RC24823 (C0907-003) July 14, 2009
Electrical Engineering

IBM Research Report

Wireless Base Station Design on General Purpose Processor with Multicore Technology

Yonghua Lin, Qing Wang, Jianwen Chen, Lin Chen, Zhenbo Zhu
IBM Research Division
China Research Laboratory
Beijing, 100094
P.R.China

Wireless Base Station Design on General Purpose Processor with Multicore Technology

Yonghua Lin[1], *IEEE Member*, Qing Wang[1], Jianwen Chen[1], Lin Chen[1], Zhenbo Zhu[1]
[1]IBM China Research Lab, Beijing, China, 100094

Abstract— **Software radio (SWR) is one of the major goals for future wireless communication engineering. Driven by multicore technology, we can think about implementing wireless base station on multicore general purpose processor (GPP). IBM Cell processor and Intel Clovertown processor represent the state-of-the-art multicore processor of two different main streams, heterogeneous and homogenous multicore architectures. In this paper, using Worldwide Interoperability for Microwave Access (WiMAX) as example, we introduce the design of wireless base station over these two different multicore GPP platforms, with deep analysis of the software designs and workload mapping. Based on it, those key aspects of multicore architecture design, such as multicore, multiple issue, multithreading, and memory structure are thoroughly discussed, together with the performance impact from our experiment results and different program models. From this paper, people can know how to implement the SWR base station on multicore GPP with high performance, how the processor architecture impacts the performance, and what other factors related to system design need to be considered in future.**

I. Introduction

In recent years, to support lager system capacity, higher service data rates, larger coverage areas and higher mobility, many new wireless technologies have evolved and various wireless standards have been formed, including Wideband Code- Division Multiple Access (WCDMA), 802.16d/e (represented by WiMAX, i.e., Worldwide Interoperability for Microwave Access), Long Term Evolution (LTE), and LTE advanced. Most of the existing Global System for Mobile communication (GSM) operators are considering to upgrade the GSM networks into new generation mobile networks [1]. The differences among wireless standards mainly involve the wireless access network, especially the wireless base stations (BSs). As the main components of wireless access networks, base stations usually represent the largest monetary investment associated with a mobile network. Usually it exceeds 60% of total infrastructure investment. So, the cost of base stations when upgraded to different wireless standards will be very important for operators. How to provide low cost base station solution to support different wireless standards is also important for network equipment providers (NEPs).

Software Defined Radio Forum (SDRF) has defined five stages, hardware radio, software controlled radio, software defined radio (SDR), ideal software radio (SWR), and ultimate software radio [2]. SDRF defines an SDR as one that implements a specified range of capabilities through elements that are software-reconfigurable. If the functionalities of the radio can be totally redefined in software, this would be the ideal implementation of SWR [3]. Most of current base station designs are still stay in the SDR stage, where the combination of programmable hardwares such as field programmable gate arrays (FPGAs) or digital signal processors (DSPs) is the dominant approach. Though they can meet the processing and timing requirements of modern high-speed wireless protocols, there are two major issues. Firstly, the base station hardware platform based on the mixture of DSP processors, ASICs and FPGA will have less flexibility to be shared among different wireless standards [3]. It will increase the upgrade cost of mobile operators. Secondly, the programming of FPGAs and DSPs are difficult tasks. Developers have to learn how to program on each particular embedded architecture, and the productivity will be low. When the hardware platform is changed, the software implementation on FPGAs and DSPs will need lots of porting efforts.

With the rapid progress in multicore technology, general-purpose processor (GPP) can be considered to support the entire base band processing in wireless base station (BS). Nowadays, multicore, simultaneous multithreading (SMT) and (single instruction multiple data) SIMD processing are the three main technologies widely utilized in processor architecture to provide high parallel performance with lower power [4]. They have been used in a combinational way. For example, Intel Nehalem is the processor with 4 cores, two hardware threads per core, and 128-bit stream SIMD extension (SSE) instructions for vector execution. IBM's Cell processor has 9 cores and 128-bit SIMD engine on each synergistic processing element (SPE) cores [5]. With these parallel architecture designs, Cell processor can provide 204.8GFLOPs (Giga Floating Point) peak performance on single chip, where Nehalem can support similar level of peak performance as well. It gives the hope to industry that, modern multicore GPP may be able to support the wireless base station. If so, we can enter into the ideal SWR stage, where different wireless standards can be implemented in pure software without the tightly couple with hardware. When wireless technology being upgraded, no base band hardware platform need be replaced. With the rich software tools and more familiar development environment, the productivity of development teams can be improved a lot. It can significantly speed up the adoption of new wireless research fruits in the commercial mobile system.

However, before we can enter into this ideal stage, two questions need be answered. One is, how to implement the wireless base station software stack on these modern multicore GPP with high performance, especially for the heavy physical

(PHY) layer processing. The second question is, what will be the system requirements and challenges when build up a wireless base station system based on GPP platform. With limited space for this article, we will mainly focus on the first question in this paper, and some analysis for the second question will be added in the end of this paper to drive more discussion in future.

From 2007, we started the SWR design on IBM Cell, Intel Cloverntown and Nehalem platforms, and created the first WiMAX 802.16e base station prototype on these platforms [14] [15]. So, in this tutorial, we will introduce our software design on multicore GPP, and share our experience on multicore and multithreading architecture.

