A Case for Clumsy Packet Processors

Arindam Mallik and Gokhan Memik
Electrical and Computer Engineering Dept.
Northwestern University

Overview

- Faults
- Correctness is overrated
- What if the higher levels take care of it?
  - Processor can be even more aggressive/speculative
- Application-specific correctness
  - Networking applications
  - How do we measure?
- Tools for architects
  - Relation between overclocking and faults

Treat correctness as an objective, not a requirement

Outline

- Introduction
- Application description and error metrics
- Error models for overclocking a cache
- Processor configuration
- Measurement definitions
- Simulations

Motivation

- Performance, energy requirements
- Reliability / Probabilistic Circuits
- Circuit designers have to be conservative
  - Worst-case design

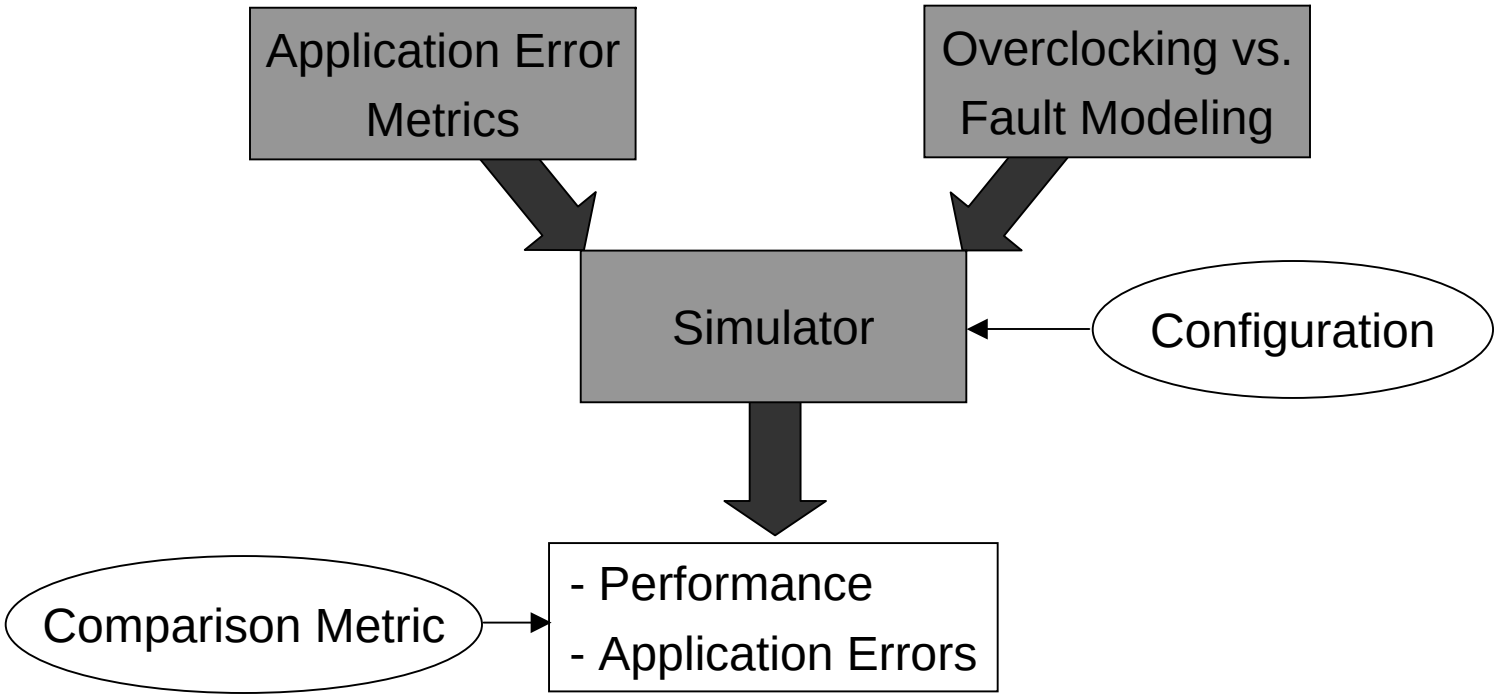
Introduction

- Inherent possibility of fault occurrence
  - Adverse environmental conditions
  - Aggressive scaling of supply voltage
  - Smaller manufacturing technologies
- Need for analysis
  - More Transistors ⇨ Higher fault probability
- Effect on system integrity
  - Transient faults
  - Permanent faults

