PICOCEPTOR™: ADVANCED ARCHITECTURE FOR MINIATURE SOFTWARE DEFINABLE RADIO SYSTEMS

Clark Pope (DRS Signal Solutions, Gaithersburg, Md; clark.pope@DRS-SS.com)
Michael Kessler (DRS Signal Solutions, Gaithersburg, Md; michael.kessler@DRS-SS.com)

ABSTRACT

This paper introduces a novel software definable radio (SDR) architecture, called Picoceptor™[1], which combines tight RF integration, a reconfigurable field-programmable gate array (FPGA) system on a chip (SOC), embedded Linux, and a USB 2.0 on-the-go (OTG) interface in a low-cost, low-power platform that fits into a shirt pocket. This advanced SDR system technology provides extensive savings in size, weight, and power (SWaP) by both increasing the functionality and flexibility of the processing hardware as well as eliminating the need for several typical system components. This paper also provides a brief history of SDR approaches and technology as well as a detailed description of the design objectives and choices when developing the Picoceptor™ architecture. The development tool set and environment are described. Finally, applications such as geo-location and spectral search are discussed with detailed implementation notes to highlight the adaptability and flexibility of the architecture for a wide range of SDR functions.

HISTORY OF SDR TECHNOLOGY

In the late 1980's, DRS Signal Solutions, Inc. (DRS-SS) was one of the first companies to incorporate digital signal processor (DSPs) in its radios. As soon as radios contained reprogrammable DSP devices, instead of fixed analog circuits, the desire for a software definable radio arose.

The evolution of various platforms is depicted in Figure 1. Ten years ago a typical SDR consisted of VXI or VME modules from various vendors. At least five major components were required for a system: a Slot 0 controller, a VXI rack, a tuner, a digitizer, and a DSP.

Five years ago, it became possible to integrate the tuner, digitizer, and DSP into a single module. This resulted in the two-slot solution shown in the middle of Figure 1. This evolution cut the cost, weight, power, and size of systems by half or more. This generation of DRS-SS products was dubbed Sunrise.

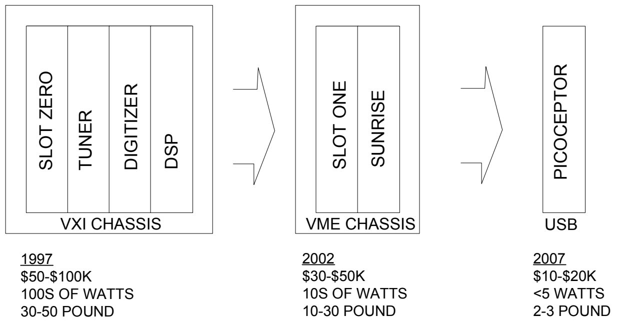


Figure 1. Evolution of SDR Platforms

With the advent of SOC FPGAs, feature-rich and embeddable operating systems, standardized USB bus peripherals, and advances in miniature RF components, it is today possible to implement the same functionality in the palm of a hand at a fraction of the SWaP. This generation of technology is called Picoceptor™.

The software for SDR systems has undergone a similar evolution. Ten years ago SDR processing was mostly a cobbled together code base consisting of pieces of vendor-provided libraries along with customized local intellectual property (IP). Today there are entire suites of software tools to aid in the development and deployment of SDR applications. In the open source world there is GNU radio. This approach is free and feature-rich, but does not provide professional-level support. For professional applications The MathWorks MATLAB™, Xilinx, Pentek, and iVeia (among others) can provide full tool suites. For applications requiring adherence to the Joint Tactical Radio Service/Software-Compliant Architecture (JTRS/SCA)[2] , there are several operating systems (OSs), middleware, and operating environment toolsets from Prismtech, Zeligsoft, et al.

1. SDR REQUIREMENTS

A generic SDR system is shown in Figure 2. The three major components are the tuner, which converts radio frequency (RF) energy to some lower intermediate frequency (IF) suitable for digitization; a demodulator; and some control asset such as a personal computer.

