Performance Enhancement by Splitting ALU in Error Resilient Low Cost Processors

Naveen Kumar
YMCA University of Science and Technology
Faridabad, India

Rahul Raj Choudhary
Govt. Engineering College Bikaner
Bikaner, India

Pradeep Dimri
YMCA University of Science and Technology
Faridabad, India

ABSTRACT
Performance enhancement has always been a primary design goal for designers. Modern automation technology demands for reliable low cost controllers, required for specific applications. The restricted resources limits the number of functional units available on a processor which results into lowered performance. The better performance can be achieved with splitting existing functional unit into smaller independent units. The split of arithmetic and logic unit into two independent units namely arithmetic unit and logic unit, provides the facility for simultaneous operation by both of the units. This results into lesser execution time. The simulator Simplescalar has been modified to simulate split ALU. The benchmarks are run in order to collect performance statistics. The simulation parameters, including execution time have been recorded for standard simulator having integral ALU unit, and for split ALU unit and further compared to show enhanced performance with split ALU.

Keywords
Reliable low cost controllers, functional units, split ALU, simulation parameters.

1. INTRODUCTION
Rapid advances in VLSI technology and aggressive design methodologies are resulting into successful realization of industrial automation and many low power, portable computing devices. Modern embedded automation systems are built around an extremely low cost processors in order to cater inexpensive automation solutions for specific operations. The cost constraint of these processors restricts the available silicon area and further the number of available functional units. However, the demand for reliability and enhanced computing speed is growing, as usual, exponentially. Though the centralized and integrated control systems solutions are readily available in the form of distributed control systems (DCS) and SCADA, but such solution requires very expensive data acquisition network as well as very expensive control circuitry comprising very costly processors. Such costly processors can accommodate vast number of functional units in order to attain exceptional operating speed as well as high degree of reliability. The cost and complexity of such automation solutions are affordable only by big industrial organizations. The recent development in Embedded Technology provides the cheaper automation solutions which are not only low cost solutions but also extremely less complex for installations. Such solutions are targeted for smaller and specific control applications. This has opened up a new market for low cost small processors.

Such demand of lower cost with better reliability and enhanced performance has posed new challenges for designers. Most of the reliability solutions are based upon redundancy, which impose either silicon area or performance penalty. The need for enhancement of speed under restricted conditions for resources compels designers for rearrangement of different sections of processors. The splitting and floating nature of resources can enhance the processor operating speed [1]. A functional unit is designed as an integral unit and can serve a single instruction at a time. Though there may be some part of a functional unit which is not at all required during execution of an instruction, even though such hardware cannot be assigned for another instruction during that execution. Such unutilized hardware could be utilized if the functional units are split in smaller independent units. The arithmetic and logic unit is a unit which serves execution of arithmetic as well as logical instructions. The superscalar architecture provides the facility for simultaneous execution of multiple instructions in order to reduce the execution time. In such environment, the split ALU in arithmetic unit AU and logic unit LU can execute two instructions simultaneously. Such split would reduce the execution time very effectively.

This paper focuses the simulation of splitting the ALU in two independent units. The performance is further evaluated by running commercially available benchmarks. The paper is organized as follows. Section 2 describes the previous work in the domain. Section 3 describes an overview of Simplescalar Simulator. Section 4 presents the Simulation statistics of benchmarks. Finally, we conclude with section 5.

2. PREVIOUS WORK
The error resilience requires detection of errors before final system crash. The provision of error detection is generally incorporated by comparison of redundant outputs. The inclusion of redundancy has proved to be major phenomena to deal with errors. Since the low cost processor are already running short of silicon area, these cannot afford to have spatial redundancy. The time redundancy is only the left solution for such processors.

The hardware duplication technique Triple Modular Redundancy (TMR) (2/3 logic) uses a majority voting strategy and provides fairly good fault tolerance capability, but it has 200 percent hardware overhead.

