SEU Characterization of Digital Circuits

Using Weighted Test Programs

Philip P. Shirvani and Edward J. McCluskey

Center for Reliable Computing
Stanford University

REE University Kickoff Meeting
Pasadena, CA, Sep. 6-7, 2000

Outline

● Motivation
● Background & Previous Work
● Proposed Method
● Weighted Test Programs
● Clock Frequency Dependency
● Linear Regression
● Conclusions
Motivation

- Major Objective
  - Single Event Upset (SEU) tolerance
- Specific Objective
  - SEU rate estimation
  - Functional unit cross sections
  - Achilles’ Heel

Why?

1- Tune FT strategies
   - Order in terms of importance (sensitivity)
   - Target most vulnerable unit first
2- Predict application error rate
Approaches to SEU Characterization

- **Hardware**
  - Good controllability and observability
  - Intrusive
    - Side effects, operating frequency, loads
  - Expensive
- **Scan**
  - Good for scanned elements
  - Still need error location & diagnostics
- **Software**
  - Limited controllability and observability
  - Operating condition same as real application

SEU Characterization Using Software

- More Complex Than Error Detection
  - Error location & diagnosis
- Problem: Controllability and Observability
  - Different faults
    - Same error
  - Error detection latency
    - Error propagation
  - Reading out internal state after error
SEU Characterization of Digital Circuits

SEU Characterization Using Software

⇒ Targeting One Functional Unit
  - Impossible or inaccurate

● Proposed Solution
  - Weighted test programs
  - System of linear equations
    - Linear regression

Processor SEU Characterization

● Functional Units
  - Fetch & decode, control logic
  - Register files (integer and floating-point)
  - ALU, multiplication & division
  - Floating point unit (FPU)
  - On-chip caches, TLBs, load/store buffers
  - Program invisible registers
    - e.g., pipeline latches
**Terminology**

- $Q_{\text{crit}}$
  - Min. charge required for changing state
- Cross Section ($\sigma$)
  - # errors / particle/cm$^2$
- Linear Energy Transfer (LET)
  - Particle energy
  - Si, GaAs
  - $\text{LET}_{\text{th}}$

**Goal**

- Functional Unit Cross Sections
  - [8][13]
Previous Work

- Program Mix
- Concentrated on Memory Elements
  - Good controllability and observability
  - e.g., register file
  - Cross section of whole chip
    - From bistable cross section [1][3][21]
    - Scan chain [3]
- Combinational vs. Sequential Circuits
  - Separate area on die [16]
  - Fault injection using laser [5]

Limits

- Hardware Approaches
  - Expensive, intrusive
- Direct Test Programs
  - Targeting one unit impossible with software
- Error Classifications
  - Inaccurate error location

Going Beyond the Limits
Outline

- Motivation
- Background & Previous Work
  - Proposed Method
  - Weighted Test Programs
- Clock Frequency Dependency
- Linear Regression
- Conclusions

Program Dependency

- Different Programs
  - Different cross sections [8][13][14][22][24][25]
- Duty Factor: \( f_i \)
  - % time unit is active or holds live value
- Predicting Cross Section of a Program [8]

\[ \sigma_T = \sum_i \sigma_i f_i \]

- Dynamic program behavior
  - e.g., software tool for estimating register utilization [1]
Proposed Method

- Example

\[ \sigma_T = \sigma_{\text{reg}} f_{\text{reg}} + \sigma_{\text{ALU}} f_{\text{ALU}} + \sigma_{\text{rest}} f_{\text{rest}} \]

- Measure \( \sigma_T \) for Different Programs
  - Different duty factors (\( f_i \)'s)
  - Equally likely inputs

Proposed Method (cont.)

