Golden-Finger and Back-Door: Two HW/SW Mechanisms for Accelerating Multicore Computer Systems

Slo-Li Chu *, Chih-Chieh Hsiao

Department of Information and Computer Engineering, Chung-Yuan Christian University

200, Chung Pei Rd., Chung Li, 32023 Taiwan.

Received 25 October 2011; received in revised form 27 November 2011; accepted 22 December 2011

Abstract

Continuously requirements of high-performance computing make the computer system adopt more processors within a system to improve the parallelism and throughput. Although multiple processing cores are implemented in a computer system, the complicated hardware communication mechanism between processors will decrease the performance of overall system. Besides, the unsuitable process scheduling mechanism of conventional operating system can not fully utilize the computation power of additional processors. Accordingly, this paper provides two mechanisms to overcome the above challenges by using hardware and software mechanisms, respectively. In software aspect, we propose a tool, called Golden-Finger, to dynamically adjust the scheduling policy of the process scheduler in Linux. This software mechanism can improve the performance of the specified process by occupying a processor solely. In hardware aspect, we design an effective hardware mechanism, called Back-Door, to communicate two independent processors which can not be operated together, such as the dual PowerPC 405 cores in the Xilinx ML310 system. The experimental results reveal that the two mechanisms can obtain significant performance enhancements.

Keywords: multicore, Xilinx ML310, hardware interprocessor communication.

1. Introduction

The continuously requirements of multimedia and streaming processing consumes more computing power of modern computer systems. These heavy loading jobs, such as MPEG4 encoding, music playing, movie playing, and 3D gaming, are usually executed simultaneously. The state-of-the-art computer architectures usually integrate multiple processing cores into a single chip to improve overall throughput. However, the inefficient hardware inter-processor communication mechanisms enlarge the communicating latency, and decrease the performance enhancements of additional cores. Besides, the general process scheduling mechanism for multicore system also cannot handle the performance requirements of a mission-critical task suddenly due to its fair round-robin based scheduling policy, such as in Linux.

The scheduling mechanisms of modern multicore operating systems generally arrange and schedule the processes in Round-Robin fashion according to the priorities of processes. For example, the scheduling mechanism in Linux kernel 2.6.11 will schedule the processes to all the processors with a load balance mechanism which migrate processes between processors,

* Corresponding author. E-mail address: slchu@cycu.edu.tw
Tel.: +886-3-2654721; Fax: +886-3-2654799

when the workload of processors are not balance. In such scheduling mechanism will postpone the execution of mission-critical applications, and delay their response time.

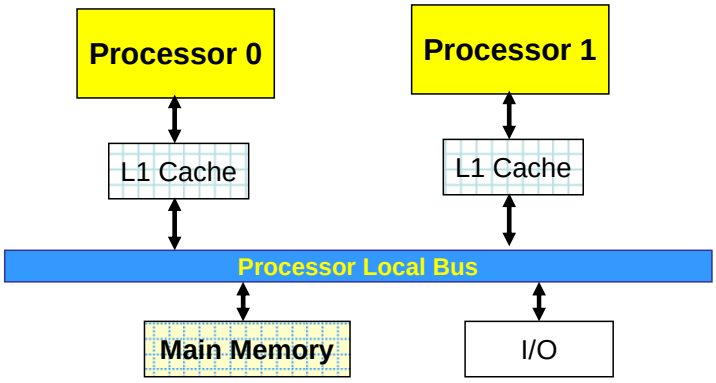

Fig. 1 Architecture of conventional dual-core processor

Besides, in the conventional multi-core architectures, such as Core 2 Duo (as Fig. 1) [8] and Athlon64 x2, there is no direct connection between processors. The only way to connect all processors is through processor local bus. The communication latency will be increased accordingly. It will waste a lot of time during sharing memory and system bus while processing streaming data. Even the conventional operating system, such as Linux, poorly supports to embedded multicore system, and often wastes too much time on scheduling. These problems can limit the overall performance of multicore system.

In this paper, we propose hardware and software mechanisms to solve the problems of real-time process scheduling and hardware interprocessor communication problems mentioned above. In software aspect, we propose a user-adjustable dynamically process scheduling mechanism, called Golden-Finger to make the specified mission-critical process to occupy a processor solely, that can achieve the maximum performance of the specified process. The other non-critical processes that are originally executed on the processor will be migrated to other processors. The above software mechanism is implemented in Linux operating system.

In hardware aspect, we propose an effective hardware communication mechanism called Back-Door for two PowerPC 405 processors in Xilinx ML310 FPGA development platform (as Fig. 2). The conventional multicore FPGA system, such as Xilinx Virtex II pro that the lacks of communication mechanism between two cores. This important function isn't implemented in original Xilinx Virtex II pro FPGA. Therefore, it is very difficult to let both dual cores alive.

The proposed Back-Door hardware communication mechanism can connect dual PowerPC cores in the Vitrex II pro FPGA that can overcome the problem of the dual cores FPGA but only can enable one processor for execution. Either in vender design tools, Xilinx EDK, or the corresponding Linux version, cannot enable dual PowerPC 405 concurrently. Therefore, the proposed hardware mechanism can improve the performance of Xilinx ML310 system dramatically by fully utilizing dual PowerPC 405 cores.

The organization of this paper is as following.Section 2 is the related works about scheduling mechanism under Linux and related project on Xilinx ML310. Section 3 presents implementations for both software flow and hardware flow. In section 4, we demonstrate the experimental results and the performance enhancements obtained when our proposed mechanisms are used. In the end, there is a conclusion in section 5.

