SEU Characterization of Digital Circuits

SEU Characterization of Digital Circuits
Using Linear Regression
by
Philip P. Shirvani
RATS
July 10, 2000

Outline
● Motivation
● Background & Previous Work
● Proposed Method
● Weighted Test Programs
● Linear Regression
● Conclusions

Motivation
● Fault Tolerance Against SEUs
  ☛ Single Event Upsets
● SEU Rate Estimation
  ☛ In different Functional Units (FUs)
  ☛ Finding the Achilles’ Heel
● Why?
  1- Tune FT strategies
    ● Order in terms of importance (sensitivity)
    ● Target most vulnerable first
  2- Predict application error rate

Approaches
● 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 Than Just Error Detection
  ● Error location & diagnosis
● Problem: Controllability and Observability
  ● Different faults
    ● Same error
  ● Error detection latency
  ● Error propagation
  ● Reading out internal state after error
  ⇒ Targeting one FU
    ● Mostly impossible or inaccurate

SEU Characterization Using Software
● Proposed Solution
  ● Weighted test programs
  ● System of linear equations
  ● Linear regression
**Terminology**
- **Q_{crit}**: Min. charge required for changing state
- **Cross Section (\(\sigma\))**: \#errors / particle/cm\(^2\)
- **Linear Energy Transfer (LET)**: Particle energy
  - Si, GaAs
  - \(LET_{th}\) (from ref. [8])

**Previous Work**
- Mix of Test Programs
- Mostly Concentrated on Memory Elements
  - Good controllability and observability
  - e.g., register file
- Infer Cross Section of Whole Chip
  - From cross section of latches [1][3][21]
- Scan Test [3]
- Combinational vs. Sequential Circuits
  - Separate area on die [16]
  - Fault injection using laser [5]

**Example**
  - Separate irradiation
    - Processor core
    - L2 tag SRAM
    - L2 cache
    - ALU: numerical factorization
    - FPU: calculating \(\pi\)
    - Register and cache
      - Write pattern and read back in loop
      - Under DOS and Windows

**Test Programs**
- FU Targeting Tests
  - ALU, I/O, FPU intensive [11][15]
  - Some parts always active
- Percentage of Instruction Set [22]
  - Active chip area
  - Dynamic activity not considered

**Error Classification**
- Error Behavior
- Bit Error, Word Error and Complex Error [8]
- Catastrophic Error
  - e.g., unexpected jump
  - Attributed to PC and instruction decoder
- Data, Sequencing, Address, Other Errors [25]

**Example**
- Testing Z-80 and NSC-80 [7]
  - Register file vs. internal latches in Z-80
    - No combinational
    - Data, address and control pins
    - Single and multiple errors
    - Single-bit error in data or address pins
      - Bit-flip in general purpose registers
    - Single-bit error in control pins
      - Bit-flip in internal latches
**SEU Characterization of Digital Circuits**

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

**Going Beyond the Limits**

### Program Dependency
- Different Programs
  - Different cross sections [8][13][14][22][24][25]
- Live and Relevant Registers [14]
- FU Duty Factor: $f_i$
  - % time unit is active or holds live value
- Predicting Cross Section of a Program [8]
  \[
  \sigma_T = \sum \sigma_i f_i
  \]
- Dynamic program behavior
  - e.g., software tool for estimating register utilization [1]

### Proposed Method
- Example
  \[
  \lambda = \sigma_{reg} f_{reg} + \sigma_{alu} f_{alu} + \sigma_{reu} f_{reu}
  \]
- Measure $\lambda$ for Different Duty Factors ($f_i$’s)
- System of Linear Equations
  \[
  \lambda_1 = a_1 x + b_1 y + c_1 z \\
  \lambda_2 = a_2 x + b_2 y + c_2 z \\
  \lambda_3 = a_3 x + b_3 y + c_3 z
  \]
- Solve for Unknowns Cross Sections ($\sigma_i$’s)

### Varying Register Utilization
- 32 GPRs
- $f_{reg}$ = 10/32 or 20/32

### Varying ALU Utilization
- Different m and n’s
- Continuous instruction fetch and execution

**Example**

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

```c
init r1, ..., r20; 
sum = 0; 
loop { 
  add sum,sum,r1
  add sum,sum,r2
  ...
  add sum,sum,r10 
  add sum,sum,r11
  ...
  add sum,sum,r20
  } 
check sum;
```
SEU Characterization of Digital Circuits

Matching Activities

```
outer loop {
    loop1 n times {
        add sum, sum, r1
        add sum, sum, r2
        ...
        add sum, sum, r20
    }
    loop2 m times {
        add r0, sum, r1
        add r0, sum, r2
        ...
        add r0, sum, r20
    }
}
check sum;
```

Instruction Latencies and Stall Cycles

```
outer loop {
    loop1 n times {
        mult sum, sum, r1
        mult sum, sum, r2
        ...
        mult sum, sum, r20
    }
    loop2 m times {
        mult r0, sum, r1
        mult r0, sum, r2
        ...
        mult r0, sum, r20
    }
}
check sum;
```

Cache Effects

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

Datapath Cross Section

- ALU Errors
  - Inseparable from errors in pipeline latches
    \[ \sigma_{ALU} \rightarrow \sigma_{ALU} + \sigma_{latches} \]
  - \( \lambda = \sigma_{reg} f_{reg} + \sigma_{ALU} f_{ALU} + \sigma_{latches} f_{latches} + \sigma_{ret} f_{ret} \)
  - \( f_{ALU} \) and \( f_{latches} \) linearly dependent
- Using Frequency Dependency
  - e.g.:
    - \( \sigma_{ALU} \) linear [4][5][16][19][20]
    - \( \sigma_{latches} \) constant [5][23]

Statistical Estimation [12]

- Multiple Linear Regression Model
  - Assuming independent \( x_i \)'s
    \[
y = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_k x_k + \epsilon
    \]
  - \( n \) observations
    \[
    Y = \mathbf{X} \alpha + \epsilon
    \]
  - Estimation of \( p \) Parameters (\( \alpha_i \)'s)
    - Method of least squares
      \[
      L = \sum \epsilon_i^2
      \]
      \[
      \hat{\alpha} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}
      \]
  - Statistical Prediction [12]
    - 100(1 - \( \alpha \)) Percent Confidence Interval
      \[
      \text{Prob}[L_i \leq a_i \leq U_i] = 1 - \alpha
      \]
    - \( t \) Distribution with \( (n - p) \) Degrees of Freedom
      \[
      \hat{\alpha}_i - t_{a/2,n-p} \sqrt{\hat{\sigma}^2 C_i} \leq a_i \leq \hat{\alpha}_i + t_{a/2,n-p} \sqrt{\hat{\sigma}^2 C_i}
      \]
    - Predict \( y \) for a set of \( x_i \)'s
### SEU Characterization of Digital Circuits

#### 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

#### References (1)

#### References (2)

#### References (3)
References (4)


References (5)