PAPER *Special Section on VLSI Design and CAD Algorithms*

Reliability Evaluation Environment for Exploring Design Space of Coarse-Grained Reconfigurable Architectures

Takashi IMAGAWA[†a)], ***Nonmember***, **Masayuki HIROMOTO**[†], **Hiroyuki OCHI**[†], ***and*** **Takashi SATO**[†], ***Members***

SUMMARY This paper proposes a reliability evaluation environment for coarse-grained reconfigurable architectures. This environment is designed so that it can be easily extended to different target architectures and applications by automating the generation of the simulation inputs such as HDL codes for fault injection and configuration information. This automation enables us to explore a huge design space in order to efficiently analyze area/reliability trade-offs and find the best solution. This paper also shows demonstrative examples of the design space exploration of coarse-grained reconfigurable architectures using the proposed environment. Through the demonstrations, we discuss relationship between coarse-grained architectures and reliability, which has not yet been addressed in existing literatures and show the feasibility of the proposed environment.

key words: *soft error, TMR, reliability, methodology*

1. Introduction

As CMOS process technology enters in the range of a few tens of nanometers, NRE cost for manufacturing LSIs is increasing rapidly. The process scaling also has made various phenomena which disturb the normal operation of LSI systems notable. For example, process variation such as varying doping concentration can cause delay fault or stuck-at fault, SEU (Single-Event Upset) and SET (Single-Event Transient) caused by cosmic particles and alpha particles can cause soft-errors. Soft-errors have been paid attention especially because they often cause troubles or raise problems in satellite systems [1] and avionics systems [2]. Recently, reconfigurable devices, especially FPGAs have been regarded as one of the solutions for both problems. It is widely known that the reconfigurability can reduce NRE cost. It is expected that reconfigurable devices can also extend the life time of LSI systems using their reconfigurability [3]. The systems with faulty units can continue normal operation with configurations which avoid using the faulty units.

Due to process scaling, soft-error resilience is expected to be also a major concern even for consumer-oriented systems in the near future. It means that not only performance, circuit area and power consumption but also reliability of the systems have to be evaluated. The evaluation has to be performed with the fault model which is best suited for operational environment. There is a significant difference in acceptable cost margins for reliability between aero-space systems and consumer-oriented systems. These make LSI-system design even more complex. Therefore, the next-generation methodology for designing LSI systems which meet various requirements including reliability has to be established.

It is expected that reconfigurable devices enable us to solve these problems. The reconfigurability enables us not only to implement the various functions onto one chip, but also to apply the appropriate techniques for reliability enhancements depending on the priority of reliability or the environment factors, e.g. soft-error incidence ratio. Therefore, reconfigurability facilitates answering the wide variety of demands such as reliability with cost effectiveness.

Recently, especially in multimedia and stream processing of consumer-oriented embedded systems, coarse-grained reconfigurable architectures have become gradually noticeable. Such architectures are advantageous to the fine-grained counterpart (FPGAs) in terms of performance and energy efficiency. More noteworthy is the fact that they have a much smaller amount of configuration memory than FPGAs, and this can reduce the incidence of soft-errors. Hence, the design space of coarse-grained reconfigurable architectures is worth exploring from reliability aspect. However, only a limited number of studies have been made for reliability-oriented coarse-grained reconfigurable architectures [3]–[5]. Therefore, this paper focuses on exploration of coarse-grained reconfigurable architectures.

The goal of our project is to establish such an LSI system design methodology. To that end, we should be able to (1) make quantitative comparisons of reliability of several architectures and implementations, and (2) explore the reconfigurable architecture from reliability aspect to find optimal usages of the techniques for reliability enhancement under specific constraints. Furthermore, it is strongly desired to automate (1) and (2).

In this paper, we propose the reliability evaluation environment for designing coarse-grained reconfigurable architectures. In this proposed environment, application mapping and generation of simulation inputs such as HDL codes for fault injections are automated. This enables efficient design space exploration and helps evaluation of various techniques such as architecture-tailored application mapping, selective redundancy injection, etc. in order to enhance reliability. This paper also shows demonstrative examples of the design space exploration of coarse-grained reconfigurable architectures using the proposed environment, including (1)

Manuscript received March 17, 2010.
Manuscript revised June 21, 2010.

[†]The authors are with the Dept. of Communications and Computer Eng., Graduate School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: imagawa@easter.kuee.kyoto-u.ac.jp

DOI: 10.1587/transfun.E93.A.2524

locating memory elements to be protected from SEUs, (2) comparing ALU architectures in terms of SEU tolerance, (3) area-reliability trade-off using selective TMR, and (4) comparison of routing architectures for several applications.

The remainder of this paper is organized as follows. Section 2 discusses the existing coarse-grained reconfigurable architectures and the evaluation of their reliability. Section 3 describes the proposed environment. Section 4 shows the demonstrations. Section 5 concludes this paper and describes future works.

2. Existing Coarse-Grained Reconfigurable Architectures and Their Evaluation

A lot of coarse-grained reconfigurable architectures have been proposed [6]. [7] discusses about the design-space exploration of energy-delay-area efficient datapath. However, only a limited number of studies have been made for reliability-oriented coarse-grained reconfigurable architectures [3]–[5].

In [4], flip-flops and combinational circuits are multiplexed in different ways. It achieves the same reliability as TMR with lower area-overhead. However, there is no area-reliability trade-off in this architecture, because the multiplexings are always activated in the whole circuit.

