DESIGN AND PERFORMANCE TRADEOFFS IN DIGITAL RADIO PROCESSING ARCHITECTURES

Mujun Song, Jason R. Pennington, Mark D. Silvius, Ryan W. Thomas, Richard K. Martin
(Air Force Institute of Technology (AFIT), Wright-Patterson AFB, OH, USA,
{mujun.song.kr, jason.pennington, mark.silvius,
ryan.thomas, richard.martin}@afit.edu);
Charles W. Bostian (Center for Wireless Telecommunications, Wireless@Virginia Tech,
Blacksburg, VA, USA, bostian@vt.edu).

ABSTRACT

The level of reconfigurability in digital radios varies significantly. At one extreme, dedicated devices, using single-function analog components and dedicated digital chips, operate for a specific wireless application. These devices lack the flexibility to change or adapt to different communication requirements, waveforms, or applications. At the other extreme, software-defined radio platforms use A/D converters to sample at RF or IF frequencies, allowing reconfigurable software running on general processing architectures to operate over a wide variety of applications. While these devices may offer the most flexibility in terms of reconfigurability, they can also have significant overhead in terms of power consumption, cost, size, and performance. Between these two extremes lie *firmware defined radios*, a class of digital radio that is still highly reconfigurable, but uses firmware and modifiable components to determine at what point traditional software is run in the digital receive process. This paper investigates the tradeoffs between different processing architectures running in firmware and software, and reviews current processing architectures and products on the market that are using them. Next, the paper defines a series of metrics that can be used to evaluate and compare these architectures in a typical use case. We use these results to show the potential advantages and disadvantages of these processing architectures.

1. INTRODUCTION

In the last few years, rather than settling on a standard processing architecture, the software defined radio (SDR) world has instead produced an ever increasing number of variants. The SDR Forum has amalgamated all of these under their definition of a SDR [1]:

… a collection of hardware and software technologies where some or all of the radio's operating functions (also referred to as physical layer processing) are implemented through modifiable software or firmware operating on programmable processing technologies.

Despite falling under a single definition, the different processing architectures have a large tradespace in performance, cost, power and ease of use. For a researcher just entering into the SDR world, discerning what technologies are appropriate can be a daunting challenge.

Before we begin, we must define a new category of digital radio that we term *firmware defined radio* (FDR). The Institute of Electrical and Electronics Engineers (IEEE) Standard Glossary of Software Engineering Terminology, Std 610.12-1990, defines firmware as [2]

… the combination of a hardware device and computer instructions and data that reside as read-only software on that device.

From the above definition, *firmware* designates not only hardware, but also software. A key point when defining FDR is that a digital radio should be operated and controlled by the firmware. That is, it is *firmware* that defines the operation of the digital radio. And generally, firmware includes not only general purpose processors (GPPs) and digital signal processors (DSPs), but also field programmable gate arrays (FPGAs), and application specific integrated circuits (ASICs). However, ASICs are nearly impossible to be modified in a meaningful way once manufactured; therefore, FDRs are largely relegated to digital radios that are operated by FPGAs. In general, we define a FDR as

… a digital radio in which the operation is characterized by firmware as the primary functional actor. Firmware is defined as user-modifiable, read-only software that controls the hardware functionality.

To analyze the tradeoffs of different processing architectures in FDR and SDR systems, we utilize a common SDR functionality related to Dynamic Spectrum Access (DSA). Under a DSA scheme, SDRs and FDRs can sense radio spectrum, locate spectrum *whitespaces*, and

opportunistically use these vacancies to establish radio communication links.

We will use the baseline functionality required to accomplish this spectrum sensing task as a benchmark to compare different FDR and SDR processing architectures. Using this baseline requires taking precise digital samples of the environment; windowing the time series data; performing a Fast Fourier Transform (FFT); using the result to compute the power spectral density (PSD); and then estimating which of the frequency bins of the PSD has a signal present using a threshold test.

This baseline functionality occurs in several different DSA applications. For instance, in a public safety and disaster communications response scenario [3], cognitive radios (CR) performing DSA, identify the presence of existing signals in the frequency band. The CR can then join and communicate on an existing link, or act as a gateway to bridge multiple communications links together.

Existing solutions have relied heavily on GPP-based architectures. However, due to a large area form-factor, and consumption of a significant amount of power, these architectures could not be supported with a hand-held battery-pack. This limited deployments to ground-stations, or at most, vehicle-mounted applications [4]. To move beyond a fixed ground-station, or vehicle-mounted mobile radio design, it is clear that hardware miniaturization needs to occur to allow for the development of a small, power-efficient hand-held device.

