An efficient and flexible modelling approach for multi-DSP system

Zheng-Mao Zhou, Shun-Hong Zhong, Ming Cai*

School of Computer Science of Technology, Zhejiang University, Hangzhou of Zhejiang, China

Received 1 July 2014, www.tsi.lv

Abstract

With the development of the information technology, single digital signal processor (DSP) cannot meet the requirements of massive data processing. Multi-DSP parallel processing mode has been commonly used in real-time information processing system. New technology is also making it much easier to integrate multiple DSPs into a single silicon chip. However, designers of a new multi-DSP system and software are confronted with problems such as short product life-time. Meanwhile, product verification is indispensable before launching into the market. In this paper, a multi-DSP simulation platform is developed to solve these problems. The designed multi-DSP platform is based on an ISS-SystemC structure and has three common interconnect interfaces. An AMBA bus-shared memory model is designed for the expansion of the simulation system. A thread-agent method is proposed to optimize the performance of SystemC thread and the experiment results show that the multi-DSP parallel processing mode can improve the processing performance of the system significantly.

Keywords: simulation platform, multi-DSP, ISS-SystemC, SystemC optimization

1 Introduction

With the rapid development of information technology, Digital Signal Processor (DSP), with its unique structure and fast data processing capabilities, has been widely used in mobile communication, radar signal processing, real-time image processing and other fields. However, with the increasing amount of data processing, a single DSP system cannot meet the requirements of large-scale computation. Multi-DSP parallel processing system with characteristics of real-time, high accuracy and large data throughput has already been applied to complex large-scale data processing systems [1]. As designers of new multi-DSP parallel system and software are increasingly faced with short product life-time. The resulting time-to-market constraints are contradicting the continually growing system complexity. Nevertheless an extensive design-space exploration and product verification is indispensable for a successful market launch [2]. In this case, simulation tools are essential both for designers and researchers in computer architecture, due to their ability of studying and validating new designs without the cost of actually building the hardware.

For this purpose, a Hardware/Software co-simulation is very useful for the validation of both hardware and software components in multi-DSP parallel system. Co-simulation can also evaluate the performance of the whole system at an earlier stage before building a prototype. However, there are usually gaps between software components and hardware components in traditional mixed co-simulators. It is not until the emergence of SystemC that the hardware and software are bumped together, which makes the co-simulations of software and hardware seamlessly. SystemC is one of the most popular system-level modelling languages as it provides a common language for both the hardware and software designers [3].The single simulation engine (SystemC) ensures the design of the co-simulation to be easier and more efficient.

As is shown in Figure 1, a novel ISS-SystemC framework is proposed for designing multi-DSP systems. An open source ISS (C6Xsim) [4] of DSP is used to abstract the model of the real programmable device where the software should run and SystemC is used for transparent integration of ISSs with other peripherals. The ISS is designed as a c++ class with an ISS-wrapper interface and each DSP core is an instance of this class. As the interconnect between DSPs is very flexible, we designed several peripheral interfaces (EMIF, HPI, McBSP) for the expansion of the platform with SystemC. The ISS-wrapper is used as an intermediate transformation layer for mutual transformation between the read (write) requests from ISS and SystemC transactions, which can be deal with interface module. The SystemC Module in Figure 1 consists of communication mediums (Shared Memory, FIFO), AMBA Bus and Decoding Controller, etc. Each DSP core is connected to this component through a peripheral interface for communicating with each other.

The main contributions of this paper consist of the following two aspects:

1) We provide a flexible and scalable multi-DSP model for C62x- series processor at a cycle-accurate level of abstractions, which could meet the needs of debug, functional verification of the multi-DSP system, thus helping shorten the system development cycle, improve

* ***Corresponding author*** e-mail: zhouprogram@zju.edu.cn

product quality and reliability, and reduce development costs.

2) A serial scheduling mechanism is used in the SystemC-kernel to reduce the design complexity of the system [5], which allows only one sc_thread running at the same time in system. We proposed an OS-thread agent approach to avoid the shortcomings of sc_thread serial execution. This method makes all the DSP cores to run in parallel when there is no communication between any two DSP cores, which can accelerate the simulation speed greatly, especially in the modern multi-core host machine.

The paper is organized as follows: Section 2 introduces the related work and Section 3 introduces the implementation of multi-DSP simulation platform. Section 4 introduces the system expansion and performance optimization, followed by experiment verification. Section 6 gives the conclusion of the paper.

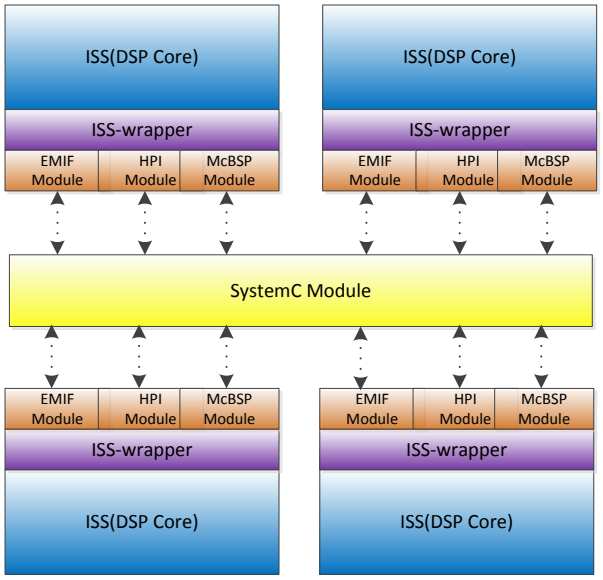


FIGURE 1 ISS-SystemC Framework.

