SPEECH CODEC OPTIMIZATION BASED ON CELL BROADBAND ENGINE

Zhenbo Zhu, Qing Wang, Bo Feng, Ling Shao

IBM China Research Lab, Beijing, 100089
{zhuzb / wangqing / fengbo / shaol} @ cn.ibm.com

ABSTRACT

Cell broadband engine (CBE) is a multi-core processor jointly developed by Sony, Toshiba and IBM. The multi-SPE architecture makes it powerful for streaming media processing, such as voice over IP application. In this paper, a CBE based IP media server (IMS) architecture is proposed and the workloads of ITU-T G.723.1 and GSM-AMR encoder on CBE are analyzed with some preliminary optimization result from both CBE simulator and real CBE hardware. Branch instruction reduction, data level parallelism (SIMD), instruction level parallelism and aligned SPU local store access techniques are applied in our single precision floating-point based reference software optimization on Cell. The performance we have achieved demonstrates the Cell competence for computation intensity applications in telecom industry

Index Terms — Multimedia System, Cell Broadband Engine, Speech Codec, Optimization, Voice Conference

1. INTRODUCTION

IP media server (IMS) is an essential element of next generation network (NGN). The market pressure and network deployment worldwide make IMS develop quickly in recent years. The central function of IMS is to process media stream, especially for voice related applications. A typical IP media system should complete announcements, interactive voice response (IVR), conferencing, speech recognition and Fax. Some IMS can support video processing for video conference implementation.

For telecom media processing, it needs to process very high density streams, such as compressed voice stream. For example, if an IP media system is required to support several hundreds of three-party conferences with ITU-T G.723.1 compressed format, the computation load is very high. Therefore, the computation capability is the key issue for an IMS. In current market, both hardware and software solutions have been proposed. For hardware solution, the media processing is processed by DSP or ASIC, while for software solution, the work is done by general purpose processor (GPP) [1]. Compared with software solution, the most advantage of hardware solution is the computation capability since it's designed dedicated for specific purpose. Using multiple DSPs to co-process the stream data can significantly improve the computation capacity [2]. However, both of the software and hardware solutions face the challenges of performance/cost (price or watt). In order to improve the computation density, hardware solution providers have two choices. One is to optimize the media processing algorithms to reduce the workload; the other is to select more powerful DSPs or novel processors.

Cell broadband engine (CBE) is a single-chip multi-core processor jointly developed by Sony, Toshiba and IBM. It's directed toward distributed processing targeted for media-rich applications such as game consoles, home media server, and accelerating systems. Its special architecture makes it powerful for streaming processing. In our work, we proposed and developed a simple IMS architecture with basic Voice Conference function on single CBE processor, and have preliminary optimization results for the key components of the IMS system -- G.723.1 and AMR encoder. The optimization techniques are summarized and the results are obtained on simulation environment and a real Cell Blade Server [3].

The paper is organized as follows. The cell broadband engine and general software development flow are introduced in section 2. In section 3, the overview of the IMS System on CBE is given, and the workloads of speech codec are analyzed and the optimization work is introduced. The testing results are presented in section 4. Finally, we conclude the paper.

2. CELL BROADBAND ENGINE

The Cell processor consists of one PowerPC Processor Element (PPE), eight Synergistic Processor Elements (SPE) and memory flow controllers (MFC). The PPE unit is a general purpose 64-bit RISC core used for operating systems and program control, while the SPE is optimized for efficient data processing. These units are interconnected with a coherent on-chip element interconnect bus (EIB).

The system frequency of Cell can achieve 3.2GHz and the computation capability is 256GFlops. The block diagram of Cell architecture is shown as Fig 1.