The rest of this artical is organized as follow. Section 2 introduces the WiMAX base station design on different multicore GPP platforms. We then analyze those architectural considerations in Section 3, especially to evaluate the performance improvement got from different kinds of CPU architecture. In Section 4, related system requirements are discussed for building up a wireless base station with multicore GPP platform. Finally, Section 5 describes related work and Section 6 provides conclusion.

II. WiMAX Base Station Design on Multicore Platform

A. PHY layer and MAC layer of 802.16e

In wireless base station, the two major layers defined in standards are PHY layer and MAC layer. PHY layer is the lowest layer in the seven-layer open systems interconnection (OSI) model. In wireless system, PHY layer usually will include channel coding, modulation and multiplexing in the transmitter side, and synchronization, channel estimation, equalization, demodulation and channel decoder in the receiver side. So, PHY layer can be regarded as the stream processing with a series of signal processing components pipelined with each other. MAC layer is the second layer in the OSI model. It provides some channel access control mechanism, such as system access, bandwidth allocation, connection establishment, and connection maintenance. Some of the wireless MAC layer also support some security mechanism including authentication, secure key exchange and encryption. Compared with PHY layer, the processing of MAC layer is more for the packet processing and it needs to handle massive connections.

WiMAX forum proposes some profiles for system specifications' setting [16] to meet different system requirements, such as the system bandwidth, the coverage radius and the user supported. The system profiles also determines the required computation resources and I/O throughput. To demonstrate the multicore GPP capability in a more convincing way, we select the typical system profile which can meet most of the wireless deployment needs, as shown in table I. According to IEEE 802.16e, we implemented the PHY layer and MAC layer. Figure 1 shows the PHY layer operations of the WiMAX BS transceiver [14], which is the focus in this article.

In a wireless BS, one PHY layer instance and one MAC layer instance can work together to process the digitized radio

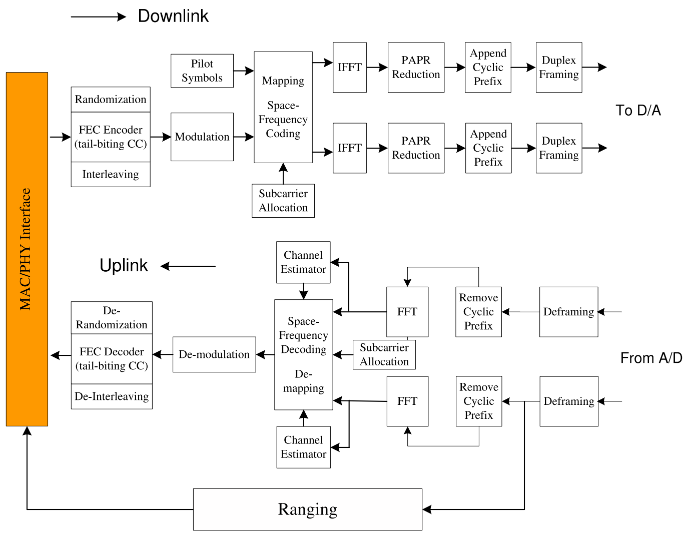

Fig. 1. WiMAX BS PHY layer system structure

TABLE I
System Parameters

Parameters	Value
System Channel Bandwidth (BW in MHz)	10
Sampling Frequency (Fs in MHz)	11.2
Duplex Mode	TDD
TDD DL:UL	1:1
FFT size	1024
Frame duration	5 ms
Zone Permutation	DL PUSC UL PUSC
Preamble	Support
FEC	CC: 1/2, CTC: 1/2
Modulation	16QAM, QPSK
STC support	Matrix A (2x1), Matrix B (2x2)
OFDMA ranging	Initial ranging, Periodical ranging

stream for one radio unit, which is depicted in Figure 2. Here, the radio unit contains the Digital-to-Analog (D/A) and Analog-to-Digital (A/D) converters, filters, down/up convertors, and power amplifier. The base band processing of PHY layer and MAC layer is located in GPP platform. The data throughput between GPP platform and the radio unit will be very high, which is the digitized raw data from radio channel. We can consider the High Speed Serial Link (HSSL) to support this interface, e.g. 10GbE, PCI express or Infiniband, which have been widely adopted in commercial servers and supported by lots of IPs and chip sets. In 802.16e standard, the packets out from the MAC layer will be transmitted to access service network (ASN) gateway. Between the MAC and PHY layer instances, we design an adaptor layer. This adaptor layer provides the data queues and management queues for both UL and DL between MAC and PHY layers, and it also provides some functionalities to hide the design gaps if the MAC and PHY layers are not provided by the same development teams, e.g. bit and byte conversion, OFDMA symbol forming, etc.

