The Sandbridge SB3011 SDR platform

John Glossner, Daniel Iancu, Mayan Moudgill, Gary Nacer, Sanjay Jinturkar, and Michael Schulte
Sandbridge Technologies, Inc.

***Abstract*— This paper describes the Sandbridge Sandblaster real-time software defined radio platform. Specifically we describe the SB3011 system on a chip multiprocessor. We describe the software development system that enables real-time execution of communications and multimedia applications. We provide results for a number of interesting communications and multimedia systems including UMTS, DVB-H, WiMAX, WiFi, and NTSC video decoding. All results presented are from completely implemented systems from RF through baseband.**

***Index Terms*— Multithreaded processors, signal processing, parallel processing, real-time systems.**

I. Introduction

BUILDING large parallel processing systems is a difficult task. Programming them efficiently is even more challenging. When non-associative Digital Signal Processing (DSP) arithmetic is included, the challenge of automated software development for a Chip Multiprocessor (CMP) system is amplified.

A Software Defined Radio (SDR) platform that is processor based is a CMP system. The SDR Forum [1] defines five tiers of solutions. Tier-0 is a traditional radio implementation in hardware. Tier-1, Software Controlled Radio (SCR), implements the control features for multiple hardware elements in software. Tier-2, Software Defined Radio (SDR), implements modulation and baseband processing in software but allows for multiple frequency fixed function RF hardware. Tier-3, Ideal Software Radio (ISR), extends programmability through the RF with analog conversion at the antenna. Tier-4, Ultimate Software Radio (USR), provides for fast (millisecond) transitions between communications protocols in addition to digital processing capability.

II. The SB3011 SDR Platform

The Sandbridge SDR platform is a Tier-2 implementation as defined by the SDR Forum. Figure 1 shows the SB3011 implementation. It is intended for handset markets. The main processing complex includes four DSPs [2] each running at a minimum of 600MHz at 0.9V. The chip is fabricated in 90nm technology. Each DSP is capable of issuing multiple operations per cycle including data parallel vector instructions. The microarchitecture of each DSP is 8-way multithreaded allowing the SB3011 to execute up to 32 independent instruction streams each of which may issue vector operations.

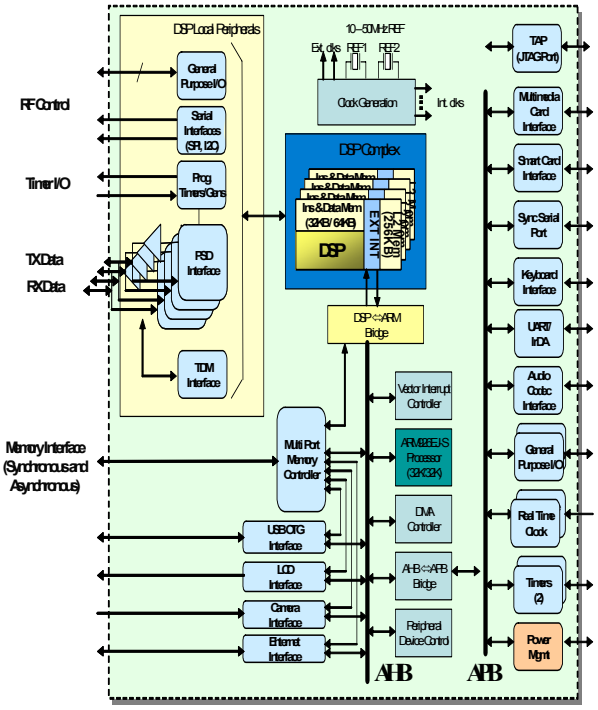

Figure 1. Sandblaster SB3011 Chip

Each DSP has an level-1 (L1) 32kB set-associative instruction cache and an independent L1 64kB data memory. In addition a global level-2 (L2) 1MB memory is shared among all processors. The implementation guarantees no pipeline stalls for L1 memory accesses. For external memory accesses or L2 accesses only the thread that issued the load request stalls. All other threads continue executing independent of which processor issued the memory request.

The Sandblaster DSP is a true architecture in the sense that from the programmer's perspective there are no inter-instruction dependencies and all instructions complete prior to the next instruction issuing – on a per thread basis.

In addition to the parallel multithreaded DSP compute complex, there is an entire ARM complex with all the peripherals necessary to implement input/output (I/O) devices

John Glossner, Ph.D. is CTO and Executive Vice President at Sandbridge Technologies, Inc. 1 North Lexington Ave., White Plains, NY, 10601, phone: +1 (914) 287 8500, email: jglossner@sandbridgetech.com.

