RC24643 (C0809-009) September 19, 2008
Electrical Engineering

IBM Research Report

SPU Based Network Module for Software Radio System on Cell Multicore Platform

Jianwen Chen
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, 100094
P.R.China

Shuwei Bai, Qingguo Zhou
Distributed and Embedded System Lab
Eng. Res. Ctr. of Open Source Software and Real-time Syst. Ministry of Education
Lanzhou University
Lan Zhou
P. R. China

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

SPU based Network Module for Software Radio System on Cell Multicore Platform

Jianwen Chen[1], Shuwei Bai[2], Qingguo Zhou[3]

Abstract — *Wireless baseband processing is characterized by high computation complexity and high data throughput, which is regarded as the most challenging issue for software radio (SR) systems. Recently, with the rapid development of multicore technology, the multicore platforms can provide high computation capacity and high data throughput, which suit the SR systems well. We have presented a SR system on the Cell multicore platform. However, it is observed that the data communication between the baseband processing module and the RF module consumes a mass of system resources. The data I/O communication becomes the bottleneck for the SR system on multicore platform. To resolve this problem, in this paper, an efficient SPU based network module is presented. With the proposed module, all the network packets parsing and packaging is handled by the SPU, the PPU resources are released and the system performance will be improved greatly*[1].

Index Terms — Multicore Architecture, Software Radio, Cell, I/O communication

I. INTRODUCTION

Wireless broadband data communication is experiencing a rapid expansion. In the last few decades, many wireless broadband data communication techniques are proposed and widely used. To achieve a broadband wireless communication system with high mobility requirement, sophisticate baseband signal processing techniques, such as orthogonal frequency division multiple access (OFDMA), multi-input multi-output (MIMO) and space-time block coding (STBC) are proposed in the transceiver design for the mobile version of WiMAX [1] (802.16e). Following the proposals in this standard, two critical issues should be considered in the system design: complexity and adaptability. Firstly, when the requirement of data rate is increased, the baseband signal processing architecture of the transceivers becomes more and more complicated. This causes a lot of implementation challenges due to the complexity, especially for the base station (BS) side, since the BS has to handle synchronization and detection for all users. Secondly, the wireless standards are evolving and new algorithms are proposed occasionally. That means the designed system should have a good adaptability to catch up with the revision of the standards and new algorithms. However, it is very difficult to achieve this by using application-specific integrated circuit (ASIC), which is the current design methodology.

In order to meet both the performance and the adaptability requirements, software radio (SR) [2] is introduced to fixed functional hardware based systems. The programmable devices, such as DSP or general purpose processor, are used to accommodate various standards and protocols by updating the software. Generally, most of the complex algorithms such as synchronization, channel estimation and channel decoding are handled in physical layer, which is the most computation intensive layer in the wireless systems.

To mitigate the implementation challenge caused by high computation complexity of physical layer, novel hardware is needed. Compared with the single-core processor, the multicore chips do not run that fast, but they promise a higher overall performance by handling more work in parallel. Thus the multicore platforms become more prevalent in vendor's solutions. Many system and semiconductor companies are developing platforms involving several cores (MCU, DSP, IP, and etc) on a single chip. STI Cell [3] is the latest state-of-the-art multicore processor designed by the joint adventure of Sony, Toshiba and IBM (STI). With the support of SIMD (Single Instruction Multiple Data) instructions and the powerful synergistic processor elements (SPEs), Cell processor can provide significantly high computation capacity, which is appropriate for SR systems. Based on the concept of software radio, we have tried to implement a WiMAX SR baseband system on the Cell processor [4]. The block diagram of the BS physical layer baseband transceiver (OFDMA mode) is illustrated in Fig.1.