Application Errors

- For desktop processor or server
  - Capture and eliminate all faults
- Networking – Communication
  - A certain level of error is acceptable
  - Nevertheless
    - The integrity of the system behavior must be maintained
    - System impact
      - Excessive “resubmission”
  - Program output

Overview of Approach



Error Classification

- Fault vs. Error
- Effect or duration
  - Volatile Error
    - Occurs mostly while processing a packet
    - Effects unit data element
      - Error in a single packet
  - Non-volatile Error
    - Occurs in the static data structures
    - Effects seen in many elements
      - Error in routing table

Error Metrics for Applications

- Categorization of NetBench Applications
  - Low or micro-level
    - Routines related to lowest layers of network stack
  - Routing-level
    - Applications similar to traditional IP routing (Layer 3-4 of the network stack)
  - Application-level
    - Traditional as well as emerging applications
- Common property of all applications
  - Control level tasks
  - Data level tasks

Error Measurement Procedure

- Mark data structures in NetBench apps
  - Important Data Structures
    - Routing Table Entries, TTL Value, …
  - Outputs of Key Function Units
    - Checksum Value, NAT Address
- Perform simulation
  - Introduce hardware faults
- Mark the change
  - Data values change
  - Application behavior changes
- Define the application error rate

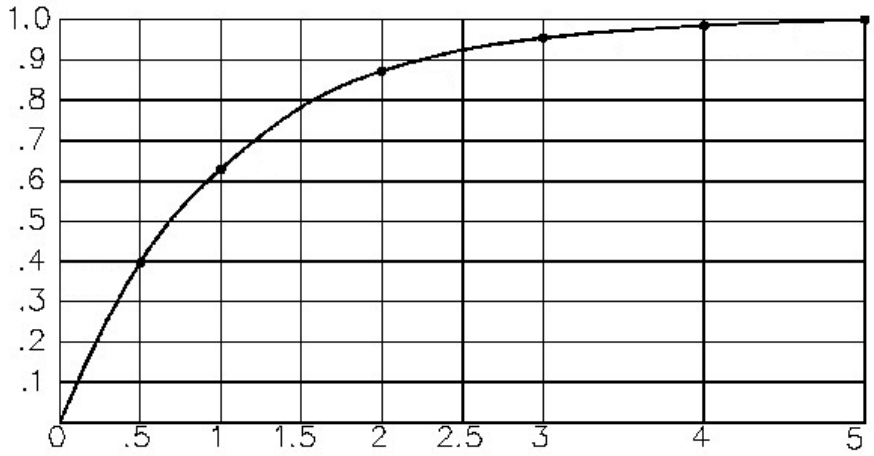
A Sample Application - Route

- Route – one of the most common networking applications
  - Implements IPv4 routing
  - Receives each packet – table lookup – processes it to decide the next network hop
- Error Keys
  - Routing Table Initialization (IMPORTANT !!)
  - Checksum value
  - TTL Value
  - Path traversed in Routing Table for each packet

Fault Models for Overclocking

- Overclock a component
  - Increased performance
  - Reduced energy
  - Increase in fault probability
- Goal
  - Find fault vs. overclocking aggressiveness
    - Particular circuit design
- Parameters
  - Voltage swing, noise

