C-06

A Survival Architecture for Reconfigurable SoC

Gilyoung Kang, Sungchul Yoon, Gunbae Kim, Sungho Kang
Department of Electrical and Electronic Engineering, Yonsei University
shkang@yonsei.ac.kr

***Abstract* — The more the technological level is developed, the more fault-tolerance becomes important out of design factors for implementation of SoC (System-on Chip). Among many ways of implementing hardware for fault-tolerance, the remarkable one is using the reconfigurable hardware. But most of existing architectures must have a module for diagnosis, repair and control. So, in this paper, a new survival architecture for reconfigurable SoC is proposed. This architecture uses a self-checking module and a self-repairing module. Since it detects the faults during the operations, the test mode is not necessary. Therefore the new architecture is more survivable than the previous approaches.**

Keywords* — *Survival, Embryonics, Locality, Self-checking, Self-repairing.

1. Introduction

As the process technology goes to deep sub-micron (DSM), the complexity of System on Chip (SoC) has more increased continuously. As a result of this complexity, it is almost impossible to find transient or soft faults that cannot be detected in the manufacturing stage. They threaten the reliability of the SoC or the system. Considering the fact that the importance of fault-tolerance in designing a SoC chip increases gradually, it is important to find a solution for the problem. The term, *fault-tolerance*, is defined as the ability to maintain the function in the faulty cell under execution. Namely, the above problem can be solved by the fault-tolerant hardware.

Generally, the fault-tolerant hardware assumes the form of a reconfigurable architecture. An example of reconfigurable hardwares is a field programmable gate array (FPGA). FPGA is usually used in the case of implementing a part of very large scale integrated circuit (VLSI) without the high cost of developing the optimized architecture. But, recently, it is used to provide survival hardware which is inspired from the biology, by means of changing its configuration to fix the faulty cell.

The use of FPGA for survival of hardware is based on the evolution algorithm (EA). The change of the configuration is obtained according to the processes of EA. This is known for evolutionary hardware (EHW) which is good for constituting survival architecture of SoC [1].

Most EHWs for survival hardware have to be provided with the ways to detect faults and to locate. Though these components are necessary, they operate in the test mode when the configuration, the data path and the functions are different from the original ones.

Generally, a specific additional hardware for this purpose is used, such as the design for test (DFT) methodologies like built-in self test (BIST). If there is a fault in EHW, the faulty cell is excluded from the new one. But the normal operation must be stopped for repair and the processes in the test mode takes much time. In order to derive a more efficient survival methodology, a new survival architecture for the reconfigurable SoCs is proposed.

The paper is organized as follows. In Section 2 the previous methods approaching the survival hardware is introduced, while a new architecture for survival SoCs is described in detail in Section 3. In Section 4 the performance evaluation between the proposed one and the previous ones are presented, while the conclusions are drawn in Section 5.

2. Previous works

There are a lot of architectures for the reconfigurable hardwares. The reconfigurable hardwares are similar to programmable hardwares. Most of them are simply on the level of multi-cellular architecture. These methods use a lot of functional cells (FCs) with connections between themselves. But if high reliability of the system is required, it is necessary to design an additional algorithm or hardware. Therefore a survival hardware should be used as a solution at the initial phase of the design.

For hardware survival, the attempts to research the reconfigurable architecture inspiring the biology have increased. One of the reconfigurable architectures for fault-tolerant is the Embryonics approach for self-repairing and self-replicating hardwares [2]. The Embryonics approach proposes the fault-tolerant processor array inspired by the ontogenetic development of the multi-cellular organism [3]. Figure 1 shows its behavior with the simplest reconfiguration strategy, row elimination, which removes an entire row when one of its members has a fault.