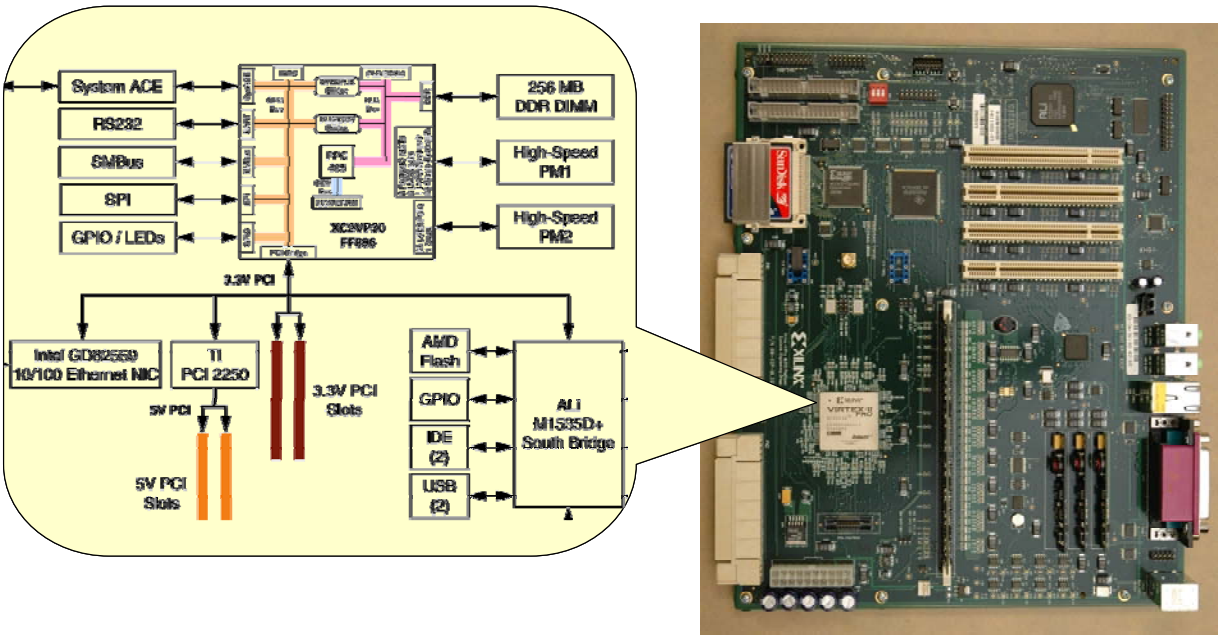

Fig. 2 Xilinx ML310 development platform and detailed system architecture of the FPGA.

2. Related Works

2.1. Introduction to the scheduler of the Linux kernel

The scheduling mechanism of the process scheduler in Linux 2.6.x[1][2][4][5][6] [7] is discussed below. It can find the most suitable task to execute in most of the running situations. When a program is executed and entered into the scheduling stage, this task will be putting in the corresponding priorities run queue. When the preset period of scheduling cycle is time-out, the system timer interrupts processor and triggers the scheduling function: schedule(), to check status of current tasks and the remaining running time.

If the tasks are time-out, they will swap out of run queue, and schedule() will find the candidate tasks in the run queue. According a normal queue, if there is a task running out its time slice, it will be removed from the head of queue to the tail of queue. In order to reduce time complexity of scanning the run queue, the scheduler maintains two priority arrays: active and expired for each processor.

The active array keeps the tasks that haven't time-out, and expired array includes the tasks that are time-out. The time slice of the task will re-calculate when a task runs out of its time slice before moving to expired array. Active array and expired arrays will be exchanged while all of the tasks in active array run out of its time slice. Fig.3 shows the mechanism of process scheduler in the Linux operating system.

First, schedule() calls sched_find_first_set() to find the first bit in active array, and this bit corresponds to highest priority and executable task. Then, this task will be executed by processor. The executing time of these statements doesn't influence on the number of tasks in the system due to its time complexity is O(1). While Linux kernel manages a shared memory multiprocessor system, every processor has its own run queue. Besides, within fix latency, the kernel has to check amount of tasks running on each processor is balance. If not, load_balanced() will move the tasks between processors to maintain balanced workload of each processor.

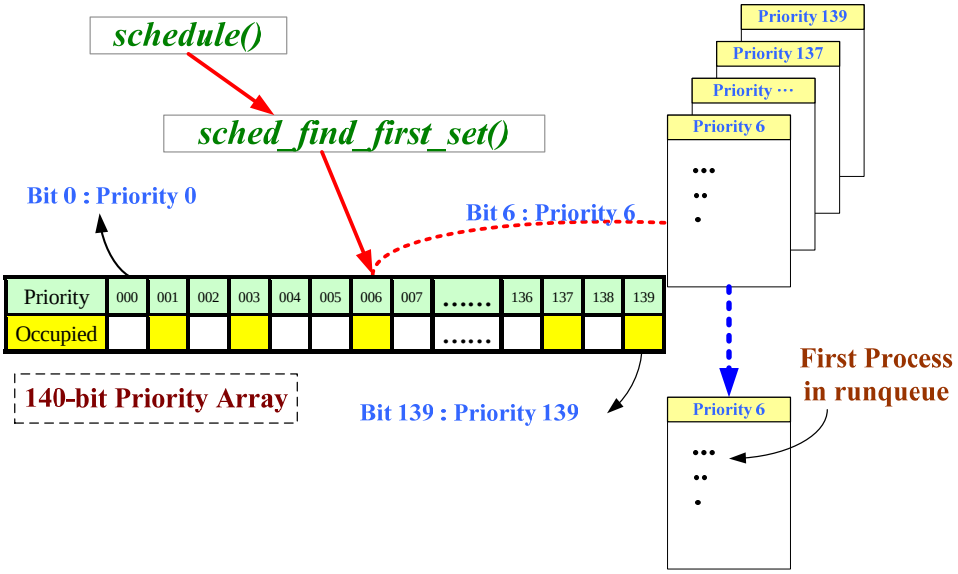

Fig. 3 The schedule mechanism of the conventional Linux kernel 2.6

2.2. *ARTiS*

ARTiS is a real-time extension of Linux that targets on shared memory multiprocessor system. The goal of ARTiS (Asymmetric Real-Time Scheduling) Project [3] is accelerating the response time of real-time tasks. RT0 means the hard real-time task which has to be done as soon as possible while RTn is soft real-time tasks. When ARTiS is booting, all of the processors will be partitioned into two groups, RT and NRT. The processors belong to RT group are assigned to execute real-time tasks, and the processors of NRT group are specialized to execute non real-time tasks.

