International Journal of Scientific Research and Management Studies (IJSRMS)
ISSN: 2349-3771 **Volume 1 Issue 1, pg: 33-41**

AUTOMATIC FAULT TOLERANCE USING SELF ADAPTIVE SYSTEM

R. Logesh[1] and S. T. Sadish Kumar[2]
[1]Department of Electronics and communication Engineering, Nandha Engineering College, Erode, Tamilnadu, India
[2]Associate Professor, Nandha Engineering College, Erode, Tamilnadu, India

ABSTRACT

Fault tolerance (F-T) was an important issue in electronic device. However, such system was still impractical in many cases, particularly due to the complex rerouting process that follows cell replacement. In this project, proposed a dynamic reconfiguration capabilities system, therefore, a faulty module could be replaced and the whole system's functions and connections were maintained by simply assigning the same encoded data to a spare (stem) module. The proposed system reroutes data flow. To detect permanent errors at runtime, a novel in-line test (ILT) method using spare wires and a test pattern generator wass proposed. In addition, an improved syndrome storing-based detection (SSD) method was presented. In the presence of permanent errors, the probability of correct transmission in the proposed systems was improved by up to 140% over the standalone hamming code. Furthermore, this method achieves up to 38% area, 64% energy, and 61% latency improvements.

KEYWORDS: *Fault tolerance, ILT, SSD, Genome, stem cell, work cell.*

I. INTRODUCTION

In the development of a critical application, designers have to consider the possibility of including F-T structures in the circuit. Nowadays these techniques have to be inserted in the design descriptions manually, because no CAD tools are available for this task. The relatively small number of Fault Tolerant applications did not make attractive for CAD vendors the development of tools specific to the design of fault-tolerant circuits. The insertion of Fault Tolerant techniques requires large design efforts that affect negatively design productivity and time-to market. The design of a fault tolerant circuit is usually accomplished by introducing fault-tolerant structures in a previously designed circuit. Within the current HDL-based methodologies, the insertion of fault-tolerant structures is usually performed by manually modifying the HDL code in order to insert hardware redundancy, information redundancy or time redundancy at critical points in the circuit. Then, the modified, fault-tolerant design obtained follows a similar design flow consisting in automatic logic synthesis and place and route.

F-T techniques rely on the concept of redundancy. Redundancy is the addition of resources, time or information beyond what is needed for normal system operation. Redundancy can take several forms [1].

1. Hardware redundancy is the addition of extra hardware, in order to detect or tolerate faults.
2. Information redundancy is the addition of extra information to implement a given function.
3. Time redundancy uses additional time to perform system functions.

There are three basic forms of hardware redundancy: passive, active and hybrid. Passive techniques use the concept of fault masking to hide the occurrence of faults and prevent the faults from resulting on errors. Active techniques use fault detection, fault location and fault recovery to achieve FT. The active approach achieves FT by detecting the existence of faults and performing some action in order

to get the circuit into correct operation in a satisfactory length of time. Hybrid techniques combine the features of the previous approaches. Fault masking is used in hybrid systems to prevent erroneous results from being generated. Fault detection, location and recovery are also used in the hybrid approaches to improve FT by removing faulty hardware and replacing it by spares (redundant extra elements in circuit, not needed until another element in circuit becomes faulty). Passive hardware redundancy uses voting mechanisms to mask the occurrence of faults. Most passive approaches are developed around the concept of majority voting. For example, Triple Modular Redundancy (TMR) performs a triplication of hardware modules and a majority vote to determine the output of a circuit. If one of the modules becomes faulty, the two remaining fault-free modules mask the results of the failed element when the vote is performed. In a more general application, N-Modular Redundancy consists in the same concept but with n-replication of the hardware modules.

Active Hardware Redundancy techniques, as previously said, attempt to achieve FT by fault detection, fault location and fault recovery. No fault masking is achieved. Consequently, active approaches are most common in applications that can tolerate temporary, erroneous results as long as the system is able to reconfigure and regain its operational status in a satisfactory length of time. For example, Duplication with Comparison is a typical technique to achieve FT with active hardware redundancy. The basic concept of this technique is to develop two identical pieces of hardware that perform the same computations in parallel, and compare the results of those computations. In the event of a disagreement, an error signal is generated. In its most basic form, the duplication concept can only detect the existence of faults, not tolerate them, because there is no method for determining which of the two modules is faulty. Other techniques for active hardware redundancy are Standby Sparing, where each module has a fault detection capability and only one is operational at a particular time if the on-line module becomes faulty, then it is removed from operation and replaced with a spare module, [1].

