# A Cost-Effective Selective TMR for Heterogeneous Coarse-Grained Reconfigurable Architectures based on DFG-Level Vulnerability Analysis

Takashi Imagawa, Hiroshi Tsutsui, Hiroyuki Ochi and Takashi Sato
Graduate School of Informatics, Kyoto University, Japan
Email: paper@easter.kuee.kyoto-u.ac.jp

***Abstract*—This paper proposes a method to determine a priority for applying selective triple modular redundancy (selective TMR) against single event upset (SEU) to achieve cost-effective reliable implementation of an application circuit to a coarse-grained reconfigurable architecture (CGRA). The priority is determined by an estimation of the vulnerability of each node in the data flow graph (DFG) of the application circuit. The estimation is based on a weighted sum of the features and parameters of each node in the DFG which characterize impact of the SEU in the node to the output data. This method does not require time-consuming placement-and-routing processes, as well as extensive fault simulations for various triplicating patterns, which allows us to identify the set of nodes to be triplicated for minimizing the vulnerability under given area constraint at the early stage of design flow. Therefore, the proposed method enables us efficient design space exploration of reliability-oriented CGRAs and their applications.**

## I. Introduction

Single-event upset (SEU) has been a major cause of problems in satellite [1] and avionics systems [2]. As CMOS process technologies enter into the range of a few tens of nanometers, consumer-oriented system designs also require serious consideration for SEU vulnerability. Coarse-grained reconfigurable architectures (CGRAs) are suitable for such applications [3] than their fine-grained counterpart (FPGAs) in terms of performance, energy efficiency, and SEU tolerance [4]. This is because CGRAs have a much smaller amount of configuration memory than FPGAs, which reduces the incidence of SEU.

Recently, a reliability-aware CGRA which adopts selective triple modular redundancy (TMR) trying to further enhance the SEU tolerance is proposed [5]. In this architecture, a part of the components in the given circuit is triplicated when a design constraint (e.g., area) does not allow to protect all the components by TMR. The components to be triplicated must be carefully determined because the impact observed at the output data greatly depends on the components protected. To evaluate the importance of each component, a quantitative measure to evaluate the SEU impact is required. In this paper, we use mean absolute error (MAE) as a generic criterion to measure the error observed at the outputs. The definition of MAE is given as

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|x_i - \tilde{x}_i|,$$

where $N$ is the number of words in the output data, $x_i$ and $\tilde{x}_i$ are $i$-th word of actual and error-free output, respectively. Note that errors are calcutated word by word in MAE; the number of digits $k$ in a word $w_i$ is typically 8 or 16 and each digit is weighted (e.g., the MSB is given $2^k$ times larger weight than the LSB). The nontrivial problem here is that, within given cost constraints, we need to determine the components (nodes in a data flow graph (DFG)) that have greater SEU impact than others and thus needed to be triplicated. Contrary to FPGAs, for which many studies on SEU reliability analysis [6]–[8] and protection prioritizations [9,10] exist, the prioritization analysis for CGRA has not been throughly investigated.

Exhaustive node ordering in a DFG for triplication requires prohibitively long simulation time. Instead, we propose a quick but effective method to determine a node priority for applying selective TMR that achieves cost-effective and reliable implementation of an application circuit on a heterogeneous CGRA composed of ALUs and LUTs. SEU vulnerability of a node is estimated by a linear function. The variables used in the function are circuit topology, the function of the nodes, etc. and general coefficients for the variables are trained in advance. No time-consuming process, such as placement-and-routing and fault simulations for various triplicating patters, are necessary, which allows us to quickly identify the set of nodes to be triplicated for minimizing the vulnerability at the early stage of design flow.

With the proposed method, efficient design-space exploration under given area constraint becomes possible for new applications implemented on heterogeneous CGRAs. The set of nodes that requires triplication are validated through fault-injection simulations which target the configuration memories to quantify the improvement of vulnerability by the partial triplication. Based on the the simulations, it is demonstrated that the proposed method enables us to achieve cost-effective selective TMR without the time-consuming processes.

The remainder of this paper is organized as follows. Section II introduces the target CGRA model considered in this paper. Section III describes the overview of our framework for designing reliability-aware LSI systems with CGRAs. Section IV describes the proposed method. Section V demonstrates the effectiveness of our proposed method. Finally, Section VI concludes this paper.

