A Verification Environment for a SCMP Architecture

Wenbin Yao, Nianmin Yao, Shaobin Cai, and Jun Ni
College of Computer Science and Technology, Harbin Engineering University, Harbin, China
{yaowenbin, yaonianmin, caishaobin, nijun}@hrbeu.edu.cn;

Abstract

The computer architecture of Single-chip multiprocessor (SCMP) is one of important research topics in developing the next-generation of computer hardware. A verification environment in the SCMP architecture, base of RISC microprocessor, acts as a functional verification simulator that elaborates its functions. This paper reports a simulation of the operational behavior in terms of function units. The simulation was in the mode of cycle-by-cycle when programs execute. The results of the SCMP simulation show that the simulation and its implementation can be used to effectively study the feasibility and applicability of the SCMP architecture.

1. Introduction

With the rapid development of semiconductor technology, one can oversee that within the next 5 years it is realistic to integrate a billion transistors on a silicon chip based on Moore’s law. Such integration promises to efficiently utilize the transistors, if a software parallelism is used together. The traditional Single-chip multiprocessor (SCMP) still has its merit of simple design process, compared with the other complex architectures [1-4]. The SCMP implements multithreads by employing multiple CPU cores into a single chip.

The CMP architecture is a promising technology, besides of system-on-chip technology [5]. In the system implementation, the CMP generally exploits very simple control logic, compared with other technologies, such as superscalar [6]. Its internal units can even be reused based on the existed function units. Speaking to the CMP’s performance improvement, it yields a speedup on data transaction among processors on running parallel programs, especially when the threads between processors communicate frequently. The CMP is an ideal architecture for future CPU design. However, due to the architectural complexity, the development of such system is still a challenge task, especially in verifying its functionality and performance.

The paper presents a simulation used for the verification of CMP architecture design. It is an execution-driven simulation of the memory system [7], which combines with a static analysis of CPU performance to obtain a reasonably accurate performance model for a CMP architectures. The simulation of the memory system and the processor pipeline provides an evaluation that refines an accurate cycle-by-cycle mode.

The rest of the paper is organized as follows. Section 2 gives the system implementation of the verification environment. Section 3 proposes sample CMP architecture (S-CMP) to describe and evaluate the performance of the verification environment. Section 4 presents and analyzes the simulation results, followed by a conclusion and future works.

2. System Implementation of the Verification Environment

Designing a CMP is a complex process, demanding on many reliable process segments to verify the correction of function implementation through out the whole design and implementation procedure. To achieve this, we developed a verification environment for the CMP architecture. The basic prototype system to achieve our tangible goal is composed of the following procedures

1. Provide an environment to verify the function and expandability of CMP;
2. Offer a simulation environment for the optimization of software compiler;
3. Assess the performance of current proposed CMP architecture, and thus refine the further design.

The proposed verification environment is an integrated system that provides the functions of both the architecture development and system verification. The system consists of two parts: development and verification studios,
respectively. The logic structure and flowchart are illustrated in Fig.1.

The verification environment implemented in C++ language has the capability of defining the structure of CMP architecture and further verifying its characters. It is a function verification model, which can exactly simulate the operational behavior during the execution of CMP microprocessor in the mode of cycle-by-cycle. A snapshot of the execution studio in the verification environment is shown in Fig.2. Two sub-windows in the lower middle of the main framework display the operational information of independent running CPU cores, respectively. The information includes the content of different levels of caches, values of different register files, etc. The studio provides very useful information about how to analyze and debug the execution status of the processors.

Figure 2. Snapshot of the Verification Environment

Based on construction consideration, the function units in the CMP architecture designed on the verification environment can be customized according to the users' demands. For example, it can adapt parameter inputs for various applications. The configurable units include the number of and types of the CPU cores, the cache level capacities (numbers and the levels). The CMP can own a single shared cache memory, two shared caches, or even none shared caches, respectively. Users of the environment can also modify the implementation of these configurable units at ease.

Beside these configurable units, the environment also provides a standard interface which can be used to insert user-defined components to the current CMP architecture.

3. Sample CMP

3.1 System Overview of S-CMP

A sample CMP architecture (S-CMP) is designed to testify the environment. The S-CMP model architecture depicted [8]. The experimental S-CMP architecture coupling multi-processors, integrates two identical RISC microprocessors into a chip. Its logic overview is shown in Fig.3. The S-CMP operates functions described as follows:

1. Two identical 32-bit RISC-based CPU cores.
3. A centralized bus arbitration mechanism that manages the control privilege of the shared bus.
4. Multi-processor control unit (MCU), which processes external interrupt distribution and inter-processor interrupts. It provides hardware support for synchronization of the processors.
5. Main memory interface (MIU), which handles all the interfacing transactions from/to S-CMP including main memory accesses and external snoop processing.
6. An I/O interface unit.

