A Retargetable Software Timing Analyzer Using Architecture Description Language

Xianfeng Li: Peking Univ. (PKU)
Abhik Roychoudhury: National Univ. of Singapore (NUS)
Tulika Mitra: National Univ. of Singapore (NUS)
Prabhat Mishra: Univ. of Florida (UFL)
Xu Cheng: Peking Univ. (PKU)
Timing Analysis

- Problem statement
  - Given a program and an architecture, estimate the maximum/minimum execution time of the program for all possible inputs

- Worst Case Execution Time (WCET)
  - Essential information for real-time scheduling
Worst-Case Execution Time (WCET)

Distribution of Execution Time

- Observed BCET
- Estimated BCET
- Observed WCET
- Estimated WCET

Measured execution times
Possible execution times
Timing Predictability

Time
Timing Analysis

- Problem statement
  - Given a program and an architecture, estimate the maximum/minimum execution time of the program for all possible inputs

- Worst Case Execution Time (WCET)
  - Essential information for real-time scheduling

- Static WCET Analysis
  - Derive a conservative estimate on WCET
  - Program path + instruction timing
Microarchitecture Modeling

- Microarchitecture innovations
  - For average case performance
  - Poor predictability – bad for real-time

- Microarchitecture modeling started 15+ years ago
  - Human understanding
  - Custom modeling algorithms
  - Handcrafted code

- A retargetable framework is needed
  - Input:
    - Program
    - Processor spec. (Architecture Description Language)
  - Output: WCET
WCET Analysis Framework

- Integer Linear Programming (ILP) based analysis

maximize

\[ T_{\text{prog}} = \sum N_B \times C_B \]

where

\[ B \in \text{BasicBlocks}(\text{prog}) \]

\( N_B \): execution count of \( B \)

\( C_B \): WCET of \( B \) affected by hardware

subject to Control Flow Graph (CFG) constraints
Basic Block Timing Analysis ($C_B$)

- Execution graph:
  - Nodes: <instr, stage>
  - Relations:
    - dependences
    - contentions
    - parallelisms

- Time interval based
  - Not simulation: all cases are covered
  - No state space explosion: avoid enumeration on single time points

Algorithm outline:
Initialization(G);
repeat {
  for each node $v$ in G {
    earliest_time($v$);
    latest_time($v$);
  }
} until (a fixed point is reached)

Modeling Out-of-Order Processors for WCET Analysis
Li et al., Real-Time Systems Journal, Nov 2006
Architecture Description Language (ADL)

- Formal description of processor architecture
  - ISA + Microarchitecture

- Existing ADLs:
  - HMDES, LISA, MADL, UPFAST, EXPRESSION...

- Investigated retargeting issues
  - Compilation, simulation, synthesis, validation
EXPRESSION ADL

- Developed by Prof. Dutt’s group at UC-Irvine

Processor ADL

ISA

Microarchitecture

Structure

Behavior

described in main ADL file

described in separate C++ files
An Example ADL (acesMIPS)

(Pipeline section
(Pipeline fetch decode read_execute wb)
(Read_execute (alternate read_ex0 ... read_ex4)
(read_ex0(pipeline alu1_read alu1_ex))
...
(read_ex4(pipeline ldst_read ldst_ex))

(Architecture section
(Subtype unit fetchunit decodeunit operadunit ...)
...
(decodeunit decode
(capacity 12)
(opcodes all)
...

