# Tunable Fault Tolerance via Test and Reconfiguration

R. D. (Shawn) Blanton*, Seth Copen Goldstein†*, Herman Schmit*
*Dept. of Electrical and Computer Engineering
†Dept. of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
{blanton,seth,herman}@ece.cmu.edu

## 1.0 Introduction

The advent of reconfigurable hardware as a computing medium creates a multi-dimensional design space where trade-offs between power, speed, cost, and reliability are now possible. Dynamic reconfiguration allows these trade-offs to be made on-line to adapt to various environmental changes. Furthermore, the cost of designing reliable circuits is greatly reduced since varying levels of reliability can be added to any circuit, even if it was not originally designed to run reliably, either at the time of application start-up or during application execution.

This paper briefly discusses two ways that reconfiguration and hardware virtualization [1,2] can be used to produce or modify the level of circuit reliability. The first method addresses Single-Event-Upset (SEU) faults by allowing the level of redundancy in the circuit to be dynamically varied. Detection and retry mechanisms provide a level of reliability similar to that of triple modular redundancy, at a cost of only two times (instead of the traditional three times) the original hardware or 50% performance. The second technique described provides reliability in the presence of local hardware faults. This technique works by rotating a built-in self-test (BIST) operation around the reconfigurable hardware periodically, while continuing to execute the application.

Both methods may be used to modulate the amount of reliability provided on an application-by-application basis or during application execution. The former case is used when a single field programmable gate array (FPGA) is used for many different applications and applies to both current commercial FPGAs as well as more modern architectures. When an application is loaded on the reconfigurable hardware, a level of reliability is specified. The latter case, which is more powerful, is suited only to newer architectures which support high-speed reconfiguration such as PipeRench [1,2].

## 2.0 Variable Fault Tolerance

Figure 1 illustrates our approach to addressing SEU failures through varying levels of redundancy. In this scenario, applications that require high performance and/or low power and no fault tolerance are represented by the single module of Figure 1a. Here, the original design is configured by the fabric controller and compiler for optimal performance. Applications that require some degree of fault tolerance can be implemented as shown in Figure 1b. This mode of operation (termed error-detect mode) requires that the original module be duplicated and operated in parallel with the original. A comparison module is also added to detect any discrepancies between the two module outputs. The duplication of the module and the addition of the comparator can take place at configuration time using the original circuit (from Figure 1a). For the highest level of fault-tolerant operation the circuit can be configured to use N-way modular redundancy (NMR) as shown in Figure 1d. Any of these three modes can be specified at application load time.

Figure 1c shows a more dynamic method of providing N-way modular redundancy without requiring N copies of the module to be present at all times. The circuit is originally configured in error-detect mode (Figure 1b). If an error is detected in the circuit, the fabric is reconfigured to operate in NMR mode and the failed computation is retried, as illustrated in Figure 1c. After all N circuits agree on the result for some time, the circuit is reconfigured back to error-detect mode. We call this method error-detect, rollback, and N-way modular redundancy (EDRNMR).

EDRNMR requires a reconfiguration time that is less than the maximum latency allowed by the system. This will preclude the use of commercial FPGAs, which have typical reconfiguration times in the range of 1 to 100 milliseconds. Newer architectures such as PipeRench [1,2] can completely reconfigure in less than a half a microsecond, making this technique feasible for many real-time systems. This technique will allow for reliability that approaches NMR, while usually requiring only the hardware associated with error-detect mode.

Hardware virtualization enables the execution of hardware designs that exceed the capacity of the device by time multiplexing the design. In PipeRench, hardware virtualization is made possible by reconfiguring a pipeline stage of an application in a single clock cycle. This allows a pipeline of any depth to execute on any size PipeRench. Thus, a given fabric of fixed size will support larger designs, but at lower performance.

Without hardware virtualization, all of the actual hardware is required to meet the performance requirements at the highest level of fault tolerance. Moreover, when lower levels of reliability are acceptable, it is difficult to take advantage of the extra hardware. With virtualized hardware, the performance of a smaller hardware

