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

Our Approach

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

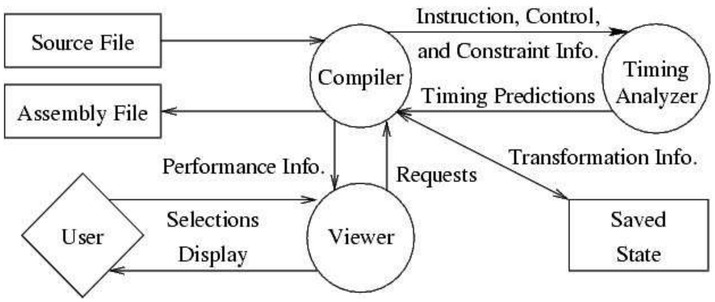
Outline of Rest of Presentation

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

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
  - Kulkarni, et al, LCTES '03

Framework for This Research



Target Architecture: StarCore SC100 Processor

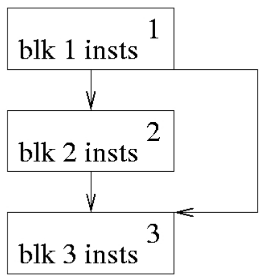
- A digital signal processor for embedded systems.
- No caches and no operating system.
- A simple five stage pipeline machine with transfer-of-control and target misalignment penalties.
- The size of instructions varies from 1 word to 5 words.

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?



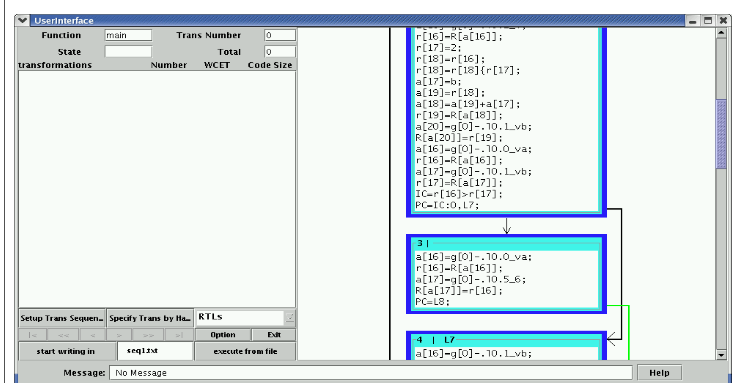
VPO Interactive System for Tuning Applications (VISTA)

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

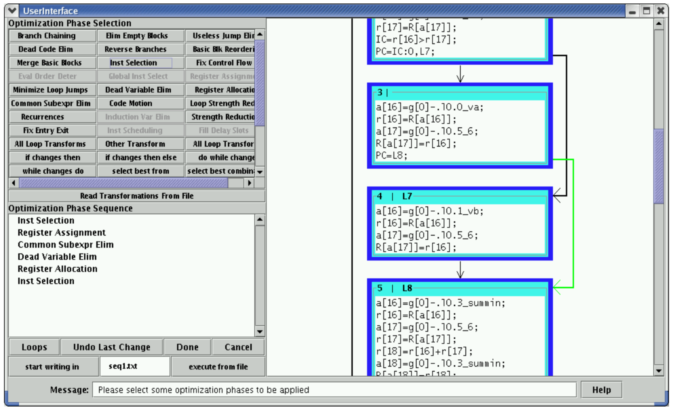
VISTA: Functionality

- 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.