We implement one set of this stack on Intel Clovertown (quad cores @2.66GHz) platform, and compare the required computation resources among PHY layer, MAC layer and

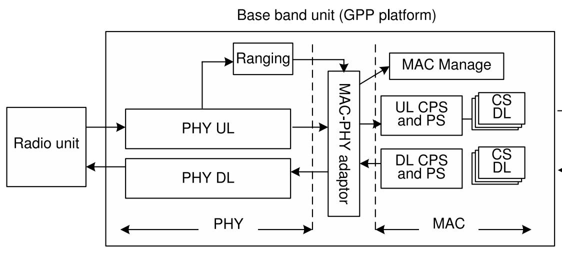

Fig. 2. BS stack for single radio unit

TABLE II

COMPARISON OF REQUIRED COMPUTATION RESOURCES

	PHY layer	MAC layer	PHY-MAC adaptor
computation resources	DL: 1 core, UL: 1 core	DL+UL: 1 core	DL+UL: 1 core
CPU utilization	DL: 21%, UL: 74.5%	DL+UL: 7.8%	DL+UL: 7.7%

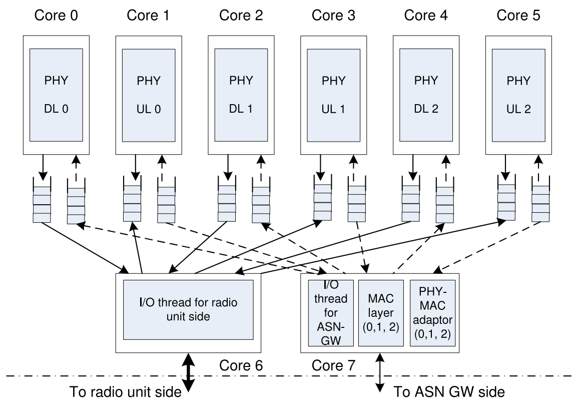

Fig. 3. Software structure for 3 BS instances on x3550 M2 with 2 Intel Clovertown processors

adaptor in table II. It is obvious that, PHY layer is much heavier than other two components. So, in the rest of this tutorial, we will focus on the multicore GPP implementation for PHY layer.

Though both Cell and Clovertown are multicore processors, the CPU architecture of them are quite different. Thus, the software designs for wireless BS are also different on these two multicore GPPs. In the following sections, we will introduce the software architectures designed on these two multicore processors.

B. Design over Intel Clovertown architecture

Intel Clovertown processor is a quad-core processor based on Intel Core 2 microarchitecture. It uses homogenous multicore architecture, where all the four cores are the same. Large size of L2 cache ($\geq$ 4MBytes) is shared among these four cores. The software design on Intel Clovertown will be simple. As long as the processing latency of an UL or DL sub-frame is within the frame duration, we don't need to use the pipeline model which will make the design complex. We have tested that, on single core of Intel Clovertown @2.66GHz, the sequential processing latency for an UL sub-frame is 2.9ms and the one for a DL sub-frame is 1.7ms. Both of them satisfy the 5ms frame duration, so we can use the simple software structure to support the multiple PHY layer instances.

Lots of state-of-the-art commercial IT platform supports symmetric multiprocessing (SMP) architecture, where two or more identical processors can connect to a single shared main memory. With the proper operating system (OS) support, any processor can work on any task no matter where the data for that task are located in memory. So SMP systems can easily move tasks between processors to balance the workload efficiently. For example, IBM System x3550 M2 is the server with two Clovertown processors and IBM QS21 blade server is the server with two Cell processors.

Thus, different from the traditional FPGA and DSP based platform, we need to consider how to support multiple base station stacks on a multicore GPP based IT platform. So, the highest cost and power efficiency can be achieved. Taking the x3550 M2 as example, we have tested that one core can support 3 MAC layer instances and 3 PHY-MAC adaptor instances with 99.99% quality assurance. Here the "99.99% quality assurance" means 99.99% frames can be handled within strict duration limitation, which is 5 ms in our experiment. With the other 6 cores supporting 3 PHY layer UL instances and 3 PHY layer DL instances, we can use single x3550 M2 platform to support 3 sets of WiMAX base station stack. Usually, each mobile cell will be divided into 3 sectors with 120 degree for each. Single x3550 M2 platform can support a base station with 3-sector typical configuration.

Figure 3 describes the software structure of 3 base station sectors (including PHY layer, MAC layer and PHY-MAC adaptor) supported by IBM System x3550 M2 server with two Clovertown processors. In this software structure, besides the 6 cores occupied by the 3 PHY layer instances, there is one dedicated core for all the processing of MAC layer, PHY-MAC adaptor and the I/O thread towards the ASN gateway side. And we use another core to handle all the data I/O towards the radio unit side, where the throughput is heavy (2.15Gbps).

C. Design over Cell architecture

Unlike Intel Clovertown, Cell processor is a kind of heterogeneous multicore processor. A single chip has one PowerPC Processor Element (PPE) and eight SPEs. The PPE unit on Cell is a general purpose 64-bit RISC core with 2-way hardware multithreading, used for operating systems and system control, and the 8 SPEs are 128-bit RISC processors specialized for data-rich, compute-intensive SIMD and scalar applications. These units are interconnected with a coherent on-chip element interconnect bus (EIB) [5].

Based on the architecture of Cell processor, we consider using PPE for central management of the PHY layer, and map the computational components in PHY layer into the 8 SPEs. There are two questions here. One is how to map those UL and DL components into 8 SPEs with the highest resource efficiency. The other one is how to design the framework on PPE to communicate with these 8 SPEs.