Time Redundant Micro-architectures like ReStore [3], REESE [8], AR-SMT [2] are fault tolerant. AR-SMT micro-architecture implements program level time redundancy with threaded re-execution. This approach proposes the use of a test threads on simultaneous multi-threaded processor to provide a means of detection of life time failures. The AR-SMT technique focuses super-scalar architectures which supports simultaneous multithreading. Symptom Based Soft Error Detection in Microprocessors (ReStore) Architecture [3]

is also augmentation of modern high performance processor. This uses a combination of short term on-chip architectural checkpoints with symptom - based error detection and focus to achieve fault tolerance in superscalar architectures. Redundant Execution Using Spare Element (REESE) executes every instruction two times by interleaving P and R streams instruction executions. High performance penalty is expected by REESE architecture. These Architectures are suitable only for superscalar machines where enough resources are available for re-computing the work. The architecture proposed by Sohi et al [7] recommends for each instruction to have duplicate instruction for re execution purpose and handles only functional unit errors. Metra et al [4] have proposed a hidden code based-technique to deal with the soft errors in controller. The inherited redundancy by microprocessor control logic is discovered and utilized for on-line testing at low cost. However, this technique is limited to the controller circuitry. Naseer [6] analysed different soft error mitigation techniques. Micro architecture-Based Introspection (MBI) [6] proposes the utilization of wasted processing bandwidth during long-latency cache misses for redundant execution of the instruction stream.

This approach suffers high performance over head in the form of pipeline-fill-penalty and forced introspection penalty. Also hardware over head in the form of storage cost for back log buffer is very prominent. Kim and Somani [9] proposed SSD technique while Shamsiri et al[10] also proposed the instruction level test methodology.

The Floating Resources and Extended Pipeline (FREP) architecture [1] proposes the split of resources and further make them available at any pipeline stage in order to implement simultaneous execution of instructions in pipelined architecture.

3. OVERVIEW OF SimpleScalar SIMULATOR

This work is aimed at performance testing of superscalar processors using modified SimpleScalar [11] Simulator. The objective is to generate test statistics using standard super scalar architecture and further split ALU architecture.

SimpleScalar is an open source computer architecture simulator. This is a set of tools that model a virtual computer system with CPU, Cache and Memory Hierarchy. Researchers can build modeling applications that simulate real programs running on a range of modern processors and systems with a use of this simulator. The tool set is comprised of various simulators ranging from a fast functional simulator to a detailed, dynamically scheduled processor model that supports non-blocking caches, speculative execution, and state-of-the-art branch prediction which also includes performance visualization tools, statistical analysis resources, and debug and verification infrastructure. The simulator can simulate Alpha and PISA (Portable ISA). The PISA instruction set is a simple MIPS-like instruction set maintained primarily for instructional use. The compiled binaries are submitted to the SimpleScalar architecture which simulates their execution on one of several provided processor simulators.

The specific simulator Sim-outorder is used and modified for this work which is detailed micro-architectural simulator. Sim-outorder models in detail and out-of-order microprocessor with all of its features, including branch prediction, caches, and external memory. The 3900 line highly parameterized simulator code can emulate machines of varying numbers of execution units.

4. SIMULATIONS CARRIED OUT

For simulations Sim-outorder simulator is used for detailed and rigorous testing. Two configurations for standard ALU based architecture and two configurations for split ALU architecture have been used in order to have in depth statistics. For standard ALU the original Sim-outorder simulator code was used as it was and number of ALUs were selected one and four for two different configurations. For split ALU architecture, the source code of Sim-outorder was modified and in addition to that the Instruction Set Architecture file (ISA) was also modified to deal the instruction with newer set of functional units.

