

# Software Based Fault Tolerance – a Survey

**Goutam Kumar Saha**

Member ACM, Senior Member IEEE

**Mail to: CA – 2 / 4 B, Baguiati, Deshbandhu Nagar, Kolkata 700059 India**

[sahagk@gmail.com](mailto:sahagk@gmail.com) [gsaha@acm.org](mailto:gsaha@acm.org)

This article aims to present a survey of important software based (or software controlled) fault tolerance literature over the period of 1966 to 2006. Nowadays, fault tolerance is a much researched topic. A system fails because of incorrect specification, incorrect design, design flaws, poor testing, undetected fault, environment, substandard implementation, aging component, operator errors or combination of these causes [1,7]. Modern microprocessors having faster and denser transistors with lower threshold voltages and tighter noise margins make them less reliable [6] but such transistors yield performance enhancements [4,5]. At the same time, such transistors render processors more susceptible to *transient faults*. Transient faults are intermittent faults that are caused by external events or by the environment [7], for examples, energetic particles [84,93] striking the chip or electrical surges [1,3] etc. Though these faults do not cause permanent faults [2], but they may result in incorrect program execution by inadvertently altering processors' states, signal transfers or stored values on registers etc. If a fault of such type affects program's normal execution, it is considered to be a *soft error* [2,8,85]. Though programming bugs is considered to be an important reason of the most system failures at present but the recent studies suggest that soft errors are increasingly responsible for system downtime. Computing system is becoming more complex and is getting optimized for performance and price but not for availability. This makes soft errors an even more common case. Using denser, smaller and lower voltage transistors has the potential threats to be more susceptible [92] to such increased transient errors. Soft errors

are the errors, which occur because of the unintended transitions of logic state in a circuit typically caused by external source of ionizing radiations. The ionization creates excess free carriers, which recombine with the stored charges, thereby corrupting the state of transistor [8]. Device scaling, reduction in feature size and voltage levels of the transistor, along with high density transistors have increased the risk of hardware faults due to soft errors [85]. Research in [6] predicts that soft error rate (SER) per chip of logic circuits will increase nine orders of magnitude from 1992 to 2011 and at that point will be comparable to the SER per chip of unprotected memory elements [8]. Works in [9,10] have verified the soft error rate (SER) of 52000 FIT (1 FIT equals 1 failure in  $10^9$  Hours) on a 16 Mbit DRAM chip. A system with hundred such chips will experience a fail rate of almost one per week. Studies in [9,10] also claimed that a typical processor's silicon can have a soft-error rate of 4000 FIT, of which 50% will affect processor logic and 50% the large on-chip cache [8]. Until now techniques such as Error Correction Codes (ECC) have been used to correct errors in main memory and system interconnects. Work in [11] states that it is difficult to shield systems effectively from transient faults using fault avoidance techniques. It suggests employing some other means, which are based on fault- masking [27,28] and fault recovery through check pointing or rollback etc [20,21,22,24,75,76,77,88] in order to assure appropriate levels of transient fault tolerance or insensitivity to transient faults. Having analyzed this problem, the study in [11] identifies critical design points and outlines some practical solutions that refer to efficient on-line detectors (detecting errors [24,25,56,61,62,78,80,87,89,91] during the system operation) and error handling procedures [13,14,15,17,19,23]. This framework provides a basis for understanding transient fault problems in digital systems. It can be helpful in selecting optimum techniques to mask or eliminate transient fault effects in developed systems [11,12].

Studies in [16] state that most of the microprocessor-based systems provide little or no checking for failures in the microprocessor itself. These systems rely on the high reliability of VLSI and the inbuilt run-time checks (e.g., virtual memory protection and illegal instruction exceptions) to ensure data integrity. Work in [26] states that reliability of microprocessors is dictated by the faults that cause a reduction of it – Design faults, manufacturing faults and operational faults. Work in [16] presents a model to analyze systems' vulnerability [18] to undetected data corruptions due to transient faults. Though time redundancy provides us dependability against transient faults, however, works in [72,73,74,79] show that this scheme is also effective against permanent faults.