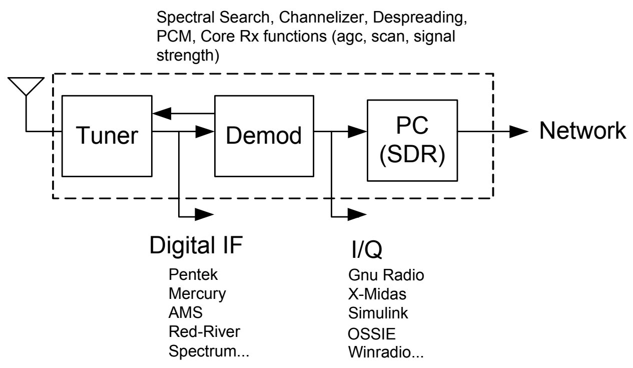


Figure 2. Typical SDR Platform

As shown, there are three key data spigots that must be accessible in some form: raw analog-to-digital converter (ADC) output data, baseband in-phase and quadrature-phase (I/Q) data, and demodulated data or bits. The raw ADC data allows users to build IF/RF panoramic displays to help search for signals. The baseband I/Q data provides the filtered data to users, who can then feed it into software baseband analysis and demodulation programs. Finally, the demodulated data is important for users wanting direct access to the information on the signal of interest.

As long as the three data taps described above are available, the SDR system is able to accommodate virtually any application. There is, of course, a large trade space where the designer must decide which portions of the processing to put in software, FPGAs, or DSPs. Each designer will generally choose a different mix based on the desired application and their particular skills and experience.

In addition to providing a flexible data path, the usual metrics that apply to radios also apply to SDR platforms. Specifically, SWaP and cost can never be too small and tuning range, dynamic range, and maximum bandwidth can never be too large.

Finally, the architecture should provide for expanded capability in multiple dimensions (more memory, software upgrades, larger FPGAs, etc.) and not require users to pay for, or consume power for, features that they do not use.

2. PICOCEPTOR™ ARCHITECTURE

This section provides some background on the technologies available and the ultimate design choices made for the Picoceptor™ RF front end, signal processing hardware, and software.

2.1. RF Front End

In general, three technologies currently exist to allow converting RF energy to baseband where it can be processed: direct digitization, direct conversion using RF application-specific integrated circuits (ASICs), and the traditional discrete superheterodyne receiver.

The superheterodyne approach is shown in Figure 3. Here the RF energy (after preselection and amplification) is mixed with the 1st local oscillator (LO) to an IF. The IF is then filtered and amplified before mixing with the 2nd LO whose output is then digitized. Since the radio selectivity is determined by the IF filter (at a fixed frequency), it is much easier to optimize the design for maximum dynamic range. As a result, superheterodyne systems provide the best overall radio performance at the expense of some additional complexity.

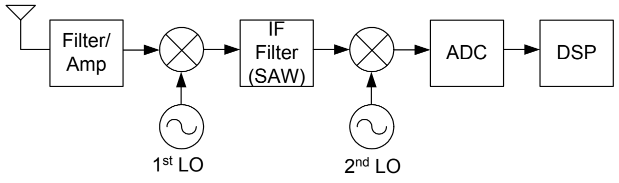


Figure 3. Superheterodyne Approach

With direct digitization (Figure 4) a large portion of the RF spectrum is fed into an ADC. By the Nyquist theorem, the energy up to one half the ADC clock rate can be accessed. Depending upon the input bandwidth of the ADC, particular bands can be subsampled to extend the maximum frequency further. These systems usually consume a lot of power and have marginal dynamic range, though they are very inexpensive. The technology does not yet exist to provide a good multi-gigahertz general-purpose receiver using direct digitization although companies like Hypres[12] are improving.

