IEICE TRANS. FUNDAMENTALS, VOL.E97–A, NO.12 DECEMBER 2014

PAPER Special Section on VLSI Design and CAD Algorithms

Reliability-Configurable Mixed-Grained Reconfigurable Array Supporting C-Based Design and Its Irradiation Testing

Hiroaki KONOURA[†,††], ***Student Member***, **Dawood ALNAJJAR**[†,††], ***Nonmember***, **Yukio MITSUYAMA**[†††,††a)], **Hajime SHIMADA**[††††,††], **Kazutoshi KOBAYASHI**[†††††,††], **Hiroyuki KANBARA**[††††††,††], **Hiroyuki OCHI**[*,††], ***Members***, **Takashi IMAGAWA**[**], ***Nonmember***, **Kazutoshi WAKABAYASHI**[***], ***Fellow***, **Masanori HASHIMOTO**[†,††b)], **Takao ONOYE**[†,††c)], ***Members***, ***and*** **Hidetoshi ONODERA**[**,††], ***Fellow***

SUMMARY This paper proposes a mixed-grained reconfigurable architecture consisting of fine-grained and coarse-grained fabrics, each of which can be configured for different levels of reliability depending on the reliability requirement of target applications, e.g. mission-critical applications to consumer products. Thanks to the fine-grained fabrics, the architecture can accommodate a state machine, which is indispensable for exploiting C-based behavioral synthesis to trade latency with resource usage through multi-step processing using dynamic reconfiguration. In implementing the architecture, the strategy of dynamic reconfiguration, the assignment of configuration storage and the number of implementable states are key factors that determine the achievable trade-off between used silicon area and latency. We thus split the configuration bits into two classes; state-wise configuration bits and state-invariant configuration bits for minimizing area overhead of configuration bit storage. Through a case study, we experimentally explore the appropriate number of implementable states. A proof-of-concept VLSI chip was fabricated in 65 nm process. Measurement results show that applications on the chip can be working in a harsh radiation environment. Irradiation tests also show the correlation between the number of sensitive bits and the mean time to failure. Furthermore, the temporal error rate of an example application due to soft errors in the datapath was measured and demonstrated for reliability-aware mapping.

key words: *reconfigurable architecture, soft error, radiation test, behavioral synthesis, state machine*

Manuscript received March 17, 2014.
Manuscript revised June 30, 2014.

[†]The authors are with the Department of Information Systems Engineering, Osaka University, Suita-shi, 565-0871 Japan.
[††]The authors are with CREST, JST, Kawaguchi-shi, 332-0012 Japan.
[†††]The author is with the School of Systems Engineering, Kochi University of Technology, Kami-shi, 782-8502 Japan.
[††††]The author is with the Information Technology Center, Nagoya University, Nagoya-shi, 464-8601 Japan.
[†††††]The author is with the Graduate School of Science and Technology, Kyoto Institute of Technology, Kyoto-shi, 606-8585 Japan.
[††††††]The author is with Advanced Scientific Technology & Management Research Institute of KYOTO, Kyoto-shi, 600-8813 Japan.
[*]The author is with the Graduate School of Information Science and Engineering, Ritsumeikan University, Kusatsu-shi, 525-8577 Japan.
[**]The authors are with the Department of Communications and Computer Engineering, Kyoto University, Kyoto-shi, 606-8501 Japan.
[***]The author is with NEC Corporation, Kawasaki-shi, 211-8666 Japan.
a) E-mail: mitsuyama.yukio@kochi-tech.ac.jp
b) E-mail: hasimoto@ist.osaka-u.ac.jp
c) E-mail: onoye@ist.osaka-u.ac.jp
DOI: 10.1587/transfun.E97.A.2518

1. Introduction

Coarse-grained reconfigurable architectures (CGRA) have been studied to fill the performance gap between FPGA and ASIC by reasonably limiting application domains and programmability. Recently, the reliability of reconfigurable devices is drawing attentions, since implementing mission-critical applications with high reliability on reconfigurable devices is highly demanded for saving NRE costs. Especially, soft errors are one of serious concerns threatening reliability of mission-critical applications. Soft errors include single event upset (SEU) where a charge occurring in memory elements causes a bit-flip and single event transient (SET) where a charge occurring in a combinational logic propagates to memory elements and causes a bit-flip. From reliability point of view, CGRA is inherently superior in soft error immunity to FPGA, since the amount of configuration bits is by orders of magnitude smaller than that of FPGA. [1]–[3] proposed several CGRAs with reliability consideration. Previously, we developed a reliability-configurable CGRA where the reliability level of each processing element (PE) can be chosen flexibly depending on applications and environments [4]. In [4], the trade-off between soft error immunity and area is successfully demonstrated on a 65 nm test chip under irradiation.

Thus far, while many CGRAs have been proposed (e.g. [5]–[8]), their adoption for commercial use is limited, especially when compared to FPGAs despite of their larger power dissipation and area. CGRA is basically composed of an array of ALUs handling multi-bit operands, and is suitable for data-path implementation. On the other hand, it is not good at efficiently implementing one-bit operations that are often found in flag computation, conditional branching and state machine. Especially, the incompatibility with state machine implementation is a significant problem preventing CGRA from being widely used, since RTL designers and existing behavioral synthesis tools for ASIC and FPGA synthesize data-path circuits that are controlled by state machines. Consequently, CGRA is not taking full benefit of the IP reuse nor the standard ANSI C/C++ source codes available.

To overcome this issue, several CGRA architectures that are compatible with state machine implementation are proposed. For instance, [9] proposes DRP architecture

which consists of a state transition controller and multiple PEs having 16 contexts and 8-bit/1-bit operators. This architecture enabled both one-bit operation and state machine implementation. Another example is XPP device [10], which has a configuration manager to enable the state transition in each tile. However, none of architectures have attained the compatibility with behavioral synthesis and reliability considerations. To expand the application domains of CGRA, an architecture having high compatibility with design automation tools and high flexibility in reliability to cover various applications is highly demanded.

In this paper, we propose a mixed-grain reconfigurable architecture that supports both behavioral synthesis and flexible reliability. The proposed architecture follows the concept of flexible reliability configuration presented in [4], which enables system designers to systematically trade off area for improving the soft error immunity without having a deep knowledge of reliability enhancement techniques. This work newly introduces one-bit PEs to implement a state machine that broadcasts the state signal to the array and dynamically reconfigures instructions given to CGRA. Consequently, designers can select one from various application implementations, e.g. a small area implementation with the large number of states or a low latency implementation with the small number of states.

We develop the architecture mentioned above which enables state-wise cycle-by-cycle dynamic reconfiguration, in contrast to multi-cycle reconfiguration [11]. To achieve this, we need to increase the capacity of local configuration memory in each PE in proportion to the number of states. In addition, to attain the immunity to soft errors, we need to introduce redundancy and error elimination mechanism into the configuration memory. For example, if triple modular redundancy (TMR) is adopted, the three-fold memory capacity becomes necessary. These two factors could tremendously increase the memory capacity for configuration bits, which could degrade area efficiency of the architecture.

