The VIVACE Tools: Variable Implementations for Vector Architecture and Compiler Exploration

Corinna G. Lee
Parham Aarabi, Derek DeVries, Courtney Gibson, Kristin Hofstee, Mark Stoodley

Department of Electrical and Computer Engineering
University of Toronto
Toronto, Ontario, Canada

The VIVACE Tools

SUN:vsuif (Derek DeVries)

- vectorizes C (and Fortran) programs and produces stripmined VMIPS assembly code

SUN:vgas (Courtney Gibson)

- assembles VMIPS (and Torrent) assembly code into an ELF MIPS object file

SGI:vmable (Courtney Gibson, Parham Aarabi)

- executes a VMIPS (or MIPS) executable and generates compressed, pixie address traces
- use to show functional correctness of vsuif-generated code or handwritten VMIPS code

SGI:vcello (Parham Aarabi)

- times execution of programs (i.e., address traces) executing on different types of processors (in-order superscalar, pit-of-order superscalar, vector + in-order superscalar) and any 2-level cache memory system
- able to modify micro-architectural parameters at run-time
- generates many histograms and tables characterizing the utilization and performance of different parts of the processor: IPC, re-order buffer use, branch statistics, cache use, etc.
- includes debug option for visualizing timing of instruction execution

SGI,SUN:Unix scripts (Kristin Hofstee, Mark Stoodley)

- facilitates use of tools and adding new benchmarks to the performance suite
- generates performance graphs

Architecture and Compiler Exploration

varying microarchitectural features

- run-time input to vcello via configuration files (see next slide for an example)

varying vector length, number of vector registers

- 1 or 2 compile-time constants in vsuif

new compiler algorithm

- modify vsuif

new ISA feature

- vsuif: modify to generate new instructions
- vgas: update tables in opcodes.c program that automatically generates opcode-encoding tables
- vmable: add code that performs the execution stage of the new instructions
- vcello: may not require changes if issue rules are not affected

