IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 6, June 2015.
www.ijiset.com
ISSN 2348 – 7968

# Improved Compiler-Directed Soft Error Mitigation for Embedded System

**Kalluri N V Satya Naresh, Divya Vani .Y**
Shri Vishnu Engineering College for Women
Bhimavaram , Andhra Pradesh, India

**Abstract.** Some faults during the program execution does not cause much damage but causes incorrect execution in the program and such faults are called soft errors (transient errors). Mitigation of these soft errors can be done using software or hardware methods or both in combination (co-design). There are various methods to reduce software errors such as TMR1 (Triple Modular Redundancy), TMR2 and Swift-R. In these methods, Swift-R is more effective than TMR1 and TMR2 but it consumes more energy. This paper focuses on an approach to make use of register swapping technique along with Swift-R which reduces the energy consumption. This energy efficiency is achieved by inducing compiler managed RF protection schemes. In this RF protection scheme, the reduction of soft errors is done by protecting not all registers but those which are used in the program. The experiments have shown that this approach increases the program reliability as well as efficient utilization of energy due to reduced code execution.

**Keywords:** Fault tolerance, reliability, soft error, single, embedded systems design, hardware/software co-design, design space explorationTMR1, TMR2, SWIFT-R, Register Files

## 1 Introduction

Day to day power consumption and reliability are becoming important factors [2] in program designing. To achieve reliability, soft errors in the program must be reduced or removed [1,2].Soft ware and hardware methods can be used for reducing errors in the program or co-design technique is used efficiently. However hardware methods provide reliability but there are more costly and time consuming [1,3,4]. In order to reduce cost, software methods can be used which takes less cost and time and are also easy to implement [1,7]. There are software methods like TMR1, TMR2 and Swift-R methods to reduce errors. TMR1 and TMR2 deal with errors where arithmetic operation errors can only be reduced [1,8].These methods don't deal with register file errors so to deal with register file errors Swift-R method can be used. This method reduces soft errors efficiently than TMR1 and TMR2 by using SOR (Sphere of Replication) method [1, 5].

Swift-R method reduces errors by adding redundancy to the code based on basic blocks in the program [1]. First basic blocks are found and then redundancy is added to the program wherever required [1, 9]. As redundancy is added number of errors in the program gets reduced but execution time of the program gets increased. Power consumption increases as execution time increases. So to utilize energy efficiently along with program reliability RF (Register File) protection scheme with compiler managed optimization technique can be used [1,10].

In Swift-R method soft errors are reduced by handling all register files used in the program [1, 11]. In the RF scheme only those registers which are mostly in the program and which causes more vulnerability to the program are handled for reduction of soft errors [2,12].This are also called Partial Protected Register Files. As only some registers are handled for error reduction by adding redundancy, power consumed by the program is less compared to Swift-R method. Hence the reliability of the program and also efficient utilization of energy by the program is achieved. Errors can be reduced in register files by using faults tolerant design [2]. Hardware techniques can be used for fault tolerant design in which parity and error correction code methods helps for adding reliability to register files [2,11].Finding which registers are mostly used in the program is carried out in hardware technique and this done statically[2,13]. Software techniques can be used for finding which registers can be reallocated for performing operations of the program ,so that the power consumption is more reduced [2,14,27] .This technique is called Software PPRF approach and this approach works more efficiently for reducing power consumption than hardware approach[2,15]. Also further more energy as well as much amount of time is saved by using the register swapping technique [2].

## 2 Related Work

Faults can be of three types, viz. silent data corruption, unACE (unnecessary for Architecturally Correct Execution), and Hang [1]. If the program executes but don't get the expected output then it is silent data corruption (SDC) fault. If the program executes correctly and gives partial correct output is called unACE. If there is abnormal finish of the executing program or the program is in infinite loop then the condition is called as hang. Only the SDC faults are handled [1] and are considered as soft errors.

Faults occur in the program while executing, which does not causes more damage to the program but these faults cause incorrect execution of program that might take place. These types of faults in program are addressed as soft errors [1, 6]. Reducing the effect of soft error in the program can be done by either hardware methods or software methods [3]. Hardware methods are expensive and time consuming for reduction of soft errors, and so software methods are used where the cost is comparatively less [4]. For making the reduction of soft errors to be more efficient and effective we can go for co-design method where we can use software methods along with hardware methods [5].