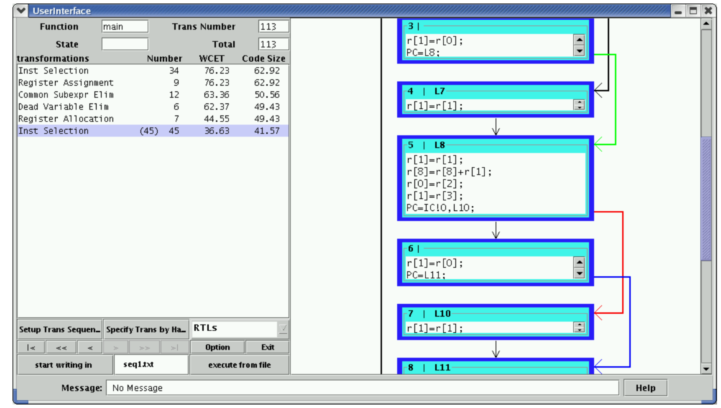
Main Window of VISTA



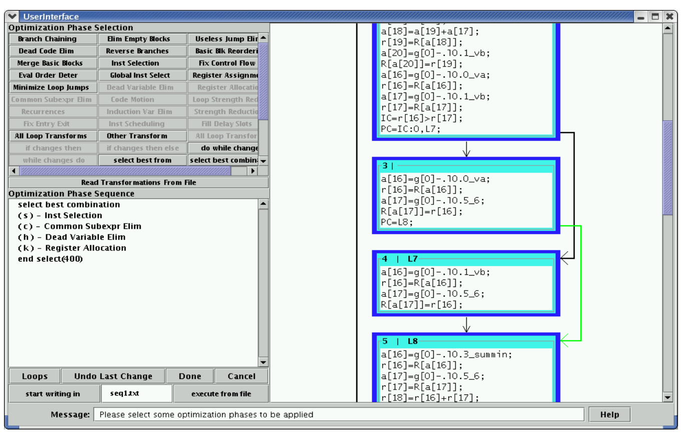
Select Optimization Phases



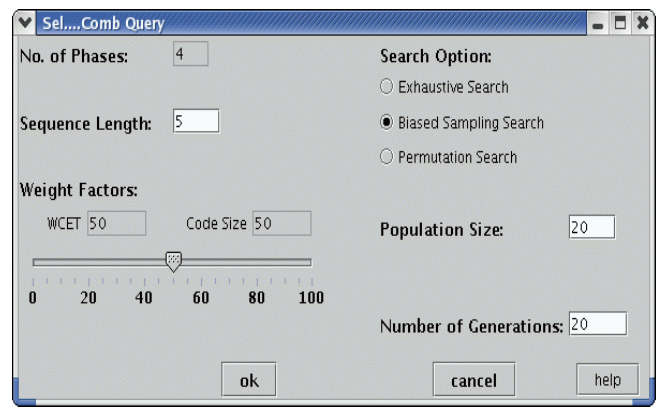
Main Window of VISTA (again)



