# Analysis of On-chip Interconnection Network Interface Reliability in Multicore Systems

Yong Zou, Yi Xiang, Sudeep Pasricha

Colorado State University, Fort Collins, CO, U.S.A
yong.zou@colostate.edu, yix@colostate.edu, sudeep@colostate.edu

***Abstract***— In Networks-on-Chip (NoC), with ever-increasing complexity and technology scaling, transient single-event upsets (SEUs) have become a key design challenge. In this work, we extend the concept of architectural vulnerability factor (AVF) from the microprocessor domain and propose a network vulnerability factor (NVF) to characterize the susceptibility of NoC components such as the Network Interface (NI) to transient faults. Our studies reveal that different NI buffers behave quite differently on transient faults and each buffer can have different levels of inherent fault-tolerant capability. Our analysis also considers the impact of thermal hotspot mitigation techniques such as frequency throttling on the NVF estimation.

## I. INTRODUCTION

With the rising number of cores on the die, the complexity of the interconnection network-on-chip (NoC) fabric has been on the rise. Some previous works have addressed the problem of making NoC fabrics more reliable. For instance, Pasricha et al. [2] presented a turn model based fault tolerant routing algorithm that provides resilience against intermittent and permanent faults at runtime on NoC links and routers. However, fault tolerance for some NoC components such as the network interfaces (NIs) has received little attention. NIs are comprised of highly utilized buffers, and are thus increasingly susceptible to soft errors today.

In general, techniques to ensure fault tolerance have become a critical design concern in recent CMP designs. Fortunately, not all faults eventually affect the final program outcome. Mukherjee et al. [1] defined a structure's Architectural Vulnerability Factor (AVF) as the probability that a transient fault in the structure finally produces a visible error in the output of a program. At any point of time, a structure's AVF can be derived via counting all the important bits that are required for Architecturally Correct Execution (ACE) in the structure, and dividing them by the total number of bits of the structure. Such ACE analysis has been used to derive an upper bound on AVF using performance simulation for intra-processor buffers and cache structures in recent years.

In this paper, for the first time, we introduce the concept of a network vulnerability factor (NVF) and perform a detailed NVF analysis for sub-components in a state-of-the-art AXI protocol based NI architecture, using full system simulation running Linux OS. The impact of thermal variations on NVF is also analyzed.

## II. NETWORK INTERFACE ARCHITECTURE

We developed two AXI-based NI architectures: processor NI and memory NI. The processor NI issues access requests to the memory modules. The memory NI interfaces with memory modules, accepting write and read requests from other cores, and does not initiate any requests itself. Table I summarizes details of the 10 NI buffers we analyzed (number of entries, bit width of each entry, and buffer functionality) in our vulnerability analysis.

TABLE I. NETWORK INTERFACE BUFFER DESCRIPTION

| Buffer Name | Size (entries) | Width (bits) | Function |
|---|---|---|---|
| Reorder_Unit (mem) | 32 | 32 | reorder request transaction data |
| Ready_Queue (mem) | 10 | 32 | queue pending memory requests |
| Read_Dat_Get (mem) | 20 | 32 | store received read data from memory |
| axiw_ctl_queue (µp) | 32 | 54 | store write request AXI control signals from processors |
| axiw_dat_queue (µp) | 32 | 46 | store write data from processors |
| axir_ctl_queue (µp) | 32 | 54 | store read request AXI control signals from processors. |
| R_data_queue (µp) | 5 | 32 | store read data from memory |
| Wctl_flit_backup (µp) | 10 | 32 | backup write control signals sent from processor NI |
| Rctl_flit_backup (µp) | 1 | 32 | backup read control signals sent from processor NI |
| Dat_flit_backup (µp) | 15 | 32 | backup data sent from processor NI. |