Work in [29] has the intention of NASA's Remote Exploration and Exploration (REE) project to use commercial off-the-shelf, scalable, low-power, fault-tolerant, high-performance computation in space. It has observed that most of the faults caused by the radiation environments in regions of space of interest to REE (Deep Space, Low Earth Orbit) are transient, single event etc. Some of these faults can cause errors at different application levels. The study [29] shows that system and applications software can potentially detect and correct some or many of these errors by using different software fault tolerance approaches such as replication, voting, and masking with a focus on algorithm-based fault-tolerance [7, 31,32,33,34,35,37] or by using a combined software and hardware approaches [30,36,38] such as fault avoidance, redundancy, masking, and reconfiguration. However, such approaches allow trade-offs between reliability, power, cost, and computation power for spacecraft in a low-to-moderate radiation environment [29]. The task of fault detection (for example, through algorithm based fault tolerance (ABFT) [32,81] or through assertions [19, 31] etc.) and recovery becomes easier when we make a system as simple as possible as correctly stated "*A system should be as simple as possible in order to achieve its required function—and no simpler*" in [30]. The software implemented fault tolerance (SWIFT)

schemes [2,17,27,90] aim to increase reliability by inserting redundant code to compute duplicate versions of all register values and inserting validation instructions before control flow and memory operations [2]. The redundant and validation instructions are inserted by the compiler and are used to increase the reliability of systems without any hardware requirements. Again, the studies in [40, 41, 42, 43, 82] aim to tolerate errors in program control flow that account for 33%-77% of all errors. Work in [45] aims to treat software fault-tolerance as a robust supervisory control (RSC) problem and propose a RSC approach to software fault-tolerance. In this approach the software component under consideration is treated as a controlled object that is modeled as a generalized Kripke structure or finite-state concurrent system [44,45].

We have several software fault tolerance schemes as proposed in [46,47,48,49,50] are based on software design diversity in order to tolerate software design bugs. They include the recovery block scheme (RBS) programming, consensus recovery block programming, N-version programming (NVP), N Self-checking programming (NSCP) and data diversity programming etc. The major ideas underlying these fault-tolerant schemes are redundancy and diversity, or diverse redundancy.

After the design task is over, a fault-tolerant system needs to be evaluated with respect to a system's specifications either on using a Markov model (an analytical model) to determine a system's possible states and the probable chances of states transitions, or by fault injection into a simulated or into a real system [7,39,51,52,53,54,55,57,58,59,60,63,65] for fault coverage etc.

Though multi-version or N-version programming is considered as a method for increasing the overall dependability [64,66,67,68,69,70] of safety-critical systems. However, the increased cost of using this approach may mean that this increase in dependability is not worth the extra expense involved. Analytical results from experiments carried out in [70] show that "(a) a

single-version system [86] is much more dependable than any individual version of the multi-version system, and (b) despite the poor quality of individual versions, the multi-version method still results in a safer system than the single-version solution. Although these results could not be considered conclusive in the general sense.” However, an enhanced single – version - programming scheme (using a kind of robust programming approach appropriate to an application) often shows dependable result in a non-safety critical application [17,71,83] without an extra expense on hardware and diversified software.

## **References:**

- [1] B. W. Johnson, “Designing and analysis of fault tolerant digital systems,” Addison-Wesley, 1989.
- [2] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, D. I. August and S. S. Mukherjee, “Software - controlled fault tolerance,” ACM Transactions on Architecture and Code Optimization, Vol. V, No. N, December 2005, pp.1–28.
- [3] G.K. Saha, “EMI Control by Software for a RF Communication System,” in Book – Electromagnetic Environments and Consequences, Editor: D.J. Serafin, (Proceedings of the IEEE EUROEM’95), Part-II, 1995, pp. 1870-1875, France.
- [4] R.C. Baumann, “Soft errors in advanced semiconductor devices-part I: the three radiation sources,” IEEE Transactions on Device and Materials Reliability, 1(1), March 2001, pp.17–22.
- [5] T. J. O’Gorman, J.M. Ross, A.H. Taber, J.F. Ziegler, H.P. Muhlfeld, I.C.J. Montrose, H.W. Curtis, and J.L. Walsh, “Field testing for cosmic ray soft errors in semiconductor memories,” IBM Journal of Research and Development, 1996, 41–49.