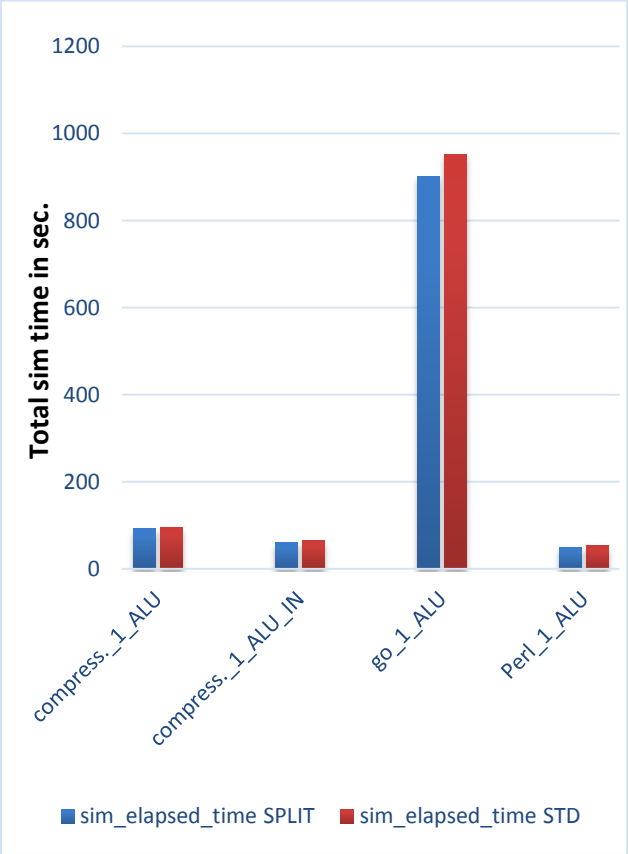

Figure 1. Total time elapsed in seconds with single ALU

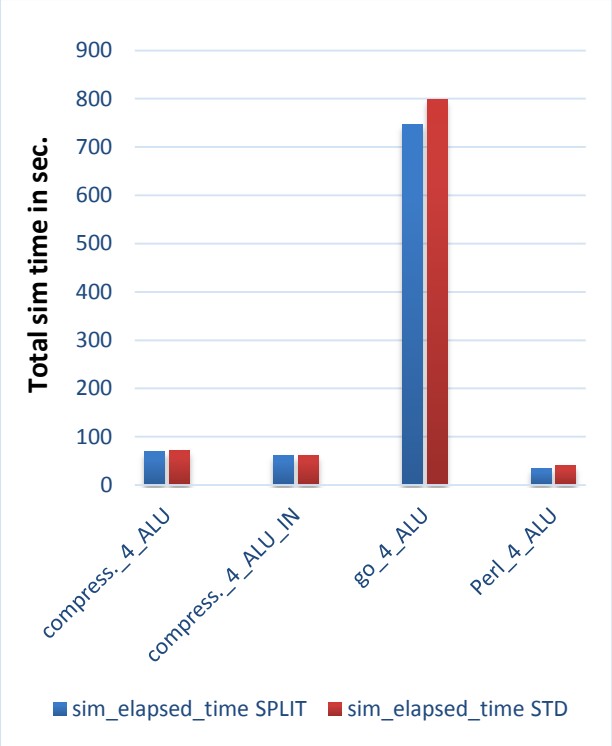

Figure 2. Total time elapsed in seconds with 4ALU

Simulations show that average 5.3% simulation time is reduced with split ALU in case of single ALU configuration where as in case of four number of ALUs are selected an average 6.2% simulation time is reduced. This is depicted in Figure 1 and 2 respectively.

