Issued a Novel Method for Multimedia Processors

Szu-Hsiung Ko
St. John's University, Taiwan
96N04008@student.sju.edu.tw

Jih-Fu Tu
St. John's University, Taiwan
tu@mail. sju.edu.tw

Abstract *- A configurable dual-core embedded system for multimedia application of System-on-Chip (SoC) was introduced in this paper, in which we described a SoC consisting of a master processor and a slave processor. The master processor is represented by the simulator of SimpleScalar. In addition, the slave processor collocates with Xtensa processor, which is able to establish excellent multimedia application than the traditional designs. This proposed architecture can be configured in multiprocessors architecture, and verified by a provided simulation program. We get benefit in cost control of arbiter restructure. The characteristics of IP reused and portable architecture are exactly corresponded to modern complicate SoC design. The main functional blocks integrated in this system includes dual cores, local memory, cache memory, shared memory, shared bus, and so forth.*

Keywords - SimpleScalar simulation, configurable processor, instruction set simulators, and dual-core.

1. Introduction

The demand of multimedia communication embedded systems in mobile and portable devices application is growing nowadays. To realize multimedia communication, the implementations of audio and video compression standards are essential. More than that, a system demanding better performance requires higher clock frequency. Such that, multiple-function handhold devices are often challenged by power-saving, clock, speed, and heat dissipation.

To achieve higher performance with flexibility, the hybrid architecture has been proposed. The Operation-intensive functions are implemented with hardwired blocks, while other functions of less complexity are implemented with software which is executed by an application specific instruction processor. Current multimedia handhold devices are often using build-in multi-core approaches [1]. The system control and multimedia computation are executed by separated cores. Although heterogeneous cores reduce burdens by pre-defined tasks [10], but maintaining two sets of developing environments requires significant cost and manpower.

For example, multimedia on OMAP Solution, which combined an ARM processor with TI's digital signal processor (DSP) [12], implemented every function with software using an accelerated instruction set for multimedia processing while keeping the flexible software structure [6]. However, it is worthwhile to point out that OMAP is not designed only for multimedia embedded processor [12].

In this paper, we implement a dual-core system, and a shared memory bus arbitration dealing with shared memory accesses. Other components, like another core or hardware accelerators, can be added to this simulator system for co-simulation as long as they have a shared memory bus interface.

This paper is organized as follows. In Section describes the used simulation tools. Section 3 issues the proposed architecture of dual cores for applying to multimedia system. In Section 4 represents the implementation methodologies. Finally, we remark the conclusions and point out the future works.

2. Simulation Tools

A new configurable processor called Xtensa has been recently developed by Tensilica [11]. The Xtensa processor is based on instruction set architecture (ISA) that includes the basic instruction set, and an extensible function of adding user-defined instruction sets. It also allows the configuration of options, like interface options, memory subsystem options [5], and various OS options.

By generating the processor from the high-level language description, the platform designer regains the control over the cost, performance, and function attributes of the processor subsystem without being a microprocessor design expert [8]. The four hard-wired configurable categories [15] of Xtensa is shown in Table 1.

Table 1 Configurable Hard-wired of Xtensa

Configurable Hard-wired	Description
Instruction Set Architecture	- ALU functions on general registers - Floating-point unit, registers, state, interface - Coprocessors with new application specific data types configuration - High performance parallel arithmetic and DSP - Five or seven pipeline stages - One or two Load Store Units - Tensilica Instruction Extension (TIE) 16 and 24-bit instructions
Memory System	- Instruction cache size, associativity, line size - Data cache size,

	associativity, line size, write policy - Memory protection, translation Instruction, data RAM, ROM size, address range
Interface	- External bus width, protocol, address maps - Direct connection of system registers, queues, multi ported memories to internal data ports - Multiprocessor interconnect - JTAG debug and trace ports
Peripherals	- Timers - Interrupt controller: interrupt count, priority, type, fast switching registers - Exception vector addresses - Hardware breakpoint controls