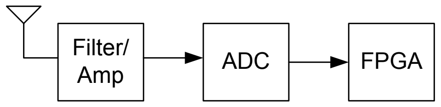


Figure 4. Direct Digitization

In recent years the various RF components have been increasingly integrated into ASICs with direct conversion schemes where the RF energy of interest is directly mixed to baseband and its I/Q components are digitized. See Figure 5. These systems are very cheap, but their performance is limited by the ability to compensate for phase and gain differences in the I/Q paths. Even with elaborate compensation techniques, the dynamic range of direct-conversion receivers is several tens of decibels worse than the superheterodyne systems.

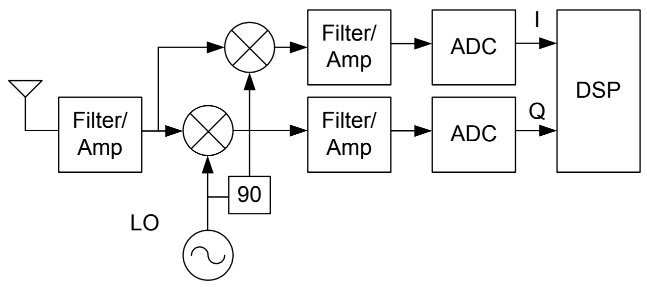


Figure 5. Direct Conversion.

Given the state of the art of these three approaches, the superheterodyne approach used in the Picoceptor™ front end is still the most suitable architecture for general-purpose, wide-coverage signal intelligence (SIGINT) receivers. The Picoceptor designers used a traditional conversion scheme but carefully selected VCOs, mixers, and amplifiers to reduce the circuit complexity while still covering the full 3 GHz tuning range.

2.2. Processing Hardware

Given the current state of technology, the design choices for body-wearable radios are fairly straightforward. First, in order to accommodate multiple applications and still provide sufficient computation resources, it must contain a modern FPGA. Second, in order to provide the features users expect, it must be able to host a sizeable operating system. This determines the random access memory (RAM) and FLASH memory requirements. Finally, in order to provide system expansion, only serial bus standards are appropriate. USB2.0 OTG allows the unit to operate as either a device or a host, requires low power, and provides sufficient bandwidth for most applications.

There are several controversial issues. The first is whether to use a separate processor or a processor embedded in the FPGA, itself. Good arguments can be made on either side but the Picoceptor™ team opted for the embedded processor so that we would have full control over the connectivity of the processor. This allows one to add auxilliary processing units (APUs), busses, peripherals, direct memory access (DMA) engines, and switch clocks dynamically as required. It also protects the design from obsolescence and allows optimal sharing of the memory resources between the central processing unit (CPU) and the signal processing in the FPGA fabric.

A second controversy is whether to include an Ethernet interface in the design. Ethernet is actually very expensive both in power consumption (500-750 mW) and in the size of the large RJ-45 connector. In the context of a body-wearable radio, Ethernet ultimately makes no sense because there is no place to plug in the other end of the Ethernet cable. With USB 2.0, however, users who want an Ethernet interface can simply add a USB-to-Ethernet adaptor dongle and/or emulate Ethernet over a simple USB cable using the Linux USBnet driver.[3]

The final back end design is shown in the block diagram of the Picoprocessor motherboard as shown in Figure 7. The core design requires only five chips: RAM, FLASH, FPGA, the USB physical layer (PHY), and a system management processor (PIC). The design is modular such that any one of these components can be upgraded with little effect to the overall system and software.