## II. Target CGRA model

This section describes the target CGRA model cosidered in this paper. This model is motivated by a reliability-aware CGRA proposed in [5]. Figure 1 provides an overview of the CGRA. It has a cluster array architecture designed to achieve various levels of reliability. It is a two-dimensional array of clusters, each of which consists of four cells that have an execution module (EM), configuration memories (CFG), voting circuits (VCs), and a configuration switch matrix (CFG-SM). Each EM includes an ALU for word operation, and registers in datapath and storing a constant values.

To realize flexible reliability, the CGRA also introduces a redundancy control unit (RDU) and a comparing-and-voting unit (CVU). The cluster has four operation modes, each with a different redundancy. In this paper TMR mode and single modular with multi-context (SMM) mode are considered. In TMR mode, three cells in a cluster with CVU form a triplicated module, and both configuration memory and datapath are triplicated. In SMM mode, four cells in a cluster operate independently, and there is no redundancy in either configuration memory or datapath. It is noteworthy that this CGRA supports selective TMR since the operation mode of every cluster can be selected independently, which offers an area-reliability trade-off, i.e., we can increase reliability by selecting more clusters to be in reliable modes at the expense of area usage.

To exchange the operand and processed data with external system, there are clusters for memory on top and bottom edges of the array. This cluster has the memory elements instead of the EMs. In this paper the target of the fault injection is only the clusters with the EM, because the memory element is expected to be protected with other techniques, such as ECC.

In this paper, we consider another kind of cluster. In the cluster, each EM have a LUT for bit operation. For efficient application implementation, some conditional judgments should be parallelized as well as data processing in CGRAs. However the conditional judgments are composed of bit level logical functions, so it is inefficient to implement these operations with the cluster including ALU. The LUT cluster enables us to implement a wider range of applications efficiently.

## III. Overview of our framework

Our framework explores the design space for reliability-aware CGRAs and applications. This framework has following features.

- For a target application design entry, register-transfer-level (RTL) description is allowed, and configuration bitstream of the partly triplicated circuit is generated as a result.
- Time-consuming fault simulation is not necessary to evaluate the vulnerability of each component in the application circuit.

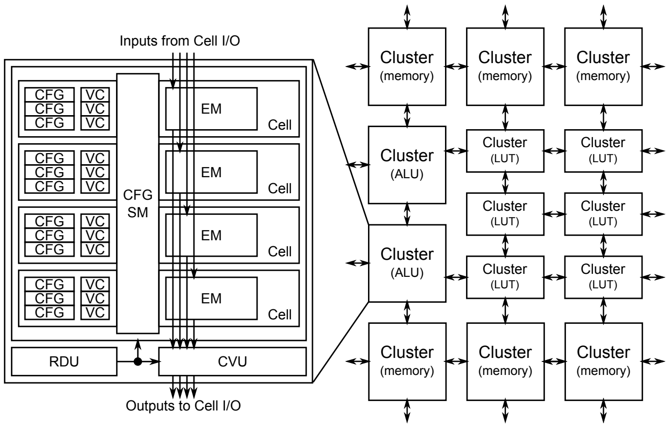

Fig. 1. The target reliability-aware CGRA.

The following is the design flow of our framework to generate a cost-effective reliability-aware implementation of given circuit for the CGRA (Figure 2).

1) A DFG is generated with parsing an RTL description of a taget application circuit. The DFG is composed of nodes and edges: the nodes stand for input/output terminals for the memory elements and the external system, operations, registers and constant values: the edges stand for their connection.
2) Technology mapping for the target CGRA described in Section II is applied. Each operation node is assigned to an ALU or a LUT cluster. The registers and constant values are also assigned to the clusters.
3) Selective TMR is applied to the DFG. The components which are to be triplicated are determined automatically based on the proposed method described in Section IV. A user specified number of components as a area restriction are selected to be triplicated in the order of vulnerability analyzed by the proposed method. Optionally, the components which are to be triplicated can be determined manually.
4) The placement and routing tool generates the configuration bitstream to implement the given application circuit on the target CGRA.
5) (Optional) The simulation scripts to verify the functionality of the target CGRA circuit and the target application circuit implemented on the CGRA are generated automatically with the scripts for the RTL descriptions.
6) (Optional) The fault injection scripts are also generated automatically to evaluate the vulnerability of the target circuit and the cost-effectiveness of the applied selective TMR.