2.1 SimpleScalar

The simulator, SimpleScalar, is a sim-outorder [13] and to use to simulate the master processor in this study. SimpleScalar is a set of execution-driven cycle-accurate instruction set simulators (ISS) of superscalar microprocessors. It comes with a complete development environment (compiler, debugger, and profiler) which allows the quick porting of any ANSI C application to SimpleScalar. The SimpleScalar toolset is composed of a GCC compiler, which was ported for SimpleScalar architecture and can generate SimpleScalar binary files. The assembler and loader along with the necessary libraries in the toolset produce SimpleScalar executables that can be fed directly into any of the simulators in SimpleScalar. The simulators themselves are compiled with the host platform's native ANSI C compiler. The simulators are equipped with their own loaders, thus you do not need to build the GNU binary libraries to run simulation.

The micro-architecture of SimpleScalar is derived from various ISA microprocessors. It supports speculative execution. The memory system has a load/store queue. The values are stored in this queue if it is speculative. Load instructions are dispatched to the memory system when the addresses of all previous stores are known. Loads may be completed either by the data from the memory system or by a value stored in the queue if their addresses match to each other. Speculative loads may generate cache misses but speculative TLB misses stall the pipeline until the branch condition is met.

2.2 Xtensa Instruction Set Simulation

Xtensa® Xplorer is an all-processor software development tool. This tool has an integrated graphical user interface. Used for processors new creations, simulation, Profile, debug and analysis program code. Tensilica's XCC C / C^{++} compiler [11] is an optimizing compiler with advanced optimization techniques, such as profile-directed feedback compilation, the process optimization, software pipeline analysis, static single assignment optimization, and reduce code size. The Xplorer DE [11] shows graphical results of Xtensa® Instruction Set Simulator (ISS) [11]. The program codes can accurately model the execution of processors, such as cache performance, execution cycle, branches, exceptions, pipeline states. The results are presented by forms and graphics. In addition, another tool, Xtensa® Modeling Protocol (XTMP) [11], is provided for modeling options.

2.3 Xtensa Modeling Protocol

The slave processor in this study was simulated with Xtensa ISS & Xtensa® XTMP. It provides a database-type application programming interface (API) to ISS, such that developers will be able to do complex hardware systems by getting the results from the simulation of the processors, and to complete the original concept of the model eventually. Xtensa XTMP is a set of software development tools which could create customized multi-thread simulation. A quick and accurate simulation of system-on-chip design consists of one or more processor cores become possible [9].

According to the contents in [11], XTMP is used to simulate multi-processor subsystems, or a single processor with complex structure. An initial multi-processor system can be linked with customized peripheral devices by using XTMP. In the early stage of a new design, XTMP is able to debug, profile, and validate the integration of SoC and software architecture [7]. Because XTMP Simulator executes in a higher level description, the simulation time can be drastically reduced comparing to HDL simulations.

3. Proposed Architecture

The traditional structure [14] shown in Figure 1 is referring the dual-processor architectures in [9]. In this research, the Xtensa ISS is replaced by SimpleScalar2, and we proposed multimedia processor of Xtensa as slave processor [18]. The Xtensa processor ISA that includes the basic instruction set, an extensible function of DSP engine for multimedia applications.

The simulator framework of we proposed is shown in Figure 2. The modules within the frame are implemented in our simulation. We chose to build the multi-core simulator based on SimpleScalar and Xtensa ISS, which are implemented in C language, all the components are implemented in SystemC which provide interfaces to be connected by SystemC channels.

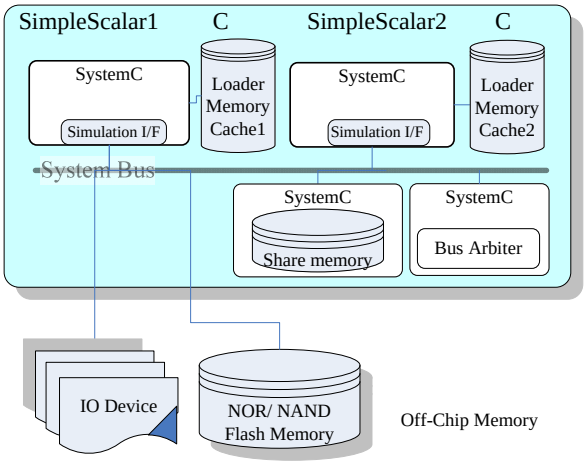

Fig. 1. The architecture of Ref [14].