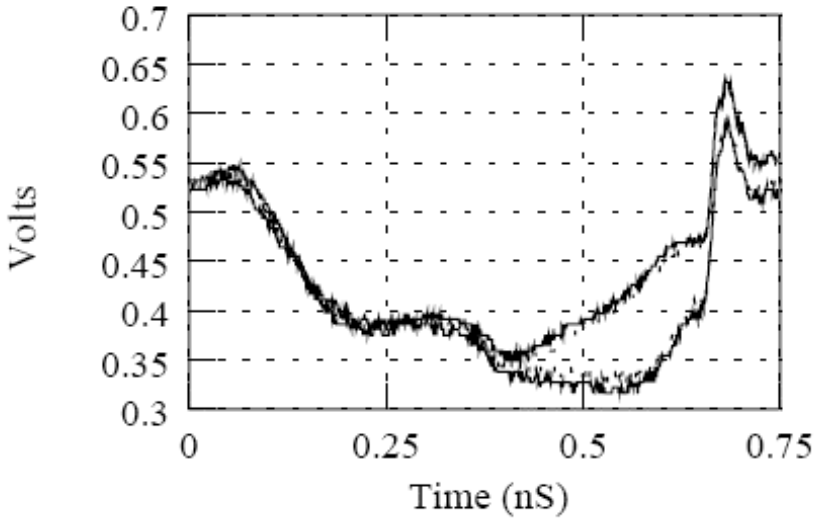
Opportunity for overclocking



Voltage Swing vs. Time

- Voltage swing
  - Rapid increase at first
  - Slow increase later

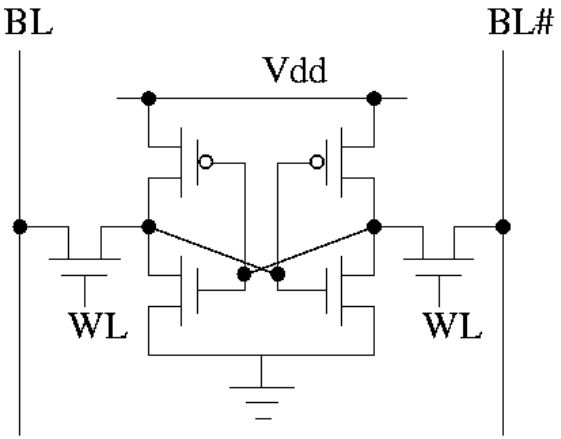
Not so fast, my friend!



- Noise (inductive and/or capacitive)
  - Signal deviation
- Overclocking
  - Reduced immunity