In [3], hot-swapping concept is proposed. In this architecture, the faulty units are swapped for spares with run-time fault diagnoses using partial reconfiguration. This architecture achieves area- and performance-reliability trade-offs by varying the ratio of active units and spares. The reliability is evaluated using a statistical approach, but not with actual applications. The design-space exploration in basic components such as routing structure and instruction set from reliability aspect is not carried out.

In [5], cluster array architecture for flexible reliability is proposed. The cluster has four modes of operation, which enable several degrees of reliability. Area-reliability trade-off is achieved by varying the degrees of reliability for each cluster in the array independently, i.e. selecting more reliable modes of operation at the expense of area usage. The reliability is evaluated in statistical approach and with actual applications. However, similar to [3], the design-space exploration in basic components such as inter- and intra-routing structure from reliability aspect is not carried out.

Most of the existing research on reliability in LSI systems, including coarse-grained reconfigurable architectures, assume single fault only. However, [8] shows that multiple fault such as MCU (multi-cell upset) occurrence will be more frequent in future process. Therefore, the multiple fault model should be employed. The model needs the vast number of fault pattern and immense simulation time but they can be cut down efficiently by using physical distance restriction between faulty elements.

Therefore, efficient platform for exploration and evaluation is necessary. Huge design space, several reliability enhancement methods and multiple fault models underscore the need for a platform that can take all of these into account, explore solutions and evaluate them in a systematic manner.

In order to perform reliability-aware design space exploration of coarse-grained reconfigurable architecture, we need a design environment which includes (1) a retargetable compiler that covers a certain range of target coarse-grained reconfigurable architectures to be explored, and (2) a tool which evaluates reliability of architectures quantitatively. However, there is no design environment that has (1) and (2). Little attempts have been made for (1), because each conventional design environment is dedicated for its own reconfigurable architecture and is not usable as a retargetable compiler. VPR [9] is a unique retargetable place-and-route solution for FPGAs, but it does not support coarse-grained reconfigurable architectures. To solve (1), we need an architecture model of coarse-grained reconfigurable architecture and architecture-independent place-and-route algorithms. (2) is also a challenging issue, because we need a universal criterion to compare reliability of different coarse-grained reconfigurable architectures.

3. Proposed Reliability Evaluation Environment

Based on the discussions so far, this section presents the proposed reliability evaluation environment for exploring design space of coarse-grained reconfigurable architectures.

First of all, we have to decide the abstraction level of analysis; higher-level analysis, in general, is efficient in terms of time and memory, while lower-level analysis gives us more accurate results. Low-level analysis based on gate-level simulation with accurate delay model is necessary if we need to handle SETs, but full-chip gate-level simulation is time consuming or even impossible. On the other hand, high-level approach such as statistical-analysis-based method can be considered. However, it is hard to formulate the reliability of a reconfigurable fabric on which specific applications are implemented. We decided to employ "mid-level" analysis based on RTL simulation to estimate the effect of SEUs for specific architectures and applications.

One of important features of our reliability evaluation environment is automated generation of simulation inputs (e.g., configuration data for given architecture and application, and fault injection script for given architecture and assumptions) to make efficient design space exploration possible.

3.1 Target Architecture Model

In this paper an ALU-array-based architecture model is established which covers a certain range of coarse-grained reconfigurable architectures. The target coarse-grained reconfigurable architecture model in the proposed environment is shown in Fig. 1. The basic unit which is called as cluster are arranged in a two dimensional array. The proposed environment targets both homogeneous and heterogeneous arrays.

A cluster is composed of some cells for execution of operations. A cell is composed of flip-flops for configuration information, an ALU which has two inputs and basic

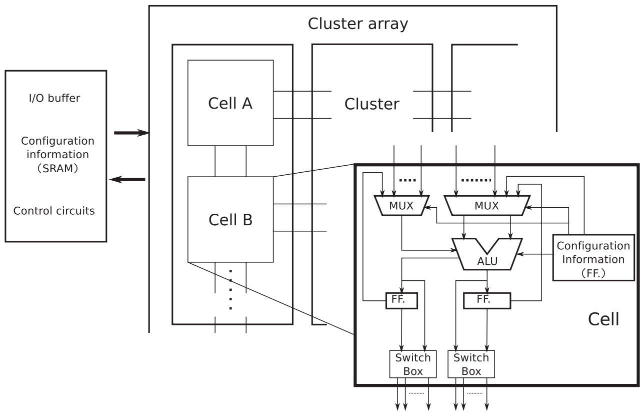


Fig. 1 The target coarse-grained reconfigurable architecture model.

