Tortola: Addressing Tomorrow’s Computing Challenges through Hardware/Software Symbiosis

Kim Hazelwood
University of Virginia
Intel Corporation, VSSAD
Modern Computing Challenges

• Performance
• Power
  – Energy consumption, max instantaneous power, di/dt
• Temperature
  – Total heat output, “hot spots”
• Reliability
  – Neutron strikes, alpha particles, MTBF, design flaws
• Approaches: Circuit, microarchitecture, compiler
• Constraint: Fixed HW-SW interface (e.g., x86)
Typical Approaches

- Optimize using SW or HW techniques in isolation
- Performance
  - SW: Compile-time optimizations
  - HW: Architectural improvements, VLSI technology
- Reliability: Code/data duplication (HW or SW)
- Power & Temperature
  - HW control mechanisms
  - Profile + recompile cycle
Modern Design Constraints

Compilers – “Compile once, run anywhere”
  – Cannot ship “MS Office for 1Q05 batch of Pentium-4 3GHz, > 1GB RAM, BrandX power supply, located in high altitudes…”

Microarchitecture – Limited window of application knowledge (past must predict the future)

VLSI – Guaranteed correctness, reliability

We currently must optimize for the common case (but must design for the worst case)
The Power of Virtualization

• A HW-SW interface *layer*

```
SW Applications

Binary Modifier

HW

Initially

x86

Eventually

x86

SWI

HWI
```
Dynamic Binary Modification

- Creates a modified code image at run time

**Examples:**
- Dynamo (HP)
- DAISY/BOA (IBM)
- CMS (Transmeta)
- Mojo (Microsoft)
- Strata (UVa)
- Pin (Intel)

- Always triggered by software events … until now
Tortola: Symbiotic Optimization

- Enable HW/SW Communication
Simulation Methodology

- SimpleScalar 4.0 for x86
- Wattch 1.02 power extensions
- Pin dynamic instrumentation system (x86/Linux version)
Tortola Applications

• Combine global program information with run-time feedback
  – System-specific power usage
  – Application-specific heat anomalies
  – Workload/input specific performance optimization

• Reduce hardware complexity
  – No more backwards compatibility warts
  – Fix bugs after shipment
  – Reduce time to market for new architectures

• One such application: The di/dt problem
The Di/dt Problem

• Low-power techniques have a negative side effect: current variation
• Voltage stability is important for reliability, performance

• Dips (undershoots) in supply voltage – can cause incorrect values to be calculated or stored
• Spikes (overshoots) in supply voltage – can cause reliability problems
Detecting Imminent Emergencies

Operating Voltage Range

- Phantom firing - increases current (at the expense of power)
- Resource throttling - reduces current (at the expense of performance)
A Di/dt Stressmark

BEGIN_LOOP:
...
ldt $f1, ($4)
divt $f1, $f2, $f3
divt $f3, $f2, $f3
stt $f3, 8($4)
ldq $7, 8($4)
cmovne $31, $7, $3
stq $3, $(4)
stq $3, $(4)
stq $3, $(4)
...
JUMP BEGIN_LOOP

But...Actuator engages every loop iteration degrading performance

Why not correct the problem in the code?
Proposed Solution

- Leverage our additional software layer to supplement existing solutions
- Microarchitecture provides feedback to our software-based virtual layer
Loop Unrolling & SW Pipelining

Problematic loop:

Unrolled loop:

Software pipelining smoothes profile

Loop unrolling disrupts resonance pulse

Current

Iteration=1

Iteration=2

Iteration=3

Current
Unrolling the $\text{Di/dt}$ Stressmark

Before Loop Unrolling

After Loop Unrolling
Summary

• Symbiotic program optimization is a powerful approach
• The di/dt problem – well suited for a symbiotic solution
• The Tortola design can also target power reduction, temperature reduction, reliability, etc.

http://www.tortolaproject.com/