To reduce soft errors in program using software methods, many algorithms proposed like TMR1, TMR2 and Swift-R methods. These methods, deal with the soft errors by adding redundancy to the program [1]. First basic blocks in the program is found using basic block algorithm. Based on these basic blocks found, redundancy is added to the code or program wherever required and also depending on the method that is chosen for reduction of soft errors [1]. In TMR1 and TMR2 methods, we can reduce errors in the program but they do not deal with register file errors [1]. The can reduce soft errors for arithmetic operations. To deal with register file errors Swift-R method is being used. In the Swift-R method, the reduction of errors is more efficient than TMR1 and TMR2 methods [1].

Swift-R method is used effectively for a fault reduction in register files and in this method, redundancy is added to the code is to deal with faults [1, 9, 5]. By adding redundancy to the program the probability of error occurrence is reduced i.e. the mean work to failure metric is reduced in the program [1,8]. In this method, use of SOR technique and the parameters like InSoR and OutSoR is done [1,9]. This method reduces errors but it takes more time to execute and because of which more power is consumed [2].

To mitigate the soft errors of the register files for a program and also to make it energy efficient, the approach of compiler managed RF (Register File) mechanism is be used. This approach adds reliability to the code and utilizes energy efficiently [2]. In this approach, protecting all the registers in the program cannot be done, only the registers where data operations are performed are selected and protected i.e. only the top "k" vulnerable registers are protected. In Software optimizing approach optimizing the code is performed by frequently reallocating registers [7]. Increasing the program reliability as well as efficient utilization of energy can be done with the help of register swapping methodology [2].

Hardware techniques can be used for fault tolerant design in which parity and error correction code methods helps for adding reliability to register files [2,11].Finding which registers are mostly used in the program is carried out in hardware technique and this done statically[2,13]. Software techniques can be used for finding which registers can be reallocated for performing operations of the program ,so that the power consumption is more reduced [2,14,27] .This technique is called Software PPRF approach and this approach works more efficiently for reducing power consumption than hardware approach[2,15]. Also further more energy as well as much amount of time is saved by using the register swapping technique [2].

The number of logic variables which are affected by the fault due to failure event describes extent of a fault [16, 17]. Control flow of a graph may be modified due to soft errors [18]. Value of the computation may also be changed due to soft errors and EDDI (Error detection by duplicate instructions) is one technique to protect from these types of errors without using hardware [19, 20].

**SoR (Sphere of replication)**

SoR is a technique which is used to replicate the data which helps in mitigation of soft errors [1,22]. The boundaries of SoR work as a protection for the registers. In the SoR, the sphere is considered; the instruction enters inside it which having some data (read the value from memory, reading an input port or loading the value into the register) is called as inSoR[1,26]. Subsequently

the data go out of the SoR by the execution of instruction (storing the values to the memory, writing to the output port) is called outSoR.[1,22,23]

The data is crossing the boundaries of SoR is depending upon the implemented technique. Suppose, if the memory system is present inside the sphere. So, the read and write operation can be done inside the sphere, no data crossing the boundaries of the SoR [1,24]. But if the memory subsystem is present outside the sphere then for the reading operation from the memory and writing operation into the memory can cause some data to cross the sphere boundaries. So, these instructions must be handled in a special way [1,22, 26].

**Fig 1: Sphere of Replication**

The figure defines the input and output SoR and the operation that are perform in and out of memory [1].

The instruction goes outside the SoR should be error free because if the instruction is not error free it will provoke some other unrecoverable errors [2,22, 25].

Then the code is decomposes into nodes called as basic blocks of CFG after each instruction comes from outSoR[2,23,25].

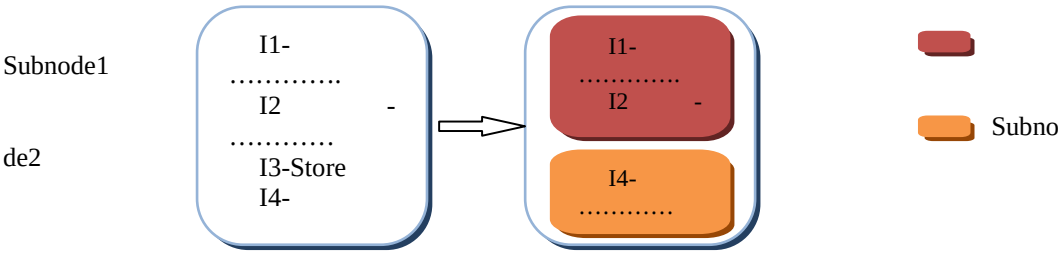


**Fig 2: SoR Basic Block Generation.**

## 3 Approach

The approach looks toward making the use of the register swapping method along with the swift-r technique. This hybrid approach mainly focuses on the mitigation of soft errors with used of minimum energy overhead.

