MULTI-THREADED PROCESSOR FOR SOFTWARE-DEFINED RADIO

John Glossner, Erdem Hokenek, and Mayan Moudgill
Sandbridge Technologies, Inc.
White Plains, NY
914-287-8500
{glossner,hokenek,moudgill}@sandbridgetech.com

ABSTRACT

In this paper we discuss broadband communications systems and technologies being developed to allow real-time execution using software implementations. We discuss a new multithreaded SDR core designed by Sandbridge Technologies capable of executing RISC, DSP, and Java code. We then describe the software tools developed for the hardware. We then describe our 2Mbps WCDMA C code implementation and the SB9600^{TM} baseband product. We show that it is possible to design communications systems in high-level languages and automatically generate efficient code that runs in real-time on the SB9600^{TM} platform.

1. INTRODUCTION

High-speed communications are proliferating and digital signal processors (DSPs) are accelerating this trend [1]. DSPs have become a ubiquitous enabler for integration of audio, video, and communications [2]. Furthermore, DSPs are the driving force accelerating wireless communications. In the future world of convergence devices, efficient Control, DSP, and Java execution may be important components of system performance.

Tremendous hardware and software challenges exist to realize convergence devices. First, power dissipation constraints are requiring new techniques at every stage of design - architecture, microarchitecture, software, algorithm design, logic design, circuit design, and process design. With performance requirements exploding as bandwidth demand increases, power conscious design becomes more difficult. SOC integration and low voltage process technologies will contribute to lower power system-on-a-chip (SOC) integrated circuits (ICs) but are insufficient as the only solution for streaming multimedia.

Second, convergence applications are fundamentally DSP applications. In addition, these applications are becoming very complex. In wireless communications, GSM and IS-54 data rates were limited to less than 15 Kbps. Future third-generation (3G) systems may provide data rates more than 100 times the previous rates. Higher communication rates are accelerating higher processing requirements.

Third, complexity is driving the need to program applications in high-level languages. In the past, when only small kernels were required to execute on a DSP, it was acceptable to program in assembly language. Today, resource constraints prohibit these practices. A popular high-level language suggested for use in cellular applications is Java. Java is a C++ like programming language designed for general-purpose object-oriented programming [3]. An appeal for the usage of such a language is its ``write once, run anywhere'' philosophy [4]. This is accomplished by providing a JVM interpreter and runtime support for each platform [5][6]. In theory, any platform that supports the Java runtime environment will produce the same
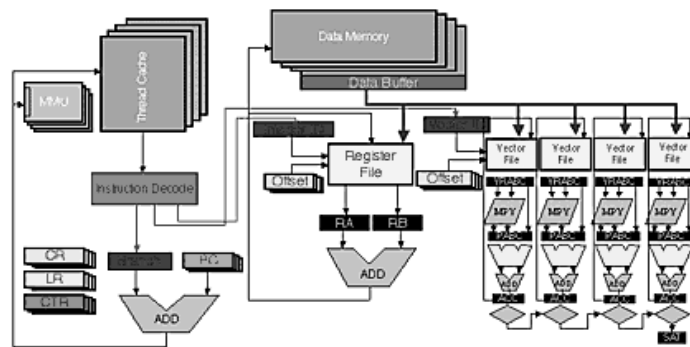
Figure 1.   *Multithreaded DSP*

execution results independent of the platform. Due to its characteristics and possibilities, Java has been extensively used as a programming language of choice. Java may become the dominant programming paradigm for 3G systems. NTT DoCoMo recently rolled out Java-based services for its cellular subscribers and hardware solutions for efficient Java execution are being proposed [7].

Fourth, unlike many past developments, hardware designers will need to understand the complexities of software systems so that compilation techniques can be effective. With a large number of standards both existing and proposed for wireless communications, a programmable platform will be required for timely implementation.

Fifth, embedded and DSP wireless applications have distinct requirements when compared with general purpose processors [8]. The predominant algorithmic difference is that inner loops are easily described as vectors of moderate length. A key point is that the native datatype is often fixed-point fraction. This is in distinct contrast to general purpose processors (and most high-level languages) which operate on integer datatypes.

Finally, in addition to algorithmic differences, most convergence devices will be deployed in embedded environments where real-time constraints are prevalent. Real-time

behavior has a dominant influence in the design of these devices [9]. Whereas general-purpose applications can often manage with variable latency response, convergence applications, in contrast, should be able to precisely guarantee the latencies within the system.

2. SANDBLASTER MULTI-THREADED DSP

Sandbridge Technologies has designed a multi-threaded processor capable of executing DSP, Control, and Java code in a single compound instruction set optimized for handset radio applications. The Sandbridge design overcomes the deficiencies of previous approaches by providing substantial parallelism and throughput for high-performance DSP applications while maintaining fast interrupt response, high-level language programmability, and very low power dissipation.