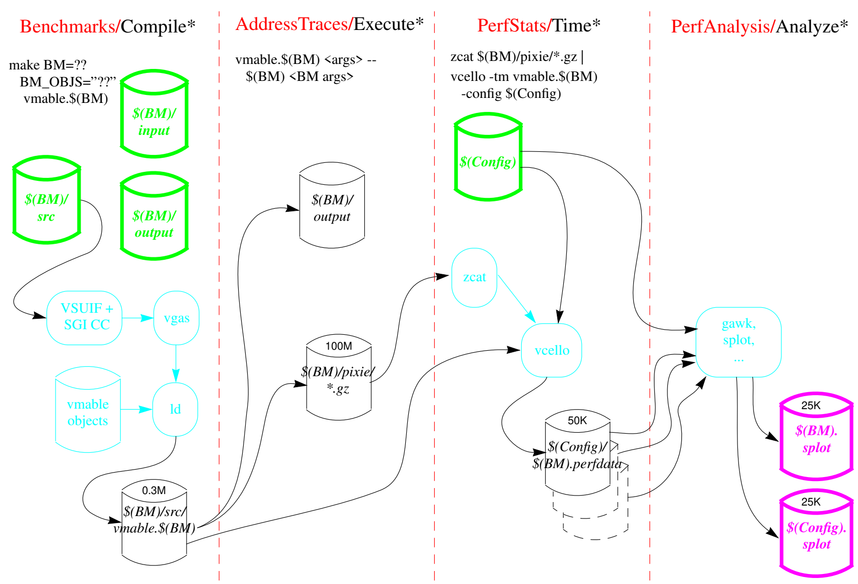
different ISA

- replace vsuif, vgas, vmable
- vcello: change idecode.c to decode new ISA

Configuration Defaults

```
;default cello configs except for in_order and chaining

p_to_s_latency              0
p_to_m_latency              0

icache_size_log             15
icache_linesize_log         9
i_associativity_log_size    1

dcache_size_log             15
dcache_linesize_log         9
d_associativity_log_size    1
dcache_banks_log            4

scache_size_log             23
scache_linesize_log         9
s_associativity_log_size    1
scache_banks_log            4

memory_access_time          0

cpu_clock_rate_mhz          250

num_int_FUs                 4
num_branch_FUs              1
num_fp_FUs                  1
num_ldst_FUs                2
num_vector_FUs              1

dref_handlers               2
physical_rf_size            256

chaining
in_order
reorder_buf_size            48
fetch_width                 16
issue_width                 4
graduate_width              4

vector_pipe_width           8

*_latency                   0
```

Use of Tools to Generate and Analyze Performance Data



Directory Structure for Generating and Analyzing Performance Data

```
VectorExpt
  bin
    Makebin*
    SUN
      Makefile.vsuif
      breathe, vgas
      vsuifbin
        {<vsuif passes & scripts>}
    SGI
      Makefile
      {vmable*.o, vcello}
    Makefile@, Makefile.vsuif@
    vmable*.o@, ...., vsuifbin@
  Benchmarks/
  AddressTraces/
  PerfStats/
  PerfAnalysis/
```


Directories for Compiling and Executing Benchmarks

```
Benchmarks
  README, Compile*
  Makefile@, Makefile.vsuif@
  (breathe, vgas, vsuifbin)
  {<BM>}
    input/
    output/
    src
      o_files
      *.vc, *.c, *.h
      *.F, *.f
    {superscalar, vector}
      *.[c,vc]@, *.h@
      *.o
      vmable.$BM
```

```
AddressTraces
  README, Execute* (vmable)
  {<BM>}
    args
    {superscalar, vector}
      input/
      output/
      pixie/*.gz
```


Directories for Timing and Analyzing Benchmarks

```
PerfStats
  README, Time* (vcello)
  config.default
  {SS,V}config{A,B,....}
    $config.desc
    {<BM>}.data
```

```
PerfAnalysis
  Analyze* (gawk, splot)
  configurations
    {<BM>}.{perf,splot,ps}
  benchmarks
    {<config>}.{perf,splot,ps}
```

Status

vsuif

- able to compile vectorizable C code; vectorizable FORTRAN path still needs work

vgas

- assembles all VMIPS instructions

vmable

- simulates integer vector instructions and most floating-point vector instructions correctly

vcello

- simulates execution behavior of vector instructions: checks whether vector instructions can issue based on availability of vector units, vector registers, memory, chaining, etc.
- simulates cache accesses for vector memory instructions
- able to modify micro-architectural parameters at run-time using configuration file
- need to verify performance results: use vchart facility to visualize instruction timing

Unix scripts

- able to add new benchmark to performance suite easily: m88ksim, alvinn, ijpeg
- able to compile vectorizable code with vsuif and link with SGI C-compiled code easily
- able to automatically generate performance graphs

Demonstration

Real-time Examples:

demonstrate entire tool flow; correctness of vsuif, vgas, vmable; robustness of tools

idea1: cycle.vc

- IDEA program that reads a message from stdin, encrypts it, then decrypts it
- single, fairly complex C file that is correctly compiled by vsuif

ideaN: main.c, interchanged.vc, encode.c

- same as above except program split into scalar-compiled and vector-compiled files

Berkeley application: fft.vc(?)

- demonstrate ease of adding new benchmark

Prepared Examples:

demonstrate performance data produced by vcello

decrypt.inter, decrypt.unroll

- IDEA decryption (from PGP software package) using interchanged version and unrolled version

Summary

VIVACE: a collection of tools for exploring architecture and compiler alternatives for vector processing

- consists of vsuif, vgas, vmable, vcello, and Unix scripts

architecture and compiler exploration

- microarchitecture studies: run-time configuration input to vcello
- register studies: simple re-compile of vsuif
- compiler studies: modify vsuif
- ISA studies: changes to vsuif, vgas, vmable, and maybe vcello