A MULTITHREADED PROCESSOR ARCHITECTURE FOR SDR

John Glossner, Tanuj Raja, Erdem Hokenek, and Mayan Moudgill

Sandbridge Technologies, Inc.
White Plains, NY
glossner@SandbridgeTech.com
914-287-8500

Abstract

In this paper we discuss a multithreaded baseband processor capable of executing all physical layer processing of high data rate communications systems completely in software. We discuss the enabling technology for a software defined radio approach and present results for GPRS, 802.11b, and 2Mbps WCDMA. All of these protocols can be executed in real-time on the SB9600 chip using the Sandblaster core.

INTRODUCTION

Ever increasing complexities of mobile terminals combined with the desire to generate many handset models with increasing features have led to the adoption of a Software Defined Radio (SDR) based approach in the wireless industry. The previous generation of mobile terminals was primarily designed for use in geographically restricted areas. Growth for the wireless industry was dependant upon signing up new users. This has clearly changed. ARPU (Annual Revenue per User) for wireless carriers has been on a steady decline. The penetration levels in countries such as South Korea, Japan and other south Asian countries is high and new revenue streams (from technologies such has 3G) have been slow to materialize due to lack of sufficient mobile terminals. True convergence of multimedia, cellular, location and connectivity technologies is expensive, time consuming, and complex at all levels of development for not only mobile terminals, but infrastructure as well. Moreover, the standards themselves have failed to converge which has led to multiple market segments. In order to maintain market share a handset development company must use dozens of combinations of handsets. This requires the handset companies to support multiple platforms and multiple hardware solutions from multiple 3rd parties.

The current hardware based solutions require about 18 months for commercial System on a Chip implementations (SOCs) and another 9 to 12 months for successful handset development with no ability to change, update or modify the phone's feature set. Moreover, in the case of multimode baseband modems, there is no ability to quickly verify updates or add new functionality because any modification requires a redesign of the SOC. This is costly and a time consuming proposition. This requires both the wireless carriers and handset developers to anticipate end user requirements and standards evolution up to three years in advance of product introduction. Clearly, this is not a productive solution. In fact the entire wireless supply chain from end user to

component supplier in a traditional hardware based solution requires an unacceptable turn around time for new features, updates, and functionality.

SDR enabling technologies answer these problems at multiple levels, for end users, it provides:

- Mobile Terminal Independence with the ability to "choose" desired feature sets.
- Global connectivity with ability to roam across operators
- Future Scalability and a longer lifetime of the handset.

Wireless carriers growth comes from the ability to differentiate their offerings from their competition and the possibility constantly adding new services. This will in return increase their ARPU.

Infrastructure providers can reduce cost by embracing a reconfigurable platform which addresses the need to supply multiple solutions for all prevalent standards. It also provides them the capability to upgrade services, add features, and enhance security – all done through software.

Most importantly, the handset developers can use the SDR approach to develop terminals quickly, provide a rich set of features, and address multiple markets with a single platform. By utilizing the low cost, reconfigurable SDR approach, the entire supply chain can be revitalized, with the end-users finally getting the performance and features that have been desired for many years.

SDR Definition

So, what is SDR? Various definitions for SDR exist today. As defined by the SDR Forum, "the term software defined radios (SDRs) is used to describe radios that provide software control of a variety of modulation techniques, wide-band or narrow-band operation, communications security functions (such as hopping), and waveform requirements of current and evolving standards over a broad frequency range.[1]" While there are different levels reconfigurability within the SDR-based approach; true reconfigurability comes from the ability to define the entire physical layer of the communication systems in software. In order to effectively realize this goal, a number of challenges exist for a SDR solution.

Sandbridge SB9600 Product

Power dissipation constraints require new techniques at every stage of design - architecture, micro-architecture, software, algorithm design, logic design, circuit design, and process design. With performance requirements exploding as bandwidth demand increases, power conscious design becomes more difficult. System-on-a-chip integration and low voltage process technologies will contribute to lower power SOC integrated circuits (ICs) but are insufficient as the only solution for streaming multimedia.