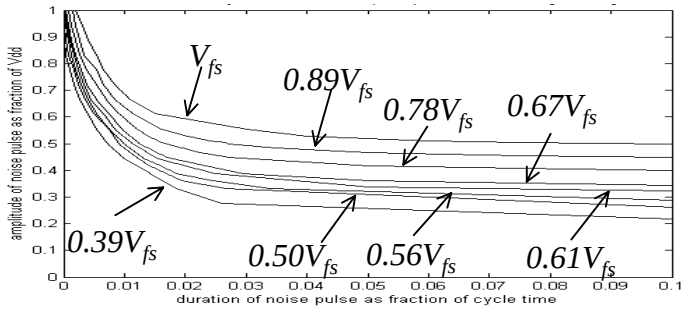
Approach

- Analyze each component separately

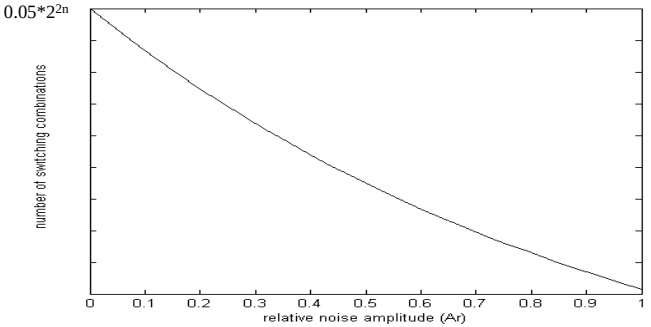


- 6-transistor SRAM cell
  - Input, clock, feedback loop

Finding fault probability



Noise immunity curves

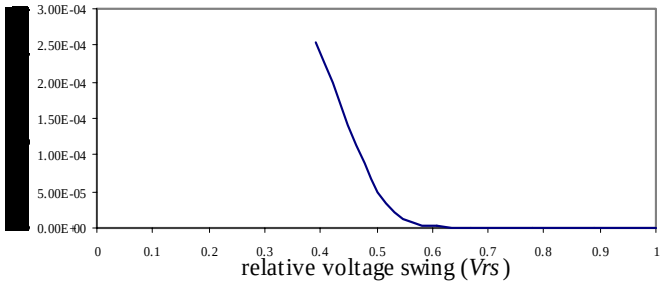


Noise amplitude for switching comb.

- Analyze the impact of noise on the feedback loop ⇨ Noise immunity curves
- Different noise amplitude probabilities
  - Check all switching combinations

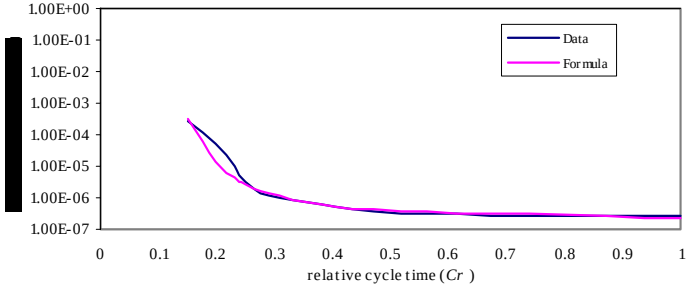
$$P(A_r) = 28.8 * e^{-28.8 A_r}$$

Estimation model



Fault probability versus voltage swing

- Fit distribution into immunity
- Combine it with voltage swing vs. time



Fault probability versus relative clock frequency

$$P_E = 2.59 * 10^{-7} * e^{\frac{1}{6*C_r^2}} = 2.59 * 10^{-7} * e^{\frac{F_r^2}{6}}$$

Outline

- ✓ Introduction
- ✓ Application description and error metrics
- ✓ Error models for overclocking a cache
- Processor configuration
- Measurement definitions
- Simulations

Processor Configuration

- Fault detection
  - No detection
  - Parity
    - One-strike, two-strikes, three-strikes
- Overclocking
  - Static
    - 75%, 50%, and 25% of the original
  - Dynamic
    - Processors adapts according to fault observed
    - Frequency is adjusted at the end of each epoch

Measurement definitions

- Comparison between ideal and erroneous execution
  - Traditional parameters – unfair competition ✗
  - Consider both performance and reliability ✓

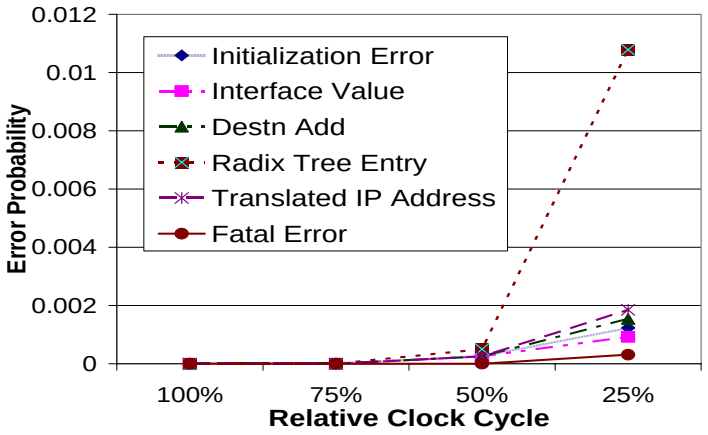
- Energy-Delay-Fallibility product
  - $\text{Energy}^k \times \text{delay}^m \times \text{fallibility}^n$
  - Fallibility = unit error occurrence probability
  - Can adjust the importance of faults by changing n
  - In present work, k = 1; m = 2; n = 2