One of the representative methods of self-repairing digital systems is the MUXTREE method [4], [5], [12]. In this method, a digital circuit is converted into an array of MUXTREE cells and the initial connection information among the MUXTREEs is encoded as a gene in each MUXTREE cell. The system develops very large-scale integrated circuits capable of self-repair and self-replication. An adaptive hardware system with dynamic routing, reconfiguration, and on-chip reprogramming was used as a self-repairing system [6], [20]. Another example of a self-repairing system is the self-healing digital system based on a lookup table (LUT) and designed by Lala, it can also replace a faulty cell with a spare cell [7], [17]. Another self-repairing system called Unitronics took inspiration from the prokaryotes [8]–[11]. It is an on-line self-repairing system that has a low overhead by introducing a new method for configuration memory reduction. Other endocrine-inspired systems employing a new data processing method on the basis of endocrine signalling were developed for fault tolerance [21], [22].

This paper is organized as follows. Section 2 presents the features of the existing system inspired by endocrine cellular communication. Section 2.1 overview of the proposed system, Section 2.2 describes the new dynamic reconfiguration capabilities. Section 2.3 shows the control mechanism that recovers the faulty module in the functional layer without collision. In Section 2.4, the proposed system is compared with existing methods from the points of view of overhead and scalability. In Section 2.5, the proposed system is implemented in a digital platform and the results of several experiments are illustrated. Section 3 summarizes the proposed system and describes future studies.

II. EXISTING SELF REPAIRING MECHANISM

Among the various methods of cell-to-cell communication in our body, endocrine cellular communication is particularly interesting. Basically, an endocrine cell releases a hormone from the signalling endocrine cell, and the hormone flows through the blood vessel until it binds to the target cell [23]. Although the blood contains various hormones, only the receptor on the target cell receives the selected hormone. The special method of endocrine cellular communication that inspired this paper is based on a specific endocrine cell that secretes a hormone only if it receives another hormone from another endocrine cell.

The hormone from the anterior pituitary stimulates the endocrine cell in the peripheral endocrine gland to secrete the peripheral gland hormone. As a concrete example, corticotropin enters the blood

and travels to the adrenal cortex where it stimulates the release of cortisol [24]. Thus, the blood vessel delivers the hormones between cells. When a functioning endocrine cell dies through apoptosis, the special endocrine cellular communication maintains hormone delivery by differentiating an SC into a cell having the same genome part as that of the dead cell. Thus, in addition to its own functioning, the genome in the endocrine cell has the information about connections between cells. The inspiration we obtained from the biological endocrine system lies in the efficient and flexible communication mechanism between endocrine cells. In the endocrine system, the information between endocrine cells is exchanged via the hormones, and this forms a complex communication network.

The structure of this network is flexible and easily changed by choosing and adjusting the hormones to be used for secretion and reception in each endocrine cell. Even if an endocrine cell dies through apoptosis, a new endocrine cell having the same function of the dead cell is produced by differentiation and the overall communication network is recovered. Inspired from this feature of the endocrine system, the proposed system has been developed. In particular, to implement an electronic module that resembles the feature of an endocrine cell receiving and secreting the hormones selectively, multiplexers (MUXs) for selection of inputs and demultiplexers (DEMUXs) for selection of outputs have been introduced.

It is also noted that the hormone which is selectively received and secreted by the endocrine cell is determined by the genome that is expressed for differentiation of an SC. We have thus designed our system such that the selection bits of MUXs and DEMUXs in addition to LUT in the electronic module are determined by the stored genome memory. In addition, to resemble the cell replacement of dead endocrine cells by newly differentiated SCs, four spare electronic modules are arranged around each working electronic module for fault recovery. Moreover, in the endocrine system, the hormones do not flow through a special path between endocrine cells but rather they are in contact with endocrine cells through a blood vessel. In the proposed system, explicit connections between a working module and spare modules are designed, so that the working module can be rerouted and connected to a spare module when the working module fails and is replaced with the spare module.