For evaluating the soft error Swift-R algorithm is used. It is the software technique used to recover all the faults of the RF.

The steps are as follows:

1. Identify the nodes (Basic Blocks) and sub nodes of the program and build the CFG.
2. First time the data triplication is takes place of all nodes that comes into SoR. Triplication is performing only in RF but not in memory subsystem. We assume that the memory has its own protection mechanism.
   Therefore for every instruction considered as inSoR two additional copies are created. These copies are created by coping the data of the registers, the memory ad port access are not repeated.
3. These three copies perform operation on data. Operation are logic, arithmetic, shift rotate etc.
4. The data consistency is verified for the instructions such as outSoR instruction [] and the other instruction located prior to conditional branch. As it can have an effect on the flags, this verification is necessary.
5. Unused register in the program can be vacated.

The problem of Swift-R algorithm is how to find the top registers which are currently used. By which the soft error as well as the energy, time overhead is minimized. In first approach the

allocation of register is done for all the registers for both purposes i.e. time and power. And the second approach in which register swapping is used.

The first approach is complex; allocation of all register is a NP –Hard problem. Decrease in performance is takes place when the register is re-allocated which require register re-compilation for whole program. The second approach having several advantages in which no recompilation is takes place for the whole code and no decrement in the performance.

Our approach is working on register swapping method. This method is divided into two levels. One is program level and second is function level. In the first approach register assignment is takes place at program level. Swapping is takes place between one function to another. For the swapping information register reassignment vector (RRV) is needed which works for all functions. In the second function level approach, assignment of register is different for each function. For the swapping information of register RRV for each function is required and for the whole program one table is required for storing information called as register reassignment table (RRT). In the function level register swapping the table is maintained by caller and callee to saved register called as t-register and s-register respectively. They can change their own register groups. By this method FRS problem divide into two versions they are: FRS/t and FRS/s. the calling convention is not takes place when changes are done in the program consistently.

Problem Domain

1. Our main goal is to minimize the energy used by the register and to maximize the reliability per energy. Let ΔVul is the vulnerability of register files and ΔEn be the energy for RF protection. Let the total vulnerability of the system is $V_0$ + ΔVul and total energy of the system will be $E_0$ +ΔEn, where $V_0$ is the vulnerability of the system without the use of RF and $E_0$ is the energy used by the system without RF protection. Our aim is to find that the value of $V_0$ >> ΔVul and $E_0$ >> ΔEn.

$$\begin{aligned} \text{Min(VulEn)} &\Leftrightarrow (V_0 + \Delta\text{Vul})(E_0 + \Delta\text{En}) \\ &\Leftrightarrow E_0\Delta\text{Vul} + V_0\Delta\text{En} + \Delta^2 \end{aligned}$$

Here the value of α is equal to $V_0/ E_0$ which is a constant value.

2. The maximum use of energy is takes place when protection of register takes place by using redundancy techniques. So, the measurement of energy is takes place by using two components. First is the dynamic component and the other is static component. Where the dynamic component is measured by calculating the number of read operation and write operation in the register while the static component is measured at runtime. The dynamic energy component is calculated by measuring the energy overhead by the number of accesses to protected register. Each access to the register has different energy consumption. The energy overhead in the RF is :

$$\Delta\text{En} = e_r N_{RF^{(r)}} + e_w N_{RF^{(w)}}$$

Where $e_r$ and $e_{w\ are}$ the energy overhead of protected RF for read and write operations respectively. $N_{RF}^{(r)}$ and $N_{RF}^{(w)}$ are number of read and write operations respectively.

3. Swapping of registers: The PRS and FRS are same when program having only one function (while PRS have large scope as compare to FRS). So, in FRS let Rs be the number of register it may be t-register, s-register or PRS –swapping register and NUM is the number of function in the program. Then the solution of FRS is represented by using RRT T= $\{\rho^{\square fun} | fun= 1……. N\}$ means each function has its own RRV.
In the RFV $\rho^{\square fun}$ the function fun implies that all the occurrences of register $\rho_r^f$. In the assembly code of function f will be replaced with r in the transformed program or can say that all the register assigned to $\rho_r^f$ now assigned to register r in the transformed program.