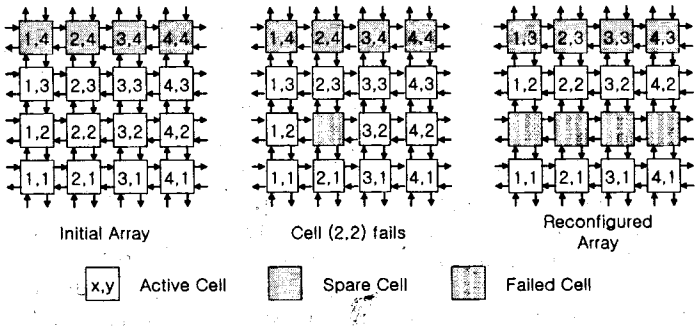

Figure 1. An example of the Embryonic array reconfiguration

In the Embryonics approach, if an FC in an array is found to be faulty, the row containing the faulty cell is replaced by a row of the spare cells (SCs). All the SCs in the new row are

reconfigured to function as the cells in the replaced row. But the disadvantage of this approach is that the row elimination also forces the non-faulty FCs in the replaced row to move to the spare row and to transfer their functions to the SCs. Since all SCs are in a single row, a single faulty cell in a row makes use of all the SCs.

To reduce above wastes, a modified Embryonics approach is proposed in [4]. In this way, a faulty cell is repaired by using one SC. But its process is more complicated than the original method. In m by n arrays, if a fault is detected in *(a, b)* cell, the traditional method accesses n cells first, and $n \times (m - a)$ cells in the second place. But the total number of accessing cells can be reduced to $(m - a + 1)$ as shown in Figure 2.

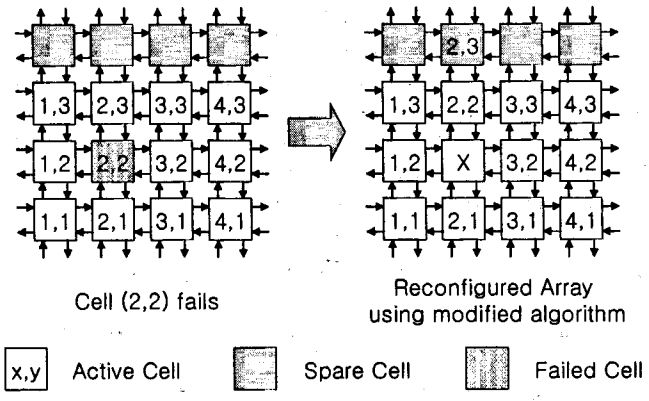

Figure 2. An example of the modified Embryonics approach

This modified approach can repair more faulty cells by saving the SCs and the wasted cells. However, this approach needs a more complicate routing method than the original method. And there must be the central control unit that dominates whole repair. This is not the original concept of Embryonics with high locality, which is the degree that indicates how many cells are needed to adjust.

To develop the reconfigurable architecture without an external control module, there are many researches for self-monitoring and self-repairing techniques are in progress [5]. These methods are useful for guaranteeing fault-tolerance of the architecture.

3. A new architecture for reconfigurable SoC

3-1. Overall architecture

Figure 3 shows a new architecture for reconfigurable SoC based on the modified Embryonics and the human immune system. All FCs in Figure 3 are the same. The SC is identical to the FC. The routing cell (RC) is similar to the FC but it is modified by eliminating its several internal blocks. It is only used for connecting the reconfigured SC and faulty FC.

An FC has four cells on the four sides, two SCs and two RCs. Without elimination of row of cells, an FC can be replaced by one of two SCs to adjust. And all cells are programmable with imitating the biological organisms; their functions are determined by the genes that are information of each cell. The behaviors of the genes are emulated in the FCs and the SCs of the proposed architecture.

The FC is able to perform a few basic logic operations on a 2-bit data. Also a few basic arithmetic units are able to be members of the FC. Each cell is set with its gene which indicates the functions to be carried out, the connections between the active FCs and the results of self-checking, monitored simultaneously with its operation. That is, the changing of the genes is capable of changing the function in the same FC.

In this paper the interconnection faults between the cells are not considered. It is assumed that the interconnection faults hardly happen comparing to the frequency of the fault in an FC. But it is impossible to design a reconfigurable architecture without repairing the interconnection faults. Therefore the interconnection faults are not regarded in the proposed architecture [6].