As shown in Figure 1, the design includes a unique combination of modern techniques such as a SIMD Vector/DSP unit, a parallel reduction unit, and RISC-based integer unit. Instruction space is conserved through the use of compounded instructions that are grouped into packets for execution. The resulting combination provides for efficient Control Code, DSP, and Java processing execution.
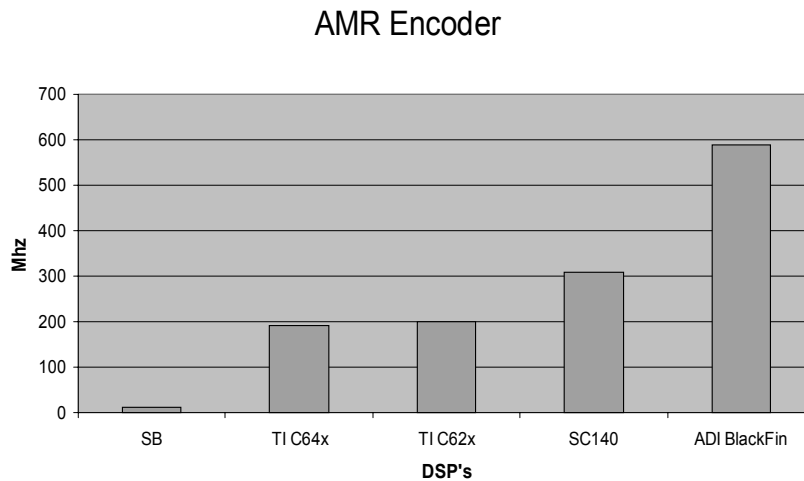
AMR Encoder

Figure 2.   AMR out-of-the-box Compiled Encoder Performance

Java Execution

Future 3G wireless systems will make significant use of Java. A number of carriers are already providing Java-based services [7]. JVM translation designers have used both software and hardware methods to execute Java bytecode. The advantage of software execution is flexibility. The advantage of hardware execution is performance. The Delft-Java architecture, designed in 1996, introduced the concept of dynamic translation of Java code into a multithreaded RISC-based machine with Vector SIMD DSP operations [11][12]. Another of the authors also explored dynamic translation [13]. The important property of Java bytecode that facilitated this translation is the statically determinable type state [10]. The Sandbridge approach is a unique combination of both hardware and software support for Java execution. Special instruction set support is provided to allow for efficient translation of Java code into the Sandblaster[TM] instruction set.

3. SOFTWARE COMPILATION

Programmer productivity is one of the major concerns in complex DSP applications. Because most classical DSPs are programmed in assembly language, it takes a very large software effort to program an application. For modern speech coders, [9] for example, it may take up to nine months or more before the application performance is known. Then, an intensive period of design verification ensues. If efficient compilers for DSPs were available, significant advantages in software productivity could be achieved.

Sandbridge Technologies has developed its own highly optimizing compiler. Software compilation enables the efficient translation of high-level language such as C/C++ into optimized machine language. A unique aspect of the Sandbridge compiler is that DSP operations are automatically generated. The type mismatch between modulo C arithmetic and fractional saturating DSP arithmetic is automatically recognized and the proper assembly language is generated. In addition, parallelism is automatically extracted by the compiler and executed in the parallel Vector/SIMD unit. Of
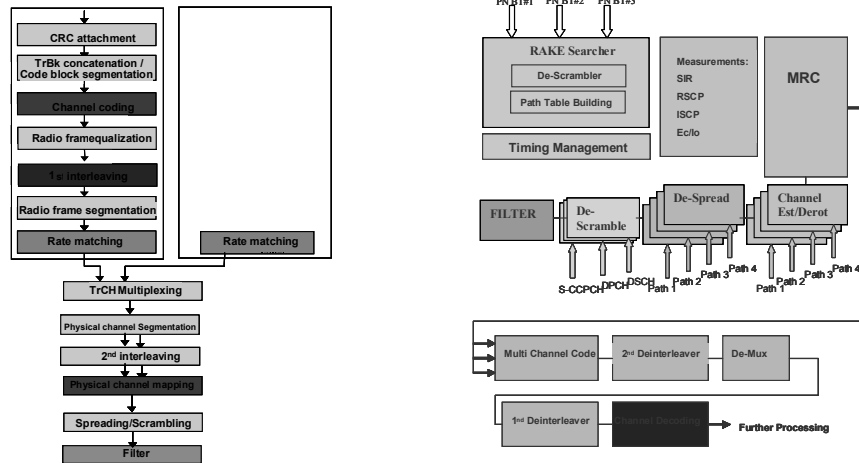
CRC attachment