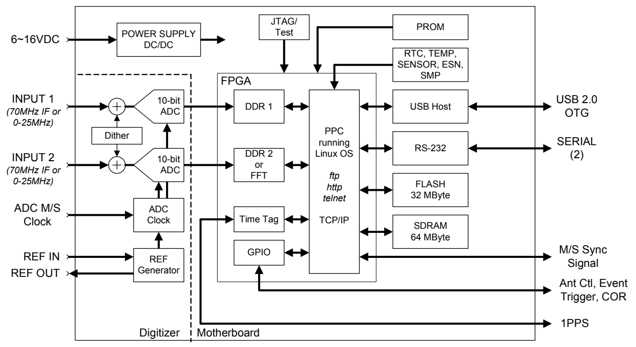
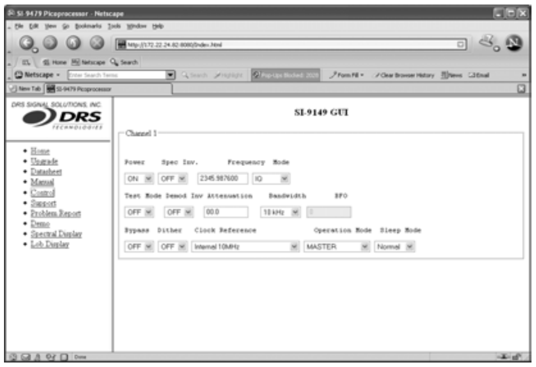


Figure 7. Picoprocessor Block Diagram

2.3. Software

There are several vendors providing lightweight, real-time, embedded operating systems. For maximum features and capabilities the Picoceptor™ uses embedded Linux (EL) with a 2.6 kernel. The EL OS design choice recognizes that real-time extensions can be added if needed and that most any other OS can run in the same space as EL. The EL OS gives us the option to be flexible in our software. It also provides many methods of communication to the Picoceptor™, such as Ethernet over USB using Transmission Control Protocol/ Internet Protocol (TCP/IP) or User Datagram Protocol (UDP), RS-232, and USB. The user may control the Picoceptor™ using three-letter mnemonics or through the Picoceptor™-based Web Server using a Web browser as shown in Figure 8.

The Control Application is a Linux process that receives messages from any of the various external interfaces and performs command and control of the Picoceptor™. It is a socket-based program that accepts input from the user, parses the command, and calls a custom device driver that controls the FPGA.



SI-9149 GUI

Channel 1

Power	Spec Inv.	Frequency	Mode
ON	OFF	2345.987600	IQ

Test Mode	Demod Inv	Attenuation	Bandwidth	BFO
OFF	OFF	00.0	10 kHz	

Bypass	Dither	Clock Reference	Operation Mode	Sleep Mode
OFF	OFF	Internal 10MHz	MASTER	Normal

Figure 8. Picoceptor Web-based GUI

A low-level custom device driver is provided that allows the user to control and access the Picoceptor™ FPGA hardware. The result of this effort is a kernel module that contains functions that can be called by the control application using an application programming interface (API). The device driver allows the user to change parameters such as frequency and attenuation that are mapped to FPGA registers. The device driver also allows the user to get snapshot data from the FPGA.

Linux also provides the user with the capability to write their own applications and load them onto the Picoceptor™. Users can place their own applications in the RAMDISK on the Picoceptor using File Transfer Protocol (FTP) and execute them via a telnet session or run them as a Linux service.

2.4. Development Tools

Based on experience with previous SDR developments, it is clear that user requirements range from turnkey solutions with little or no technical knowledge requirements to "blank slate" systems for users wanting to load their own OS, application software, and/or signal processing circuits. As a result the Picoceptor™ is delivered as a fully functioning radio; nevertheless, with the purchase of the Pico Developer's Package (PDP), users can get full access to the development tools used at the factory.

The PDP contains manuals, programming cables, several USB devices, test cables, RF cables, and a DVD. The DVD contains a VMWare[4] image of our development environment.

The VMware image runs in the VMware player as shown in Figure 9. The virtual machine (VM) is a complete Ubuntu 6.06 Linux[5] installation containing the cross-compilers, source code, and FPGA tools for the Picoceptor™.