- [6] P. Shivakumar, M. Kistler, S.W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic," Proceedings of the 2002 International Conference on Dependable Systems and Networks, 2002, pp. 389–399.
- [7] A.K Somani and N.H. Vaidya, "Understanding fault tolerance and reliability," IEEE Computer, April 1997, pp. 45-50.
- [8] A. Garg, "Soft error fault tolerant systems: cs456 survey," URL: [www.csc.ncsu.edu/faculty/xie/softerrors.html](http://www.csc.ncsu.edu/faculty/xie/softerrors.html)
- [9] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, B. Chin, M. Nicewicz, C. A. Russell, W. Y. Wang, L. B. Freeman, P. Hosier, L. E. LaFave, J. L. Walsh, J. M. Orro, G. J. Unger, J. M. Ross, T. J. O'Gorman, B. Messina, T. D. Sullivan, A. J. Sykes, H. Yourke, T. A. Enger, V. Tolat, T. S. Scott, A. H. Taber, R. J. Sussman, W. A. Klein, and C. W. Wahaus, "IBM experiments in soft fails in computer electronics (1978-1994)," IBM Journal of Research and Development, 40(1), pp. 3-18, Jan. 1996.
- [10] J. F. Ziegler, H. P. Muhlfeld, C. J. Montrose, H. W. Curtis, T. J. O'Gorman, and J. M. Ross, "Accelerated testing for cosmic soft error rate," IBM Journal of Research and Development, 40(1), pp. 51-72, Jan. 1996.
- [11] J. Sosnowski, "Transient fault tolerance in digital system," IEEE Micro, 14(1), 24-35, Feb 1994.
- [12] D. K. Pradhan, "Fault-tolerant computer system design," Prentice Hall PTR, 1996.
- [13] N. S. Bowen and D. K. Pradhan, "Processor and memory-based checkpoint and rollback recovery," IEEE Computer, 26(2), pp. 22-31, Feb. 1993.
- [14] A. Messer, P. Bernadat, G. Fu, D. Chen, Z. Dimitrijevic, D. Lie, D. D. Mannaru, A. Riska, and D. Milojcic, "Susceptibility of commodity systems and software to memory soft errors," IEEE Transactions on Computers, 53(12), pp.1557-1568, Dec. 2004.

- [15] D. Milojicic, A. Messer, J. Shau, G. Fu, P. Alto, and A. Munoz, "Increasing relevance of memory hardware errors: a case for recoverable programming models," ACM SIGOPS European workshop, pp. 97-102, Kolding, Denmark, Sept. 2000.
- [16] R. Horst, D. Jewett and D. Lenoski, "The risk of data corruption in microprocessor-based systems fault - tolerant computing," FTCS-23, pp. 576 – 585, June 1993.
- [17] G.K. Saha, "Using software to control ESD/EMP in microprocessor-based system," RF Design (EMC Test & Design), November 1995, Argus Inc., pp. 8-11, USA.
- [18] N. S. Bowen and D. K. Pradhan, "Processor and memory based checkpoint and rollback recovery," IEEE Computer, 26(2), pp.22-31, Feb. 1993.
- [19] M. Z. Rela, H. Madeira, and J. G. Silva, "Experimental evaluation of the fail silent behavior in programs with consistency checks," International Symposium on Fault-Tolerant Computing, pp. 394-403, Sendai, Japan, June 1996.
- [20] T. N. Vijaykumar, I. Pomeranz, and K. Cheng, "Transient-fault recovery using simultaneous multithreading," Proceedings of the 29th Annual International Symposium on Computer Architecture, pp. 87–98, May 2002.
- [21] M. Gomaa, C. Scarbrough, T. N. Vijaykumar, and I. Pomeranz, "Transient-fault recovery for chip multiprocessors," Proceedings of the 30th Annual International Symposium on Computer Architecture, 2003.
- [22] S. K. Reinhardt and S. S. Mukherjee, "Transient-fault detection via simultaneous multithreading," Proceedings of the 27<sup>th</sup> Annual International Symposium on Computer Architecture, pages 25–36, June 2000.
- [23] E. Rotenberg, "AR-SMT: A microarchitectural approach to fault tolerance in microprocessors," Proceedings of Fault-Tolerant Computing Systems, 1999.
- [24] G. K. Saha, "Software implemented fault tolerance through data error recovery," ACM Ubiquity, 6(35), 2005, ACM Press.

