Using Static Program Analysis Techniques for Energy Analysis

Simon Wegener, AbsInt GmbH
Outline

- Who am I / What do I do
- Introduction to our timing analysis framework
- Ideas for integrating energy analysis in our framework
- Discussion / Tool demo
Outline

- Who am I / What do I do
- Introduction to our timing analysis framework
- Ideas for integrating energy analysis in our framework
- Discussion / Tool demo
AbsInt Angewandte Informatik GmbH

- Provides advanced development tools for embedded systems, and tools for validation, verification, and certification of safety-critical software.
- Founded in February 1998 by six researchers of Saarland University, Germany, from the group of programming languages and compiler construction of Prof. Dr. Dr. hc. mult R. Wilhelm.
- Privately held by the founders.

Selected customers:
Key Products: AI-based Static Analyzers

**aiT WCET Analyzer**
- Proving the correct timing behavior
- Safe upper bounds on the worst-case execution time of tasks in real-time systems

**StackAnalyzer**
- Excluding stack overflows
- Safe upper bounds on maximal stack usage of tasks

**Astrée**
- Proving the absence of runtime errors (division by zero, arithmetic overflow, invalid pointer accesses, etc.) in C programs
Outline

- Who am I / What do I do
- Introduction to our timing analysis framework
- Ideas for integrating energy analysis in our framework
- Discussion / Tool demo
aiT Timing Analysis Framework

Executable Program

AIS File

CFG Builder

Loop Trafo

CRL File

Static Analysis

Loop/Value Analyzer

Cache/Pipeline Analyzer

CRL File

Path Analysis

ILP Generator

ILP Solver

Evaluation

WCET, Visualization
aiT Timing Analysis Framework

Executable Program → CFG Builder → Loop Trafo → CRL File

AIS File

Static Analysis → Cache/Pipeline Analyzer → CRL File

Loop/Value Analyzer

Path Analysis → ILP Generator → ILP Solver → Evaluation → WCET, Visualization
Control Flow Reconstruction
aiT Timing Analysis Framework

Executable Program
→ CFG Builder
→ Loop Trafo
→ CRL File

AIS File

Static Analysis
→ Loop/Value Analyzer
→ Cache/Pipeline Analyzer
→ CRL File

Path Analysis
→ ILP Generator
→ ILP Solver
→ Evaluation
→ WCET, Visualization

AbsInt
Loop Transformation
aiT Timing Analysis Framework

Executable Program

- AIS File
  - CFG Builder
    - Loop Trafo
      - CRL File

Static Analysis

- Loop/Value Analyzer
  - Cache/Pipeline Analyzer
    - CRL File

Path Analysis

- ILP Generator
  - ILP Solver
    - Evaluation
      - WCET, Visualization
Loop / Value Analysis

- Computes the contents of registers and memory cells.
- Based on Abstract Interpretation [Cousot 1977].
- Used mainly for three things:
  - Bound the number of loop iterations.
  - Determine the addresses of memory accesses.
  - Exclude infeasible program paths.
Abstract Interpretation

- Substitute *concrete* values with *abstract* values.
- The abstract values represent sets of concrete values.
# Abstract Interpretation

<table>
<thead>
<tr>
<th></th>
<th>Concrete Domain</th>
<th>Abstract Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value</strong></td>
<td><strong>Integer</strong></td>
<td><strong>Integer Interval</strong></td>
</tr>
<tr>
<td></td>
<td>8, −30</td>
<td>[0..8], [−30..4]</td>
</tr>
<tr>
<td><strong>Operations</strong></td>
<td>8 + (−30) → −22</td>
<td>[0..8] + [−30..4] → [−30..12]</td>
</tr>
<tr>
<td><strong>Partial Order</strong></td>
<td>3 ≤ 8 → true</td>
<td>[3..4] ⊆ [0..8] → true</td>
</tr>
<tr>
<td><strong>Join</strong></td>
<td></td>
<td>[3..4] ⊔ [0..8] → [0..8]</td>
</tr>
</tbody>
</table>
Semantic Reinterpretation