TrBk concatenation / Code block segmentation

Channel coding

Radio frameequalization

1 st interleaving

Radio frame segmentation

Rate matching

Rate matching

TrCH Multiplexing

Physical channel Segmentation

2nd interleaving

Physical channel mapping

Spreading/Scrambling

Filter

PN BT#1   PN BT#2   PN BT#3

RAKE Searcher

De-Scrambler

Path Table Building

Measurements:
SIR
RSCP
ISCP
Ec/Io

MRC

Timing Management

FILTER

De-Scramble

De-Spread

Channel Est/Derot

S-CCPCH DPCH DSCH Path 1 Path 2 Path 3 Path 4   Path 1 Path 2 Path 3 Path 4

Multi Channel Code   2nd Deinterleaver   De-Mux

1nd Deinterleaver   Channel Decoding   → Further Processing

Figure 3.   WCDMA Transmission System

particular importance is the unique bit-exact preserving parallelization of non-associative saturating arithmetic.

Figure 2 shows the results of various compilers on out-of-the-box ETSI C code. The y-axis shows the number of MHz required to compute frames of speech in real-time. The AMR code is completely unmodified and no special include files are used. Without using any compiler techniques such as intrinsics or special typedefs, the Sandbridge compiler is able to achieve real-time operation on the Sandblaster™ core at hand-coded assembly language performance levels. Note that it is completely compiled from high-level language.

4. 2MBPS WCDMA TRANSMISSION SYSTEM

Previous communications systems have been developed in hardware due to the high computational requirements. DSP's in these systems have been limited to speech coding and orchestrating the custom hardware blocks. In high-performance 3G systems there may be over 2 million logic gates to implement the system. A complex 3G system may also take many months to implement. After logic design is complete,

any errors in the design may cause up to a 9 month delay is correcting and refabricating the device. This labor intensive process is counter productive to fast handset development cycles. The Sandbridge design takes a completely new approach to communications system design.
Sandbridge Communications Design

Rather than designing custom blocks for every function in the transmission system, Sandbridge has implemented a processor capable of executing operations appropriate to broadband communications. The small and power efficient core is then highly optimized and replicated to provide a platform for broadband communications. This approach scales well with semiconductor generations and allows flexibility in configuring the system for future specifications and any field modifications that may be necessary.

The Sandbridge process is to design the communications system in Matlab thus ensuring the bit and block error rates for the transmission system are achieved. The Matlab system design is then ported to fixed point C code. From that point, no further programmer intervention is required. The Sandbridge highly optimizing compiler extracts the parallelism and DSP operations and optimizes performance on the SandBlaster™ DSP.
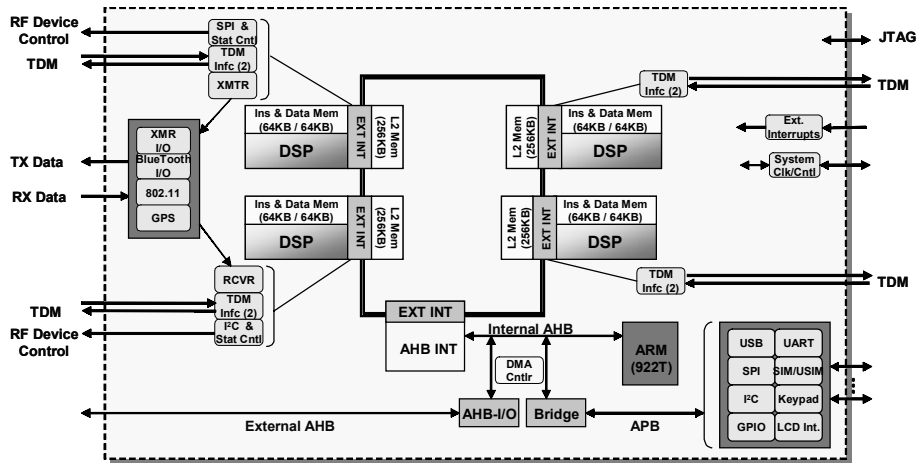
Figure 4.   SB9600 Chip

Sandbridge Technologies has developed complete C code for the UMTS WCDMA physical layer standard. Using our internally developed compiler on our own algorithms, Sandbridge has achieved real-time performance on a 768kbps transmit chain and a 2Mbps receive chain which includes all the blocks shown in Figure 3. The entire transmit chain including bit, symbol, and chip rate processing requires less than 400MHz of processor capacity to sustain a 768 kbps transmit capability.

5. SB9600 HANDSET SDR PRODUCT

Figure 4 shows the SB9600^{TM} baseband chip. It contains multiple SandBlaster™ cores and an ARM microcontroller. The performance of the SB9600^{TM} is more than sufficient to sustain a 2Mbps 3G transmission in real time. Initial test silicon of the SandBlaster™ core is available and will be integrated into the multicore SB9600^{TM}.