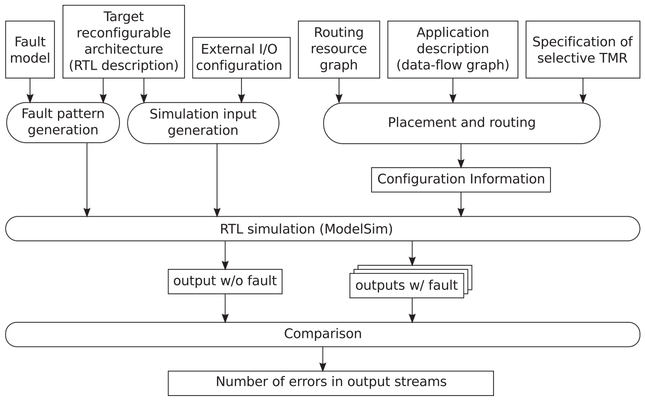
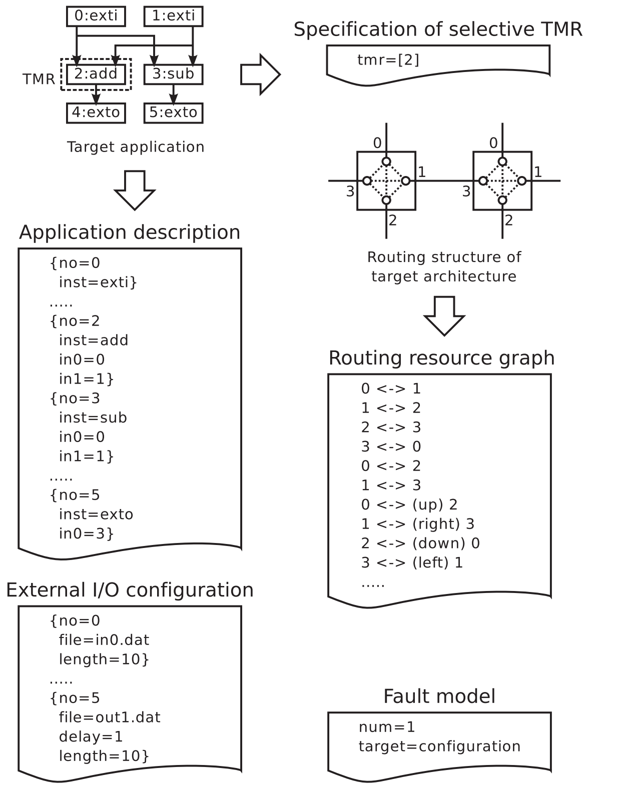

Fault model
Target reconfigurable architecture (RTL description)
External I/O configuration
Routing resource graph
Application description (data-flow graph)
Specification of selective TMR
Fault pattern generation
Simulation input generation
Placement and routing
Configuration Information
RTL simulation (ModelSim)
output w/o fault
outputs w/ fault
Comparison
Number of errors in output streams

Fig. 2 The evaluation flow of the proposed environment.



```
tmr=[2]
```

```
{no=0
 inst=exti}
.....
{no=2
 inst=add
 in0=0
 in1=1}
{no=3
 inst=sub
 in0=0
 in1=1}
.....
{no=5
 inst=exto
 in0=3}
```

```
0 <-> 1
1 <-> 2
2 <-> 3
3 <-> 0
0 <-> 2
1 <-> 3
0 <-> (up) 2
1 <-> (right) 3
2 <-> (down) 0
3 <-> (left) 1
.....
```

```
{no=0
 file=in0.dat
 length=10}
.....
{no=5
 file=out1.dat
 delay=1
 length=10}
```

```
num=1
target=configuration
```

Fig. 3 The examples of the inputs for the proposed environment.

instructions or three inputs and specific instructions such as voting instruction and product-sum instruction, multiplexers for selecting inputs of the ALU and switch-boxes for routing outputs of the ALU. The cluster and cell can have additional circuits for reliability enhancement such as an encoder and a decoder for ECC and a voter for TMR. There are SRAMs for configuration information, a control circuit for distributing the information to each cluster and cell and I/O buffers outside the cluster array.

3.2 The Fault Model in the Proposed Environment

The fault model in the proposed environment is summarized as follows.

1. Only SEUs in flip-flops are taken into account.
2. The number of bits which flip at one time is one or two. If a two bit flip occur, the bits exist in the same cluster.

1. is based on [10] and [11] which state that the probability of a SEU in flip-flops occurring are much larger than SEU in SRAMs or SET in 32 nm and below. 2. is based on [8] which states that the probability of one or two bit SEUs occurring is larger than 95% of all SEUs in 65 nm and below.

3.3 Reliability Evaluation Flow

Figure 2 shows the reliability evaluation flow of the proposed environment. The detailed processes are summarized as follows.

3.3.1 Input Data

The input data set of the proposed environment which should be prepared by the users is summarized as follows.

(1) Fault model, which specifies the assumption on multiple faults and the target flip-flops (configuration FFs, data-path FFs or both).
(2) RTL description of target architecture which is used for RTL simulation, circuit area evaluation, and fault injection.
(3) External I/O configuration.
(4) Routing resource graph.
(5) Application Description in the form of technology-mapped data-flow graph (DFG) for the target architecture.
(6) Specification of selective TMR.

Figure 3 shows the examples of (1), (3), (4), (5) and (6). (3) specifies the data sets of input, the file name of output stream, the length of input and output stream, and the output delay. (4) represents the available routing resources and their connectivities of the target architecture, and similar information has been generally used by retargetable automated place-and-route tools for FPGAs [12]. (6) defines which nodes in target DFG will be applied selective TMR.

3.3.2 Preparations for Simulation

The retargetable compiler we developed for simple 2-dimensional ALU array described in Sect. 3.1 executes the automated placement and routing and configuration information is generated using the routing resource graph, the target DFG and the specification about selective TMR. The placement algorithm is a pairwise exchange method using a simulated annealing algorithm [13]. The routing algorithm consists of two-stages: global routing and detailed routing. The global routing is executed based on the negotiation-based algorithm and the detailed routing is executed based on the rip-up and re-route algorithm [9], [14]. We used versatile algorithms for placement and routing such as simulated annealing, negotiation-based global routing, and so on. We avoided ad-hoc algorithms which causes the tool architecture-specific.