## IV. Estimation of priority for selective TMR

In this section, we propose an evaluation function for estimating priority for TMR. The function estimates the vulnerability of each node in a DFG of application circuit. In this context, a “vulnerable node” means that SEUs in the component which implements the function of this node tend to cause a serious damage (i.e., large MAE) observable at the output data.

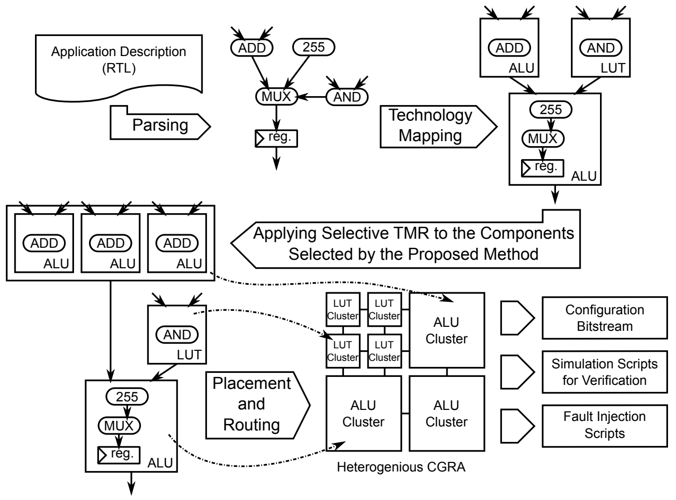

Fig. 2. Applying selective TMR for the DFG of given RTL description in our framework.

We use an evaluation function $f$ of the following form.

$$f = \sum_i w_i a_i \tag{1}$$

In the above expression, an $a_i$ is a term which represents the key feature or parameter of a node in DFG, such as the operation executed in the node and the depth from output node. A $w_i$ is a weight associated with $a_i$. It is beneficial to use a weight function of linear form as Eq. (1) since $w_i$ can be determined directly using generalized inverse matrix as explained in Section IV-B. Each $a_i$ is normalized to take a real value between 0.0 and 1.0. The restriction enables us to compare the $w_i$ each other directly in vulnerability aspect, that is, an $a_i$ corresponding to a large $w_i$ is an important feature or parameter for vulnerability.

The following subsections discuss two important points: what kind of features or parameters should be considered as $a_i$ and how to determine the appropriate weight $w_i$.

## A. Terms of the evaluation function

The nodes in a DFG can be characterized in various aspects. Some of them have a strong relationship to the vulnerability. The difference of these feature yields the difference of vulnerability among the nodes. From our preliminary experiments, we found some key features and parameters which have strong relationship to the vulnerability: operation, mapping target (ALU or LUT), utilization of register, distance from primary input/output, and output cone group. In the following, we explain the definitions of them and the motivation why we adopt them.

*1) Operation of the nodes:* We introduce terms which indicates the operation of the node. The number of these terms is equal to the number of possible operations of the processing element. For example, when the operation of node is OR, the corresponding term takes 1 and the other terms take 0.

Generally, the logical operations, such as OR operation, often mask the influence of the erroneous inputs. Therefore, whether a node itself implements a logical operation is important features. For example, assuming that the signals have 50% probability to take logical 0, then the erroneous value arrived at an input of a 2-input AND gate will be masked with 50% probability. The logical masking effect is non-negligible especially when many these operations are chained in series.

*2) Mapping target:* We introduce two terms for each node. One of two terms is 1, if the operation of the node is mapped to ALU and 0 otherwise, and vice versa.

In the target heterogeneous CGRA model, most of the arithmetic and word-level operations are implemented on ALU and the others are on LUT. The amount of the configuration memory for implementing its operation are different between ALU and LUT. Therefore, it has a considerable impact whether an operation is implemented on ALU or LUT.

*3) Utilization of register:* We introduce one term for each node, in order to indicate that the register in the processing elements for the node are enabled or not.

When the incorrect result of an operation is stored in a register in the processing element, it is expected that the error has a persistent impact for several clock cycles. So the utilization of register has an impact on whether the error in the node is temporal or persistent.

*4) Distance from primary input/output:* We introduce two terms for each node, in order to indicate the distance from primary input and primary output, respectively. The value is normalized by dividing with the maximum distance in the DFG, so that the value falls into the range from 0.0 to 1.0. If the value of the distance term is $d$, we also introduce a term that takes $1/d$. This is intended to emphasize the nodes that are very close to the edge of the DFG, while $d$ indicates the distance from the edge of the DFG linearly.