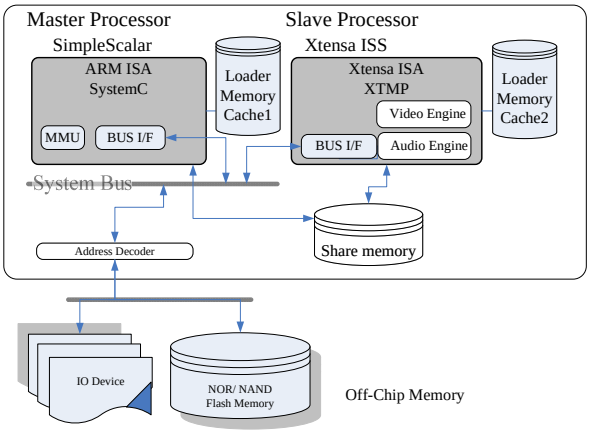

Fig. 2. The proposed architecture.

3.1 Memory Design

This proposed system uses a shared memory, shown in Table 7. In this simulation system, the common memory can be configured into different sizes. The same procedure will be executed in the simulation systems of different memory sizes, and it will cause different speed performance. Memory units can be configured into different widths and sizes in order to meet the requirement of data accuracy and bandwidth. However, if the memory size is huge, its cost will be higher than the core.

Table 2 Memory Management

Region Protection	**Description**
Memory Region	The 4G space is divided into 8 equally sized regions with 512M bytes
Memory Region Access Mode	Through Access-Mode setting. Bypass, Allocate, and No Allocate, Write-Back, Write-Through

The cache memory speed is configured into optimal values to improve the performance of local memory. When local memory or cache is configured, the impact to their processors will be taken into consideration [17]. The performance is the tradeoff between area, power consumption, and speed. Cache miss uses more bandwidth. We adjusted the cache and local memory configuration to optimize the performance. Memory management shown in Table 2 is performed by the MMU configuration. If memory configuration adds a virtual memory unit, it allows processor to handle more complex programs. If virtual memory does not exist, the application programs will be located in fast memory. We use MMU to utilize master processor efficiently and shown in Figure 2.

3.1.1 Memory Model The defaults of memory model and the cache model are shown in Table 3 and Table 4, respectively. Both are used to sim-cache for the split organization.

Table 3 Local Memory Model

Local Memory Model	**Description**
Inst-RAM [0,2] Size	128Kbytes
Inst-ROM [0,1] Size	256Kbytes
Data-RAM [0,2] Size	128Kbytes
Data-ROM [0,1] Size	256Kbytes
RAM Access Width	64 bits
RAM Access Latency	1 Cycle of access latency

Table 4 Cache Model

Cache Model	**Description**
L1 Inst-Cache size	16 Kbytes
L1 Data-Cache size	16 Kbytes
L2 unified cache size	256 Kbytes
Cache Write Policy	alternate Write-Back and Write Through for Data-Cache
Write Buffer	16 Entries
Cache Replacement	L1 and L2:LRU policy
Associative	L1:2-way set associative caches L2: 4-way
Line Size	L1:Cache locking per line, line size 32 bytes L2:64 bytes
Cache Access Width	64 bits
Cache Memory Access latency	L1:1 cycle of access latency L2:10 cycle of access latency

3.1.2 Memory Mapping The specific purpose functions of this proposed multi-core processor are allocated in the memory addresses shown in Table 5, and the memory management unit and shared memory model are listed in table 6 and Table 7, divisionally..