Finally, the endocrine cellular communication is recovered by resuming the reception and secretion of the same hormones, which is realized by SC differentiation controlled by the expression of the same genome as the dead endocrine cell. Similarly, our proposed system can recover the connection between modules using the same genome (stored in the spare module) as the faulty module. In these ways, the proposed system can maintain the connection between modules as well as the function of a module by module replacement.

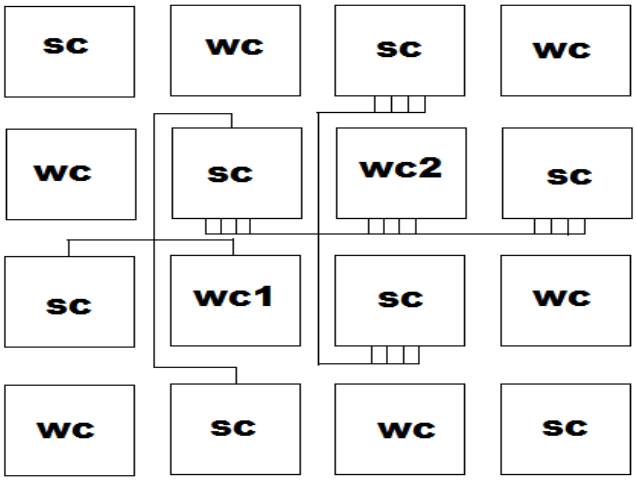

Figure 1. Artificial Endocrine Routing Architecture.

2.1. Overview of the Proposed Digital System

The fault detection unit in the artificial endocrine cell monitors the integrity of the genome. According to [25], nearly 90% of system crashes in a workstation are caused by soft memory errors rather than hardware failure. Moreover, the space applications into which our system can be embedded are

exposed to radiation constantly in the space. Therefore, the radiation causes a voltage spike and switches the memory state between "0" and "1" as a transient error [26]. In the proposed system, the genome is the memory value in each cell, it is highly important for the operation of the system. Hence, we focus on the fault that occurs in the genome.

The proposed fault detection unit [Figure 1] can detect and correct memory faults. S denotes the original genome and S_ indicates the genome of the inverted bits. On the right side, there are four even parity bits for each line. The left one of the even parity bit on line S is the even parity bit for the genome of an odd order (black-colored) on line S, and the right one of the even parity bit on line S is the even parity bit for the genome of odd order (gray-colored) on line S. The even parity bits on line S_ are determined in the same way. The third line shows the sum of S and S_. The fault detection unit [Fig. 2(a)] can be divided into two systems. The one in charge of black-colored bits and another one in charge of gray-colored bits can correct the error of one bit and detect up to three simultaneous errors. After integrating the two systems, it can correct the error of one bit and two bits, consisting of one bit with an odd order and another bit with an even order.

It also can detect up to three simultaneous errors. Figure 2 illustrates the overall flowchart of the fault detection unit, showing how it can distinguish a permanent memory fault from a transient memory fault. When a fault is detected, the system determines whether or not the fault occurs immediately after the genome recovery process. If a fault is detected consecutively in this manner, the fault is considered as a permanent fault because the genome does not change, even after genome recovery. The fault signal is then sent to the gene-control layer for replacement and an SC is differentiated for the faulty cell with the permanent fault. However, if it is not detected consecutively, it is a transient fault that can be recovered only by changing the data. If the location of the fault in the genome is identified, only faulty bits are corrected. Otherwise, the entire genome is replaced by a normal genome.