Parsing RTL description extracts the list of registers in target RTLs. Fault patterns are generated automatically based on the list and the fault model described in Sect. 3.2. Simulation input such as RTL description for fault injections are generated automatically based on the simulation parameters.

These processes are fully automated. This automation enables us to explore a huge design space in order to efficiently analyze area/reliability trade-offs and find the best solution.

3.3.3 Simulation and Evaluation

The RTL simulation is executed using the RTL description of the architecture, the generated fault injection script, and the configuration information with ModelSim. The simulation outputs show the circuits' behavior with and without the fault injections. By comparing these outputs, errors appeared in the output streams can be evaluated. By applying different ALU architectures, different routing architectures, different applications, and different partial TMR directives, we can obtain valuable information for design-space exploration, as demonstrated in the next section. One of the most useful and universal reliability criteria for mutual comparison of reconfigurable architectures is "sensitive bit" which has been used for FPGAs [15]. The proposed environment supports evaluation of sensitive bit, which is demonstrated in Sect. 4.3 and Sect. 4.4.

4. Demonstrations of Design Space Exploration with the Proposed Environment

This section demonstrates design space exploration of coarse-grained reconfigurable architectures from various aspects with the proposed environment. Through the demonstrations, we discuss relationship between coarse-grained architectures and reliability and show the feasibility of the proposed environment.

The contents of the demonstrations are summarized as follows. Sect. 4.1 demonstrates that the proposed environment can locate memory elements to be protected from SEUs. Sect. 4.2 compares cluster and cell architectures in terms of SEU tolerance and demonstrates the feasibility of selective TMR with built-in voter. Sect. 4.3 demonstrates that the proposed environment with automated P&R feature successfully analyzed that there is a meaningful trade-off between area and reliability using selective TMR for a recent coarse-grained reconfigurable architecture with built-in voter [5]. Sect. 4.4 compares routing architectures for several applications

4.1 Locating Memory Elements to be Protected

[15] states that the importance of a memory element in terms of impact induced by SEU depends on its role in the entire circuit. The quantitative order of the impact means the priority of each unit for protection. In this demonstration, the proposed environment can figure out the priority quantitatively.

4.1.1 Target Architecture

The target architecture is a simple 2-dimensional ALU array, a special case of the cluster array model where each cluster has only one cell. Each cell has an ALU which has two inputs and ten arithmetic and logical instructions and flip-flops for accumulation.

4.1.2 Target Application

The target application is ten-tap FIR filters. The number of error bits which appear in output streams depends on the location of the fault. Therefore, those cells which generate a lot of errors when an SEU occurs to their own flip-flops should be protected preferentially.

4.1.3 Experiment and Result

It is natural to assume that the rate of SEU incidence is constant spatially and temporally. Based on this assumption, the behaviors of the target circuit with one-bit SEUs are simulated exhaustively, where the SEU can occur at any flip-flops and at any clock cycle from the beginning of data input to the ending of data output. The pseudo-code of this simulation is described as follows;

```
err=0;
for (c=c0;c<=c1;c++) {
 for (f=0;f<n;f++) {
  err += NumberOfErrorAppeared
                InOutputStream(c,f);
 }
}
return(err);
```

where c0 and c1 are the clock cycle when data input starts

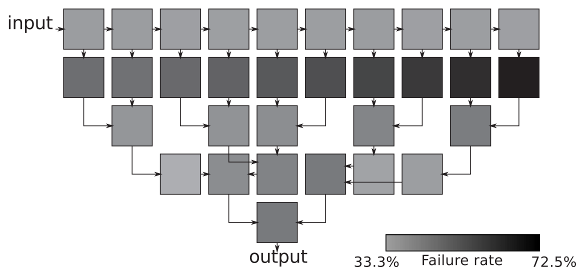


Fig. 4 The number of errors in the primary output when SEUs occur in the flip-flops for configuration.

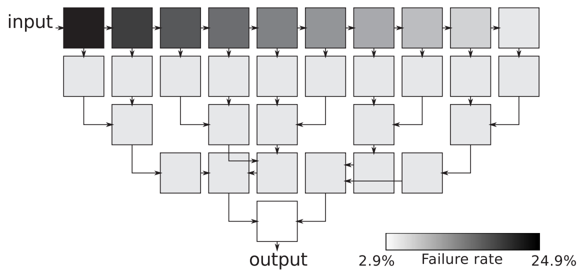


Fig. 5 The number of errors in the primary output when SEUs occur in the flip-flops for accumulation.

and data output ends, respectively, and n is the number of flip-flops. NumberOfErrorAppearedInOutputStream(c,f) invokes a simulation run with an SEU that occurs in a flip-flop f at clock cycle c and returns the number of erroneous words appeared in the output stream.

Figure 4 illustrates the result of the experiment. Each square represents a cell and each arrow represents their connectivity. The color of each cell represents how many errors in the primary output of the FIR filter are generated when SEUs occur in its flip-flops for configuration. Black cells generates many errors in output streams when SEUs occur in it so they should be protected. The cell located on the right edge of the second row generates the largest number of errors, that is, 72.5% of output stream are faulty. The cell located on the left edge of the first row generates the smallest number of errors, that is, 33.3% of output stream are faulty.

