International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 03 | Mar 2019 www.irjet.net p-ISSN: 2395-0072

Analysis of Micro Inversion to Improve Fault Tolerance in High Speed VLSI Circuits

Somashekhar[1], Vikas Maheshwari[2], R. P. Singh[3]

[1]Research Scholar, Dept of ECE, SSSUTMS, Sehore (M.P), India-
[2]Associate Professor, Dept of ECE, Bharat Institute of Engineering and Technology, Hyderabad. India-
[3]Vice-Chancelor & Professor, Dept of ECE, SSSUTMS, Sehore (M.P), India-
[1]somashekhar49@gmail.com, [2]maheshwarivikas1982@gmail.com, [3]provc@sssutms.co.in

---

Abstract - With technology scaling, the reliability of circuits is becoming a rising concern. The emergence of logic errors in the field cause by faults escaping manufacturing testing, aging, single event upsets, or process variations is increasing. Conventional techniques for online testing and circuit protection repeatedly require a high design effort or result in high area overhead and power consumption and are unsuitable for low cost systems. The primary motive for introducing fault tolerance in VLSI circuits is yield enhancement, increasing the percentage of fault free chips obtained. The active area of monolithic VLSI chips has always been limited by random fabrication defects, which appear impossible to eliminate in even the best manufacturing processes. The larger the circuit, the more likely it will contain such a defect and fail to operate correctly. Thus, the defect density in any fabrication line limits the size of the largest defect free chip producible with commercially viable yields. Larger circuits demand a fault tolerance capability to overcome fabrication defects while avoiding unreasonable costs. In nm technologies, circuits be more and more sensitive to a variety of perturbations. Transient faults can take place in a processor as a result of electrical noise, like crosstalk, or high energy particles, like neutrons and alpha particles. These faults be able to cause a program running on the processor to behave erratically, if they propagate and change the architectural state of the processor. These faults can occur in memory arrays, sequential elements or in the combinational logic in the processor. Protection against transient faults in combinational logic has not received much attention traditionally because combinational logic has a natural barrier stopping the propagation of the faults. System performance is increased when the nodes are able to recover locally from most errors caused by transient faults. The circuitry added for concurrent error detection generally reduces performance. By means of a technique called micro rollback, it is achievable to eliminate the performance penalty of concurrent error detection.

Keywords: Micro inversion, Fault Tolerance, VLSI, Processor, IC, Register file.

1. INTRODUCTION

As with every piece of machinery, ICs are prone to failure. Through technology scaling, transistor sizes are reduced to open the way for increased functionality with reduced in general power dissipation, device dimensions and manufacturing costs but despite those advantages, the reliability of ICs has been affected. The increasing probability of circuit failure caused by increasing device complexity and the errors caused by increased delay due to temperature rise in CMOS circuits. Faults in a distributed embedded system can be permanent, intermittent or transient (also known as soft errors). Permanent faults cause long-term malfunctioning of components. These faults emerge for a short time. Causes of intermittent faults are within system boundaries, while causes of transient faults are external to the system. They might damage data or lead to logic miscalculations, which can outcome in a fatal failure. Due to their higher rate, these faults cannot be addressed in a cost-effective way by applying traditional hardware-based fault tolerance techniques suitable for tolerating permanent faults. Embedded systems with fault tolerance have to be carefully designed and optimized, in order to satisfy strict timing requirements without exceeding a certain limited amount of resources. Moreover, not only performance and cost related requirements have to be considered but also other issues such as debug ability and testability have to be taken into account.

2. LITERATURE REVIEW

A fault-tolerant system may be able to tolerate one or more fault-types including -- i) transient, intermittent or permanent hardware faults, ii) software and hardware design errors, iii) operator errors. Gayathri and Prabakaran

[1] discussed some important factors of failures. One important factor is arbitrary node or link failure which results in denial of service. In cloud computing, load balancing is required to distribute the dynamic local workload evenly across all the nodes. It helps to attain a more user fulfillment and resource utilization ratio by ensuring an efficient and fair allocation of every computing resource. Identified some of the load balancing algorithms which distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve best possible resource utilization, maximize throughput, minimize response time, and avoid overload. When all these issues are addressed naturally the system becomes a fault tolerant one.

Win Naing [2] proposed a fault-tolerance management framework for private clouds development. Previous researchers developed Eucalyptus in organize to facilitate the creation of private clouds. But Eucalyptus is no fault tolerant system and no VM monitoring is performed thus limiting the support for advanced VM placement policies (e.g., consolidation). Eucalyptus does not also include any self-healing features and strictly distinguishes between cloud and cluster controllers. Therefore, they proposed the fault-tolerance management framework over Eucalyptus by adding new component Cluster Controller Manager (CCM).

