THE VANU SOFTWARE RADIO SYSTEM

Dr. John M. Chapin (Vanu, Inc., Cambridge, MA, USA, jchapin@vanu.com)
Dr. Vanu G. Bose (Vanu, Inc., Cambridge, MA, USA, vanu@vanu.com)

ABSTRACT

Vanu Software Radio is a software architecture, implementation, and system design that has been used to implement a variety of waveforms, including voice and data standards for cellular and public safety systems. VSR applies modern software engineering to the high-speed signal processing code of a radio. The goal of the design is to combine software modularity, portability, and rapid development with excellent performance. This paper provides an update on the VSR design, which has evolved significantly in recent years from its origins in the MIT SpectrumWare system, and gives status and performance results of recent projects, including an all-software GSM basestation demonstrated at CTIA 2002 and a multimode multiband sleeve for a commercial PDA.

1. INTRODUCTION

Vanu, Inc. licenses SDR software and provides design consulting services to communications device manufacturers and users. The company focuses on the software problem of software radio: how to reduce the engineering cost of the highly sophisticated code needed for SDR systems. The company's approach includes strong software engineering, innovations in signal processing algorithms, a commitment to high level languages and portable code, and COTS-based system designs that combine these benefits with high performance and cost-effectiveness.

Vanu, Inc. was founded as a spinoff of the 1994-1998 SpectrumWare research project at MIT. SpectrumWare developed the technology needed to use COTS PCs as SDR platforms. The project's novel aim was to use the general-purpose CPU of the PC as the SDR signal processing engine. In fact, SpectrumWare ran the SDR application as a normal process on top of a standard desktop operating system. Vanu, Inc. has since branched out to employ a variety of embedded platforms, signal processing engines, and operating systems. However, the SpectrumWare heritage remains visible in the company's focus on implementing all SDR software, including the high-speed signal processing subsystem, as a portable application that exploits high-level languages and standard operating system APIs to the extent possible.

The company is active in military, public safety, intelligence, and commercial applications of SDR. For example, Vanu won a JTRS Step 2B contract for small handhelds and developed a telematics prototype for a tier-1 supplier to the automotive industry. Through its activities the company has developed implementations of a variety of waveforms. Highlights include GSM basestation, IS-136 TDMA mobile, IS-95 CDMA mobile, Project 25 mobile (a standard digital public safety waveform), and Mobitex (used by the RIM Blackberry).

The experience gained in developing these waveforms has guided ongoing improvement of the Vanu Software Radio approach, or VSR. VSR comprises the software architecture, high-speed implementation techniques, and system design that work together to significantly reduce the cost of SDR software. This paper provides a snapshot of where VSR is today and a roadmap for future development.

2. DESIGN PHILOSOPHY

Software costs are a significant and growing component of SDR engineering costs. In all but the highest-volume applications, the amortized cost of software is a major part of SDR device unit cost. While these facts are widely known, few SDR developers follow through to the logical conclusion: that software cost issues must be considered as a primary engineering tradeoff throughout SDR system design. In Vanu Software Radio, this imperative is reflected in a set of design choices that maximize software reuse and minimize development and maintenance effort.

2.1 Portability

VSR achieves much better software portability than other SDR approaches, especially when the high-speed signal processing software is considered. Most of the high-speed software is written in portable high-level code (C and C++). This is supported by selecting processing engines for which there are compilers that can produce highly efficient code. General-purpose CPUs have the best compilers, while some DSPs support acceptably good ones. Other DSPs require hand-tuning of assembly code to achieve good performance, and are thus less desirable. Similarly, current FPGA and reconfigurable hardware tool chains can only produce efficient output if given a toolchain-specific restricted form of source code. The VSR philosophy is to use the latter processing engines to the minimum extent needed to meet customer requirements, while keeping the bulk of the processing on an engine that can efficiently execute the portable version of the code. For the same reasons, VSR systems use

COTS operating systems and rely on standard OS APIs to the extent possible.