Table 5 Memory Mapping

Memory Mapping	**Description**
Data RAM 0	0x3FFE 0000
Data RAM1	0x3FFC 0000
RAM0	0x4000 0000
System ROM	0x5000 0000
Reset Vector	0x5000 0000
System RAM	0x6000 0000
Double Exception	0x6000 03C0
Window Vectors	0x6000 0000
L5 Interrupt	0x6000 0240
L4 Interrupt	0x6000 0200
L3 Interrupt	0x6000 01C0

L2 Interrupt	0x6000 0180
L1 (User Interrupt)	0x6000 0340
Kernel Exception	0x6000 0300
Debug Exception	0x6000 0280
NMI	0x6000 02C0

Table 6 MMU Model

MMU	Description
Memory Management Terms	Page Table Entry, Isolate, Identify Map, Static, Wired, Auto-Refill, Ring, and Address Space Identifiers
Translation Look aside Buffer (TLB)	I-TLB with 64 entries fully associative D-TLB with 128 entries fully associative
MMU for OS	Linux provides demand paging and memory protection

Table 7 Shared Memory Model

Shared Memory	Description
ROM Size	16Mbytes

3.2 Bus Design

Communication bus is an important interface. According to the conclusion of [14], the channel width and access speed can be configured independently. We designed this bus as master-slave architecture. If master device needs to communicate with others, it will send a request signal to the arbitration mechanism. When the permission is obtained, master device gets the right to access the bus. On the other hand, the slave device is not titled to send request signals. They are waiting for requests passively.

Table 8 shows the bus model of Xtensa. We also consider using a share bus, which is able to have many devices and arbitration to share with, not only communication method should be considered. The bus interface needs suitable bandwidth for high performance hard-wired needs.

The bus arbitration mechanism is central parallel arbitration. When two or more masters at the same time to have a request, arbitration mechanism will do the permission response in order in accordance with pre-define priority code.

Table 8 Bus Model

Bus Model	Description
Bus Interface	64 bit Width 32 Interrupt mechanism Master-Slaver Architecture
Communication Mechanism	Shared memory allocation, Deletion, Mail box services
Arbitration Mechanism	Central Parallel Arbitration 32 bit Programmable Register

3.2.1 Communication Mechanism According to the conclusion [14], they can take shared memory as the inter-processor communication media to work with the synchronization scheme in our simulator framework. Any processing component that wants to access the shared memory should implement special load/store instructions. We assume that all the processing components are connected to the shared memory using a shared memory bus. A dedicated bus can ensure fast access and communications. At the same time, bus arbitration is provided to avoid conflict. The shared memory module is written in SystemC.

In Table 9 shows the communication mechanism, we also provide a library of shared memory related communication services to facilitate parallel computing tests. The services are designed for share memory allocation, deletion and mail box services. Other services are to be added in the near future, such as semaphore and message queue.

Table 9 Communication Mechanism

Communication Mechanism	Description
Shared Memory Allocation	Allocate a piece of memory space in the shared memory and return the index id of this piece of memory.
Shared Memory Deletion	Free a piece of memory space in the shared memory according to the index id.
Mail Box Services	Wait/Send a message from/to the Mail box services

3.2.2 Arbitration Mechanism According to the conclusion [14], the safest approach to dual-core synchronization is round-robin. Round-robin allows every processor to run cycle-by-cycle alternately.

Our experiment synchronizes one SimpleScalar module and one Xtensa ISS running in Central Parallel Arbitration approach. A method is needed for inter-core synchronization to get well communication performance while guarantees the accuracy. In this paper, we designed a communication based synchronization approach.

In this approach, synchronization between dual cores and optional I/O devices are achieved when communication is necessary. We can integrate this synchronization mechanism into the arbitration mechanism.

When a shared memory access instruction from processors is decoded, the shared memory access request and the simulation cycle count will be sent to the arbitration. Arbitration compares the current cycle number of all the processing components in the system, and grants the one with the smallest cycle number access to the shared memory.

Table 10 Central Parallel Arbitration

Processing components	Number Description
Processor_1	Master Processor
Processor_2	Slaver Processor
I/O Device_1	USB (Optional)
I/O Device_2	Ethernet (Optional)
I/O Device_3	DMA (Optional)
Reversed component_1	(Optional)
Reversed component_2	(Optional)
Reversed component_3	(Optional)
Reversed component_4	(Optional)
Reversed component_5	(Optional)

We proposed the priority from 0 to 5 decreases, the priority of 0 processing component is the highest.