2 Related work

In this section, we begin with a brief introduction of the open source IIS of DSP (C6Xsim). Then we discuss the IIS-SystemC framework. Finally, the flexible interconnect of c62x- series DSP is presented.

2.1 C6XSIM

C6Xsim [4], designed by Vinodh Cuppu in the University of Maryland, is an open source VLIW processor simulation tool. It offers a complete cycle accurate, execution driven simulation environment of Texas Instruments TMS320C62x series of very long instruction words DSP processors. This ISS accurately simulates various stages of the pipeline and gathers statistics to a considerable degree of accuracy of execution. So it can be used for microarchitecture development, performance analysis and application analysis on DSP processors. In our system, we have added additional functionality to the original C6Xsim to enable the interconnection between the DSPs.

2.2 ISS-SYSTEMC FRAMEWORK

The IIS-SystemC is a popular HW/SW co-simulation framework, which has been used by many academic institutes. [6] is based on qemu-SystemC structure and the experiment results show co-simulation at the cycle-accurate (CA) level is much faster than the conventional ones. [7] developed a complete multi-ARM simulation system based on SWARM-SystemC framework [8] and they found that the effectiveness of a particular system configuration strongly depends on the application domain and generated traffic profiles. [9] proposed a transaction level modelling (TLM) approach for designing an OpenRISC-SystemC co-simulation framework and [10] designed multiple interconnected processors with distributed memory in the SimpleScalar-SystemC framework. [11] proposed an ISS-SystemC co-simulation framework in which HW models can be modified on the fly while keeping the SW parts unchanged. This means that a new ISS can be added to the system with no complex changes.

2.3 INTERCONNECT OF C62X-SERIES DSP

As we know that the cascaded modes of multiple DSPs and high-speed data transfer between them are significant in multi-DSP parallel processing systems. Therefore, the high-speed data or special interfaces are generally used as the cascaded interfaces among multi-DSP systems to meet the requirement of data transfer rate. The Texas Instruments provides three high-speed interfaces for the interconnect of C62X-series DSPs [12]. Table 1 shows the characteristics of these interfaces. The external memory interface (EMIF) is the interface between the external storage and C6X DSP. This interface is a generic mass data transmission channel and the transmission speed can reach 16 Gbit/s in general. The host port interface (HPI) is an interface between master and DSP. The master cannot only access all storage space of the DSP directly, but also the chip memory mapped peripherals. This interface is mainly used to control and configure the slave DSP. The multi-channel buffered serial port (McBSP) is usually used to connect the serial peripheral and the transmission speed is only 0.125Gbit/s.

TABLE 1 Interface of C62x-series DSP

Name	Speed (Gbit/s)	Number of Signal Lines	Typical Interconnect Structure
EMIF	16	36	shared memory
HPI	0.8	40	master-slaver
McBSP	0.125	6	peer-peer

Figure 2 shows the typical interconnect of C62xx-series DSPs with the referred interfaces in our simulation platform. Figure 2a shows the DSP interconnect based on

a shared memory. This interconnect structure is usually used for large amount of data exchanges between DSPs, which could be handled according to the high-speed transmission characteristics of EMIF. Figure 2-b shows a typical master-slave interconnect structure based on HPI and it requires no additional storage medium. As the centre of the topology, the master-DSP is usually used for the control centre of the whole system and each slave-DSP receives data (control commands and configuration data) from the master-DSP. Figure 2-c shows the peer-to-peer topology structure, which is mainly used for small amount of data communication between the two DSPs.