Figure 5 illustrates the result of the experiment with SEUs in flip-flops for accumulation. The cell located on the left edge of the first row generates the largest number of errors, that is, 24.9% of output stream are faulty. The cell located on the fifth row generates the smallest number of errors, that is, 2.9% of output stream are faulty.

4.1.4 Discussion

Figure 4 shows that the cells on second row generate many errors. The reason comes from that these cells operate with an immediate operand which requires many bits for configuration. The error counts of cells on the second row are different because their immediate operand values are different. This suggests that vulnerability of cells in an FIR filter strongly depends on their constant coefficients. Note that this new insight is gained by experiments using the proposed environment.

The cells on first row generate fewer errors than others. This is because only a limited number of configuration bits are essential to implement functions on these cells.

Figure 5 shows that the locations of the cells impact their reliability, that is, SEUs in the cells which are far from the primary output of the FIR filter generate many errors. It should be noted that Figs. 4 and 5 show different trends. This suggests that different approaches are effective for flip-flops for configuration and datapath to enhance reliability.

These results show that the proposed environment can figure out the priority of each unit for protection quantitatively. Furthermore, these results suggest that selective TMR in coarse-grained reconfigurable architectures is applicable.

4.2 Comparing Cluster and Cell Architectures in Terms of SEU Tolerance

There are various techniques for reliability enhancement such as TMR and Hamming codes. Their effectiveness for reliability depend on the assumed fault model. Under single fault model, both TMR and Hamming code can achieve the same reliability. In contrast under multiple fault model, achievable reliability would be different.

The cluster and cell architectures for implementing the techniques affect the reliability. For example, various implementations of voter for TMR can be considered (e.g., a voter implemented by multiple cells using native instructions such as AND/OR, a voter implemented by single ALU which has a dedicated instruction for voting operation, and a built-in voter within a cluster) and different implementations can result in different reliability.

This demonstration compares cluster and cell architectures in terms of SEU tolerance under multiple fault model with the proposed environment.

4.2.1 Target Architectures

We will use four target architectures; one is the same architecture introduced in Sect. 4.1.1 (denoted by "Simple ALU Array"), and the others are its extensions summarized as follows.

a Triplicated flip-flops
All flip-flops in each cluster and cell are triplicated. All reconfigurable components receive the configuration information through built-in voters. Note that built-in voters have no vulnerability to SEUs because the functionality of built-in (or hard wired) circuit is not affected by SEU.

b Voting operation in the ALU's instruction set
Each cluster is composed of two types of cells. One cell has a three-input ALU which has voting operation in instruction set. Five patterns of binary code ("0000,"

Table 1 Failure rate of each architecture and implementation.

Architecture and implementation	Amount of configuration information [bit]	Failure rate [%]	
		1 bit SEU	2 bit SEU
(1) Simple ALU array	376	94.41	99.68
(2) (1) with full-TMR	2232	3.58	23.64
(3) Triplicated flip-flops	1128	0.00	0.17
(4) Voting operation	1520	0.00	18.04
(5) Hamming code	1209	0.00	1.72

"0001," "0010," "0100," "1000") are assigned to the voting operation and "0000" are used normally. This make the correct voting operation under one-bit SEUs possible. The other cell is the same as those in simple ALU array.

c Hamming code
All configuration information and data for accumulation are protected by Hamming code encoder and decoder.

4.2.2 Target Application

The target Application is MixColumns which is a part of AES encryption/decryption process. Unlike FIR filters, even only one-bit error in the primary output of MixColumns is fatal in encryption and decryption.

4.2.3 Experiment and Result

This demonstration assumes SEUs in configuration flip-flops only.

The behavior of the MixColumns with one-bit and two-bits SEUs. The definition of the "failure rate" which is used for evaluation of reliability in this experiment is given as follows;

- one-bit SEU:

$$\frac{\left(\begin{array}{c}\text{The number of configuration bit}\\ \text{which affect the output stream with SEU}\end{array}\right)}{\text{(The total number of configuration bit)}}$$

- two-bit SEU:

$$\frac{\left(\begin{array}{c}\text{The number of combinations of configuration bit}\\ \text{which affect the output stream with SEU}\end{array}\right)}{\text{(The total number of combinations of configuration bit)}}$$

Unlike Sect. 4.1, the timing of SEU injection is fixed at the beginning of the data input, because the experiment in this section targets only the flip-flops for configuration which are not overwritten during the runtime by the application.

Table 1 shows the failure rate of each circuit. (1) and (2) are the circuit implemented using simple ALU architectures. (1) is implemented without any redundancy, while (2) implements full-TMR using combination of simple ALUs. (3) is implemented using **a**. (4) is implemented using **b** with full-TMR. (5) is implemented using **c**.

4.2.4 Discussion

(2), (3) and (4) are different in the amounts of configuration information for executing voting operation. (2) requires 5 clusters for implementing a voting operation, so many configuration information is required. In (3) All built-in voters are always activated, so no additional configuration information is required. (4) requires 4-bit configuration for executing voting operation. These difference is the reason why failure rates differ especially under multiple fault model. The result of (5) shows that Hamming code is less efficient than TMR with built-in voter. These result shows that TMR with built-in voter is the most efficient and promising implementation.