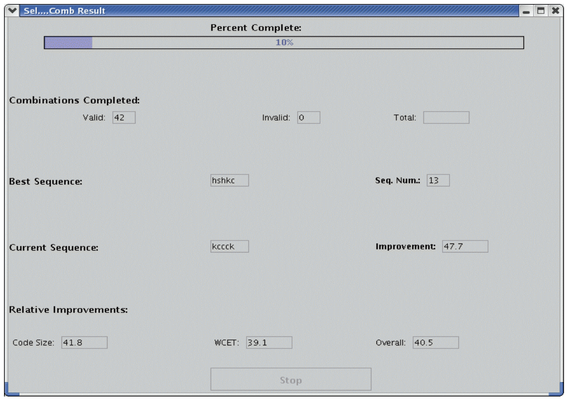
Select the Candidate Phases



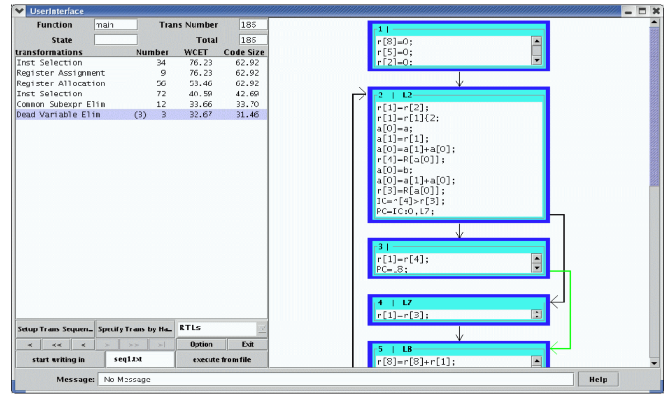
Selecting Search Options



Window Showing the Search Status



GA Results



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

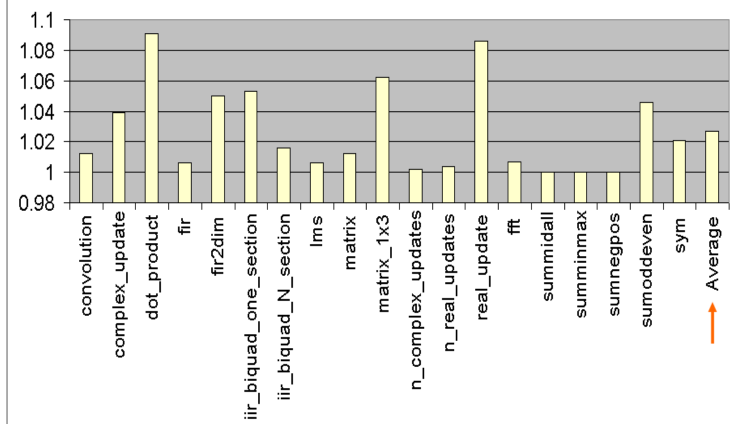
DSPstone Benchmarks

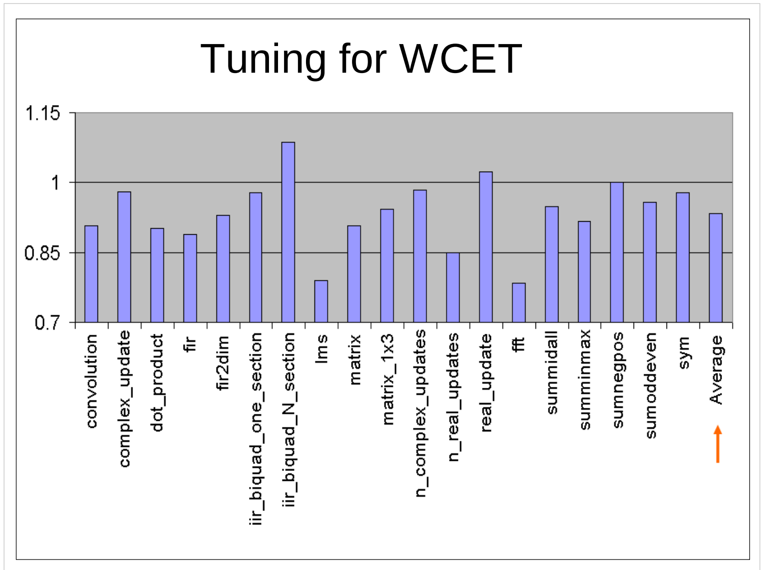
Program	Description
convolution	perform a convolution filter
complex_update	performs a single mac operation on complex values
dot_product	computes the product of two vectors
fir	perform a FIR filter
fir2dim	perform a FIR filter on a 2D image
iir_biquad_one_section	perform a IIR filter on one section
iir_biquad_N_sections	perform a IIR filter on multiple sections
lms	least mean square adaptive filter
matrix	computes matrix product of two 10x10 matrices
matrix_1x3	computes matrix product of 3x3 and 3x1 matrices
n_complex_updates	performs a mac operation on an array of complex values
n_real_updates	performs a mac operation on an array of data
real_update	performs a single mac operation