- [25] G. K. Saha, "Software based fault tolerant array," *IEEE Potentials*, 25(1), pp.41-45, Jan 2006, IEEE Press.
- [26] W. Chris and A. Todd, "A fault tolerant approach to microprocessor design," *Proceedings of the Dependable Systems and Networks*, July 2001.
- [27] G. K. Saha, "Software based computing security & fault tolerance," *ACM Ubiquity*, 5(15), June 2004, ACM Press, USA.
- [28] N. Oh, P. P. Shirvani, and E.J. McCluskey, "Error detection by duplicated instructions in superscalar processors," *IEEE Transactions on Reliability*, 51, pp. 63–75, 2002.
- [29] R. Sengupta, J. D. O.enberg, D. J. Fixsen, D. S. Katz, P. L.Springer, H. S. Stockman, M. A . Nieto-Santisteban, R. J. Hanisch and J. C. Mather, "Software fault tolerance for low-to-moderate radiation environments," *Astronomical Data Analysis Software and Systems X ASP Conference Series*, Vol. 238, 2001, Eds: F. R. Harnden Jr., F. A. Primini, and H. E. Payne.
- [30] M. Banatre, and P. Lee, "Hardware and software architectures for fault tolerance," Springer Verlag, 1994.
- [31] G.K. Saha, "Application semantic driven assertions toward fault tolerant computing," *ACM Ubiquity*, 7(22), June 2006, ACM Press, USA.
- [32] G. K. Saha, "Algorithm based fault tolerant computing for a scientific application," *International Journal – SAMS*, 34(4), pp. 509-523, 1999, Gordon & Breach, USA.
- [33] Y. Huang and C. M. R. Kintala, "Software implemented fault tolerance: technologies and experience," *Proceedings of 23<sup>rd</sup> International Symposium on Fault Tolerant Computing (FTCS-23)*, pp.2-9, June 22-24, 1993.
- [34] P. P. Shirvani, N. Saxena and E. J. Mccluskey, "Software-implemented EDAC protection against SEUs," *IEEE Transactions on Reliability*, 49. 273–284, 2000.

[35] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, and D.I. August, "SWIFT: Software implemented fault tolerance," Proceedings of the 3rd International Symposium on Code Generation and Optimization, 2005.

[36] G. A. Reis, J. Chang, N. Vachharajani, R. Rangan, D. I. August, and S. S. Mukherjee, "Design and evaluation of hybrid fault-detection systems," Proceedings of the 32th Annual International Symposium on Computer Architecture, 148–159, 2005.

[37] Y. Yeh, "Triple-triple redundant 777 primary flight computer," Proceedings of the 1996 IEEE Aerospace Applications Conference, 1, 293–307, 1996.

[38] Y. Yeh, "Design considerations in Boeing 777 fly-by-wire computers," Proceedings of the Third IEEE International High-Assurance Systems Engineering Symposium. 64 – 72, 1998.

[39] M.-C. Hsueh, T. K. Tsai, and R. K. Iyer. Fault Injection Techniques and Tools. *IEEE Computer*, 30(4), pp. 75-82, April 1997.

[40] B. Nicolescu and R. Velazco, "Detecting soft errors by a purely software approach: method, tools and experimental results," Proceedings of DATE03, 2003.

[41] O. Goloubeva, M. Rebaudengo, M.S. Reorda and M. Violante, "Soft-error detection using control flow assertions," Proceedings of the DFT03, 2003.

[42] G. K. Saha, "Software based fault tolerant computing," ACM Ubiquity, 6(40), 2005, ACM Press, USA.

[43] Z. Alkhalifa, V. Nair, N. Krishnamurthy, and J. Abraham, "Design and evaluation of system-level checks for on-line control flow error detection," IEEE Transactions on Parallel and Distributed Systems, 10(6), 1999, pp.627-641.

software fault-tolerance.

[44] R.Kumar and V.K.Garg, "Modeling and Control of Logical Discrete Event Systems," Kluwer Academic Publishers. 1995.

