Performance Evaluation of the Superscalar Speculative Execution

HaL SPARC64 Processor

by

Andrew Essen and Stephen Goldstein

ae@hal.com  steven@hal.com

Outline of Talk

⇒ Machine description
⇒ Performance Prediction Methods
  - Options Evaluated
  - Validation of Predictions
  - Measuring Superscalar Execution
  - “What-if” Studies of Performance
  - A Brief Survey of Superscalar Performance
  - Conclusions and Perspective
Machine description

64 bit SPARC V9 CPU

Instruction Issue
- 4way superscalar
- Speculative

Inst. Execution
- Out of Order
- Precise Traps

MMU
MSU queue
Xlate
Look-aside Tables

Memory cards

MCM

See HaL track at COMPCON95 and papers in ISSCC95 for details

Processor Details

BHT
BRU
I-zero Cache
I-one cache
MMU
MSU queue
Xlate
Look-aside Tables

Memory cards

MCM
Performance Estimates

Elapsed Time = CPI x Cycle time x Inst. count

- **Cycle per Inst**: Predicted by the Timer software model
  Input from HaLsim traces
  Statistical sampling techniques were used

- **Cycle time**: Determined by technology and logic design

- **Inst. count**: Measured with applications run on HaLsim

*Sometimes it's more convenient to use Instructions per Cycle: IPC = 1 / CPI*

---

Simulation Process

1. **Source** → **Compile / Make** → **Application / Benchmark**
2. **SPARC V9**
3. **HaL OS (64 bit)** → **HaLsim** → **Trace Pre-process** → **"TIMER"**
4. **Unix** → **Work station**

---

Aug. 95
Outline of Talk

- Machine description
- Performance Prediction Methods

⇒ Options Evaluated

⇒ Validation of Predictions
  - Measuring Superscalar Execution
  - "What-if" Studies of Performance
  - A Brief Survey of Superscalar Performance
  - Conclusions and Perspective

Timer is a Flexible Tool

200+ Independent Options Handled as Switches in Timer

⇒ Functional unit Latency, Number of, and Queue depth
⇒ Cache sizes, Organizations and Levels
⇒ Branch prediction, Fetch policies
⇒ Issue rules, Peak issue rate, and Instruction window size
⇒ Translation and Memory latency and bandwidth (incl. TLB miss modeling)
⇒ Magic Options (e.g. perfect caches, perfect branch predict, . . .)
⇒ . . .

Options Not Easily Handled in Timer

⇒ Speculative side effects
⇒ I/O -- contention from DMA
⇒ Major changes in pipeline stages
Timer Predictions Track Logic

022.Li

Logic Model
Timer Prediction

Test Strip 2

IPC

Instructions

0 1.0E+05 2.0E+05 3.0E+05 4.0E+05

This strip has a 3% error ratio = (Measured cycles / Predicted cycles)

Timer Validation

Distribution of Error Ratios from 29 test strips

Count

0.95 1.00 1.05 1.10 1.15 1.20

# Cycles Measured on Logic Model
# Cycles Predicted by Timer

Aug. 91
Outline of Talk

- Machine description
- Performance Prediction Methods
- Options Evaluated
- Validation of Predictions

⇒ Measuring Superscalar Execution
⇒ “What-if” Studies of Performance
- A Brief Survey of Superscalar Performance
- Conclusions and Perspective

Time Line, Cycle, and Issue Three Views -- Same Data

Time Line

Valid Inst. Issued

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>No I Fetch</td>
<td>Other</td>
<td>Branch Mispredict</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Cycles

... 17 18 19 20 ...

80 issue opportunities in 20 cycles

Summarized by Cycles

Cycles with "n" instructions issued
n = 0 to 4

Fraction of cycles

0.5

0.4

0.3

0.2

0.1

0

Number of Valid Inst. Issued

0 1 2 3 4

Summarized by Issue Opportunities

Instruction Issued

Other 10%

Branch Mispredict 32.5%

No I Fetch 20%

Instruction Issued 37.5%
### 3.1-13

**IPC Calculations**

**Formula:**

\[
IPC = \sum_{n=0}^{4} P(n\text{-issue cycles}) \times n
\]

**For this example:**

\[
IPC = 0.5(0) + 0.05(1) + 0.15(2) + 0.05(3) + 0.25(4)
\]

\[
IPC = 1.50
\]

**IPC = % inst. issued x opportunities/cycle**

**For this example:**

\[
IPC = 37.5\% \times 4
\]

\[
IPC = 1.50
\]

---

### 3.1-14

**Timer Output Shows Details**

- **Valid Inst. Issued:**
  - Number of valid instructions issued over cycles.
- **Cycles:**
  - Number of cycles for each instruction.
- **Teaching Example:**
  - Not real data.
- **Legend:**
  - I = issue
  - W = write results
  - L = load
  - C = complete
  - A = address generation
  - X = execute
  - T = waiting for source data
  - L = load

**Issued but not completed:**

- **Branch Mispredict:**
  - Branch prediction failure.

**Other:**

- **No I Fetch:**
  - No instruction fetch.

Aug. 95
SPEC92 Integer Benchmarks

6 benchmarks

- 008.espresso Logic Design Gens and optimizes programmable arrays
- 022.li Interpreters Lisp interpreter solves 9 queens problem
- 023.eqqtott Logic Design Translates logic to Boolean truth tables
- 026.compress Data Compress Compresses using Lempel Ziv coding
- 072.sc Spreadsheet Calculates budgets etc.
- 085.gcc Compiler Translates C source into Sun assembly

Typically each benchmark executes $10^9$ to $10^{10}$ instructions.