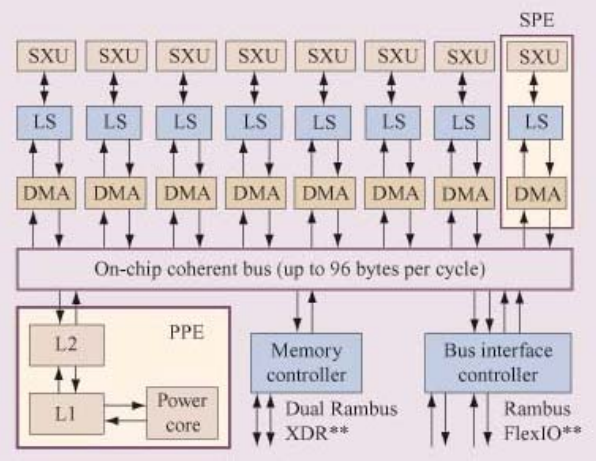

Fig. 1 Cell Processor Block Diagram [4]

The SPE unit is designed for high performance computing. It has 256KB local memory and supports up to 16-way 128-bit SIMD (single instruction multiple data) operation. It offers a high bandwidth interface to a direct memory access engine that can transfer 32 GB/sec to and from the 256KB local memory. Each SPE can initiate up to 16 independent DMA transfers [5]. Therefore, it provides a coherent offload engine for the PPE.

A typical software development flow on CBE is as follows:

- Module mapping to the multi-core CBE architecture.
- Workload and bottleneck analysis against hardware constraints for the algorithms implementation.
- Developing PPE and SPE code.
- Accelerating the computing consuming code on SPU by SIMD and other optimized methods.
- Re-balance the computation and data movement
- Turning the whole system

3. SPEECH CODEC WORKLOAD ANALYSIS AND OPTIMIZATION

In this section, an overview of the IMS based on Cell processor will be presented at first. The implementation and workload analysis of the IMS's key components -- speech codecs will be discussed based on CBE following. Finally some basic optimization techniques for the speech codecs are given.

3.1. Overview of the IP Media Server on CBE

We proposed a simple IMS architecture on a single CBE, which has the basic functions of a Voice Conference System shown as Fig. 2. Considering the uncouple relationship between different components in one frame-duration and the limited text code size, no functional component will be split and mapped into two SPU or more. The PPU of the CBE will be dedicated to network packets processing and the scheduler of the whole work flow of the system. It will process the network packets from/to IP network, control the data flow, and assign, schedule and link the work of each SPU.

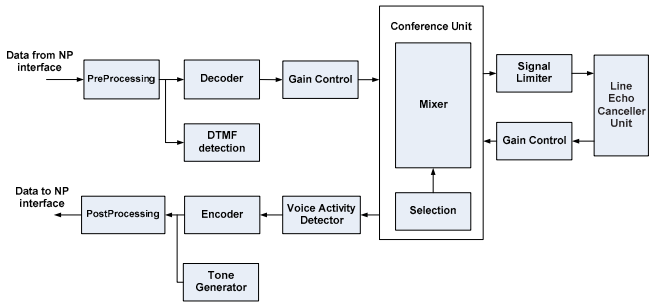

Fig. 2 Function Diagram of the IMS prototype on CBE

We selected ITU-T G.711a/u, G.723.1 and GSM-AMR as the supported speech codecs in our IMS prototype due to their wide usage in current VoIP and GSM system. According to their workloads on the Texas Instruments' DSP and Intel CPU, we believe that the G.723.1 and AMR speech encoder will be the most computation consumed components in our system. Our work will focus on the G.723.1 and AMR speech encoder optimization on the SPU at the first stage.

3.2. Workloads analysis of speech codec on the SPU

As the beginning of the development work for the IMS system based on Cell processor, we analyzed the workload of G.723.1 and AMR codec by porting their reference code from ITU-T and ETSI organization on Mambo [6], which is a full system simulator of CBE and can simulate the Cell processor very accurately.

ITU-T and ESTI provides the reference codes both for fixed-point and floating-point implementation. For Cell processor, it is compatible with fix-point and float-point operations. A workload test is executed for the reference code selection and workload analysis. We ported both the single precision floating-point and fix-point original reference codes of the G.723.1 / AMR encoder to single SPU on Mambo. The cycles per frame of the six most cycle consumed functions of the G.723.1 6.3kbps encoder are listed in Table 1. The peak cycles consuming per frame of the whole encoder is listed in Table 1 also.