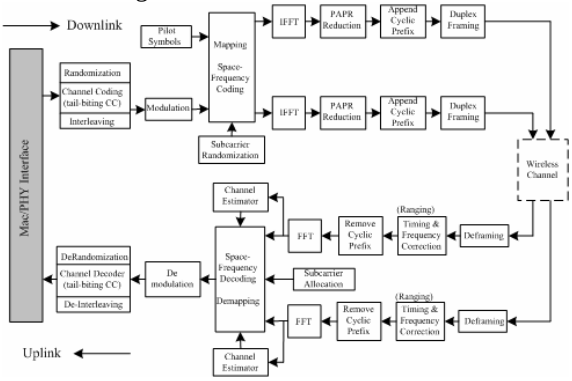


Fig. 1 WiMAX Physical Layer System Structure

However, for Software Radio system, the data communication between the Baseband processing board and the RF module is characterized by high throughput. The

[1] Jianwen Chen, IBM China Research Laboratory, Beijing, China, jianwenc@cn.ibm.com

[2]. Shuwei Bai, Distributed & Embedded System Lab (DSLab), Engineering Research Center of Open Source Software and Real-time Systems Ministry of Education,Lanzhou Univ., Lan Zhou, China, baishuwei@dslab.lzu.edu.cn

[3]. Qingguo Zhou, Distributed & Embedded System Lab (DSLab), Engineering Research Center of Open Source Software and Real-time Systems Ministry of Education, Lanzhou Univ., Lan Zhou, China, zhouqg@lzu.edu.cn

system needs to ensure that the I/O interfaces can support the required throughputs. For example, in the WiMAX PHY system with 20Mbps throughput for both uplink and downlink, if an 8 bit DAC is used, the output of downlink is about 658Mbps. For 3 sectors, the overall downlink throughput is about 1.975Gbps. On the Cell platform, the PPU is a general purpose processor and is not efficient on the network protocols parsing and packets handling. It is observed that in the implemented baseband system, the I/O data communication between the baseband processing module and the RF module consumes a mass of system resources for receiving, parsing and sending packets and it becomes the bottleneck for the SR system on Cell multicore platform.

In this paper, we present an efficient SPU based network module on the Cell platform. With this module, all the network packets parsing and packaging is handled by the SPU, the PPU resources are released and the system performance will be improved greatly.

The rest of this paper is organized as follows. Section II will briefly describe the architecture of WiMAX BS physical layer and the Cell multicore platform. Section III will introduce the proposed SPU network module. The network processing module design details are all discussed in this section. In Section IV, the system performance and result analysis are provided. Section V presents the conclusions and the future work.

II. System Architecture

Cell Broadband Engine (CBE) is a heterogeneous multicore microprocessor with 1 Power Processor Element (PPE) and 8 Synergistic Processor Elements (SPEs) [3]. PPE and SPE can work at frequency up to 3.2GHz. The PPE is a general purpose processor and the SPE is a special purpose processor which consists of the Synergistic Processor Unit (SPU) with 256 Kbyte Local Store (LS) and the Memory Flow Controller (MFC). The SPU and the MFC can work in parallel. Additionally, SPE supports a variety of SIMD instructions, high efficient (128-bit) memory access and high bandwidth I/O interface. The architecture block diagram of Cell processor is illustrated in Fig.2. In the proposed system, Cell blade server QS21 which consists of 2 Cell processors is used. 16 SPEs and 2 PPEs can be used to handle all the computation tasks.

The block diagram of the proposed BS baseband transceiver (OFDMA mode) is illustrated in Fig.1. These modules are the essential components for the physical layer of the WiMAX communication systems. All these modules are implemented on Cell SPEs. 2 SPEs are used to support 1 sector downlink data processing and 3 SPEs are used for 1 sector uplink data processing. The system framework is illustrated in Fig.3. The PPU will handle all the synchronization tasks and I/O data communication tasks. Experimental results show that the I/O data communication tasks consume almost 95% PPU resources and the system performance can not meet the required throughput. In the next session, we propose a novel network module on SPU, all the I/O data communication tasks can be allocated on this module without the interference of PPU. The high throughput I/O data communication will not influence the whole system performance.