In the last, there are less technique that works for both energy and reliability of registers. Achieving both at the same time by minimizes the energy consumption and maximizing the reliability is very rare. Register Files are the most important part for microprocessors, our approach mitigates the soft errors with less energy consumption is advantageous and effective.

## 4 Results

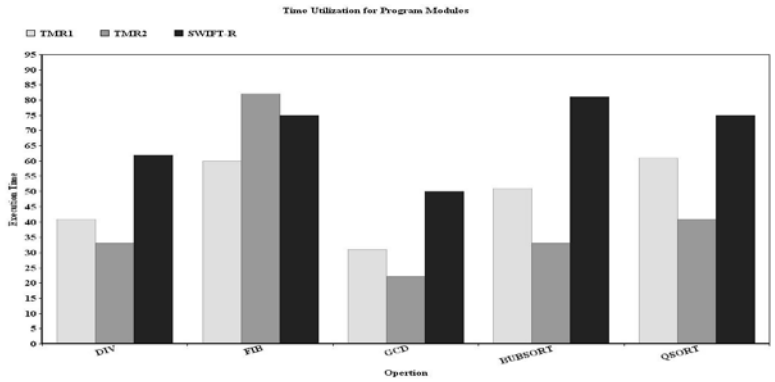


**Fig 3: Time Utilization for Program Modules**

The graph gives the execution time details about the program modules which are used for swapping the data by the TMR1, TMR2 and SWIFT-R method. The utilization of the difference of the time cycle is done for transfer of the data while exchanging the data in between two variables. This result shows the difference in time through the registers for different techniques like TMR1, TMR2 and SWIFT-R.

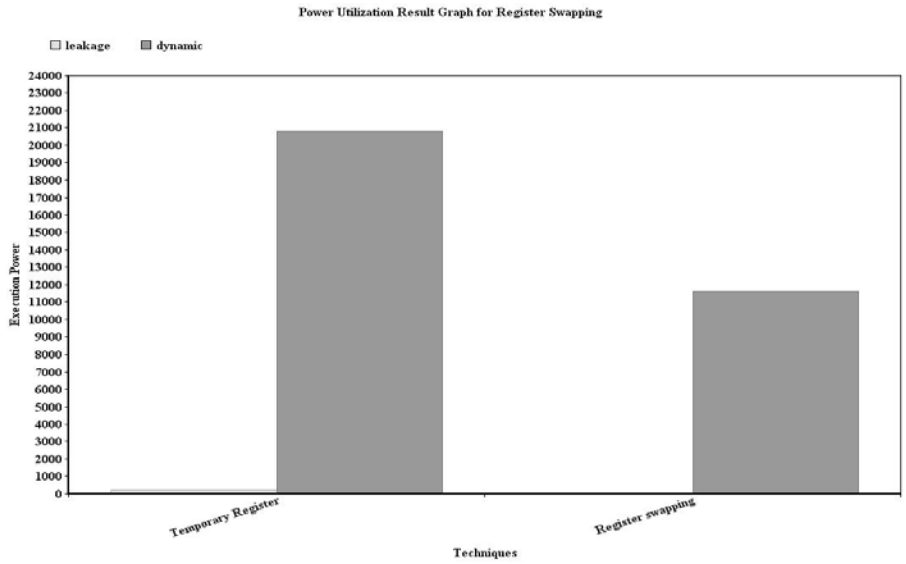


**Fig 4: Power Utilization Result Graph for Registers**

The graph gives the power consumption details about the register swapping used by the software method by taking the advantage of the difference of the time cycle used for transfer of the data while swapping in between two registers. This result also shows that the leakage power through the registers is less in the register swapping technique than the one which does not uses it.