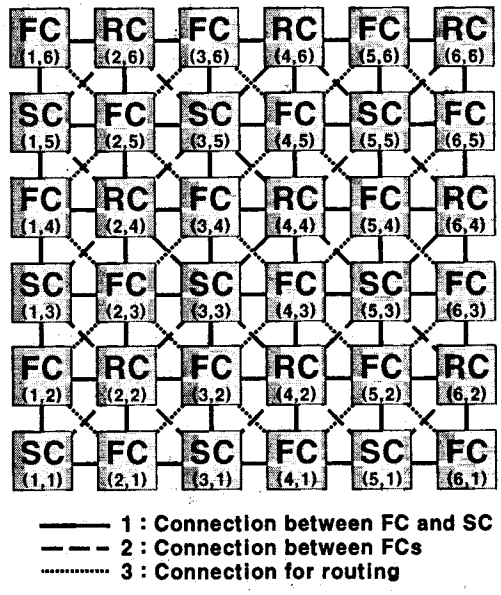

Figure 3. Proposed survival architecture

The FC and the SC have a self-checking module (SCM) for monitoring their survival and a self-repairing module (SRM) for repairing. The SCM checks whether the FC has a fault or not in real time, so that it broadcasts its survival information to four directions. The SRM in the SC receives the survival information of the adjacent FC and replaces the faulty cell with the SC detecting the faults. This replacement is inspired from the biology in which an individual evolves by means of cloning himself so that the dominant one survives while the recessive one dies, namely *the law of the survival of the fittest.* When both SCs receive a fault notification, the priority on broadcasting directions is required. For instance, if the FC in (3, 4) is faulty, the SCs in (3, 3) and (3, 5) are sensitive to the signal of self-checking and ready to substitute for the faulty FC. And if both SCs change their configurations, the whole configuration is collapsed. So there is a priority on repairing. The north cell in (3, 5) is prior to the other SC in (3, 3). Also if two adjacent cells which is in (3, 2) and in (4, 3) are faulty, the case is extremely rare, the cells in (3, 3) and (5, 3) are used for repairing the problem cells.

3-2. Architecture of Basic Cell – FC, SC and RC

Figure 4 is an FC. It has logic blocks composed of a few function modules described in previous section. Generally, conventional reconfigurable hardware elements such as PLD (Programmable Logic Device) and FPGA (Field Programmable Gate Array) have the logic blocks, the memory (or several registers) and the elements for routing used in repairing cells. The logic blocks consist of several simple functions whose sizes are very small [7].

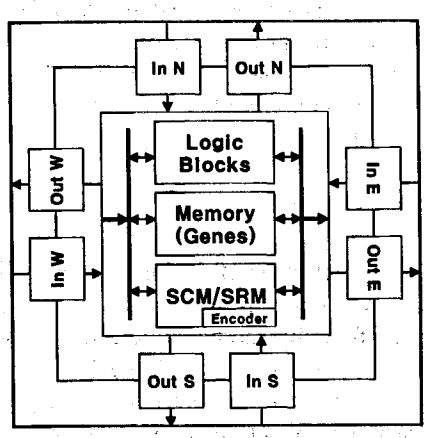

Figure 4. An architecture of the FC and the SC

The memory keeps the genes of the FC; that is all information of the FC. The genes are composed of several kinds of flags inspired from the organic base. The first part of the genes is configuration flags that determine which function blocks are needed in the FC. The second part is flags for inputs and outputs. The third part is the flag which stores the result. This result is continuously updated and broadcasted to the adjacent function cell by means of encoding the independent codes, such as 00, 01, 10 and 11 in normal state. The fourth part is used for comparing the flags received from the faulty cell with correct results and for repairing the cell. The last part is prepared for routing on reconfigured architecture; a few cells are implemented as a path between the FCs in optimized architecture. In this case, the path is easily decided so that the routing schematic is also simple. For example, the routing schematics can be realized using the complex switches. In this paper, the routing is accomplished using the RCs which have the same architecture except basic logic blocks. Also the size of the many in RC is the half of that in FC.