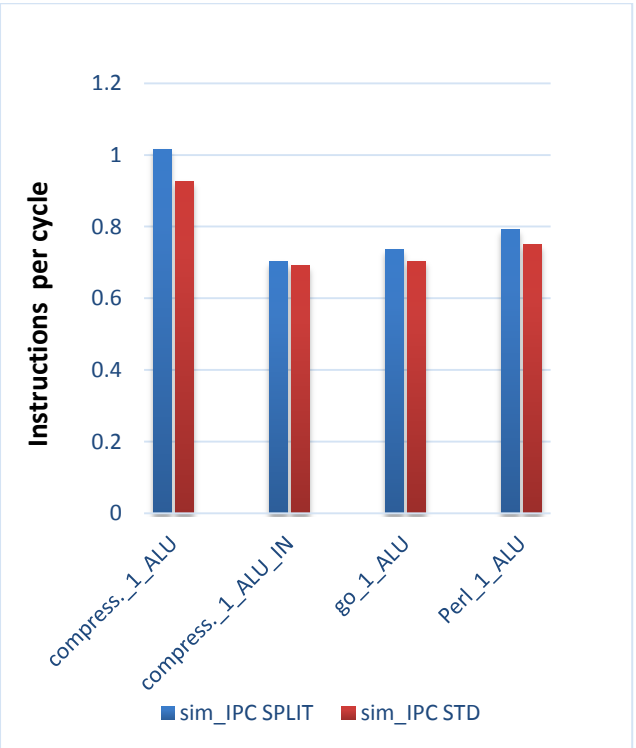

Figure 3. IPC count with single ALU

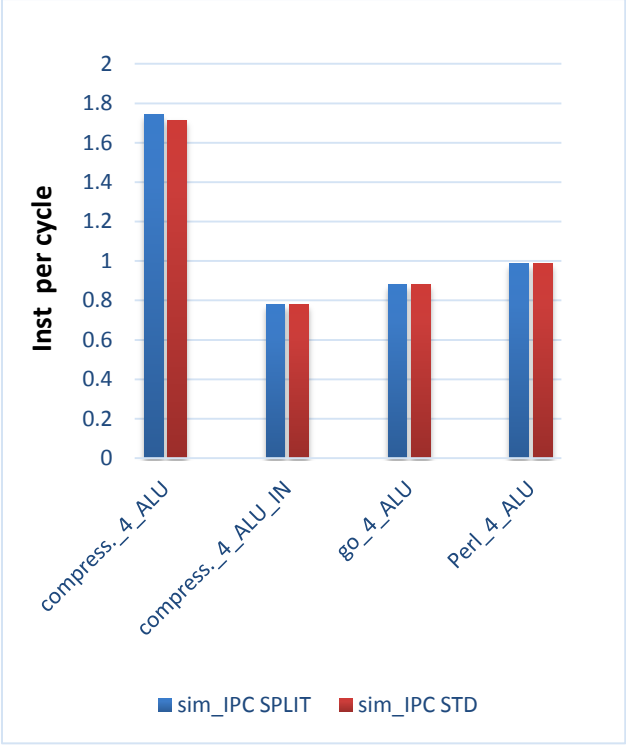

Figure 4. IPC count with four ALUs

Simulation shows that executed instruction per cycle is also enhanced by 5.8 percent average with single ALU configuration while 0.6 percent enhancement observed with four ALU configuration. This is illustrated in Figure 3 and 4.