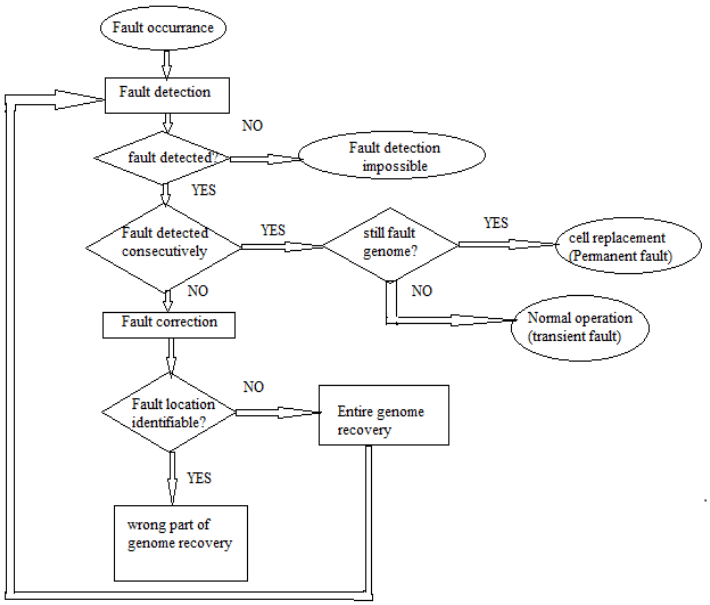

Figure 2. Fault detection and correction

2.2. Permanent-Error Correction

Permanent-error correction in on-chip links using spare wires is a two-step process. First, the permanent error must be detected; then, the link must be reconfigured to avoid transmit- ting over the faulty wire. The proposed adaptive link framework is shown in Figure 3 and consists of a transmitter, a link, and a receiver. The incoming -bit-wide data word is encoded in the transmitter to a codeword of width, which is transmitted through the link and decoded in the receiver. The decoder is responsible for correcting any errors and outputs the original –bit data word. A number of spare wires

are available. Reconfiguration units at the transmitter and receiver determine which of the lines carry data and which are left idle.

The reconfiguration control units pass reconfiguration information between the receiver and transmitter and synchronize reconfiguration. The error detection and reconfiguration central control unit detects permanent errors and initiates reconfiguration. The inputs to this unit depend on which detection method is used. For the SSD method, the syndrome and error vector from the decoder are needed. For the ILT method, test outputs from the spare wires under test are needed. The ILT method requires a test pattern generator (TPG) block and test inputs to produce test signals.

Apply techniques to permanent and intermittent errors in the link; the logic units are assumed to function correctly. Two methods are presented here, namely, the proposed ILT method and the improved SSD method.

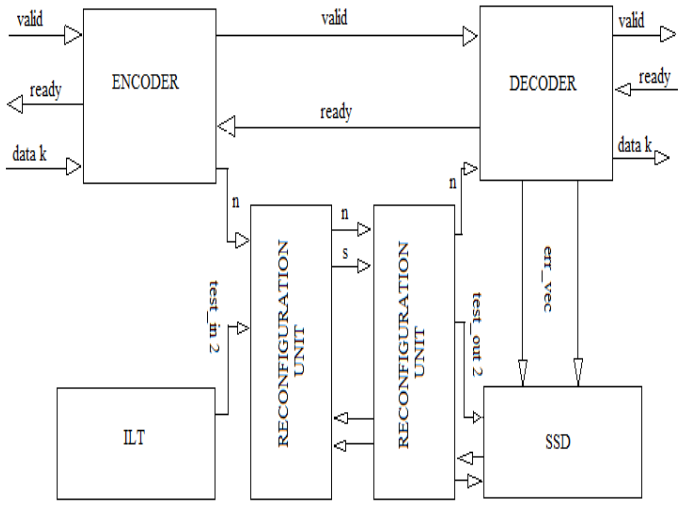

Figure 3. Reconfigurable Link System

2.3. ILT Method

The proposed ILT method sequentially routes data from each pair of adjacent wires to a set of available spare wires, allowing each pair to be tested for intermittent and permanent faults. This is achieved during normal operation, without interrupting data transmission, by making use of the reconfiguration system to be described in Section IV. To protect against runtime permanent errors, the ILT is run periodically, with a period that can be shortened to improve error resilience or increased for energy efficiency. In addition to this periodic testing, the ILT can be triggered when an error is detected beyond the error correction capability of the code protecting the link. The trigger can use a simple timer or be adaptively controlled by an upper protocol layer (e.g., application) to save energy during idle periods.

ILT Procedure: To begin the test, the ILT control unit reconfigures the link such that the first pair of wires is connected to the TPG (the data on those lines are rerouted to spare wires). The TPG issues a series of test patterns using the signal. The ILT control unit compares the received signal to a lookup table to determine if there is a permanent error in that pair of wires. The lookup table indicates which line(s) need(s) to be flagged as erroneous. Functional wires are reconfigured to carry data once again, and the process is repeated for each pair of wires (i.e., the test is shifted from wires 1 and 2 to wires 2 and 3, etc.). Note that, during each test, wires that were flagged as faulty are retested to prevent intermittent errors from wasting wire resources.