A large number of increasingly complex standards with ever increasing communication rates require increasing processing capabilities. High level programming for these extremely complex systems is needed to reduce development time and realize time to market goals.

Based on the Sandblaster™ baseband processor core, the SB9600 SOC is designed to address these challenges. The SOC is reprogrammable and is delivered with a complete software development kit

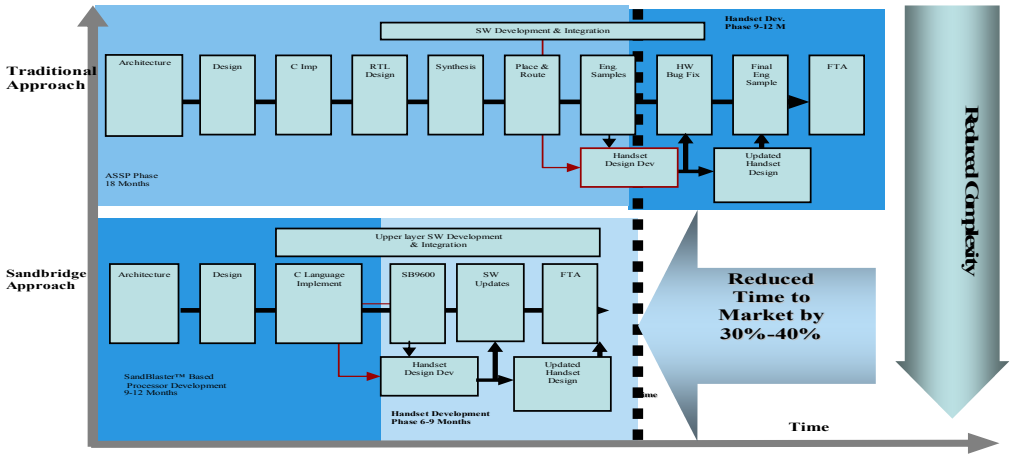

Figure 1. SDR Development Cycle

including the Sandblaster™ IDE (integrated Development Environment). Handset developers can use the development environment to program multiple physical layers. The platform allows the freedom to choose a particular set of operating modes (such as GSM, GPRS, WCDMA, etc.) and the desired local area connectivity (Bluetooth, Wireless LAN, etc). Quickly implementing a variety of standards and any desired functionality reduces development time and effort. It also enables handset developers to realize constrained time to market goals.

By using the SB9600 platform, development time may be reduced by more than 40%. Subsequent updates and additional features can be added with very little additional effort. For example, a new physical layer algorithm may be added simply by programming the algorithm in C, compiling and simulating it using the IDE and simply downloading it to the SOC by using the very same integrated design environment. This is valid for any subsystems or complete systems (of the physical layer).

Figure 1 illustrates a comparison between a traditional hardware approach and the SB9600 development approach. In the traditional approach, the SOC development effort requires 18 months to achieve successful delivery of commercial samples. While some of the handset prototype development can be done in parallel, it still requires 9 to 12 months for completing the prototype. This is primarily due to the long verification cycles (SOC verification after system integration) and the need to do hardware modifications.

In the Sandbridge approach, the system level validation and handset prototype verification can start the moment optimized C Code for the physical layer and the associated protocol stacks is available. In fact, if desired, the prototype development platform and the physical layer programming can be done in parallel, enabling simultaneous verification, type approval testing, and field testing. So, by conservative estimates the development time can be reduced by 30% to 40%. In addition, once the engineering team is familiar with the platform, the complexity of dealing with a new platform for each