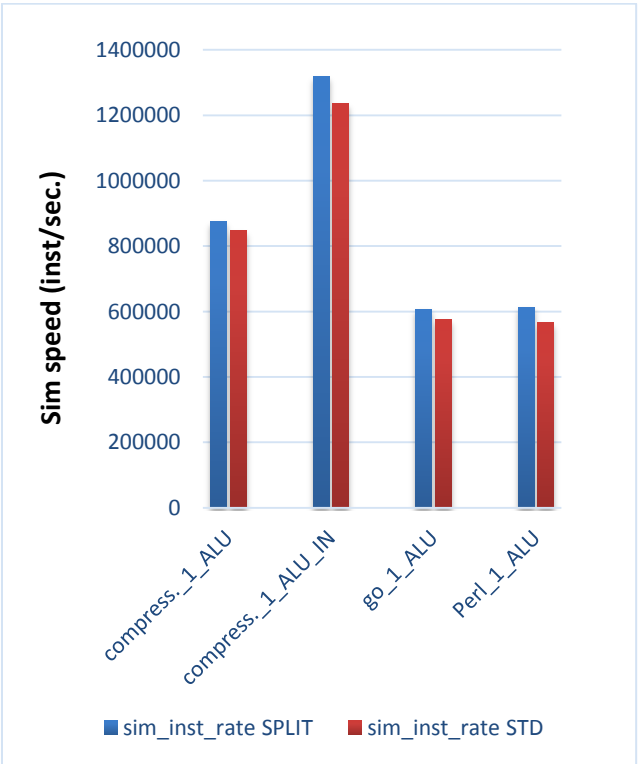

Figure 5. Simulator speed with single ALU

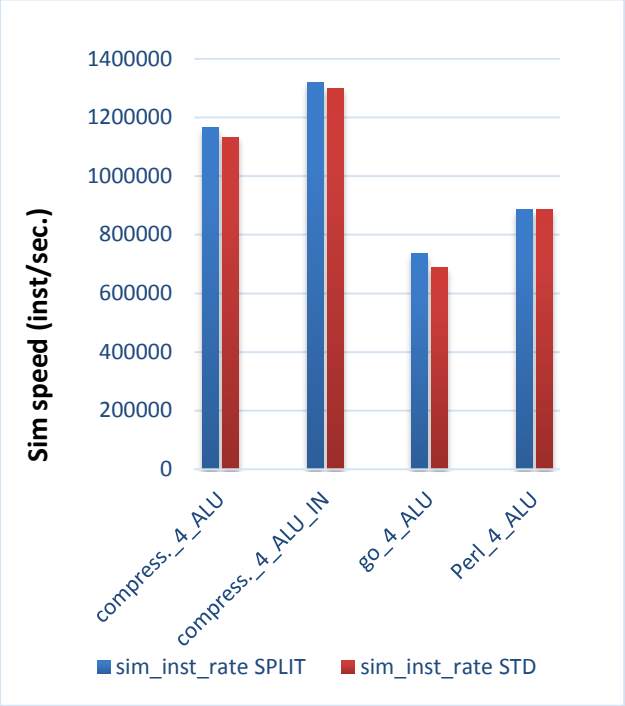

Figure 6. Simulator speed with four ALU

Improvement in average Simulation speed i.e. instructions executed per second is also observed with both of configurations. Figure 5 and 6 show that executed instruction per second is enhanced by 5.4 percent average with single ALU configuration while 2.4 percent enhancement observed with four ALU configuration.