Firstly, ARTiS arranges all RT tasks to RT processor, and NRT tasks will be moved to NRT processor. When there are free RT processors, NRT tasks will be moved to these RT processors. If the amount of RT tasks is larger than available RT processors, those un-assigned RT tasks will be arranged to NRT processor by the load balancer of ARTiS system. In order to implement the above capabilities, ARTiS adopts a task FIFO to save the moving tasks, instead of locking two run queues to diminish latencies during moving tasks, between NRT processors and RT processors. When there are any available RT processors, the RT tasks will be migrated to RT processors through the task FIFO. Therefore, these two run queues do not require waiting for the spin lock. Although this study proposes an asymmetric task scheduling mechanism to arrange the specific task on the assigned processor, extended from Linux kernel, ARTiS cannot executed on Xilinx Virtex II Pro FPGA. The scheduling mechanism of ARTiS system cannot be applied on Xilinx ML310 system.

2.3. *ATLAS*

ATLAS[10] is the first implementation of transactional memory coherence (TCC) and consistency architecture as a scalable implementation for transactional parallel systems. ATLAS is a FPGA-based system that primarily serves as a rapid software development platform for the transactional memory model. ATLAS uses the two PowerPC hard cores and attaches to PLB (Processor Local Bus of IBM CoreConnect Architecture) that connects the data port of PPC to TCC cache with configurable capacity as 8, 16, or 32 kB. The internal PPC data caches are bypassed and disabled to prevent interference with TCC.

Instruction fetches are straightly attached to the DDR controller. The internal 16-kB, 2-way set-associative instruction-side caches in the PPC is activated since instruction fetches bypass the TCC caches. Finally, BRAM (Block RAM, Xilinx on-chip SRAM cells) connects directly to each PPC through the OCM (On-Chip Memory) bus for transactional check point storage. ATLAS proposed a TTC architecture for both PPC processors. However, it still require complicated cache coherence design and additional software programming model. The operating system support of ATLAS is still needed to be improved.

3. The Designs of Software Scheduling and Hardware Communication Mechanisms

In modern multicore computer systems, the process scheduling capabilities and the interprocessor communication efficiency dominate the performance of computation. In this section, we proposed two mechanisms, from software and hardware aspects, to solve the above challenges. The former is a novel process scheduling mechanism, Golden-Finger mechanism, which can arrange the execution order of the processes and scheduling queues of the processors, clean up a free processor, and execute the specified mission-critical process on the processor solely. The later is an efficient hardware communicating mechanism, Back-Door, for interprocessor communication in the multicore system. The detailed description of these two mechanisms is mentioned below.

3.1. The Golden-Finger Scheduling Mechanism

The process scheduling mechanism in conventional Linux operating system is focused on fairly round-robin assignment, which is suitable for symmetric multiprocessor system. However, when the computer system is consisted of asymmetric multiprocessor, such as IBM Cell processor and TI OMAP, the imbalanced computation power of these processors will make the fairly scheduling policy inefficient. Besides, when the user wants to execute the urgency process in real-time, the fairly scheduling policy will lead to problems. Accordingly, we proposed the Golden-Finger mechanism which modifies the scheduling mechanism of operating system to improve the real-time capabilities. This mechanism allows the user to assign a program to occupy the particular processors. The detailed scheduling states of the proposed Golden-Finger mechanism are illustrated in Fig. 4.

The processing steps of Golden-Finger can be divided into five states. Firstly, the Golden-Finger is activated in the State 0. The user can assign a CPU as the target CPU, and the application, which is a mission-critical task and have to get the response as soon as possible. While Golden-Finger validates the correctness of above information, then the mechanism processes the next state. The second stage, State 1, will check the run queue of the target CPU and determine the candidate processes within the run queue, to empty the run queue of the target CPU. The total workload of these candidate processes will be evaluated for the following scheduling states. The third step, State 2, will retrieve the status of the alive CPUs by scanning their run queues, to determine the workloads of these CPUs. Then the mechanism will identify the most lightly-loading CPU to execute the processes that are migrated from the target CPU. The forth stage, State 3, will actually migrate the candidate processes from the target CPU. In order to implement this special system call, we modified the Linux Kernel and the patch of real-time capabilities for Linux in the ARTiS [3] system.

Therefore, the Golden-Finger can move the required processes from one CPU to another. The final stage of Golden-Finger mechanism is to fork the mission-critical process that is assigned by user. Then, the Golden-Finger mechanism can go back to State 0 for next round of scheduling cycle. A simple scheduling example of Golden-Finger mechanism is as demonstrating in Fig. 5 and Fig. 6.