Daniel Iancu, Ph.D. is Chief Communications Architect at Sandbridge Technologies.

Mayan Moudgill, Ph.D. is Chief Architect at Sandbridge Technologies.

Gary Nacer is V.P. of Engineering at Sandbridge Technologies.

Sanjay Jinturkar, Ph.D. is Director of Software at Sandbridge Technologies

Michael Schulte, Ph.D. is an Assistant Professor at UW Madison.

in a smart phone. The processor is an ARM926EJ-S running at up to 300MHz. Sandbridge has ported Linux to this platform and the processor functions as the Man-Machine Interface (MMI) for smart phone applications.

Using an AMBA Advanced High-Speed Bus (AHB) and Advanced Peripheral Bus (APB), the peripherals are able to support multiple concurrent data processing. Attached to the APB is a Multimedia Card (MMC) and Smart Card (SD Card) interface for connecting external Flash storage. Keyboard and mouse interfaces are included along with multiple UARTs and IRDA interfaces. AC-97 audio is directly supported on the APB bus.

A number of other general peripherals are supported on the APB including a real-time clock and general purpose timers which are used to keep system time.

The AHB is used to move high-speed data such as memory requests and video into the chip for processing or out of the chip for display or storage. A seamless connection to an LCD display is provided and can be accessed from either the ARM or DSP processors. Similarly, a high-speed seamless camera interface is provided to capture raw video data that may be encoded or processed in the DSP or ARM processors.

The SB3011 includes a full USB 2.0 on-the-go (OTG) implementation for seamless connection to printers, cameras, storage, or other USB devices. An ethernet interface is also included on the chip for wired local area network (LAN) connections.

External memory requests which both the ARM and DSPs can initiate are routed through a Multiport Memory Controller (MMC) attached to the AHB. The external memory can be synchronous or asynchronous. Typical memories include Flash, SDRAM, DRAM, and SRAM. The MMC supports multiple simultaneous requests whether generated through Direct Memory Access (DMA) devices both for the ARM and DSP (not shown) or directly by the processors (ARM and DSP). All memory is mapped into a 32-bit global address space and is shared among all processors.

In addition to the ARM peripherals, there are a number of DSP specific peripherals primarily intended to move RF or TDM voice data directly into the DSP's L2 memory. Multiple Parallel Streaming Data (PSD) Interfaces are provided for direct data movement into the DSP complex. These are generally used for gluess connection to Radio Frequency (RF) cards. A Time Division Multiplexed (TDM) interface serves a similiar function for seamless connection to T1/E1 interfaces. The TDM interface can support up to 2048 channels. Support for synchronization of transmitted or received data bursts is accomplished through the use of dedicated I/O timers. When configured, these timers can be operated with an external (system) clock source and are internally used to gate the DMA transfers on the PSD and TDM interfaces. This feature is important for slot-based communications systems such as GSM.

A number of interfaces are provided for general purpose control of external components typically found in smart phones. These include general purpose timers which can be used as external clocks, SPI and I2C buses which are common in RF control logic, and general purpose I/O. The SPI and I2C peripherals allow the DSPs to compute in software functions such as Automatic Gain Control (AGC) and send the information seamlessly to the RF control interface.

III. Power Management

To facilitate flexible system-level power management the Sandblaster SB3011 incorporates thirteen independent power domains. Five of the domains are for the ARM and the four Sandblaster cores. Each is independently controllable by the Device Power Management Unit (DPMU). The DPMU is a programmable peripheral that allows for the following options: 1) The ability to place the device in power down where all data and state of the processing unit is not maintained and 2) The ability to place the processing units in power down where each core does not maintain state but the L2 memories are back-biased and thus retain state.

In addition to the voltage control features, clock management is also included in two forms: 1) Instruction-based automatic clock enable/disable operation where the hardware dynamically controls clocks to internal sub-blocks when inactivity is detected and 2) Operating System (OS) or application-based clock enable/disable which can control DSP cores, AHB peripherals, LCD, Camera, USB, Ethernet, and APB peripherals.

While not comprehensive, some typical profiles of low power configurations include: 1) Device Deep Sleep where the all the SB3011 functional blocks are powered off with the exception of the Device Power Management Unit. No state is retained in this mode. 2) Processing Unit Deep Sleep Mode where all the processor cores are shut down without state retention. However, L2 memories and peripherals retain state and may function. 3) Device Standby where all DSP cores and the ARM processor clocks are disabled but full state is retained.