Within single PHY layer instance, we use pipeline mode for both UL and DL processing. Two pipelines are designed

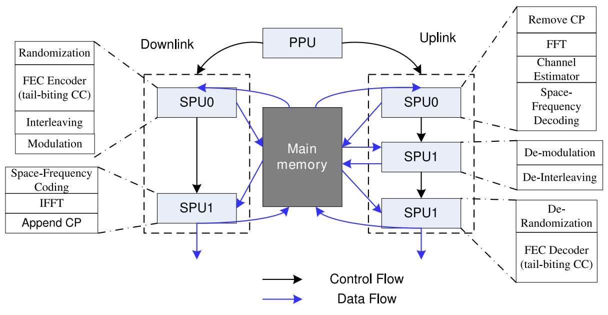

Fig. 4. Software structure for one PHY layer instance on Cell

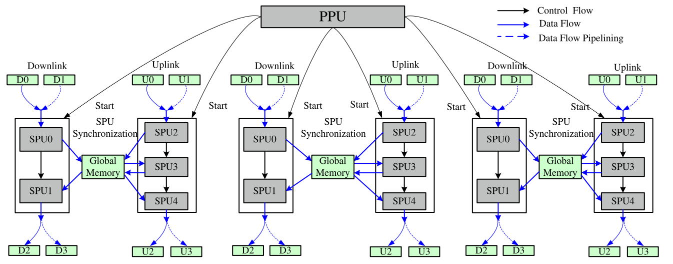

Fig. 5. Software framework for 3 PHY layer instances on QS21 with 2 Cell processors

for UL and DL respectively. In pipeline design, we define pipeline node which is the hardware core (SPE) to hold a group of components. The components within one pipeline node will be executed in serial way. The components on different pipeline nodes will be executed simultaneously, but the packets processed on different pipeline nodes are of different timeslots. Synchronization is required between two adjacent pipeline nodes, and it will adds communication overhead. To reduce the latency and increase the throughput, we try to use less pipeline nodes (SPEs) and group as much as possible components into one SPE. SPE has 256KB local store and the data movement between main memory and SPE local store needs Direct Memory Access (DMA) tranfers. So, two important constrains need be considered when determining one SPE can hold how many components.

- The total memory size of those components in local store should be less than 256KB.
- The total SPE processing latency of the serial components should be within the system requirement. In our test case of table I, it is 308.7 microsecond (us).

Based on the optimization results, those components in downlink can be partitioned into two SPEs, and those of uplink can be partitioned into 3 SPEs, as shown in Figure 4. As analyzed in [15], to avoid PPE becoming bottleneck of the system, we use the SPE synchronization method in this design. SPE will manage all the synchronization tasks between SPEs without the interference of PPE.

If IBM blade server QS21 with two Cell processors is used to support the base band processing of base station, there will be totally 16 SPEs on a server. Figure 5 shows the software framework of WiMAX PHY layer supporting 3 sectors on QS21 platform. Totally 15 SPEs are used and the remained SPE can be used for other purpose.

In section III, we will further discuss the software design considering the limitation of Cell memory structure. The alterative software framework is introduced under the help of some advanced features supported by Cell software development kit (SDK).

III. Analysis of Architectural Consideration

As mentioned in section I, multicore, SMT, multiple-issue are the main technologies widely utilized to provide high parallel performance with lower power today. With them, the processor architecture can support the multiple instruction streams, multiple data stream (MIMD) computation, according to the categorization from Flynn [18]. The PHY layer processing in wireless base station exhibits natural parallelism. The only question is, how much performance improvement of the PHY layer processing can be got from these technologies. Because the adoption of each of these technologies will increase the power and cost of processor. On the other hand, when design a high performance wireless base station stack on multicore GPP, we also need to consider the memory structure of the processor, which will highly impact the programming model. The difference between Cell and Intel Clovertown is the best example. So in this section, all these issues related to architecture designs will be analyzed.

A. *Parallelism with Multicore*

To analyze the performance improvement got from different parallel technologies, we need to check what resources need be shared in each parallel technologies. For multicore technology, cores in a multi-core device may be coupled together tightly or loosely. For example, cores may share caches (e.g. Intel Clovertown and Nehalem) or may not share caches (e.g. IBM Cell processor), and they may implement message passing or shared memory inter-core communication methods. No matter if they will share the cache or not, these cores will share the main memory and the bus. So, the performance improvement brought by multicore will depend on the characteristics of applications, such as load balance, synchronization, and sensitivity to memory latency.

In wireless BS stack supporting multiple sectors, there is no data dependency among these sectors. Within on sector, no data dependency exists between the uplink and downlink logics as well. Therefore, all the 6 instances running on core 0,1,...,5 in Figure 3 haven't dependency on each other. In order to measure the performance improvement by multicore, we take one experiment on Intel Clovertown, where different numbers of PHY layer instances are enabled. Each UL or DL instance occupies a dedicated core, and all the I/O data for PHY layer are put into the system memory. The result in table III shows that, there are around 36% performance loss for DL and 24% performance loss for UL when the number of parallel sectors increased from 1 to 4. The main reason for the performance loss is that, when there are 4 sectors (4 for UL and 4 for DL), the frequence of memory

TABLE III

AVERAGE THROUGHPUT OF SINGLE INSTANCE ON SINGLE CORE

Number of Sectors	Downlink (Mbps)	Uplink (Mbps)
1	35.64	12.86
2	28.19	11.50
3	26.45	10.02
4	22.73	9.69

access will significantly be increased. In DL processing, it will write the I/Q data into system memory with throughput of 716Mbps, and in UL it will also read the I/Q data from system memory with throughput of 716Mbps. When there are 4 sectors, the competition at the bus will be increased significantly. Compared with UL processing, DL processing requires less computation cycles. With the same amount of throughput requirement, impact from memory access will be heavier in DL than that on UL. Accordingly, the performance improvement ratio will be 2.55x for DL and 3.01x for UL when the number of cores is increased from 2 cores to 8 cores.