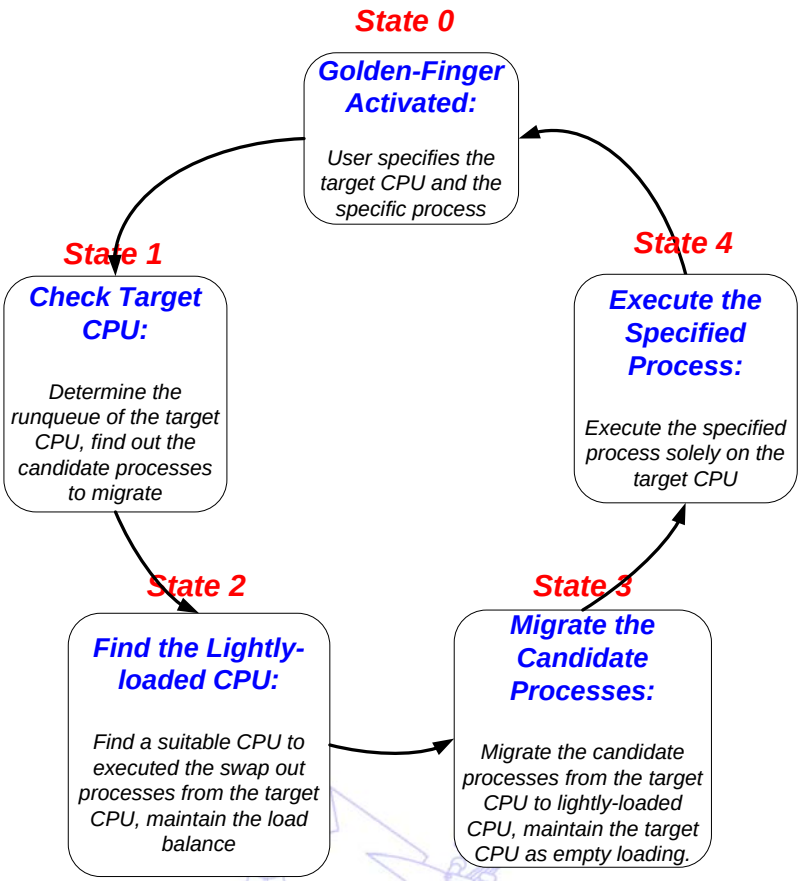

Fig. 4 The call graph of tasks' movements

In Fig. 5, it illustrates the snapshot of scheduling results by using conventional Linux process scheduler. In this snapshot, there are 11 processes scheduled in the runqueues of CPU 0, CPU 1, CPU 2, CPU 3 respectively. These processes are scheduled by fairly round-robin policy. Based on the execution situation of Fig. 5, the scheduling result of Golden-Finger mechanism is illustrated in Fig. 6. It assumes that the user assigns CPU 3 as the target CPU, and will execute the mission-critical process: "Process G". After scheduling by the Golden-Finger mechanism illustrated in Fig. 4, the Process 4 & Process 6 will be moved to CPU 0 due to its most lightly-loading run queue. Finally, the assigned "Process G" will be executed on CPU 3 solely, and can achieve its best performance to accompolish its real-time critical mission.

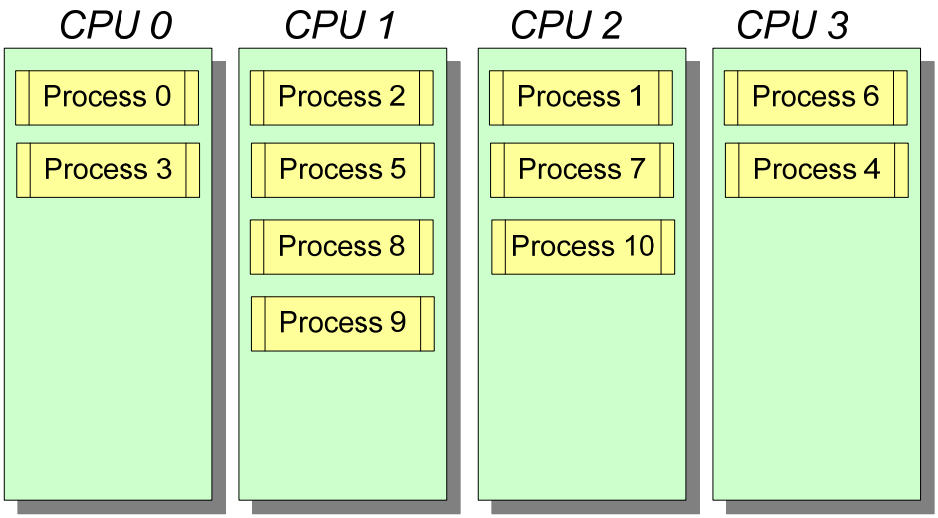

Fig. 5 The snapshot of processes executing status on conventional Linux system

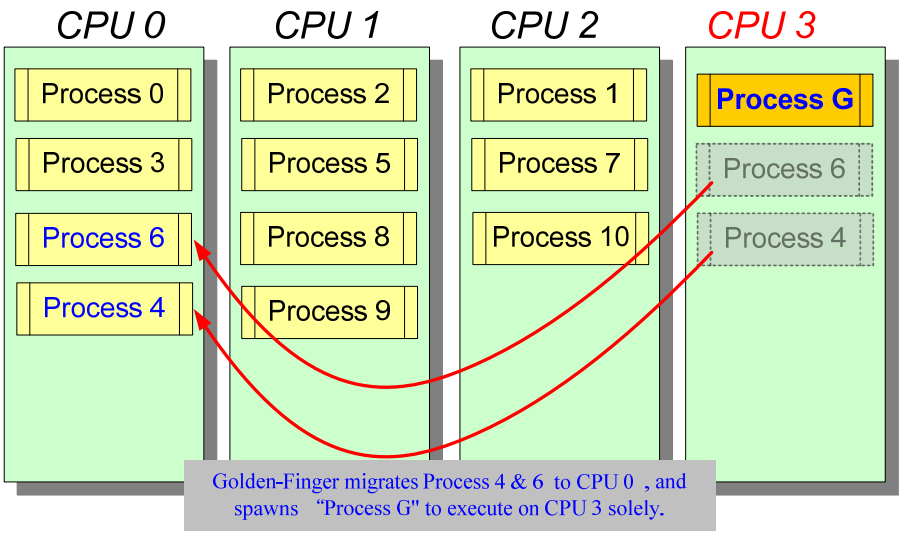

Fig. 6 The scheduling result of Golden-Finger mechanism

3.2. The Back-Door Interprocessor Communication Mechanism

The Back-Door interprocessor communication mechanism is based on Xilinx ML310 development platform, as shown in Fig. 7, which is consisted of a main FPGA chip, Xilinx Virtex-II Pro (XC2VP30), 256Mbytes DDR Memory, Ethernet, USB, PCI physical chips and connectors. The integrated System ACE CF controller is deployed to perform board bring-up and to load applications from 512MB Compact Flash card. The Xilinx Virtex-II Pro FPGA chip contains 2 PowerPC 405, 30,000 logic cells, and 2,400KBits BlockRAM (BRAM). The whole hardware system is developed by Xilinx ISE and EDK, which can generate whole pre-defined system with user-defined hardware modules.

Due to the unsupported features of Xilinx EDK, when two PowerPC 405 both connect to on the same PLB, EDK doesn't support the multicore features and only allows single PowerPC 405 to use the bus and booting at the same time. Therefore, we need to modify the whole system and find a suitable solution. The vender-suggested operating system of Xilinx ML310 is MontaVista Linux. Because it doesn't support multicore feature and doesn't provide kernel source codes, we modified the open-source Linux Kernel of PowerPC 405 version to implement the program loader of Back-Door hardware mechanism. .

