Seminar Formal Methods for Fun and Profit

Verifikation der Fliesskomma-Arithmetik bei Intel

Seminarleiter Jun.Prof. Dr. Bernhard Beckert

Dennis Willkomm 16.08.2005

U N I V E R S I T Ä T
KOBLENZ · LANDAU

Gliederung

- Der Pentium-Bug
- Formale Methoden bei Intel
- Die Verifikation einer Fliesskommaeinheit
- Fazit und Ausblick

Der Pentium-Bug

- Durch zunehmende Komplexität schleichen sich schneller „Bugs“ ein
- Reaktionen auf Bugs
  - Workarounds
  - Änderung der Dokumentation
  - Bugfix
- Prominente Bugs
  - Ariane
  - Pentium-Bug

Der Pentium-Bug

- Benannt nach dem Assemblerbefehl FDIV
- 1994 entdeckt
- Ergebnisse nach dem Runden wichen ab
- Intel musste CPUs austauschen
- Schaden von ca. 500 Millionen $
- Reaktion:
  - Veröffentlichung aller entdeckten Fehler
  - Formale Methoden

Formale Methoden bei Intel

- Steigende Komplexität macht Prozessoren fehleranfälliger
- Auffinden der Bugs durch
  - Tests
  - Formale Verifikation
- Beweise in Mathematik recht formal
- Tests in Computertechnik geläufiger
- Trotz umfangreicher Tests Fehler in Pentium Prozessor

Formale Methoden in der Industrie

- Model Checking
  - Algorithmische Methode
  - Verifiziert finite-state System formal
  - System häufig als gerichteter Graph dargestellt
- Theorem Proving
  - Prüft mathematische Theoreme
  - Herleitung eines Satzes aufgrund von Regeln

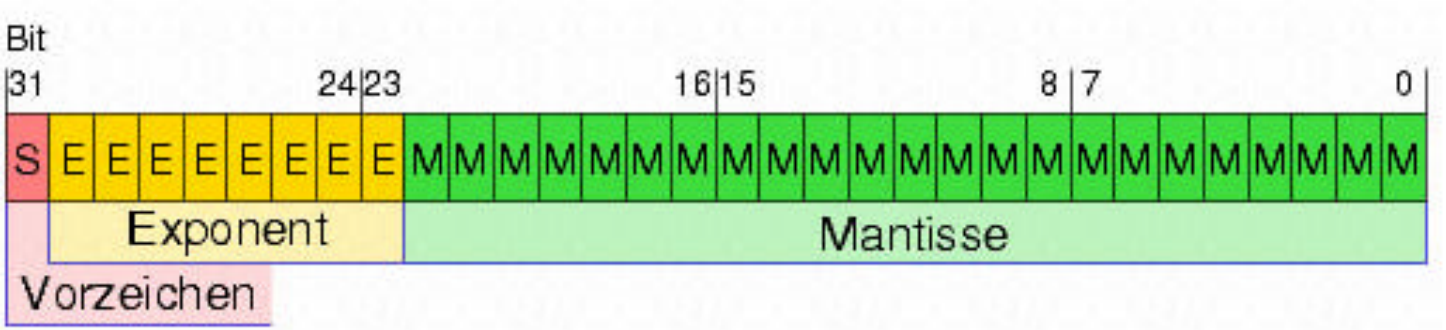
Die Arbeit bei Intel

- Nach Pentium-Bug starker Einsatz von Formalen Methoden
- Viele Bugs werden in Pre-Silicon Phase gefunden
- Formale Verifikation Standard für Fliesskomma-Einheiten
- Theorem Proving per Hand ist sehr mühsam und langwierig
- Einsatz von Werkzeugen wie HOL-Light

Repräsentation von Fliesskommazahlen

- Bestandteile von Fliesskommazahlen
  - Vorzeichen
  - Mantisse
  - Exponent

$$R = \frac{s \cdot m \cdot 2^{e}}{2^{P} \cdot 2^{bias}}$$



Repräsentation von Fliesskommazahlen

- Genauigkeit
  - 32bit single
  - 64bit double

- Rundungsmodi
  - round(toZero, R, V)
  - round(toNegInf, R, V)
  - round(toPosInf, R, V)
  - round(toNearest, R, V)

