International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Amorphous Slack Methodology for Autonomous Fault Handling in Memory and Image Processing Applications

Talwade Yuvaraj[1], Sadhana Choudhary[2]

[1]Visvesvaraya Technological University, Lingrajappa Engineering College, Bidar-585401, India

[2]Visvesvaraya Technological University, Lingrajappa Engineering College,Bidar-585401, India

Abstract: ***The proposed amorphous fault handling methodology utilizes runtime redundancy in order to enhance the survivability of the design. Here we are proposing a fault isolation algorithm, this algorithm will be solicited upon fault detection, this fault detection scheme uses health metric of application, this is applicable if and only if signal to noise metric is known and also the application should not possess associated metric for recognition of faulty behavior. This is demonstrated by designing a memory controller, which is being implemented using a HDL designer and modelsim software and also we are designing a sobel edge detector and same is implemented using MATLAB.***

Keywords: Self adaptability, Error resilience, Fault isolation, Memory controller

1. Introduction

With introduction of Nano scale devices and with the introduction of 20nm CMOS device technology permanent faults and as well as degradation effects have become more prominent in resources and as well as in the interconnects and more over this problem becomes more unpredictable in harsh environment conditions and due to process level variability. Hence due these challenges faced the important subjects of interest are error resiliency and self-adaptability. In our approach we are proposing fault isolation algorithm for fault isolation purpose, which will evaluate all the processing elements, once faulty processing element is identified that processing element is discarded and its functionality is assigned to a healthy processing element.

For demonstrating the proposed algorithm we are designing a memory controller and evaluating it using the proposed algorithm. And it has been implemented using HDL designer and modelsim software. We are also designing sobel edge detector and isolating the noise from the given image using proposed algorithm. It is implemented using MATLAB

2. Methodology:
Amorphous Slack Approach

N, Ns, Input signal characteristics, QP
Output: Φ
1. Initialize Φ=[x x x … x]T, i=1
2. While ({k| k∈Φ, k=0} = ϕ) do
3. Designate PEs as checker(s) ;(N/2+1)≤s≤(N/2+Ns/2)
4. While (i ≤ N/2) do
5. Reconfigure AS(s) with the same functionality as PE si
6. Perform N-Modular Redundancy (NMR) majority
   Voting to identify at least one healthy AS,
   Φi ← 0 for PE which shows no discrepancy
   Then go to step-11, Φ ← x otherwise
7. i ← i+1
8. end while
9. Move the AS by updating N/2=N/2-Ns/2,
   Re-initialize i=1
10. end while
11. Use a healthy AS to check all other PEs

Once the fault is detected by observing the heath metric, fault isolation algorithm is activated. Initially Φ is a vector containing processing elements, which are all treated as faulty, among them the processing element which is having lower priority is taken as a reconfigurable checker element, and then this reconfigurable checker element is assigned with same functionality as that of the processing element performing important function. Then with the help of N modular redundancy we are identifying at least one healthy processing element. This process is carried out only for half number of terms, and rest can be obtained by shifting the half. So with the help of that healthy processing element faulty processing elements can be identified and isolated, if it contains any important functionality, then it is assigned to a healthy processing element

3. Memory Controller: BIST

BIST (built in self-test) was once reserved for complex digital chips, now the feature has been enabled for chips with small digital content. This move has enabled analog devices, data converters to incorporate this functionality into the system.
BIST can help to provide greater characterization of the process, by enabling deep insight into the process. Greater benefits of BIST can be realized at the system level, by incorporating the BIST functionality at the system level. And more over if the system becomes more complex, we can introduce BIST operation at each component and thus introducing testing feature hierarchically and hence increasing system reliability.

3.1 Static RAM

Static ram differs from dynamic ram, which has to be periodically refreshed. This SRAM uses bi-stable latching circuitry to store each bit. SRAM even though not required to be periodically refreshed but still is volatile when memory is not powered up.