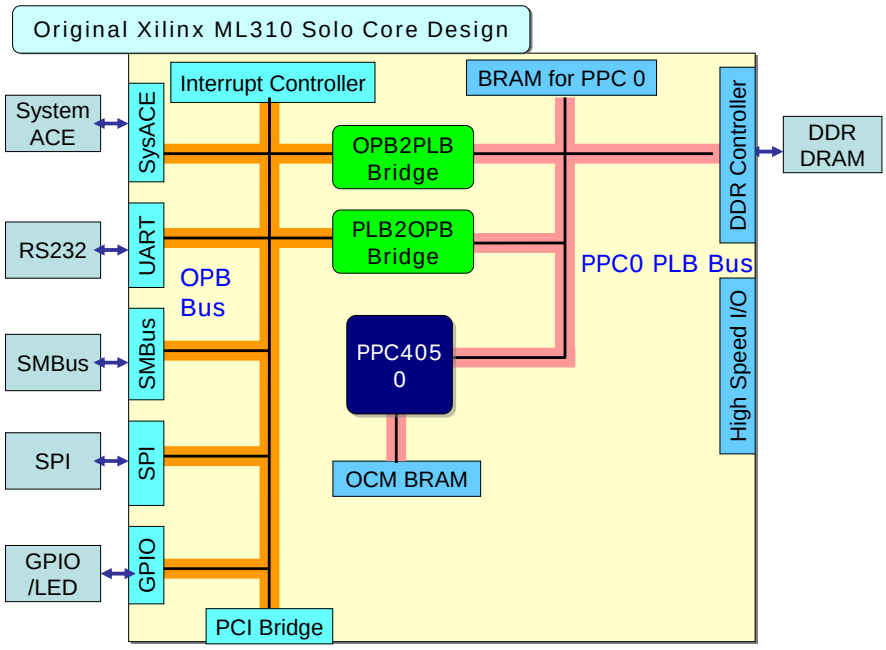

Fig. 7 The original architecture of Xilinx ML310 with sole PowerPC 405 core

The implementation of proposed Back-Door hardware communication mechanism is as below. Firstly, the sole core configuration of Xilinx ML310 is generated via Xilinx EDK, as shown in Fig. 7, to construct the fundamental system architecture. This system is consisted of a PowerPC 405 (PPC405 0), a processor local bus (PPC0 PLB Bus), a DDR DRAM controller (DDR Controller), low speed OPB bus (OPB Bus), inter-bus bridges (OPB2PLB, PLB2OPB Bridges), and required low speed peripherals (Interrupt Controller, SysACE, UART, SMBus, SPI, GPIO, PCI Bridge), which are integrated into Xilinx Virtex-II Pro FPGA.

Since the Xilinx EDK doesn't support the dual PowerPC configuration, the second PowerPC 405 only can be attached on an individual PLB bus. Then, the second PowerPC 405 (PPC405 1) and corresponding PLB bus (PPC 1 PLB Bus) is constructed manually, as shown in Fig. 8. However, it still lacks the capabilities of booting dual PowerPC 405, so the proposed Back-Door mechanism is constructed to connect two unconnected PLB buses to solve the problem that PPC405 1 only can access BRAM for PPC 1 but not DDR.

The proposed Back-Door mechanism is attached on PPC0 PLB Bus and PPC1 PLB Bus simultaneously. After the implementation of Back-Door mechanism, the PPC405 0 can communicate with PPC405 1 via Back-Door, by using conventional method of memory map I/O. When PPC405 1 communicates with PPC 405 0 via Back-Door, it can adopt the method of direct memory access, as shown in Fig. 9..

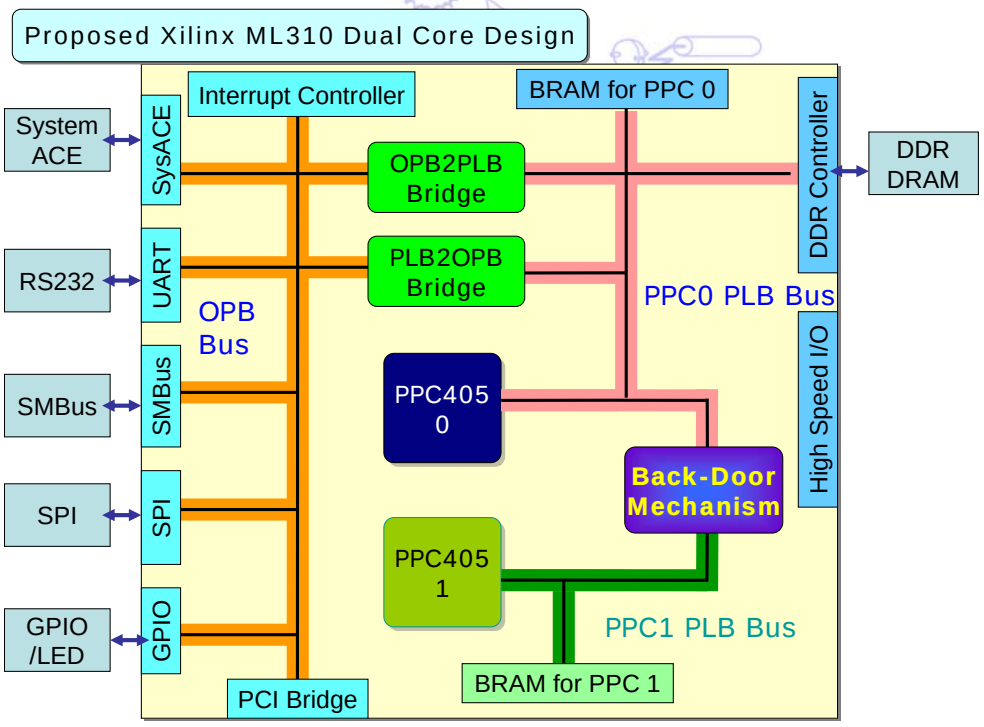

Fig. 8 The proposed Back-Door mechanism with dual PowerPC 405 cores

Accordingly, Linux operating system can be porting on his new Xilinx ML310 platform which is consisted of two PowerPC 405 processors and the proposed Back-Door mechanism. Due to the limitation of Xilinx EDK and Linux kernel, the operating system has to boot on PPC405 0, to control all of the peripherals. The Back-Door is recognized as a specialized MTD device. Then, the PPC405 1 has to be executed a loader program which is responsible for loading the application from Back-Door mechanism which is assigned by PPC405 0, and then execute it. After the assigned program is finish, the results can be sent back via Back-Door mechanism. The PPC405 0 can receive the results from PPC405 1 when notifying by Back-Door mechanism.