Table 1 Comparing results from the un-optimized fix-point and single precision floating-point based reference software on single SPU

Function of G.723.1 6.3kbps encoder	Fix-point implementation cycles per frame (FX)	Floating-point implementation cycles per frame (FP)	Cycles Ratio (FX / FP)
Find_Fcbk	32974050	3962326	8.3
Find_Acbk	21372478	1701510	12.6
Estim_Pitch	5807765	392469	14.8
Comp_Lpc	2382348	139308	17.1
Lsp_Qnt	1750182	258887	6.8
Comp_Ir	1710425	248228	6.9
Encoder	**67857340**	**6978988**	**9.7**

From table 1, we notice that the floating-point implemented speech codec has significantly better performance on SPU. The reasons are analyzed as follows:

- The reference code quality impacts codec performance significantly. For example, there are too much branch operations in the reference code which heavily reduce the performance of SPU because of the huge stall cycles of branch operation [7].
- SPU is optimized for single precision floating-point computation, and few fix-point operations are not hardware supported. For example, there is no saturation operation instruction on SPU. All the saturation operations will be handled using branch or shift operation with much overhead.
- Fix-point based implementation will introduce more instructions for an essential floating-point application like speech encoding.

And there are some other reasons driving us use floating-point implemented speech codec on CBE. The coefficients of the speech models and the excitations in G.723.1 and AMR standards are floating-point based essentially. Hence, the floating point is much suited if the hardware supports. And, the single precision floating-point has 23 significant bits. It can fully satisfy the precision requirements of speech codec with a few additional double precision operations.

Based on the above discussion, we used floating-point reference code for our IMS implementation on CBE. And based on the algorithm analysis for the G.723.1 and GSM-AMR encoder, we think the primary operation of the G.723.1 and AMR encoder is dot product. And the basic computation is multiply-add. We believe these speech encoding are computation consumed workload and will be very suited for CBE. In our work, we set all the code of the encoder in the local store of the SPU, and change the Heap memory allocation to the Stack memory allocation to reduce the local store control of the SPU, and give a basic optimizing for the speech encoder.

3.3. Optimization Techniques of speech codec on the SPU

SPU core on Cell is a tailored and sensitive vector processor. The code should be vectorized to exploit the strength of SPU. Another problem is the local memory of SPU. The size of Local Store in each SPU is only 256KB, which should be allocated for the Application Binary Code, Data, Heap, Stack, DMA Transfer Buffer and SPU side Software Framework Code. A carefully tradeoff between the size and the performance of the code must be considered in the optimizing work.

We adopt the following optimization techniques for AMR encoder optimization on Cell platform.

1) Reduce Text Code Size

As we known, the G.723.1 and GSM-AMR are multi-rate speech codec, there are several different Tables and codes for specific compressing rate and will not be used together by the encoder in one frame-duration. And one SPU has only 256kB local store which is limited for the whole application like speech encoding. So we divide the speech encoder into the rate-related part and non- rate-related part. The rate-related part will not resident in the SPU Local Store and will be exchange when the speech encoder change its compressing rate. It is much effective for the adaptive multi-rate speech encoder.

2) Reduce Branch

We notice that branch can significantly influence the efficiency of the SPU in some components of the Speech Encoder. Because SPU is an in-order processor with no branch prediction, any judgment will result in the SPU stall, such as the “if” operation, rolling (“for”), min / max judgment and absolute operation. And we believe that lots of judgment can be avoided by some optimized technique. For example, using the compare–select function instead of short judgment function is a good optimization method for most branches in Speech Encoder.

3) Align SPU Local Store Access

The best assess pattern for SPU is data and structure aligned with vector operation. The Scalar and unaligned access will result in many additional instructions for data aligned and scalars extracted from vectors. In some case, we can operate the scalar as the vector. This method solves the data access problem of the SPU which can not be made as SIMD pattern.

4) Optimize instruction pipeline for the instruction level parallelism

The SPU has two pipelines for instruction issue, one for computing instructions and the other for instructions accessing Local Store. If two conjoint instructions can be placed in the different pipeline with no dependency, the two instructions can be dual-issue. Each instruction has its

latency and Stall cycles which will influence the efficiency of the SPU due to the dependency.

