AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors

Eric Rotenberg

Computer Sciences Department
University of Wisconsin — Madison
ericro@cs.wisc.edu
High-Performance Microprocessors

• General-purpose computing success
  - Proliferation of high-performance single-chip microprocessors

• Rapid performance growth fueled by two trends
  - Technology advances: circuit speed and density
  - Microarchitecture innovations: exploiting instruction-level parallelism (ILP) in sequential programs

• Both technology and microarchitecture performance trends have implications to fault tolerance
Technology and Fault Tolerance

• Technology-driven performance improvements
  - GHz clock rates
  - Billion-transistor chips

• High clock rate, dense designs
  - low voltages for fast switching and power management
  - high-performance and “undisciplined” circuit techniques
  - managing clock skew with GHz clocks
  - pushing the technology envelope potentially reduces design tolerances in general

=> Entire chip prone to frequent, arbitrary transient faults
Microarchitecture and Fault Tolerance

• Conventional fault-tolerant techniques
  - Specialized techniques (e.g. ECC for memory, RESO for ALUs) do not cover arbitrary logic faults
  - Pervasive self-checking logic is intrusive to design
  - System-level fault tolerance (e.g. redundant processors) too costly for commodity computers

• A microarchitecture-based fault-tolerant approach
  - Microarchitecture performance trends can be easily leveraged for fault-tolerance goals
  - Broad coverage of transient faults
  - Low overhead: performance, area, and design changes
Talk Outline

- Technology and microarchitecture trends: implications to fault tolerance
  - Time redundancy spectrum
  - AR-SMT
  - Leveraging microarchitecture trends
    - Simultaneous Multithreading (SMT)
    - Speculation concepts
    - Hierarchical processors (e.g. Trace Processors)
- Performance results
- Summary and future work
Time Redundancy Spectrum

Program-level time redundancy

Instruction re-execution

Processor
Program P

Processor
Program P’

Processor

Program P

Instr i

dynamic scheduling
parallel execution units

FU
FU
FU
FU
Time Redundancy Spectrum

Processor

Program P

Program P'

AR-SMT
time redundancy
“A” => “Active stream”

“R” => “Redundant stream”

“SMT” => “Simultaneous MultiThreading”
AR-SMT Fault Tolerance

• Delay Buffer
  - Simple, fast, hardware-only state passing for comparing thread state
  - Ensures time redundancy: the A- and R-stream copies of an instruction execute at different times
  - Buffer length adjusted to cover transient fault lifetimes

• Transient fault detection and recovery
  - Fault detected when thread state does not match
  - Error latency related to length of Delay Buffer
  - Committed R-stream state is checkpoint for recovery
Leveraging Microarchitecture Trends

- Simultaneous Multithreading
- Control Flow and Data Flow Prediction
- Hierarchy and Replication
SMT

• Simultaneous multithreading
  [Tullsen, Eggers, Emer, Levy, Lo, Stamm: ISCA-23]
  - Multiple contexts space-share a wide-issue superscalar processor
  - Key ideas
    • **Performance**: better overall utilization of highly parallel uniprocessor
    • **Complexity**: leverage well-understood superscalar mechanisms to make multiple contexts transparent
  • SMT is (most likely) in next-generation product plans
Control and Data “Prediction”

(a) program segment  (b) base schedule  (c) with prediction

- Delay Buffer contains perfect “predictions” for R-stream!
- Existing prediction-validation hardware provides fault detection!
Trace Processors

- **Trace**: a long (16 to 32) dynamic sequence of instructions, and the fundamental unit of operation in Trace Processors

- Replicated PEs => inherent, coarse level of *hardware redundancy*
  - Permanent fault coverage: A-/R-stream traces go to different PEs
  - Simple re-configuration: remove a PE from the resource pool
AR-SMT on a Trace Processor

FETCH

A-stream
Trace Predictor
Trace Cache

R-stream
Trace "Prediction" from Delay Buffer

DISPATCH

Decode Rename Allocate PE

ISSUE
EXECUTE

RETIRe

PE
A-stream
PE
A-stream
PE
A-stream
PE
A-stream

Reg File
D-cache Disambig. Unit

Commit State
Reclaim Resources
time-shared
space-shared
Performance Results

![Normalized execution time bar chart](chart.png)

- **comp**: 1.11
- **gcc**: 1.22
- **go**: 1.22
- **jpeg**: 1.28
- **li**: 1.30

**Legend**:
- 4 PE
- 8 PE

**Configurations**:
- 4 PE
- 8 PE
Summary

• Technology-driven performance improvements
  - New fault environment: frequent, arbitrary transient faults

• Leverage microarchitecture performance trends for broad-coverage, low-overhead fault tolerance
  - SMT-based time redundancy
  - Control and data “prediction”
  - Hierarchical processors

• Introducing a second, redundant thread increases execution time by only 10% to 30%
Other Interesting Perspectives

  - (Quote) Fault-tolerant architectures have not kept pace with the rate of change in commercial systems.
  - Fault tolerance must make unconventional in-roads into commodity processors: leverage the commodity microarchitecture.

  - Implications of very high clock rate, dense designs
Future Work

• Performance and design studies
  - Isolate individual performance impact of SMT-ness and prediction aspects
  - Debug Buffer size vs. performance: SMT scheduling flexibility
  - Support more threads
  - Design and validate a full AR-SMT fault-tolerant system

• Derive fault models (collaboration?)

• Simulate fault coverage
  - fault injection
  - test different parts of processor: coverage varies?
  - vary duration of transient faults
  - permanent faults and re-configuration