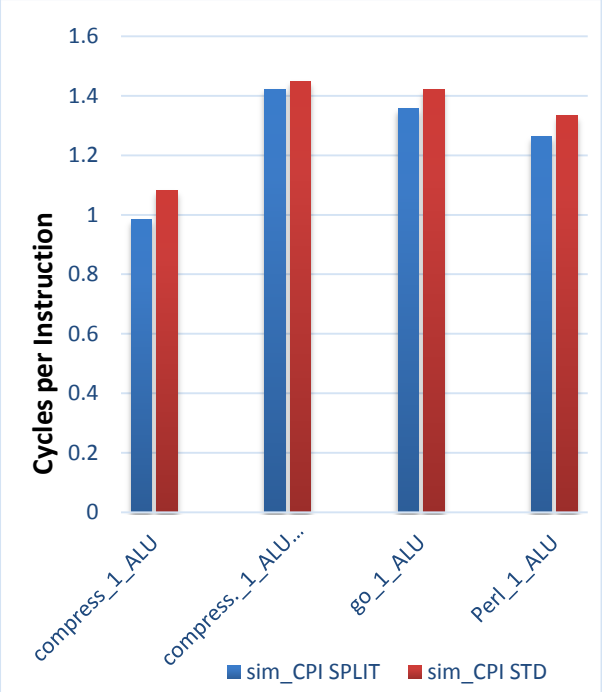

Figure 7. Cycles used per Inst. with single ALU

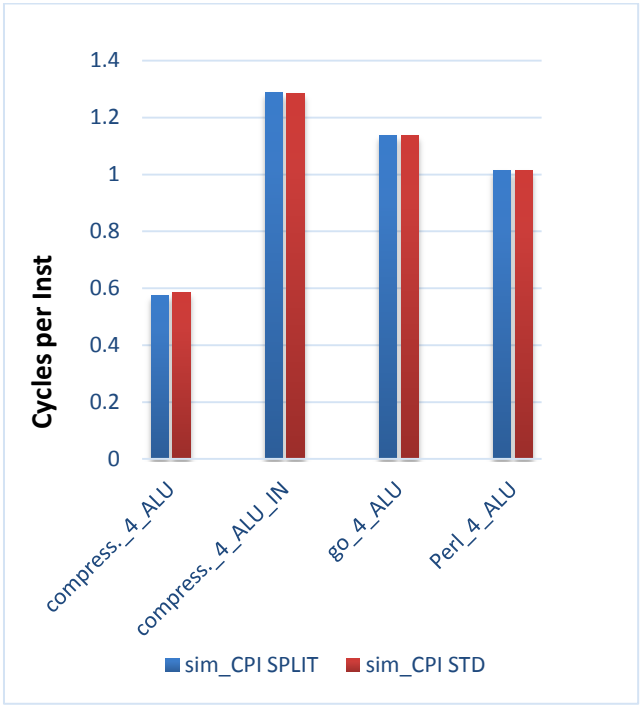

Figure 8. Cycles used per Inst. with four ALUs

Simulation shows reduction in used cycles per instruction with both of the configuration using split ALU architecture. The cycles per instruction are reduced by 4.8 percent average with single ALU configuration while 0.1 percent reduction observed with four ALU configuration. This is illustrated in figure 7 and 8.

5. CONCLUSION

This paper highlights issues that are pertinent to performance degradation due to inclusion of reliability and fault tolerance. The paper proposes enhancement of fault tolerance low cost processor by splitting of functional units, in order to deal with multiple instruction simultaneously. The reexamination and modified arrangements of functional units in order to reduce the silicon area as well as enhanced performance is our future goal.

6. REFERENCES

[1] Viney Kumar, Rahul Raj, and Virendra Singh, 'FREP: A Soft-Error Resilient Pipelined RISC Architecture', IEEE East-West Design and Test Symposium (EWDTS) 2009, Moscow, Russia, Sep 2009, pp. 196-199.

[2] Eric Rotenberg, 'AR-SMT: A Microarchitectural Approach to Fault tolerance in Microprocessors', Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing, 1999. Page(s):84 - 91.

[3] Wang N.J. and Patel S.J., 'ReStore: Symptom Based Soft Error Detection in Microprocessors, IEEE Transactions on Architecture, Volume 3, Issue 3, July-Sept. 2006 Page(s): 188 - 201.

[4] C. Metra D. Rossi, M. Omana, A. Jas, and R. Galivanche, 'Function-Inherent Code Checking: A New Low Cost On-Line Testing Approach For High

Performance Microprocessor Control Logic', 13th European Test Symposium, May 2008.

[5] Naseer, Riaz Bhatti, Rashed Zafar Draper, Jeff, 'Analysis of Soft Error Mitigation Techniques for Register Files in IBM Cu-08 90nm Technology',Circuits and Systems, 2006. MWSCAS'06.

[6] Qureshi, M.K.; Mutlu, O. Patt, Y.N., 'Microarchitecture-Based Introspection: A Technique for Transient-Fault Tolerance in Microprocessors', Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference, Volume , Issue , 28 June-1 July 2005 Page(s): 434 - 443

[7] G. Sohi, M. Franklin, and K.K. Saluja, 'A Study of Time-Redundant Fault Tolerance Techniques for High-Performance Pipelined Computers', Nineteenth International Symposium on Fault-Tolerant Computing, FTCS-19, Jun 1989 Page(s):436 -443.

[8] J.B. Nickel, and A.K. Somani, 'REESE: A Method of Soft Error Detection in Microprocessors', Proc. of Int. Conf. on Dependable Systems and Networks, 2001.

[9] S. Kim, A. K. Somani, 'SSD: an Affordable Fault Tolerant Architecture for Superscalar Processors', in Proc. of Int. Symp. on Dependable Computing, pp. 27-34, 2001.

[10] S. Shamshiri, H. Esmaeilzadeh, and Z. Navabi, 'Instruction-level test methodology for CPU core self-testing', ACM Trans. Des. Autom. Electron. Syst., vol. 10, no. 4, pp. 673689, Oct. 2005.

[11] SimpleScalarR, SimpleScalar LLC toolset by Todd Austine, www.simplescalar.com