High performance IPC hardware accelerator and communication network for MPSoCs

Moonmo Koo and Soo-Ik Chae
School of Electrical Engineering and Computer Sciences
Seoul National University, Seoul, Korea
{jeffley, chae}@sdgroup.snu.ac.kr

Abstract — **In this paper, we explain a configurable IPC module for multimedia MPSoCs, which was implemented in a MPW chip that include three ARM7 CPU cores. According to the test results for an M-JPEG and a H.264 decoder, its IPC synchronization overheads are not more than 1% when the synchronization period is about 5000 cycles.**

MPSoC; IPC; synchronization; multimedia; H.264 (key words)

I. Introduction

MPSoCs, which employs multiple processor cores, are now essential for implementing multimedia applications, especially high-definition TV, digital camcorders, 3D game players, and smart phone, which support complex functions and various multimedia standards [1]. Architecture for the MPSoC need to be optimized for such high-performance applications that require large data flows for inter-processor communication.

Inter-communication interfaces to integrate heterogeneous component were defined in [2], which did not address their issues for high performance applications but implicitly covered channel reconfigurability. Dynamic reconfiguration of the topology of tasks and channels was described in [3] where the number of channels and each functions could not be changed. To cover a wide range of applications, however, the IPC architecture should be reconfigurable while dedicated hardware for channels should be provided. The purpose of this paper is to report the performance evaluation of a reconfigurable IPC block, which was proposed in [4]. The IPC block was designed to reduce synchronization overheads and implemented with three ARM7 cores into silicon through a MPW project.

II. Architecture

Simplified architecture for the IPC test chip is shown in Fig. 1, which includes three ARM7 cores connected through a JTAG chain and hardware blocks, which are connected to a AHB bus to accelerate inter-processor communication (IPC).

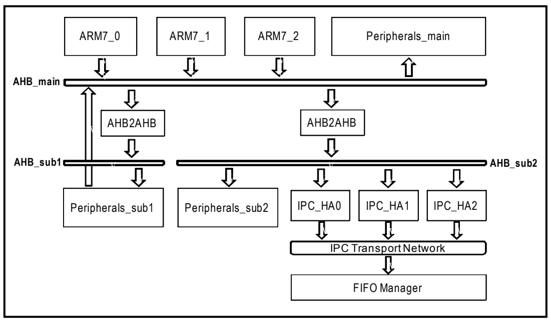

Figure 1. The simplified architecture diagram of our chip

A. IPC synchronization hardware agents

For *x*=1, 2, and 3, a block, IPC_HA*x,* is an IPC synchronization hardware agent associated with ARM7_*x*. Each IPC synchronization agent has the following three features to reduce the synchronization cost.

- All channels accessed at some time point are grouped and synchronized at a specific time. In the H.264 decoder example, the reduction of synchronization time is 58.5% with this feature.
- As computation and synchronization can be executed concurrently, the latency of synchronization is shortened.
- Each IPC synchronization agent includes a thread scheduler, which can monitor its related channels through the IPC transport network. In the M-JPEG example, the total execution cycles is reduced by 9.8% with this scheduler.

B. FIFO Mananger and IPC trnasport network

The FIFO manager connected to the IPC transport network controls the number of channels and functions of each channel. The IPC transport network connects all IPC synchronization agents to the FIFO manager. The number of the channels that can be supported with this architecture can be increased easily, which is only limited by the bus bandwidth of the IPC transport network

III. Chip Design & Verification

The summary of our test chip is shown in Table I. Although the operating frequency for the synthesis result is 125MHz, we achieved only 71.4MHz after P&R because our effort for P&R was limited.

TABLE I. The Specification of Our test chip

Tech.	Freq. after synth.	Freq. after P&R	Toal Area	IPC HW Cost	On-chip SRAM
0.18 ㎛ 1P6M	125MHz	71.4MHz	892.5 Kgates	87.8 Kgates	57.5KB (single-port)

Fig. 2 shows a verification board for the test chip. Because three RS-232C interfaces are attached on the board, the executing status for each CPU can be reported through each serial interface. Moreover, source-level debugging can be performed using ARM Multi-ICE. Even though all CPUs cannot be stopped or started simultaneously, each CPU can be probed by connecting it to the AxD debugger.