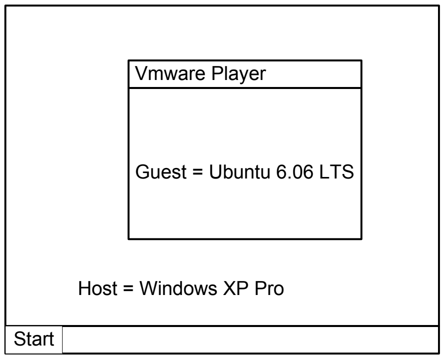


Figure 9. PDP VMware Image

The VM (a) eliminates installation problems (b) ensures all users have the exact same tools and versions, and (c) allows Windows users to easily access the Linux-based tools. Some disadvantages of the VM are that it has heavy disk and memory requirements, runs more slowly than native installations, and has limited hardware support. None of these limitations has presented any significant problem for Picoceptor™ developers.

With the VM, users are able to reconfigure/rebuild the Linux kernel, debug applications, modify the root file system, build applications and drivers, download new FPGAs, reflash the unit OS, and reprogram the Picoceptor™ FPGA.

It is important to note that Picoceptor™ users are not limited to the provided operating system or FPGA tools. Alternate solutions (from iVeia, The MathWorks, Xilinx, Prismtech, Ossie[6], Green Hills, Gnuradio[7], and Pentek, among others) can be adapted to run inside the Picoceptor™. The choice is largely based on the developer's experience and existing IP.

2.5. Analysis

There are several key benefits of the architecture. First, it is OS agnostic, meaning virtually any operating system can be hosted by the hardware. Designing for embedded Linux was key to ensuring this because EL generally has a bigger footprint than the more optimized real-time embedded OSs. The tight integration of the processing hardware with the tuner enables optimized spectral search as well as monitoring frequency-hopped signals. There is no power or cost to users for unused features. The design is "future-proof" in that the processor and VHDL will carry forward through the next generations of FPGAs.

Basically, the architecture remains the same even as the underlying technologies advance (e.g., larger FPGA, more memory, USB 3.0, etc.)

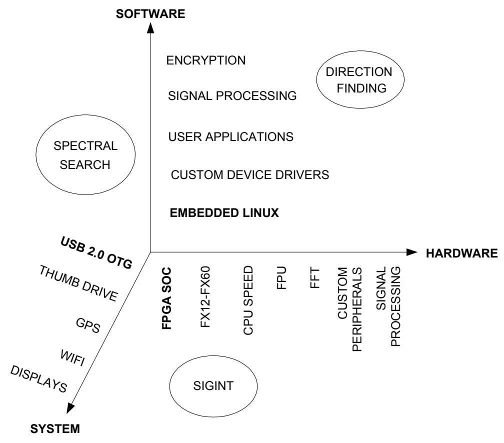


Figure 10. System Expansion.

The capabilities expansion is shown in Figure 10. Because of the FPGA SOC-based design, floating point units, DMAs, and fast-fourier transform (FFT) engines can be added to the system as desired, virtually any software application (including database, analysis, and network applications) can be cross-compiled and run on the platform. Finally, any existing device with a USB interface can be attached to the system so long as it has a suitable Linux driver.

Figure 11 demonstrates how multiple Picoceptor™ units can be cascaded to form an N-channel system. The design supports cascaded references so a single 10 MHz or 1PPS clock from a Global Positioning System (GPS) source can be used to phase lock multiple units. Additionally, because of the network-centric design, complex algorithms can then be distributed across the N units to provide greater computational capacity.

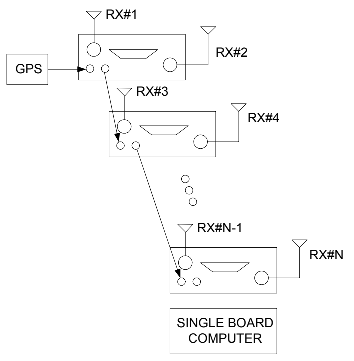


Figure 11. Cascaded Units

3. APPLICATIONS

Two prominent applications for miniature SIGINT platforms are direction finding/geolocation and spectral search. It is important that any radio platform take into account the special requirements of these applications in the design approach.