To cope with this issue, we first adopt a strategy that the configuration bits for interconnection are not state-dependent, and only instructions to PE change according to the state signal. This strategy contributes to saving the memory capacity for configuration bits. Another important parameter to decide is the number of implementable states (#ImplSt). For exploring a wider trade-off between area and latency, the architecture that can implement the larger number of states is desirable. When #ImplSt is small, the obtainable trade-off between area and latency is limited. Meanwhile, when #ImplSt is excessively large, the area of configuration memory becomes significantly large. Especially, when a low latency implementation is selected from the trade-off, the number of used states is usually small. In this case, only a small portion of configuration memory is utilized, and consequently the unused memory results in area overhead. Furthermore, when #ImplSt is large, the state machine tends to be complex, and hence more one-bit PEs are necessary. This means that #ImplSt also affects the ratio of coarse- and fine-grained elements. Thus, #ImplSt must be carefully determined when implementing the architecture.

To investigate the appropriate #ImplSt, this work quantitatively evaluates two relationships between the number of states and resource usage and between #ImplSt and silicon area of a PE, respectively. Combining these evaluations, the achievable trade-offs between used silicon area and latency are illustrated for various #ImplSt, which suggests an appropriate #ImplSt.

The proposed architecture was fabricated with a 65 nm CMOS technology, and its reliability for soft error was clarified in the following two ways. A live video demonstration was performed showing data processing at different reliability levels with the presence of radiation. The proposed architecture can attain different soft error immunity levels through application mapping even with the same VLSI chip. Second, quantitative data of soft error rate was obtained using alpha-particle irradiation experiments and an FPGA. The obtained data is useful for reliability-aware mapping.

The rest of this paper is organized as follows. Section 2 reviews related works, and Sect. 3 presents the proposed architecture. A quantitative evaluation on #ImplSt is shown in Sect. 4. Section 5 explains the silicon implementation of the architecture. Experimental results including irradiation tests are shown in Sect. 6 and concluding remarks are given in Sect. 7.

2. Related Work

This section reviews conventional works of CGRA for attaining the compatibility with behavioral synthesis. None of these conventional works have supported reliability enhancement for reliability demanding applications.

DAPDNA architecture [13] contains digital application processor (DAP) and distributed network architecture (DNA). DNA is composed of reconfigurable ALU, delay, and RAM elements having four configuration codes. DAP core triggers the state transition of DNA, and DAP will receive an interrupt signal from DNA notifying reconfiguration completion after several clock cycles. [14] developed DAPDNA-FWII, which compiles and maps a C source code into DAP and DNA.

XPP architecture [10] has one context in each PE and reconfigures it by configuration manager (CM). In this architecture, CM, which consists of a state machine and internal RAM for configuration caching, reconfigures processing array elements (PAE) within a few clock cycles through their individual configuration caches when triggered by event packets from PAEs. [15] introduced XPP-VC high-level compiler which maps C programs to XPP.

DRP architecture, which has multiple contexts in each PE and implementing state machines by context switch, and its behavior synthesis are presented in [9]. In this architecture, PEs containing 8-bit/1-bit operators and 16 different configuration codes, are reconfigured within one clock cycle by the broadcasted state number from a state transition controller (STC).

KAHRISMA architecture [16] is composed of coarse-

and fine-grained encapsulated data-path elements (CG-EDPEs and FG-EDPEs) which are dynamically reconfigurable. CG-EDPE includes a context memory, a local sequencer, and a direct memory access. These components control the local state, i.e., dynamically reconfigure the operation of a processing block. [16] also presents a software framework which enables high-level compilation and mapping a C source code into CG- and FG-EDPEs.

3. Proposed Architecture

This section firstly explains the compatibility with behavioral synthesis, and then describes the proposed architecture.

3.1 Compatibility with High Level Synthesis

Compatibility with behavioral synthesis requires architectural supports that help provide a trade-off between latency and resource usage (area). For this purpose, multi-step processing through dynamic reconfiguration should be utilized. The same blocks have to be used in different time slices to perform different operations. Figure 1 shows a simple example demonstrating how multi-step processing is performed. The figure demonstrates how a C program can be implemented in one cycle and in two cycles. In the one cycle implementation of Fig. 1(a), two PEs are required (adder and subtractor). In this case, the configuration of the PEs is fixed, and the two PEs constantly perform the same operations until the device is reconfigured for another application. This is an ordinary output of common synthesis tools.

Our architecture supports not only one-cycle implementation of Fig. 1(a) but also multi-cycle implementation. In the two-cycle implementation of Fig. 1(b), one PE including two multiplexers and a register, and a state machine are necessary. Dynamic reconfiguration of the PE is repeated. The state machine has two states, S0 and S1. During S0, the addition operation is selected to add a and b, and during S1, the c is subtracted from the output of the previous operation. Such a trade-off between latency and area obtained by various implementations enables various types of desirable specifications.

In order to achieve this trade-off, the following elements are required: one-bit PEs to implement state machines, coarse-grained PEs to perform various types of data processing with dynamic reconfiguration depending on the state signal, register files to save temporal data, and large memories to store large bulk data. Here, although an embedded CPU might be able to control the state of coarse-grained PEs, we selected one-bit PEs for pursuing low-latency state control. With these elements, behavioral synthesis allows designers to explore the solution space and select an implementation that satisfies their requirements.

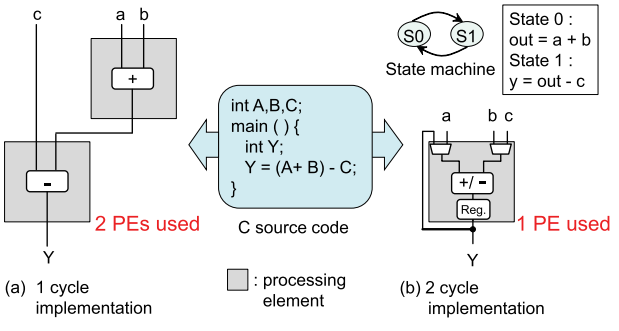

Fig. 1 Examples of implementations with different latency.

3.2 Architecture Design Overview

The proposed architecture is composed of coarse-grained elements as ALU clusters, fine-grained elements as LUT clusters, and memory blocks called as MEM clusters, where the basic element is noted as cluster. When an application is mapped on the proposed architecture, data-paths are assigned to ALU clusters. Meanwhile, state machines and one-bit operations are assigned to LUT clusters. Each ALU cluster behaves as different functional units depending on the state. Due to this, on the other hand, ALU clusters themselves cannot hold register values which will be used after a while. To store the temporal intermediate data across the different states, register blocks called as REG clusters are also included in the proposed architecture.

ALU, LUT and REG clusters have the similar inner structure with three cells illustrated in Fig. 2, while execution modules (EM) in cells are different. EMs for ALU, LUT and REG clusters are ALU, LUT and register file. On the other hand, MEM cluster includes a single SRAM macro, and hence the structure inside the cluster is different. The details of each cluster are explained in the next subsection. Several LUT clusters are organized in a two dimensional array forming a LUT block. ALU, REG and MEM

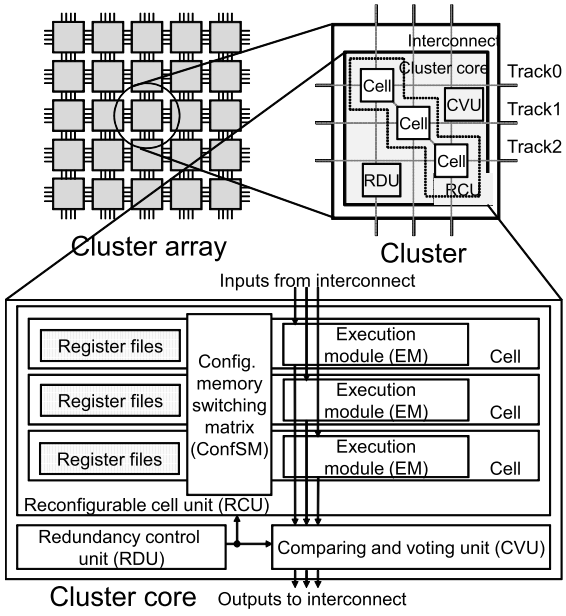

Fig. 2 Cluster and cluster interconnection. The structure inside cluster shown this figure corresponds to ALU, LUT and REG clusters.

clusters and LUT blocks are placed in a two-dimensional array.