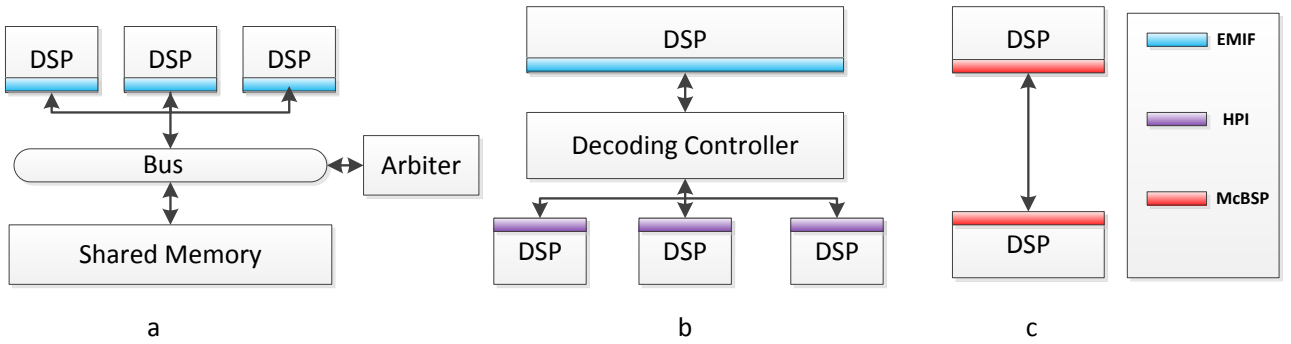


FIGURE 2 Topology of DSP interconnect

3 Multi-DSP simulation platform

Integrating multiple Instruction Set Simulations (ISSs) of DSP into a unified system simulation framework has several non-trivial challenges. In this section, we will introduce each of the major modules of the simulation system. First we add several components to the original open source cycle accurate ISS in order to meet the basic I/O requirements. As SystemC is selected as system model language, a SystemC wrapper is necessary for the conversion between read (write) requests and SystemC transactions. Then three special interfaces are provided for the interconnect of DSPs in accordance with current popular DSP interconnect structure. Finally, an AMBA Bus-Shared Memory Model used in our simulation system is introduced.

3.1 PROCESSING MODULE

As shown in Figure 3, we add three components (Interrupt Controller, Timer Manger and I/O Manager) to the original ISS in order to enable interconnection between the DSPs, which also ensures a full-fledged real-time operating system (RTOS) to run on it. The Interrupt Control component is used for the management of external interrupts, which are mainly from the I/O modules. The clock interrupt is triggered by the Timer, which provides support for a RTOS running on the simulator. The I/O Manager module is used for mapping the I/O address and providing a uniform I/O management interface.

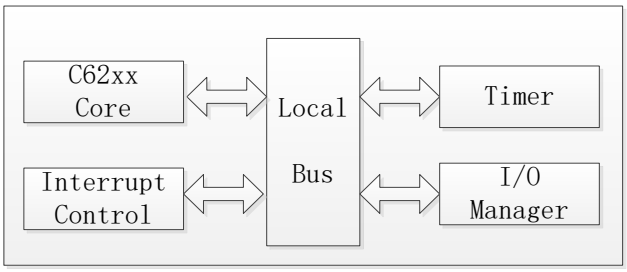


FIGURE 3 Processing module architecture

3.2 SYSTEMC WRAPPER

There are significant differences between the ISS and the interconnect structure, as the ISS is developed with the c program language while the interconnect structure consists of SystemC modules. An approach to make up the difference is to completely embed the ISS within the SystemC module. In other words, we design a class not only with the instruction set simulation function and but also with the function of transforming normal c function call request into the SystemC transaction level request.

As shown in Figure 4,we design two classes, namely the SystemC module *DSP_Wrapper* and the processing module *DSP_Core*. The *DSP_Wrapper* contains an instance of *DSP_Core* and launches the ISS simulation. The two member functions *(tran_acc_in and tran_acc_out)* are used for the synchronization with the environment (Bus, peripherals and other DSP Core). The member function, *specail_mem_access* in *DSP_Core* shows that access to specific memory (I/O ports or I/O control registers, etc.) can be detected. The access is suspended not until the *DSP_Wrapper* receives the corresponding response signal and then sets the member variable *s_mem_end* as *true*.