// DerivedUnit.h
class DecodeUnit : public SimpleUnit
{
  private:
    int instBufsize;
    ...
}
From ADL To Execution Graph

- Execution graph: nodes + relations

- processor description

  - ADL architecture spec.
  - ADL component library

  - graph
  - node relations

  - execution graph

- WCET component library

  - component mapping

  - node relations

  - graph

  - execution graph

25-Jan-2007

12th ASP-DAC, Japan
WCET Component Library

- Models characterizing WCET properties of components

- Goals:
  - Abstract, architecture independent
  - ADL component to WCET component mapping is easy

- Observations:
  - Pipeline path: a series of **processing elements** interfaced by **storage elements**
  - Behaviors of **PEs** are often dictated by **SEs**
  - Limited classes of **SEs** based on their timing behavior
WCET Component Library (cont)

- Array model
  - access latency

Diagram:
```
Array (I-Cache) --IF-- [l, u] <l, IF>
```
WCET Component Library (cont)

- Array model
  - access latency

- FIFO model
  - producer-consumer
  - size limit
  - parallelism limit

Diagram:
- Array (I-Cache) connected to IF connected to FIFO (I-Buffer) connected to ID
- Nodes: <I, IF>, <I, ID>, <I-n, ID>, <I+p, IF>
  - Edges:
    - <I, IF> to <I, ID>
    - <I-n, ID> to <I+p, IF>
    - [l, u] connecting nodes
WCET Component Library (cont)

- **Array model**
  - access latency

- **FIFO model**
  - producer-consumer
  - size limit
  - parallelism limit

- **Pool model**
  - No statically decided access order
  - No general timing characterization
  - Some can be abstracted with limited retargetability (Out-of-order issue queue)
Case Study: acesMIPS

- A MIPS R4000-like processor (by EXPRESSION group)

- Retargeting effort: ~2 weeks
  - ADL-WCET component mapping
  - Handcrafted code
  - Simulation, estimation, and result analysis

- One-time effort: ~2 months
  - WCET component library
  - execution graph construction and WCET analysis
  - EXPRESSION-WCET interface

Effort on writing acesMIPS ADL description is not accounted

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Simulation</th>
<th>Estimation</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>fdct</td>
<td>10310</td>
<td>12310</td>
<td>1.19</td>
</tr>
<tr>
<td>fft</td>
<td>2937925</td>
<td>3880257</td>
<td>1.32</td>
</tr>
<tr>
<td>isort</td>
<td>149057</td>
<td>175724</td>
<td>1.18</td>
</tr>
<tr>
<td>matmul</td>
<td>57431</td>
<td>72150</td>
<td>1.26</td>
</tr>
<tr>
<td>matsum</td>
<td>442556</td>
<td>482570</td>
<td>1.09</td>
</tr>
</tbody>
</table>
Related Work

- A retargetable technique for predicting execution time of code segments

- Retargetable static software timing analysis
  Chen, Malik, August [ISSS 2001]

- Efficient analysis of pipeline models for WCET computation
  Wilhelm [WCET Workshop, 2005]
Related Work [RTS 1994]

- Basic idea: micro-analysis
  - Instr => primitive operations
  - Timing analysis guided by pattern-driven timing rules

- Pros
  - Primitive ops are architecture-independent

- Cons
  - Timing rules are architecture-dependent
    => based on understanding of an architecture’s timing model
  - Based on simpler architectures
    => not powerful enough for modern processor
Related Work [ISSS 2001]

- Basic idea: MESCAL and MADL
  - MESCAL => compiler and simulator
  - Obtain WCET of a code fragment by simulation

- Pros
  - Avoid most handcrafted code

- Cons
  - Simulation does not guarantee WCET
  - For VLIW with restrictions on pipeline complexities

- Comparison with our work
  - Retargetable simulation vs retargetable static analysis
Related Work [WCET 2005]

- Basic idea: HDL spec
  - Define pipeline analysis as computations on FSMs
  - With the help on Binary Decision Diagram (BDD)

- Pros
  - Automated analysis

- Cons
  - State explosion problem

- Comparison with our work
  - Based on higher level of abstraction
  - A compact pipeline model based on execution graph
Summary

- The major barrier of WCET analysis is retargetability

- Retargetable WCET analysis framework
  - Processor ADL specification as input
  - Execution graph based basic block timing analysis
  - WCET component library
  - ADL to execution graph

- Case study (acesMIPS)