3.1. Spectral Search

Spectral search involves scanning multiple RF bands looking for and logging the presence of new energy to a file to create a baseline of radio activity so that an operator may discover new signals as they appear.

The key requirements for a good spectral search engine are a fast FFT-processing engine, large memory for storing the FFT bins, and tight integration with the tuner for maximum processing pipelining. All three requirements are met with the Picoceptor™: the FFT can be calculated in the FPGA fabric, the large double data rate (DDR) memory provides ample storage for spectral data, and the FPGA has direct access to the tuner.

A state machine depicting the spectral search process is shown in Figure 12. First, the front end is tuned to the desired frequency, next data is collected, and then lastly the FFT processing is performed. The resulting data is averaged against the stored spectral data and new energy alarms are calculated and recorded. The process repeats as the front end is stepped across the band(s) of interest.

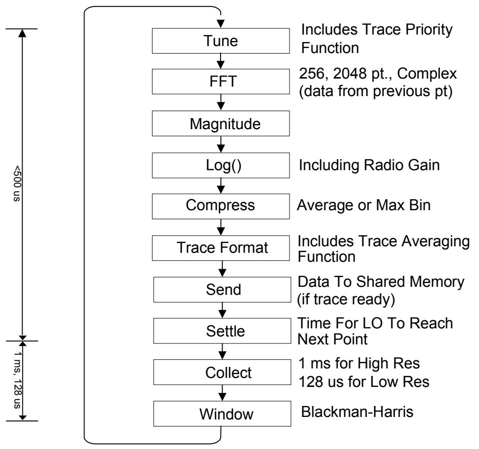

<500 us
1 ms, 128 us
Tune
Includes Trace Priority Function
FFT
256, 2048 pt., Complex (data from previous pt)
Magnitude
Log()
Including Radio Gain
Compress
Average or Max Bin
Trace Format
Includes Trace Averaging Function
Send
Data To Shared Memory (if trace ready)
Settle
Time For LO To Reach Next Point
Collect
1 ms for High Res
128 us for Low Res
Window
Blackman-Harris

Figure 12. Spectral Search State Machine

It is important to realize that the ultimate scan rate is limited by three factors: tune time, collection time, and processing time. The tune time is determined by the front end hardware and is usually set by the narrowest phase locked loop (PLL) in the tuner. The Picoceptor™ has sub-millisecond tuning. Collection time is determined by physics. To obtain a particular spectral resolution in hertz, one has to collect data that approximates the inverse of that resolution in time. For example, 1 kHz resolution requires 1 millisecond of data. Finally, processing time is determined by the available computation resources, usually a DSP processor or FPGA.

The FFT engine in the Picoceptor™ is shown in Figure 13. First a Blackman-Harris window is applied to the data. Then an FFT is performed. A coordinate rotation digital computer (CORDIC) module is used to convert the complex bin data to magnitude. Finally, the log base 2 is taken (because it is easy to calculate and scales linearly with log base 10), and the result is output to a first-in first-out (FIFO) buffer for collection and storage with the spectral data by the main processor. This particular implementation uses a 1024-point FFT, 16-bit input/output resolution, and requires about 130 microseconds to run. Since the processing can overlap the tune and collection times, it is expected that scan rates of several GHz/sec are possible with the Picoceptor™.

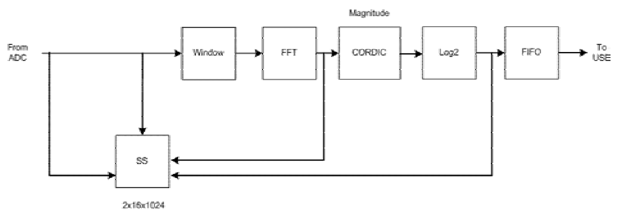


Figure 13. FFT Processing

3.2. Direction Finding (DF)

There are many different methods for determining the angle of arrival (AOA) or line of bearing (LOB) of a signal.[8] The single-channel techniques generally require multiplexing the signals from multiple antenna elements to feed one receiver which has the processing to demultiplex the signals to perform DF. If two receivers are available a straightforward vector correlation can be performed (with a 180 degree ambiguity). Both the single-channel and dual-channel methods can be extended to N elements using a commutator to quickly switch around the antenna. Commutation requires that the signal of interest is stable for long periods. Finally, time difference of arrival (TDOA) methods can also be used.