In the architecture design, we selected the following strategy. ALU functionality and ALU operand selection are dynamically reconfigured according to the broadcasted state signals, which will be detailed in the next section, while the inter-cluster interconnection is unchanged. All the inter-cluster routings to provide data to clusters are fixed for all the states, and each ALU cluster selects a few data as operands depending on the state from all the data delivered to the ALU cluster. The reason why this strategy was selected is that the amount of configuration bits for inter-cluster interconnection is quite large, and it is difficult to multiply it by #ImplSt from area perspective.

The architecture has two global signals; context signal and state signal. Both of these global signals are generated by designated LUT clusters. The purpose of context signal is to switch the mapped application or algorithm, and then the inter-cluster interconnection is changed according to the context. This context signal is not discussed further in this paper.

Before explaining details of ALU, LUT, MEM and REG clusters, the treatment of the state signal in each cluster is briefly summarized. ALU cluster changes its functionality and data-path/flag operands according to the broadcasted state signal. Also, REG cluster selects input data and changes write address depending on the state signal, since the store of intermediate data depends on the state. On the other hand, the functionalities of LUT and MEM clusters are unchanged.

3.3 Details of Four Clusters

3.3.1 ALU Cluster

ALU cluster is derived from the reliability-flexible architecture proposed in [4]; however, it is highly improved to support cycle-by-cycle dynamic reconfiguration. The cycle-by-cycle dynamic reconfiguration is controlled by a state machine implemented with LUT clusters. ALU cluster receives the state signal from the state machine, selects an instruction from the local configuration memory depending on the current state and executes the selected instruction, which is the most significant difference from [4] in ALU cluster.

As shown in Fig. 2, an ALU cluster consists of a reconfigurable cell unit (RCU) processing various types of operations, a redundancy control unit (RDU) for flexible reliability, a comparing and voting unit (CVU), switches and wires. An RCU is composed of a configuration memory switching matrix (ConfSM) and three cells, each of which contains an execution module (EM), register files for storing configuration bits, and voters. In EM, arithmetic operation including multiplication, logic operation, and shift operation are performed.

The cluster interconnection has three tracks (Track 0 – 2), and each cell inside a cluster is placed on one of them. Thus, each cell in a cluster can be connected to the cells

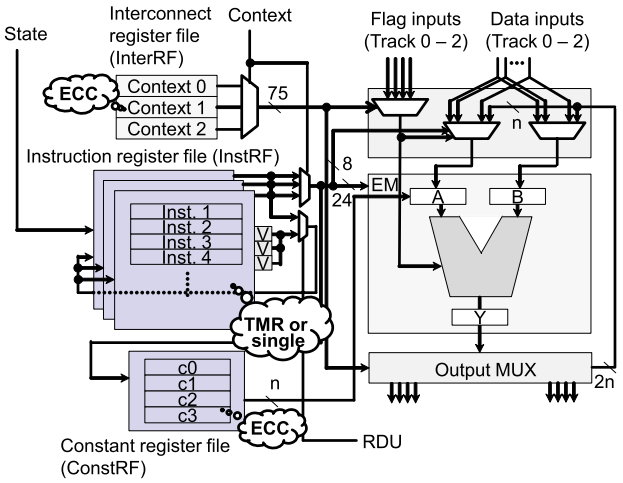

Fig. 3 Architecture of cell in ALU cluster.

in adjacent clusters on the same track. The interconnection also has a diagonal track, connecting cells within one cluster. Switches to control these interconnections are implemented by multiplexers.

The cell architecture of ALU cluster is illustrated in Fig. 3. In order to implement dynamic reconfiguration with reduced area overhead, configuration bits are divided and stored in three types of register files: instruction register file (InstRF), interconnection register file (InterRF), and constants register file (ConstRF). The bit widths of InstRF, InterRF and ConstRF are 32, 75, and 16 bits, respectively. As mentioned earlier, instructions for ALU are locally stored in the cluster and the number of instructions stored is #ImplSt, where an instruction represents a set of ALU configuration bits for a single state. On the other hand, the configuration bits for inter-cluster connection stored in InterRF are fixed for all the states in each context. In this paper, InterRF is implemented so that it can store three contexts. ConstRF is used to store up to four constants that are required in the application, and one of four constants is selected by the 2-bit signal from InstRF. This implementation is selected for area reduction because at most only a few instructions need constants in most applications.

To attain soft error immunity, InterRF and ConstRF are protected with ECC. The selected code is single error correction/double error detection (SEC/DED) hamming code. For every read of InterRF and ConstRF, the error corrected bits are regularly restored in InterRF and ConstRF to prevent error accumulation. In addition, three contexts of InterRF and four constants of ConstRF are restored by re-writing the data itself through another SEC/DED encoder/decoder in rotation. On the other hand, InstRF is implemented with bit-wise TMR, since the path from the state signal to the register file output includes only a voter and its delay is small. This small delay is important, since this delay is necessarily included in the critical path. This is the reason why ECC, which needs ECC decoder having large delay, was not selected for InstRF.

ALU cluster supports three operation modes: triple

Table 1 Redundancy and reliability to soft errors in three operation modes in ALU cluster.

Operation mode	Redundancy		SEU in InstRF	SEU in EM	SET in EM	Utilization	
	InstRF	EM				#contexts	#cells
TMR	3	3	D & R	D & R	D & R	3	3
SMS	3	1	D & R	D	ND	1	1
SMM	1	1	ND	D	ND	3	1

D & R : Detection and recovery, D : Detection, ND : Does not detect

modular redundancy (TMR), single modular with single context (SMS), and single modular with multi-context (SMM), as summarized in Table 1. These operation modes offer different capabilities of dynamic reconfigurability (no. of contexts) and throughput per cluster. TMR in which both InstRF and data-paths are triplicated provides the highest soft error immunity. Meanwhile, in SMS, only InstRF is triplicated but the data-path is singular. In both TMR and SMS, an SEU occurring in the InstRF will be repaired when the next configuration clock is given, since the voted value is re-written to the register file every configuration clock cycle. Here, the configuration clock signal is given to the memory elements for configuration information separately from the system clock. This configuration data is stored with bit-wise TMR, and therefore, multiple SEUs in different bits will also be corrected when the next configuration clock is given. On the other hand, in SMM, the voters are disabled, and three contexts are stored independently using three InstRFs, each of which is included in individual cells. With this implementation, users can flexibly choose the operation modes, depending on the importance of SEUs in InstRF and SEU/SET in EM.

In this architecture, as pointed out earlier, InstRF consisting of a larger number of words can accommodate a larger state machine, which enables area-efficient implementation trading larger latency. However, as #ImplSt becomes larger, the silicon area of ALU cluster increases, and the area overhead originating from the unused words of InstRF tends to be significant. This trade-off will be discussed in Sect. 4.