The results of this demonstration are derived only by user's effort for implementing the target architecture, owing to the automation of the application mapping and reliability evaluation. This is one of the benefits of the proposed environment.

4.3 Area-Dependability Trade-Off Using Selective TMR

Sections 4.1 and 4.2 show the feasibility of selective TMR and built-in voter. There is a coarse-grained reconfigurable architecture which is designed based on these concept [5]. This section shows a demonstration that the architecture and the proposed environment, especially automated placement and routing, enables us to draw an area-dependability trade-off curve. Furthermore, the experiment demonstrates that the same overhead in area provides different reliability depending on which units are configured as TMR.

4.3.1 Target Architecture

The target architecture is a slight modification of [5] to fit the Sect. 3.1 architecture model. This demonstration deals with two modes of operation: 1) TMR mode, which triplicates the operational circuit, the data-path and the configuration information by using three cells in a cluster; and 2) SMM mode, which enables four cells in a cluster to operate independently and maximizes the area efficiency in exchange for the reliability.

4.3.2 Target Application

The target application is Viterbi decoders (constraint length is 3). The process of Viterbi decoding is divided into three parts: branch metric unit, path metric unit and path memory unit.

4.3.3 Experiment and Result

This demonstration assumes only one-bit SEU in configuration flip-flops. For comparison, we evaluated five patterns

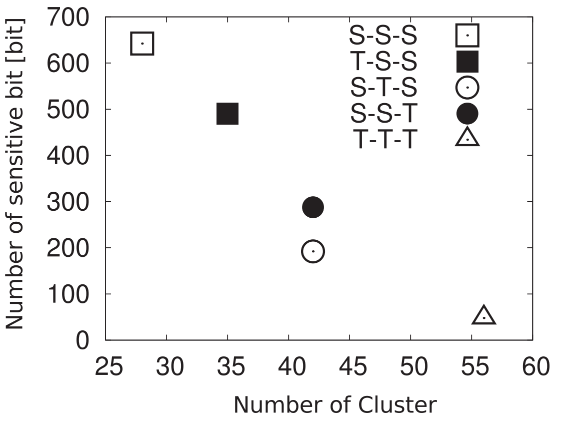


Fig. 6 Area-reliability trade-off of the viterbi decoders.

of Viterbi decoder implementations: fully triplicated implementation using TMR mode only (denoted by T-T-T), fully single-modular implementation using SMM mode only (denoted by S-S-S), and partially triplicated implementations in which only one of three parts is fully triplicated (denoted by T-S-S, S-T-S, and S-S-T). The reliability of each circuit is evaluated in a similar way to [15], that is, by counting the number of "sensitive bits."

Figure 6 describes the result of the evaluation. X-axis (circuit area) is the number of cluster. Y-axis (number of sensitive bit) is averaged over 50 input data.

4.3.4 Discussion

There is a considerable trade-off between area and reliability in Fig. 6. Comparing "S-T-S" with "S-S-T," the area of the both circuits are the same but the number of sensitive bits are different, that is, the same overhead in area provides different reliability depending on which units to be configured as TMR. This result is mainly attributed to the difference between the functions of the units. In Viterbi decoder, only path metric unit has feedback paths, hence, it seems that the path metric unit should be protected with higher priority than path memory unit, if we use partial TMR. To achieve high reliability efficiently, it is important to identify which part of the target circuit should be protected, and this is where the proposed environment is effectively used.

4.4 Comparison of Routing Architectures for Several Applications

This experiment demonstrates a design space exploration of routing architecture.

There is complex relationship among routing architecture, circuit area, and reliability. Architectures with less-flexible routing sometimes achieves area-efficient implementation for small circuits. However, large and complex applications demand large amount of routing resources, and thus architectures with poor routing resources results lower usage of cells. This makes target system not only area-inefficient but also less reliable. In contrast, rich routing architectures can implement large and complex applications with area-efficiently. For small applications, however, many routing resources are left unused which can disturb the normal operations of the neighboring resources when SEU occur. Selective TMR makes the relationship more complex.

This demonstration evaluates area-reliability trade-off using selective TMR for three types of routing architectures based on [5] and three applications.

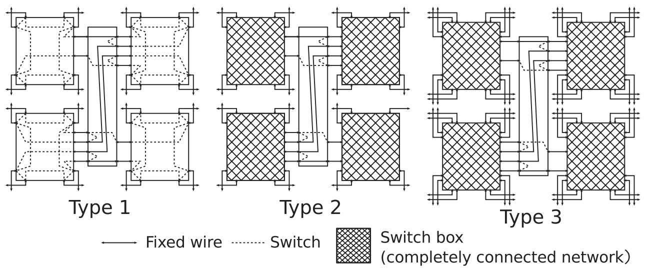


Fig. 7 The target routing architectures.

4.4.1 Target Architecture

Figure 7 illustrates the target routing architectures. Each figure means intra-cluster connection, that is, four squares represent switch-boxes for intra-cell connection, a central rectangle represents switch-box for inter-cell connection, each arrow represents fixed wires.

Type 1 is equivalent to [5] and the intra-cell connectivity is restricted compared to the others. Type 2 is implemented by changing the intra-cell connectivity of type 1 into completely connected network. Type 3 is implemented by adding inter-cluster fixed wires.

The area of type 1 is small but the flexibility is low, so type 1 is suited for simple and small applications. In contrast, types 2 and 3 have high flexibility and large area, so they are suited for complex and large applications.