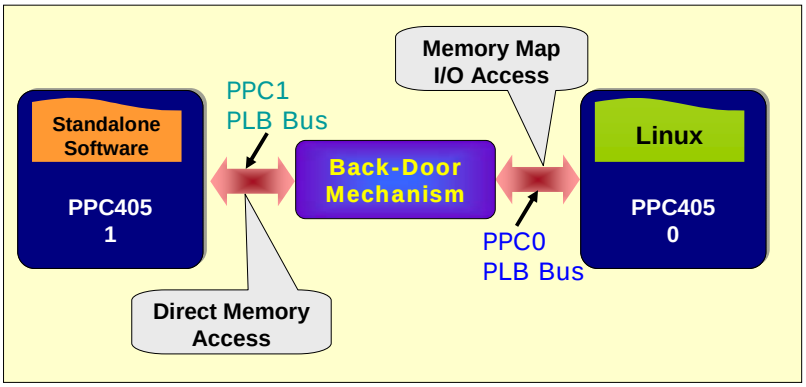

Fig. 9 The communication mechanism between PowerPC 405 1 and PowerPC 405 0

4. Experimental Results

The proposed Golden-Finger software mechanism and Back-Door hardware mechanism have be implemented on a dual core Intel PC and Xilinx ML310 platform respectively. The experimental results of these mechanism will be discussed in the following subsections respectively.

4.1. Experimental Results of Proposed Golden-Finger Software Mechanism

The target machine of implementing Golden-Finger mechanism is Intel x86 dual-cores PC, which is consisted of a Intel Pentium D 2.8GHz, 1GB DDR SDRAM, Linux Kernel 2.6.11 operating system, X-windows (GNOME 3.x) and GCC 3.4 compiler.

The benchmarks adopted in this experiment include LAME encoder, MPEG player, MPEG Decoder, and MPEG encoder. The experimental results are as shown in Fig, 10, and Fig. 11.

The prefix, “Heavy“, denotes that a LAME MP3 encoder is executed in the background when the experiment is taken to simulated the heavy-loading situation. The prefix, “Medium“, denotes that the MPlay plays a MPEG-1 video and a MPEG-4 video simultaneously to simulate the medium loading operating system.

Finally, the prefix, “Light“, denotes that the MPlay plays a MPEG-1 video when take the Golden-Finger experiments. The programs to be the special applications by using Golden-Finger mechanism are MPEG Decoder and LAME Decoder. “MP3 Enc (40MB)” and “MP3 Enc (100MB)”denote that the scheduled application by Golden-Finger mechanism is to execute LAME mp3 encoder to convert 40MB wave file and 100MB wave file to mp3 format, respectively.

Similarly, “MEPG4 Enc (50MB)” and “MEPG4 Enc (350MB)” denote that the scheduled application by Golden-Finger mechanism is to execute MEPG4 encoder to convert 50MB wave file and 350MB MPEG1 file to MPEG4 format respectively.

The execution time and speedup comparisons of conventional Linux and proposed Golden-Finger mechanism, which are both evaluated by the configurations of above three system loading (Heavy, Medium, Light), and four kinds of assigned applications for Golden-Finger mechanism, as shown in Fig. 10 and Fig. 11 respectively.

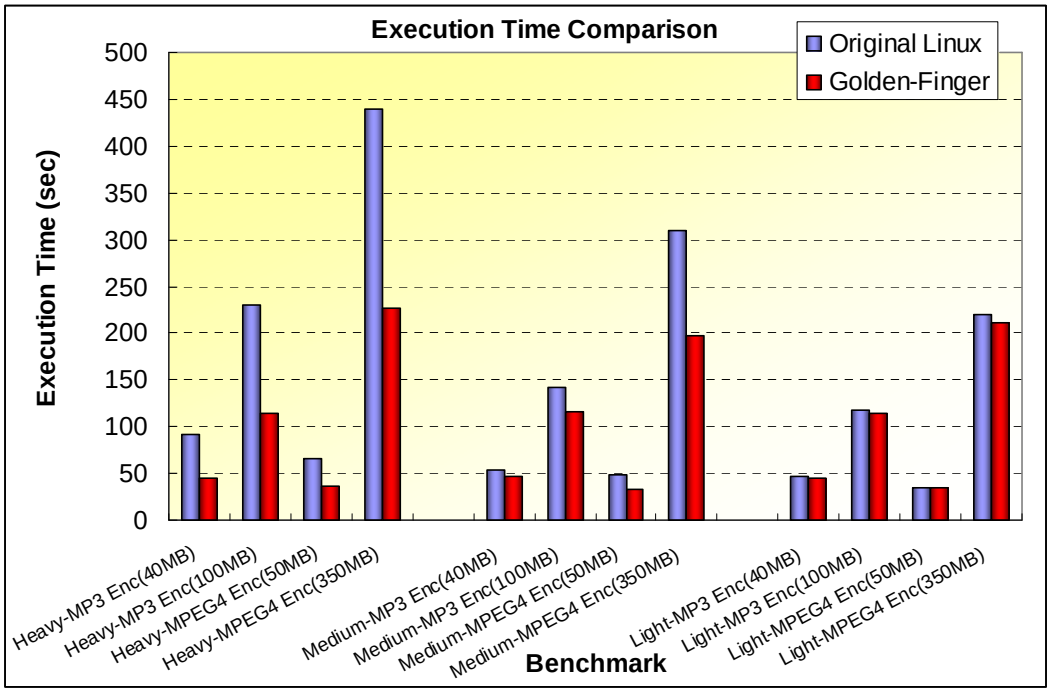

Fig. 10 The execution time comparison of conventional Linux and proposed Golden-Finger mechanism