- [45] K-Y Cai *and* X-Y Wang, "Towards a Control-Theoretical Approach to Software Fault-Tolerance," Proceedings of the QSIC04, 2004.
- [46] B. Randell, "System structure for software fault tolerance," IEEE Transactions on Software Engineering, SE-1(2), 1975, pp.220-232.
- [47] A. Avizienis, "Fault tolerant systems," IEEE Transactions on Computer, C-25, 1976, pp.1304-1312.
- [48] M. Lyu, (editor), "Software fault tolerance," John Wiley & Sons, 1995.
- [49] M. Hiller, "Software fault-tolerance techniques from a real-time systems point of view – an overview," Technical Report No.98-16, Department of Computer Engineering, Chalmers University of Technology, Sweden, 1998.
- [50] P. Ammann, J. C. Knight, "Data diversity: an approach to software fault tolerance," IEEE Transactions on Computer, 37(4), 1988, pp. 418-425.
- [51] D. E. Eckhardt, and L. D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors," IEEE Transactions on Software Engineering, SE-11(12), Dec. 1985, pp. 1511-1517
- [52] J. C. Knight, and N. G. Leveson, "An experimental evaluation of the assumption of independence in multiversion programming," IEEE Transactions on Software Engineering, SE-12(1), 1986, pp. 96-109.
- [53] T. Anderson, P.A. Barrett, D.N. Halliwell, and M.R. Moulding, "An evaluation of software fault tolerance in a practical system," Proceedings of the FTCS-15, pp. 140-145.
- [54] J-C. Laprie, "Dependable computing and fault tolerance: concepts and terminology," Proceedings of the FTCS-15, pp. 2-11.
- [55] T. Gustafsson, H. Hallqvist and J.Hansson, "A similarity-aware multiversion concurrency control and updating algorithm for up-to-date snapshots of data," Proceedings of the 17th Euromicro Conference on Real-Time Systems (ECRTS'05), 2005.

- [56] S. Krishnaswamy, I. L. Markov and J. P. Hayes, "Logic circuit testing for transient faults," Proceedings of the European Test Symposium (ETS'05), 2005.
- [57] F. Como, F. Esposito, M. Soma Reorda and S. Tosato, "Evaluating the effects of transient faults on vehicle dynamic performance in automotive systems," Proceedings of the ITC International Test Conference, 2004.
- [58] Y. Monnet, M. Renaudin and R. Leveugle, "Asynchronous circuits transient faults sensitivity evaluation," Proceedings of the DAC2005, June 13-17, 2005, pp. 863-868, Anaheim, California, USA.
- [59] M. Bellato, P. Bemardi, D. Bortolato, A. Candelori, M. Ccschia, A. Paccagnella, M. Rebaudengo, M. Sonza Reorda, M. Violante and P. Zambolin, "Evaluating the effects of SEUs affecting the configuration memory of an SW – based FPGA," Proceedings of the Design Automation and Test in Europe Conference (DATE04), 2004, pp. 584-589.
- [60] D. Alexandrescu, L. Anghel and M. Nicolaidis, "Simulating single event transients in DVSM ICs for ground level radiation," Proceedings of the 3rd IEEE Latin American Test Workshop (LATW'OZ), Montevideo, Uruguay, February 10-13, 2002.
- [61] A. Li and B. Hong, "A Low-Cost Correction Algorithm for Transient Data Errors," ACM Ubiquity, 7(21), 2006.
- [62] D. Ilić and E. Troubitsyna, "Formal Development of Software for Tolerating Transient Faults," Proceedings of the 11th Pacific Rim International Symposium on Dependable Computing (PRDC'05), 2005.
- [63] A. Costes, J. E. Doucet, C. Landrault and J. C. Laprie, "SURF a program for dependability evaluation of complex fault-tolerant computing systems," Proceedings of FTCS-25, Volume 111, 1996.

- [64] P.M. Townend, J. Xu and M. Munro, "Building dependable software for critical applications: N-version software versus one good version," Technical Report, No. 400, Dept. of Computer Science, University of Durham, August 2000.
- [65] D.F. McAllister and M.A. Vouk, "Fault-tolerant software reliability engineering," Handbook of Software Reliability Engineering, (Ed. M.R. Lyu), McGraw-Hill, pp.567-614, 1996.
- [66] P. Popov, L. Strigini and B. Littlewood, "N-version design versus one good version: what does the evidence show," Proceedings of the International Conference on Dependable Systems and Networks, Fast Abstract, pp. B42-B43, June 2000.
- [67] L. Hatton, "N-version design versus one good version," IEEE Software, 14(6), pp.71-76, 1997.
- [68] M.K. Joseph, "Architectural Issues in Fault-Tolerant Secure Computing Systems," Ph.D. Dissertation, Dept. of Computer Science, UCLA, 1988.
- [69] J.P.J. Kelly, T.I. McVittie and W.I. Yamamoto, "Implementing design diversity to achieve fault tolerance," IEEE Software, pp. 61-71, 1991.
- [70] P. Townend, J. Xu and M.Munro, "Building Dependable Software for Critical Applications: Multi-Version Software versus One Good Version," Proceedings of the 6<sup>th</sup> International Workshop on Object-Oriented Real-Time Dependable System (WORDS'01), 2001.
- [71] G. K. Saha, "Transient fault tolerance using single - version algorithm," ACM Ubiquity, 6(28), 2005, ACM Press.
- [72] J. Aidemark, P. Folkesson and J. Karlsson, "On the probability of detecting data errors generated by permanent faults using time redundancy," Proceedings of the 9th IEEE International On-Line Testing Symposium (IOLTS'03), 2003.