The nodes near the primary output nodes are considered as vulnerable because the faults in such nodes are rarely masked logically. On the other hand, the errors generated in the nodes near the primary inputs are sometimes masked logically when the erroneous data pass through logical operations. However, such errors can affect a wide range of the DFG. Therefore, the relationship between the vulnerability and the distance is so complex that the relationship is worth while evaluating including its importance.

*5) Output cone group:* We define four categories for outputs, memory write data, memory address, memory write enable, and external done signal, which are explained later. We introduce four terms for each node, in order to indicate whether the transitive fanouts of the node belong to the above categories.

Besides the previous popular features, this paper focuses on a novel feature of nodes in DFG taking into account that the target framework is based on CGRAs. As described in Section III, in the target framework, the operand and processed data is exchanged with the external system through the memory element. Therefore the primary output of the target application can be categorized into the following groups: write data, read/write address and write enable signal for memory elements, and a done signal for the external system. Therefore, each node can be characterized with above categories of its transitive fanouts. As Fig. 3 illustrates, each node in a DFG

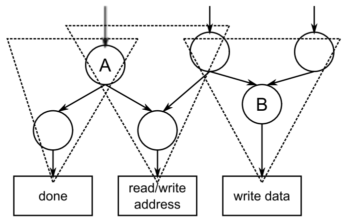

Fig. 3. Each node in a DFG can be categorized based on the output cones.

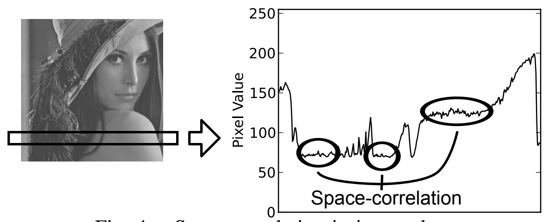

Fig. 4. Space-correlation in image data.

belongs to the grouping cones whose tops are the primary output (or the memory port) nodes. For example, the node A belongs to the done signal cone and the read/write address cone, while the node B belongs only to the write data cone. This feature is not limited for the target CGRA described in Section II, because other general CGRAs also have memory elements as interfaces with external systems.

It is expected that the nodes belonging to the done signal cone are relatively robust, since the signal is generated by some conditional judgments which are mainly composed of logical functions, and hence high probability of logical masking is expected. If input and output signal has space- or time-correlation which can be often seen in audio, image and video data (Fig. 4), the nodes belonging to the read/write address cone are expected to relatively robust. When the fault generates a little error in the read address, the differences of the operand values are also a little. The case of write address is similar.

In order to quantitatively show that nodes in the cone of the address are robust as compared with the ones in other cones, we present a simple example. Assume that a pixel value (pixel0) in a memory (named mem) at an address (adrs0) is copied to another memory at the same address.

- When pixel0 is corrupted by a fault, the value of write data changes randomly to pixel1.
- When adrs0 is corrupted by a fault, the value of read/write address changes randomly to adrs1 and the write data is mem[adrs1].

The errors in the above two cases are evaluated as |pixel0 − pixel1| and |pixel0 − mem[adr1]|, respectively. In order to calculate the averages of above errors, 10,000 Monte Carlo trials for one hundred pictures are executed assuming that word length of both address and data of the memory are 8-bit. The averages of errors with faults in data and address are 96.94 and 39.52, respectively. This means that the faults in the address cone have 60% less impact than the data cone.

## B. Coefficients of the evaluation function

To use the proposed evaluation function (Eq. (1)), we need to determine the coefficients $w_i$. This subsection describes the process to determine an appropriate coefficients with sample applications and fault injection simulation.

1) Finding the ideal priority of applying selective TMR for some sample circuits:
At first, DFGs for the given RTL descriptions of some sample application circuits are generated, and technology mapping for the target CGRA (described in Section II) is applied using our framework. For each configuration memory bit of the obtained netlist, we inject an SEU and run simulation to evaluate MAE observed at the output data raised by the SEU. This simulation gives us MAEs induced by errors on each processing element. Based on the information, the ideal priority for applying selective TMR to the processing elements is extracted. For example, assuming as follows, the ideal priority is $[n_1, \ldots, n_m]$
   - There are $m$ processing elements $(n_1, \ldots, n_m)$ in a netlist.
   - The triplicating efficiency of each element is $(e_1, \ldots, e_m)$, which is the ratio of the MAE improvement to the area overhead when the element is triplicated, and their magnitude correlation is $e_1 > \cdots > e_m$.

