Evaluating Compiler Support for Complexity Effective Network Processing

Pradeep Rao and S.K. Nandy
Computer Aided Design Laboratory.
SERC, Indian Institute of Science.
pradeep, nandy@cadl.iisc.ernet.in
http://www.serc.iisc.ernet.in/cadl/
Outline

• Why network processors (NP) ?
  – Why complexity effective NPs ?
• NP design issues
• Statically scheduled processors for NPs
  – Compiler optimizations
    • Classical
    • Superblock
    • Hyperblock
  – Performance Data
Network Processors

• Why do we need network processors?
  – Significant time spent in protocol stack
  – Increasing data rates
    • Increased performance requirements
  – New protocols and services
    • Software based functionality
    • Flexible (vs. ASIC)
    • Faster time to market

• Players
  – Cisco, Intel IXP, IBM PowerNP, Motorola (C-Port) C5, Broadcom, ClearWater ...
Complexity Effective NPs

• Complexity-Effective hardware
  – Low design, verification and testing times
  – Impacts time to market
  – Low power
    • Fixed power budgets for line cards
    • Network enabled mobile devices
  – Performance goals met?

• Performance
  – Exploit parallelism
  – Push clock frequencies
NP Design Issues

- System Design: Organization of memory, interconnection, processing element (PE) and its local memory …

- Inadequate performance data for the design of future network processors
Static Scheduling for NPs

• Keep hardware simple by offloading complexity onto the compiler
• The compiler has a ‘global’ view of the program
• Performance data for
  – In-order superscalar (IOS)
  – VLIW
Methodology

• IMPACT Toolset (UIUC)
• Architectures
  – In-order Superscalar
  – VLIW
• Compiler optimizations
  – Classical
  – Superblock
  – Hyperblock

• Applications
  – Checksum computation: \textit{crc}
  – Deficient round robin scheduling: \textit{drr}
  – Shortest path computation: \textit{dijkstra}
  – Diffie Hellman public key encryption/decryption: \textit{dh}
  – Reed Solomon codec: \textit{reed\_enc}, \textit{reed\_dec}
The Superblock

- Essentially a trace with single entry multiple exits
- Reduces bookkeeping required to support side entrances
- Code motion with compiler controlled speculation
- General speculation model for minimal hardware support.
The Hyperblock

- Adds predicated execution for superbblocks
Application Characteristics

- **Op-code Frequencies**
  - 40% integer operations
    - Addition and shifts account for > 80% ops
  - SB optimizations do not change the op freq.
    - No additional stress on resources
  - HB optimizations reduce conditional branches by if-conversion.
    - Predicate instructions account for 0-37%
Application Characteristics…

- Branch Statistics
  - Avg. branch prediction accuracy: 92.32%, with < 9% deviation
  - Branch prediction accuracy for SB and HB are higher
Application Characteristics...

- **Block Size**
  - Indicative of potential parallelism
  - BB Avg: 5 instructions
  - SB/HB Avg: 13 instructions
Application Characteristics...

- **Cache Performance**
  - Effect of SB/HB on cache performance
  - D$ unaffected
  - I$, for equivalent cache sizes the miss rate increases by 40%
Architectural Evaluation…

- Speedup plots with perfect caches for VLIW
- Up to 2.4x speedup with SB/HB optimization
- Predication overhead at low issue widths
- Performance gain from HB (over SB) at high issue < 8%
- Leveling indicates decrease in processor utilization
Architectural Evaluation…

- Effect of real cache
  - Greater impact on VLIW than IOS
  - However, the performance benefit of IOS over VLIW is less than 1.8%, suggesting VLIW for complexity effective designs
  - Average network rates of 6.6Gbps @ 500MHz for *drr*

<table>
<thead>
<tr>
<th></th>
<th>IOS</th>
<th>VLIW</th>
</tr>
</thead>
<tbody>
<tr>
<td>BB</td>
<td>1.06%</td>
<td>1.08%</td>
</tr>
<tr>
<td>SB</td>
<td>5.6%</td>
<td>6.8%</td>
</tr>
<tr>
<td>HB</td>
<td>5.6%</td>
<td>7.4%</td>
</tr>
</tbody>
</table>
Frequency Effects

- Increase in memory/FU latency (empirical)
- Increase in performance not commensurate with frequency increase
  - Performance improvement with doubled frequency (B-M1) is < 37%, (M1-M2) < 31%
- Need for efficient latency hiding techniques
  - (SMT, TCP) ?
Conclusions

- This study provides performance data for statically scheduled processors, for networking applications
- Operation frequencies differ from SPEC and Media applications
  - Organization of FU’s
- High static branch prediction rates
  - Make static scheduling attractive for networking applications
- Speedup due to SB and HB optimizations can be as high as 2.4
Conclusions…

- HB optimizations improve performance by < 8%
  - The additional complexity might not be justified
- The performance advantage of an IOS over VLIW is less than 1.8%
  - VLIW being CE might be more attractive
- Simulation results show average network rates of 6.6Gbps for drr, at 500MHz for 8-issue VLIW with SB optimization
- Need to exploit packet level parallelism
Thank You