Formale Prozessorverifikation
mit algebraischen Methoden

G.M. Greuel, W. Kunz, M. Wedler

Fachbereich Elektrotechnik und Informationstechnik

Fachbereich Mathematik
Characteristics of (Complete) Formal Module Verification

**Usage model: Largely automated code inspection**

- Fully formal solution for achieving best possible quality of complex modules
- Best possible means:
  - I / O of module uniquely, intuitively and compactly described by set of properties
  - for a processor implementation correct wrt. data sheet
- Quality comes along with high productivity:
  - 2-5 k lines of exhaustively verified RTL per person month
- Precise termination criterion for verification
  - Done, when properties have been set up, verified and completeness check succeeds
Simulation vs. Complete Formal Module Verification

- Simulation
- Complete formal module verification
- Setting up properties
- Proving completeness

Graph shows the comparison between simulation and complete formal module verification in terms of achieved quality and effort.
TriCore 2 Microprocessor System of Infineon

Architectural characteristics
- unified 32-Bit-RISC/DSP/MC architecture
- 853 instructions
- 6-stage superscalar pipeline
- multithreading extensions
- coprocessor support/floating point unit

Current Implementation
- 0.13 micron technology
- 3 mm² core area/8 mm² hardmacro area
- typical frequency ~ 500 MHz
- typical compiled code 1.5 MIPS / MHz
- 2 MMACS/MHz, 0.5 mW/MHz @ 1.5 V

Deployment
- primarily in automotive high-end
The Tricore project – some results

Performance of property checking

- 99.9 % of properties run in less than 2 minutes on solaris machine
- current property suite runs in 40 hours on 1 solaris machine

Productivity

- 2k LoC per person month exhaustively verified

Quality

- formal techniques identified bugs that are hard or impossible to find by conventional technique
- drastic reduction of errata sheets seems realistic
Infineon Tricore 2 project – scientific challenges

Advanced DSP features of embedded processors impose new challenges for formal verification

**MAC Unit:** multiply, multiply/add, multiply/subtract
satisfaction, rounding, shift-bits

\[ \approx 300 \text{ different instruction variants} \]

e.g.

- **MUL.H**
  - packed multiply
  - 2 parallel multiplications
  - 8 variants +12 special cases
  - 16 bit operands
  - 64 bit result

- **MADD(S).Q**
  - multiply/add in Q-format
  - 40 variants + 24 special cases
  - 32/16 bit operands
  - 64/32 bit results
  - some variants with saturation
Verification of Integer Pipelines

**Goal:** Prove that instructions are performed correctly

**Example:**

```plaintext
theorem mul;       // packed half word multiplication
assume:
    at t: command_dec(MUL,op1,op2);
    during[t,t+3]: no_reset;
    during[t,t+3]: no_cancel;
...
prove:
    at t+3: ip_res[31:0]
        == op1[15:0]*op2[15:0];
    at t+3: ip_res[63:32]
        == op1[31:16]*op2[31:16];
end
```
Pipelined data path with multiplication

- Data path implements large number of operations
- Complex interaction between control and data path

⇒ bit width reduction may miss corner cases

- Optimizations for performance and resource sharing
- Designer specifies arithmetic blocks at the bit-level

⇒ word level abstraction impossible
New Approach

- **Arithmetic Bit-Level Description (ABL)**
  network of elementary arithmetic operations
  - modulo 2 addition (XOR, HA, FA))
  - partial products
  - comparators

- **ABL Normalization** using arithmetic reasoning

Arithmetic correctness formally proved for all Tricore2 instructions with multiplication and all their variants ($\approx 300$) at full bit width
Zwischenbilanz

- Bearbeitung komplizierter arithmetischer Pipeline möglich

- Erfordert weitgehend getrennte Behandlung von Arithmetik und Logik

- Kompliziertes Zusammenspiel aus Arithmetik und Logik weiterhin problematisch

**Ziel:** neuer, einheitlicher Formalismus zur Behandlung heterogener Logik/Arithmetik-Strukturen auf Basis algebraischer Modellierung
Schaltungsbeschreibung auf RT-Ebene (HDL)

Vorverarbeitung durch Front-end des Property Checkers

Generierung der ABL-Beschreibung für arithmetische Blöcke

Normalisierung der ABL-Beschreibung und Beweis der Eigenschaften

Abstraktion der ABL-Blöcke (z.B. Wortebene)

Algebraische Modellierung des Gesamtsystems (Arithmetik + Logik)

Algebraische Beweismethoden durch Erweiterung von SINGULAR

Eigenschaften für arithmetische Teile

Eigenschaften für Gesamtsystem

Zertifizierung oder Gegenbeispiel

Beitrag der Elektrotechnik

Beitrag der Mathematik