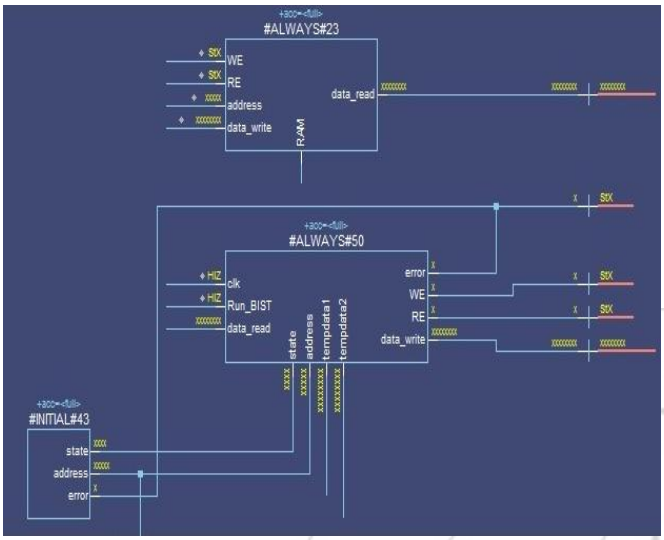

Figure 1: Interfacing diagram of BIST controller and SRAM module

Here we are designing a memory module i.e. SRAM, which is having 32 bytes of data and 4 bytes of address and we are considering 2 SRAM modules out of which one is faulty and other one is considered as healthy.

For demonstrating purpose, we are introducing error at 16th address location in one of the SRAM module and SRAM module is kept fault free, which is considered as healthy. Data is written into SRAM module by enabling WE (write enable). Data is read from the SRAM module by enabling RE (read enable). Output of both SRAM modules is given to BIST controller.

Initially data read, which is input to the BIST controller is zero, data is written to both SRAM modules by enabling WE (write enable). As mentioned earlier, error is introduced at the 16th address location in one of the SRAM.

Here BIST controller is designed w.r.t to proposed algorithm. Here the fault is detected by the BIST controller, which will evaluate all thirty two memory locations and finally after accessing all the thirty two memory locations it will indicate if any error has occurred. This is fault detection, in the next step it will indicate where exactly the error has occurred, and this is fault identification. Once the fault has been identified, that processing element is isolated and if that processing element contains any important functionality, then that functionality is assigned to another processing element.

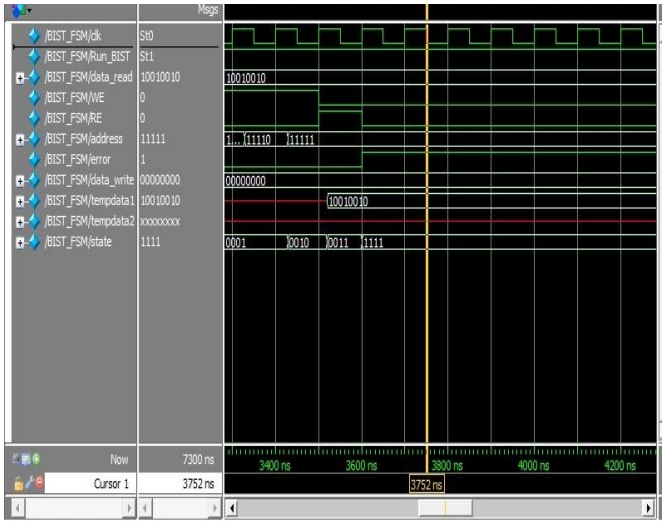

Figure 2: Output of BIST_FSM module

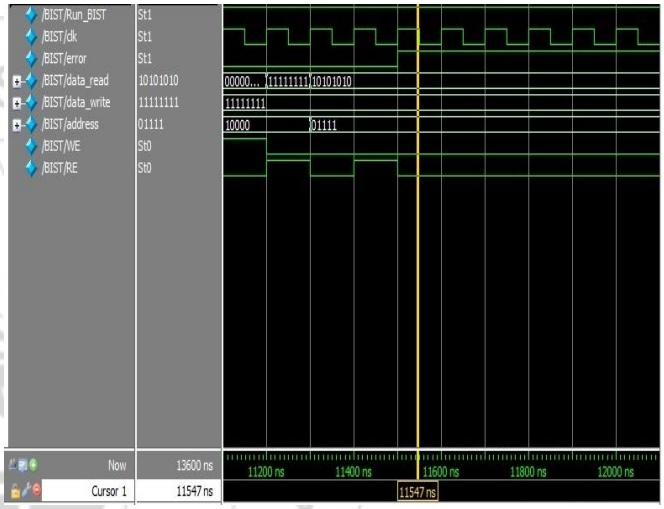

Figure 3: Output of BIST module

4. SOBEL Edge Detector

Edge detection application operating in the harsh environment conditions, their survivability is quite desirable. Sobel edge detector is characterized by its capability to be used in different applications.
Here we are using two 3x3 kernels, one kernel estimates the gradient in x-direction, while other kernel estimates the gradient in y-direction.
Here image is convoluted with both the kernels, to approximate the derivatives in horizontal and vertical directions. The fig illustrates the result of fault handling for a given image.