2.2 Component Reuse

VSR software uses highly standardized internal interfaces and a custom-developed middleware package to ensure that processing components can be reused across families of related waveforms.

Component reuse is not a perfect strategy. Some software components are widely used and well-defined, but too slow if implemented with a reusable component. Examples include Fourier transforms and FIR filters, for which the code needs to be specialized both to the platform and to the particular variant of the task being performed. In cases like this, VSR exploits tools that automatically generate the efficient implementation for a given task on a given platform. Research to develop these tools has been part of the work carried out at Vanu, Inc.

2.2 Generic hardware architecture

VSR systems follow a generic hardware architecture. Many existing SDR systems take a different approach, for example having on hardware component specialized for spreading, another for coding, and another for baseband processing. Even if the individual components are software-defined, use of a specialized architecture strongly limits what waveforms can be executed on the platform, even if the available processing cycles are sufficient.

2.3 Classification of existing systems

Taken together, the set of design approaches in VSR significantly improves the economics and potential rate of improvement of SDR systems. This is highlighted by the classification illustrated in Table 1.

Category	Examples
Modal SDR Software configures the radio ASIC or fixed hardware does processing	Dual-mode cell phone
Reconfigurable SDR All signal processing reconfigurable Significant use of FPGA or assembly	SpeakEasy AirNet
Software Radio (SWR) Exploits Moore's Law Supports software reuse across platforms	Vanu

Table 1: SDR classification

The simplest example of a software radio is a dual mode cell phone. This is an example of a *Modal SDR*. A dual mode cell phone has two hardware radios in it, one for each standard that it supports. Software determines which standard needs to be run, and activates the correct radio. While this type of SDR provides the flexibility to switch between modes that were build in to the radio, it does not provide the user with the ability to upgrade the system with new waveforms.

Reconfigurable SDR is the type of software radio that has been built for defense applications over the last decade. These systems typically involve a combination of processing technologies such as ASICs, FPGAs, and DSPs. These specialized systems offer excellent performance as designed but, if the software investment is high, rapidly become obsolete as technology development continues. For example, the SpeakEasy system was built around a combination of FPGAs and 40 MHz TI C40 DSPs. By the time the first prototype was demonstrated, COTS DSPs were available at 166 MHz. As the SpeakEasy software was tied not only to the C40 but to a specific layout of C40s and FPGAs, the new DSPs could not be exploited.

Software Radio (SWR) is a type of SDR that maximizes software reuse across platforms and hardware generations. The key technology that enables this is to write the signal processing software as an application-level program running on top of a standard operating system (whether on GP CPU, DSP, or other processing engine). In addition to reducing software development costs, use of application-level software and an OS allows the underlying hardware components to be upgraded without incurring the high cost of redeveloping the software. As a result, SWR systems can track the Moore's Law performance curve over time at a much lower cost than other types of SDRs.

3. ARCHITECTURE

The Vanu Software Radio architecture separates the hardware and software portions of the system.

3.1 Hardware Architecture

The hardware architecture, pictured in Figure 1, groups the hardware components into three blocks representing the antenna, RF-to-digital and processing subsystems. No hardware component in the architecture is specialized to any particular waveform. While the architecture places no limitation on the achievable waveforms, any given implementation of the architecture can only support some waveforms. Each implementation will support a limited range of RF frequencies, bandwidths, and amount of computational power. For example, in order for a platform to be software upgradeable from 2G to 3G cellular standards, the implementation must be able to receive a 5 MHz wide band in the appropriate frequency ranges and have enough computational power to perform the 3G processing.

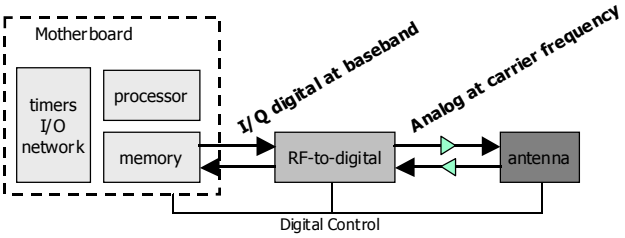

Figure 1: Hardware Architecture

The rightmost block in Figure 1 represents the antenna subsystem. The interfaces to the antenna block are RF transmit and receive analog lines and a digital control interface. With these interfaces, the architecture can accommodate traditional passive antennas (for which the digital interface has no function) as well as advanced systems such as electrically controllable antenna arrays. Note that the architecture does not specify a particular type of digital connection (e.g. RS-232) as this is a detail of the implementation.

