Intel Itanium Line Processor Efforts

Xiaobin Li

PASCAL
EECS Dept. UC, Irvine
Outline

- Intel Itanium Line Roadmap
- IA-64 Architecture
- Itanium Processor Microarchitecture
- Case Study of Exploiting TLP at VLIW
<table>
<thead>
<tr>
<th>Year</th>
<th>Original</th>
<th>Now</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001</td>
<td>Merced (Itanium)</td>
<td>2001 Merced (Itanium)</td>
</tr>
<tr>
<td>Mid-2002</td>
<td>McKinley (Itanium2)</td>
<td>Mid-2002 McKinley (Itanium2)</td>
</tr>
<tr>
<td>Mid-2003</td>
<td>Madison</td>
<td>Mid-2003 Madison (1.5GHz, 130nm)</td>
</tr>
<tr>
<td>2004</td>
<td>Montecito</td>
<td>2004 Madison9M (9MB of L3 Cache)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2005 Montecito (Dual-core, 90nm)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delayed!</td>
</tr>
</tbody>
</table>
IA-64 Architecture

EPIC (Explicitly Parallel Instruction Computing)
- An evolution of VLIW
- Three components:
  1. Explicit parallelism
  2. Features to enhance ILP
  3. Resources for parallel execution
IA-64 Bundle and Instruction Formats

Bundle Formats

128 bits

Instruction 2  Instruction 1  Instruction 0  Template

41 bits  41 bits  41 bits  5 bits

Instruction Formats

<table>
<thead>
<tr>
<th>op</th>
<th>Reg 1</th>
<th>Reg 2</th>
<th>Reg 3</th>
<th>Predicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 bits</td>
<td>7 bits</td>
<td>7 bits</td>
<td>7 bits</td>
<td>6 bits</td>
</tr>
</tbody>
</table>

UCI
University of California, Irvine
Predication

- Conditional execution of instructions
- Removes branches, results in larger basic blocks
- Reduces associated mispredict penalties

\[
\text{if } (r1 == r2) \\
\text{r9 = r10 - r11;} \\
\text{else} \\
\text{r5 = r6 + r7;}
\]

Branch prediction with speculative execution

- (if r1 == r2), branch predicted
- Speculative instructions executed
- Branch resolved (misprediction)
- Speculation instruction squashed
- Correct instruction executed

Predication code

\[
\text{cmp.eq p1, p2 = r1, r2;;} \\
(p1) \text{sub r9 = r10, r11} \\
(p2) \text{add r5 = r6, r7}
\]

Predication execution

- Compare (r1 == r2)
- Both paths executed simultaneously
Control Speculation

- Code motion: moving a load (and its uses) past a branch
- Hardware exception deferral

**Traditional scheduling**

```
instr A
instr B
...
br
-----------------
ld8 r1 = [r2]
use r1
...
```

**IA-64 scheduling**

```
ld8.s r1 = [r2]
use r1

instr A
instr B
...
br
-----------------
chk.s
...
```
Data Speculation

- Code motion: moving a load advance to its preceding stores, even if the load and stores may alias
- Hardware support: advanced load address table (ALAT)

Conservative Scheduling

- instr A
- instr B
- ...
- store
- ld8 r1 = [r2]
- use r1

IA-64 Scheduling

- ld8.a r1 = [r2]
- use r1
- ...
- instr A
- instr B
- ...
- store
- chk.a

ALAT Organization

<table>
<thead>
<tr>
<th>dest reg#</th>
<th>addr</th>
<th>size</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>[r2]</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

stores CAM on addr field
Software pipelining for loop-level parallelism: prologue, kernel, epilogue

Reduced software pipelining overheads:

* **Register rotation** (register rename base (RRB) register) => provides renaming of intermediate results from previous iterations, eliminating register copy operations in the loop body;
* **Stage predicate** => control execution of instruction in the stage;
* **Loop count** (LC) => determines whether the kernel of the loop will be executed or not;
* **Epilogue count** (EC) => schedule the number of the loop iterations required to drain the software pipeline
Register Rotation

<table>
<thead>
<tr>
<th>Register Set</th>
<th>Static</th>
<th>Rotating</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Registers</td>
<td>0-31</td>
<td>32-127</td>
</tr>
<tr>
<td>FP Registers</td>
<td>0-31</td>
<td>32-127</td>
</tr>
<tr>
<td>Predicate Registers</td>
<td>0-15</td>
<td>16-63</td>
</tr>
</tbody>
</table>

In the range of rotating: Reg# = RRB + Reg#

Loop: ld4 r34 = [r10], 4
    nop
    st4 [r11] = r32, 4
    swp_branch Loop;;
Itanium Processor Microarchitecture

From: “Intel(R) Itanium(TM) Processor Microarchitecture Overview”
The Itanium Processor Core Pipeline consists of 10 stages:

**Front End**
- Pre-fetch/Fetch of up to 6 instructions/cycle
- Hierarchy of branch predictors
- Decoupling buffer

**Execution**
- 4 single cycle ALUs, 2 ld/str
- Advanced load control
- Predicate delivery & branch
- Nat/Exception/Retirement

**Instruction Delivery**
- Dispersal of up to 6 instructions on 9 ports
- Reg. remapping
- Reg. stack engine

**Operand Delivery**
- Reg read + Bypasses
- Register scoreboard
- Predicated dependencies
Thread-Level Parallelism

- ILP: VLIW, predication, control speculation, data speculation, software pipelining
- TLP: CMP, multi-threading, SMT

<table>
<thead>
<tr>
<th></th>
<th>Superscalar</th>
<th>VLIW/EPIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>MT</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>SMT</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>
Dual-Core VLIW Processor Case Study
128-bit VLIW Instruction Packets

- 32 bits | 32 bits | 32 bits | 32 bits
- \( l_0 \) | \( l_1 \) | \( l_2 \) | \( l_3 \)

- **Header**
- 00 = 1 Instruction
- 01 = 2 Instructions
- 10 = 3 Instructions
- 11 = 4 Instructions

- **Opcode** | \( rd \) | \( rs3 \) | \( rs1 \) | \( rs2 \)
- 4 bits | 7 bits | 7 bits | 7 bits | 7 bits

- OpCodes available for 4-register specifier instructions
- "rd" field used for more opcode bits for other instructions
**Chip Block Diagram**

- North UPA, 4KB I/O Buffer
- Memory Controller Rambus DRAM
- PCI
- 16 KB Instruction Cache
- Graphics Preprocessor (GPP)
- 16 KB Instruction Cache
- Switch
- 16 KB Shared Data Cache
- South UPA

- 250 MHz
- 400 MHz
- 66 MHz
- Vertical multithreading
- CMP
- Speculative multithreading

**Vertical multithreading**

Multithread share the resources in a processor unit on a cycle-by-cycle basis.

**CMP**

Threads are scheduled to different processor units.
Conclusion

- Intel Itanium Line Roadmap
- IA-64 Architecture
- Itanium Processor Microarchitecture
- Case Study: MAJC 5200 Processor