## III. NETWORK VULNERABILITY FACTOR (NVF)

We use the idea of ACE bits to calculate NVF for the NI buffers. ACE bits in our case are the subsets of buffer entries that contain useful data during flit transmission. Any errors that involve these bits will cause a visible error in the final application results. In contrast, unACE bits do not affect the application results, even if soft errors cause changes to these bits. Then the NVF for a hardware component $\mathcal{H}$ of size $S_{\mathcal{H}}$ bits over an analysis window of $N$ cycles can be expressed as

$$NVF = \frac{\sum_{i=1}^{N} (no.of\ ACE\ bits\ in\ \mathcal{H}\ at\ cycle\ i)}{N \times S_{\mathcal{H}}}$$

which allows us to calculate soft error rate (SER) for $\mathcal{H}$ as: $SER = S_{\mathcal{H}} \times (FIT/bit) \times NVF \times TVF$, where *FIT/bit* is the Failure in Time per bit, and *TVF* is the timing vulnerability factor encapsulating the fraction of each cycle that a bit is vulnerable.

There are several reasons that can lead to the presence of unACE bits in a structure that end up making it less vulnerable to soft errors. The major detectable causes of unACE bits that are relevant to NI components in a NoC fabric are: *(i) Idle*: this is the most basic scenario - when there are no flits or data saved inside a buffer, we consider the buffer to be in an idle state. In this case, even when there are transient errors inside the buffer, as there is no critical information saved inside it at that moment, the final application result will not be affected; *(ii) Write-Write:* this scenario corresponds to a write-after-write event. It may so happen that a data that is propagated through the NI buffers and the NoC is overwritten before it is used. In such a case, even if the data gets corrupted, it does not impact the correctness of the program in any way; *(iii) Read Mask:* often, data that is read into a processor may not be used in its entirety. For instance, during an *xor* operation,

978-1-4577-1954-7/11/$26.00 ©2011 IEEE

some bits may not affect the outcome of the operation. These bits can be considered unACE; *(iv) AXI Protocol:* for specific NoC implementations, it is possible that some of the transferred information by the AXI protocol may not be relevant.

## IV. NVF ESTIMATION FLOW

Figure 1 shows the estimation flow we use to calculate NVF for each NI buffer. We boot up Linux OS on a M5 full system simulator to execute SPLASH-2 benchmarks, and output traces that record time stamps, core IDs, instructions executed, memory access types, destination addresses, and so forth for each memory-related operation. A Python script was developed as a trace analyzer to process the instruction trace for extraction of memory access behavior and to detect masking scenarios. These traces are ported to a cycle-accurate NoC simulator enhanced with our NI models, router architectures, and protocols. Thus, a trace-driven NoC simulation can be executed to obtain NVF for the NI buffers of our interest. To account for thermal effects during runtime (e.g., performance slowdown due to hotspots) in our NVF analysis, M5 was used to generate periodic statistics for McPAT to calculate the power trace of the CMP system, which was sent to Hotspot, to generate the thermal profile of the die. The thermal trace over time was used to identify points where frequency should be scaled to address hotspot scenarios. This scaling information was fed into the trace driven simulation, to generate thermal-variation aware NVF estimates for the NI buffers

