Trends in Low Power Handset Software Defined Radio

John Glossner[1,3], Daniel Iancu[1], Mayan Moudgill[1], Michael Schulte[2], and Stamatis Vassiliadis[3]

[1] Sandbridge Technologies, 1 N. Lexington Ave., White Plains, NY 10601
[2] UW Madison, Dept. of EECS, Madison, Wisconsin
[3] Delft University of Technology, EEMCS, Delft, The Netherlands
jglossner@sandbridgetech.com

Abstract. This paper presents an overview of trends in low power handset SDR implementations. With the market for SDR-enabled handsets expected to grow to 200M units by 2014, the barriers to efficient handset implementations – both hardware and software – have been removed based on new and innovative architectures. We describe advances in DSP architectures and compilers that are enabling SDR handset implementations and present some results for a specific SDR design.

Keywords: Software Defined Radio, SDR, DSP, Multithreaded processors.

1 Introduction

Spectrum is scarce with precious little available for future technologies. Most of the world's available frequencies have already been allocated to specific services. What precious little remains is auctioned by government agencies typically for billions of dollars.

From a mobile operator's perspective delivering services to customers is dependent upon network capacity and coverage. Capacity is concerned with allowing more non-interfering users within a basestation cell or providing higher bandwidth to users within range of a basestation. Each new generation of communications systems designs attempts to provide additional capacity based on technological advances in the field. Recently MIMO-OFDM systems have been proposed [1] .

Coverage is concerned with providing voice and data services over large distances – with quality of service. In the future, data services may be the focus of cellular operators. Generally as a user moves farther away from a particular base station or service area the data speeds available to a user decreases. Additionally, cost is a concern. Providing high-speed cellular connections may be expensive. A common scenario proposed is providing wireless LAN hotspot coverage over a few hundred feet, WiMax coverage over a few miles, and cellular coverage (HSxPA, 1xEVDO) everywhere else.

Fig. 1 shows that the situation is actually more complicated than the above discussion. There are in fact multiple standards world-wide. Interestingly, the same is true of multimedia, location-based services, and user interfaces.

Software Defined Radio (SDR) has been proposed as a solution to providing better coverage. The SDR Forum [2] 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. This is an underlying technology necessary to realize cognitive radios.

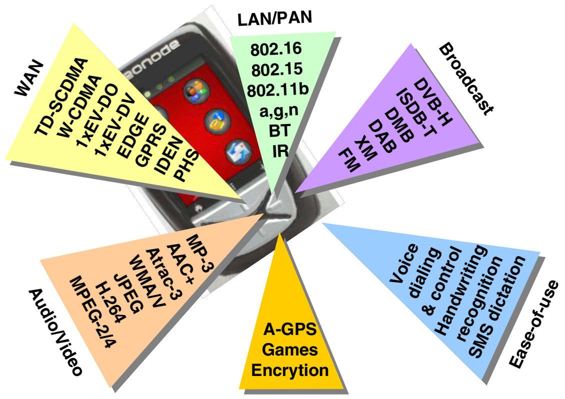

Fig. 1. Competing and Complementary Technologies

The advantages of reconfigurable SDR solutions versus hardware solutions are significant. First, reconfigurable solutions are more flexible allowing multiple communication protocols to dynamically execute on the same transistors thereby reducing hardware costs. Specific functions such as filters, modulation schemes, encoders/decoders etc., can be reconfigured adaptively at run time. Second, several communication protocols can be efficiently stored in memory and coexist or execute concurrently. This significantly reduces the cost of the system for both the end user and the service provider. Third, remotely reconfigurable protocols provide simple and inexpensive software version control and feature upgrades. This allows service providers to differentiate products after the product is deployed. Fourth, the development time of new and existing communications protocols is significantly reduced providing an accelerated time to market. Development cycles are not limited by long and laborious hardware design cycles. With SDR, new protocols are quickly added as soon as the software is available for deployment. Fifth, SDR provides an attractive method of dealing with new standards releases while assuring backward compatibility with existing standards.

