Tuning the WCET of Embedded Applications

Wankang Zhao ¹, Prasad Kulkarni ¹, David Whalley ¹, Christopher Healy ², Frank Mueller ³, Gang-Ryung Uh ⁴

1. Florida State University
2. Furman University
3. North Carolina State University
4. Boise State University

Why Reduce the WCET?

- more likely to meet timing constraints
- can lower clock rate to reduce power consumption

Outline of Rest of Presentation

- Related Work
- Research Framework
  - target architecture, compiler, timing analyzer
- Functionality
  - include quick demo
- Experiments
- Future Work
- Conclusions

Our Approach

- interactive compilation system
- timing analyzer invoked on demand
- automatically searches for an optimization phase sequence that best reduces the WCET
Related Work

- methods to reduce WCET in critical sections
  - Marlowe, et al, System Integration ’92
  - Hong, et al, PLDI ’93
- reduce WCET on a dual instruction set processor
  - Lee, et al, WCET ’03
- genetic algorithms to search for effective optimization sequences to improve speed, space, or a combination of both
  - Cooper, et al, LCTES ’99
  - Kulkami, et al, LCTES ’03

Framework for This Research

Target Architecture:
StarCore SC100 Processor

- A digital signal processor for embedded systems.
- No caches and no operating system.
- A simple five stage pipeline machine with transfer-of-control and target misalignment penalties.
- The size of instructions varies from 1 word to 5 words.

Our Timing Analyzer

- Calculates WCET for each path, loop, and function in the program.

Features
- WCET pipeline analysis - RTSS ’95
- WCET cache analysis - RTSS ’94, RTAS ’97
- automatically calculates the number of loop iterations - RTAS ’98
- detects infeasible paths due to branch constraints - RTAS ’99
Estimating WCET with Transfer of Control Penalties

- What is the WC path?

VISTA: Functionality

- Provides a graphical display of the low-level program representation.
- Directs order and scope in which the optimization phases are applied.
- Shows feedback on the WCET and code size improvement.
- Reverses previously applied transformations.
- Uses a genetic algorithm to search for the best order of optimization phases.

VPO Interactive System for Tuning Applications (VISTA)

- Has been previously used to tune applications for ACET and code size.
- Now interacts with our timing analyzer to determine WCET improvement.

Main Window of VISTA
Select Optimization Phases

Main Window of VISTA (again)

Select the Candidate Phases

Selecting Search Options
Experiments

- Evaluated effectiveness of VISTA's GA search for improving WCET.
  - Each phase is considered a gene.
  - Each sequence of phases is considered a chromosome.
- Much faster to interact with a timing analyzer to obtain WCET than a simulator to obtain ACET.

Candidate Optimization Phases

- branch chaining
- remove useless blocks
- remove unreachable code
- common subexpression elimination
- register allocation
- block reordering
- minimize loop jumps
- remove useless jumps
- loop transformations
- merge basic blocks
- evaluation order determination
- dead assignment elimination
- strength reduction
- reverse jumps
- instruction selection
Genetic Algorithm (GA) Parameters

- Sequence length (chromosome) is 1.25 times the number of phases that were successfully applied by the batch compiler.
- Population size: 20 sequences
- Generations: 200
- 4 sequences are replaced by crossover operations.
- Mutation rate: 10% lower half, 5% upper half
- 3 different fitness criteria:
  - 100% WCET, 100% code size, 50% WCET and 50% code size

DSPstone Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>convolution</td>
<td>perform a convolution filter</td>
</tr>
<tr>
<td>complex_update</td>
<td>performs a single mac operation on complex values</td>
</tr>
<tr>
<td>dot_product</td>
<td>computes the product of two vectors</td>
</tr>
<tr>
<td>fir</td>
<td>perform a FIR filter</td>
</tr>
<tr>
<td>fir2dim</td>
<td>perform a FIR filter on a 2D image</td>
</tr>
<tr>
<td>lms</td>
<td>perform a FIR filter on one section</td>
</tr>
<tr>
<td>lms</td>
<td>perform a FIR filter on multiple sections</td>
</tr>
<tr>
<td>matrix</td>
<td>least mean square adaptive filter</td>
</tr>
<tr>
<td>matrix_1x3</td>
<td>computes matrix product of two 10x10 matrices</td>
</tr>
<tr>
<td>n_complex_updates</td>
<td>computes matrix product of 3x3 and 3x1 matrices</td>
</tr>
<tr>
<td>n_complex_updates</td>
<td>performs a mac operation on an array of complex</td>
</tr>
<tr>
<td>real_update</td>
<td>performs a mac operation on an array of data</td>
</tr>
<tr>
<td>real_update</td>
<td>performs a single mac operation</td>
</tr>
</tbody>
</table>

Other Benchmarks

<table>
<thead>
<tr>
<th>Program</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fft</td>
<td>128 point complex FFT</td>
</tr>
<tr>
<td>summidall</td>
<td>sums the middle half and all elements of a 1000 integer vector</td>
</tr>
<tr>
<td>summinmax</td>
<td>sums the minimum and maximum of the corresponding elements of two 1000 integer vectors</td>
</tr>
<tr>
<td>sumnegpos</td>
<td>sums the negative, positive, and all elements of a 1000 integer vector</td>
</tr>
<tr>
<td>sumoddEven</td>
<td>sums the odd and even elements of a 1000 integer vector</td>
</tr>
<tr>
<td>sym</td>
<td>test if a 100x100 matrix is symmetric</td>
</tr>
</tbody>
</table>

WCET vs. Observed Cycles
Tuning for WCET

Tuning for Code Size

50% WCET and 50% Code Size

Result of the Three Fitness Criteria
Future Work

- Develop compiler optimizations that use worst-case path information to improve WCET.
- Example:
  - change order of basic blocks to reduce transfer of control penalties for worst-case paths

Conclusions

- Developed the first system where a compiler can invoke a timing analyzer on demand.
- Showed that WCET can be used as a fitness value to a genetic algorithm to find an effective optimization sequence.
- WCET and code size were simultaneously improved by 6% and 5%, respectively.