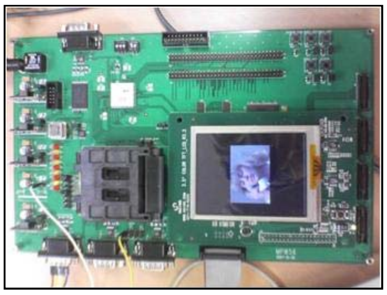

Figure 2. Verification board for the test chip

IV. Experimental Result

To test the chip, we used a software H.264 decoder that consists of 9 threads. In the decoder, communication among the threads is based on a 4x4 block and their average synchronization period is 5919 cycles. In the experiment we employed an ARM7-compatible RISC core with multiple register sets so as to measure the synchronization cost easily by eliminating the overhead of thread context switching. And we used Foreman QCIF 100 frames with GOP of IPP (N=10, without B picture) as a test image sequence. The H.264 decoder shows the performance of 5 frames/sec if it is executed on a single RISC core at the operating frequency of 200MHz. Although it does not provide high performance, it is a good example to test the case with high synchronization rate because its synchronization period is relatively short. The execution of the decoder is summarized in Table II.

TABLE II. The H.264 decoder execution results

(A) 1 thread on 1 CPU	(B) Ver. 1: Multiple threads on 1 CPU	(C) Ver. 2: Multiple threads on 1 CPU	(D) Multiple threads on 3 CPUs
3988674989 cycles	4167357160 cycles	4024592873 cycles	2054190319 cycles

In Table II, the IPC overhead is not included in the result of experiment (A) because it has only a single thread. We assume that channel data such as FIFO element indices and message pointers need to be read from IPC hardware agents in experiment (B) while not in experiment (C). The communication overheads of experiments (B) and (C) are 4.29% and 0.89% of the total execution time, respectively. Consequently, we should introduce an efficient method for transferring channel messages to reduce the difference, which is 3.4%.

In the H.264 decoder, we mapped inter-prediction and intra-prediction to one CPU, which occupies 40.3% of the computation load. Because it is the longest critical path, the performance enhancement gain we can obtain by using multiple CPUs is bounded by 2.48 times. According to the result of experiment in (D), the performance gain we obtained by using three CPUs is 1.94 times. The main cause of lower performance gain is due to data-dependency among H.264 threads because the communication overhead is 4.29% as described in the above. Obviously, more performance gain can be obtained if the decoder is partitioned into finer-grain threads and they are evenly distributed to all CPUs.

V. Conclusion & Future work

In this paper, we explained configurable IPC synchronization architecture for multimedia MPSoCs. We verified it with a MPW test chip including three ARM7 cores. It can be applied for multimedia applications such as M-JPEG and H.264 decoders where synchronization period is relatively short. A future work is to increase the performance of the H.264 decoder by optimizing its thread partitioning. Furthermore, new features should be added to the IPC module for more frequent synchronization requests.

Acknowledgment

This work was supported by the IC Design Education Center (IDEC) in KAIST, and the Seoul R&BD Program.

References

[1] Grant Martin, “Overview of the MPSoC design challenge,” Proc. of 43rd Design Automation Conference(DAC), San Francisco, July 2006.

[2] Peter van der Wolf, Erwin de Kock, Tomas Henriksson, Wido Kruijtzer, Gerber Essink, “Design and programming of embedded multiprocessors: an interface-centric approach,” CODES+ISSS, Stockholm, Sweden, Sep. 2004

[3] Martijn Rutten, Evert-Jan Pol, Jos van Eijdhoven, Karel Walters, Gerben Essink, “Dynamic reconfiguration of streaming graphs on a heterogeneous multiprocessor architecture,” SPIE Electronics Imaging: Embedded processors Multimedia and Communications II, vol. 5683, San Jose, USA, Jan 2005

[4] Moonmo Koo, Soo-Ik Chae, “Hardware implementation of inter-processor communication in MPSoCs for multimedia applications,” International Technical Conference on Circuits/Systems, Computers and Communications, pp. 775–776, July 2007.