To provide even greater protection against permanent errors, the system is designed to take into account the number of spare wires when running a round of tests. The ILT control unit determines the number of remaining spares by checking the status of the last wire in the link. If one spare remains, the upper layer system can be alerted to the lack of spare resources, and the system only reroutes one wire at a time to that remaining spare.

Instead of the two-wire test, the system performs a single-wire test that can detect opens in the line but cannot detect a short between the wire under test and its neighbors. If a wire adjacent to the wire under test is disabled due to a previously detected error, the ILT unit will perform the two-wire test on that pair. If no spare wires remain, the system will periodically retest each disabled wire in an effort to recover from intermittent errors.

2.4. SSD Method

Syndrome decoding is a common technique for decoding linear block codes. The syndrome is calculated by matrix multiplication, where is a received code word vector of length (where is a transmitted code word and is an error vector, both of length), is the transpose of the parity-check matrix, and is a syndrome vector of length. The syndrome gives the minimum weight error vector index, so the error vector can easily be determined.

The correction is done by, eliminating the error from the received data word. The basic idea behind SSD is that the error syndrome of an error control code contains information about the errors of a received code word. If the syndromes of a number of consecutive received code words are the same, then it can be concluded that there is a permanent error in the link (limitations described momentarily).

The error location can be extracted from the syndrome using the normal decoding procedure. The effectiveness of this approach comes from the fact that it takes advantage of the code and decoder already present at the system [9]. If there are more errors in the link than the error correction code is capable of correcting, the syndrome will be decoded incorrectly, providing a wrong error location. An important design decision for the SSD method is to determine how many syndromes to consider before deciding that an error is permanent. Refer to this number of cycles as the observation period. In [9], this period was set to three transmissions.

Now present a more detailed analysis of the tradeoffs between and reliability and provide guidelines for selecting. If an intermittent error is misdiagnosed as a permanent error, a spare wire is consumed. In SSD, there is no method for recovering spare wires once they have been assigned, so the error observation period can result in wasted wire resources if set too short. On the other hand, too long an observation period may result in a large number of cycles before the error is detected or may even leave errors undetected. This is because the detection of stuck-at faults is data dependent; in order to be detected, the error must occur in all data words during the observation period.

For example, a stuck-at-1 fault can only be detected if all the data bits passing through that wire over cycles are 0. The upper limit for the number of cycles before the permanent error should be detected can be derived from the transient bit error rate (BER). Since a permanent error in a link may prohibit the detection and correction of a transient error, the number of cycles to detect a permanent fault should be much smaller than the mean time between transient errors.

2.5 Experimental Results

The simulated results of system with fault and system without fault is described below, based on the presence of faults the output of the system is varied according to the working MUX cell. The simulated output of the system with the fault is shown in from Figure 4 to Figure 7. The output of the work cell neighbouring to the faulty cell is displayed.

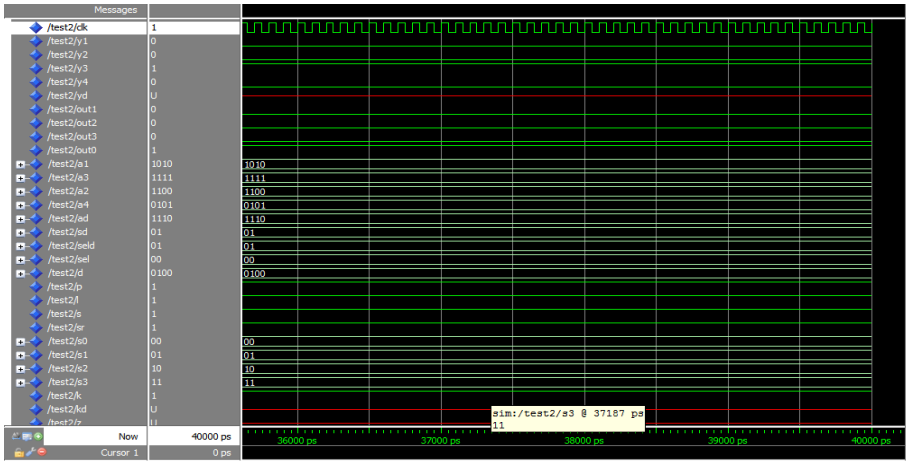

Figure 4. The output shows that after fault occurrence left spare cell act as the work cell.