The next block to the left in Figure 1, labeled *RF-to-Digital*, is the only layer of the system that contains radio-specific analog components. On the receive side, its sole function is to generate a digitized representation of a downconverted slice of the radio spectrum. On the transmit side, it generates an upconverted radio signal from a digitized representation. This block does not perform waveform specific processing such as demodulation or equalization.

The name of the third block, *Motherboard*, is borrowed from the PC world because software radios built to the Vanu architecture look much more like computers than like legacy radios. Like a PC motherboard, this layer contains memory and processor components, and provides I/O to a network, to the user, timing support, and similar functions.

3.2 Software Architecture

Waveform implementations have two parts that face significantly different portability challenges. The *control* part configures and controls the system, and implements higher level functions such as protocol state machines and network routing. The *signal processing* part implements the transforms between user data and a sampled representation of a RF waveform. The software architecture, shown in Figure 2, graphically illustrates how these two parts are built upon a common operating system.

The Operating System layer sits on top of the motherboard. Use of an operating system is critical to the design approach because it isolates the signal processing application from the hardware and thereby significantly improves its portability. Vanu systems place no unusual demands on the operating system. Any POSIX operating system that supports high data rate transfers between

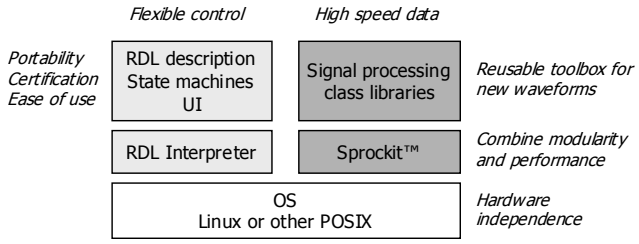

Figure 2: Software Architecture

applications and the underlying hardware can be used. The current implementation runs on Linux and on the real-time operating system QNX.

The signal processing component consists of the Sprockit™ middleware layer which performs data movement and module integration, and a library of signal processing modules for functions such as FM modulation, Viterbi encoding, or FIR filtering.

The control component consists of a runtime system that interprets state machines and processing descriptions, and a downloadable application that determines which waveform or communications standard the system will implement. The portion of the downloadable application that describes the signal processing pipeline is written in a Vanu-developed language called RDL, the Radio Description Language.

In the Vanu software signal processing system, there is a library of processing modules corresponding to the objects that may be in the RDL description. The runtime system instantiates an implementation object from the library for each processing stage in the RDL description, sets its parameters as specified by the application, and makes the appropriate high-speed data connections between the objects and lower-speed control connections to the application part. Some of the library objects are fully generic, such as filters and decoders, while others provide specialized services for different waveforms.

4. IMPLEMENTATION EXAMPLES

There are two primary classes of radio systems, infrastructure devices and client devices. Infrastructure systems, such as basestations, typically handle multiple channels simultaneously and have high availability requirements. Client devices, such as cell phones and vehicle-based telematics systems, typically operate on a single channel at a time but have more stringent power dissipation and form factor requirements. While the system architectures for both classes of devices are the same, the different system requirements lead to different implementations of the architecture. This section presents two example implementations of the VSR architecture, a handheld device based on the HP iPAQ PDA and a GSM basestation.

