In this case, the freedom of rerouting is limited in 6-tap FIR filer and partial P&R less probably succeeds, which results in lower MTTF.

5.3. Consideration of spare aging

In the previous section, life-time evaluation was performed assuming spare BEs would not age. Here, we evaluate how this assumption affects the life-time enhancement.

Fig. 11 shows the relationship of 6-tap FIR filter between the number of spare/unused BEs and MTTF with and without spare aging. Here, spare aging means that life-time of each spare BE is also assumed to follow Weibull distribution as well as life-time of each used BE. Each figure has a ideal line of upper limits in the case that available spare/unused BEs can be necessarily used for fault avoidance, i.e. the number of avoided fault BEs is equal to the number of spare/unused BEs. Fig. 11 shows that MTTF significantly decreases due to spare aging degradation. For

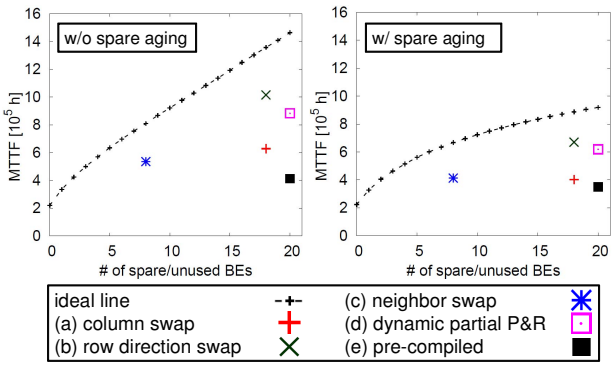


Fig. 11. Relationship of 6-tap FIR filter between number of spare/unused BEs and MTTF.

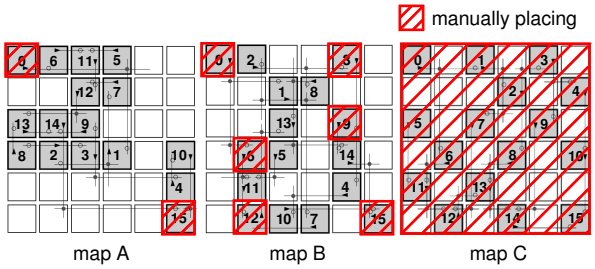


Fig. 12. Different initial mappings.

Table 2. MTTF with different initial mappings.

	MTTF [10^5 h]		
	map A	map B	map C
no fault avoidance	2.22		
(d) dynamic partial P&R	7.98	8.41	8.83
(e) pre-compiled	3.77	3.97	4.13

method (b), MTTF decreases to 66.0% (= 6.70/10.15). Similarly, MTTF of method (d) decreases to 70.1% (= 6.19/8.83). Thus, when spare BEs also degrades, a benefit of life-time enhancement with fault avoidance considerably spoils. Therefore, to make the best possible use of fault avoidance methods, mechanisms to keep spare BEs fresh, such as power-cutoff, are required.

5.4. Initial mapping for partial P&R

For methods (d) and (e), initial mappings which have high compatibility with partial P&R are expected to contribute more life-time extension. To confirm the dependence of MTTF on initial mappings, MTTF is evaluated for different mappings in Fig. 12. Map A was almost automatically generated with P&R algorithms of VPR and PathFinder. In map B, the location of some BEs was manually selected. Besides, in map C, BEs were placed by hand so that spares were uniformly placed, and only routing was automatically performed with the algorithm. A feature of these mappings is that the first partial P&R can be carried out even while a fault arises at any BEs.

Table 2 lists the MTTF with different mappings. For method (d), comparing map A with maps B and C, the differences of MTTF are 5.4% and 10.7%, respectively. Similarly, for method (e), they are 5.3% and 9.5%. Thus, the dependence of life-time on initial mappings was found. The investigation of the reason causing MTTF difference is one of our future works.

6. CONCLUSION

In this paper, quantitative life-time evaluations with five fault avoidance methods on dynamically reconfigurable devices were performed. The evaluation results revealed that MTTF was well characterized with the number of avoided faulty BEs, and each fault avoidance method has different efficiency in spare/unused usage for fault avoidance. It was demonstrated that to make the maximum use of fault avoidance methods, spare BEs should be prevented from aging degradation. More comprehensive evaluation considering additional hardware involved with fault avoidance will be performed as a future work.

7. ACKNOWLEDGEMENT

The authors would like to thank the project members of JST CREST of Kyoto University, Kyoto Institute of Technology, Nara Institute of Science and Technology, and ASTEM RI for their discussions.

8. 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, et al., "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. MICRO*, pp. 7 – 18, Dec. 2003.
[3] S. Mukhopadhyay, et al., "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS," *IEEE Trans. on CAD*, vol. 24, no. 12, pp. 1859 – 1880, 2005.
[4] O. Khan and S. Kundu, "A self-adaptive system architecture to address transistor aging," in *Proc. DATE*, pp. 81 – 86, Apr. 2009.
[5] A. Doumar, et al., "Defect and fault tolerance FPGAs by shifting the configuration data," in *Proc. DFT*, pp. 377 – 385, Nov. 1999.
[6] N. J. Macias, et al., "Adaptive methods for growing electronic circuits on an imperfect synthetic matrix," Biosystems, vol. 73, no. 3, pp. 173 – 204, 2004.
[7] L. Shang, et al., "A Domain partition model approach to the online fault recovery of FPGA-based reconfigurable systems," *IEICE Trans. on Fundamentals*, vol. 94, no. 1, pp. 290 – 299, 2011.
[8] Z. E. Rakosi, et al., "Hot-Swapping architecture extension for mitigation of permanent functional unit faults," in *Proc. FPL*, pp. 578 – 581, Sept. 2009.
[9] L. Zhang, et al., "On topology reconfiguration for defect-tolerant NoC-based homogeneous manycore systems," *IEEE Trans. on VLSI Systems*, vol. 17, no. 9, pp. 1173 – 1186, Sept. 2009.
[10] D. Alnajiar, et al., "Coarse-grained dynamically reconfigurable architecture with flexible reliability," in *Proc. FPL*, pp. 186 – 192, Sept. 2009.
[11] A. Shibayama, H. Igura, M. Mizuno, and M. Yamashina, "An autonomous reconfigurable cell array for fault-tolerant LSIs," in *Proc. ISSCC*, pp. 230 – 231, Feb. 1997.
[12] V. Betz, et al., "VPR: A new packing, placement and routing tool for FPGA research," in *Proc. FPL*, pp. 213 – 222, Sept. 1997.
[13] L. McMurchie, et al., "PathFinder: a negotiation-based performance-driven router for FPGAs," in *Proc. FPGA*, pp. 111 – 117, Feb. 1995.
[14] Y. H. Lee, et al., "Prediction of logic product failure due to thin-gate oxide breakdown," in *Proc. IRPS*, pp. 18 – 28, Mar. 2006.