6. SUMMARY

Sandbridge Technologies has introduced a completely new and scalable design methodology for implementing multiple transmission systems on a single chip. Using a unique multithreaded architecture specifically designed to reduce power consumption, efficient broadband communications operations are executed on a programmable platform. The processor uses completely interlocked instruction execution providing software compatibility among all processors. Because of the interlocked execution, interrupt latency is very short. An interrupt may occur on any instruction boundary including loads and stores. This is critical for real-time systems.

The processor is combined with a highly optimizing compiler with the ability to analyze programs and generate DSP instructions. This obviates the need for assembly language programming and significantly accelerates time-to-market for new transmission systems.

To validate our approach, we designed our own 2Mbps WCDMA physical layer. First, we designed a MATLAB implementation to ensure conformance to the 3GPP specifications. We then implemented the algorithms in fixed point C code and compiled them to our platform using our internally developed tools. The executables were then simulated on our cycle accurate simulator that runs at up to 100 million SandBlaster™ instructions per second on a high end Pentium thereby ensuring complete logical operation. Having designed our own 3GPP

compliant RF front end, we execute complete RF to IF to baseband and reverse uplink processing in our lab. Our measurements confirm that our WCDMA design will execute within field conformance requirements in real time completely in software on the SB9600™ platform.

7. REFERENCES

[1] M. Gagnaire, **An Overview of Broad-Band Access Technologies**, *Proceedings of the IEEE*, Vol. 85, Issue 12, December, 1997, pp. 1958-1972.

[2] T. Cantrell, **DSP Doings**, *Circuit Cellar Ink*, No. 109, August, 1999, pp76-82.

[3] James Gosling, Bill Joy, and Guy Steele, editors. *The Java Language Specification*. The Java Series. Addison-Wesley, Reading, MA, USA, 1996.

[4] James Gosling and Henry McGilton. **The Java Language Environment**: A White Paper. Technical report, Sun Microsystems, Mountain View, California, October 1995.

[5] Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification*. The Java Series. Addison-Wesley, Reading, MA, USA, 1997.

[6] T. Lindholm and F. Yellin, **"Inside the Java Virtual Machine"**, *Unix Review*, Vol. 15, No. 1, pp. 31-39, January, 1997.

[7] Junko Yoshida. **Java chip vendors set for cellular skirmish**. EE Teims, January 30, 2001.

[8] P. Lapley. *DSP Processor Fundamentals*. IEEE Press, New York, 1997.

[9] M. Saghir, P. Chow, and C. G. Lee. **Towards Better DSP Architecture and Compilers**. In *Proceedings of the International Conference on Signal Processing Applications and Technology*, pages 658-664, October, 1994.

[10] J. Gosling, **"Java Intermediate Bytecodes"**, *ACM SIGNPLAN Workshop on Intermediate Representation (IR95)*, pp. 111-118, January, 1995.

[11] J. Glossner and S. Vassiliadis, **"The Delft-Java Engine: An Introduction"**, *Lecture Notes in Computer Science*. Third International Euro-Par Conference (Euro-Par '97), pp 776-770, Passau, Germany, August, 1997.

[12] J. Glossner and S. Vassiliadis, **"Delft-Java Dynamic Translation"**, *Proceedings of the 25th EUROMICRO conference (EUROMICRO '99)*, Milan, Italy, September, 1999.

[13] K. Ebcioglu, E. Altman, and E. Hokenek, **"A Java ILP Machine Based on Fast Dynamic Compilation"**, *IEEE MASCOTS International Workshop on Security and Efficiency Aspects of Java*, Eilat, Israel, January, 1997.

AMR Results based on out-of-the-box C code. C64x IDE Version 2.0.0 compiled without intrinsics using -k -q -pm -op2 -o3 -d"WMOPS=0" -ml0 -mv6400 flags with results averaged over 425 frames of ETSI supplied test vectors. C62x IDE Version 2.0.0 compiled without intrinsics using -k -q -pm -op2 -o3 -d"WMOPS=0" -ml0 -mv6200 flags with results averaged over 425 frames of ETSI supplied test vectors. Starcore SC140 IDE version Code Warrior for StarCore version 1.5, Relevant Optimization Flags (Encoder Only): scc -g -ge -be -mb -sc -O3 -Og Other: No Intrinsic Used. Results based on execution of 5 frames. ADI Blackfin IDE Version 2.0 and Compiler version 6.1.5 compiled without intrinsics using -O1 -ipa -DWMOPS=0 –BLACKFIN with results averaged over 5 frames of ETSI supplied test vectors for the encoder only portion.