An SCM is self-checking module. The SCM can be implemented by two ways. One way is a hardware-oriented method. Like Built-in Self Test (BIST), this method uses hardware modules which check the process of the FC. Generally, the signature of the results is compared with good one, and then determined whether there is a matter in the FC or not. For the memory elements, for example, there is a Built-in Current Sensor (BICS) [8]. Although there are various BICS architectures [9] [10], for the small-scale memory, the architecture in [11] is known as the best one. The other way is a software-oriented method. This method does not need a special hardware; only memory where the good results are stored. But the demerit of this method is that whole processes are active with the embedded core. In this paper, the former is adopted to obtain high locality which is measured by counting the numbers of accessing cells for cloning and of changing the path of the cells.

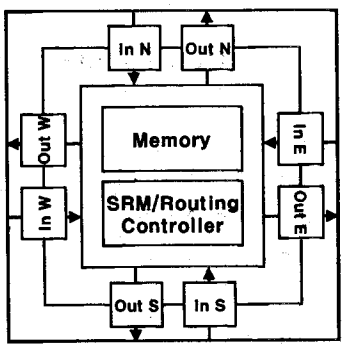

Figure 5. An architecture of the RC

The SCM has an encoder which generates the code from the signal notifying the existence and nonexistence of abnormality. If there is no problem, the encoder makes the four codes according to the directions. Since the north FC, the normality is confirmed by the code 11, similarly for the south FC, by the code 00. The RC in the east receives the code 10 and in the west receives the code 01. Since a fault makes improper results, it cannot generate all proper codes for all directions. And then, the fault is detected in the adjacent SCs and RCs. It is easy to detect the wrong codes by comparing the eight bits in parallel.

3-3. Self-Checking and Self-Repairing Module

Figure 6 is an example of the architecture of an SCM for 2-bits input and 1-bit output with the data path. It has four registers for storing its good result. The 2-bits of input are used in selecting the right output in registers. If the input is 01, the value of R0001 is selected to be driven to the comparator. The executed output of the function module is sent to the other input of the comparator, a 2-bit XNOR logic. The output of the comparator, 1, enables the encoder to generate the codes for broadcasting to four directions, NSEW. Each code in the output is independent; a chain of 8-bit codes is 00110110.

The registers from R0100 to R1101 are flags which indicate what a function is performed in this cell. The function module contains elementary Boolean functions such as 2-input AND, OR, NAND and NOR. The members of the module are capable of expanding its members to basic arithmetic operations like ADD and SUB within 10 flags. Surely a 1-bit input function like INV can be included in the function members. R1110 is fit for representing the availability of the FC and the SC, namely life or death. If R1110 has 0, the cell cannot be used for constituting the survival SoC. If there is no way to maintain its survival which is due to all 0 in R1110s of the SCs, the remedy of the local is performed in the system level. R1111 is used for identifying the cell as an FC or an SC. This is necessary for the repaired SC in order to prevent receiving the result of the self-checking module.

The values in registers are set in the first stage in which FCs are used to configure the survival SoC. The genes are easily obtained by initial downloading of the data from the external core, because the amount of the size of its results is not big.

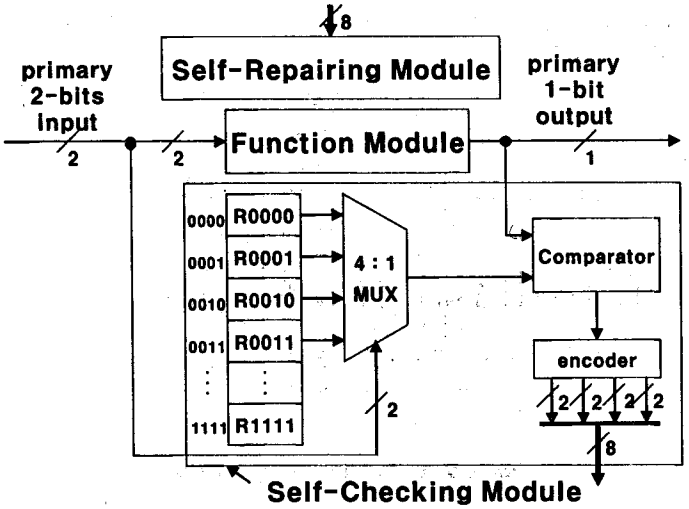

Figure 6. An Example of architecture of 2-bits input self-checking module with its data path