3.3.2 LUT Cluster

An LUT cluster supports reliable and regular modes. The LUT cluster architecture is shown in Fig. 4. The LUT cluster contains three cells and each cell contains one configuration memory (ConfMem), LUT data registers, a pipeline register, wires and selectors. In reliable mode, the three ConfMems, three LUT data registers, and three data-paths in three cells are redundantly used. By this, SEUs in ConfMem and the LUT data registers are corrected by re-writing with the voters in the VC and VL. In regular mode, no redundancy is applied, and the given data are independently processed in each cell. The operation modes in LUT cluster are summarized in Table 2. The operation modes of LUT cluster are controlled through a 1-bit triplicated value stored in the RDU.

LUT cell contains a 4-input LUT that can be cascaded with other cells to form a larger LUT. It can receive flags

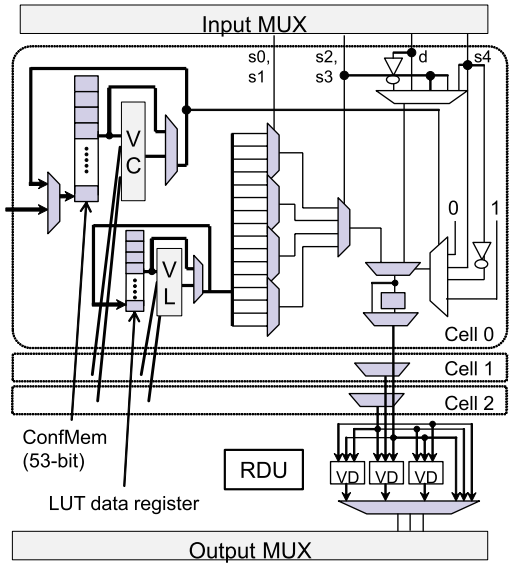

Fig. 4 LUT cluster architecture.

such as zero flag, overflow, underflow, most four least significant bits of the result, and carry generated in ALU cells, and can perform conditional operations, flag multiplexing, flag inversion, pipelining, and fixed outputs. Any single bit of n-bit ALU output can be provided to LUT cells via multi-cycle shifting. Thus, the architecture offers a significant amount of temporal flexibility in providing data to be able to take full advantage of the fine-grained fabric. With receiving necessary 1-bit data, LUT clusters can perform one-bit operations and can form a state machine efficiently.

In the array, a set of LUT clusters, whose number depends on #ImplSt, are designated to output and broadcast the state signal. Similarly, another set of LUT clusters are responsible to output the context signal. Besides, LUT clusters are supposed to be organized in a two dimensional array forming LUT blocks, which makes the area of LUT block comparable to those of other clusters. LUT blocks, ALU, register and MEM clusters are placed in a two-dimensional array, as explained earlier.

3.3.3 MEM Cluster

MEM cluster is composed of one 1,024-word $\times(n + k)$-bit two-port SRAM, where n represents the data-path width and k is the number of redundant bits. Although the SRAM itself is protected with SEC/DED hamming codes, the words which have not had a write access for a while are likely to accumulate multiple SEUs within a word, which results in

Table 2 Redundancy and reliability to soft errors in two operation modes in LUT and REG clusters.

Operation mode	Redundancy		SEU in ConfMem	SEU in LUT data register & register file	SET in data-path	Utilization	
	ConfMem	data-path				#contexts	#cells
Reliable	3	3	D & R	D & R	D & R	1	3
Regular	1	1	ND	ND	ND	1	1

D & R : Detection and recovery, ND : Does not detect

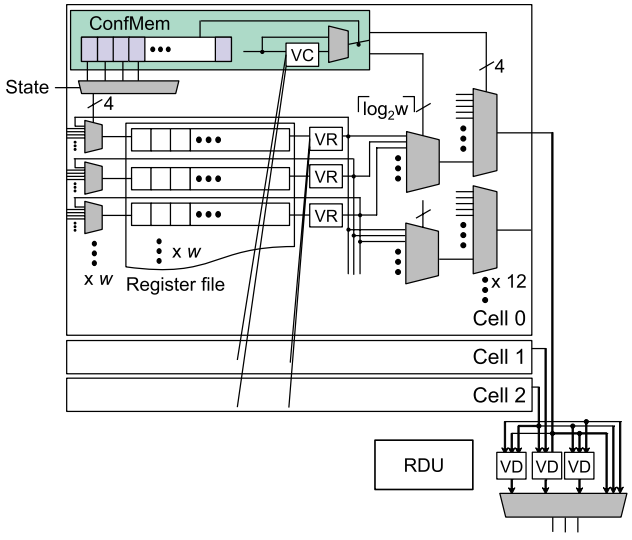

Fig. 5 REG cluster architecture.

uncorrectable errors. To avoid such uncorrectable errors, MEM cluster offers reliable mode in addition to regular mode. In regular mode, two ports of SRAM are independently utilized to read/write data. Meanwhile, in reliable mode, one port is used to read/write data, and the other port is designated for periodic overwriting through ECC decoder and encoder. Note that regular mode could be robust enough for applications that keep data for a short time, since SEU accumulation less likely happens.

MEM cluster has only one cell due to the size of the SRAM. However, it is compatible with the three-cell implementation of ALU cluster, the three-track flag and data interconnection, and the reliability modes of ALU cluster. MEM cluster also contains three tracks, all connected to the same cell. Input signals of SRAM such as address and enable are drawn with three tracks and voted in front of the SRAM macro. The read data of SRAM is also distributed to the three tracks. Herewith, single error points except for the inside of SRAM macro are minimized.

3.3.4 REG Cluster

Architecture of REG cluster is shown in Fig. 5. REG cluster has three cells composed of a ConfMem, a w-word register file, wires and switches.

Similarly to LUT cluster, REG cluster supports reliable and regular modes. In reliable mode, three ConfMems, three register files, and three data-paths in three cells are redundant. By this, SEUs in ConfMem and register files are corrected by re-writing through the voters in the VC and VR. Also, SETs in data-path can be corrected by the voters in the VD. In regular mode, no redundancy is applied, and the given data are independently processed in each cell. The reliability modes in REG cluster are summarized in Table 2. The REG cluster operation mode is controlled through a 1-bit triplicated value stored in the RDU.

The REG cluster is responsible for storing and exchanging temporal data across different states. For this purpose, input data must be delivered to one of the registers depending on the state. This data delivery is achieved by w input multiplexers, and controlled by ConfMem, where ConfMem can store #ImplSt configuration sets. The relationship between #ImplSt and the area of REG cluster is evaluated in Sect. 4.

4. Architecture Evaluation

As explained earlier, when implementing the proposed architecture, #ImplSt is a key parameter that determines the silicon overhead and the achievable trade-off between latency and used silicon area. This section experimentally evaluates the appropriate #ImplSt through a case study. The reliability of the proposed architecture will be evaluated with silicon implementation in Sect. 6.

In this evaluation, a behavioral synthesis tool [17], [18] slightly customized for the proposed architecture is used to obtain various implementations with different latencies and ALU usages from a C source code of 512-point radix-8 FFT. Note that the bit width of ALU is supposed to be 16, and the operation frequency is set to 100 MHz. Each cluster is implemented with Verilog HDL and synthesized with a 65 nm cell library. The area of each cluster is estimated from the reports of logic synthesis tool. The reliability modes of ALU/LUT/REG clusters are configured with TMR/reliable/reliable, respectively.

4.1 Number of Used Clusters vs. Number of States

The numbers of used ALU/LUT/REG clusters change depending on the number of states. Figure 6 shows the relationship between the number of states and the number of ALU clusters in use. We can see that the number of ALU clusters in use decreases as the number of states increases, which is a well-known relationship obtained by behavioral synthesis. Second, Fig. 7 shows the relationship between the number of states and the number of LUT clusters in use. As the number of states increases, the state machine becomes complex, and the number of LUT clusters increases.