B. Parallelism with multiple issue

Multiple-issue processors come in two basic flavors: superscalar processors and Very Long Instruction Word (VLIW) processors. VLIW is widely used in embedded processors such as DSP processor, while superscalar is popular in server processor area, e.g. Intel Clovertown, IBM Power 6 etc. A superscalar processor executes more than one instruction during a clock cycle by simultaneously dispatching multiple instructions to redundant functional units on the processor. Each functional unit is not a separate CPU core but an execution resource within a single CPU such as an arithmetic logic unit (ALU), a bit shifter, or a multiplier. Mutiple issue technology provides the parallelism in instruction level, called Instruction-Level Parallelism (ILP) [17]. Usually, the ILP can be achieved by good compilers. It is obvious that, for the same set of codes, the performance improvement ratio got from multiple issue will highly depend on the number of functional units in a CPU core.

One experiment is taken on both Intel Clovertown and IBM Power 6. We implement the Viterbi decoder on both of them with the same algorithm and optimization. The only difference is the SSE instructions on Intel Clovertown is changed to vector multimedia extension (VMX) instructions for Power6. The results are listed in table IV. In Viterbi decoder, SIMD instructions are the major part. Though both IBM Power6 and Intel Clovertown are superscalar processors supporting multiple issue, core of Power6 has only one SIMD execution unit, while Intel Core 2 has 3 SSE execution units and can issue 3 SIMD instructions at one time. So, the CPI of the Intel Clovertown can be much lower than that of Power6. Removing the impact of CPU frequency, the performance improvement ratio is between 2x 2.5x here.

C. Parallelism with SMT

With SMT technology, multiple threads share the execution units of a single core in an overlapping fashion. If one thread

TABLE IV

VITERBI DECODER PERFORMANCE ON INTEL CLOVERTOWN AND IBM POWER6

Number of Sectors	Intel Clovertown	IBM Power6
Frequency	2.66GHz	4.03GHz
SIMD instruction proportion	67%	66%
Cycle per instruction (CPI)	0.68	1.7
Decoder throughput	33.1Mbps	24.5Mbps

TABLE V

PERFORMANCE COMPARISON BETWEEN SINGLE THREAD AND 2-WAY SMT

	1024-FFT	Viterbi Decoder
CPI with single thread	0.467	0.96
Iterations per second with single thread	272331	3187
Iterations per second with 2-way SMT	266898	4942

stalls, perhaps waiting for the memory I/O, the execution unit continues to execute the other thread, resulting in a more fully utilized CPU. It is also called Thread-Level Parallelism (TLP). As explained in [17], actually SMT came from the modern multiple-issue processors, which often have more functional unit parallelism available than a single thread can effectively use. So, SMT exploits the TLP and ILP simultaneously. Compared with multiple-issue and single-thread processor, SMT can further improve the parallelism performance for those workloads with low ILP performance in single thread environment.

We use the two most important components, 1024-FFT and Viterbi decoder, to implement an experiment on Intel Nehalem who is using the 2-way SMT design. The results are listed in table V. In the single thread experiment, there is only one thread enabled for the FFT or Viterbi decoder in one core. In the 2-way SMT experiment, we enable two threads, each of which will hold one FFT or Viterbi decoder instance. So the result for 2-way SMT mode is the total number from two threads. When FFT and Viterbi decoder are both executed on single thread mode, the CPI of FFT is much lower than Viterbi decoder. That means, FFT library has much better ILP performance than Viterbi decoder in single thread environment. Thus, when we enable two threads in the same core, Viterbi decoder can get significant performance improvement from TLP than FFT. For Viterbi decoder, the performance gain got from 2-way SMT is 55%. Thus, when we decide to use 2-way (or higher) SMT mode or single thread mode on SMT platform, we should first check the ILP performance of the stack under single thread mode.

D. Memory structure and program model

In Intel Clovertown, there is L1 cache on each core and the L2 cache shared among the 4 cores. When the data and instruction size is larger than the cache size or they are missing in the cache, hardware will handle these issues automatically. Thus, when we design the wireless BS software on Intel Clovertown, we only need to pay attention to the latency limitation as introduced in section II, and will not worry much

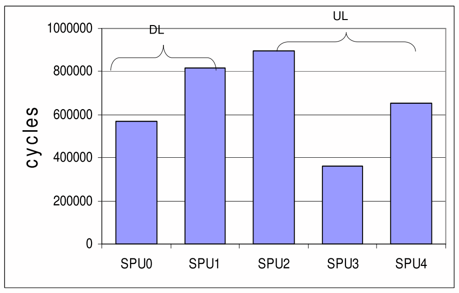

Fig. 6. Required processing cycles for a data block on different SPEs

about the data and code size of the workload.