Simulations

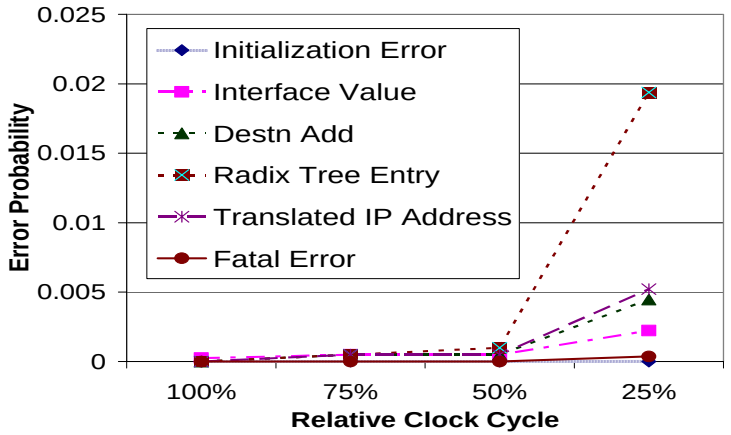
- SimpleScalar Simulator for StrongARM 110
  - Roughly an execution core of a Network Processor
  - Separate 4 KB direct mapped L1 data and instruction caches
  - 128 KB 4-way set-associative unified L2 cache

- Error Probability
  - At normal clock frequency
    - Error probability = $2.59*10^{-7}$ per bit
  - Increased error probability at higher clock rate according to the fault model