5) Apply SIMD instructions to explore data level parallelism

SIMD and the large register file are the direct ways to accelerate the code. In speech encoder, the primary operation is dot product which has less data relativity and can be easily SIMD.

4. PEROFRMANCE RESULTS

The optimized G.723.1 and AMR encoders are test on Mambo for cycle numbers of each function. The results are listed in Table 2. The speed-ups of each function of G.723.1 encoder are selected shown as Fig. 3. The encoding channel numbers supported on single CBE are test on a real Cell Blade Server at 2.4GHz shown as Table 3. (8 SPUs of the CBE are used for speech encoding.) After the preliminary optimization, we reduced 75% cycles of the AMR encoder, 78% cycles of G.723.1 5.3kbps encoder and 86% cycles of G.723.1 6.3kbps encoder consuming successfully. In another words, the peak performance of optimized encoder is 4~7 times faster than the original one. Furthermore, Cell processor can work at 3.2GHz. That means the performance results can be much better. (Shown as Table 3)

Table 2 Performance of the preliminary optimized encoder test on Mambo

Speech Encoder	G.723.1 5.3kbps	G.723.1 6.3kbps	GSM-AMR (peak)
Cyc. per frame	921672	936538	616267
Code size	139.3kB	139.3kB	178.5kB
Application Speed-up	4.59	7.45	4.1

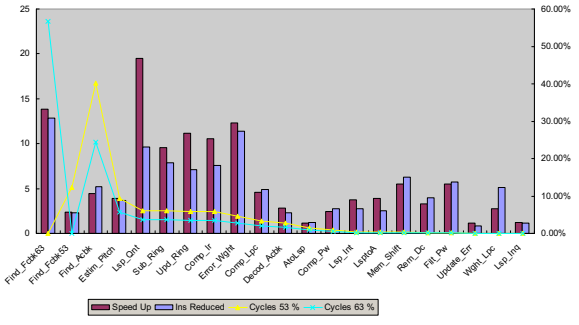

Fig. 3 Function speed-ups of the G.723.1 encoder test on Mambo

Since it is the preliminary optimization for the speech codec, the cycles per instruction (CPI) of the optimized codes are 1.46~1.54, while the theoretical optimum CPI is 0.5. And the dual-issue rates of our optimized codes are less than 35%. That means there is over 40% stall cycles cost in our optimized codes, and the double-issue rates are very low. Considering the potential of the algorithms optimizing, there are still much potential for our latter optimization.

Table 3 Performance of the preliminary optimized encoder test on real Cell Blade Server

Speech Encoder	G.723.1 5.3kbps	G.723.1 6.3kbps	GSM-AMR (peak)
Channels Supported on one Cell, achieved (8SPU at 2.4GHz)	640	616	624
Channels Supported on one Cell, expected (8SPU at 3.2GHz)	853	821	832

5. CONCLUSION

In this paper, we proposed an IP media server architecture based on Cell processor, demonstrate CBE performance for G.723.1, AMR speech codecs, and explore the potential opportunities for CBE in telecom industry. In Digital Signal Processing area, there are many applications which have the floating-point operations essentially. Traditionally, they have to convert to use fix-point implementation due to the power consuming and limited computing resource of the traditional floating-point Digital Signal Processor. We believe the Cell processor can be used widely for those applications with much better performance.

6. REFERENCES

[1] “Next-generation media processing for the modular network–Intel White Paper,” Intel.

[2] Qing Wang, Zhenbo Zhu, Yi Ge and Ling Shao, “Design of IP Media Server for Voice Conference Application”, *APCCAS06*, Singapore, Dec.4-7, 2006.

[3] http://www-03.ibm.com/technology/splash/qs20/

[4] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy, “Introduction to the Cell multiprocessor”, *IBM Journal of research and development,* Volume 49, Number 4/5, 2005.

[5] Lurg_Kuo Liu, Sreeni Kesavarapu, etc. ”Video Analysis and Compression on the STI cell Broadband Engine Processor”, IBM Research Report, Feb., 2006.

[6] http://www.alphaworks.ibm.com/tech/cellsystemsim

[7] “Cell Broadband Engine Programming Handbook” IBM Corporation