But, the memory structure of Cell is quite different. SPEs are the main computational cores. It accesses the main memory with DMA commands that move data and instructions between main memory and a private local memory, called a local store (LS). An SPE's instruction-fetches and load and store instructions access its private LS rather than shared main memory, and the LS has no associated cache. Such memory organization is a radical break from conventional architecture and programming models [5]. Cell programmer needs to consider if the data and instruction size will exceed the 256KB LS or not. If the size is larger, the sequential workload has to be partitioned onto multiple SPEs. There are following disadvantages.

- It will require extra efforts to partition the workload and get the best scheme. The best scheme means that it can use less SPEs, and the processing latency of each SPE can be balanced.
- In pipeline processing, there will be synchronization overhead.
- It's hard to get a partition scheme where all the SPEs have exactly the same latency. So the computation resources of some SPEs can not be fully utilized when the pipeline is stalled by the SPE with longest latency.

Based on our WiMAX PHY layer optimization result on Cell processor, Figure 6 shows the required processing cycles when a data block is handled through the pipelines of UL or DL. It is obvious that, the SPE0 in DL and SPE3/SPE4 in UL can not be fully utilized. In order to make the work easier and get better performance, we try the code overlay released in Cell SDK 3.1. Code overlay is a technique to overcome the physical limitations on code and data size in the SPE. In an overlay structure the local storage is divided into a root segment, which is always in LS, and one or more overlay regions, where overlay segments are loaded into the same region when needed. More details can be found in [21]. Under the support of Cell SDK 3.1, we change the original pipeline mode (in Figure 4) into the overlay mode as shown Figure 7. In this new overlay mode, we still use two SPEs for DL processing and 3 SPEs for UL processing, where each SPE will finish the entire UL or DL processing. In DL SPE, we partition all

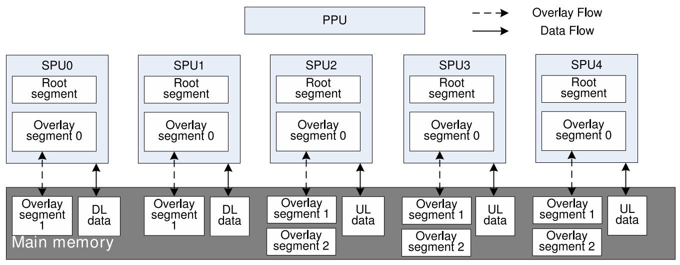

Fig. 7. Software structure for overlay mode on Cell processor

the components into two overlay segments, which means there is one code overlay in the DL processing of each data block. In UL SPE, there are 3 overlay segments, and twice code overlay will be executed accordingly. The throughput of both UL and DL is tested on the same Cell platform (IBM QS21 blade server). The average throughput of DL on single SPE is 13.98Mbps and that of UL is 9.885Mbps. Totally, 5 SPEs can support 27.96Mbps DL data processing and 29.66Mbps UL data processing. Compared with the throughput of original pipeline mode (24.15Mbps for DL and 22.80Mbps for UL), we can get 16% performance enhancement for DL and 30% for UL. More important is, the overlay mode is much easier to be implemented and optimized than the pipeline mode.

IV. Related system requirements

In this section, we will analyze other system requirements when multicore GPP is used to support wireless BS. No ideal solutions have been explored to solve these requirement yet. So we list them in this artical to get more attentions from both IT industry and wireless industry.

A. *Power efficiency requirement*

To use multicore GPP for wireless BS, the first challenge is power efficiency. Compared with the DSP from TI and Freescale, today's multicore GPP usually has 60watts$\sim$100watts power consumption, which is much higher than DSP. But the growth of architecture and CMOS technologies are very amazing in GPP market. From the roadmap of those leading processor providers, such as Intel, in 2010, the 8-core 16-thread GPP will be available. In 2011, the GPP (Sandy Bridge) with 8 cores, 16-threads and 256-bit width vector engine will be available. Based on our current result on Clovertown, we can roughly estimate that, single Sandy Bridge processor can support 5 WiMAX base station sectors including both the MAC layer and PHY layer. We don't need extra network processor, such as Intel IXP or RMI XLR to support the MAC layer, and we don't need another micro control unit (MCU) such as PowerPC processor to support the control and management. On the other hand, those advanced CMOS techniques will be adopted on GPP products aggressively. For example, in the coming 2 to 4 years, the GPP processors with 32nm and 22nm techniques will occur in the market. All of these can heavily drive the evolution of multicore GPP towards lower power and high computation density, and it can give us the expectation that multicore GPP can deliver competitive power efficiency to support wireless BS in near future.

Though multicore GPP can support all the base band processing in wireless BS and provide the ideal SWR with high flexibility, some component such as Turbo decoder can not get an efficient implementation with pure software. After optimization in software, Turbo decoder still have 10x complexity of Viterbi decoder in our implementation. To increase the power efficiency without losing flexibility, the programmable FPGA can be considered as an accelerator to support the channel decoder. For example, an FPGA board with PCI express interface can be used to work with multicore GPP platform.

