Silicon Graphics TFP Micro-Supercomputer Chipset

Peter Yan-Tek Hsu

Silicon Graphics Computer Systems
Mountain View, California

TFP is a collaborative development of MIPS Technologies Inc., Toshiba Corporation and Weitek Corporation.

What's New?

Supercomputer performance at microprocessor prices
- 300 MIPS, 300 MFLOPS peak
- 1.2 GB/s load/store bandwidth
- 8"×7" footprint, < 100 W per processor chipset
Outline

Overview
Integer Pipeline
Superscalar Instruction Dispatching
Branch Prediction
Floating-Point Unit
Split-Level Cache
Interleaved Access Patterns
Chip Facts
Summary

Product Goals

Performance of low-end vector supercomputers
- Two double-precision multiply-adds per cycle (4 FLOPS)
- Two 64-bit load/store operations per cycle
- Multiprocessor capable

Microprocessor-like cost structure
- Few high-integration CMOS chips
- Extensive caching using commodity SRAMs
- TTL signals, PGA packages

Compatibility with SGI product line
- Superscalar implementation of MIPS instruction set
- Scheduled code runs well on R4000 microprocessor
Innovation: Split Level Cache

Architect’s dilemma:
- Integer/address computations need fast caches; small ok
- Floating-point needs large, high bandwidth caches; slower ok
- Both must perform well to satisfy real workloads

Solution:
- Single-cycle on-chip cache for integer/address data
- Multi-cycle pipelined off-chip cache for FP data
- Decoupled FPU pipeline hides off-chip cache latency
- Hardware maintains coherence between on/off chip cache
### Integer Pipeline

<table>
<thead>
<tr>
<th>Fetch</th>
<th>Decode</th>
<th>Address</th>
<th>Execute</th>
<th>Writeback</th>
</tr>
</thead>
<tbody>
<tr>
<td>instruction</td>
<td>decode</td>
<td>generate</td>
<td>ALU operation</td>
<td>register</td>
</tr>
<tr>
<td>fetch</td>
<td>scoreboard</td>
<td>load/store</td>
<td>data cache access</td>
<td>file write</td>
</tr>
<tr>
<td>alignment</td>
<td>register file</td>
<td>address</td>
<td>branch resolution</td>
<td></td>
</tr>
<tr>
<td>predecode</td>
<td>read</td>
<td></td>
<td>TLB lookup</td>
<td></td>
</tr>
<tr>
<td>branch</td>
<td></td>
<td></td>
<td>exception detection</td>
<td></td>
</tr>
<tr>
<td>prediction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Low-latency RISC pipeline
- No load delay slot
- Data forwarding for dependent load after store in same cycle
- Branch and delay slot can execute together

### Instruction Fetch

In insensitive to code alignment
- Executes 0 to 4 instructions each cycle
- Always presents next 4 instructions to decoder

Decoupled instruction prefetch pipeline
- Dispatch decision takes full cycle
- Critical path controls few muxes, not RAM address

Avoid draining pipeline by branch prediction
- “One-bit” prediction scheme
- Implemented as extension of instruction cache RAM
- Gain accuracy by large size
**Instruction Alignment Buffer**

```
buffer  
M N O P  
I J K L  

on-deck  
M N O P  
I J K L  
M N O P  
Q R S T  

dispatch  
E F G H  
--- G H  
--- ---  
--- ---  
I F G H  
I J K H  

execution  
A B  
E C D  
E C D  
A B  
```

**Branch Prediction**

```
<table>
<thead>
<tr>
<th>v</th>
<th>target</th>
<th>src</th>
<th>dst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>index</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>***</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>***</td>
<td>branch</td>
<td>delay</td>
<td>***</td>
</tr>
</tbody>
</table>
```

```

BCACHE  

fpc  

+4  

ICACHE  

valid mask  

0 1 2 3  

*** target ***  

... ... ... ...  

... ... ... ...  

... ... ... ...  

... ... ... ...  

... ... ... ...  
```
Misprediction Penalty