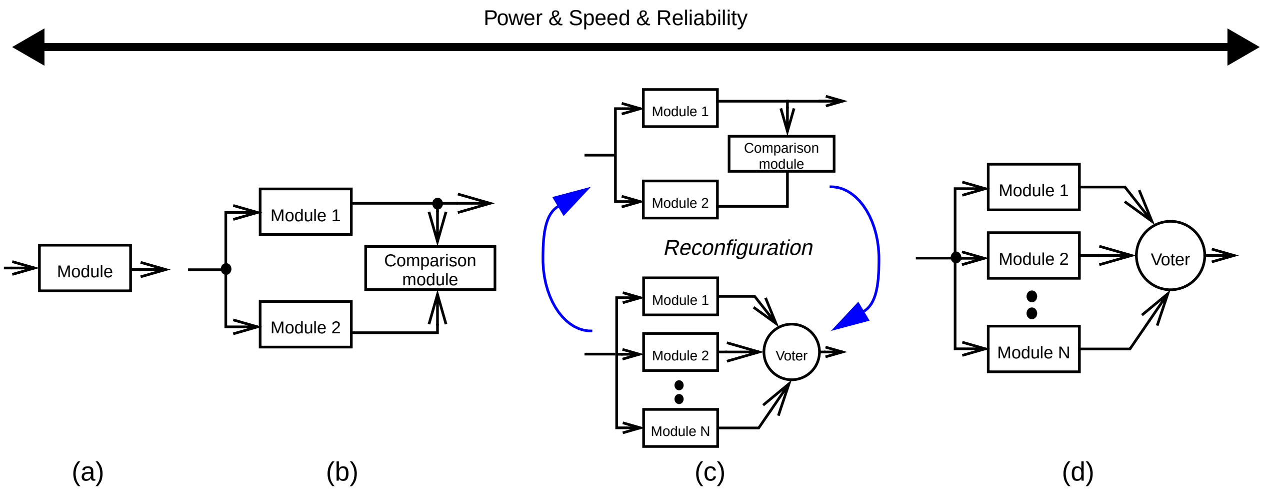

**Figure 1: Reconfigurable hardware enables a dynamic environment for choosing various operation points that optimize some combination of power, speed, and fault tolerance.**

design, as in Figure 1b, will have higher throughput at the same clock cycle as the design in Figure 1d. A quadratic reduction in power can therefore be obtained by lowering the operating voltage of the device to the point at which its throughput requirement is met.

## 3.0 FEBIST

We also envision new and highly cost-effective methods for tolerating local hardware faults through frequently executed built-in self test (FEBIST). Reconfigurable hardware is inherently regular in nature. Both the logic (LUTs, CLBs, etc.) and the interconnect are highly uniform. This regular structure combined with the reconfigurability aspects of the fabric make the BIST and diagnosis of the fabric both quick and efficient [3,4]. Some applications may have sufficient time slack that allows the fabric's test aspects to be exploited to achieve fault tolerance. In this scheme, alternating portions of the fabric are frequently reconfigured for a BIST session which: (1) Ensures the tested fabric is free of hard failures. (2) In the presence of failures, identifies the smallest diagnosable piece of the faulty fabric (CLB or interconnect segment). The diagnostic results of the BIST session can then be used to configure around the faulty fabric. Note that both application modules and circuitry dedicated to fault tolerance (voter, comparator, etc.) can have faulty fabric switched out. This switching out of faulty fabric restores the reliability of the application once a failure has been discovered, hence increasing the overall reliability.

The frequency of the FEBIST determines the level of fault tolerance achieved. If FEBIST sessions are executed often, than the likelihood of a hard failure going undiscovered and causing a system error is less probable. Less frequent FEBIST establishes lower levels of fault tolerance but increases performance. Once again, hardware virtualization achieves additional levels of reliability with either additional hardware, less performance, more power, or some combination of these three attributes.

## 4.0 Conclusions

In this paper, we have briefly outlined the benefits of using dynamically reconfigurable virtualized hardware to implement fault tolerant systems. We believe this technology allows for new trade-offs between hardware cost, power, performance and reliability. Because reconfigurable hardware can be modified dynamically to provide various levels of reliability, it is possible to obtain levels of reliability associated with NMR operation with only slightly more than twice the minimal hardware. Since the duplication of hardware and addition of comparison and voting hardware can take place at configuration time, the cost of designing hardware-based fault tolerance is significantly reduced. Because the level of fault-tolerance can be determined after the design is complete, designers need not calculate the required redundancy a priori. Finally, varying levels of FEBIST can be used to assure that a hardware failure does not lurk deep inside a design, producing erroneous results over long periods of time.

While the benefits of EDRNMR and to some extent FEBIST rely on fast reconfiguration and hardware virtualization, the ability to dynamically change the reliability of a circuit after it has been designed and in response to the environment can be applied to many commercial FPGAs.

## 5.0 References

[1] S. Cadambi, J. Weener, S. C. Goldstein, H. Schmit, D. E. Thomas, "Managing Pipeline-Reconfigurable FPGAs," *Proc. ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, 1998.

[2] H. Schmit, "Incremental Reconfiguration for Pipelined Applications," *Proc. of the IEEE Symposium on FPGAs for Custom Computing Machines*, pp. 47-55, 1997.

[3] C. S. Stroud, E. Lee, M. Abramovici, "BIST-Diagnostics of FPGA Logic Blocks," *Proc. of the 1997 IEEE International Test Conference*, pp. 539-547, 1997.

[4] C. S. Stroud, E. Lee, M. Abramovici, "Using ILA Testing for BIST in FPGAs," *Proc. of the 1996 IEEE Interanational Test Conference*, pp. 68-75, 1996.