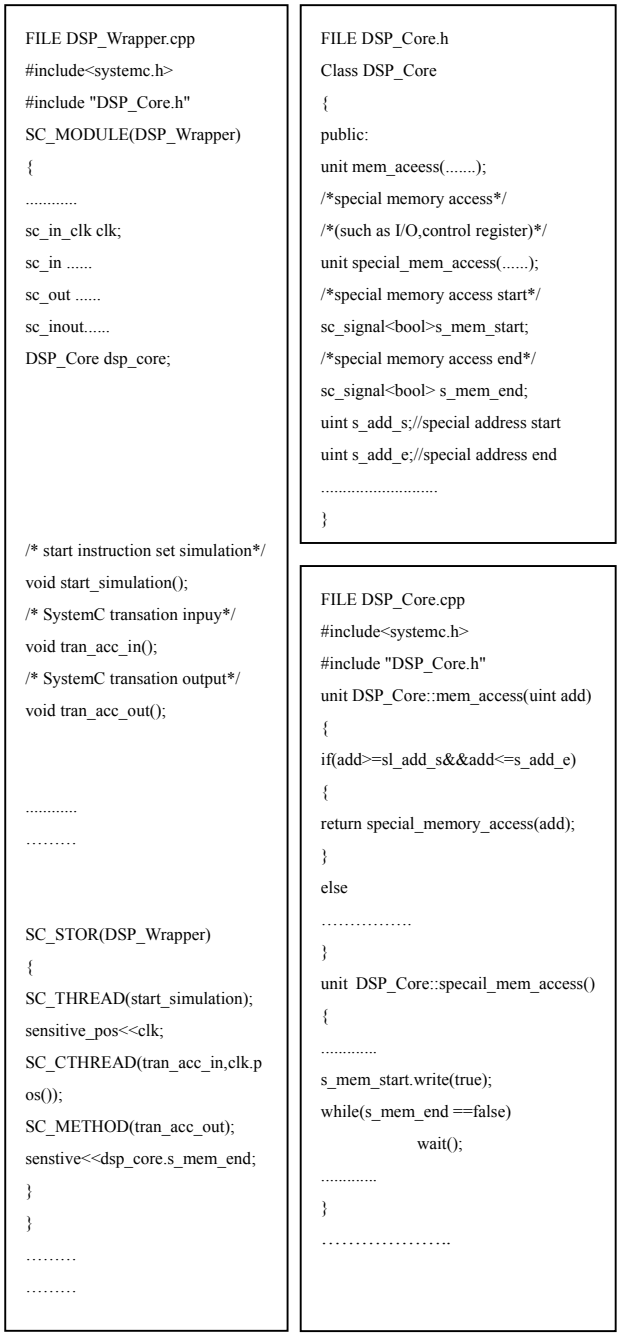


FIGURE 4 *DSP_wrapper* architecture

3.3 INTERFACE DESIGN

3.3.1 External memory interface

External memory interface (EMIF) is the only channel to access external memory in the C62x-series DSP and the transmission speed generally can reach 16 Gbit/s. The main pins and control registers are designed as shown in Tables 2 and 3. A typical access (read) to shared memory with EMIF is as follows:

- Set EM_Add as the corresponding address and EM_Read as *true*;
- Set the EM_Hold as *true*, then wait until EM_HoldA as *true*;
- Read the EM_Data value after one clock;
- Set the EM_Hold as *false* to release bus when the access finishes.

TABLE 2 Pins of external memory interface

Name	Data Type	Introduction
EM_Enable	sc_in<uint>	enable EMIF
EM_Add	sc_out<uint>	address pins
EM_Data	sc_inout<uint>	data pins
EM_Read	sc_in<bool>	indicate read
EM_Write	sc_in<bool>	indicate write
EM_Ready	sc_inout<bool>	indicate ready
EM_Hold	sc_out<bool>	bus request
EM_HoldA	sc_in<bool>	bus response
EM_Int	sc_out<bool>	interrupt
EM_Clk	sc_inout_clk	clock

TABLE 3 Control register of external memory interface

Name	Data Type	Introduction
GlbCtl	uint	global control register
SdCtl	uint	sdram control register
SdExt	uint	sdram external register

3.3.2 Host port interface

The host port interface (HPI) is a special parallel interface, existing in most of the TI DSP chip. The master-slave interconnect structure with HPI can significantly reduce the complexity of the system with no need for extra chips. The master-slave multi-DSP system is very popular in flight control system in which the reliability is the primary consideration [13]. The main pins and control registers are designed as shown in Table 4 and Table 5 respectively. A typical access to slave memory process is as follows:

- Set HPI_Cntl1 and HPI_Cntl2 as false, false;
- Write special value to the HPIC register to prepare for the access slave;
- Set HPI_Cntl1 and HPI_Cntl2 as false, true and write address to HPIA;
- Set HPI_Cntl1 and HPI_Cntl2 as true, false (or true, true) and then read data from HPID.
- In the previous step, if the values of HPI_Cntl1 and HPI_Cntl2 are set as true and false respectively, the value of HPIA will automatically increase one.

TABLE 4 Pins of host port interface

Name	Data Type	Introduction
HPI_Data	sc_inout<ushort>	data pins
HPI_Cntl1	sc_in<bool>	control pin1
HPI_Cntl2	sc_in<bool>	control pin2
HPI_Hwil	sc_in<bool>	indicate transform
HPI_Ready	sc_out<bool>	indicate ready
HPI_Int	sc_out<bool>	interrupt
HPI_Clk	sc_inout_clk	clock

TABLE 5 Control registers of host port interface

Name	Data Type	Introduction
HPIA	uint	HPI address register
HPIC	uint	HPI control register
HPID	uint	HPI data register

3.3.3 Multi-channel buffered serial port

The Multi-channel buffered serial port is one of the fundamental interfaces in the C62x series DSP. It is usually used for connecting a serial interface peripheral, such as serial AD and serial peripheral interface. It can also be used for the interconnection between the DSPs when there is only a small amount of data exchanged. The simple McBSP-to-McBSP structure can reduce the cost of system design significantly. Figure 6 and Figure 7 show the main pins and control registers of the multi-channel buffered serial port. A typical byte reception process is as follows:

- Set SPCR register as a special value to configure the receiving protocol;
- RSR gets a bit from the MS_DR pin every one clock until one byte transfer is completed;
- Check the received byte according the check code. If the byte is correct, then set the value of DRR as the received byte and set MS_CLR as true to generate an interrupt.