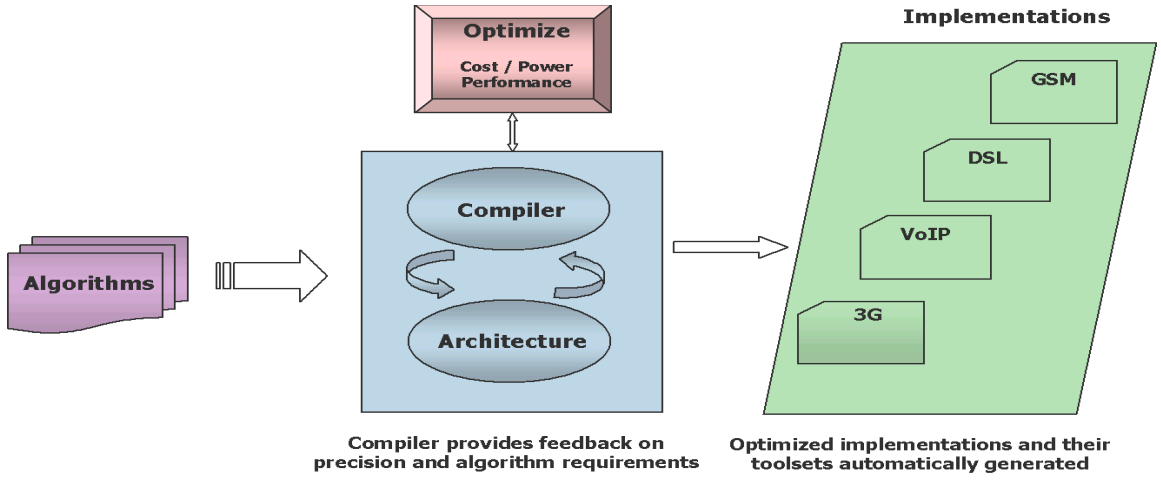

Figure 2. Compiler / Processor Development

handset is reduced. Now, there is a single platform for multiple solutions and multiple standards.

So, how is it possible? The Sandblaster™ DSP was designed specifically for the communications market; it utilizes low power techniques, provides extremely high processing capability, and requires only high level C programming (no assembly programming required).

A complete software implementation of the physical layer allows the best time-to-market. No hardware accelerators are required even for high data rates such as 2Mps WCDMA.

SANDBLASTER™ MULTI-THREADED BASEBAND PROCESSOR

It is well recognized that the best way to design a DSP compiler is to develop it in parallel with the DSP architecture [2]. As shown in Figure 2, an ideal process would consider the application domain of programs to be executed. Then, with significant influence for automated software generation, architecture trade-offs should be considered. Under the constraint that the resulting system must be programmed in a high-level language such as C or Java, additional trade-offs in the architecture are made subject to price, performance, and power constraints. Once feedback on all the parameters is acceptable, implementation-specific solutions may be generated for various performance levels. Sandbridge has used exactly this approach in developing the Sandblaster processor core.

Real-time considerations

In most physical layer processing there is a large amount of digital signal processing that must be performed. Embedded systems and DSP systems have constraints that are different than the constraints typically found in general purpose programming systems.

Execution predictability in DSP systems often precludes the use of many general-purpose design techniques (e.g. speculation, branch prediction, data

caches, etc.). Instead, classical DSP architectures have developed a unique set of performance enhancing techniques that are optimized for their intended market. These techniques are characterized by hardware that supports efficient filtering, such as the ability to sustain three memory accesses per cycle (one instruction, one coefficient, and one data access). Sophisticated addressing modes such as bit-reversed and modulo addressing may also be provided. Multiple address units operate in parallel with the datapath to sustain the execution of the inner kernel.

In classical DSP architectures, the execution pipelines were visible to the programmer and necessarily shallow to allow assembly language optimization. This programming restriction encumbered implementations with tight timing constraints for both arithmetic execution and memory access. The key characteristic that separates modern DSP architectures from classical DSP architectures is the focus on compilability. Once the decision was made to focus the DSP design on programmer productivity, other constraining decisions could be relaxed. As a result, significantly longer pipelines with multiple cycles to access memory and multiple cycles to compute arithmetic operations could be utilized. This has yielded higher clock frequencies and higher performance DSPs.

In an attempt to exploit instruction level parallelism inherent in DSP applications, modern DSPs tend to use VLIW-like execution packets. This is partly driven by real-time requirements which require the worst-case execution time to be minimized. This is in contrast with general purpose CPUs which tend to minimize average execution times. With long pipelines and multiple instruction issue, the difficulties of attempting assembly language programming become apparent. Controlling instruction dependencies between upwards of 100 in-flight instructions is a non-trivial task for a programmer. This is exactly the area where a compiler excels.