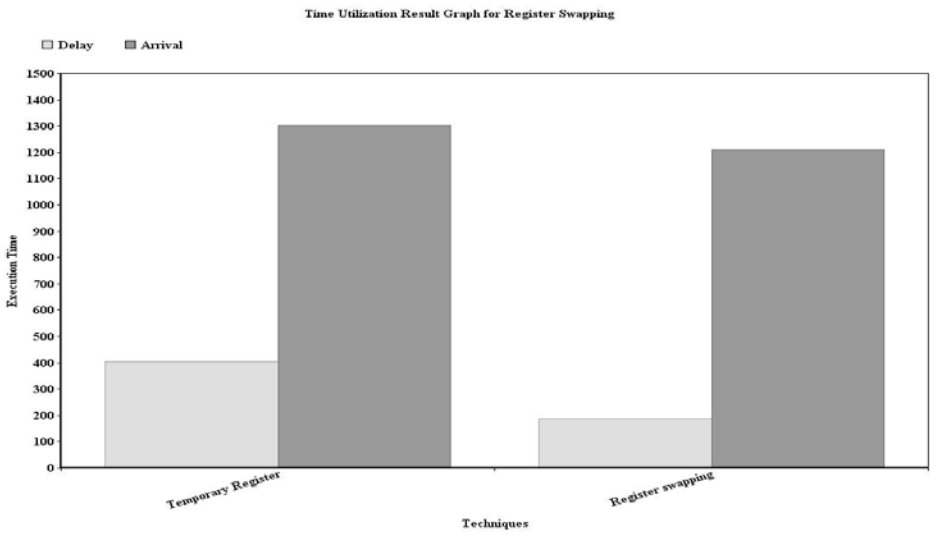


**Fig 5: Time Utilization Result Graph for Registers**

The time graph displays the differences of the delay and arrival time between two techniques used here in this approach. In the data transfer takes comparatively less time when register swapping technique is used instead of simply using temporary register swap along with SWIFT-R technique.

## 5 Conclusion

The approach in this paper presents a hybrid approach of improving the micro-architecture and compiler to have better energy efficient register file protection for embedded system whereas the TMR1, TMR2, SWIFT-R and such other software techniques concentrated only on mitigation of the soft errors. This approach proves that a more energy efficient model can be created by using both the hardware and software techniques together. This technique can also reduce the execution time required using comparatively less registers. So the co-design of the hardware and the software implementation is improved by adding the register swapping technique in it.

## References

[1]. Mitra, S.; Seifert, N.; Zhang, M.; Shi, Q.; Kee Sup Kim, "Robust system design with built-in soft-error resilience," *Computer* , vol.38, no.2, pp.43 - 52, Feb. 2005

[2]. Baumann, R.C., "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability,* , vol.5, no.3, pp.305 - 316, Sept. 2005

[3]. Shivakumar, P.; Kistler, M.; Keckler, S.W.; Burger, D.; Alvisi, L., "Modeling the effect of technology trends on the soft error rate of combinational logic," *Proceedings. International Conference on Dependable Systems and Networks, 2002. DSN 2002.*, pp.389 - 398, 2002

[4]. Karnik, T.; Hazucha, P., "Characterization of soft errors caused by single event upsets in CMOS processes," *IEEE Transactions on Dependable and Secure Computing,* , vol.1, no.2, pp.128 - 143, April-June 2004

[5]. Edwards, R.; Dyer, C.; Normand, E., "Technical standard for atmospheric radiation single event effects, (SEE) on avionics electronics," *2004 IEEE Radiation Effects Data Workshop,* , vol. 22, no. 22, pp.1 - 5, July 2004

[6]. R. Baumann, "Soft Errors in Commercial Semiconductor Technology: Overview and Scaling Trends," IEEE Press, IEEE 2002 Reliability Physics Symp. Tutorial Notes, Reliability Fundamentals, pp. 121-01.1–121-01.14, Apr. 2002.

[7]. Michalak, S.E.; Harris, K.W.; Hengartner, N.W.; Takala, B.E.; Wender, S.A., "Predicting the number of fatal soft errors in Los Alamos national laboratory's ASC Q supercomputer," *IEEE Transactions on Device and Materials Reliability,* , vol.5, no.3, pp.329 - 335, Sept. 2005

[8]. Lackey, D.E.; Zuchowski, P.S.; Bednar, T.R.; Stout, D.W.; Gould, S.W.; Cohn, J.M., "Managing power and performance for system-on-chip designs using Voltage Islands," *ICCAD 2002. IEEE/ACM International Conference on Computer Aided Design, 2002.*, vol. 10, no. 14, pp.195 - 202, Nov. 2002

[9]. Li, F.; Chen, Gu.; Kandemir, M., "Compiler-directed voltage scaling on communication links for reducing power consumption," *ICCAD-2005. IEEE/ACM International Conference on Computer-Aided Design, 2005.*, vol. 6, no. 10, pp. 456 - 460, Nov. 2005

[10]. H. Giefers and A. Rettberg, "Energy Aware Multiple Clock Domain Scheduling for a Bit-Serial, Self-Timed Architecture," *Proceeding SBCCI '06 Proceedings of the 19th annual symposium on Integrated circuits and systems design* Pages. 113 - 118, 2006.