TABLE 6 Pins of multi-channel buffered serial port

Name	Data Type	Introduction
MS_DX	sc_out<bool >	send pin
MS_DR	sc_inout<bool>	receive pin
MS_CLX	sc_inout_clk	send clock
MS_CLR	sc_inout_clk	receive clock
MS_Xint	sc_out<bool>	send interrupt
MS_Rint	sc_out<bool>	receive interrupt

TABLE 7 Control register of multi-channel buffered serial port

Name	Data Type	Introduction
DRR	uchar	receive register
DXR	uchar	send register
RSR	ushort	receive shift register
XSR	ushort	send shift register
SPCR	uint	control register

3.4 AMBA BUS-SHARED MEMORY MODEL

The bus-shared memory model is currently a very popular method of multi-machine interconnect [14]. The masters communicate with each other through the shared memory. In our simulation system, the ISS (DSP core) is the master of the bus and a shared memory is the slave of the bus. In the following parts, the bus model and the shared memory will be introduced in details respectively.

3.4.1 AMBA Bus

The AMBA Bus is applied in our simulation platform as the AMBA is a widely used standard in system on a chip (SoC) [7]. The AMBA Bus contains two standards: an advanced high-performance system bus (AHB) and a peripheral bus (APB) for minimal power consumption and connection with low-performance peripherals. We have developed a SystemC module only for the former one, given the situation of a large number of data exchange between the DSPs. As each ISS (DSP Core) in the simulation system is in peer relationships. Each bus request from the master is given the same priority. A traditional arbitration strategy (round-robin policy) is implemented in our AMBA Bus model to realize load balancing.

A Bus transaction is triggered by a bus request signal when one master (DSP) wants to access the shared memory. The arbiter receives the request and then determines whether to authorize the request or put the request into the waiting queue according to the current state of the bus. Then the master waits until the bus ownership is granted by the arbiter. At the same time, the address and control lines are driven and the data bus ownership is also granted after one clock cycle. Last, the data transformation starts when a ready signal is asserted by the slave (shared memory), indicating all the preparations have been completed and the single data transformation can be completed after the next rising edge of the clock. Besides this single transfer, specified-length bursts and unspecified-length bursts are also supported in our designed the AHB protocol.

3.4.2 Shared memory

The Shared Memory is connected to the AMBA Bus as a slave. It consists of multiple instantiations of a basic SystemC memory module and each module space is 1MB. It communicates with the masters through the AMBA bus with a typical request-ready asynchronous protocol [7].The memory read process is as follows: the address lines are assigned and then the memory module checks whether the address is within the scope of the current address space or not. If the address is effective, then a ready signal is asserted by the memory module. After one clock cycle, the value of data address lines is assigned as the content of the corresponding address in memory by the memory module.

4 System expansion and performance optimization

4.1 SYSTEM EXPANSION

As shown in Figure 2, the interconnect structure between the DSPs is very flexible. In many complex scenarios, the interconnect structure is a hybrid structure which consists of two or more interconnect structures in Figure 2 [15]. Each topology involves various parameters, such as the number of DSPs, the number of shared memory, size of each shared memory, any two DSP connection mode, etc. The hybrid topology and configurable system parameters are able to meet the needs of a variety of Multi-DSP systems simulation.

4.1.1 Hybrid topology

Mixed DSP interconnect structure involves multiple external connection interfaces and their communication with the DSP Core. In section 3.3, three external connection interfaces are designed for the DSPs

interconnect. We just need to map the ports of the three interfaces into different memory address space and to ensure the three interface module can simultaneously keep pace with the DSP Core module. Then the hybrid DSP interconnect structure can be supported in our simulation platform.

4.1.2 System configuration

The system configuration describes the DSP interconnection topology. For an ordinary user, figurative descriptions can be understood easily. Therefore, a graphical configuration interface is a very good choice. Figure 4 shows a WYSIWYG DSP interconnection parameter configuration interface.

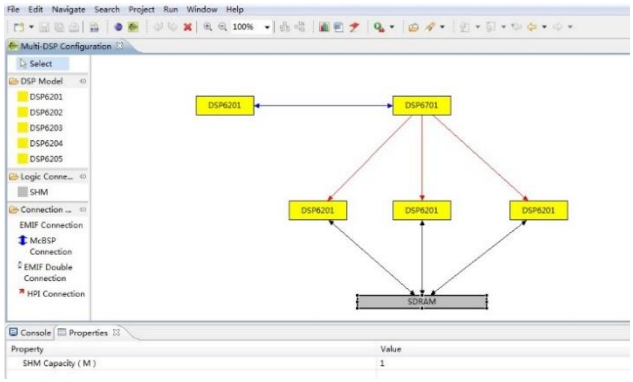


FIGURE 4 Visual configuration of simulation system

4.2 SIMULATION PERFORMANCE OPTIMIZATION

4.2.1 Performance Bottleneck Analysis

Table 5 shows the time cost of quick sort with 20000 random numbers in our simulation system. All the DSP are connected with shared memory as shown in Figure 2-a and a parallel algorithm is used in the multi-DSP system. The host machine is shown in section 5.1.

