TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik - Institut für Technische Informatik - Professur für VLSI-Entwurfssysteme, Diagnostik und Architektur

Vortrag zum Beleg

Studie zur Erhöhung der Zuverlässigkeit der SHAP-Mikroarchitektur.

Alexander Simon, s1514488@inf.tu-dresden.de

Dresden, 27.05.2009

Gliederung

1. Aufgabenstellung
2. Einleitung
3. Unterscheidung zwischen Fehlerarten
4. Methoden der Fehlererkennung und -korrektur
5. Ausblick

Aufgabenstellung

1. Literaturstudium zu Fehlererkennung und Reparatur von permanenten Hardwarefehlern.
2. Literaturstudium zu Ursachen, Erkennung und Korrektur von transienten Hardwarefehlern.
3. Identifikation von geeigneten Konzepten zur Fehlererkennung und -korrektur und deren Anwendung auf die verschiedenen Klassen von Hardware-Modulen.
4. Auswahl geeigneter Ansätze für die SHAP-Mikroarchitektur. Zusammenstellung verschiedener Szenarien zur Fehlererkennung und -korrektur.
5. Zusammenstellung und Dokumentation der Ergebnisse.

Einleitung

Zuverlässigkeit bedeutet:

- verfügbar sein → Bereitschaft zur Ausführung
- vertraulich sein → korrekte Ausführung zu jedem Zeitpunkt
- wartbar sein → Fähigkeit Reparaturen/Modifikationen durchführen zu können
- echt sein → das System darf sich nicht unwillkürlich verändern

Zuverlässigkeitsgewinn

- Fehlervorsorge → Qualitätskontrollen
- **Fehlertoleranz → Fehlererkennung und -korrektur (diese Studie)**
- Fehlerbeseitigung → Verifikation, Diagnose und Korrektur in der Entwicklung
- Fehlerprognose → Bewertung des Systems auf dessen Zuverlässigkeit

Fehlerarten [7]

- permanent
  - wenn die zugesicherte Eigenschaft oder die geforderte Funktion von der Betrachtungseinheit dauerhaft nicht erfüllt wird
- transient
  - vorübergehendes Nichterfüllen einer Funktion

Grundsätzliche Fehlerkorrekturmethoden

- Forward Error Recovery (FER)
  - nutzt mehrfach enthaltenen Module zur Fehlerkorrektur
- Backward Error Recovery (BER)
  - erstellt Kontrollpunkte, die zur Fehlererkennung/-korrektur eingesetzt werden

Unterscheidung zwischen Fehlerarten

Heuristische Methode

- Fehlerzähler in Modulen
- ab bestimmtem Schwellwert wird ein permanenter Fehler angenommen
- einfache Umsetzung, aber Schwellwert muss gut dimensioniert sein

Bayessche Methode

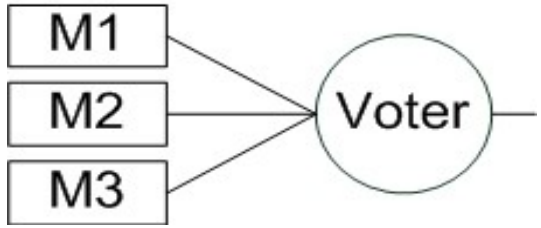
- basiert auf der Bayesschen Gleichung für bedingte Wahrscheinlichkeiten
- durch Beobachtung vorheriger Zustände wird die Wahrscheinlichkeit für das Auftreten eines permanenten Fehlers berechnet
- exaktere Unterscheidung des Fehlertyps möglich

Methoden zur Fehlererkennung und -korrektur

- redundante Systeme
- dynamische Verifikation
- Fehlerbehebung mittels Rekonfiguration
- Trace-Based Fehlerdiagnose
- Error Correcting Codes
- ActiveRedundant-SMT
- BIST/BISR

Redundante Systeme

- statische Redundanz
  - → Fehlermaskierung durch parallele Module
  - → Mehrheitsvoter entscheidet über korrektes Ergebnis
  - → weitere Zuverlässigkeitssteigerung durch den Einsatz mehrerer Voter bei Submodulen



- dynamische Redundanz
  - → ein Modul arbeitet und Reserveeinheiten stehen bereit
  - → bei entdecktem Fehler wird Reservemodul aktiv