SDR enabling technologies also have significant advantages from the consumer perspective. First, mobile terminal independence with the ability to "choose" desired feature sets is provided. Second, global connectivity with the ability to roam across operators using different communications protocols is enabled. Third, future scalability and upgradeability provide for longer handset lifetimes.

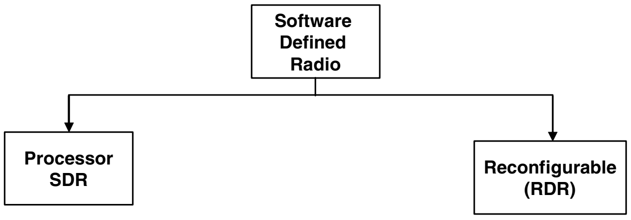

Fig. 2. SDR Variants

Fig. 2 shows two variants of Tier-2 SDR systems. Reconfigurable Digital Radio (RDR) platforms are generally FPGA-based with millisecond or longer reconfiguration times between communications systems. Of key importance is that RDRs generally follow a hardware design methodology. Processor SDR systems are instruction set computers. They follow a software design methodology and generally can be reconfigured in nanosecond timeframes. Historically, processor-based SDRs have not had enough performance to implement modern communications systems. Recently processor-based solutions with sufficient processing power have appeared [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] . The remainder of this paper focuses on processor-based SDR solutions.

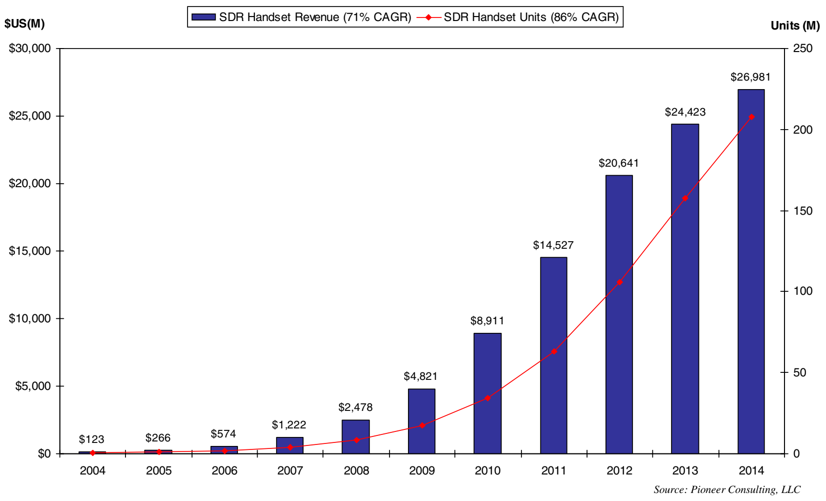

Fig. 3. SDR Handset Market (source: Pioneer Consulting, LLC)

Fig. 3 shows the growth in the SDR handset time both in terms of money value and unit shipments. We are now at the beginning of SDR-based handset deployments. In Section 2 we discuss processor related design issues both in terms of real-time parallel systems implementation and low-power. In Section 3 we discuss programmability and applications development for SDR solutions. In Section 4 we make concluding remarks on the future of SDR systems.

2 Processor Design

Since processor-based SDR systems are Digital Signal Processor (DSP) computer systems, it is helpful to recognize the historical context from which they have emerged. The types of processing include a mix of general purpose processing and signal processing. In this section we define architectural and industrial terms.

The *architecture* of a computer system is the minimal set of properties that determine what programs will run and what results they will produce [3] . It is the contract between the programmer and the hardware. Every computer is an interpreter of its *machine language* – that representation of programs that resides in memory and is interpreted (executed) directly by the (host) hardware.