B. Throughput Requirement

As shown in Figure 2, base band unit of WiMAX BS interfaces the radio unit on the left side, and the ASN gateway on the right side, which will be similar for other standards. For convenience, we define the interface with radio unit as R-B interface and the one with ASN gateway as B-GW interface. Taking one base station sector for throughput calculation, the I/O throughput on B-GW interface is around 11Mbps according to the profile defined in table I, and the I/O throughput on R-B interface is around 720Mbps. If we use the future processor with 8 cores 16 threads and 256-bit vector engine as example, single processor can support 5 WiMAX BS sectors. The throughput on the R-B interface will be around 3.6Gbps. The server with dual processors need to support 7.2Gbps throughput on R-B interface.

The other big throughput requirement will come from channel decoder accelerator. If we need FPGA board to support channel decoder, according to table I, with 8-bit soft input data, the data throughput from CPU to FPGA for single sector will be around 440Mbps. On the future server platform with 2 processors supporting 10 WiMAX BS sectors, the total throughput requirement for channel decoder will be 4.4Gbps.

So the total throughput requirement including the R-B interface and channel decoder interface is as high as 12Gbps. It brings a big challenge on the system software design, such as high efficient device drivers and the I/O threads for R-B interface and channel decoder interface.

C. Real-time Requirement

Real-time requirement lies in two aspects. One is related to R-B interface, the other one is related to the strict frame duration or timing.

On the R-B interface, we can use 10GbE, Infiniband or PCI express to transmit the DL signal to radio unit. Because the BS software stack is built upon operating system (OS), jitter will be generated under the impact from OS scheduler. On the other hand, multiple BS sectors can be supported on single server platform and share the same HSSL on R-B interface. That means, one server platform will interface multiple radio units. Switch can be used between the base band server and radio units to keep the flexibility. Then another kind of jitter will be involved by switch. The straight forward way to solve the jitter problem is to add the buffer on R-B interface, where the size of buffer will highly depend on the jitter. Then, the latency problem will occur due to the large buffer. Especially in future LTE, the allow round trip time is reduced to 6ms. So, it is a critical requirement to control the jitter within an acceptable level.

Another real-time problem comes from the strict frame duration or timing. In wireless standard, once the frame duration is defined, the strict timing will trigger the system and require all the frames ($>$99.99%) being processed within this duration. In our case, when OS scheduler is used, even if the CPU hasn't been fully utilized, we can see the processing time of a small proportion of frames will exceed the required duration. To avoid this, using dedicated hardware core for single software thread can be one of the solution. But in most case, we haven't enough hardware cores to map each software threads onto them, especially when we have the system to support both PHY layer and MAC layer. Another solution is to lower down the CPU utilization. In some of our experiments, when CPU utilization is less than 60%, this problem will disappear. But it's not a good way that both cost efficiency and power efficiency will be decreased.

Thus, we still need further efforts to deal with real-time problem, e.g. to use some advanced scheduler mechanisms in OS kernal, or to explore some new program models to handle the scheduling in application, etc.

V. Review of related work

A. Related GPP-based work for other standards

For more than 10 years, there have been some trails and experimental systems to demonstrate GPP platforms' capability in supporting wireless system.

T. Turletti et al. consider the software implementation of a GSM base station on GPP and analyze the performance of each of its radio interface modules in [6]. In his analysis, one frequency channel of GSM base station requires one and a half Pentium Pro 200 processor.

P. Mackenzie et al. use GPP and operating system to achieve a flexible software radio design for an frequency modulation (FM) wireless system [7].

Vanu Inc. delivered the first commercial base station product utilizing IT platform with GPPs, which supported both GSM and CDMA2000. The entire system, including the signal processing for the air interface, is simply a collection of software applications that can run on any general purpose processor (e.g. Pentium, Xeon, PowerPC and ARM) [8].

Sora software radio platform from Microsoft Research Asia [13] is the one built upon multicore GPP platform in 2008. The experiment is for 802.11a/g, and this platform is mainly targeting the terminal or access point other than base station.

[22] provides the summary of million instruction per second (MIPS) requirement for some key wireless standards. The one for GSM carrier is 100MIPS and the one for 802.11a/b is 9000MIPS. According to our WiMAX test-bed, single WiMAX carrier with profile in table I will require around 64000MIPS, which is much heavier than GSM and 802.11a/b

tried in [8] and [13]. On the other hand, as discussed in section III, multiple BS sectors should be supported on a GPP platform to get the better power and cost efficiency. The system design will be more complex than a single wireless access point as introduced in [13].

B. Other multicore architecture for wireless BS

Multicore architecture is not only occurred in GPP area, those processors used in embedded system are also moving towards multicore direction.

SODA [20] is a multicore processor architecture designed to support SDR. It has 4 cores, each of which contains asymmetric dual pipelines that support scalar and SIMD execution, and another ARM Cortex-M3 for control purpose. But it is still a prototype in research lab.

TMS320TCI6487 is a multicore DSP from TI, who has 3 C64+ cores in a single chip. MSC8156 is a multicore DSP processor from Freescale. There are 6 StarCore DSP SC3850 cores, and two RISC core QUICC Engines.