IV. Programming Model

Obtaining full utilization of parallel processor resources has historically been a difficult challenge. Much of the programming effort can be spent determining which processors should receive data from other processors. Often execution cycles may be wasted for data transfers. Statically scheduled machines such as Very Long Instruction Word architectures and visible pipeline machines with wide execution resources complicate programmer productivity by requiring manual tracking of up to 100 in-flight instruction dependencies. When non-associative DSP arithmetic is present, nearly all compilers are ineffective and the resulting burden falls upon the assembly language programmer. A number of these issues have been discussed in [3].

A. Multithreaded Programming Model

A good programming model should adequately abstract most of the programming complexity so that 20% of the effort may result in 80% of the platform utilization [4]. While there are

still some objections to a multithreaded programming model [6], to-date it is widely adopted particularly with the introduction of the Java programming language [5][7].

With hardware that is multithreaded with concurrent execution and adopting a multithreaded software programming model, it is possible for a kernel to be developed that automatically schedules software threads onto hardware threads. It should be noted that while the hardware scheduling is fixed and uses a technique called Token Triggered Threading (T^3) [8], the software is free to use any scheduling policy desired.

The Sandblaster kernel has been designed to use the POSIX pthreads open standard [9]. This provides cross platform capability as the library is compilable across a number of systems including Unix, Linux, and Windows.

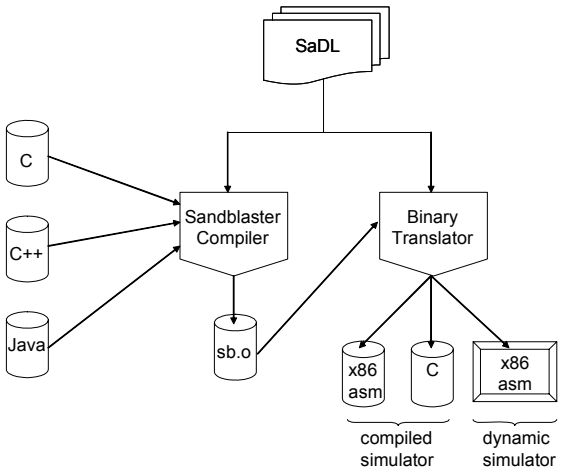

Figure 2. Sandblaster Tool Chain

B. Tools Generation

Figure 2 shows the Sandblaster® tool chain generation. The platform is programmed in a high-level language such as C, C++, or Java. The program is then translated using an internally developed supercomputer class vectorizing, parallelizing compiler. The tools are driven by a parameterized resource model of the architecture that may be programmatically generated for a variety of implementations and organizations. The source input to the tools, called the Sandbridge architecture Description Language (SaDL), is a collection of python source files that guide the generation and optimization of the input program and simulator. The compiler is retargetable in the sense that it is able to handle multiple possible implementations specified in SaDL and produce an object file for each implementation. The platform also supports many standard libraries (e.g. libc, math, etc.) that may be referenced by the C program. The compiler generates an object file optimized for the Sandblaster® architecture.

The tools are then capable of producing dynamic and static simulators. A binary translator/compiler is invoked on the host simulation platform. The inputs to the translator are the object file produced by the Sandblaster compiler and the SaDL description of the processor. From these inputs, it is possible to produce a statically compiled simulation file. If the host computer is an x86 platform, the translator may directly produce x86 optimized code. If the host computer is a non-x86 platform, the binary translator produces a C file that may subsequently be processed using a native compiler (e.g. gcc).

C. Compiler Technology

There are many challenges faced when trying to develop efficient compilers for parallel DSP technologies. At each level of processor design, Sandbridge has endeavored to alleviate these issues through abstraction. First and foremost, the Sandblaster processor is transparent in the architectural sense. This proscribes that there are no visible implementation effects for the programmer to deal with [10]. This is in distinct contrast with VLIW designs where the implementation is the architecture. A benefit of a true architecture approach is that object code will execute unmodified (e.g. without any translation required) on any Sandblaster compliant implementation.

The Sandblaster architecture uses a SIMD datapath to implement vector operations. The compiler vectorizes C code to exploit the data level parallelism inherent in signal processing applications and then generates the appropriate vector instructions. The compiler also handles the difficult problem of outer loop vectorization

Within the architecture there is direct support for parallel saturating arithmetic. Since saturating arithmetic is non-associative, out-of-order execution may produce different bit results. In some wireless systems this is not permissible [11]. By architecting parallel saturating arithmetic (i.e. vector multiply and accumulate with saturation), the compiler is able to generate code with the understanding that the hardware will properly produce bit-exact results. The compiler algorithm used to accomplish this is described in [12]. Some hardware techniques to implement this are described in [13].

Additionally, our compiler can also automatically generate threads. We use the same pthreads mechanism for thread generation in the compiler as the programmer who specifies them manually. For most signal processing loops it is not a problem to generate threads and the compiler will automatically produce code for correct synchronization.