The challenges of using VLIW DSP processors, however, include large program executables (code bloat) that results from independently specifying every operation with a single instruction, interrupt response latency due to visible memory pipeline effects in highly parallel inner loops, and requiring multiple high-bandwidth paths with multiple write ports to both memory and registers resulting in unacceptable power dissipation for handset applications.

Sandblaster Core Technology

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 3, the design includes a unique combination of modern techniques such as a SIMD Vector/DSP unit, a parallel reduction unit, and a RISC-based integer unit. Each processor core provides support for concurrent execution for up to eight threads of execution. All state may be saved from each individual thread and no special software

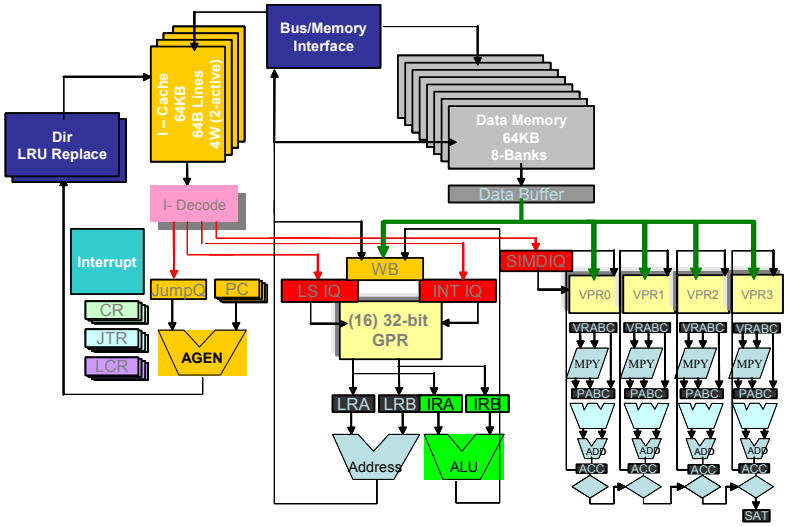

Figure 3. Sandblaster Baseband Processor Core

support is required for interrupt processing. The machine is partitioned into a RISC-based control unit that fetches instructions from a set-associative instruction cache. Instruction space is conserved through the use of compounded instructions that are grouped into packets for execution.

The cache relieves the programmer of moving large programs into SRAM and avoids overlays that burden software systems. A cache also has the advantage that a programmer need only concern themselves with working set size (e.g. the dynamic code that predominantly executes) rather than the static instruction size that resides in flash or is downloaded dynamically over the air.

The data memory does not use a cache because in most broadband communications systems the data is streamed from A/D converters and passed on for further processing. Analogous to the instruction memory, the data memory also has eight independent banks for concurrent access by each thread. The complete memory system is unified to allow easy software access to any thread data.

Special care has been taken in the design of the memory subsystem to reduce power dissipation. The pipeline design in combination with the memory design ensures that all memories are single ported and yet the processor can sustain nearly 4 taps per cycle (the theoretical maximum) in every thread unit simultaneously.

A RISC-based execution unit, depicted in the center of Figure 3, assists with control processing. Physical layer processing often consists of control structures with compute-intensive inner loops. A baseband processor must deal with both integer and fractional datatypes. For the control code, a 16 entry, 32-bit register file per thread unit provides for very efficient control processing. Common Integer datatypes are typically stored in the register file. This allows for branch bounds to be computed and addresses to be efficiently generated.

Intensive DSP physical layer processing is performed in the SIMD/Vector unit depicted on the right side of Figure 3. Each cycle, a 4x16-bit vector may be loaded into the register file while two vectors are being multiplied, saturated, reduced (e.g. summed), and saturated again. The branch bound may also be computed and the instruction looped on itself until the entire vector is processed. This may be specified in as little as 64-bits. This compares very favorably to VLIW implementations.