2) Calculating the appropriate $\boldsymbol{w}$ to obtain the ideal priority for each sample:
Based on Eq. (1), the vulnerability of each node in the DFG of a target application is represented as follow.

$$v_1 = a_{11} \cdot w_1 + a_{12} \cdot w_2 + \cdots + a_{1n} \cdot w_n$$
$$\cdots$$
$$v_m = a_{m1} \cdot w_1 + a_{m2} \cdot w_2 + \cdots + a_{mn} \cdot w_n$$

These equations can be expressed as follows using vector and matrix.

$$\boldsymbol{v} = \boldsymbol{A}\boldsymbol{w}$$

When the matrix $\boldsymbol{A}$ is invertible, $\boldsymbol{w}$ is obetained as

$$\boldsymbol{w} = \boldsymbol{A}^{-1}\boldsymbol{v}.$$

However, it is obvious that $\boldsymbol{A}$ is not invertible in many cases since $m$ is not necessarily equal to $n$. In such cases, $\boldsymbol{w}$ is obetained as

$$\boldsymbol{w} = \boldsymbol{A}^{+}\boldsymbol{v}, \tag{2}$$

where $\boldsymbol{A}^{+}$ is the generalized inverse matrix (or Moore-Penrose inverse matrix) given below:

$$\boldsymbol{A}^{+} = \begin{cases} (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T & (m > n) \\ \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1} & (m \leq n) \end{cases}$$

Based on the previous fault injection simulations, each elements of $\boldsymbol{v}$ should satisfy the magnitude correlation, $v_1 > \cdots > v_m$. Here, we use the following assignments for $\boldsymbol{v}$.

$$\boldsymbol{v} = (m/m,\ (m-1)/m,\ \ldots,\ 1/m) \tag{3}$$

With Eq. (2) and Eq. (3), the appropriate $\boldsymbol{w}$ for each sample circuit is obtained.

3) Calculating a generic $\boldsymbol{w}$.

With $\boldsymbol{w}$s of sample circuits, a generic $\boldsymbol{w}$ is calculated, which enables us to get reasonable trade-off curves between area overhead and reliability for general applications. In this paper the generic $\boldsymbol{w}$ is calculated by taking the average of $\boldsymbol{w}$s of the sample circuits.

## V. Evaluation

This section demonstrates the effectiveness of the proposed method. The contents of the evaluations are summarized as follows.

Subsection V-A derives set of coefficients for evaluation function using sample applications. The sample applications are a 1024-point FFT and three image filters, that is, a color invert filter, a horizontal-differential filter, and an edge detection filter. We also confirm that the set of coefficients determined with all the four applications enables us an efficient selective TMR for these applications themselves.

Subsection V-B evaluates the dependency on input vectors. Generally, impact of errors at output data depends on input vectors. So it should be evaluated whether the determined priority is effective for other input vectors.

Subsection V-C evaluates whether the determined priority is effective for the applications which are not used for determining the coefficients.

Subsection V-D evaluates whether the estimated priority is effective for the post place-and-route circuit on CGRAs. Since the proposed method barely takes into account the impact of the routing element in the CGRA, this subsection evaluates the trade-off curves between vulnerability and TMR cost with fault injection simulation for partly triplicated, post place-and-route circuits implemented on the CGRA.

In each subsection, the estimated trade-off curves between circuit area and MAE are evaluated with the ratio of the area enclosed by the estimated and best case curves ($A$, the hatched portion of Fig. 5) to the one enclosed by the best case and the worst case curves ($B$, the hatched or shaded portion of Fig. 5). The best case curve is obtained with the ideal priority extracted by the fault injection simulations described in IV-B, while the worst case curve is obtained with the reverse order of the ideal priority. If the ratio is zero, it means that the estimated curve is exactly same as the best case curve; on the other hand, if the ratio is one, it means that the estimated curve is exactly same as the worst case curve. In Fig. 5, circuit area and MAE are normalized to take a value between 1.0 and 3.0, and between 1.0 and 0.0, respectively. The corner with area=1.0 and MAE=1.0 corresponds to the circuit without triplicated node, while the corner with area=3.0 and MAE=0.0 corresponds to the fully-triplicated circuit.