The stem cell contains the same circuits as that of the working cell. An SC can be replaced by any of its four neighboring WCs for fault recovery. For example, the working cell goes faulty; the WC is replaced by the left side stem cell SC00.

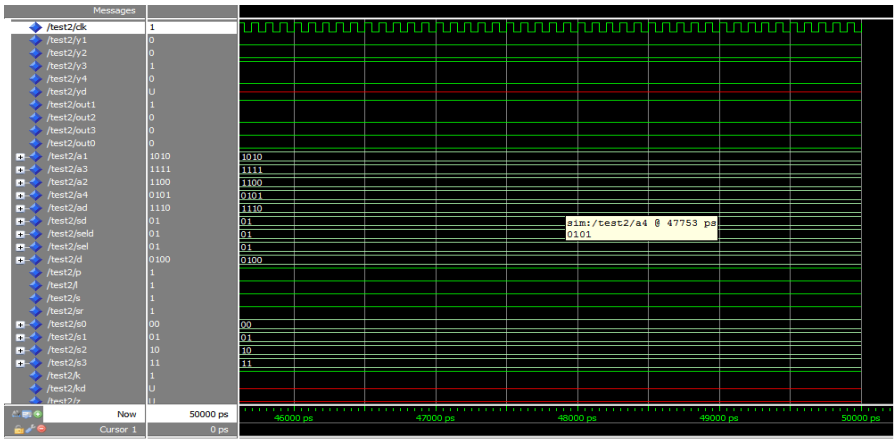

Figure 5. The output shows that after fault occurrence right spare cell act as the work cell.

The stem cell contains the same circuits as that of the working cell. An SC can be replaced by any of its four neighboring WCs for fault recovery. For example, the working cell goes faulty; the WC is replaced by the right side stem cell SC01.

The stem cell contains the same circuits as that of the working cell. An SC can be replaced by any of its four neighboring WCs for fault recovery. For example, the working cell goes faulty; the WC is replaced by the top side stem cell SC10.

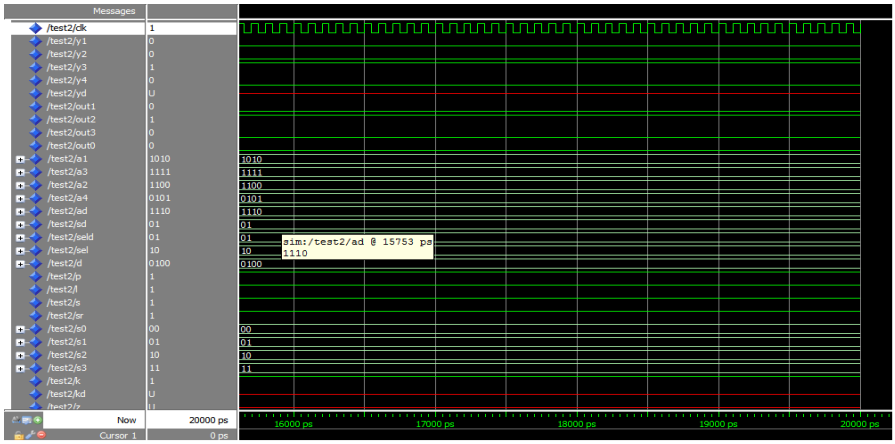

Figure 6. The output shows that after fault occurrence top spare cell act as the work cell.

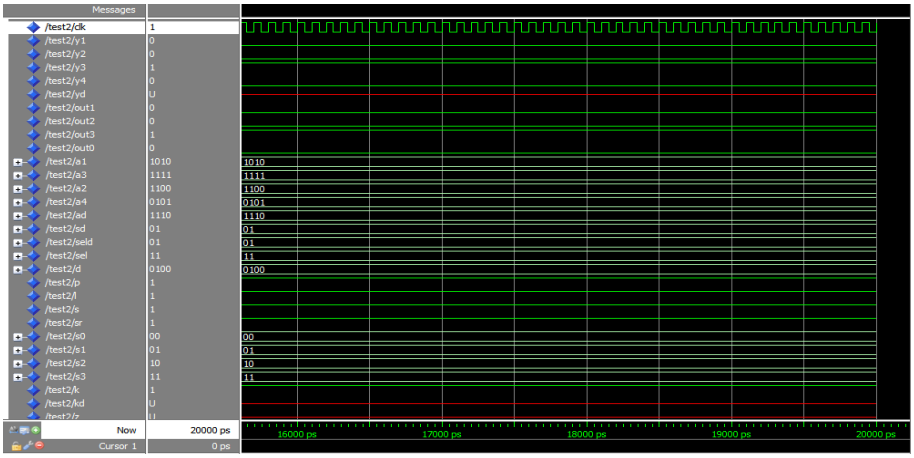

Figure 7. The output shows that after fault occurrence down spare cell act as the work cell.