The logical organization of a computer's dataflow and controls is called the *implementation or microarchitecture*. The physical structure embodying the implementation is called the *realization*. The architecture describes what happens while the implementation describes how it is made to happen. Programs of the same architecture should run unchanged on different implementations. An architectural function is *transparent* if its implementation does not produce any architecturally visible side effects. An example of a non-transparent function is the load delay slot made visible due to pipeline effects. Generally, it is desirable to have transparent implementations. Most DSP and VLIW implementations are not transparent and therefore the implementation affects the architecture [4] [5] [6] [7] .

Execution predictability in SDR 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.

A challenge of using VLIW processors includes large program executables (code bloat) that result from independently specifying every operation with a single instruction. As an example, a VLIW processor with a 32-bit basic instruction width requires 4 instructions, 128 bits, to specify 4 operations. A vector encoding may compute many more operations in as little as 21 bits (for example – multiply two 4-element vectors, saturate, accumulate, and saturate).

Another challenge of VLIW implementations is that they may require excessive write ports on register files. Because each instruction may specify a unique destination address and all the instructions are independent, a separate port must be provided for the target of each instruction. This can result in high power dissipation, which is unacceptable for handset applications.

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 cannot be restored. This requires programmers to break apart loops so that worst case timings and maximum system latencies may be acceptable.

Signal processing applications often require a mix of computational calculations and control processing. Control processing is often amenable to RISC-style architectures and is typically compiled directly from C code. Signal processing computations are characterized by multiply-accumulate intensive functions executed on fixed point vectors of moderate length. Therefore, a DSP requires support for such fixed point saturating computations. This has traditionally been implemented as one or more multiply accumulate (MAC) units. In addition, as the saturating arithmetic is non-associative, parallel execution of multiple data elements may result in different results from serial execution. This creates a challenge for high-level language implementations that specify integer modulo arithmetic. Therefore, most DSPs have been programmed using assembly language.

The problems associated with previous approaches require a new architecture to facilitate efficient convergence applications processing. An SDR architecture must allow for real-time execution, be highly parallel, and provide exceptionally low latency interrupt response times. Sandbridge Technologies has designed a processor with these characteristics. It is a multithreaded vector machine with multiple cores on a single die. The simplicity of VLIW implementations is embodied in the design with multithreading overcoming the limitations [20] .

3 Software Design

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 [21] .

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 [22] [23] . While there are still some objections to a multithreaded programming model [24] , to-date it is widely adopted particularly with the introduction of the Java programming language [23] [25] .

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 may be fixed, the software should be free to use any scheduling policy desired [26] . The POSIX pthreads open standard [27] provides cross platform capability as the library is compilable across a number of systems including Unix, Linux, and Windows.

There are many challenges faced when trying to develop efficient compilers for parallel DSP technologies. First and foremost, the Sandblaster processor is transparent in the architectural sense. This proscribes that there are no visible implementation effects for the programmer or compiler to deal with [3] . This is in distinct contrast with VLIW designs where the implementation strongly influences 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.

If a SIMD datapath to implement vector operations is utilized, the compiler must vectorize C code to exploit the data level parallelism inherent in signal processing applications and then generates the appropriate vector instructions. The compiler must also handle the difficult problem of outer loop vectorization.

Since saturating arithmetic is non-associative, out-of-order execution may produce different bit results. In some wireless systems this is not permissible [28] . By architecting parallel saturating arithmetic (i.e. vector multiply and accumulate with saturation), a compiler is able to generate code with the understanding that the hardware will properly produce bit-exact results. A compiler algorithm used to accomplish this is described in [29] . Some hardware techniques to implement this are described in [30] .

In multithreaded processors compilers should also automatically generate software threads. If the same pthreads mechanism for thread generation in the compiler is used as the programmer who specifies them manually, many economies of scale can be achieved. For most signal processing loops it is not a problem to generate threads and a compiler will automatically produce code for correct synchronization.

Fig. 4 shows the results of a number of communications systems as a percentage utilization of a 4-core 600MHz Sandbridge SB3011 platform. Particularly, WiFi 802.11b, GPS, AM/FM radio, Analog 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. For most of the systems, the values are measured on hardware from digitized RF signals that have been converted in real-time.