Moreover, Table 3 lists the required number of 16-bit registers and w-word REG clusters for inter-state data ex-

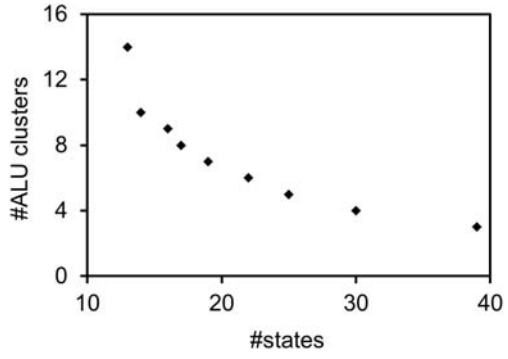

Fig. 6 Relationship between # of states and # of ALU clusters.

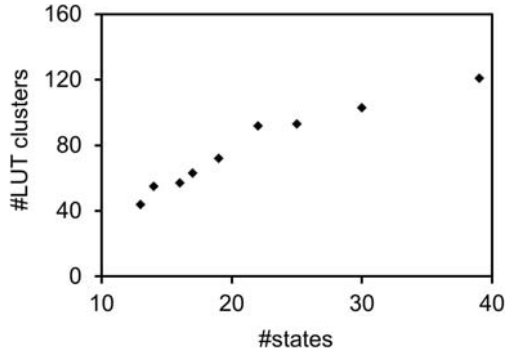

Fig. 7 Relationship between # of states and # of LUT clusters.

Table 3 Registers for inter-state data exchange.

#states	#registers for buffering	#REG clusters		
		$w = 4$	$w = 8$	$w = 16$
13	14	4	2	1
16	17	5	3	2
19	23	6	3	2
25	24	6	3	2
39	28	7	4	2

change. As the number of states increases, the number of registers for inter-state data exchange increases because the data exchange across the states occurs more often. When the number of states is 39, the required number of four-word REG clusters is seven, on the other hand, that of 16-word REG clusters is two. The area of REG cluster, which depends on the number of words and #ImplSt, is evaluated in Sect. 4.3.

4.2 ALU Cluster Area vs. Number of Implementable States

An increase in #ImplSt involves a significant increase in the area of ALU cluster having triplicated InstRFs. Remind that #ImplSt is equal to the number of words of InstRF in ALU clusters. Figure 8 shows the relationship between #ImplSt and the area of ALU cluster. As the number of states increases, the area of InstRFs linearly increases and gradually becomes dominant. When #ImplSt is 28, the area of ALU cluster reaches twofold compared with that of ALU cluster whose #ImplSt is 1. Here, it should be noted that the relation between #ImplSt and the area of ALU cluster is application-independent.

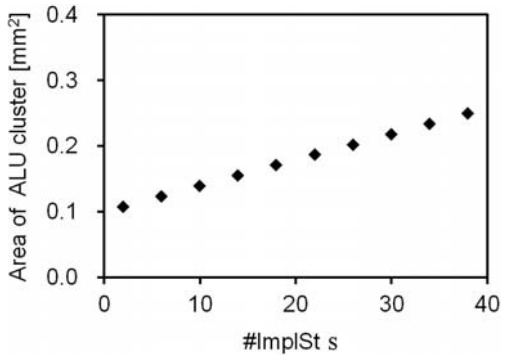

Fig. 8 Relationship between #ImplSt and area of ALU cluster.

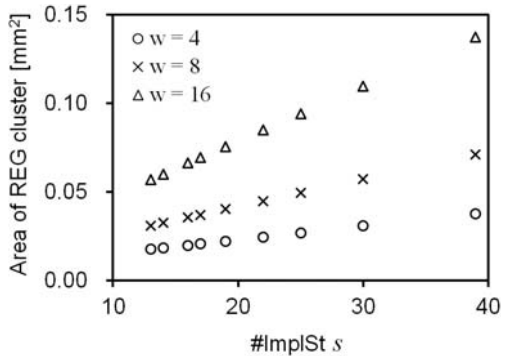

Fig. 9 Relationship between #ImplSt and area of REG cluster.

4.3 Area of REG Cluster

As shown in Fig. 5, an increase in #ImplSt enlarges configuration memories. Also, the number of words w is directly related to the sizes of register file and configuration memory. Figure 9 shows the relationship between #ImplSt and the area of REG cluster for various numbers of word, where this relationship is application-independent. There is a mostly linear relation between #ImplSt and the area of REG cluster. In a case that 39 states are implementable, the area of 16-word REG cluster (= 0.137 mm^2) is 3.7 times as large as that of a four-word REG cluster (= 0.038 mm^2), i.e., there is the proportional relation between the number of words and the area of REG cluster. Here, remind that the required number of REG clusters for application mapping is inversely proportional to the number of words in REG cluster as shown in Table 3. These results show the total area of REG clusters is not sensitive to the number of words, and we can select a reasonable number of words according to the capacity of inter-cluster interconnection.

4.4 Appropriate Number of Implementable States

We then evaluate the achievable trade-offs between used sil-

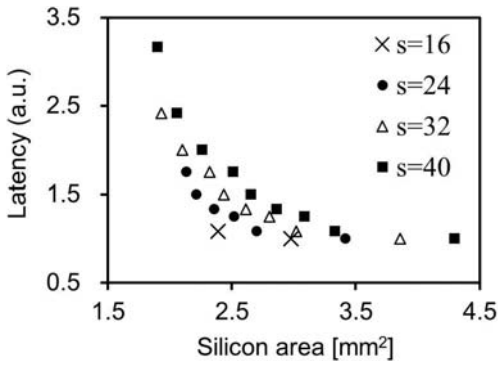

Fig. 10 Relationship between used silicon area and latency.

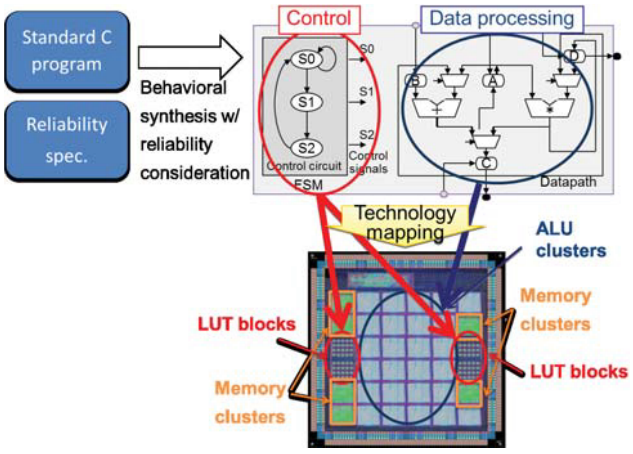

Fig. 11 Designed reconfigurable array and design flow.

icon area and latency for various #ImplSt. Figure 10 shows the results. Please remind that our previous architecture [4] cannot implement a state machine and hence such trade-offs cannot be explored with behavioral synthesis. The used silicon area includes those of used ALU, LUT, MEM, and REG clusters, and the dependency of ALU cluster area on #ImplSt (Fig. 8) is taken into consideration. Here, four MEM clusters, one of which is 0.111 mm^2, are included. We assumed to use 16-word REG clusters. As can be seen from Fig. 10, when #ImplSt is small such as 16, a small-area implementation can be achieved; however, the achievable trade-off between used silicon area and latency is quite limited. On the other hand, when #ImplSt is large such as 40, a small silicon area implementation becomes obtainable trading for latency, whereas the used silicon area of the implementation pursuing the minimum latency becomes large. While the best #ImplSt depends on the requirements of area, performance and design flexibility, in this test case, the range from 24 to 32 is a reasonable number to take advantage of behavioral synthesis with limited overhead. Thus, the proposed mixed-grained architecture can obtain various implementations on the same cluster array making use of behavioral synthesis from a C source code.