III. CONCLUSIONS

In this project, new self-repairing digital system providing good scalability and fault coverage was proposed. It consists of a functional layer and a gene-control layer. The functional layer performs the function and the gene-control layer controls the functional layer so that proper assignment of SC for the replacement of a faulty cell is carried out. New architectures for the routing and the cell in the functional layer were developed and well organized such that rerouting between neighboring cells after the replacement of a faulty cell would be done by only replacing the faulty cell with an SC. Furthermore, due to the new architecture, the cells could be arranged in a flexible manner such that the WC could be expanded to any four directions, and could also be arranged densely such that SC could be replaced by any of four neighboring WCs for fault recovery without collision due to exact control in the gene-control layer. As a result, all these make the system efficient. The proposed system was compared with other major self-repair approaches and it was found that the proposed system has good fault coverage, low overhead, and no unutilized resources for fault recovery.

A complete reconfigurable system utilizing spare wires to replace erroneous wires and enabling reconfiguration without interfering with data transmission has been presented. Two methods for error detection have been evaluated, namely, rotating ILT and SSD. The results show that our approach provides tolerance against a number of permanent errors equal to the number of spare wires in the system. In addition, the presented case study shows that Hamming and BCH error tolerances decrease significantly in the presence of permanent errors, while the error tolerance of the adaptive systems is less affected. Indeed, the ILT detection method sees no degradation at all. For example, with one permanent and one transient error, the probability for correctly transmitted words is 30% higher in the

ILT system than with the reference Hamming code approach; with two permanent errors, the improvement increases to 140%.

REFERENCES

[1]. W. C. Carter, Mar. (1973) Fault-tolerant computing: An introduction and a viewpoint, IEEE Trans. Comput., vol. 22, no. 3, pp. 225–229.
[2]. C. Ortega and A. Tyrrell,May (1998) Design of a basic cell to construct embryonic arrays, IEE Proc. Comput. Digital Tech., vol. 145, no. 3, pp. 242–248.
[3]. X. Zhang, G. Dragffy, A. G. Pipe, N. Gunton,and Q. M. Zhu,Jun. (2003) A reconfigurable self-healing embryonic cell architecture, in Proc. ERSA, pp. 134–140.
[4]. D. Mange, E. Sanchez, A. Stauffer, G. Tempesti, P. Marchal, and C. Piguet, (1998) Embryonics: A new methodology for designing fieldprogrammable gate arrays with self-repair and self-replicating properties, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 6, no. 3, pp. 387–399.
[5]. D. Mange, S. Durand, E. Sanckez, A. Stauffer, G. Tempesti, P. Marchal, and C. Piguet,Apr.–May (1995) A new paradigm for developing digital systems based on a multi-cellular organization, in Proc. IEEE Int. Symp. Circuits Syst,vol. 3, pp. 2193–2196.
[6]. W. Barker, D. M. Halliday, Y. Thoma, E. Sanchez, G. Tempesti, and A. M. Tyrrell, Oct. (2007) Fault tolerance using dynamic reconfiguration on the POEtic tissue, IEEE Trans. Evol. Comput., vol. 11, no. 5, pp. 666–684.
[7]. P. K. Lala and B. K. Kumar, Oct. (2003) An architecture for self-healing digital systems, J. Electron. Testing: Theory Appl., vol. 19, no. 5, pp. 523– 535.
[8]. M. Samie, G. Dragffy, A. Popescu, T. Pipe, and C. Melhuish, Jul.–Aug. (2009) Prokaryotic bio-inspired model for embryonics, in Proc. NASA/ESA Conf. Adapt. Hardw. Syst., pp. 163–170.
[9]. M. Samie, G. Dragffy, A. Popescu, T. Pipe, and J. Kiely, Jul.–Aug. (2009) Prokaryotic bio-inspired system, in Proc. NASA/ESA Conf. Adapt. Hardw. Syst., pp. 171–178.