Sheheryar and Fabrice [3] proposed a scheme of fault tolerance mechanism for real time computing on cloud infrastructure. It has advantages of forward recovery mechanism. It performs the reverse recovery if the node with best reliability could not achieve the SRL. There is another big advantage of this scheme. It does not suffer from domino effect as check pointing is made in the end when all the nodes have produced the result.

Y Tamir [4] proposed a Fault-tolerant system frequently rely on self-checking compute nodes. It detect errors immediately they occur, hence prevent the spread of invalid information throughout the system.

3. MICRO INVERSION

A key to achieving a high degree of fault tolerance is the ability to detect errors as soon as they occur and prevent erroneous information from spreading throughout the system. In highly reliable systems, this is usually accomplished by checkers and isolation circuits in the communication paths from each module to the rest of the system. This additional circuitry reduces performance by requiring either longer clock cycles or additional pipeline stages. This presents a technique, called micro rollback. Operations performed on this erroneous information are "undone" by means of a hardware mechanism for fast rollback of a few cycles. Straightforward realization of micro rollback will need of significant performance and chip area overheads for replicating all the storage elements in each module.

This paper discusses techniques for efficient analysis of micro rollback in VLSI systems. It focuses on the micro architecture and VLSI realization of a VLSI RISC processor that is able of micro rollback. A micro rollback of a subsystem consists of bring the subsystem back a only some cycles to a state reached in the past. It is so necessary to save the state of the subsystem at each cycle boundary [10]. If the "subsystem" is a processor, the state is the contents of all storage elements which carry useful information across cycle boundaries. It is composed of the program counter, the program status word, the instruction register, and the register file, it also includes the contents of some pipeline latches and some registers in the state machine which can be changed during the execution of a multicycle instruction. Since instructions also modify external memory, the state of the cache must also be preserved. A rollback restores the contents of the cache to its state a few cycles earlier.

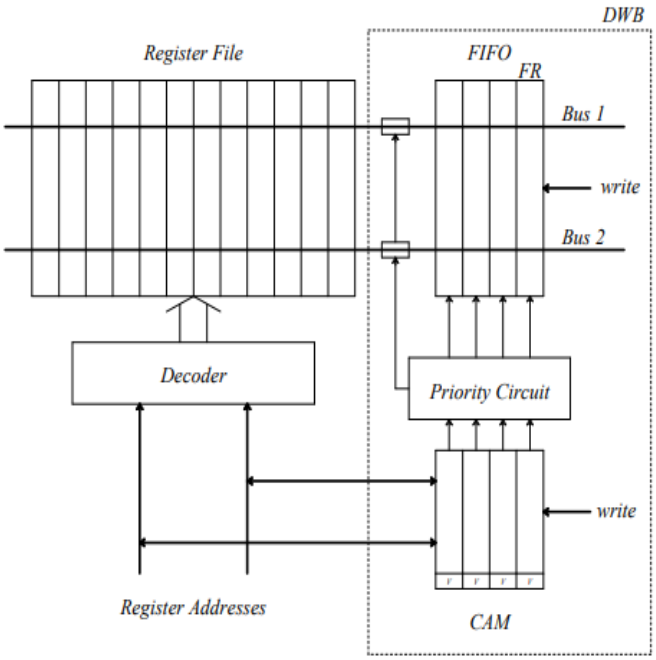

Figure 1. A register file with support for micro Inversion

4.MULTI MODULE SYSTEM

Periodic check pointing of process states and roll back to a previous state when an error is detected is a common technique for error recovery in distributed systems[11]. If each process is check pointed independently, rolling back one process may require rolling back a second process

further in time which, in turn, may cause a third process to roll back, etc. leading to an uncontrolled domino effect [11]. In the worst case this can result in all processes in the system rolling back to their state when the system is initialized. In the context of micro rollback, which is done at the level of hardware modules, the domino effect cannot occur in such system. However, if the modules are connected in an arbitrary topology, where there are several independent communication paths between pairs of modules, the domino effect could, potentially, occur. Since the range of rollback is severely limited (a few cycles), this can make recovery impossible.

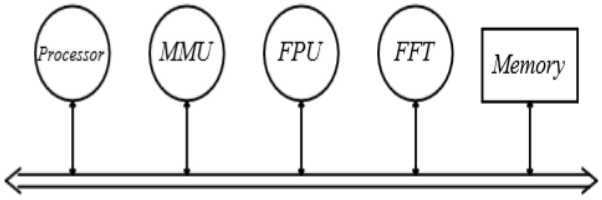

Figure 2. Multi-Module System

In a system if modules are interconnected via a common bus, this problem can be solved by using bus transactions as a common logical clock.