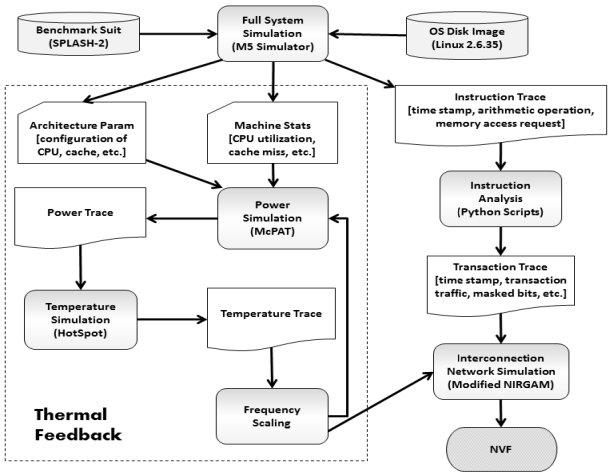


Figure 1. NVF Estimation Flow

## V. EVALUATION STUDIES

We consider a CMP platform with 8 cores and a shared global L2 cache memory, connected together by a 3×3 mesh network. The L1 instruction and data cache sizes are set to 64Kb and the L2 cache is 2048Kb. Every core runs at a baseline frequency of 2GHz. The implementation process technology is 32nm. M5 full-system simulation of parallelized SPLASH-2 benchmarks (due to lack of space, we only present results for *FMM* benchmark) is used to generate traces that are fed into a cycle-accurate NoC simulator. The detailed execution traces were generated for a span of 4 billion cycles, after fast-forwarding for 750 million cycles to account for an initial warm-up period (for OS boot up, etc).

Figure 2 shows the results for the thermal-aware estimation of buffer states. The first three buffers (leftmost three bars) correspond to memory NI buffers, while the remaining seven buffers belong to the processor NI. The figures show the % of cycles that the buffers contain ACE bits and unACE bits. The NVF for each buffer is essentially the % of cycles that a buffer contains ACE bits. It can be seen that NVF varies significantly across buffers in the NI, ranging from 0.35% to 55.75%. Figure 3 presents read masking effect broken down based on the specific instructions that cause the masking, such as shift right logical (*srl*), bit clear (*bic*), etc. The masking effect is most prominent for the *Read_Dat_Get* buffer which holds read data from memory for long periods due to network congestion before packetization and injection. While the same masking states exist for *R_data_queue* in the processor NI that eventually receives the read data from memory, the buffer in general flushes the data to the processor quickly, and thus has a low masked data occupation duration. From the results presented in Figures 2 and 3, most NI buffers can be seen to have a low NVF due to various masking effects. These results can guide the design of high reliability and low power overhead NIs, where only NI buffers with high NVF are protected.

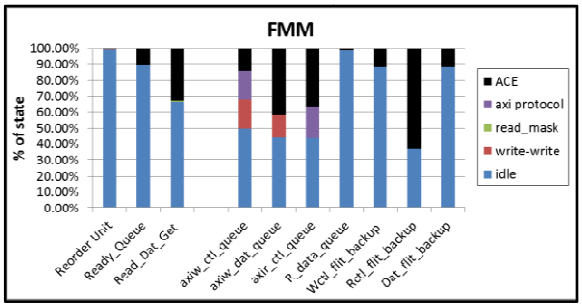


Figure 2. Breakdown of architectural and microarchitectural states for the NI buffers with FMM (NVF for each buffer is % of cycles it contains ACE bits)

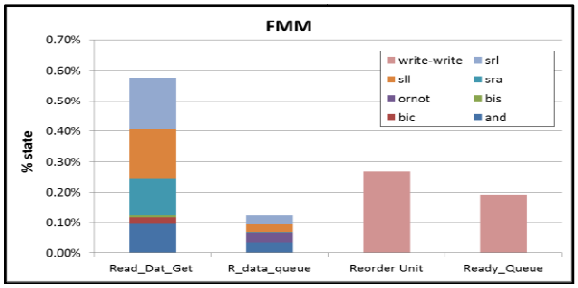


Figure 3. Breakdown of masking for subset of NI buffers with FMM

## VI. CONCLUSION

In this paper, for the first time, a detailed characterization of vulnerability was performed on an AXI-based NI architecture using full system simulation with consideration of thermal throttling, to determine the probabilities that faults in NI components manifest as errors in the final program output of CMP system. The resulting characterization can aid in the design of NI architectures possessing high reliability and low power dissipation overhead, using NVF-based opportunistic protection.

## REFERENCES

[1] S. S. Mukherjee et al., "A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor," Proc. MICRO, pp. 29 – 40, 2003

[2] S. Pasricha, Y. Zou, D. Connors, H. J. Siegel, "OE+IOE: A Novel Turn Model Based Fault Tolerant Routing Scheme for Networks-on-Chip", Proc. IEEE/ACM CODES+ISSS, pp. 85 – 93, Oct 2010