5. Test Chip Implementation

We implemented the proposed array in a 65 nm process. Figure 11 illustrates the mixed-grained array consisting 26 ALU clusters, 6 memory clusters and 4 LUT blocks with the chip layout. Here, REG clusters are not included in the test chip since our preliminary evaluation before the test chip implementation shows that small image processing applications that can be implemented on the test chip demand more ALU clusters instead of REG clusters. A customized version of Cyber Workbench [17], [18], which is a commercially-available behavioral synthesis tool, synthesizes an RTL implementation from a C program taking into account reliability specification given by [19]. Finally, the RTL implementation is mapped on the array through technology mapping and P&R.

Thanks to dynamic reconfiguration using states generated by an FSM, area-efficient mapping becomes possible. Here, in this implementation, #ImptSt was set to 16 due to the limited silicon area, while the previous section suggested that #ImptSt of more than 24 is desirable. Nevertheless, for example, an edge detection filter can be implemented with 25 ALU clusters, while 62 ALU clusters are necessary without dynamic reconfiguration (i.e. in a single state). The number of clusters is reduced by 60%. Note that the architecture without dynamic reconfiguration corresponds to our previous architecture [4]. Thus, the proposed architecture implemented on the test chip can exploit such latency-area exploration in behavioral synthesis.

The array was fabricated in 65 nm 12ML CMOS process. The die size is 4.2×4.2 mm^2. Loading the configuration bits is performed through a scan-chain consisting of 165,312 FFs. ALU, LUT and memory clusters include 120 k, 4 k and 99 k gates, respectively.

6. Measurement Results for Reliability Evaluation

This section presents measurement results of the fabricated prototype chip. We first evaluated the maximum speed of the state and context distribution using a PLL on the chip, since the maximum frequency is limited by the distribution of state and context signals. Results show that the maximum frequency possible to propagate state signals is 240 MHz, and it is 165 MHz for context signal. In the following subsections, results of irradiation testing are described.

6.1 Demonstration

To validate the functionality and reliability of the architecture, a demonstration using two mappings of SMM and SMS was performed (Fig. 12). The chip receives a live data stream from a video camera, processes it, and sends it to a monitor demonstrating the processed stream. The mapping of black and white reversal filter was generated from a C source code. Here, the black and white reversal filter is a filter that performs tone reversal, and a subtraction is performed for each pixel.

A snapshot of the results is shown in Fig. 13. Af-

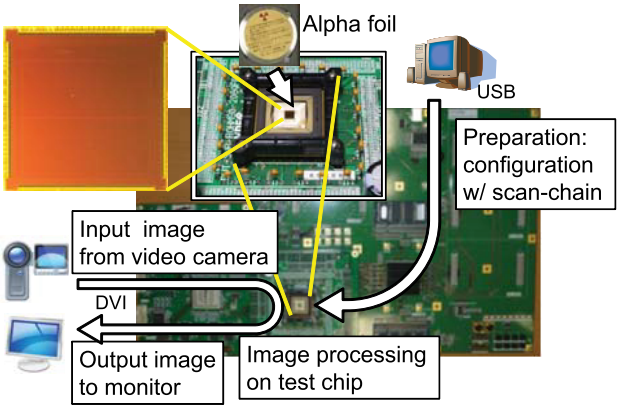

Fig. 12 Demonstration setup.

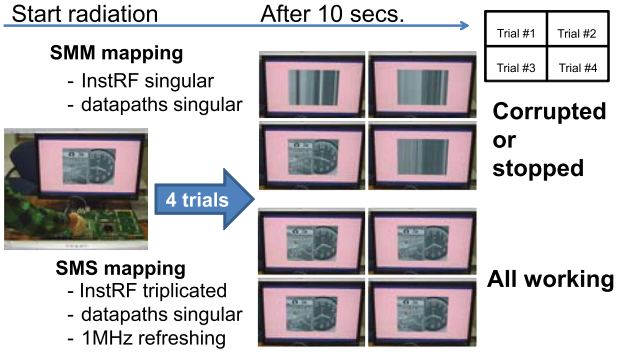

Fig. 13 Results of image processing under irradiation.

ter positioning an Am-241 alpha foil whose flux is 9 $\times 10^9$ cm^{-2}h^{-1} over the chip [20], it was observed that SMS mapping continued to output the processed video as expected. On the other hand, SMM mapping got destroyed in 2 s due to SEUs in configuration registers and video processing stopped within 10 s in all four trials. We also tested TMR mapping and confirmed the expected continuous functionality. The proposed architecture thus enables reliable operation even under harsh irradiation through application mapping.

6.2 Irradiation Experiment

To quantitatively evaluate the immunity to soft errors, we carried out experiments of alpha particle irradiation. Figure 14 illustrates the test configuration. The ALU clusters on the array are serially connected to compose a pipelined chain. For simplicity, NOT operation is selected as the ALU function. We also implemented the same function as a golden circuit on an FPGA. A PC gives the same input patterns to the array and the golden circuit. The outputs of the array and the golden circuit are compared on the FPGA, and the inconsistency between them is detected as an error. Here, there are two types of the errors; a permanent error and a temporal error. The permanent error happens when the configuration information is corrupt due to SEUs in the configuration memory and the circuit functionality becomes

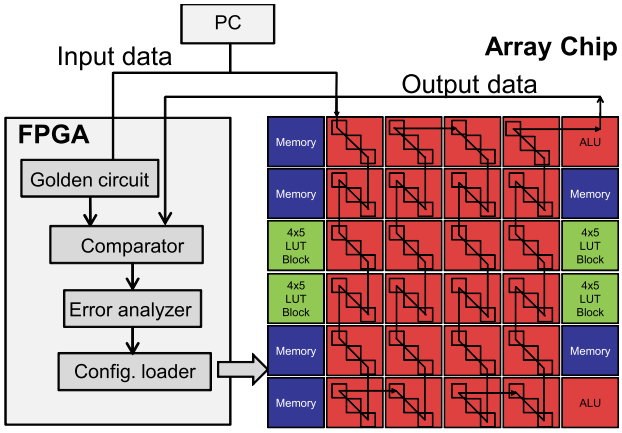

Fig. 14 Irradiation test configuration.

wrong. On the other hand, a temporal error originates from SEUs and SETs in the datapath. In the current configuration, this temporal error does not accumulate in the datapath and is eventually flushed out from the pipeline.

To distinguish the permanent errors and the temporal errors, we implemented an error analyzer on the FPGA. If the inconsistency at the outputs continues for five clock cycles, we regard this situation as a permanent error. When the inconsistency lasts for less than five cycles, we judge that there is a temporal error. We count the numbers of permanent and temporal errors. In the case of the permanent errors, we need to reload the configuration information to eliminate the SEUs in the configuration memory, and Config. loader fills this role.