[11]. Li Shang; Li-Shiuan Peh; Jha, N.K., "Dynamic voltage scaling with links for power optimization of interconnection networks," *The Ninth International Symposium on High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings.*, vol. 8, no. 12, pp.91 - 102, Feb. 2003.

[12]. Manolache, S.; Eles, P.; Zebo Peng, "Fault and energy-aware communication mapping with guaranteed latency for applications implemented on NoC," *Design Automation Conference, 2005. Proceedings. 42nd* , vol. 13, no. 17, pp.266 - 269, June 2005.

[13]. Soteriou, V.; Li-Shiuan Peh, "Dynamic power management for power optimization of interconnection networks using on/off links," *High Performance Interconnects, 2003. Proceedings. 11th Symposium on* , vol. 20, no. 22, pp. 15 - 20 Aug. 2003.

[14]. I. Koren and C. M. Krishna, "Fault-Tolerant Systems", San Mateo, CA:Morgan Kaufmann, 2007.

[15]. S. Carr, K.S. McKinley, and C.-W. Tseng, "Compiler Optimizations for Improving Data Locality," Proceedings of the Sixth International Conference on Architectural Support for Programming Languages and Operating Systems, vol. 29, no. 11, pp. 252-262, 1994.

[16]. Guangyu Chen; Kandemir, M.; Karakoy, M., "Compiler Support for Voltage Islands," *2006 IEEE International SOC Conference*, vol., no., pp.189 - 192, 2006

[17]. Kandala, M.; Wei Zhang; Yang, L.T., "An Area-Efficient Approach to Improving Register File Reliability against Transient Errors," *Advanced Information Networking and Applications Workshops, 2007, AINAW '07. 21st*, vol.1, no., pp.798,803, 2007.

[18]. Oh, N.; Shirvani, P.P.; McCluskey, E.J., "Control-flow checking by software signatures," *IEEE Transactions on Reliability*, vol. 51, no. 1, pp. 111–122, Mar.2005.

[19]. Oh, N.; Shirvani, P.P.; McCluskey, E.J., "Error detection by duplicated instructions in super-scalar processors," *IEEE Transactions on Reliability*, vol.51, no.1, pp.63 - 75, Mar 2002.

[20]. Narayanan, S.H.K.; Ozturk, O.; Kandemir, M.; Karakoy, M., "Workload Clustering for Increasing Energy Savings on Embedded MPSoCs," *Proceedings. IEEE International SOC Conference, 2005.*, vol., no., pp.155 - 160, 19-23 Sept. 2005.

[21]. Jared C. Smolens, Brian T. Gold, Babak Falsafi, and James C. Hoe. "Reunion: Complexity-Effective Multicore Redundancy", *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, Pages 223-234 2006

[22]. S. K. Reinhardt and S. S. Mukherjee. "Transient fault detection via simultaneous multithreading". *Proceedings of the 27th annual international symposium on Computer architecture*, Volume 28 Issue 2, May 2000.

[23]. K. Sundaramoorthy et al, "Slipstream processors: improving both performance and fault tolerance". *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, Volume 28 Issue 5, Pages 257 – 268, Dec. 2000 ,

[24]. H. Zhou. "A case for fault tolerance and performance enhancement using chip multi-processors". *Journal IEEE Computer Architecture Letters*, Volume 5, Issue 1, Page 6-25, January 2006

[25]. T. N. Vijaykumar et al."Transient-fault recovery using simultaneous multithreading". *ISCA '02 Proceedings of the 29th annual international symposium on Computer architecture*, Volume 30 Issue 2, Pages 87 - 98, May 2002.

[26]. S. Park, A. Shrivastava, N. Dutt, A. Nicolau, Y. Paek, and E. Earlie,"Bypass aware instruction scheduling for register file power reduction," *LCTES '06 Proceedings of the 2006 ACM SIGPLAN/SIGBED conference on Language, compilers, and tool support for embedded systems*, vol. 41, no. 7, Pages 173 – 181, 2006