An example of a single-channel DF algorithm is the Watson Watt DF (WWDF)[9]. This method highlights two required features of modern SDR platforms. First, since the WWDF algorithm requires very distinct processing relative to typical radio demodulation, it is important that the platform is easily reconfigured so that the specialized processing can be loaded on demand. Second, extensive general-purpose I/O (GPIO) is required in order to interface the radio to the various antennas. The antennas may require bias voltages, TTL control lines, and serial ports; therefore the radio must have flexible control for maximum compatibility. The Picoceptor™ provides several GPIO lines for this purpose.

The WWDF system is shown in Figure 14 and the gain pattern for the CODEM AVM-1 Antenna is shown in Figure 15. As seen in the gain pattern, there are two element pairs East-West (E-W) and North-South (N-S) oriented perpendicularly. The Picoceptor™ provides a 150 Hz modulating tone that multiplexes the N-S and E-W elements onto a single signal. The processing algorithm in Figure 16 mixes the signal with the original tones, performs DC averaging, and an arctangent is used to produce the LOB.

This particular system was implemented in a Picoceptor™ and was found to have comparable response time and accuracy to the larger DF system it replaced.

3.3. TDOA/Geolocation[10]

Geolocation is distinguished from DF in that the precise emitter location is determined rather than just an angle. TDOA is used to determine the distance of the emitter to multiple sensors (usually at least four).

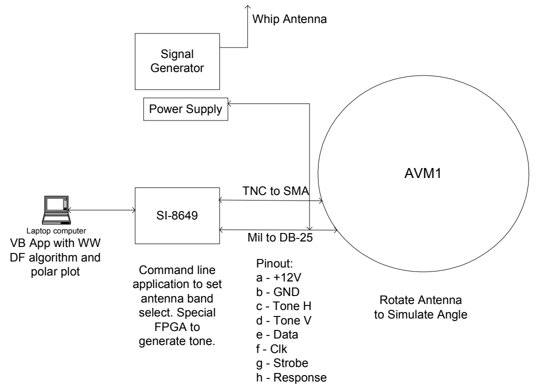


Figure 14. WWDF System

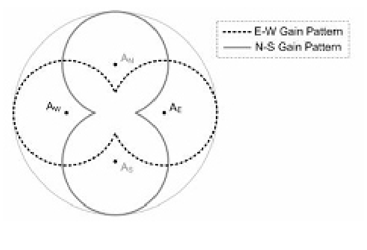


Figure 15: WWDF Gain Pattern[8]

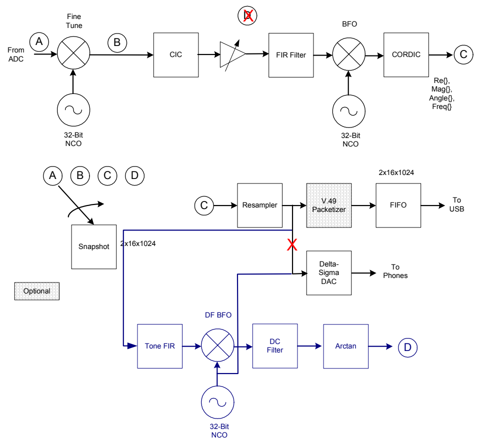


Figure 16. WWDF Processing Algorithm

Since the locations of the sensors are known, the position of the emitter can then be determined.

TDOA requires several features in the radio. First, all sensors must operate on the exact same time base. This is accomplished with GPS and one pulse per second (1PPS) clock training circuits. Second, the data processing path must be flushable (so that all sensors can be reset deterministically), and the output data must be precisely timestamped. Finally, access to some data network is required in order that the energy collected at the various sensors can be directed to some central node for processing.