To calculate the triplicating efficiency and obtain the trade-off curves, the circuit area ratio of an ALU cluster to a LUT cluster is required. From a synthesis result using a 65nm process library, the area ratio is $2.88 : 1$.

### A. Deriving coefficients

The set of coefficients is determined by the method described in IV-B to improve the trade-off curves quality of all

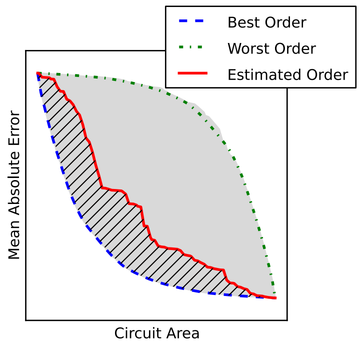

Fig. 5. Quality of trade-off curve is evaluated based on graph areas.

the four applications. It takes less than ten seconds to calculate the coefficients. Figure 6 shows the estimated trade-off curves with the obtained coefficients and the curves with the random priority orders as comparison for all the applications. The ratios are 0.112 for the color invert filter, 0.198 for the horizontal-differential filter, 0.294 for the edge detection filter and 0.311 for FFT.

Although the random priority orders sometimes makes the trade-off curves convex, the obtained set of coefficients makes all the estimated trade-off curves concave and achieves the minimum error ratios, so it can be said that the set is appropriate. The magnitude correlation and the value of the determined coefficients for output cones is (write data $>$ write enable signal $>$ done signal $>$ read/write address). The coefficients for the logical operations tend to be smaller (i.e., less sensitive) than for the arithmetic operations. Similarly, the coefficient for the operation implemented on the LUT smaller than the ALU because most LUTs execute the logical operations. These results correspond to the expectations described in Subsection IV-A.

There are some points on which the vulnerability drops steeply in the latter of the trade-off curves. This result means that the priority of some vulnerable nodes is estimated low. It suggests that further improvement of the evaluation function is desired.

### B. Input vector dependency

The quality of trade-off curves with different input vectors which are not used in determining the coefficients are evaluated. The ratios which indicate the quality of the trade-off curves are 0.061 for the color invert filter, 0.201 for the horizontal-differential filter, 0.335 for the edge detection filter and 0.396 for FFT.

The results show that the obtained coefficients are robust for the image filter applications. On the other hand, the quality of FFT's trade-off curve is a little worse, although the shape of the curve is still concave. Especially, when the input vector is artificial (e.g., sinusoidal waves or chopping waves), the degradation of quality is notable. These results suggest a necessity for some application-specific customization of the evaluation function.

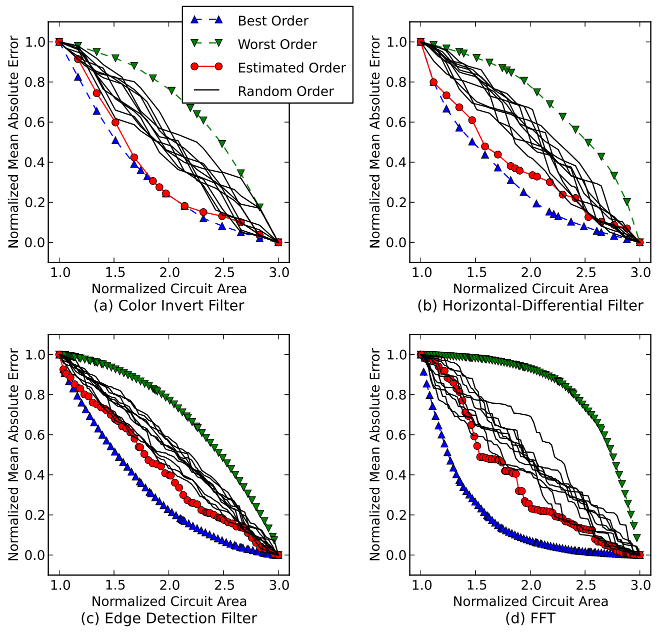

Fig. 6. Estimated trade-off curves.

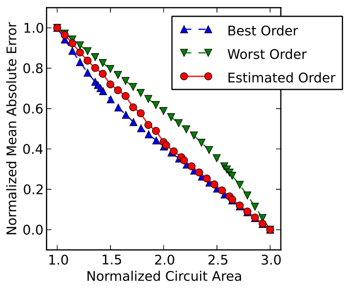

Fig. 7. Estimated trade-off curve of FIR filter.