Figure 7 is the architecture of an SRM for 2-bit input and 1-bit output with its data path. Its input has four codes of 2-bit from 4-directions, NSEW. The code, 00110110 is compared with this input in parallel. Each code products 1-bit results representing a fault of its original FC. In normal mode, all the values are 0 and its primary output is also 0 so that the additional blocks are disabled. When 1 occurrs in the results, the analyzer catches the position of the faulty cell and makes the reproduction of the information to be downloaded into the registers. The downloading controller sets a data path valid between the faulty cell and the selected SC. Surely the priority of the SCs is checked to determine which SC is adequate. Repeatedly, since a fault detected in two SCs simultaneously, an SC should have the order that which cell is to be repaired first.

After the repair is completed, a new routing configuration is needed. It can be obtained using the characteristic of the RC. An SRM in an RC is used to inform the faulty cells and the SCs for repair. Accordingly the routing result is obtained while the repair of the faulty cell is processed

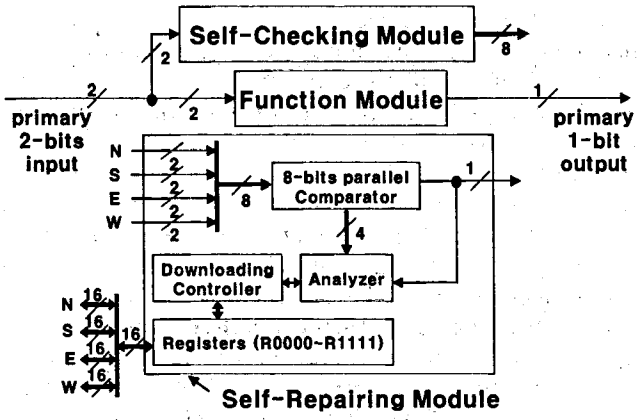

Figure 7. An Example of architecture of 2-bits input self-repairing module with its data path

4. Performance Evaluation

In the multi-cellular architectures of [3] and [4], since all FCs are the same, the probability, p that a fault occurs in each cell, is also identical. And the reconfigurable architecture must be able to be connected to any directions. Since FCs are a form of a regular quadrilateral in general, the size of the architecture is $(n \times n)$ where n is the size of a row of the rectangular. The number of faulty cells is a stochastic variable and the occurrence of a fault in each cell is independent. Therefore the probability distribution that there are r faulty cells in the array, P_n(# of faulty cells = r), is a binomial distribution. Let us assume the SCs are fault-free. The probability distribution is represented as follows.

$$P_n(\#\text{ of faulty cells} = r) = {}_{n\cdot(n-1)}C_r \times p^r \times (1-p)^{n\cdot(n-1)\cdot r}$$

Because of a characteristic of the binomial distribution, the average number of faulty cells in the array is $n \times (n-1) \times p$. So, in most of reconfigurable architectures, the value of n is approximately $\sqrt{1/p}$ when n is very large. For example, if the probability is 0.01, the size of the array is determined to (10×10). Because the architectures of [3] and [4] assume that there is a fault in an array, if there are 2 faults on the same column, the architecture becomes useless.

The new architecture is designed for the robust survival of the reconfigurable hardware for both single fault and multi-faults. And this architecture does not require the test mode which is necessary for checking the existence of faults. In some applications, like deep space explorations, replacement of the systems with self-acting ones in the nuclear silo (or reactor) and high technology army systems, the high reliability must be provided for the system. It assures that the faults are detected during the operation of FCs. The repair is performed with the operation. Repeatedly, high survivability is obtained because of ability to repair the faulty cell more rapidly.

Another advantage of the new architecture is scalability. The configuration algorithm of the system, the test methodology and repairing method are simple, since they do not depend upon its size. On the other hand, the existing architectures should have an additional hardware. Therefore the new architecture can expand its size very easily for its simplicity.