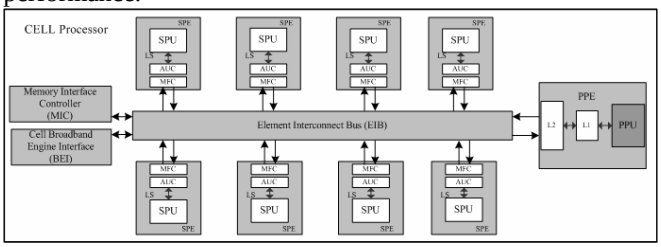


Fig.2 Block Diagram of Cell Processor

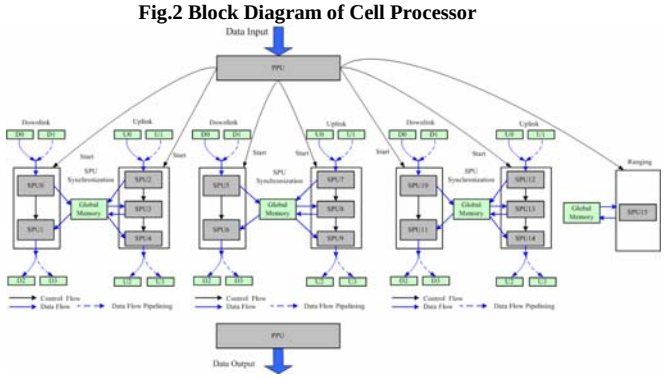


Fig.3 Framework for WiMAX physical layer on Cell Platform

III. Network Module Design for Cell Platform

As described above, for Cell platform, although the computation capacity is sufficient for the SR applications, the network communication is not efficient to meet the huge throughput requirements. In the network module prototype, to improve the performance, the network operations are divided into two parts. One part includes data encoding and package sending functions and the other one is composed of package receiving and data decoding functions. Two SPE are used to handle the receiving and sending packets respectively. Moreover, some special considerations should be taken into account in the network module design.

In the network communication, the data block size is about 1Kbytes, or smaller. SPE will parse packages of different size frequently. Worse still, many branches are used in the package parsing components. However, SPE is an in-order processor element, and SPE issues all instructions in program order. If there is dependency between two adjacent instructions, the later one has to wait to be issued until the former one completes. And this could lead to a huge performance loss. In addition, correctly predicted branches execute in one cycle, but a miss predicted branch incurs a penalty of approximately 18-19 cycles. Considering the typical SPU instruction latency of two to seven cycles, mispredicted branches can seriously degrade the packages parsing performance. So in the parsing components of the prototype, three methods are adopted to eliminate them to reduce the impact of branches, inlining, unrolling and prediction [5].

Inlining and unrolling both will increase the size of basic

blocks. In our experiment, the total size of network protocol stack and the network driver used in some embedded system is less than 15Kbytes. And in the SPE, the local store size is about 256Kbytes, which is big enough to run the network card deriver and protocol stack. SPU offers the select-bits instruction which can be used to eliminating branches for simple control –flow statements, such as if-then-else constructs. In the system, a lot of intrinsic instructions also are used to eliminate conditional branches, such as spu_cmpgt, spu_add, and spu_sel.

In the conventional network communication architecture, CPU should do all the work to handle the network packages, writing data to socket, moving data from user space to the kernel space by calling system call, encoding and dividing the data packages in the different protocol levels, and writing the data to the cable. The conventional network communication architecture is showed in Fig.4. All the tasks are executed by the same CPU, the network communication tasks and the intensive computation tasks will interact in a very complex way. In the single-core system, as the limitation of CPU resource, the problem cannot be resolved. But on the asymmetric multi-processor (AMP) platform, some special components can be moved to single core, such as network communication, which will improve the performance greatly. Our work focuses on Cell, the typical AMP platform. The design of the network communication system based on Cell should consider the system architecture. Fig.5 illustrates the network communication high-level architecture based on the Cell system.