TABLE 8 Execution time of quick sort

Quick Sort	Time(ms)
single DSP	153824
two DSP	205234
four DSP	285234

As we can see that the execution time of quick sorting rapidly increases with the incremented number of DSPs. This indicates that although the host is a multi-core (eight core) system and supports multiple threads run in parallel, multi-DSP parallel processing does not increase but reduces the system performance instead. As each DSP core is designed independently in a SystemC thread of execution in our system, we suspect that these SystemC threads may not be executed in parallel and [3, 16] confirm our guess. The SystemC kernel adopts fiber mechanism and QuickThread to encapsulate SystemC threads in the Windows and Linux respectively. Therefore, the SystemC thread is not an OS thread but a lightweight thread. The switches of these lightweight threads do not take place in the OS kernel layer. Therefore, they have very high performance; however they are executed serially as these lightweight threads are equivalent to one OS thread. So multi-DSP parallel system cannot accelerate the sorting process. The synchronization between the DSP cores and peripheral modules can also significantly affect the performance of the multi-DSP system and the synchronization overhead will increase with the number of DSPs in our system. Therefore, the overhead of sorting increases with the number of the DSP in our system, rather than decrease.

4.2.2 Performance optimization implementation

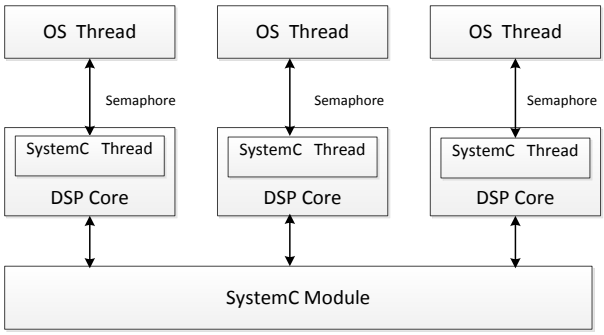


FIGURE 5 SystemC thread parallel optimization

SystemC is one of the most popular system-level modelling languages as it provides a simulation framework, which facilitates design and verification of SoC at different levels. However, the single threaded simulation kernel inherent to SystemC cannot meet requirements of multiple DSP parallel running. We present a method of thread agent to ensure multiple DSPs run in parallel on current popular multi-core machine to speed up the simulation of the whole system.

Figure 5 shows the method of thread agent to avoid SystemC threads' serial execution. Each DSP core creates an OS thread to simulate the instruction execution. As described in section 3, the ISS-SystemC framework is used in our system and the synchronization between DSPs is done through SystemC event mechanism. However, the synchronization is not in every clock cycle but only when the exchange of data between DSPs or access to the peripherals occurs. We create an OS thread in the SystemC thread to simulate the instruction execution .An OS semaphore is used for the synchronization between the OS thread and SystemC thread when the exchange of data between DSPs and access to the peripherals occurs. All OS Threads can run in parallel on the multi-core machine when there is no communication between the DSPs. Therefore, the simulation speed of the entire system could be improved greatly.

5 Experiment verification

5.1 EXPERIMENT ENVIRONMENT

1) Host Machine
- CPU: FX-8350 (4.0GHz), eight cores CPU
- RAM: 8GB (DDR1600)
- Hard Disk: 1TB
- OS: Windows 7 ultimate

2) Development Environment
- Test case compiler: Code Composer Studio 3.3
- Simulation system Compiler: Microsoft Visual Studio 2010
- SystemC version: 2.2.0

5.2 TEST CASE

We make changes on some benchmarks [17] provided by TI, to ensure the benchmarks can run on the multi-DSP parallel system. The modified benchmarks contain Fast Fourier Transformation (FFT), Matrix Multiplication and Quick Sort.

5.2.1 Fast Fourier transformation

Fourier transform, a basic digital signal processing operations, is widely used for presentation and analysis of discrete time-domain signal. Define a discrete finite time sequence $x(n), 0 \le n \le N$, and the discrete Fourier transform is:

$$X(k) = \sum_{0}^{N-1} x(n) w_N^{nk}, k = 0,1,..., N-1, w_N = e^{-j\frac{2\pi}{N}}.$$

The algorithm complexity of commonly used Discrete Fourier Transformation (DFT) is $O(n^2)$ .In our system, Fast Fourier Transformation (FFT) is used in the multi-DSP simulation system. The DFT sequence is divided into shorter DFT sequences and each shorter DFT sequence is dealt in one DSP. This allows parallel processing of Fourier transform and the algorithm complexity reduces to $O(n\log_m^n)$.

5.2.2 Matrix multiplication

Matrix multiplication is one of the DSP regular processing operations. Define $C = A \times B$, A and B are N-dimensional matrix. Then $C_{i,j} = \sum_{k=1}^{M} A_{ik} B_{kj} (1 \le i \le M, 1 \le j \le M)$ and the algorithm complexity is $O(n^3)$ .In our system, we have adopted Cannon algorithm of matrix grid division. Assumptions with m^2 DSPs in parallel system, each DSP is assigned to $\frac{n}{m}$ of the data for dealing. The processing speed of the whole system can be improved obviously and the algorithm complexity is about $o(n^2)/m$.

5.2.3 Quick sort

Quick sort, as a kind of efficient sorting method, is often used to deal with large-scale data sorting. Define that the amount of data to be sorted is n and the number of DSP is m. The algorithm complexity of the sorting data for each DSP is $\frac{n}{m}\log_2^{\frac{n}{m}}$ and the algorithm complexity of all the data is $\frac{n}{m}\log_2^{\frac{n}{m}} + n\log_2^m$.