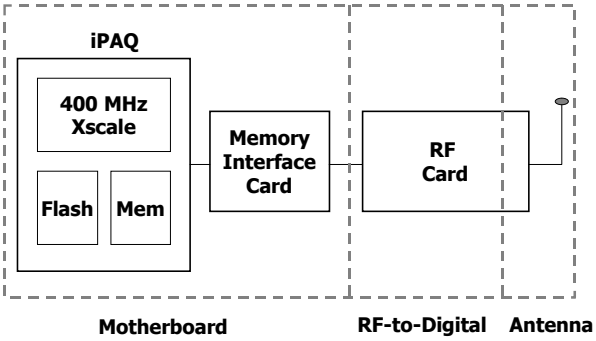

Figure 3: Implementation of VSR in a handheld

4.1 iPaq Handheld

The block diagram of the iPAQ-based handheld software radio is shown in Figure 3. The dotted gray blocks indicate how the three different components of the hardware architecture from Figure 1 are implemented.

The motherboard subsystem is comprised of the iPAQ and a memory interface card.. The iPAQ contains a 400 MHz Xscale processor, RAM and FLASH. The memory interface card, shown in Figure 4, is essentially a DMA engine added to the iPAQ, but it also contains other digital interfaces such as a GSM SIMM card module.

Figure 4: Interface card with iPaq sleeve for scale.

The RF-to-digital block is implemented as a card that covers the frequency range from 30 MHz to 2.5 GHz, and has selectable bandwidths of 30 kHz, 200 kHz, and 1.25 MHz. It is designed to operate over the temperature region of –40 °C to +85 °C and has a maximum transmit power of 0.6 watts, while providing transmit power control capable of meeting IS-95 requirements. This card also contains the A/D and D/A converters as well as a digital control interface for setting parameters and measuring variables such as received signal strength.

The antenna block includes both the antenna as well as a portion of the RF card. The RF card contains several antenna ports. The active port can be selected through software to enable use of different antennas for different bands, or multiple can be activated for applications that exploit diversity. The digital control interface to the antenna block in Figure 1 is actually implemented as part of the RF card, which is why the dotted grey line that outlines the antenna block contains part of the RF card.

At the time of this writing, the iPAQ system is being tested, so a complete list of benchmarks is not yet available. Preliminary data indicates that the IS-136 forward voice channel receiver consumes 22% of the 200 MHz StrongARM processor. The receiver software measured includes equalization, synchronization, separation of timeslots, demodulation, decoding, vocoding and message processing.

4.2 GSM Basestation

A GSM basestation requires scalability to hundreds or possibly thousands of simultaneous channels, as well as the ability to provide "five 9's" availability. These requirements lead to a very different implementation of the Vanu Software Radio Architecture, as illustrated in Figure 5.

In the basestation implementation, the *motherboard* subsystem is implemented as a distributed computation platform using 1U rackmount servers interconnected by a COTS interconnect (currently gigabit ethernet). These components were chosen for this implementation because they are riding computer industry cost curves, are widely available from a number of vendors and, most importantly, the technology tracks Moore's Law, enabling more channels to be processed with every new generation of processor that comes to market. This implementation also allows the system to take advantage of fault detection and failover techniques developed for distributed systems in order to meet the availability requirements of the basestation system.

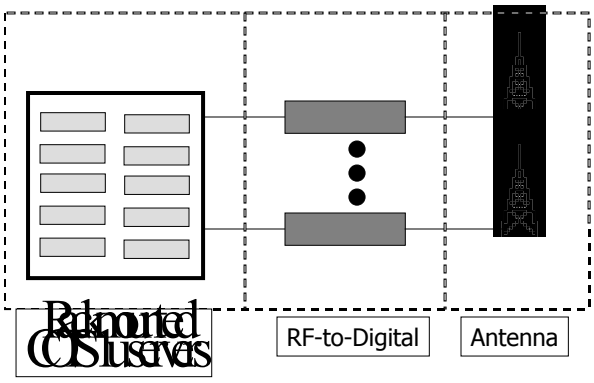

Figure 5: Basestation Implementation

The RF-to-digital section can utilize a variety of technologies, from traditional cell or PCS band up/down converters to more flexible software radio front-ends. Multiple front-ends can be connected to the system to provide support for multiple antennas and bands. The digital data is streamed over the switched interconnect between the front-ends and the processing units.