Program

<table>
<thead>
<tr>
<th></th>
<th>F</th>
<th>D</th>
<th>A</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>br</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>dly</td>
<td>br</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>x</td>
<td>dly</td>
<td>br</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>y</td>
<td>x</td>
<td>dly</td>
<td>br</td>
<td>...</td>
</tr>
<tr>
<td>z</td>
<td>y</td>
<td>x</td>
<td>dly</td>
<td>...</td>
</tr>
<tr>
<td>A</td>
<td>z</td>
<td>y</td>
<td>x</td>
<td>...</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>z</td>
<td>y</td>
<td>...</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>A</td>
<td>z</td>
<td>...</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>B</td>
<td>A</td>
<td>...</td>
</tr>
</tbody>
</table>

- Begin fetching target
- Misprediction detected
- Begin fetching good target
- 3 cycle penalty
- Resume execution

- Branch prediction associated with delay slot
- Unified E-stage logic for misprediction, exception, cache miss

Decoupled Floating-Point Unit

External Cache Pipeline (5 stages)

Crossbar

Integer Pipeline

Address Bellow

Cache Tags

Cache Data (Even Bank)

Cache Tags

Cache Data (Odd Bank)

FPU

Load Data

Floating-point Instructions
Unbuffered Interleaved Cache

Good alignment
100% efficiency

- Crossbar
- Odd address
- Even address

Bad alignment
50% efficiency

Effect of Address Bellow

Good alignment
100% efficiency

Bad alignment
100% asymptotic efficiency

Address bellow resolves bank conflicts
- Compiler worry: Uniform distribution of even/odd references
- Hardware takes care of alignment problems
- Effective for subroutine array parameters
Split Level Cache Coherence

```c
struct {
    char Ch[4]
    float Flt
    int Ival
} s

if ( s.Ch[2] == ... ) {
    s.Flt = ... 
    s.Ival = ... = s.Ival
}
```

<table>
<thead>
<tr>
<th>Dcache Action</th>
<th>Hit</th>
<th>Miss</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int St</td>
<td>merge data fetch &amp; retry (set invalid)</td>
<td></td>
</tr>
<tr>
<td>FP St</td>
<td>set invalid</td>
<td>nop</td>
</tr>
</tbody>
</table>

- Valid bit per 32-bit word in cache line
- Efficiently supports mixed integer/FP structures

### IU and FPU Facts

- **Process Technology**: 0.7μ CMOS (0.5μ L-eff), 3-Metal
- **Cycle Time**: 75 MHz
- **Die Size**: 17.3 mm × 17.2 mm
- **Transistor Count**: IU 2.6M, FPU 830K
- **Package**: 591 pin CPGA, 382 signals
- **Voltage**: 3.3 V
- **Power Dissipation**: < 15 W each @ 75 MHz
- **Interface**: LVTTL
IU and FPU Facts (cont)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue Rate</td>
<td>Four instructions per cycle</td>
</tr>
<tr>
<td>Address Space</td>
<td>$2^{48}$ virtual, $2^{40}$ physical</td>
</tr>
<tr>
<td>I-Cache, D-cache</td>
<td>16 KB each, direct mapped</td>
</tr>
<tr>
<td></td>
<td>Single cycle latency</td>
</tr>
<tr>
<td>External Cache</td>
<td>1..16 MB, 12 ns Sync. SRAM</td>
</tr>
<tr>
<td></td>
<td>Five cycle latency, pipelined</td>
</tr>
<tr>
<td>TLB</td>
<td>384 entries, 3-way set assoc.</td>
</tr>
<tr>
<td>Floating-Point Unit</td>
<td>Two 4-stage mpy-add pipelines</td>
</tr>
<tr>
<td></td>
<td>Divide latency: 20 cycles</td>
</tr>
</tbody>
</table>

Summary

Supercomputer performance at microprocessor prices
- 300 MIPS, 300 MFLOPS peak
- Handful of CMOS chips

Advanced yet low-cost memory hierarchy
- Split level cache makes optimal use of on-, off-chip RAM
- Address bellow greatly improves interleaved cache efficiency

4-way superscalar
- Delivers bandwidth of vector architecture
- Solves binary compatibility problem