A Multithreading Embedded Architecture
Lan Dong, Xiufeng Sui
Computer Engineering Department
Beijing Jiaotong University
Shang yuan village No.3, Beijing
China

Abstract: More embedded microprocessors are emerging and more multithreaded tasks need to be executed simultaneously on embedded device. This paper proposes a multithreaded architecture for embedded processor. It improves the system performance in a large degree meanwhile considering reducing energy dissipation and improving the cache utilization in limited size.

Key-Words: multithreading technology, thread, cache, fetch policy, decoder, hybrid partition

1 Introduction
Embedded computers have been grown as the fastest part in the computer market nowadays. They are spread over from cell phones, video games to most family machines. Embedded systems often deal with event under the real-time constraints. So the performance of the embedded system must be considered with these changes.

In embedded world the power and the size are the important constrains for embedded processor [1, 2]. The techniques to reduce the system energy dissipation and improve the resource utilization must be considered with these variations.

A multithreading embedded architecture is proposed in this paper in this background. In section 2, the architecture of multithreading embedded processor is described. Section 3 shows the experiment results of this architecture. In Section 4, a conclusion is made.

2 Multitreading Embedded Architecture

2.1 Instruction Set
ARM processor has occupied the majority of the embedded market distribution. To make the design practical, the architecture proposed in this paper supports ARM instruction set.

Supported by the Science and Technique Foundation of Beijing Jiaotong University 2007RC086

2.2 Architecture
The basis for this effort on introducing the multithreading embedded architecture is an ARM-derived instruction set superscalar processor [3, 4]. Only necessary changes to enable simultaneous multithreading are added to the basic superscalar architecture. The necessary to support the simultaneous multithreading are mainly: program counter, return stack for each thread, and threaded register bank, etc.

In embedded system, cache size is usually limited. It is important to redesign the cache partition mechanism to improve the hit rate in multi-tasks environment on fixed cache size. The hybrid cache partition method in this architecture proposed in the previous work [5] combines shared with distributed partition method. It is feasible for embedded systems in multithreading circumstance.

Fetch unit in this architecture supports many fetch policies [6, 7], such as Icount, Brcount, Misscount, which have a different effect on IPC corresponding to different features of programs.

Branch prediction estimator gathered with the thread switch mechanism [8] is adopted in this architecture. It can reduce the wrong instructions and switch frequencies.

3 Experiment
Simulator used here was developed based on simple-arm [9, 10]. A multithreading extension was given to it in my previous work. In this paper, several enhancement architectures described above are added in the new simulator. The experiment configuration is as table 1.
**Table 1 system configuration**

<table>
<thead>
<tr>
<th></th>
<th>L1 Instr. Cache</th>
<th>L1 data Cache</th>
<th>L2 instr. Cache</th>
<th>Function unit</th>
<th>Itlb</th>
<th>dtlb</th>
<th>memory</th>
<th>pipeline</th>
<th>predictor estimator</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.size</td>
<td>32KB</td>
<td>64KB</td>
<td>2MB</td>
<td>8</td>
<td></td>
<td>512KB</td>
<td>256KB</td>
<td></td>
<td>Gshare, JRS estimator</td>
</tr>
<tr>
<td>b.associative</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>ialu=8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c.replacement</td>
<td>LRU</td>
<td>LRU</td>
<td>distributed:shared</td>
<td>imult=2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d.block size</td>
<td>32B</td>
<td>32B</td>
<td>64B</td>
<td>memport=4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e.hit latency</td>
<td>1cycle</td>
<td>1cycle</td>
<td>6cycle</td>
<td>fpalu=8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fpmult=2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>divmult=2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Fig.1 IPC with different thread number**

From Fig.1, it can be concluded that the multithreading embedded architecture can run well in the multi-tasks environment with embedded benchmarks. IPC is obviously improved when the thread number is increased from 1 to 6.

**5 Conclusion**

In this paper, we design a multithreading embedded architecture. Different to traditional multithread architecture and embedded processors, it exploits several technologies to enhance system performance and improve the utilization of the resource.

**References:**