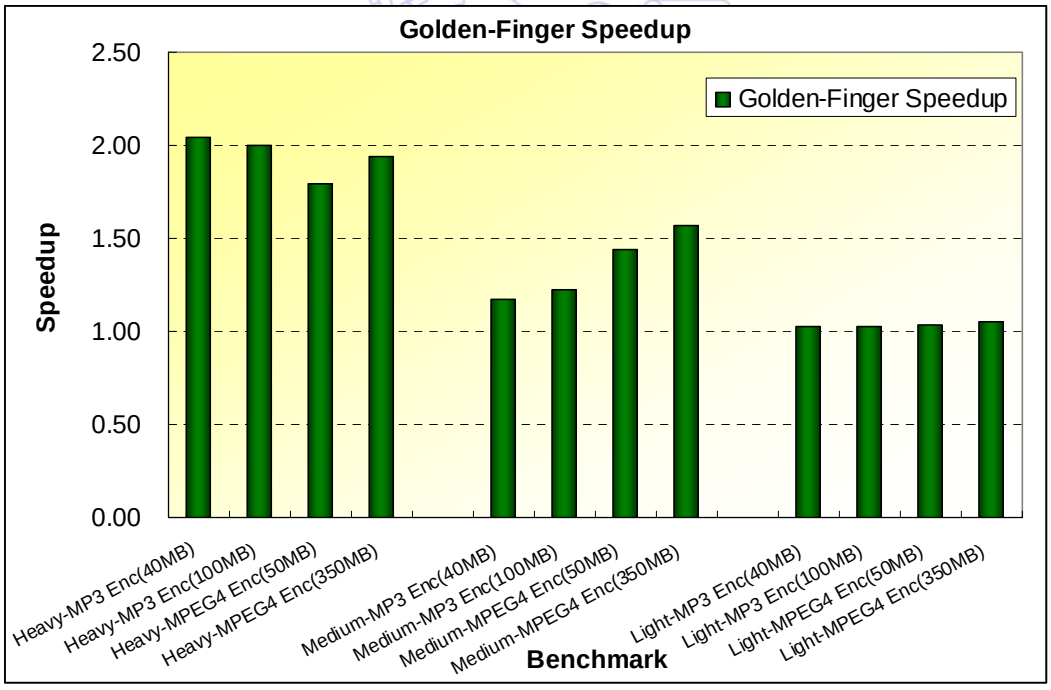

Fig. 11 The speedup comparisons of conventional Linux and proposed Golden-Finger mechanism

In these three systems loading, the proposed Golden-Finger mechanism can obtain dramatically execution time reduction, especially in the heavy system loading cases. The speedup can achieve 2.1X to fully utilize the assigned target CPU to accompolish the assigned process. In contrast to the configuration of heavy system loading, the light loading system only can obtain a few speedups, due to the assigned applications will not be delayed by the background programs in the conventional Linux cases.

4.2. Experimental Results of Proposed Back-Door Hardware Mechanism

The evaluated platform of Back-Door hardware mechanism is Xilinx ML310, as shown in Fig. 1. This experiment

adopts four benchmarks, which are selection sort (Selection Sort), dhrystone benchmark 2.1 (Dhrystone), fast Fourier transformation (FFT), and wavelet transformation (Wavelet), to evaluate the performance improvement of proposed Back-Door mechanism, from the aspects of execution time (Fig. 12) and speedup (Fig. 13).

Due to the limitation of Xilinx EDK environment, the applied operating system and complier are Linux kernel 2.4.25 and GCC 2.95.3 to meet the predefined board support package (BSP) and device drivers. The Virtex II pro FPGA is configured as 300 MHz, so the PowerPC 405 processors work at the same speed.

The execution time and speedup comparisons of original Xilinx ML310 and proposed Back-Door mechanism are both evaluated by the above four benchmarks, as shown in Fig. 12 and Fig. 13 respectively. The speedup can achieve 1.6X at the case of Selection Sort since it contains more potential parallelism and doesn't require a lot of interprocessor data transfer and DDR memory access. In contrast, the parallelism limitation of FFT makes it only can obtain 1.25X speedup by Back-Door mechanism.

It is noted that in current Xilinx ML310 platforms, even adopts Back-Door mechanism, the second processor (PowerPC405 1) can not be recognized as a normal processor to schedule by Linux, just only can be a specialized hardware accelerator to execute the manually modified applications, which can not be parallelized by conventional parallelizing compilers automatically. However, the dual-cores PowerPC 405 still can achieve the speedup from 1.25X to 1.6X. This evaluation result can reveal the capabilities of proposed Back-Door mechanism.