5.CONCLUSION

One of the keys to achieving a high degree of fault tolerance is the ability to detect errors instantly after they occur & prevent invalid information from distribution all over the system. This primary problem in achieving fault tolerance in VLSI systems is able to overcome by performing checks in parallel with intermodule communication. This paper analyses the parallel error checks in concurrence with micro rollback can be used to support fault tolerance in complex multi module high performance VLSI systems.

REFERENCES

[1] Ms. G. Gayathri and Dr. N. Prabakaran, "Achieving Fault Tolerance in Cloud Environment by Efficient Load Balancing", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) Volume 2, Issue 3, May – June , 2013 ISSN 2278-6856.

[2] WinNaing "Fault-tolerant Management for Private Cloud System", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 1, Issue 1, May-June 2012 ISSN 2278-6856.

[3] Sheheryar Malik, FabriceHuet, " Adaptive Fault Tolerance in Real Time Cloud Computing", IEEE World Congress on Services, Jul 2011, Washington DC, United States. IEEE, pp.280-287.

[4] Y Tamir, "Self-checking self-repairing computer nodes using the Mirror Processor", IEEE Journal of solid-state circuits, vol. 21. No. I. January 1992.

[5] Han C.C., Shin K. G. and Wu J., "A Fault-Tolerant Scheduling Algorithm for Real-Time Periodic Tasks with Possible Software Faults", IEEE Computers 2003.

[6] Prasenjit Kumar Patra, Harshpreet Singh, Gurpreet Singh, "Fault Tolerance Techniques and Comparative Implementation in Cloud Computing", International Journal of Computer Applications (0975 – 8887) Volume 64– No.14, February 2013.

[7] Jasbir Kaur, SupriyaKinger, "Efficient Algorithm for Fault Tolerance in Cloud Computing", International Journal of Computer Science and Information Technologies(IJCSIT), Vol.5 (5) , 2014, 6278-6281.

[8] S. Sudha Lakshmi, Sri Padmavati, "Fault Tolerance in Cloud Computing", International Journal of Engineering Sciences Research-IJESR, Vol 04, Special Issue 01,2013, issn:2230-8504, e-ISSN-2230-8512.

[9] Pandeeswari.R, Mohamadi Begum "Rsfts: Rule-Based Semantic Fault Tolerant Scheduling For Cloud Environment", Council for Innovative Research International Journal of Computers & Technology. Volume 4 No. 2, March-April, 2013, ISSN 2277-3061.

[10] W. W. Hwu and Y. N. Patt, ''Checkpoint Repair for Out-of-order Execution Machines,'' 14th Annual Symposium on Computer Architecture, Pittsburgh, PA, pp. 18-26 (June 1987).

[11] B. Randell, P. A. Lee, and P. C. Treleaven, ''Reliability Issues in Computing System Design,'' Computing Surveys 10(2), pp. 123165 (June 1978).

[12] L. D. Babu and P. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments", in Applied Soft Computing, Vol. 13(5), pp. 2292-2303, (2013).

[13] R. Kaur and P. Luthra (2012), "Load Balancing in Cloud Computing", In Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing, ITC.

[14] M. Dorigo, G. D. Caro and L. M. Gambardella (1999), "Ant algorithms for discrete optimization", Artif. Life, Vol. 5(2), pp.137-172.

[15] Virendra Singh Kushwah, Sandip Kumar Goyal and Priusha Narwariya, " A survey on various fault tolerant approaches for cloud Environment during load balancing", IJCNWMC, Vol. 4, Issue 6, Dec 2014, 25-34 ISSN(P): 2250-1568; ISSN(E): 2278-9448.

[16] I. Koren, "A Reconfigurable and Fault-Toler- ant VLSI Multiprocessor Array," Proc. 8th Ann. Symp. Computer Architecture, May 1981, pp. 425-441.

[17] N.R. Rejinpaul, L. Maria Michael Visuwasam, "Checkpoint-based Intelligent Fault tolerance For Cloud Service Providers", International Journal of Computers and Distributed Systems Vol. No.2, Issue 1, December 2012 ISSN: 2278-5183.

[18] FetahiWuhib, Mike Speritzer and Rolf Stadler (2012), "Gossip Protocol for Dynamic Resource Management in Large Cloud Environments" IEEE Transactions on Network and Service Management, vol. 9, no. 2, pp. 213- 225.

[19] Ramtilak Vemu et al, "A low-cost concurrent error detection technique for processor control logic", 978-3-9810801-3-1/DATE08 © 2008 EDAA.

[20] Peng Hu, Wei Dai, "Enhancing Fault Tolerance based on Hadoop Cluster", International Journal of Database Theory and Application Vol.7, No.1 (2014), pp.37-48.

[21] Samudrala, P. K., Ramos, J., and Katkoori, S. (2004) "Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs", Nuclear Science, IEEE Transactions on, Vol.51, No.5, pp 2957-2969.