The proposed architecture has an RC and an SC per two FCs. We assume that there is no configuration loss. In the common RC, like a switch complex, the ratio of the architecture overhead is about half a single SC. Since the SC is identical to the FC, the efficiency of the proposed survival architecture is about 60%. The efficiency is defined as the ratio of the FCs' size to the whole size including two FCs, an SC, and an RC. This figure, 60%, is not advantageous when compared with that of the architecture of [3] and [4]. Nonetheless, the new architecture is efficient in terms of the survival of the hardware.

The performance evaluations of the proposed architecture are arranged in Table 1.

Table 1. The evaluation of the new application

Performance Evaluation		[3]	[4]	The proposed Architecture
Hardware Efficiency		$\frac{n-1}{n}$	$\frac{n-1}{n}$	about 60%
Survivability	Fault-tolerance	Low	Medium	High
	Test operation	Off-line	Off-line	On-line
	Scalable	Difficult	Difficult	Easy

5. Conclusions

As the complex SoC architectures are developed, the survival of hardware becomes more important. The new survival architecture for reconfigurable SoC is developed to complement the existing Embryonics approaches. The new architecture of the FC with an SCM and an SRM is proposed. It can check the fault of the FC and repair it simultaneously with its main operation. Although there is a disadvantage of the low hardware efficiency, the high survivability of hardware can be guaranteed by the new architecture.

REFERENCES

[1] Yao, X.; Higuchi, T. Systems, "Promises and challenges of evolvable hardware", Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , Volume: 29 Issue: 1 , Feb. 1999 Page(s): 87 -97

[2] Mange, D., Sipper, M., Stauffer, A., Tempesti, G. "Toward Self-Repairing and Self- Replicating Hardware: The Embryonics Approach." Evolvable Hardware, 2000. Proceedings. The Second NASA/DoD Workshop on , 2000 Page(s): 205 -214

[3] Jackson, A.H., Tyrrell, A.M., "Asynchronous embryonics." Evolvable Hardware, 2001. Proceedings. The Third NASA/DoD Workshop on, 2001 Page(s): 201 -210

[4] Tempesti, G., Mange, D., Stauffer, A., "Embryonics: multi-cellular and multi-molecular digital systems." Evolutionary Hardware Systems (Ref. No. 1999/033), IEE Half-day Colloquium on , 1999 Page(s): 1/1 -1/4

[5] Lee, C.H., Perkowski, M.A., Hall, D.V., Jun, D.S., "Self-Repairable EPLDs: Design, Self-Repair and Evaluation Methodology," Evolutionary Computation, 2001. Proceedings of the 2001 Congress on , Volume: 1 , 2001 Page(s): 616 -623 vol. 1

[6] Hanchek, F., Dutt, S., "Methodologies for tolerating cell and interconnect faults in FPGAs", Computers, IEEE Transactions on , Volume: 47 Issue: 1 , Jan. 1998. Page(s): 15 -33

[7] Lala, P.K., Kumar, B.K., "An Architecture for Self-Healing Digital Systems." On-Line Testing Workshop, 2002. Proceedings of the Eighth IEEE International , 2002 Page(s): 3 -7

[8] Pontarelli, S., Cardarilli, G.C., Leandri, A., Ottavi, M., Re, M.; Salsano, A., "A Self-Checking Cell Logic Block for Fault Tolerant FPGAs." Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 4 , 2002 Page(s): 477 -480

[9] Calin, T., Vargas, F.L., Nicolaidis, M., "Upset-tolerant CMOS SRAM using current monitoring: prototype and test experiments," Test Conference, 1995. Proceedings., International , 1995 Page(s): 45 -53

[10] J. Tang, K. Lee, B. Liu, "A practical current sensing technique for IDDQ testing," IEEE Transaction on VLSI, pp. 303-310, June 1995

[11] Vargas, F.; Nicolaidis, M., "SEU-tolerant SRAM design based on current monitoring," Fault-Tolerant Computing, 1994. FTCS-24. Digest of Papers., Twenty-Fourth International Symposium on , 1994 Page(s): 106 -115