## *C. Application dependency*

Figure 7 shows the trade-off curves for an FIR filter which is not used in determining the coefficients. The ratio which indicates the quality of the trade-off curves is 0.270. The figure shows that the obtained coefficients are something robust for the newcomer application.

## *D. Vulnerability of post P&R circuits*

The trade-off curves for the target applications implemented on the CGRA after placement and routing are evaluated. The ratios which indicate the quality of the trade-off curves are 0.132 for the color invert filter, 0.207 for the horizontal-differential filter, 0.304 for the edge detection filter and 0.358 for FFT.

The quality of the trade-off curves entirely becomes a little worse, because the number of vulnerable routing elements increases, when the triplicated nodes disturb the routing with the shortest path. Therefore the evaluation function may be improved by taking into account of the routing efficiency when each triplicating pattern is applied.

However, it is notable that the degradation of the trade-off curves is very small compared with the result of Subsection V-A, although the coefficients used in the proposed evaluation function are not derived using any information obtained after placement and routing. Thus, this result suggests that finding an optimal selective TMR at the early stage of the design flow is possible.

## VI. Conclusion

This paper proposed a method to determine a priority for applying selective TMR to achieve cost-effective reliable implementation of an application circuit to a heterogeneous CGRA. Once the coefficients used in the evaluation function are determined, this method does not require time-consuming processes such as placement-and-routing and extensive fault simulations. This feature allows us to identify the set of nodes to be protected from SEU at the early stage of design flow. This paper also demonstrated the effectiveness and robustness of the proposed method with some sample applications.

A challenging future work is to develop a versatile set of sample applications enough to explore more appropriate terms and coefficients of the evaluation function.

## Acknowledgment

This work is part of a JST CREST project and is partly supported by Grain-in-Aid for JSPS Fellows 22·6265, and by VDEC, the University of Tokyo in collaboration with Synopsys, Inc. and Mentor Graphics, Inc.

## References

[1] H. C. Koons, J. E. Mazur, R. S. Selesnick, J. B. Blake, J. F. Fennell, J. L. Roeder, and P. C. Anderson, "The impact of the space environment on space systems," in *Proc. Spacecraft Charging Conference*, Nov. 1998, pp. 7–11.

[2] D. C. Matthews and M. J. Dion, "NSEU impact on commercial avionics," in *Proc. International Reliability Physics Symposium (IRPS)*, Apr. 2009, pp. 181–193.

[3] T. Imagawa, M. Hiromoto, H. Ochi, and T. Sato, "Reliability evaluation environment for exploring design space of coarse-grained reconfigurable architectures," *IEICE Transactions of Fundamentals on Electronics, Communications and Computer Sciences*, vol. E93-A, no. 12, pp. 2524–2532, Dec. 2010.

[4] Zain-ul-Abdin and B. Svensson, "Evolution in architectures and programming methodologies of coarse-grained reconfigurable computing," *Microprocessors and Microsystems*, vol. 33, no. 3, pp. 161–178, May 2009.

[5] 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," in *Proc. International Conference on Field Programmable Logic and Applications (FPL)*, Aug. 2009, pp. 186–192.

[6] K. Nakahara, S. Kouyama, T. Izumi, H. Ochi, , and Y. Nakamura, "Fault tolerant reconfigurable device based on autonomous-repair cells," in *Proc. the 16th International Conference on Field Programmable Logic and Applications*, Aug. 2006, pp. 461–466.

[7] M. Reorda, L. Sterpone, and M. Violante, "Multiple errors produced by single upsets in FPGA configuration memory: a possible solution," in *Proc. the European Test Symposium*, May 2005, pp. 136–141.

[8] S. Srinivasan, A. Gayasen, N. Vijaykrishnan, M. Kandemir, Y. Xie, and M. Irwin, "Improving soft-error tolerance of FPGA configuration bits," in *Proc. the IEEE/ACM International Conference on Computer Aided Design*, Nov. 2004, pp. 107–110.

[9] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin, "Improving FPGA design robustness with partial TMR," in *Proc. International Reliability Physics Symposium (IRPS)*, Mar. 2006, pp. 226–232.

[10] L. Sterpone, M. Aguirre, J. Tombs, and H. Guzman-Miranda, "On the design of tunable fault tolerant circuits on sram-based fpgas for safety critical applications," in *Proc. Design, Automation and Test in Europe*, Mar. 2008, pp. 336–341.