4.1 Code for designing sobel edge detector and isolation fault from the image

```
img=imread('C:\Users\kilari\Desktop\pic\Koala.jpg');
B=rgb2gray(img);
Subplot (2,3,1)
Imshow (B)
Pause (2)
I=double(B);
```

```
for i=1: size (I, 1)-2
for j=1:size(I,2)-2

% existing mask for x-direction:

mx=((2*I(i+2,j+1)+I(i+2,j)+I(i+2,j+2))-
(2*I(i,j+1)+I(i,j)+I(i,j+2)));

% existing mask for y-direction:

my=((2*I(i+1,j+2)+I(i,j+2)+I(i+2,j+2))-
(2*I(i+1,j)+I(i,j)+I(i+2,j)));

B(i,j)=sqrt(mx.^2+my.^2);
End
End
Subplot (2, 3, 2)
Imshow (B); title ('Existing gradient');
Inx = im noise (B,'salt & pepper', 0.1);
Subplot (2, 3, 3); imshow (inx);
Title ('NOISEY IMAGE detected');
Pause (2)

%Define a threshold value

Thresh=100;
Inx=max (inx, Thresh);
Inx (inx==round (Thresh))=0;
Inx=uint8 (inx);
Filx = medfilt2 (inx, [5 5]);
Subplot (2, 3, 4);
Imshow (filx); title (‘noised image corected ');
Subplot (2, 3, 5);
Imshow (~filx); title ('Edge detected Image');
```

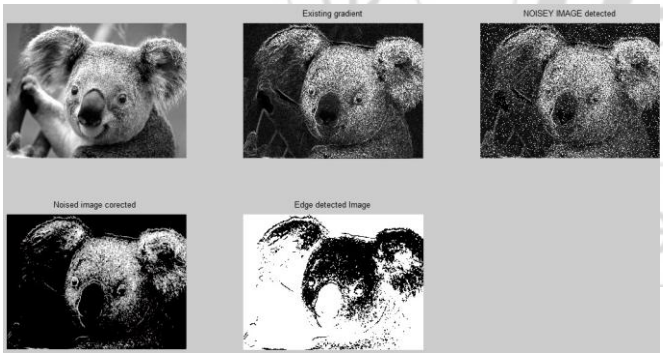

Figure 5: Input image, Existing gradient, Noise detected image, Noise corrected image, Edge detected image

5. Result Analysis

We have proposed a novel fault handling approach in order to achieve survivability of circuits in harsh conditions. Fig 1 illustrates block diagram of BIST controller and interface to SRAM modules. Fig 2 illustrates output of BIST_FSM module, where error is detected after accessing all 32 memory locations; hence here in this step error is detected. Fig 3 illustrates output of BIST module, which shows where exactly the error has occurred, hence error is identified. Fig 4 illustrates the output of sobel edge detector, where images refer to input image, gradient of the input image, faulty image, fault corrected image, and finally edge detected image.

References

[1] Naveed Imran, Jooheung Lee “Amorphous Slack Methodology for Autonomous Fault-Handling in Reconfigurable Devices”
[2] M. Agarwal, B. Paul, M. Zhang and S. Mitra, “Circuit failure prediction and its application to transistor aging”, in VLSI Test Symposium, 25th IEEE, **(2007)** May, pp. 277–286
[3] W. Rao, C. Yang, R. Karri and A. Orailoglu, “Toward future systems with nanoscale devices: Overcoming the reliability challenge”, Computer, vol. 44, no. 2, **(2011)** February, pp. 46 –53
[4] R. Hyman Jr., K. Bhattacharya and N. Ranganathan, “Redundancy mining for soft error detection in multicore processors”, Computers, IEEE Transactions on, vol. 60, no. 8, **(2011)** August, pp. 1114 –1125

Author Profile

Talwade Yuvaraj was born on 22nd October 1988, completed his Bachelor of Engineering in Electronics and Communication, currently pursuing M. Tech in VLSI Design & Embedded System in Lingaraj Appa Engineering College BIDAR, under Visvesvaraya technological university.