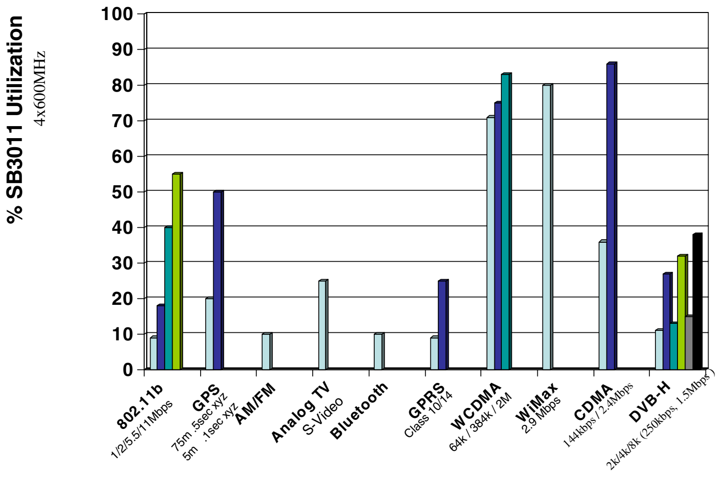

Fig. 4. Communication Systems Results as a Percentage of Utilization (4 cores at 600MHz)

4 Conclusions

The market for SDR-enabled handsets is expected to grow significantly over the next few years. This has been brought about by advances in low power DSP designs, improved software design methodologies, and a market demand for multimode multimedia devices. Such devices allow for improved carrier coverage over increasingly limited spectrum ranges. The result of SDR devices is improved coverage areas by means of dynamically reconfigurable radios.

References

[1] Stuber, G., Mclaughlin, S., Ingram, M., Pratt, T.: Broadband MIMO-OFDM Wireless Communications. Proceedings of the IEEE 92(2), 271–294 (2004)

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

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

[4] Case, B.: Philips Hopes to Displace DSPs with VLIW. Microprocessor Report, pp. 12–15 (December 1997)

[5] Wolf, O., Bier, J.: StarCore Launches First Architecture. Microprocessor Report 12(14), 1–4 (1998)

[6] Fridman, J., Greenfield, Z.: The TigerSHARC DSP Architecture. IEEE Micro 20, 66–76 (2000)

[7] Turley, J., Hakkarainen, H.: TI's New 'C6x DSP Screams at 1,600 MIPS. Microprocessor Report 11(2), 1–4 (1997)

[8] Glossner, J., Iancu, D., Lu, J., Hokenek, E., Moudgill, M.: A Software Defined Communications Baseband Design. IEEE Communications Magazine 41(1), 120–128 (2003)

[9] Lin, Y., Lee, H., Woh, M., Harel, Y., Mahlke, S., Mudge, T., Chakrabarti, C., Flautner, K.: SODA: A Low-power Architecture For Software Radio. In: Proceedings of the 33rd Intl. Symposium on Computer Architecture, pp. 89–100 (June 2006)

[10] Kneip, J., Weiss, M., Drwscher, W., Aue, V., Strobel, J., Oberthür, T., Bolle, M., Fettweis, G.: Single Chip Programmable Baseband ASSP for 5 GHz Wireless LAN Applications. IEICE Transactions on Electronics, pp. 359–367 (February 2002)

[11] van Berkel, C., Heinle, F., Meuwissen, P.P.E., Moerman, K., Weiss, M.: Vector Processing as an Enabler for Software-Defined Radio in Handheld Devices. EURASIP Journal on Applied Signal Processing 16, 2613–2625 (2005)

[12] Robelly, J.P., Cichon, G., Seidel, H., Fettweis, G.: A HW/SW Design Methodology for Embedded SIMD Vector Signal Processors. International Journal of Embedded Systems 1(11), 2–10 (2005)