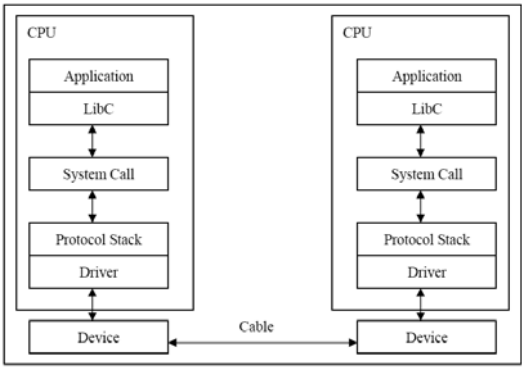


Fig.4 Conventional Communication Architecture

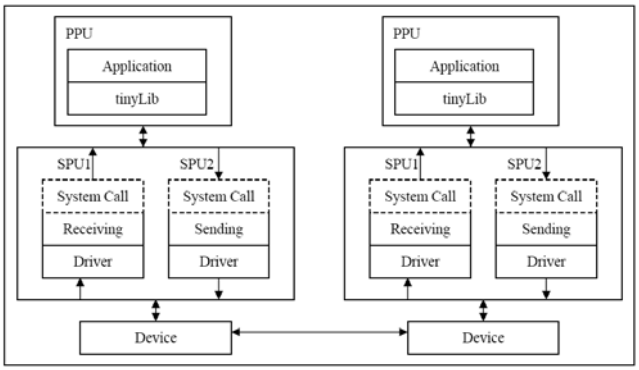


Fig.5 Communication System Architecture based on Cell

From Fig.5, it is noticed that we have modified three parts on the general platform. 1) The libc is replaced by the tinylib. As we know, the libc is offered by the Linux (the research platform is Linux), and it is located between user application and system call. In the conventional architecture, the user application should call the routines supported by libc to access the system call, and then entry the kernel space. In the new architecture, the libc is replaced by the tinyLib for the communication applications. The tinylib encapsules some low level routines which are used to access SPU. And some routines used to allocate memory are encapsuled in the tinylib too. The tinylib can offer a serial interfaces to the application programmers. Through those interfaces, the application sends or writes data to or from the protocol stack transparently. 2) The tinylib can access protocol stack directly. In the conventional communication framework, the system calls must be called. In the new framework, tinylib can communicate with the protocol direct. And some function offered by the system call is moved to SPU. 3) The third change is the most important. The protocol stack and the device driver is moved into SPU. In the communication process, data encoding and decoding is the most time-consuming job. Improve the performance of component is the key to improve the communication performance.

In the new architecture, the PPU is used for controlling and synchronizing of multiple cores. The data communication task is assigned to one SPU. This architecture does not only reduce the workloads of PPU, but also improve the network data communication throughput.

IV. PERFORMANCE AND ANALYSIS

We implement the network communication module on the Cell multicore platform with the proposed architecture. In the implemented prototype, a simple communication mechanism is used for convenience.

For the communication protocol, the package head consists of data size, frame number, CRC check parity, and destination port as depicts in Fig.6. The protocol adopts the static package size, which is 512B length. It is obvious that the data size is 480Bytes, which means, if the data size is larger than the length, it will be partitioned by the protocol stack. And if the data size is smaller than 480B, the padding segment data will be added.

32bits	16bits	16bits
CRC check parity	Data size	
Frame number	Port	Empty
Data Segment: 512B – 32B = 480B		

Fig. 6 Simple Protocol used in the Prototype

- Check parity: 32bits, the crc-32[6] value of the data segment;

- Data size; 32 bits, the size of the data segment, and it likes too long to specify the data size
- Frame number: 32bits, the serial number of the frame and it's exclusive in the communication process.
- Port: 16bits, used to identify the user application.
- Empty: 16bits, the padding bits for padding.
- Data segment: 480Bytes, the valid data in the part.

The prototype system is running on Cell blade QS21. In order to find out which kind of environment will affect the data transmission speed between the I/O device and SPU, the benchmark program will run with different environment configurations. The benchmark will also test the maximum throughput that the SPU can reach to read/write data from/to network I/O port.

At the same time, on the Cell platform, all the SPUs share one EIB bus. To simulate the practical environment, numbers of SPU is used to access the EIB bus in a time-sharing way. These SPUs will occupy the EIB bus by moving data between local store and main memory from time to time. Theoretically, when the EIB bus is occupied by other SPUs, the communication bandwidth between the network device and the SPU based network module will be degraded. In our experiment, we use different numbers of SPU to simulate the processing tasks (PT) in a practical system. The performance is provided in Fig.7. It is observed that the throughput of the communication module is just slightly degraded with the increase of processing SPU tasks. It is mainly because that the shared EIB bus on Cell platform has a very high bandwidth which can even reach up to 204.8GB/s.