Multiple different types of antennas can be supported. We have used both traditional, passive cellular and PCS antennas, as well as antenna remoting systems that transfer the RF from an antenna to the processing location over optical fiber. This type of system allows all of the antennas in a given region to be supported by an efficient centralized processing facility.

This implementation of the architecture does not limit the system to GSM. As long as the RF front-ends are capable of handling 5 MHz wide channels, the system can be upgraded to GPRS, EDGE, IS-95, 1xRTT, or WCDMA by simply downloading new software. The processing units can distribute the processing load for the more compute intense standards such as WCDMA. If necessary, the processing capability of the system can be upgraded by adding more servers to the rack. The software infrastructure is designed to handle multiple, heterogeneous processors running in the system simultaneously. This enables the use of the latest technology whenever processing capability is added.

The processing performance of the GSM basestation software on a 1.6 GHz Athlon processor is shown in Table 2. It is interesting that the control channel and voice channel cost the same to process. This is because the majority of the processing costs are associated with functions common to both types of channels: equalization, Viterbi decoding and de-multiplexing. These numbers indicate that using dual processor, 2 GHz servers that are readily available today, each 1U box in the rack can provide the processing capacity for 32 GSM channels (i.e. four single frequency sectors).

Benchmark	% CPU required
Control channel (downlink BCCH and uplink RACH)	6%
Voice channel (both downlink and uplink).	6%

Table 2: GSM basestation costs (1.6 GHz Athlon)

5. CONCLUSION

There have been tremendous advances in the underlying components of software radio systems in the last several years. Five years ago, A/D converters with sufficient spurious free dynamic range and speed only existed as defense research projects, whereas today converters that meet the requirements of many basestation applications are commercially available.

Similarly, processors have continued to get faster, enabling more channels to be processed on a single processor as well as more complicated standards to be processed on a single processor. There have also been significant advances in low power processors, enabling the construction of some mobile software radio devices.

There have also been considerable advances in RF technology, resulting in highly integrated CMOS RF chips that cover wide frequency ranges. However, this integration is focused at traditional limited-band hardware radio designs. Software radio front-ends that cover a wide frequency ranges are typically built using multiple chips and discrete components. As the software radio market emerges, we expect to see fully integrated software radio front-ends come available.

Software radio technology is mature today for infrastructure and vehicular markets, as well as for handheld markets with moderate battery life requirements. The Vanu Software Radio architecture presented in this paper is designed to minimize the amount of software that has to be re-written in order to keep pace with advances in the underlying technology. Thus, as faster processors come to market, VSR basestation systems will get smaller and cheaper, and as advances in low power processors are made, VSR handheld products will extend into markets that require longer battery life. These benefits will be coupled with the development cost benefits of software reuse to provide substantial advantages compared to traditional SDR architectures.

6. REFERENCES

[1] "AdaptaCell™ Broadband, Software-Defined Base Station." AirNet Communications Corporation. http://www.aircom.com/pr_adaptacell.htm

[2] Bonser, Wayne. "US defence initiatives in software radio." Software Defined Radio: Origins, Drivers and International Perspectives. Walter Tuttlebee (Ed.). John Wiley & Sons, Chichester. 2002. Chapter 2.

[3] Bose, Vanu. "Design and implementation of software radios using general purpose processors." MIT Ph.D. thesis. June 1999.

[4] Chapin, John. "Handheld all-software radios: prototyping evaluation for JTRS Step 2b." http://www.jtrs.saalt.army.mil/docs/documents/step2b/Vanu.pdf

[5] Chapin, John, Chiu, Andrew, and Hu, Roger. "PC clusters for signal processing: an early prototype." IEEE Sensor Array and Multichannel Signal Processing Workshop. Cambridge, MA. March 2000. http://www.vanu.com/publications/sam2000pcclusters.pdf

[6] "Programmable, scalable wireless information infrastructure." NSF Contract DMI-9960454 final report. July 11, 2000.