Table 11 Priority of two processing components

PT.	0 (Hi)	1	2	3	4	5 (Lo)
0000	Opt.	Slaver	Opt.	Opt.	Opt.	Master
0001	Opt.	Opt.	Slaver	Opt.	Opt.	Master
0010	Opt.	Opt.	Opt.	Slaver	Opt.	Master
0011	Opt.	Opt.	Opt.	Opt.	Slaver	Master
0100	Master	Slaver	Opt.	Opt.	Opt.	Opt.
0101	Master	Opt.	Slaver	Opt.	Opt.	Opt.
0110	Master	Opt.	Opt.	Slaver	Opt.	Opt.
0111	Master	Opt.	Opt.	Opt.	Slaver	Opt.
1XXX	Res.	Res.	Res.	Res.	Res.	Res.

We proposed the arbitration register is physical address. The register description is shown in Table12.

Table 12 Programmable Register Description

Register	**Description**
Arbitration Register	32 bit Access Width
Physical Address	0x01000000
Reset Value	0x00000000

4. Schematics Methodology

We proposed the architecture of dual processor using SimpleScalar and Xtensa tool [2]. The combination of these two complicated simulators raises several issues. We proposed some novel ways to solve these issues as follows:

1). Xtensa Processor is lack of WinCE operating system (OS) support. To solve this problem, we designed another processor, simulated with SimpleScalar, which is rich in popular OS support. SimpleScalar also provide ARM ISA simulator, too.

2). SimpleScalar simulator does not support modern dual-core neither simple DSP. Currently, the demands of multimedia application, such as TI's OMAP TMS320C30 DSP for fast processing [12], are growing. Such that we replaced the SimpleScalar2 processor by Xtensa ISA core. The DSP engine of Xtensa ISA core is configurable. The proposed architecture is shown in Figure 2, in where this multimedia processor is a slave processor of this proposed system.

3). The inter-processor communication of dual-core was implemented with bi-directional mailbox primitives. The mechanism of communication is very important. We design a bus interface to simulate the protocol between two processors.

4.1 System Processing

The master processor includes basic ARM instruction sets. It is represented by SimpleScalar simulator. In order to support popular OS, the processor has to consist of the options, such as MMU, memory map, cache policy, interrupts, debug interface, and except-instruction set. The schematic is shown in Figure 2 as above.

4.2 Multimedia Processing

The slave processor, using Xtensa instructions, is proposed for multimedia processing [3] [4]. The application software is cross-compiled with instructions and data code on the host PC. All the tool chains of slave processor are provided by Tensilica. We can monitor the results of the program through the JTAG interface on the host PC. Xtensa® Xplorer is one of software development tools, which are an integrated graphical user interface for processors new creations, simulation, profile, debug and program code analysis.

The memory system for multimedia software has different parameters. We issued a baseline profile in this paper. We can implement processors with various configurations by the tradeoff of area, cost and power in the future.

5. Conclusion and Future Works

We issued a novel method of an embedded dual-processor design for multimedia applications with a configurable processor in this paper. We have referred the dual core architecture in [9] and replace its SimpleScalar 2 with Xtensa [10] furthermore to support the new configurable processor, in where uses Xtensa as the slave processor (referring to Figure 2).

We proposed a mechanism of arbitration is exactly met the cost control of SoC design. We issued a programmable register in the bus model, which supports ten processing components, there are two processors, three I/O Devices (Optional) and five reserved components (Optional). The shared bus is multiplexed (instead of a three-state approach), and every one of these ten processing components can request the shared bus without any limitations. Arbitration for those accesses is performed by the programmable register.

Future, we will use ARM926-EJS processor as a master processor to handle system processing. It is rich in the operating system support. We can use the test chip with WinCE kernel to manage and control I/O. A standard extensible interface to record and playback the video and audio data from external I/O is used for reproducible real-time experiments. We can demonstrate fast processing of video and audio workloads with FPGA [16].

The inter-core communications between these processing cores was implemented with bi-directional mailbox primitives. That will be described in our future research paper. We will show the experimental results in the FPGA and ARM emulation workload by using system software provided.