We first evaluated, using the scan chain, the number of SEUs accumulated in the configuration memory when a permanent error is detected. In the configuration memory, there are don't care bits which do not affect the functionality, and hence the number of accumulated SEUs could be larger than 1. Figure 15 shows the histogram. We can see that the number of accumulated SEUs less than 50 is most frequent, but there are cases having a large number of SEUs accumulated before a permanent error arises. To accurately estimate the soft error rate of reconfigurable devices, we need to pay attention to these don't care bits.

We next evaluate the MTTF (mean time to failures) focusing on the permanent error, where the temporal errors are not regarded as failures in this evaluation. A refreshing clock of 15 MHz is given to eliminate SEUs from triplicated memory (InstRF) and registers with ECC (InterRF). We prepared five array configurations; all SMM, SMM/SMS, all SMS, SMS/TMR and all TMR. In the configuration of SMM/SMS (SMS/TMR), 50% of the ALU clusters are in SMM (SMS) level and the others are in SMS (TMR) level. Note that in all TMR, voters and inter-cluster connections are triplicated and hence there is no single point of failure.

Table 4 lists the measured MTTF and the number of permanent errors observed in about 300-s radiation. Table 4 also includes the number of sensitive bits, where a configuration memory element that impacts the primary output of

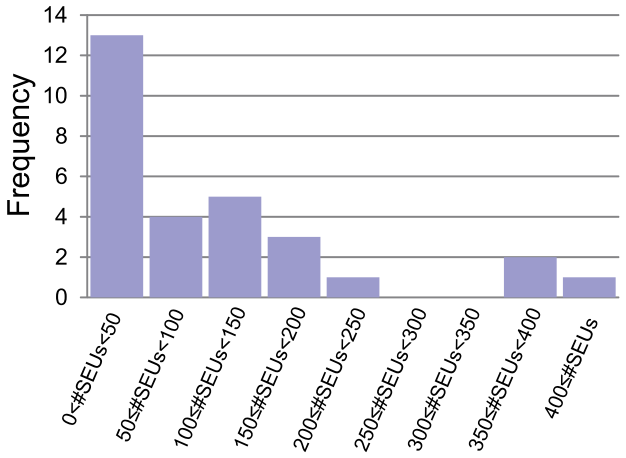

Fig. 15 Number of accumulated SEUs when a permanent error is detected.

Table 4 MTTF and sensitive bits.

Mapping	# of sensitive bits	MTTF [s]	# of permanent errors
All SMM	1,575	1.51	198
SMM/SMS	819	3.8	84
All SMS	0	>300	0
SMS/TMR	0	>300	0
All TMR	0	>300	0

a particular design is defined as a "sensitive bit". Thus, the don't care bits are not included in the sensitive bits. In all SMM configuration, the number of sensitive bits is 1,575, and it is the largest. On the other hand, the MTTF is 1.51 s, and it is the shortest. When SMM/SMS configuration is selected, the number of sensitive bits is reduced to 819, and consequently the MTTF is extended to 3.8 s. On the one hand, all SMS, SMS/TMR and all TMR configurations do not include any sensitive bits, and for these no permanent errors were observed. These results clearly show that the MTTF is strongly dependent on the number of sensitive bits, and the system designer can trade area with robustness to soft errors guided by the number of sensitive bits, which means the concept of flexible reliability firstly proposed in [4] can be exploited in the proposed architecture.

In both the proposed architecture and previous one [4], the number of sensitive bits can be controlled by mapping, i.e. the selection of TMR, SMM and SMS modes, and it can be reduced to zero by selecting TMR or SMS mode for all the ALU, LUT and REG clusters as presented above. In addition, the datapath and state machine can be triplicated in TMR mode. From this point, the soft error immunity is at the same level. Strictly speaking, on the other hand, the architecture proposed in this paper newly introduced MEM register that used an off-the-shelf SRAM macro. We applied ECC to this SRAM, but the SRAM macro might include a sensitive point, and it may degrade the soft error immunity. However, the internal circuit information of the SRAM macro was not disclosed, and hence the immunity of the off-the-shelf SRAM macro with ECC was not unclear. Meanwhile, at least unexpected reliability degradation in the proposed architecture was not observed in the irradiation test shown in Sect. 6.1.

We finally evaluate the temporal error rate. While all SMS configuration is expected to have very few permanent errors as explained, the datapath is not redundant and temporal errors are supposed to happen. We counted the number of temporal errors in all SMS configuration. During 19-min irradiation, 1,109 temporal errors were observed and the rate is 0.95 bit-flip per second. Assuming an LSI package whose alpha emission rate is $10\,\mathrm{cm}^{-2}\mathrm{h}^{-1}$, this rate corresponds to 4.3×10^3 FIT in a nominal environment. If this FIT rate is not acceptable, the system designer needs to use TMR level. On the other hand, if this FIT rate is acceptable, SMS level is a good choice for area efficiency. Thus, the proposed array can be used for various environments and reliability specifications.

7. Conclusion

We proposed a mixed-grained reconfigurable architecture supporting C-based behavioral synthesis and flexible reliability. We experimentally evaluated trade-offs between used silicon area and latency with various numbers of implementable states using 512-point radix-8 FFT as an application example. In this evaluation, the proposed architecture whose number of implementable states in the range from 24 to 32 can accommodate various implementations in latency and area obtained by behavioral synthesis. We implemented a 65 nm test chip of the proposed mixed-grained reconfigurable array. We irradiated the fabricated test chip with alpha foil and confirmed that the application mapped for high reliability kept functioning while the application mapped for ordinary reliability stopped due to permanent errors in configuration memory. In addition, we quantitatively evaluated the soft error immunity of each reliability level, and showed a concrete FIT number originating from SEUs and SETs in the datapath. These evaluation results give underlying characteristics of the array, which are indispensable in reliability-aware mapping.

Acknowledgement

The VLSI chip in this study has been fabricated in the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with eShuttle Inc. and Fujitsu Semiconductor Ltd. The authors thank Dr. Shinichi Noda for his contribution to Cyber customization and experiments for architecture evaluation.

References

[1] S.M.A.H. Jafri, S.J. Piestrak, O. Sentieys, and S. Pillement, "Design of a fault-tolerant coarse-grained reconfigurable architecture: A case study," Proc. ISQED, pp.845–852, March 2010.
[2] M.M. Azeem, S.J. Piestrak, O. Sentieys, and S. Pillement, "Error recovery technique for coarse-grained reconfigurable architectures," Proc. DDECS, pp.441–446, April 2011.
[3] T. Schweizer, P. Schlicker, S. Eisenhardt, T. Kuhn, and W.