5.3 EXPERIMENT RESULTS AND ANALYSIS

Figure 6-8 show relative execution time of three test cases (FFT, Matrix Multiplication and Quick Sort). Each test case runs on simulation systems with three kinds of topology and Table 9 shows the characteristics of these three topologies. SHM and U-SHM indicate the optimized system and the un-optimized system respectively. Finally we discuss the communication overhead and parallel acceleration according to the experiments results.

TABLE 9 Three kinds of topologies

Interconnection Method	Characteristics
SHM	The entire DSPs are connected with one Shared Memory as shown in Figure 2-a
HPI	The entire slave DSPs are connected to Host DSP with HPI as shown in Figure 2-b
Mixed	All the DSPs are connected with mixed interconnection methods as shown in Figure 4

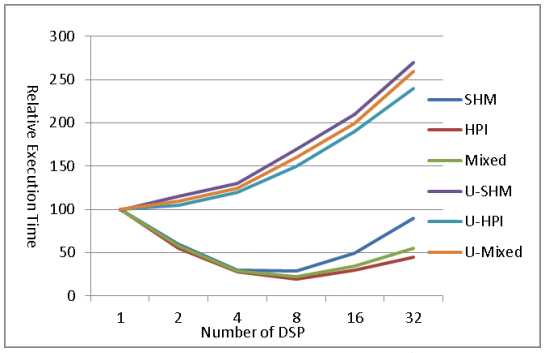


FIGURE 6 Relative execution time of FFT

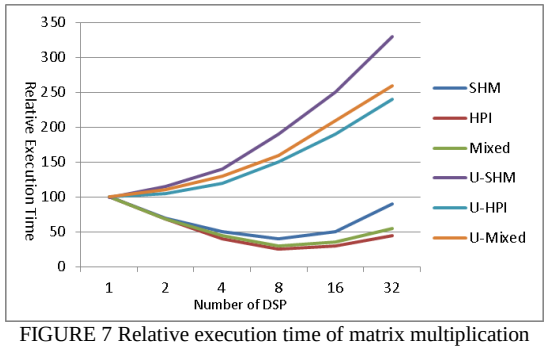


FIGURE 7 Relative execution time of matrix multiplication

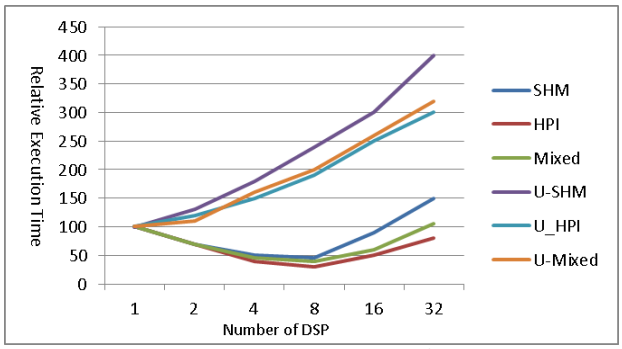


FIGURE 8 Relative execution time of quick sort

5.3.1 Communication overhead analysis

As we can see from section 4.2, the synchronization overhead will increase with the increase of the number of DSP in our system. The results of the three test cases in no performance optimization condition exactly reflect this characteristic. And the communication overhead varies with different interconnect structures. Figure 6-8 show the relative execution time of simulation system with shared memory structure is longer than other simulation systems when the number of DSP reaches four. And the trend is more obvious when the number of DSP exceeds four. This is because the multi-DSP system shared memory structure is restricted to access the only one shared memory. With the increasing of the number of DSP, the bus request time will be longer. The scales of communication are almost the same when the number of DSP is the same in our multi-DSP systems. Therefore, the communication overhead of the system with shared memory structure is greater than other systems.

5.3.2 Multiple DSP parallel acceleration

As we can see from Figure 6-8, the relative execution time of simulation system which performance optimization is used in significantly gets shorter as the number of DSP increases from one to eight. This indicates that multi-DSP simulation system can speed up the system of processing data when the number of DSP is less than the cores of machine which simulation system runs on. This is because each DSP binds to one SystemC thread and the parallel optimization ensures all the SystemC threads take full advantage of threads provided by the operating system to run in parallel. The amount of data processing is fixed, the number of DSP increasing results in an increase in the overhead of communication between DSPs. So the relative execution time does not reduce exponentially with doubling the number of DSP. When the number of DSP increases to eight or more, the relative execution time does not shorten but extend with the increase of the number of DSP. This is because the host machine has only eight cores, which leads to a maximum of eight OS threads running in parallel on the operating system. Although the number of DSP exceeds eight, there are only eight DSPs parallel running on the operating system at the same time .And additional communication will lead to longer processing time.

The above analysis shows communication overhead varies with different topologies and our multi-DSP simulation system can obliviously accelerate the processing of large amounts of data. Creating OS thread in the SystemC thread to complete the time-consuming processing can significantly improve the efficiency of serial running of SystemC threads.