[13] Duller, A., Panesar, G., Towner, D.: Parallel Processing — the picoChip Way! Communicating Processing Architectures 2003, pp. 125–138 (2003)

[14] Lodi, A., Cappelli, A., Bocchi, M., Mucci, C., Innocenti, M., De Bartolomeis, C., Ciccarelli, L., Giansante, R., Deledda, A., Campi, F., Toma, M., Guerrieri, R.: XiSystem: A XiRisc-Based SoC With Reconfigurable IO Module. IEEE Journal of Solid-State Circuits 41(1), 85–96 (2006)

[15] Mohebbi, B., Filho, E.C., Maestre, R., Davies, M., Kurdahi, F.J.: A Case Study of Mapping a Software-Defined Radio (SDR) Application on a Reconfigurable DSP Core. In: Proceedings of the International Conference on Codesign and System Synthesis, p. 103 (2003)

[16] Ungerer, T., Robič, B., Šilc, J.: A Survey of Processors with Explicit Multithreading. ACM Computing Surveys 35(1), 29–63 (2003)

[17] Smith, B.J.: The Architecture of HEP. In: Kowalik, J.S. (ed.) Parallel MIMD Computation: HEP Supercomputer and Its Applications, pp. 41–55. MIT Press, Cambridge, MA (1985)

[18] Mankovic, T.E., Popescu, V., Sullivan, H.: CHoPP priciples of operations. In: Proceedings of the 2nd International Supercomputer Conference, pp. 2–10 (1987)

[19] Tullsen, D.M., Eggers, S.J., Levy, H.M.: Simultaneous Multithreading: Maximizing on-chip Parallelism. In: Proceedings of the 22nd Annual International Symposium on Computer Architecture, pp. 392–403 (June 1995)

[20] Glossner, J., Iancu, D.: The Sandbridge SB3011 SDR Platform. In: Proceedings of the Symposium on Trends in Communications (SympoTIC'06), Bratislava, Slovakia (June 24-26, 2006)

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

[22] Goering, Richard: Platform-based design: A choice, not a panacea. EE Times (September 11, 2002), Available at http://www.eetimes.com/story/OEG20020911S0061

[23] Silvén, O., Jyrkkä, K.: Observations on Power-Efficiency Trends in Mobile Communication Devices. In: Hämäläinen, T.D., Pimentel, A.D., Takala, J., Vassiliadis, S. (eds.) SAMOS 2005. LNCS, vol. 3553, pp. 142–151. Springer, Heidelberg (2005)
[24] Lee, E.: The Problem with Threads. Computer Magazine, IEEE Press (May 2006)
[25] Gosling, J., McGilton, H.: The Java Language Environment: A White Paper. Sun Microsystems Press (October 1995)
[26] Schulte, M.J., Glossner, J., Mamidi, S., Moudgill, M., Vassiliadis, S.: A Low-Power Multithreaded Processor for Baseband Communication Systems. In: Pimentel, A.D., Vassiliadis, S. (eds.) SAMOS 2004. LNCS, vol. 3133, pp. 393–402. Springer, Heidelberg (2004)
[27] Nichols, B., Buttlar, D., Farrell, J.: Pthreads Programming: A POSIX Standard for Better Multiprocessing, O'Reilly Nutshell Series, Sebastopol, CA (September 1996)
[28] Jarvinen, K., et al.: GSM Enhanced Full Rate Speech Codec. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 771–774 (1997)
[29] Kotlyar, V., Moudgill, M.: Detecting Overflow Detection. In: Proceedings of the 2004 CODES+ISSS International Conference on Hardware/Software Codesign and System Synthesis, Stockholm, Sweden, pp. 36–41 (September 8-10, 2004)
[30] Balzola, P., Schulte, M., Ruan, J., Glossner, J., Hokenek, E.: Design Alternatives for Parallel Saturating Multioperand Adders. In: Proceedings of the International Conference on Computer Design, pp. 172–177 (September 2001)