Rosenstiel, "Low-cost TMR for fault-tolerance on coarse-grained reconfigurable architectures," Proc. ReConFig, pp.135–140, Nov.–Dec. 2011.
[4] D. Alnajjar, H. Konoura, Y. Ko, Y. Mitsuyama, M. Hashimoto, and T. Onoye, "Implementing flexible reliability in a coarse grained reconfigurable architecture," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.21, no.12, pp.2165–2178, Dec. 2013.
[5] S.C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R.R. Taylor, "PipeRench: A reconfigurable architecture and compiler," IEEE Trans. Comput., vol.33, no.4, pp.70–77, April 2000.
[6] M. Myjak and J.G. Delgado-Frias, "A two-level reconfigurable architecture for digital signal processing," Microelectronic Engineering, vol.84, no.2, pp.244–252, Feb. 2007.
[7] C. Ebeling, D.C. Cronquist, and P. Franklin, "RaPiD — Reconfigurable pipelined data-path," Proc. FPL, pp.126–135, Sept. 1996.
[8] Y. Mitsuyama, K. Takahashi, R. Imai, M. Hashimoto, T. Onoye, and I. Shirakawa, "Area-efficient reconfigurable architecture for media processing," IEICE Trans. Fundamentals, vol.E91-A, no.12, pp.3651–3662, Dec. 2008.
[9] T. Toi, N. Nakamura, Y. Kato, T. Awashima, K. Wakabayashi, and J. Li, "High-level synthesis challenges and solutions for a dynamically reconfigurable processor," Proc. ICCAD, pp.702–708, Nov. 2006.
[10] V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach, and M. Weinhardt, "PACT XPP — A self-reconfigurable data processing architecture," the Journal of Supercomputing, vol.26, no.2, pp.167–184, Sept. 2003.
[11] L. Bauer, M. Shafique, S. Kramer, and J. Henkel, "RISPP: Rotating instruction set processing platform," Proc. DAC, pp.791–796, June 2007.
[12] D. Alnajjar, H. Konoura, Y. Mitsuyama, H. Shimada, K. Kobayashi, H. Kanbara, H. Ochi, T. Imagawa, S. Noda, K. Wakabayashi, M. Hashimoto, T. Onoye, and H. Onodera, "Reliability-configurable mixed-grained reconfigurable array supporting C-to-array mapping and its radiation testing," Proc. A-SSCC, pp.313–316, Nov. 2013.
[13] T. Sugawara, K. Ide, and T. Sato, "Dynamically reconfigurable processor implementation with IPFlex's DAPDNA technology," IEICE Trans. Inf. Syst., vol.E87-D, no.8, pp.1997–2003, Aug. 2004.
[14] T. Sato, H. Watanabe, and K. Shiba, "Implementation of dynamically reconfigurable processor DAPDNA-2," Proc. VLSI-TSA-DAT, pp.323–324, April 2005.
[15] J.M.P. Cardoso and M. Weinhardt, "From C programs to the configure-execute model," Proc. DATE, pp.576–581, March 2003.
[16] R. Koenig, L. Bauer, T. Stripf, M. Shafique, W. Ahmed, J. Becker, and J. Henkel, "KAHRISMA: A novel hypermorphic reconfigurable-instruction-set multi-grained-array architecture," Proc. DATE, pp.819–824, March 2010.
[17] K. Wakabayashi and T. Okamoto, "C-based SoC design flow and EDA tools: An ASIC and system vendor perspective," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.19, no.12, pp.1507–1522, Dec. 2000.
[18] CyberWorkbench, http://www.nec.com/en/global/prod/cwb/index.html
[19] T. Imagawa, H. Tsutsui, H. Ochi, and T. Sato, "A cost-effective selective TMR for heterogeneous coarse-grained reconfigurable architectures based on DFG-level vulnerability analysis," Proc. DATE, pp.701–706, 2013.
[20] JEDEC standard JESD89, "Measurement and reporting of alpha particles and terrestrial cosmic ray-induced soft-errors in semiconductor devices."

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

Dawood Alnajjar received the B.E. degree from the University of Jordan, Amman, Jordan in 2006. He received the M.E. and Ph.D. degrees in Information Systems Engineering from Osaka university, Osaka, Japan in 2010 and 2013, respectively. His interest includes reconfigurable computing, computer architecture, and reliable systems.

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

Hajime Shimada was born in 1976 and received his B.E., M.E. and D.E. degrees from Nagoya University, Japan in 1998, 2000 and 2004 respectively. He was an assistant professor in Kyoto University from 2005 to 2009. He was an associate professor in NAIST from 2009 to 2013. He is now an associate professor in Nagoya University, Japan since 2013. He is currently focusing on computer architecture and network related researches with low power consumption and high dependability techniques. He is a member of IEEE, IPSJ, and IEICE.

Kazutoshi Kobayashi received his B.E., M.E. and Ph.D. in Electronic Engineering from Kyoto University, Japan in 1991, 1993, 1999, respectively. Starting as an Assistant Professor in 1993, he was promoted to associate professor in the Graduate School of Informatics, Kyoto University, and stayed in that position until 2009. For two years during this time, he acted as associate professor of VLSI Design and Education Center (VDEC) at the University of Tokyo. Since 2009, he has been a professor at Kyoto Institute of Technology. While in the past he focused on reconfigurable architectures utilizing device variations, his current research interest is in improving the reliability (Soft Errors and Bias Temperature Instability) of current and future VLSIs. He was the recipient of the IEICE best paper award in 2009.

Hiroyuki Kanbara received his B.E. and M.E. degrees in electronic engineering from Kyoto University, Kyoto, Japan in 1987 and 1989, respectively and Ph.D degree in Informatics from Kyoto University in 2001. In 1989 he joined ASTEM RI (Advanced Scientific Technology and Management Research Institute of KYOTO). He is currently engaged in the research of embedded system design methodology and system prototyping with reconfigurable device.

Hiroyuki Ochi received the B.E., M.E., and Ph.D. degrees from Kyoto University in 1989, 1991, and 1994, respectively, all in Engineering. From 1994 to 2004, he was an Associate Professor with Hiroshima City University, and from 2004 to 2013, he was an Associate Professor with Kyoto University. In 2013, he joined Ritsumeikan University as a Professor. He is a member of IPSJ, IEEE, and ACM.

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

Kazutoshi Wakabayashi received his B.E. and M.E. degrees and Ph.D. from the University of Tokyo and he was a visiting researcher at Stanford University during 1993 and 1994. He joined NEC Corporation in Kawasaki Japan in 1986 and he is currently a Senior Principal Researcher of Central Research Labs. and senior expert at Embedded System Solution Div. at NEC Corporation. He has been engaged in the research and development of VLSI CAD systems. He served on executive committee or organizing committee of some international conference including; ASP-DAC'09 General Chair, CODES+ISSS'09 Co-Technical Program Chair, EC of ICCAD and DAC. Also, he has served on the program committees for several conferences including: DAC, ICCAD, DATE, ASP-DAC, ISSS, SASIMI, etc. Also, he has served as a general chair, a secretary, and a Technical Program Committee member for a number of Japanese conferences. IEICE and IPSJ. He received the Yamazaki-Teiichi Prize in 2004, and the IPSJ Convention Award in 1988, Sakai Kinen Special Award in 2001, and the NEC Distinguished Contribution Award in 1993 for his logic synthesis system and in 1999 for his formal verification, and in 2006 for his High Level Synthesis. His C-based Synthesis and Verification tool suite called "CybeWorkBench" received a Grand prize of "LSI of the Year 2003" and "LSI of the Year 2007".

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

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

Hidetoshi Onodera received the B.E., and M.E., and Dr. Eng. degrees in Electronic Engineering, all from Kyoto University, Kyoto, Japan. He joined the Department of Electronics, Kyoto University, in 1983, and currently a Professor in the Department of Communications and Computer Engineering, Graduate School of Informatics, Kyoto University. His research interests include design technologies for Digital, Analog, and RF LSIs, with particular emphasis on low-power design, design for manufacturability, and design for dependability. Dr. Onodera served as the Program Chair and General Chair of ICCAD and ASP-DAC. He was the Chairman of the IPSJ SIG-SLDM (System LSI Design Methodology), the IEICE Technical Group on VLSI Design Technologies, the IEEE SSCS Kansai Chapter, and the IEEE CASS Kansai Chapter. He is currently the Chairman of IEEE Kansai Section. He served as the Editor-in-Chief of IEICE Transactions on Electronics and IPSJ Transactions on System LSI Design methodology.