- System of Linear Equations

\[
\begin{align*}
\sigma_{T,1} &= \sigma_{\text{reg}} f_{\text{reg,1}} + \sigma_{\text{ALU}} f_{\text{ALU,1}} + \sigma_{\text{rest}} f_{\text{rest,1}} \\
\sigma_{T,2} &= \sigma_{\text{reg}} f_{\text{reg,2}} + \sigma_{\text{ALU}} f_{\text{ALU,2}} + \sigma_{\text{rest}} f_{\text{rest,2}} \\
\sigma_{T,3} &= \sigma_{\text{reg}} f_{\text{reg,3}} + \sigma_{\text{ALU}} f_{\text{ALU,3}} + \sigma_{\text{rest}} f_{\text{rest,3}}
\end{align*}
\]

- Solve for Unknown Cross Sections (\( \sigma_i \)'s)
Varying Register Utilization

```
init r1, .., r10;
R = 0;
loop {
    add R, R, r1
    add R, R, r2
    ...
    add R, R, r10
}
check R;
```

- 32 GPRs
- \( f_{reg} = \frac{10}{32} \text{ or } \frac{20}{32} \)

```
init r1, .., r20;
R = 0;
loop {
    add R, R, r1
    add R, R, r2
    ...
    add R, R, r10
    add R, R, r11
    add R, R, r12
    ...
    add R, R, r20
}
check R;
```

Issues

- How Well Can We Control \( f_i \)'s?
  - Carefully written test programs
  - Nontrivial
  - Limited range: \( 0 \leq f_i \leq 1 \)
  - Matching activities
  - Instruction latencies and stall cycles
  - Not always possible
- Breaking the Linear Dependency Between \( f_i \)'s
- Low Error Rates (small cross sections)
  - e.g., \( \sigma_{ALU} \ll \sigma_{reg}, \sigma_{rest} \)
Matching Activities

```
init r1, .., r10;
R = 0;
loop {
    add R, R, r1
    add R, R, r2
    ...
    add R, R, r10
}
check R; // checksum
```

```
init r1, .., r20;
R = 0;
loop {
    add R, R, r1
    add R, R, r2
    ...
    add R, R, r10
    add R, R, r11
    add R, R, r12
    ...
    add R, R, r20
}
check R; // checksum
```

Varying ALU Utilization

```
outer loop {
    loop1 n times {
        add R, R, r1
        add R, R, r2
        ...
        add R, R, r20
    }
    loop2 m times {
        nop
        nop
        ...
        nop
    }
}
check R; // checksum
```

\[
\lambda = \sigma_{\text{reg}} f_{\text{reg}} + \sigma_{\text{ALU}} f_{\text{ALU}} + \sigma_{\text{rest}} f_{\text{rest}}
\]

\[
f_{\text{ALU}} = \frac{n}{n + m}
\]

- Different m and n’s
- Continuous instruction fetch and execution
Matching Activities

\[ \lambda = \sigma \int_{\text{reg}} + \sigma \int_{\text{ALU}} + \sigma \int_{\text{rest}} \]

- Same \( \int_{\text{rest}}, \int_{\text{reg}} \)
  - Instructions
  - Register file activity

\[
\begin{align*}
\text{outer loop} \{ \\
\text{loop1 n times} \{ \\
& \text{add } R,R,r1 \\
& \text{add } R,R,r2 \\
& \ldots \\
& \text{add } R,R,r20 \\
\} \\
\text{loop2 m times} \{ \\
& \text{add } r0,R,r1 \\
& \text{add } r0,R,r2 \\
& \ldots \\
& \text{add } r0,R,r20 \\
\} \\
\} \\
\text{check } R; // \text{ checksum}
\end{align*}
\]

Instruction Latencies and Stall Cycles

\[ \lambda = \sigma \int_{\text{reg}} + \sigma \int_{\text{Mult}} + \sigma \int_{\text{rest}} \]

- In MIPS 3000
  - Mult: 12 cycles
    - \( \int_{\text{rest}} \downarrow \)
  - 10 stall cycles

- Stalls
  - \( \int_{\text{rest}} \downarrow \)
  - (relative to ALU test)
Cache Effects

- On-Chip Cache
  - Dominates chip cross section [11][17][25]
- Test with and without Cache
  - Side effects
    - Stall cycles = different CPU activity
- Cache Misses
  - Affect duty factors
- Adjust Program and Data Size