The major difference between these embedded processors and Intel's multicore processors is, these embedded processors have no shared cache across computational cores. It will use the self-contained program mode in each core, which will generate lots of limitation on software design.

VI. Conclusion

In this paper, we introduce the wireless base station design on two different multicore platforms, IBM Cell with heterogeneous multicore architecture and Intel Clovertown with homogenous multicore architecture. 802.16e is chosen as the target standard. Due to different multicore architecture and different memory structure, the software structure designs on these two platforms are quite different. To get high performance implementation on multicore GPP, we should take the architectural impacts into account. Thus, a complete analysis on multicore, multiple issue, multithreading, memory structure and the various program models are provided in this paper, together with our performance results. From these results, people can learn how to implement the SWR base station on multicore GPP platform with suitable program model, how to support the multiple BS sectors on a multicore GPP platform, and how the multicore processor architecture will impact the performance. Besides, other key system requirements are also discussed in this paper, which will be critical when we want to build up a SWR BS solution upon multicore GPP platforms. Some of these requirements contain technical challenges in system software design, such as the system software for high throughput interface and the real-time problems, which will have high value for further study and exploration.

References

[1] M. Hata, "Fourth Generation Mobile Communication Systems Beyond IMT-2000," *Proceedings of the Fifth IEEE Asia-Pacific Conference on Communications, Beijing China*, October 18-22, 1999, pp.765-757.

[2] http://www.sdrforum.org

[3] J. Mitola III, "Technical Challenges in the Globalization of Software Radio," *IEEE Communication Magazine*, pp. 84-89 February 1999.

[4] John L. Hennessy, David A. Patterson Computer Architecture: A Quantitative Approach (Third Edition), Morgan Kaufmann, San Fransisco, CA, 2002.

[5] IBM(2006). Cell Broadband Engine Programming Handbook Version 1.0, April 19, 2006.

[6] T. Turletti, D. Tennenhouse, "Complexity of a software GSM base station,"IEEE Communication Magazine, vol. 37, no. 2, pp. 113-117, Feb. 1999.

[7] P. Mackenzie, L. Doyle, K. Nolan, D. O' Mahony, "An Architecture for the Development of Software Radios on General Purpose Processors," ISSC 2002, Jun. 2002.

[8] Vanu Anywave Software RAN Solutions; see http://www.vanu.com/media/Anywave%20Data%20Sheet.pdf

[9] Philip Mackenzie, Linda Doyle, Keith Nolan, Donal O' Mahony, "An Architecture for the Development of Software Radios on General Purpose Processors," *TBD*

[10] Y. Jay Guo, *Advances in Mobile Radio Access Networks*, Artech House, 2004.

[11] A. K. Salkintzis, H. Nie, P. T. Mathiopowlos, "ADC and DSP Challenges in the Development of Software Radio Base Stations," *IEEE Personal Communications*, vol. 6, no. 4, pp. 47-55, Aug 1999.

[12] Z. Salcic, C.F. Mecklenbrauker, "Software Radio – Architectural Requirements, Research and Development Challenges," *International Conference on Communication system, ICCS 2002,* vol. 2, pp. 711-716, Nov.2002.

[13] K. Tan, J. Zhang, J. Fang, et al. "Sora: High Performance Software Radio Using General Purpose Multi-core Processors," NSDI 2009, TBD

[14] Q. Wang, D. Fan, Y.H. Lin, "Design of BS Transceiver for IEEE 802.16E OFDMA Mode," *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, March 30 - April 4, 2008, pp 1513-1516.

[15] J.W. Chen, Q. Wang, Z.B. Zhu, Y.H. Lin, "An Efficient Software Radio Framework for WiMAX Physical Layer on Cell Multicore Platform", *ICC2009*, Jun. 2009, Dresden, Germany. Accepted.

[16] WiMAX Forum. Mobile System Profile 3– Release 1.0 Approved Specification (Revision 1.7.1: 2008-11-07), WiMAX Forum.

[17] John L. Hennessy, David A. Patterson Computer Architecture: A Quantitative Approach (Third Edition), Morgan Kaufmann, San Fransisco, CA, 2002.

[18] Flynn. M. J. "Very high-speed computing system," Proc. IEEE 54:12, Dec. 1966, pp 1901-1909.

[19] L. Eisen, J. W. Ward III, H.-W. Tast, N. Mading, J. Leenstra, S. M. Mueller, C. Jacobi, J. Preiss, E. M. Schwarz, and S. R. Carlough, "IBM POWER6 Accelerators: VMX and DFU," IBM Journal of Research and Development, 51, No. 6, 663 – 683, 2007.

[20] Y. Lin, Hyunseok Lee, M. Woh, Y. Harel, S. Mahlke, T. Mudge, "SODA: A Low-power Architecture for Software Radio," International Symposium on Computer Architecture (ISCA), Boston, Massachusetts, June 17-21, 2006, pp. 89-101.

[21] IBM, "Software Development Kit for Multicore Acceleration Version 3.1 Programmer' s Guide"

[22] M. Alsliety, and D. N. Aloi, "Signal Processing Choices and Challenges for SDR in Telematics," Proceedings of International Symposium on Signal Processing and Its Applications (ISSPA), Sharjah, United Arab Emirate, February 12-15, 2007, pp. 1-4