V. Results

Figure 3 shows the results of a number of communications systems as a percentage utilization of an 600MHz SB3011 platform. Particularly, WiFi 802.11b, GPS, AM/FM radio, NTSC Video TV, Bluetooth, GSM/GPRS, UMTS WCDMA, WiMax, CDMA, and DVB-H. A notable point is that all these communications systems are written in generic C code with no hardware acceleration required. It is also notable that performance, accuracy, and concurrency can be dynamically adjusted based on the mix of tasks desired.

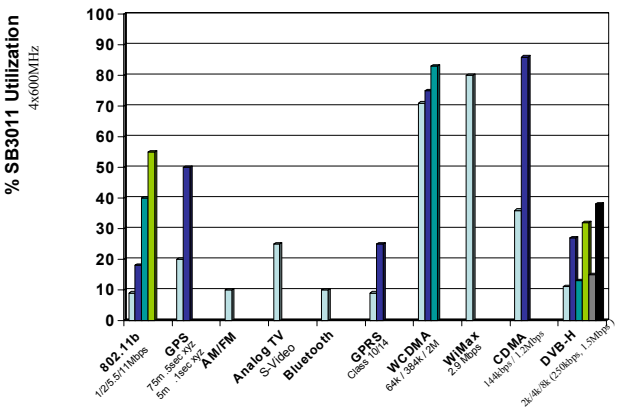

Figure 3. Communication Systems Results

VI. SUMMARY

Sandbridge Technologies has introduced a completely new and scalable design methodology for implementing multiple transmission systems on a single SDR 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 vectorizing compiler with the ability to automatically analyze programs and generate DSP instructions. The compiler also automatically parallelizes and multithreads programs. This obviates the need for assembly language programming and significantly accelerates time-to-market for streaming multimode multimedia convergence systems.

REFERENCES

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

[2] J. Glossner, T. Raja, E. Hokenek, and M. Moudgill, “A Multithreaded Processor Architecture for SDR,” The Proceedings of the Korean Institute of Communication Sci-ences, Vol. 19, No. 11, November, 2002, pp. 70-84.

[3] J. Glossner, M. Schulte, M. Moudgill, D. Iancu, S. Jinturkar, T. Raja, G. Nacer, and S. Vassiliadis, “Sandblaster Low-Power Multithreaded SDR Baseband Processor”, Proceedings of the 3rd Workshop on Applications Specific Processors (WASP’04), pp. 53-58, Stockholm, Sweden, September 7th, 2004.

[4] Goering, Richard, “Platform-based design: A choice, not a panacea”, EE Times, Sept. 11th, 2002. Available at http://www.eetimes.com/story/OEG20020911S0061.

[5] O. Silvén and K. Jyrkkä, “Observations on Power-Efficiency Trends in Mobile Communication Devices”, in Proceedings of the 5th Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), Lecture Notes in Computer Science, Vol. 3553, Samos, Greece, July 2005, pp. 142-151.

[6] E. Lee, “The Problem with Threads”, Computer Magazine, IEEE Press, May 2006.

[7] J. Gosling and H. McGilton, “The Java Language Environment: A White Paper”, Sun Microsystems Press, October, 1995.

[8] M. J. Schulte, J. Glossner, S. Mamidi, M. Moudgill, and S. Vassiliadis, “A Low-Power Multithreaded Processor for Baseband Communication Systems,” in Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation, Lecture Notes in Computer Science, Springer, vol. 3133, pp. 393-402, July 2004.

[9] B. Nichols, D. Buttlar, and J. Farrell, Pthreads Programming: A POSIX Standard for Better Multiprocessing, O’Reilly Nutshell Series, Sebastopol, CA, September, 1996.

[10] G. Blaauw and F. Brooks Jr., Computer Architecture: Concepts and Evolution, Addison-Wesley, Reading, MA, 1997.

[11] K. Jarvinen et al., “GSM Enhanced Full Rate Speech Codec," IEEE International Conference on Acoustics, Speech, and Signal Processing, 1997, pp. 771-774.

[12] V. Kotlyar and M. Moudgill, “Detecting Overflow Detection”, Proceedings of the 2004 CODES+ISSS International Conference on Hardware/Software Codesign and System Synthesis, September 8-10, 2004, Stockholm, Sweden, pp. 36-41.

[13] P. Balzola, M. Schulte, J. Ruan, J. Glossner, and E. Hokenek, “Design Alternatives for Parallel Saturating Multioperand Adders,” Proceedings of the International Conference on Computer Design, September, 2001, pp. 172-177.