Die Verifikation einer Fliesskomma-Einheit

- Vorgehensweise
  - Aufstellen einer formalen Spezifikation
  - Zeigen, dass die Implementation die Spezifikation erfüllt
- Problem: Spezifikation muss in reine Integeroperationen umgewandelt werden, da das verwendete Framework nur mit Integer umgehen kann
- Beispiele:
  - FMUL
  - FDIV

Die Verifikation einer Fliesskomma-Einheit

- Spezifikation der Fliesskomma-Multiplikation

$$V_{FMUL} = \frac{s_1 \cdot m_1 \cdot 2^{e_1}}{2^P \cdot 2^{bias}} \cdot \frac{s_2 \cdot m_2 \cdot 2^{e_2}}{2^P \cdot 2^{bias}}$$

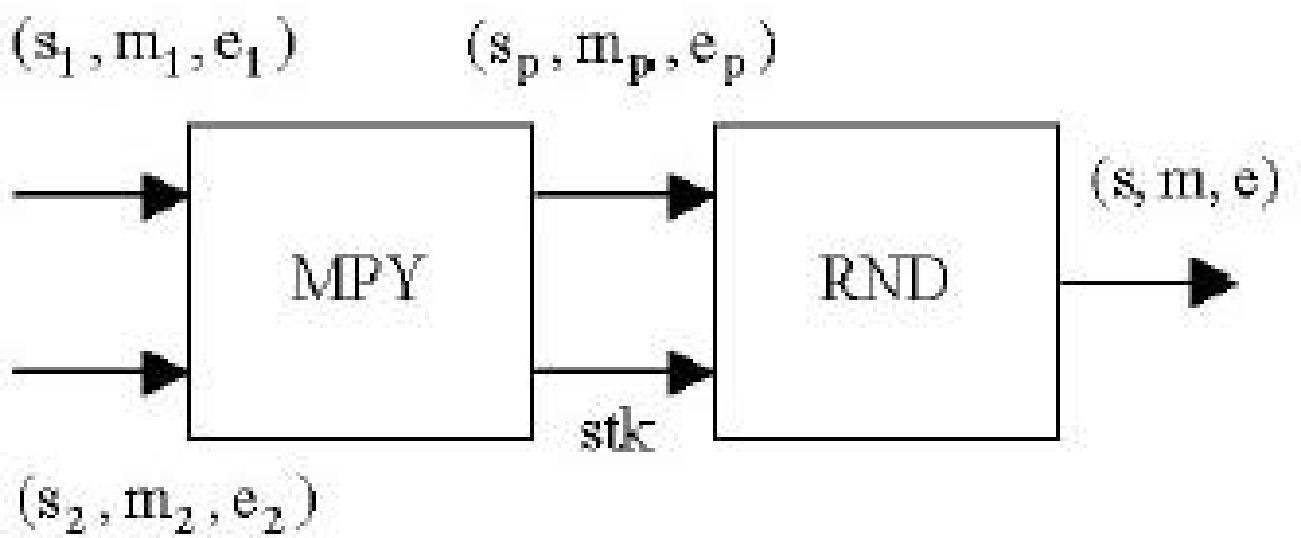
- Umwandlung in Integeroperationen

$$round(toNegInf, R_{FMUL}, V_{FMUL}) = ((s \cdot m \cdot 2^e) \cdot 2^{P|bias} \leq V) \wedge (V' < (s \cdot m \cdot 2^e + B^+) \cdot 2^{P+bias})$$

$$V' = (s_1 \cdot m_1 \cdot 2^{e_1}) \cdot (s_2 \cdot m_2 \cdot 2^{e_2})$$

Die Verifikation einer Fliesskomma-Einheit

- Implementation der Fliesskomma-Multiplikation



Die Verifikation einer Fliesskomma-Einheit

- Verifikation von FMUL
  - Zerlegung in Multiplikationsmodul und Rundungsmodul
  - Verifikation der Lower-Level Spezifikation mit Model Checking
  - Theorem Proving kombiniert Ergebnisse
- Zu zeigen:
  - $m_P \leq m_1 \cdot m_2$
  - $m_P + 1 > m_1 \cdot m_2$
  - $e_P = e_1 + e_2 + P + bias$
  - $s_P = s_1 \cdot s_2$