4.4.2 Target Applications

The first target is a four-tap FIR filter as a simple and small application. The result of Sect. 4.1 shows the units in the same row have similar trends in reliability. Therefore, eight patterns of FIR filters are implemented with varying the redundancy (triplicated or not) row-by-row. The second target is MixColumns used in Sect. 4.2 as a medium-scale application. This application can be divided into four parallel circuits. Therefore, sixteen patterns of MixColumns are implemented with varying the redundancy of every circuit. The third target is Viterbi decoder used in Sect. 4.3. Five patterns which are same as Sect. 4.3 are implemented.

4.4.3 Experiment and Result

This demonstration assumes one-bit SEU in configuration flip-flops only. The reliability of each circuit is evaluated by counting the number of "sensitive bits."

Figures 8, 9, and 10 show the result of the FIR filters,

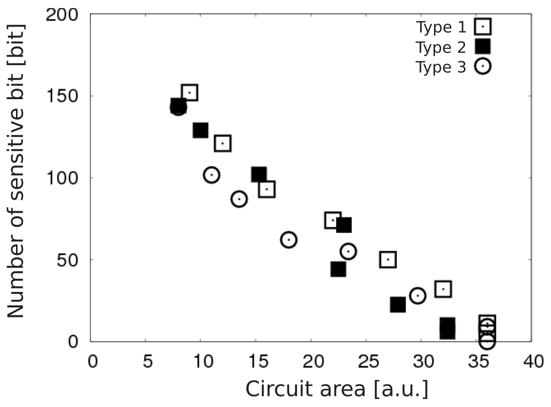


Fig. 8 The result of FIR filters.

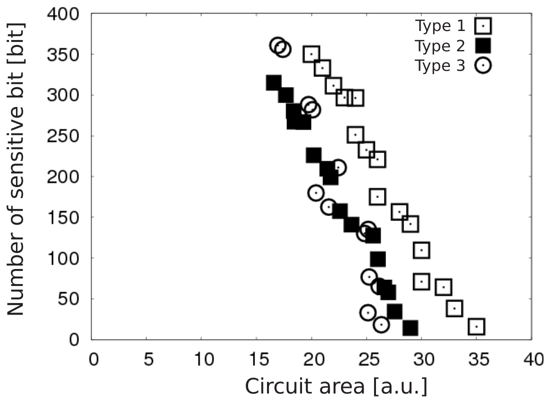


Fig. 9 The result of MixColumns.

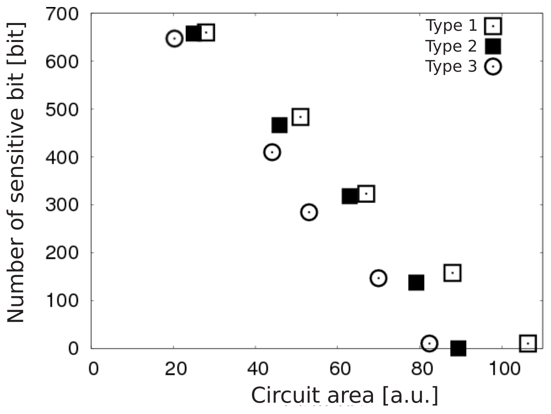


Fig. 10 The result of Viterbi decoders.

the Mix-Columns, and the Viterbi decoders, respectively. The circuit areas are evaluated using $0.13\,\mu$m CMOS standard cell library, and Synposys Design Compiler. The x-axis of these figures are normalized by the gate count of a cluster of type 1.

4.4.4 Discussion

The result of Viterbi decoders shows the highest flexible routing architecture achieves the best circuit area and reliability. In this demonstration, therefore, highly-flexible routing architectures are suited for large and complex applications such as Viterbi decoder. In contrast, the result of FIR filters shows that if the constraint for circuit area is severe, type 3 is appropriate, and if large area overhead is permitted type 2 is appropriate. The same is equally true for reliability constraints. The result of MixColumns shows that type 2 is appropriate for the severe constraints for area and type 3 is appropriate when large area overhead is permitted.

These results suggest that the proposed environment is useful for exploration of the appropriate routing architecture suited for various area and reliability constraints and target applications.

5. Conclusions

In this paper, we propose the reliability evaluation environment for designing coarse-grained reconfigurable architectures. This environment enables us to evaluate the reliability of a coarse-grained reconfigurable architecture, and helps to explore design space of architectures and find an optimal usage of the reliability hardening techniques under specific constraints (e.g., from Fig. 8, we can select Type 2 or Type 3 architecture to minimize number of sensitive bit depending on the allowable circuit area.)

This environment is designed so that it can be easily extended to different target architectures and applications by automating the generation of the simulation inputs such as HDL codes for fault injection and configuration information. Therefore, this environment also can be used to evaluate the feasibility of a certain technique for increasing reliability of a certain reconfigurable architecture.

This paper also shows demonstrative examples of the design space exploration of coarse-grained reconfigurable architectures from various aspect using the proposed environment. These demonstrations generate various discussions such as the relationship between routing architectures and applications, which is not addressed in existing literature.

In this paper, only area-reliability trade-off is evaluated, but power-reliability trade-off is also very important. We expect that the comparison of relative power consumption between various types of architecture is possible using existing tools. In addition, we are interested in the comparison of power consumption overhead between spatial and time redundancy.