Outline

- Motivation
- Background & Previous Work
- Proposed Method
- Weighted Test Programs
  - Clock Frequency Dependency
- Linear Regression
- Conclusions
Datapath Cross Section

- ALU Errors
  - Inseparable from errors in pipeline latches
  \[ \sigma_{ALU} \rightarrow \sigma_{ALU} + \sigma_{ALU \_latches} \]
  \[ \sigma_T = \sigma_{ALU} f_{ALU} + \sigma_{ALU \_latches} f_{ALU \_latches} \]
  \[ + \sigma_{rest} f_{rest} + \sigma_{reg} f_{reg} \]
- \( f_{ALU} \) and \( f_{ALU \_latches} \) Linearly Dependent
- Difference
  - ALU: combinational circuit
  - Latches: storage element (sequential circuit)

SEUs in Storage Elements

- Flip-flops, Latches, Embedded RAMs
- Soft Errors
- Clock Frequency Dependency
  - Constant [5][23]
    - Transient << clock cycle [5]
  - Nonlinear [19]
  - Linear [10][19]
    - shift register; flip-flops [10]
SEUs in Combinational Circuits

- Transients
- ALU
  - Both low and high cross section [14][16]
- Clock Frequency Dependency
  - Linear [4][5][16][19][20]

Separating Datapath Cross Sections

- Using Clock Frequency Dependency
  - \( \sigma_{ALU} \) linear
  - \( \sigma_{ALU \_latches} \) constant

\[
\sigma_{T,1} = \sigma_{ALU @ w1}f_{ALU} + \sigma_{ALU \_latches @ w1}f_{ALU \_latches} \\
+ \sigma_{rest}f_{rest} + \sigma_{reg}f_{reg}
\]

\[
\sigma_{T,2} = \sigma_{ALU @ w2}f_{ALU} + \sigma_{ALU \_latches @ w2}f_{ALU \_latches} \\
+ \sigma_{rest}f_{rest} + \sigma_{reg}f_{reg}
\]

- \( \sigma_{ALU} \) and \( \sigma_{ALU \_latches} \) Linearly Independent
Key Points

- Controlling Unit Activity
  - Accurately and independently
- Detecting All Errors in Active Units
  - Error detection coverage
- Filtering or Counting External Events
- Classifying Errors
  - Not needed when there is ambiguity
    - e.g., ALU vs. decoder

Outline

- Motivation
- Background & Previous Work
- Proposed Method
- Weighted Test Programs
- Clock Frequency Dependency
  - Linear Regression
- Conclusions
Statistical Estimation [12]

- Multiple Linear Regression Model
  - Assuming independent $x_i$’s
    \[ y = a_0 + a_1x_1 + a_2x_2 + \ldots + a_kx_k + \varepsilon \]
  - $n$ observations
    \[ Y = XA + \varepsilon \]
- Estimation of $p$ Parameters ($a_i$’s)
  - Method of least squares
    \[ L = \sum_{i=1}^{n} \varepsilon_i^2 \quad \hat{A} = (X^T X)^{-1} X^T Y \]
- For 95% confidence: \( (n - p) \approx 5 \)

Experiments

- ARGOS
  - SEUs mostly in main memory
  - Next biggest contributor?
    - Difficult to find due to low error rate
- REE
  - SEU characterization of processors
  - Ground tests
    - Higher error rate
Summary

- SEU Characterization of Digital Circuits
  - Register cross section not sufficient
  - Combinational circuits gaining importance
- Duty Factors Important
  - In estimation and prediction
- Weighted Test Programs (or Inputs)
  - Using
    - Duty factors
    - Clock frequency dependency
- Linear Regression

Conclusions

- Direct Software Methods
  - Good for some units
- Proposed Method
  - Alleviating some limits of previous methods
  - Less dependent on error classification
  - Microarchitecture information
    - Tests not always directly portable
  - In conjunction with previous methods
    - e.g., scan approach
  - Using other cross section dependencies
Future Work

- Verify Estimation Accuracy
  - Fault injection simulations
    - Verilog model
- REE Adaptation
  - Architecture and experiment specifics

References (1)

References (2)


References (3)

References (4)


References (5)