SPEC™ of Standard Performance Evaluation Corporation
**Hal SPARC64 Design Point**

- Dynamic stall: 6%
- Issue rules: 13%
- Branch mispredict: 25%
- No I Fetch: 16%
- Other (machine dep.): 10%

\[ \text{.30 issued/slot} \times 4 \text{ slot/cyc} = 1.2 \text{ IPC} \]

*Estimate for SPECint92*
- Avg. of 8 benchmarks
- Each benchmark equal weight

---

**"Back of the Envelope" Sanity Check**

\[ \text{IPC}^{-1} = \text{CPI} = .25 + 8 \times \text{MR(Br)} + 4 \times \text{MR(b)} + 28 \times \text{MR(L1)} \]

- 4 issue/cyc
- 8 cyc/miss
- 4 cyc/l-zero miss
- 28 cyc to DRAM
- .025 miss/inst
- .015 miss/inst
- .0056 miss/inst

This gives a rough estimate of performance

\[ \text{CPI} = .67 \text{ or } \text{IPC} = 1.5 \]

MR is Miss Rate expressed as Misses per Instruction Retired
**Perfect Branch Predict**

- Dynamic stall: 13%
- Issue rules: 17%
- No I Fetch: 22%
- Other (machine dep.): 11%
- Instruction issued: 37%

Branch mispredict = 0%

IPC = 1.47 or +22%

*Estimate for SPECint92
- Avg. of 6 benchmarks
- Each benchmark equal wt.*

---

**Perfect Caches**

- Dynamic stall: 5%
- Issue rules: 18%
- Branch mispredict: 25%
- Other (machine dep.): 11%
- Instruction issued: 38%

Branch mispredict = 3%

IPC = 1.52 or +27%

*Estimate for SPECint92
- Avg. of 6 benchmarks
- Each benchmark equal wt.*
All execution in one cycle

- Issue rules: 13%
- Branch mispredict: 14%
- No I Fetch: 21%
- Instruction issued: 43%
- Other: (machine dep.) 9%

Dynamic stall = 0%

IPC = 1.72 or +43%

Outline of Talk

- Machine description
- Performance Prediction Methods
- Options Evaluated
- Validation of Predictions
- Measuring Superscalar Execution
- "What-if" Studies of Performance

⇒ A Brief Survey of Superscalar Performance
⇒ Conclusions and Perspective
→ **SPECint92 and SPECint92 / MHz are good metrics**

→ **Broader comparisons allowed**
  - IPC's are seldom published
  - SPECint results are publicly available

→ **Directly comparing IPC's has “the MIPS problem”**
  - Different architectures make IPC comparisons meaningless across vendors
  - A machine running the same benchmark with different binaries can give misleading IPC results
Efficient Superscalar Operation is Difficult

Recent Performance Claims of 64 bit RISC Processors

<table>
<thead>
<tr>
<th>Model</th>
<th>SPECint92 @ MHz</th>
<th>SPECint / MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>UltraSPARC</td>
<td>240 @ 167</td>
<td>1.4</td>
</tr>
<tr>
<td>R10000</td>
<td>300+ @ 200</td>
<td>1.5+</td>
</tr>
<tr>
<td>SPARC 64</td>
<td>256 @ 154</td>
<td>1.7</td>
</tr>
<tr>
<td>620</td>
<td>225 @ 133</td>
<td>1.7</td>
</tr>
<tr>
<td>PA8000</td>
<td>360 @ ~200</td>
<td>1.8</td>
</tr>
</tbody>
</table>

SPECint92 / MHz is a Figure-of-Merit for Micro-Architectures, Most Vendors Claim to be Doing About Equally Well.

Vendor data from recent Microprocessor Reports

3.1-28

Conclusions

⇒ It's hard to sustain high IPC's
  - Simply increasing peak issue rates runs into Amdahl's law
  - Parallel out-of-order execution is not a panacea

⇒ Impacts of proposed features must be evaluated
  - Test cases should be realistic
  - Modeling can identify true bottlenecks and opportunities
  - Quick analyses plus detailed simulations make a powerful tool

⇒ Benchmark execution time is the correct metric
  - Focusing solely on IPC's is dangerous
We've come a long way from 1 SPECint92 machines
The SPEC92 "reference machine" is
- 5 MHz, single issue, CISC architecture
- circa 1978
From 0.2 to 1.67 SPECint92/MHz
- Over a factor of 8 improvement
5 MHz to 154 MHz
- Over a factor of 30 improvement

These Improvements Multiply giving
250+ Times Faster Uniprocessors

Acknowledgment

THE AUTHORS WISH TO THANK and ACKNOWLEDGE
THE CONTRIBUTIONS OF:

- THE HAL ENGINEERING
- THE HAL TEST TEAMS
- THE HAL SOFTWARE OS AND TOOLS TEAMS
- THE HAL COMPILER DEVELOPMENT TEAM
- OTHER MEMBERS OF HAL ARCHITECTURE AND PERFORMANCE
  PARTICULARLY:
  VENKAT BUDUMA, MYKE SMITH, AND DR. WINFRIED WILCKE