- [73] J. Aidemark, J. Vinter, P. Folkesson and J. Karlsson, "Experimental evaluation of time-redundant execution for a brake-by-wire application", Proceedings of the Int'l. Conf. on Dependable Systems and Networks, Washington DC, USA, 2002.
- [74] J. Aidemark and O. Askerdal, "Use of Time Redundancy for Detection of Data Errors caused by Non-Transient Faults," Report No. 03-09, Department of Computer Engineering, Chalmers University of Technology, Sweden, 2003.
- [75] Y. Zhang and K. Chakrabarty, "Fault Recovery Based on Checkpointing for Hard Real-Time Embedded Systems," Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03), 2003.
- [76] K. M. Chandy, J. C. Browne, C. W. Dissly and W. R. Uhrig, "Analytic models for rollback and recovery strategies in database systems," IEEE Transactions on Software Engineering, 1, pp. 100-110, March 1975.
- [77] S. W. Kwak, B. J. Choi and B. K. Kim, "An optimal checkpointing-strategy for real-time control systems under transient faults," IEEE Trans. Reliability, 50, pp. 293-301, September 2001.
- [78] C. Constantinescu, "Experimental Evaluation of Error-Detection Mechanisms," IEEE Transactions on Reliability, 52(1), pp. 53-57, March 2003.
- [79] A. Ejlali, B. M. Al-Hashimi, M.T. Schmitz, P. Rosinger, and S. G. Miremadi, "Combined time and information redundancy for SEU-tolerance in energy-efficient real-time systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 14(4), APRIL 2006, pp. 323-335.
- [80] A. Orailoglu and R. Karri, "Automatic Synthesis of Self-Recovering VLSI Systems," IEEE Transactions on Computers, 45(2), February 1996, pp. 131-142.
- [81] K.H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," IEEE Transactions on Computers, 33, pp. 518-528, December 1984.

- [82] S. Yau and F. Chen, "An approach to concurrent control flow checking," IEEE Transactions on Software Engineering, SE-6(2), 1980, pp.126-137.
- [83] G. K. Saha, "A software tool for fault tolerance," Journal of Information Science & Engineering, 22(4), July 2006.
- [84] E. Normand, "Single event effects in avionics," IEEE Transactions on Nuclear Science, 43(2), pp. 461-474, April 1966.
- [85] C. Weaver, J. Emer, S. S. Mukherjee and S.K. Reinhardt, "Techniques to reduce the soft error rate of a high-performance microprocessor," Proceedings of the 31<sup>st</sup> Annual International Symposium on Computer Architecture, 2004
- [86] G. K. Saha, "A single – version scheme of fault tolerant computing," Journal of Computer Science & Technology, 6(1), pp. 22-27, 2006.
- [87] M.A. Gomma and T.N. Vijaykumar, "Opportunistic transient – fault detection," Proceedings of the 32<sup>nd</sup> International Symposium on Computer Architecture (ISCA'05), 2005.
- [88] G. K. Saha, "Transient fault tolerance through algorithms," IEEE Potentials, 25(5), Sep-Oct 2006, IEEE Press, USA.
- [89] S.K. Reinhardt and S.S. Mukherjee, "Transient fault detection via simultaneous multithreading," Proceedings of the 27<sup>th</sup> ISCA'00, June 2000.
- [90] G. K. Saha, "Software implemented hardware – transient fault detection," International Scientific Journal of Computing, 5(1), February 2006.
- [91] J.H. Patel and L.Y. Fung, "Concurrent error detection on ALU's by recomputing with shifted operands," IEEE Transactions on Computers, 31(7), pp.589-595, 1982.
- [92] C. da Lu and D.A. Reed, "Assessing fault sensivity in MPI applications," Supercomputing, Pittsburgh, Pennsylvania, November 2004.

[93] T. P. Ma and P. Dussendorfer, "Ionizing radiation effects in MOS devices and circuits," Wiley, New York, 1989.

**Source: Ubiquity Volume 7, Issue 25 (July 5, 2006 - July 10, 2006 )**

Goutam Saha is an associate editor of Ubiquity.