References

[1] Chien-Chang Wang, "*A Dual RISC Core SoC Platform*," The Master Thesis of Department of electrical engineering, National Cheng-Kung University, Taiwan, June 2004.

[2] Chen-Chien Wang, "*Design and Implementation of a dual-ISA embedded microprocesso*," The Master Thesis of Computer and Communications Engineering, National Cheng-Kung University electrical engineering Institute , Taiwan, June 2004.

[3] Chin-Ming Chen, " *MPEG Audio Codec Design Using High Frequency*, "The Master Thesis of Institute of Electronics and Information Engineering,, National Yunlin University of Science and Technology. Taiwan, June 2004.

[4] Jia-Hsiung Huang , "*MPEG-2/4 low-complexity advanced audio coding optimization and implementation on dual-core processor*" ◦ The Master Thesis of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan, July 2006.

[5] Grant Martin, "*Recent Development in Configurable and Extensible Processor*", Application-specific Systems, *Architectures and Processors*, IEEE, pp. 39 - 44. 2006

[6] Yu-Yuan Su "*Design and Analysis of a DSP Scheduler and a Dynamically Partitioned H.264 Encoder on Dual-Core Platforms*" ◦ The Master Thesis of Institute of Computer Science and Engineering, National Chiao Tung University ◦ Hsin-Chu, Taiwan, June 2006.

[7] Albert Wang, Earl Killian, Dror Maydan, Chris Rowen, "*Hardware/Software Instruction Set Configurability for System-on-Chip Processors*", *Design Automation Conference*, IEEE, 2001, pp. 184 - 188.

[8] David Goodwin, Chris Rowen, Grant Martin, "*Configurable Multi-Processor Platforms for Next Generation Embedded Systems*" Computer Design. Proceedings. International Conference, IEEE, Sept 17-20, 2000, pp. 335 – 342.

[9] Fei Sun, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha. "*A Synthesis Methodology for Hybrid Custom Instruction and Co-processor Generation for Extensible Processors*" Computer-Aided Design of Integrated Circuits and Systems, IEEE, Volume 26, Issue 11, Nov. 2007, pp. 2035 – 2045.

[10] Fei Sun, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. "*Application-Specific Heterogeneous Multiprocessor Synthesis Using Extensible Processors*". Computer-Aided Design of Integrated Circuits and Systems, IEEE, Volume 25, Issue 9, Sept. 2006, pp. 1589 – 1602.

[11] http://www.tensilica.com/products/xtensa_overview.htm.

[12] http://focus.ti.com/general/docs/wtbu/OMAP Platform.

[13] http://www.simplescalar.com

[14] Rongrong Zhong, Yongxin Zhu, Weiwei Chen, Mingliang Lin "*An Inter-core Communication Enabled Multi-core Simulator Based on SimpleScalar*" AINAW07.The 21st International Conference on Advanced Information Networking and Applications Workshops, 2007, Volume 1, Issue , 21~23 May 2007 pp. 758-763.

[15] Chris Rowen, Dror Maydan "*Automated Processor Generation for System-on-Chip*". The proceedings of the 27th European Solid-State Circuits Conference on 2001. ESSCIRC 2001, 18-20 Sept. 2001 .pp.464-469.

[16] Jin Ho Han, Mi Young Lee, Young hwan Bae, and Hanjin Cho "*Application Specific Processor Design for H.264 Decoder with a Configurable Embedded Processor*" *ETRI Journal*, Volume 27, Number 5, October 2005, pp.491-496.

[17] Mohsen Soryani, Mohsen Sharifi, Mohammad Hossein Rezvani. "Performance Evaluation of Cache Memory Organizations in Embedded Systems," *International Conference on Information Technology* (ITNG'07), 2006, pp.491-496.

[18] Szu-Hsiung Ko and Jih-Fu Tu, "The Configurable Architecture Design of Multimedia Processor," The proceeding of conference on 2008 Asia Pacific High Speed Circuit Design. July 22~July 23, 2008, Taiwan, pp. 130-137.