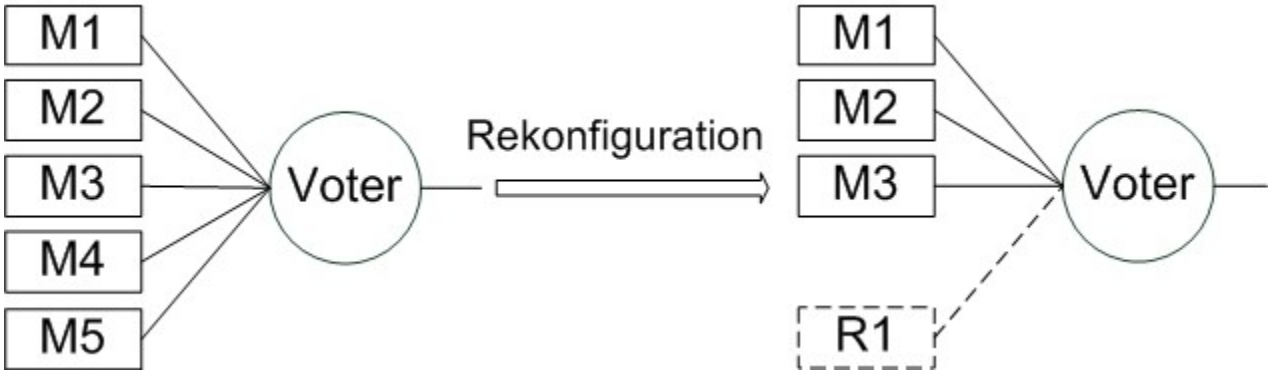
Redundante Systeme

- hybride Redundanz
  - → Kombination aus statischer und dynamischer Redundanz
  - → bei Ausfall von fehlerhaften Modulen stehen Reserven zur Verfügung

- self-purging Redundanz
  - → alle Module sind über je einen Schalter an Voter angeschlossen
  - → bei einem Fehler wird so das Modul einfach abgeklemmt
  - → Einbinden der Reservemodule entfällt

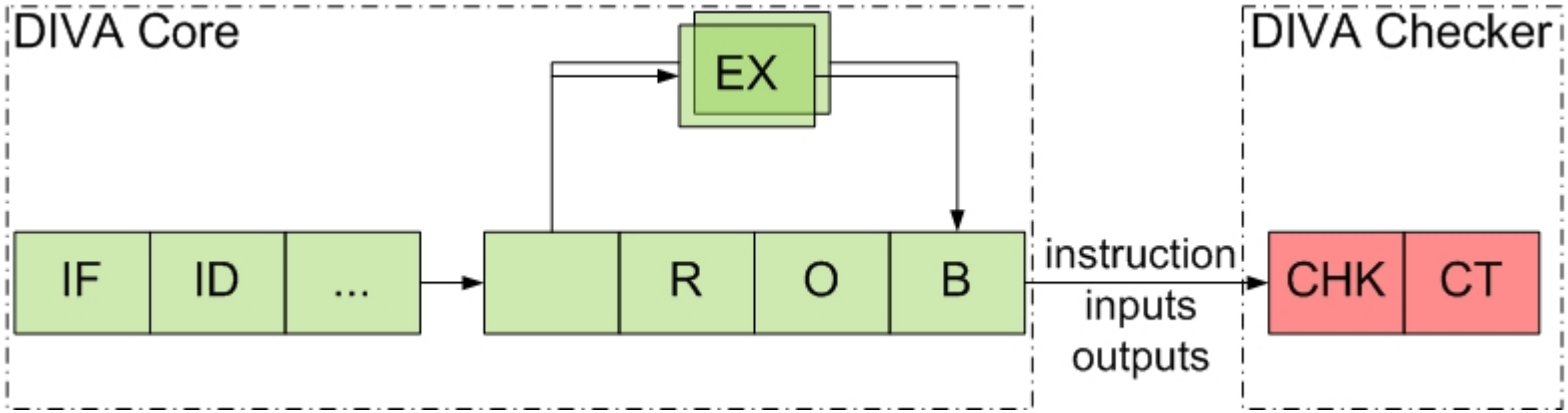
Redundante Systeme

- Rekonfigurationsmöglichkeiten
  - aus statischer Redundanz mit 5 Modulen kann das System je nach Anzahl der Fehler zu einem TMR umkonfiguriert werden
  - ein Doppelfehler und ein Einzelfehler oder 3 Einzelfehler können korrigiert werden



Dynamische Verifikation

- dient zur Detektion und Korrektur permanenter und transienter Fehler, als auch von Designfehlern
- Verifikation aller Ergebnisse mit einer Checkereinheit
- bei detektiertem Fehler wird Pipeline geleert und neu gestartet