As future work, by extending these tools, we will explore other kinds of coarse-grained reconfigurable architectures which are not ALU-array-based in order to find an appropriate one for achieving reliability efficiently. To make the design space exploration efficient, It is desired that the generations of the input files for the proposed environment are automated. For example, it will be better if application definition (DFG) is generated automatically from higher-level description like C, and RTL description of target architecture and routing resource graph are generated automatically from an architecture description language which specifies the instruction set of ALUs and routing architectures. Also we intend to compare the advantages and disadvantages over FPGA implementations from the standpoints of reliability, performance, power consumption and so on. Moreover, various techniques such as time and information redundancy and dynamic reconfiguration will be targets of our environment.

Acknowledgement

This work is part of a JST CREST project and is partly supported 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, and J.F. Fennell, “The impact of the space environment on space systems,” Proc. 6th Spacecraft Charging Conference, pp.7–11, Nov. 1998.

[2] D.C. Matthews and M.J. Dion, “Nseu impact on commercial avionics,” Proc. 2009 IEEE International Reliability Physics Symposium (IRPS), pp.181–193, April 2009.

[3] Z.E. Rakosi, M. Hiromoto, H. Ochi, and Y. Nakamura, “Hot-swapping architecture extension for mitigation of permanent functional unit faults,” Proc. 19th International Conference on Field Programmable Logic and Applications (FPL), pp.186–192, Aug. 2009.

[4] S. Baloch, T. Arslan, and A. Stoica, “Radiation hardened coarse-grain reconfigurable architecture for space applications,” Proc. 21th IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp.1–8, March 2007.

[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,” Proc. 19th International Conference on Field Programmable Logic and Applications (FPL), pp.186–192, Aug. 2009.

[6] Z. 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.

[7] S. Purohit, M. Lanuzza, and M. Margala, “Design-space exploration of energy-delay-area efficient coarse-grain reconfigurable datapath,” Proc. 2009 22nd International Conference on VLSI Design, pp.45–50, Jan. 2009.

[8] A. Dixit, R. Heald, and A. Wood, “Trends from ten years of soft error experimentation,” Proc. 2009 IEEE Workshop on Silicon Errors in Logic - System Effects (SELSE), pp.III–1, March 2009.

[9] V. Betz and J. Rose, “VPR: A new packing, placement and routing tool for FPGA research,” Lect. Notes Comput. Sci., vol.1304, pp.213–222, 1997.

[10] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi, “Modeling the effect of technology trends on the soft error rate of combinational logic,” Proc. International Conference on Dependable Systems and Networks (DSN), pp.389–398, June 2002.

[11] B. Gill, N. Seifert, and V. Zia, “Comparison of alpha-particle and neutron-induced combinational and sequential logic error rates at the 32 nm technology node,” Proc. 2009 IEEE International Reliability Physics Symposium (IRPS), pp.199–205, April 2009.

[12] V. Betz, J. Rose, and A. Marquardt, Architecture and CAD for deep-submicron FPGAs, Kluwer Academic Publishers Norwell, MA, USA, 1999.

[13] S. Kirkpatrick, “Optimization by simulated annealing: Quantitative studies,” J. Statistical Physics, vol.34, no.5, pp.975–986, 1984.

[14] L. McMurchie and C. Ebeling, “PathFinder: A negotiation-based performance-driven router for FPGAs,” Proc. ACM 3rd International Symposium on Field-Programmable Gate Arrays, pp.111–117, 1995.

[15] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M.J. Wirthlin, “Improving FPGA design robustness with partial TMR,” Proc. 2006 IEEE International Reliability Physics Symposium (IRPS), pp.226–232, March 2006.

Takashi Imagawa received his B.E. degree in Electrical and Electronic Engineering, MSc degree in Communications and Computer Engineering, from Kyoto University in 2008 and 2010, respectively. 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.

Masayuki Hiromoto received his B.E. degree in Electrical and Electronic Engineering, MSc degree in Communications and Computer Engineering, and Ph.D. degrees in Informatics from Kyoto University in 2006, 2007, and 2009, respectively. Presently, he is a JSPS research fellow at the Department of Communications and Computer Engineering, Kyoto University. He is a member of IEEE and IPSJ.

Hiroyuki Ochi received the B.E., M.E., and Ph.D. degrees in Engineering from Kyoto University in 1989, 1991, and 1994, respectively. In 1994, he joined Department of Computer Engineering, Hiroshima City University as an associate professor. Since 2004, he has been an associate professor of Department of Communications and Computer Engineering, Kyoto University. He is a member of IPSJ, IEEE, and ACM.

Takashi Sato received the B.E. and M.E. degrees from Waseda University, Tokyo, Japan, and the Ph.D. degree from Kyoto University, Kyoto, Japan. He was with Hitachi, Ltd., Tokyo, Japan from 1991 to 2003, with Renesas Technology Corp., Tokyo, Japan, from 2003 to 2006, and with Tokyo Institute of Technology, Yokohama, Japan. In 2009, he joined the Graduate School of Informatics, Kyoto University, Kyoto, Japan, where he is currently a professor. He was a visiting industrial fellow at the University of California, Berkeley from 1998 to 1999. His research interests include CAD for nanometer-scale LSI design, fabrication-aware design methodology, and performance optimization for variation tolerance. Dr. Sato is a member of IEEE. He received the Beatrice Winner Award at ISSCC 2000 and the Best Paper Award at ISQED 2003.