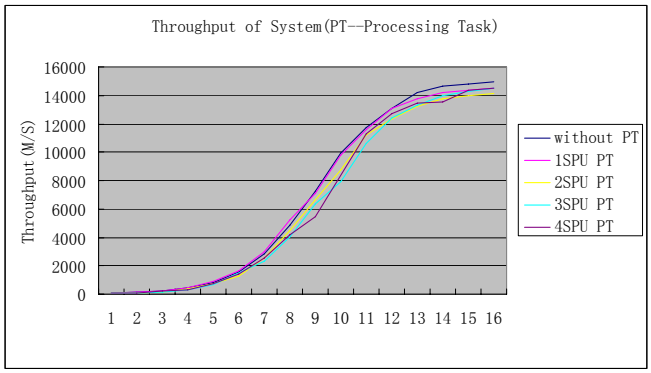
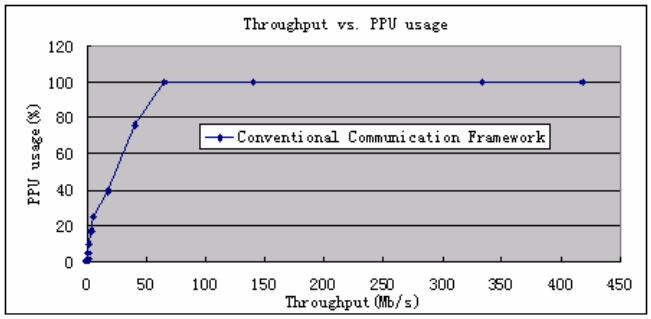


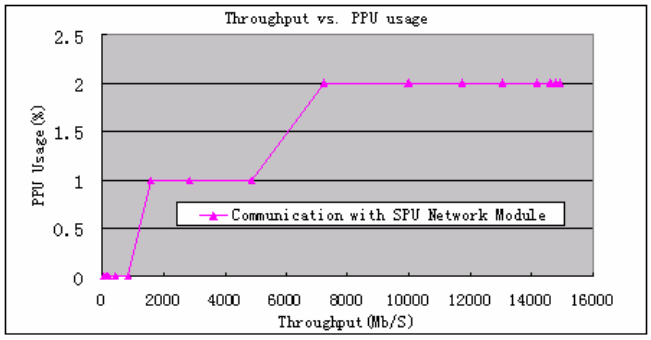
Fig 7 the bandwidth of communication system

It can be also noticed that the throughput of the network communication module can even reach 14GB/s for the worst case, which is sufficient for SR applications.

On the implemented SR system, the PPU is used for tasks synchronization and management. The PPU resource usage is very important for an efficient system. In Fig.8 (a), the PPU usage status for the conventional communication framework on Cell platform, in which all the network communication operations are handled by PPU, is provided. In Fig.8 (b), the PPU usage for the framework with the proposed SPU network module is provided. Although when the system throughput increases, the PPU usage of the two cases will also increase. It can be observed that in the system with the proposed SPU based network module, the PPU usage will just slightly increase with the throughput. It is because that almost all the receiving, parsing and sending operations are handled by the SPU independently.



(a)



(b)

Fig 8 PPU usages of different communication framework

V. CONCLUSIONS

This paper presents a novel SPU based communication module for SR physical layer on the heterogeneous Cell multicore platform. A prototype of the communication module has been implemented. The performance of the network I/O communication framework is provided. The results show that the PPU resources are released and the SR system performance can be improved greatly with the new network communication module.

REFERENCES

[1] IEEE Std 802.16e-2005. Part 16: Air Interface for Fixed, Mobile Broadband Wireless Access Systems Amendment2:Physical, Medium Access Control Layer for Combined Fixed, and Mobile Operation in Licensed Bands.", Feb. 2006.

[2] Joe Mitola. The software radio architecture. In IEEE Communications Magazine, May 1995, 26-38.

[3] IBM Cell Broadband Engine Resource Center website (http://www.128.ibm.com/developerworks/power/Cell/)

[4] Qing Wang, Da Fan, Yonghua Lin, Jianwen Chen, Zhenbo Zhu, "Design of BS transceiver for IEEE 802.16E OFDMA mode", Proc. IEEE ICASSP, Mar. 2008.

[5] Software Development Kit for Multicore Acceleration Version 3.0, Programming Tutorial, Book, 2005, 2007 @ Copyright International Business Machines Corporation, Sony Computer Entertainment Incorporated Toshiba Corporation

[6] Henriksson, T. Eriksson, H. Nordqvist, U. Larsson-Edefors, P. Liu, D, VLSI implementation of CRC-32 for 10 Gigabit Ethernet, Electronics, Circuits and Systems, 2001. ICECS 2001. The 8th IEEE International Conference, 2001, ISBN: 0-7803-7057-0