Dynamische Verifikation

- Annahmen: - Speicher- und Registerinhalte werden mit ECCs geprüft
  - Reihenfolge der Befehle ist korrekt
  - die Ausführungen des DIVA Checkers sind fehlerfrei
  - Austausch zwischen Core und Checker ist korrekt

- Vorteile: - online Verifikation mit geringen Geschwindigkeitseinbußen
  - jede Berechnung wird am Ende korrekt abgespeichert
  - einfache Implementierung des Checkers
  - geringer Hardwareaufwand wenn Hazards gebilligt werden

- Nachteile: - Strukturhazards wenn Checker auf Speicher zugreifen will

Fehlerbehebung mittels Rekonfiguration

- Erweiterung des DIVA Ansatzes
- enthält DIVA Checker zur Fehlerdetektion
- Prozessor wird in mehrere Einheiten unterteilt
- jedes Modul enthält Fehlerzähler zur Detektion eines permanenten Fehlers
- bei Überschreitung eines Schwellwertes wird die entsprechende Einheit als „ständig benutzt“ markiert
- Speichereinträge werden maskiert
- Funktionseinheiten werden nicht mehr verwendet

Fehlerbehebung mittels Rekonfiguration

- Vorteile: - Fehlerkorrektur möglich
  - dadurch Steigerung der Gesamtperformance
  - geringer zusätzlicher Hardwareaufwand

- Nachteile: - Fehlerdetektion in Checker nicht geklärt

Trace-Based Fehlerdiagnose

- nur in Multicore-Systemen einsetzbar
- bei entdecktem Fehler wird das System auf sicheren Zustand zurückgesetzt und dieser wird auf anderen Kern geladen
- durch Vergleich beider Durchläufe kann das defekte Modul identifiziert werden

Error Correcting Codes

- finden Verwendung beim Austausch von Daten
- benutzen Coderedundanz zur Fehlererkennung und -korrektur
- Beispiele: CRC, Paritätsprüfung, Hamming-Codes, Diamond-Codes

ActiveRedundant-SMT

- nutzt zeitliche Redundanz, um transiente Fehler tolerieren zu können
- Thread wird nach bestimmter Zeit noch einmal ausgeführt
- bei unterschiedlichen Ergebnissen wird Befehl erneut ausgeführt

BIST/BISR

- Module werden mit einer integrierten Testschaltung getestet
- System kann auf entsprechende Reserveeinheiten zurückgreifen

Ausblick

Offene Aufgaben

- detaillierte Bewertung weiterer Mechanismen
- genaue Einordnung studierter Methoden auf verschiedene Klassen von Hardware-Modulen
- Studium der SHAP-Mikroarchitektur
- Zusammenstellung verschiedener Szenarien zur Verbesserung der Zuverlässigkeit der SHAP-Mikroarchitektur
- Zusammenfassung und Dokumentation der Ergebnisse

Literaturverzeichnis(1)

[1] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell and Vytautas Magnus U. Fundamental concepts of dependability, 2000.

[2] Fred A. Bower, Daniel J. Sorin, and Sule Ozev. A mechanism for online diagnosis of hard faults in microprocessors. In MICRO 38: Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture, pages 197–208, Washington, DC, USA, 2005. IEEE Computer Society.

[3] Fred A. Bower, Daniel J. Sorin, and Sule Ozev. Online diagnosis of hard faults in microprocessors. ACM Trans. Archit. Code Optim., 4(2):8, 2007.

[4] Man-Lap Li, Pradeep Ramachandran, Swarup Kumar Sahoo, Sarita V. Adve, Vikram S. Adve, and Yuanyuan Zhou. Trace-based microarchitecture-level diagnosis of permanent hardware faults. In DSN, pages 22–31. IEEE Computer Society, 2008.

Literaturverzeichnis(2)

[5] M. Pizza, Lorenzo Strigini, Andrea Bondavalli, and Felicita Di Giandomenico. Optimal discrimination between transient and permanent faults. In HASE ’98: The 3rd IEEE International Symposium on High-Assurance Systems Engineering, pages 214–223, Washington, DC, USA, 1998. IEEE Computer Society.

[6] Stephen Y. H. Su and Richard J. Spillman. An overview of fault-tolerant digital system architecture. In AFIPS ’77: Proceedings of the June 13-16, 1977, national computer conference, pages 19–26, New York, NY, USA, 1977. ACM.

[7] Bundesamt für Sicherheit in der Informationstechnik. Definitionen und Metriken für die Hochverfügbarkeit. 2009.