- **Generate** analyzers from description of concrete semantics and abstract domains.
- **Transformer Specification Language (TSL)** [Lim & Reps 2013]
Semantic Reinterpretation

- Example: PowerPC instruction `stwu r1, -1232(r1)`

```cpp
template<typename T>
void instruction_stwu (State<T> & state,
                      Resource<T> rS, Resource<T> d, Resource<T> rA)
{
    T base = state.get(rA);
    T offset = d.asImmediateConstant();
    T val = state.get(rS);

    T ea = T::add(base, offset);

    state.set(ea, val);
    state.set(rA, ea);
}
```
aiT Timing Analysis Framework

Executable Program → AIS File

CFG Builder → Loop Trafo → CRL File

Static Analysis
- Loop/Value Analyzer → Cache/Pipeline Analyzer

Path Analysis
- ILP Generator → ILP Solver → Evaluation → WCET, Visualization
Cache / Pipeline Analysis

- Determines whether a memory access is a cache hit or not.
- Models the flow of an instruction through the pipeline.
  - This includes stalls, conflicts, short paths, branch prediction, speculation ...
- Computes the number of cycles a basic block resides inside the pipeline.
aiT Timing Analysis Framework

Executable Program

- CFG Builder
  - Loop Trafo
  - CRL File

Static Analysis
- Loop/Value Analyzer
  - Cache/Pipeline Analyzer
  - CRL File

Path Analysis
- ILP Generator
  - ILP Solver
  - Evaluation
  - WCET, Visualization
Path Analysis

- Computes the maximal overall execution time from the execution times of the individual basic blocks.
- The control flow graph is encoded as a flow problem in an ILP.
  - The first block in the program has a maximum count of 1.
  - All other blocks preserve the flow: the sum of the incoming flow must equal the sum of the outgoing flow.
  - Loop back edges are bounded.
- The objective function is the sum of all blocks with their associated costs (i.e. basic block WCETs).
- A solver like clpsolve or CPLEX is used to compute a maximized solution.
Outline

- Who am I / What do I do
- Introduction to our timing analysis framework
- Ideas for integrating energy analysis in our framework
- Discussion / Tool demo
Energy as Property of the Architecture

- **Coarse grained:**
  - Instruction stwu uses 10 mW
  - Instruction addi uses 2 mW
  - ...

- **Fine grained**
  - Using the ALU costs 1 mW
  - Accessing the register file costs 1 mW
  - Fetching an instruction from the cache costs 2 mW
  - Accessing the memory costs 7 mW
  - ...

(All numbers are made up)
Energy as Part of Program Semantics

Executable Program → AIS File → CFG Builder → Loop Trafo → CRL File

Static Analysis:
- Loop/Value Analyzer → Cache/Pipeline Analyzer → CRL File

Path Analysis:
- ILP Generator → ILP Solver → Evaluation → WCET, Visualization
Energy as Part of Program Semantics

- Define a base unit for the target architecture
  - One base unit equals 3 mW for the MPC5554.
  - One base unit equals 1 mW for the ARM Cortex-M3.
  - ...

- Define an abstract domain for energy consumption and reinterpret the instruction semantics.
  - A memory access costs 7 base units.
  - A register access costs 2 base units.
  - An addition costs 1 base unit.
  - A multiplication costs 5 base units.
  - ...

(All numbers are made up)
Outline

- Who am I / What do I do
- Introduction to our timing analysis framework
- Ideas for integrating energy analysis in our framework
- Discussion / Tool demo
References

- [Cousot 1977]
  Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints.

- [Lim & Reps 2013]
  TSL: A system for generating abstract interpreters and its application to machine-code analysis.
  In ACM Trans. on Program. Lang. and Syst. 35, 1, Article 4 (April 2013), 59 pages.