An important power consideration is that the Vector File contains a single write port for threads. This is in distinct contrast to VLIW implementations that must specify an independent write port for each VLIW instruction (often up to 256-bits with 4 or more write ports). Since write ports contribute significantly to power dissipation, minimizing them is an important consideration in handset design.

Parallelism

To enable physical layer processing in software, the processor supports many levels of parallelism. Thread-level parallelism is supported by providing hardware support for up to 8 independent programs to be simultaneously active on a single Sandblaster core. This minimizes the latency in physical layer processing. Since many algorithms have stringent requirements on response time, multithreading is an integral technique in reducing latencies.

In addition to thread-level parallelism, the processor also supports data-level parallelism through the use of a Vector unit. In the inner kernel of signal processing or baseband routines, the computations appear as vector operations of moderate length. Filters, FFTs, convolutions, etc., all can be specified in this manner. Efficient, low power support for data level parallelism effectively accelerates inner loop signal processing.

To accelerate control code, the processor supports issuing multiple operations per cycle. Since control code often limits overall program speed-up (e.g. Amdahl's Law), it is helpful to allow control code and vector code to be overlapped. This is provided through a compound instruction set. The Sandblaster core provides instruction level parallelism by allowing multiple operations to issue in parallel. Thus, a branch, an integer, and a vector operation may all issue simultaneously per thread unit. In addition, many compound operations are specified within an instruction class such as load with update, and branch with compare.

Finally, the SB9600 product includes four processor cores per chip to provide enough computational capability to execute the entire WCDMA baseband processing in software in real-time.

Java Execution

Future 3G wireless systems will make significant use of Java. A number of carriers are already providing Java-based services and may require all 3G systems to support Java[13]. Java is a C++ like programming language designed for general-purpose object-oriented programming [14]. An appeal for the usage of such a language is its ``write once, run anywhere" philosophy [15]. This is accomplished by providing a Java Virtual Machine (JVM) interpreter and runtime support for each platform[16].

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 [17][18]. Another of the authors also explored dynamic translation [19]. The important property of Java bytecode that facilitated this translation is the statically determinable type state [14]. The

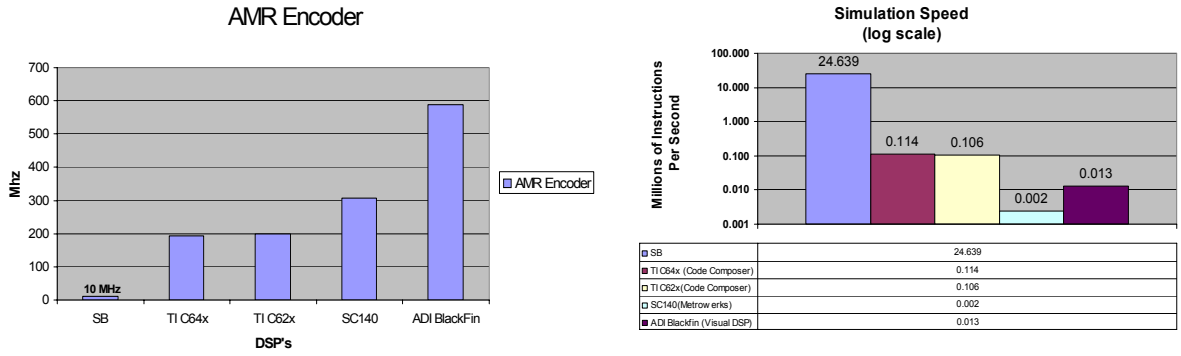

Figure 4. Out-of-the-box AMR encoder native and simulation performance results

Sandbridge approach is a unique combination of both hardware and software support for Java execution.

Interrupts

A challenge of visible pipeline machines (e.g. most DSPs and VLIW processors) is interrupt response latency. Visible memory pipeline effects in highly parallel inner loops (e.g. a load instruction followed by another load instruction) are not typically interruptible because the processor state can not be restored. This requires programmers to break apart loops so that worst case timings and maximum system latencies may be acceptable. This convolutes the source code and may even require source code changes between processor generations.