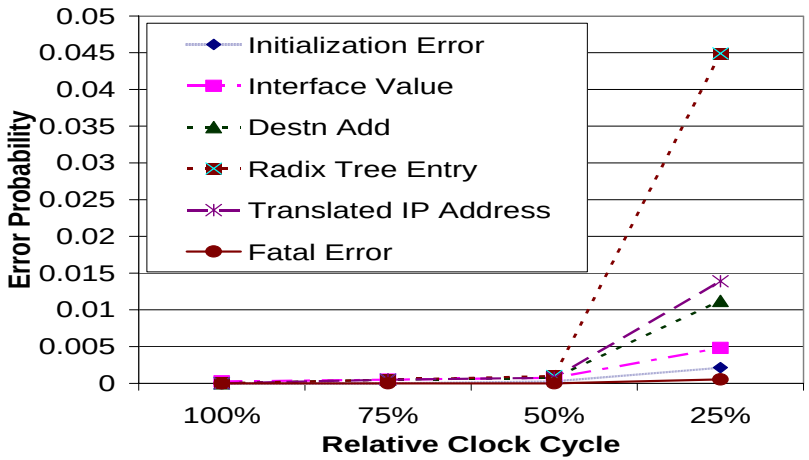
Application Error Behavior



Control plane



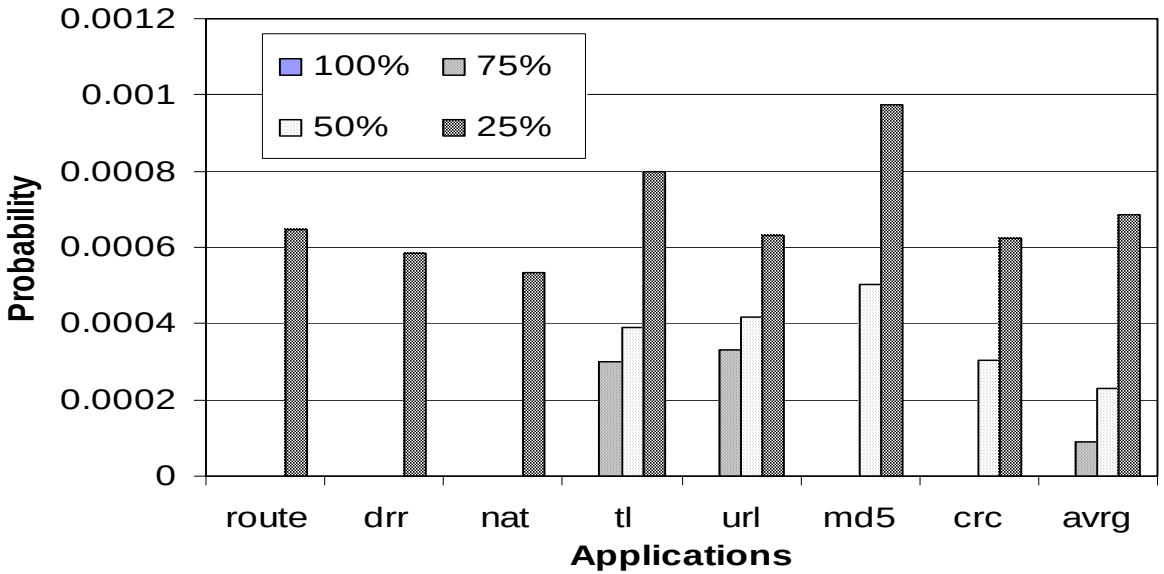
Data plane



Error introduced in both control and data plane

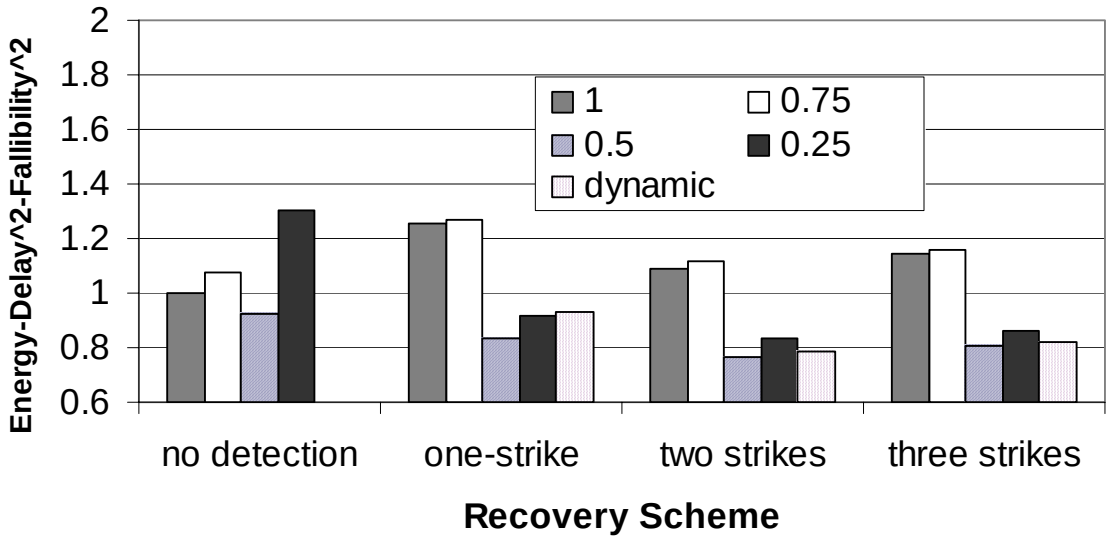
Fatal Error Probability

- Curse on the system
  - Destroys integrity – unacceptable
  - Increases with high clock frequency
- Observed on system with no error detection



Energy-Delay-Fallibility Values

- High Energy-Delay-Fallibility
  - Higher fallibility rate
  - Increased execution cycle
    - Extra instructions due to errors
    - Erroneous load ⇨ cache miss



Conclusions

- Release correctness constraint
- Application-Specific Processors
  - Utilizing released correctness
  - Application-Specific error metrics
- Overclocking
  - Fault modeling for overclocking a data cache
- Error weighting – metrics

Thanks!

- Anonymous reviewers

- Dr. Masud Chowdhury, Dr. Yehea Ismail, Sasha Jevtic, Dr. Bill Mangione-Smith, Dr. Seda O. Memik, Matthew Wildrick