6 Conclusion and future work

We provide a flexible and scalable multi-DSP model for C62x-series DSP at cycle-accurate level of abstractions. An ISS-SystemC framework which provides a hardware and software collaborative development environment is used in our system. Three kinds of interconnection interfaces are designed for a flexible interconnection structure. A kind of optimization for SystemC threads is also used to accelerate the speed of the system simulation and the experiment results verify that the simulation platform is very efficient. Future extensions of our work include two aspects:

1) The ISS can be designed to be more perfect and close to the real DSP. For example, direct memory access (DMA) and cache modules need to be added.

2) Each SystemC thread creates an OS thread to simulate each DSP's instruction execution, which ensures that multiple DSPs run in parallel on multicore machines in situation of no communication between multiple DSPs. However, the simulation of each DSP's communication is implemented by one SystemC thread, which results serial simulation of all the DSPs' communications. Providing parallelizing SystemC kernel will be an efficient choice to improve the performance of the simulation platform.

Acknowledgments

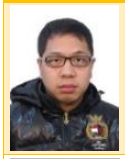
This work has been supported by China National Natural Science Foundation (No.51175462) and National Defense Research Projects.

References

[1] Huang F, Qiao C, Wang Y, Wang G 2007 *Computer Engineering* **33**(23) 200-33 *(in Chinese)*
[2] Braun G, Nohl A, Hoffmann A, Schliebusch O, Leupers R, Meyr H 2004 *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **23**(12) 1625-39
[3] Pasricha S 2002 Transaction level modeling of SoC with SystemC 2.0 *Synopsys Users Group Conference SNUG 2002* India 2002
[4] Cuppu V 1999 Cycle Accurate Simulator for TMS320C62x, 8 way VLIW DSP Processor

http://www.cs.cmu.edu/afs/cs/academic/class/15745-s07/www/c6xref/c6xsim.pdf
[5] Buchmann R, Greiner A 2007 A Fully Static Scheduling Approach for Fast Cycle Accurate SystemC Simulation of MPSoCs *Proc. Int. Conf. Microelectron (ICM 2007)* Cairo Egypt 101-4
[6] Chao M C, Yeh T C, Tseng G F 2011 *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **30**(4) 593-606
[7] Benini L, Bertozzi D, Bogliolo A, Menichelli F, Olivieri M *Journal of VLSI Signal Process* **41**(2) 169-82
[8] Dales M SWARM-Software arm: http://www.cl.cam.ac.uk/˜mwd24/phd/swarm.html.
[9] Boukhechem S, Bourennane E B 2008 TLM Platform Based On SystemC For STARSoC Design Space Exploration *Proc. NASA/ESA Conf* AHS 22-5
[10] Boyer F, Yang L, Aboulhamid E, Charest L, Nicolescu G 2003 Multiple Simplescalar Processors with Introspection under SystemC *Proceedings of the 46th IEEE International Midwest Symposium on Circuits and Systems* 1400-4
[11] Fummi F, Perbellini G, Loghi M, Poncino M 2006 ISS-Centric Modular HW/SW Co-Simulation *GLSVLSI'06: Proceedings of the 16th ACM Great Lakes symposium on VLSI* 31-6
[12] TMS320C6000 Instruction Set Simulator Technical Reference Manual: http://www.ti.com/lit/ug/spru600i/spru600i.pdf
[13] Bao C, Wang L H, Zhang L, Zhang S N 2010 *Science Technology and Engineering* **10**(33) 8287-92 *(in Chinese)*
[14] Zou Z Q 2006 *Computer Engineering* **32**(16) 232-5 *(in Chinese)*
[15] Xu T F, Zhang B, Ni G Q 2005 *Transactions of Beijing Institute of Technology* **25**(11) 990-2 *(in Chinese)*
[16] Ezudheen P, Chandran P, Chandra J, Simon P B, Ravi D 2009 Parallelizing SystemC Kernel for Fast Hardware Simulation on SMP Machines *PADS' 09: Proceedings of the 2009 ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation* 80-7 Washington DC USA
[17] Texas Instruments benchmarks: http://www.ti.com/lsds/ti/dsp/c6000_dsp/c674x/benchmarks.page

Authors



Zheng-Mao Zhou, born in 1988, Hubei, China

Current position, grades: doctoral candidate of Zhejiang University.
University studies: B.S. degree in Computer Science from Huazhong University Science and Technology in 2010.
Scientific interest: digital signal processing, real-time operating system, software reliability.
Publications: 2.

Shun-Hong Zhong, born in 1986, Zhejiang, China

University studies: B.S. degree, M.S. degrees both in Computer Science from Zhejiang University, China, in 2009, 2012, respectively.
Scientific interest: digital signal processing.

Ming Cai, born in 1974, Zhejiang, China

Current position, grades: Associate professor at Zhejiang University.
University studies: M.S. and Ph.D. degrees both in Computer Science at Zhejiang University, China, in 1999 and 2002, respectively.
Scientific interest: digital signal processing, real-time operating system, web service, manufacturing resource discovery.
Publications: 15.