The Sandblaster core allows any instruction from any thread to be interrupted on any processor cycle. This is critically important to real-time constraints imposed by physical layer processing. The processor also provides special hardware support for a specific thread unit to interrupt another thread unit with very low latency. This low-latency cross-thread interrupt capability enables fast response to time critical events.

SOFTWARE TOOLS

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, [3] 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.

However, there are a number of issues that must be addressed in designing a DSP compiler including fundamental mismatches in datatypes, non-associative saturating arithmetic, and supercomputer class optimizations [8][9][10][11]. Partial solutions to these problems have included building large libraries, language extensions [4][5], intrinsic functions [6][7], and raw assembly language coding.

A unique aspect of the Sandbridge compiler is that DSP operations are automatically generated. Sandbridge compilers use a technique called semantic analysis. In semantic analysis, a sophisticated compiler must search for the meaning

of a sequence of C language constructs. A programmer writes C code in an architecture independent manner - such as for a micro controller - focusing primarily on the function to be implemented. If DSP operations are required, the programmer implements them using standard modulo C arithmetic. The Sandbridge compiler analyzes the C code, automatically extracts the DSP operations and synthesizes optimized DSP code without the excess operations required to specify DSP arithmetic in C code. This technique has a significant software productivity gain over intrinsic functions.

In addition, Sandbridge has implemented supercomputer class optimizations in its vectorizing compiler. The compiler is efficient at extracting data level parallelism and also handles the difficult problem of outer loop vectorization which is often a requirement for inner loop optimizations.

Figure 4A 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. With the lone exception of turning off the WMOPS profiler in the downloaded C code, 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.

Efficient compilation is just one aspect of software productivity. Figure 4B shows the post-compilation simulation performance of the AMR encoder for a number of DSP processors. All programs were executed on the same 1GHz Pentium laptop computer. The Sandbridge tools are capable of simulating 25.6 Million instructions per second for the optimized AMR encoder. This is more than *two orders of magnitude faster* than the nearest competitor. We achieved this by using our own compilation technology to accelerate the simulation.

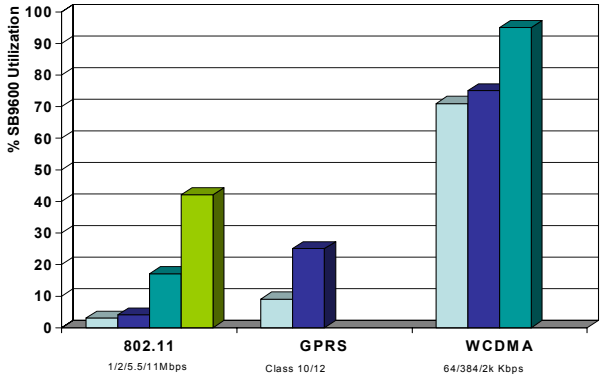

Figure 5. Baseband Communications System Performance

COMMUNICATIONS DESIGN

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 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 in 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.

Rather than designing custom hardware blocks for every function in the transmission system, Sandbridge has implemented a processor capable of executing operations appropriate to broadband communications in software.

Figure 5 shows the performance requirements for 802.11, GPRS, and WCDMA as a function of the SB9600 chip utilization for a number of different transmission rates. Providing processing capability for 2Mbps WCDMA FDD-mode also provides sufficient processing capability for 802.11b and even concurrent capacity for multiple communications systems.

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 software defined programmable platform. 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, GPRS, and 802.11b physical layers 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. We executed the same compiler generated program on engineering samples of our Sandblaster core. 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.

REFERENCES

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

[2] 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.

[3] European Telecommunications Standards Institute, Digital cellular telecommunications system, **ANSI-C code for the GSM Enhanced Full Rate (EFR) speech codec** (GSM 96.53), March, 1997, ETS 300 724.

[4] K.W. Leary and W. Waddington, **"DSP/C: A Standard High Level Language for DSP and Numeric Processing"**, *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, IEEE, 1990, pp. 1065-1068.