The Picoceptor™ is designed to lock to a 1PPS clock from most any GPS receiver. The jitter on typical GPS units is on the order of one hundred nanoseconds. To improve time resolution, the "sloppy" 1PPS clock signal is cleaned up by the circuit in Figure 17. Timestamped data is provided in VITA-49[11] packets. VITA-49 is an industry initiative to standardize the format and routing of digital IF data among vendors.[16] Since the Picoceptor™ can attach virtually any USB device the networking options are great: Wi-Fi, Bluetooth, cellular modems, tactical radios, point-to-point microwave, etc.

The Picoceptor™ is currently being integrated into multiple TDOA systems. The performance is comparable to legacy geolocation systems except that it cannot accommodate the large ovenized oscillators required for the system to still operate ("fly wheel") in the absence of GPS. This is easily remedied with the use of an external GPS module with a suitable oscillator. Note that advanced oscillators such as the chip scale atomic clock (CSAC)[15] will likely soon remedy this problem.

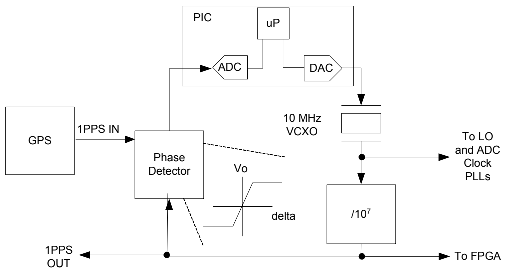


Figure 17. Clock Training Loop

4. CONCLUSION

This paper has described the design objectives and considerations that went into the development of the Picoceptor™ SDR platform. The architecture and development tools were discussed and several real-world applications were implemented and described for the platform. The Picoceptor™ represents a significant leap forward in SDR technology and is able to perform many functions at a fraction of the SWaP of legacy systems. By combining FPGA SOC, Embedded Linux, and USB OTG technologies, the Picoceptor™ provides a flexible architecture that is positioned to capitalize on the technology advancements in these areas for years to come.

5. REFERENCES

[1] DRS-Signal Solutions, "SI-8649 Datasheet", May 2008.
[2] http://sca.jpeojtrs.mil/
[3] David Brownell, The GNU/Linux "usbnet" Driver Framework, http://www.linux-usb.org/usbnet/
[4] http://www.vmware.com/
[5] http://www.ubuntu.com/
[6] http://ossie.wireless.vt.edu/
[7] http://www.gnu.org/software/gnuradio/
[8] Harter, Nathan M, "Development of a Single-Channel Direction Finding Algorithm", Master's Thesis, Virginia Tech University, 2007-04-13.
[9] DRS-Signal Solutions, "Piconote: Watson Watt Single Channel DF Implementation", July 2008.
[10] DRS-Signal Solutions, "Piconote: Time Difference of Arrival", July 2008.
[11] VITA Radio Transport (VRT) Draft Standard, VITA-49.0 – 2007 Draft 0.21 31 October 2007
[12] http://www.hypres.com/
[13] Gio Cafaro et. al.,"A 100 MHz–2.5 GHz Direct Conversion CMOS Transceiver for SDR Applications", Motorola Labs. 2007 IEEE Radio Frequency Integrated Circuits Symposium.
[14] DRS-CODEM, "AVM1 Antenna Manual", PN# 913187.
[15] R. Lutwak, et. al., "The Chip-Scale Atomic Clock – Recent Development Progress", Proceedings of the 35th Annual Precise Time and Time Interval (PTTI) Systems and Applications Meeting, December 2-4, 2003, San Diego, CA, pp. 467-478
[16] Bob Normoyle and Paul Mesibov, "The VITA Radio Transport as a Framework for Software Definable Radio Architectures", SDR Forum Conference, November 2008.