Figure 3. Logical overview of S-CMP architecture

3.2 Structure of the S-CMP

The main functions in S-CMP are described as follows:
3.2.1 CPU cores. The S-CMP has a legacy RISC-based microprocessor (Thump-107), which is developed at Tsinghua University. The processor has a 4k bytes instruction cache, a 4k bytes data cache, and a 7-stage pipeline. In the S-CMP, two special instructions are supplemented for implementing synchronization primitives required for standard CMPs.

3.2.2. Cache hierarchy. S-CMP adopts a two-level cache hierarchy to reduce the communication frequency between identical processors. In the implementation strategy, each processor has a pair of primary instruction and data caches, while all processors share a single, large on-chip secondary cache. The secondary cache is a unified cache whose capacity is 1M-bytes. The configuration is shown in Table 1.

3.2.3. Bus arbitration mechanism. Data exchange between the processors is carried out via a write-through/update bus under the control of the centralized bus arbitration mechanism. When one processor modifies data shared by the other processor, a writing broadcast over the bus updates the copies of the other processor. For a permanent machine, the state is written back to the secondary cache.

4. Simulation Results

The improvement of performance can be evaluated coarsely through running programs. To testify the functional features and the performance of the S-CMP architecture, we designed several programs that runs on a single-processor and the S-CMP respectively. The results of the consumed cycles for the four sample programs are listed in the Table 2.

The initial state of the S-CMP is just “boot-up” before the execution of sample toy programs. That means both levels of caches are empty. In such a mode, the programs must be filled into cache sequentially via the shared bus under the control of the centralized bus arbitration mechanism. Such a waste of cycles used for filling up the empty caches may be released on the conditions of the execution of large and long programs. The speedup of performance is not very high. That is due to the fact the size of the sample programs still stays in the toy level. With the increase of program size and the decrease of communications between threads of processors, the value of speedup raise significantly. From the results, we conclude that the centralized bus mechanism is the major bottleneck of blocking the improvement of performance. Therefore, we further refine the bus mechanism by expending the number of read/write ports, and at the same time add the bypass paths to the caches. The succeeding coarse testing results show that the refined S-CMP improve the average performance about a factor of 1.0~1.2, compared with the previous setting of S-CMP. The results of comparisons between different running modes (including CMP mode and sequent mode) are shown in Fig.4.

5. Conclusion and future works

The paper presented a verification simulation environment for verifying the functionality and applicability of the CMP architecture and implementation. The environment simulates the operational behaviors of function units accurately in the mode of cycle-by-cycle; thus provides a fundamental studio to further study the CMP architectures.

The verification environment that has a flexible organization to add more processors and other controlling structures to the existed CMP architecture is designed to be configurable to adapt more complex architectures. The further works include the performance improvements on the shared bus structure, the optimization of software compilers, mechanisms for fault tolerance for reliability, and system security.

6. Acknowledgement

This work is sponsored by the National Natural Science Foundation of China (No: 60403026 and No: 60503055) and the Basic Research Foundation of Harbin Engineering University (No: HEUFT05012 and No: HEUFT05011).

7. References


Table 1. Cache configuration of S-CMP

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>L1 cache</th>
<th>L2 cache</th>
<th>Main memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>Separate I SRAM and D SRAM</td>
<td>Shared, on-chip SRAM</td>
<td>Off-chip DRAM</td>
</tr>
<tr>
<td></td>
<td>cache, Pairs for each CPU</td>
<td>cache</td>
<td></td>
</tr>
<tr>
<td></td>
<td>core</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capacity</td>
<td>4K bytes each</td>
<td>1M bytes</td>
<td>128M bytes</td>
</tr>
<tr>
<td>Bus width</td>
<td>32-bit bus connection to CPU</td>
<td>64-bit read bus+ 32-bit</td>
<td>32-bit bus</td>
</tr>
<tr>
<td></td>
<td>write bus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Access time</td>
<td>1 CPU cycle</td>
<td>10 CPU cycles</td>
<td>At least 100 cycles</td>
</tr>
<tr>
<td>Associativity</td>
<td>2 way</td>
<td>4 way</td>
<td>N/A</td>
</tr>
<tr>
<td>Line size</td>
<td>32 bytes</td>
<td>64 bytes</td>
<td>4k bytes pages</td>
</tr>
<tr>
<td>Write policy</td>
<td>Write through, no allocate on</td>
<td>Write back, allocate on</td>
<td>Write back (virtual</td>
</tr>
<tr>
<td></td>
<td>write</td>
<td>writes</td>
<td>memory)</td>
</tr>
<tr>
<td>Inclusion</td>
<td>N/A</td>
<td>Inclusion enforced by L2 on</td>
<td>Includes all cached data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L1 caches</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Running results under the different running modes

<table>
<thead>
<tr>
<th>Cycles</th>
<th>Independent Execution</th>
<th>CMP Mode</th>
<th>Sequent Mode</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>50,102</td>
<td>162,328</td>
<td>191,728</td>
<td>1.18</td>
</tr>
<tr>
<td>P2</td>
<td>141,626</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>61,785</td>
<td>103,879</td>
<td>146,547</td>
<td>1.41</td>
</tr>
<tr>
<td>P4</td>
<td>84,762</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. Comparison between different modes on the aspect of running cycles