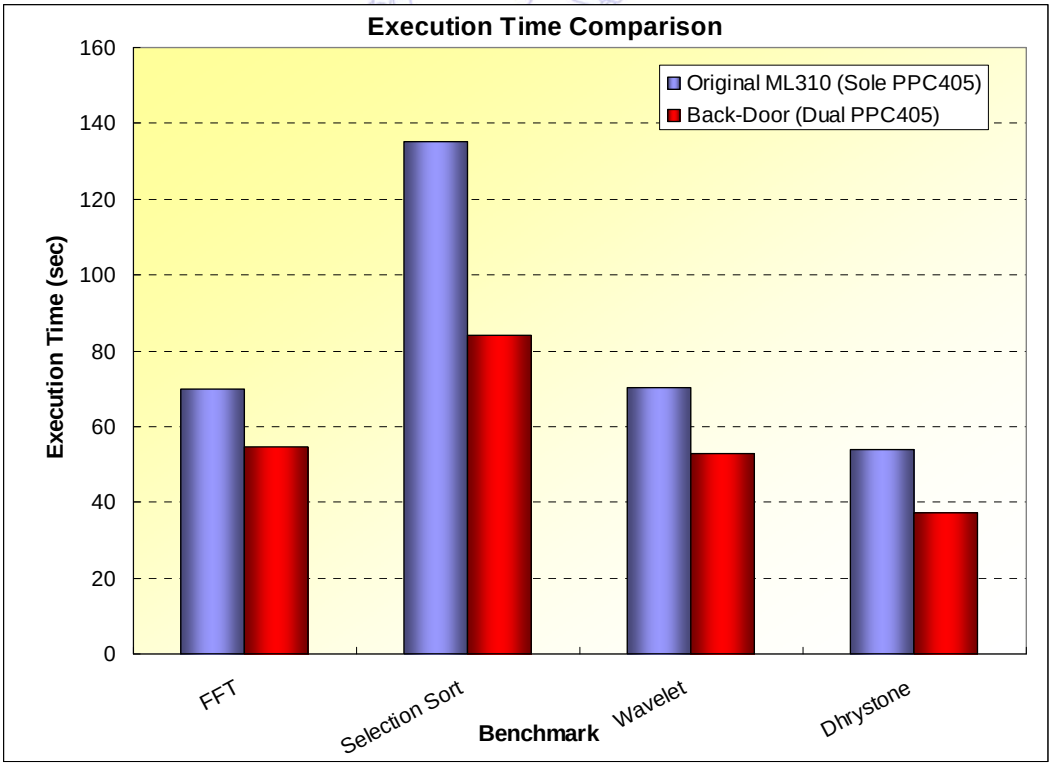

Fig. 12 The execution time comparisons of original Xilinx ML310 and proposed Back-Door mechanism

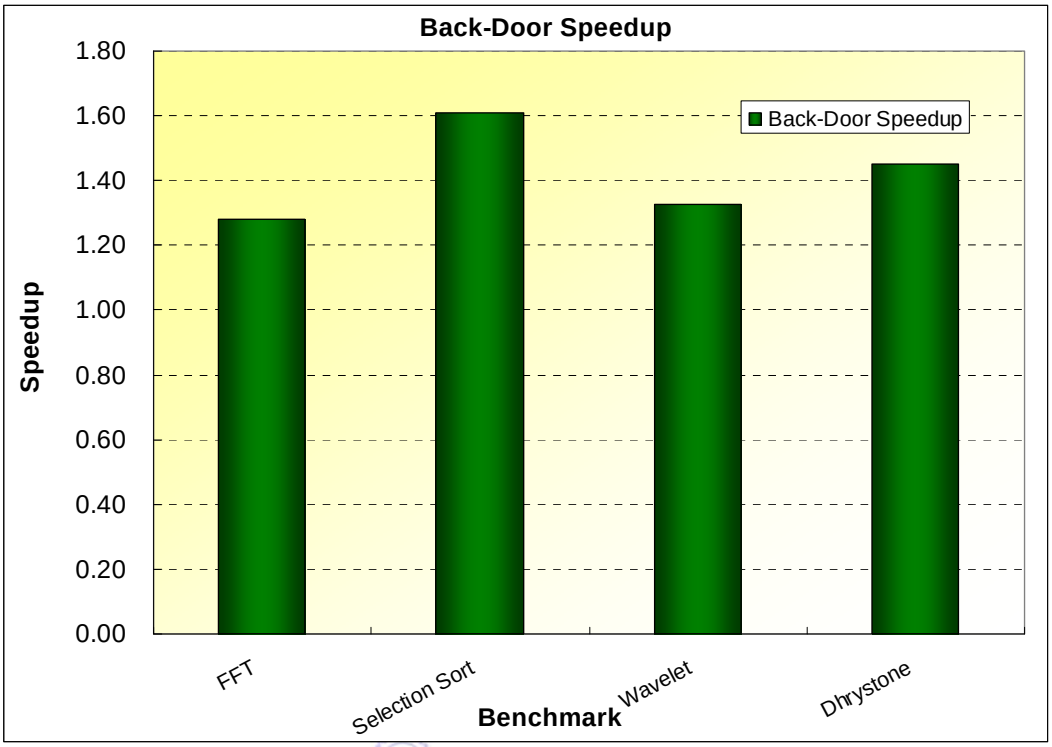

Fig. 13 The speedup comparisons of original Xilinx ML310 and proposed Back-Door mechanism

5. Conclusions

Continuously requirements of high-performance computing makes the computer system adopt more processors within a system to improve the parallelism and throughput. This paper proposed two mechanisms, Golden-Finger and Back-Door, to fully utilize the processor capabilities of modern multicore system, from hardware and software mechanisms, respectively. The proposed Golden-Finger software mechanism can dynamically adjust the scheduling policy to improve the performance of the specified process by occupying a processor solely. The hardware Back-Door mechanism can communicate two independent processors which can not be operated together, such as the dual PowerPC 405 cores in the Xilinx ML310 system. The experimental results reveal that proposed two mechanisms at different dual-cores computer system can obtain 2.1X and 1.6X speedups from variant benchmarks. These results can reveal the capabilities of the two mechanisms on multicore computer system.

Acknowledgement

This work is supported in part by the National Science Council of Republic of China, Taiwan under Grant NSC 100-2221-E-033-043

References

[1] J. Aas, Understanding the Linux 2.6.8.1 CPU Scheduler, Silicon Graphics, Inc., 2005.
[2] R. Love, Linux Kernel Development, SAMS, Developer Library Series, 2003.
[3] E. Piel, P. Marquet, J. Soula, and J.L. Dekeyser, “Asymmetric Real-Time Scheduler on Multi-Processor Architecture”, 20th International Parallel and Distributed Processing Symposium, Apr. 2006, pp. 25-29.
[4] G. E. Allen and B. L. Evans. “Real-time sonar beamforming on workstations using process networks and POSIX threads”, IEEE Transactions on Signal Processing, pp. 921-926, Mar. 2000.
[5] K. Morgan, “Preemptible Linux: A reality check”, Cuba: MontaVista Software, Inc., 2001.

[6] J. D. Valois. “Implementing lock-free queues”. In Proceedings of the Seventh International Conference on Parallel and Distributed Computing Systems, Oct. 1994.
[7] I-Tao Liao, Koan-Sin Tan, Shau-Yin Tseng, and Wen-Feng Chen, “Interprocessor Communication for PAC”, ITRI SoC Technical Journal, No.002.
[8] Intel Corp. Intel® Core™ Microarchitecture. http://www.intel.com/technology/architecture/coremicro/index.htm
[9] IBM Corp. The Cell Architecture. http://www.research.ibm.com/cell
[10] N. Njoroge, S. Wee, J. Casper, J. Burdick, Y. Teslyar, C. Kozyrakis, and K. Olukotun, “Building and Using the ATLAS Transactional Memory System”, 12th International Symposium on High-Performance Computer Architecture (HPCA), 2006.