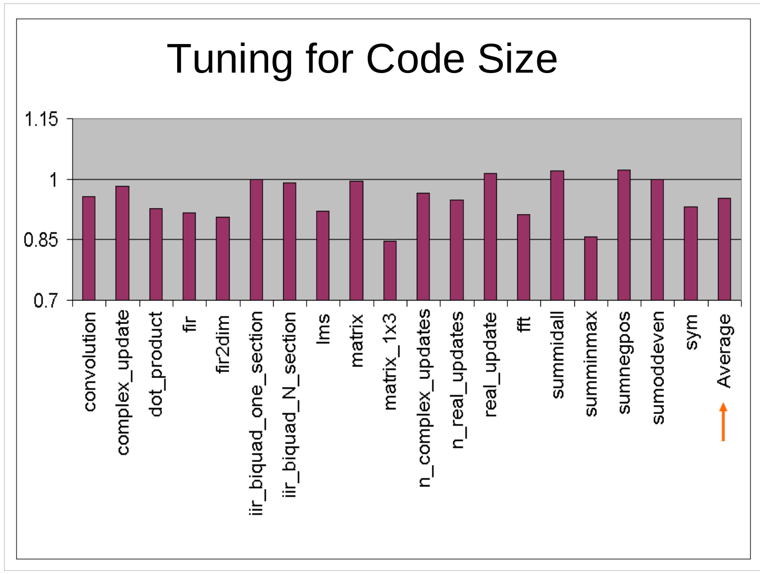
Other Benchmarks

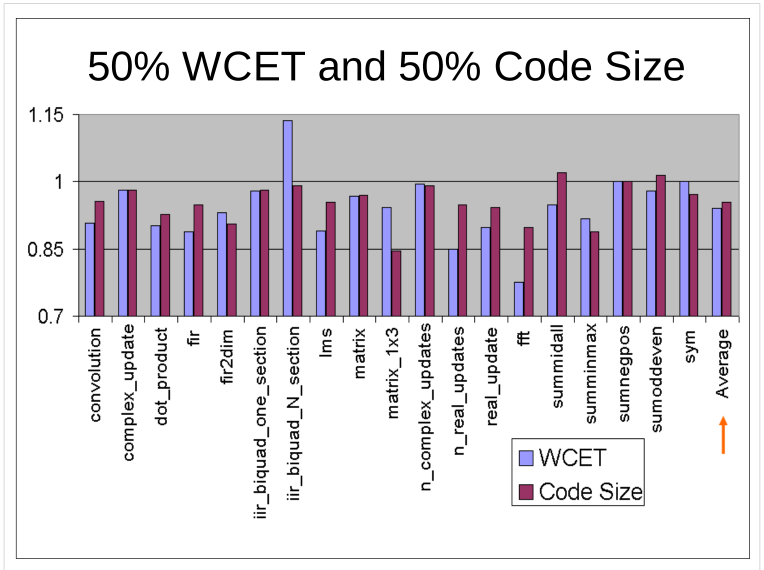
Program	Description
fft	128 point complex FFT
summidall	sums the middle half and all elements of a 1000 integer vector
summinmax	sums the minimum and maximum of the corresponding elements of two 1000 integer vectors
sumnegpos	sums the negative, positive, and all elements of a 1000 integer vector
sumoddeven	sums the odd and even elements of a 1000 integer vector
sym	test if a 100x100 matrix is symmetric

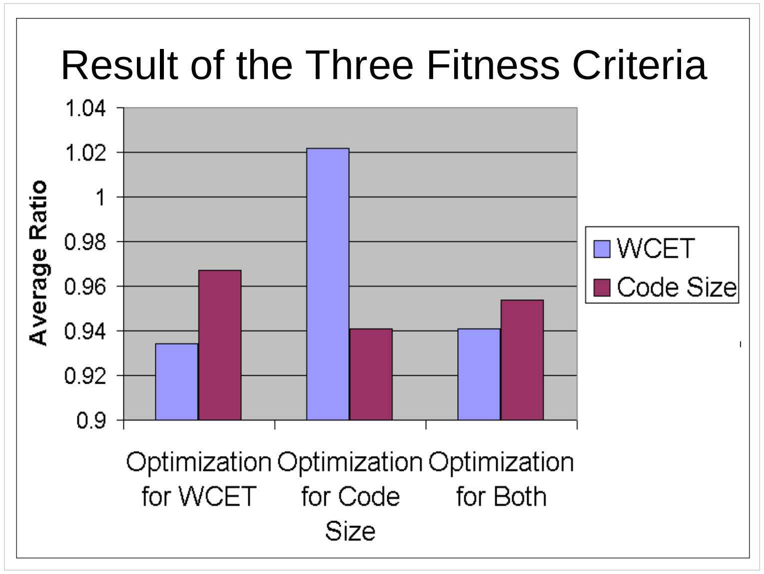
WCET vs. Observed Cycles




Tuning for WCET
1.15
1
0.85
0.7
convolution
complex_update
dot_product
fir
fir2dim
iir_biquad_one_section
iir_biquad_N_section
lms
matrix
matrix_1x3
n_complex_updates
n_real_updates
real_update
fft
summidall
summinmax
sumnegpos
sumoddeven
sym
Average


Tuning for Code Size
1.15
1
0.85
0.7
convolution
complex_update
dot_product
fir
fir2dim
iir_biquad_one_section
iir_biquad_N_section
lms
matrix
matrix_1x3
n_complex_updates
n_real_updates
real_update
fft
summidall
summinmax
sumnegpos
sumoddeven
sym
Average


50% WCET and 50% Code Size
1.15
1
0.85
0.7
convolution
complex_update
dot_product
fir
fir2dim
iir_biquad_one_section
iir_biquad_N_section
lms
matrix
matrix_1x3
n_complex_updates
n_real_updates
real_update
fft
summidall
summinmax
sumnegpos
sumoddeven
sym
Average
WCET
Code Size


Result of the Three Fitness Criteria
Average Ratio
1.04
1.02
1
0.98
0.96
0.94
0.92
0.9
Optimization for WCET
Optimization for Code Size
Optimization for Both
WCET
Code Size

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.