Die Verifikation einer Fliesskomma-Einheit

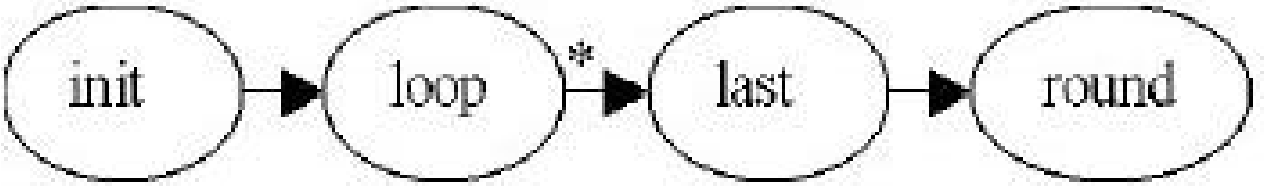
- Spezifikation von FDIV

$$V_{FDIV} = \frac{\frac{s_2 \cdot m_2 \cdot 2^{e_2}}{2^P \cdot 2^{bias}}}{\frac{s_1 \cdot m_1 \cdot 2^{e_1}}{2^P \cdot 2^{bias}}}$$

- Umformung für reine Integeroperationen

Die Verifikation einer Fliesskomma-Einheit

- Implementation von FDIV



- Implementiert durch iterative Algorithmen
- Iterationen als Loop ansehen
- Anfangs- und Endzustand annehmen
- Rundungsmodul

Die Verifikation einer Fliesskomma-Einheit

- Test der Kontrollstrukturen
  - Annahme: man hat *n* Iterationen
  - Zu zeigen:
    - Bei Zyklus 0 ist man im Ausgangszustand
    - Während Loop muss Zyklus *1..n* sein
    - Im finalen Zustand muss Zyklus *n+1* sein
    - Rundungszustand *n+2*
  - Beweis durch Induktion
    - Zyklus 0, init, Loopcount n, nächster Zustand: loop
    - Wenn (Zustand = loop, Loopcount > 0) dann ist der nächste Zustand = loop, Loopcount—
    - Wenn (Zustand = loop, Loopcount = 0) dann ist der nächste Zustand = final
    - Wenn Zustand = final dann ist der nächste Zustand = round

Die Verifikation einer Fliesskomma-Einheit

- Verifikation erfolgt mit Model Checking
- Kombination der Ergebnisse mit Theorem Proving

- Zweiter Schritt: Verifikation der Dateninvariante
  - Raum, aus dem die Werte zu wählen sind
  - Werte der Zahlen
- Auch hier Kombination von Model Checking und Theorem Proving

Ergebnisse bei Intels Pentium Pro

- Ziele der Verifikation
  - Finden und Konstruieren von Spezifikationen für alle von der FloatingPoint Execution Unit (FEU) durchzuführenden Operationen
  - Verifikation der korrekten Datenpfade für alle spezifizierten Operationen
  - Verifikation von Flags und Fehlermeldungen
  - Abdeckung aller Varianten von Operationen
  - Abdeckung aller Präzisionen und Rundungsarten

Ergebnisse bei Intels Pentium Pro

- Aufteilung in Arbeitsbereiche
  - Arbeitswerkzeuge
  - Entwicklung von Methoden zur Verifikation
  - Verifikation an sich
    - Hauptsächlich FADD, FSUB, FMUL…
- Verständnis der Spezifikation
- Verständnis des Designs
- Verständnis des Model Checkers, Theorem Provers
- Möglichkeit die Methoden bei anderen Prozessortypen wiederzuverwenden

FAZIT und Ausblick

Drei wichtige Erkenntnisse

- Fehler können schon in der Pre-Silicon Phase erkannt und behoben werden
- Ausgaben, da man qualifizierte Mitarbeiter für formale Verifikation benötigt
- Benötigte Kenntnisse sind noch nicht fester Bestandteil von Computer- und Elektrotechnik

FAZIT und Ausblick

- Formale Methoden immer beliebter in der Industrie
- Spart Kosten
- Einzug in kommerzielle Welt (Infineon)
- Intel setzt weiterhin auf Formale Methoden