[5] B. Krepp, "**DSP-Oriented extensions to ANSI C"**, Proceedings of the International Conference on Signal Processing Applications and Technology (ICSPAT '97), DSP Associates, 1997, pp. 658-664.

[6] D. Batten, S. Jinturkar, J. Glossner, M. Schulte, and P. D'Arcy, **"A New Approach to DSP Intrinsic Functions"**, *Proceedings of the Hawaii International Conference on System Sciences*, Hawaii, January, 2000.

[7] D. Batten, S. Jinturkar, J. Glossner, M. Schulte, R. Peri, and P. D'Arcy, **"Interaction Between Optimizations and a New Type of DSP Intrinsic Function"**, *Proceeding of the International Conference on Signal*

Processing Applications and Technology (ICSPAT '99), Orlando, Florida, November, 1999.

[8] A. Aho, R. Sethi, and J. Ullman, *"Compilers: Principles, Techniques and Tools"*, Addison-Wesley Publishing Company, CA, 1986.

[9] M. Lam, **"Software Pipelining: An effective scheduling technique for VLIW Machines"**, In *Proceedings of the SIGPLAN '88 Conference on Programming Language Design and Implementation*, Atlanta, GA, June, 1988.

[10] S. Jinturkar, J. Thilo, J. Glossner, P. D'Arcy, and S. Vassiliadis, **"Profile Directed Compilation in DSP Applications"**, *Proceedings of the International Conference on Signal Processing Applications and Technology (ICSPAT'98)*, September, 1998.

[11] W. Hwu, **"Super block: An effective technique for VLIW and superscalar compilation"**, *Journal of Supercomputing*, Volume 7, pp. 229-248.

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

[13] J. Yoshida, **"Java chip vendors set for cellular skirmish"**, *EE Times*, January 30th, 2001.

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

[15] J. Gosling and H. McGilton, **"The Java Language Environment: A White Paper"**, *Sun Microsystems Press*, October, 1995.

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

[17] 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.

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

[19] 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.

John Glossner is CTO & EVP of Engineering at Sandbridge Technologies. Prior to co-founding Sandbridge, John managed the Advanced DSP Technology group, Broadband Transmission Systems group, and was Access Aggregation Business Development manager at IBM's T.J. Watson Research Center. Prior to IBM, John managed the software effort in Lucent/Motorola's Starcore DSP design center. John received a Ph.D. in Computer Architecture from TU Delft in the Netherlands for his work on a Multithreaded Java processor with DSP capability. He also received an M.S. degree in Engineering Management and an M.S.E.E. from NTU. John also holds a B.S.E.E. degree from Penn State. John has more than 40 publications and 12 issued patents.

Mr. Tanuj Raja is Vice President of Business Development. With more ten years of industry experience, Tanuj joined Sandbridge Technologies in February, 2002. Prior to Sandbridge Tanuj spent four years in the Wireless Design group at Cadence Design Systems, Inc. where he held positions as Director of Business Development, Worldwide Business Manager for 3G Technologies and 3G Program Manger. Previously he was a Project Manager, leading SOC Digital Design projects in Design Services group at Cadence. He also has five years of software development experience. Tanuj Raja received his bachelors degree in Electrical Engineering from Northeastern University.

Erdem Hokenek received BS and MS degrees from Technical University, Istanbul (Turkey) and PhD from Swiss Federal Institute of Technology (ETH Zurich, Switzerland). After his PhD in 1985, he joined IBM T. J. Watson Research Center where he

worked on the advanced development of POWER and PowerPC processors for the RS/6000 Workstations. He also worked in various technical and management positions on the high performance compilable DSP and Cross Architecture Translations. He is co-founder of Sandbridge Technologies Inc.

Mayan Moudgill obtained a Ph.D. in Computer Science from Cornell University in 1994, after which he joined IBM at the Thomas J. Watson Research Center. He worked on a variety of computer architecture and compiler related projects, including the VLIW research compiler, Linux ports for the 40x series embedded processors and simulators for the Power 4. In 2001, he co-founded Sandbridge Technologies, a start-up that is developing digital signal processors targeted at 3G wireless phones.