Another application that uses this baseline to conduct spectrum sensing, is not individual radios, but a larger sensor network. Here a collection of CRs can relay spectrum utilization information back to central database, or *DSA Broker* [5], where the data can be fused to give a more accurate depiction of the radio environment. In addition to identifying the presence of signals, the multiple independent measurements allow a user to geo-locate the origin of the signals to complement the radio environment map [6].

For military applications, the presence, as well as the locations of the signals, can be significant in identifying adversaries in the band. This knowledge gives military planners the option of either avoiding adversaries' signals or jamming them. This process of identification has been demonstrated previously using a GPP-based architecture [7].

To develop a larger-scale sensor network, with multiple, battery-powered, inexpensive, and potentially disposable nodes, we would again need to focus on miniaturization, with the goal of developing small, power-efficient, disposable nodes.

Thus, this paper makes the following contributions. First, we review current processing architecture options and the most popular products on the market which utilize them. Next, using a set of metrics we developed, and explain in Section 3, we evaluate the three most common of these processing architectures, specifically focusing on the tradeoffs between SDRs and FDRs.

2. PROCESSOR ARCHITECURES

We begin by reviewing the processor hardware available for use in software defined radio and cognitive radios.

The GPP is one of the foundational processor architectures in SDRs. Its high clock rate and large on-chip memories make GPPs appropriate for SDR. However, because its main purpose is not only for signal processing, but also for performing various other system operations, it is less efficient and consumes more power than other devices. However, since the GPP is continuously being developed and improved, some specialty processing features are now found on GPPs.

A DSP is a microprocessor for digital signal processing applications. DSPs are flexible and can be programmed with a HLL (high level language) such as C. Through this HLL, modifications and upgrades can be made easily. Most DSPs execute sequential and parallel operations. Even if the DSP has provisions for parallel execution of instructions, such as parallel multiply and accumulate, the size and number of units are fixed, and they may not be optimal for a particular task. So, for high-sample rate process, more cycles are required and the data is processed slower. However, DSPs are comparatively cheaper than other hardware.

FPGAs are hardware devices that provide a matrix of reconfigurable logic resources on a single chip, which can be programmed to implement a user specified circuit. FPGAs can execute parallel operations, maintaining an optimal word size for a particular algorithm. Therefore,

Table 1. Evolution of FPGA Technology [8-11]

	LUTs (Per a slice)	Input capacity of LUT	Flip-flops (Per slice)	Slices (per CLB)	DSP	Capability of multiplier in DSP
Virtex-2	2	4-input	2	4	None	None
Virtex-4	2	4-input	2	4	32-512	18X18 multiplier
Virtex-5	4	6-input / 5 input-dual output	4	2	32-512	25X18 multiplier
Virtex-6	4	6-input / 5 input-dual output	8	2	288-2016	25X18 multiplier

they can achieve a much higher computational performance than a DSP. Furthermore, they require less engineering cost than an ASIC, because they are reconfigurable and can leverage existing intellectual property. FPGAs are programmed using Hardware Description Languages (HDLs). However, since FPGAs are highly configurable, HDLs are often considered more difficult to learn and use than HLLs.

Since modern FPGAs can meet many of the performance and power consumption requirements of ASICs, as well as the flexibility and low development costs of GPPs and DSPs, they are increasingly being used and improved.

The key evolutionary trends of FPGA development are: 1.) increasing resources (termed *slices*), 2.) the use of specialized embedded blocks, serving to improve delay, power, and area if utilized by the application, but waste area and power if unused, and 3.) device families with additional mixes of features [12]. As an example, Table 1 summarizes the number of look-up-tables (LUT) and flip-flops in the Xilinx family of FPGAs. The increase in the number of LUTs and flip-flops indicates that the computational power of each is getting larger. In addition, there is an increase in the input capacity of each LUT. However, since area increases with the number of inputs, but the logic depth decreases, the trend for larger LUTs also reflects an increase in interconnection logic delay.

The number of embedded DSP blocks in FPGAs reflects that embedded blocks are becoming increasingly common in FPGAs. For instance, the Xilinx Virtex 2 family has no embedded DSPs, but the Virtex 6 has a maximum of 2016 embedded DSPs in the FPGA. Also, the capability of each DSP has increased. At first, DSP had 18x18 multipliers, but now it has 25x18 multipliers, (of course, FPGA DSPs have more capabilities than just multipliers, such as adders and accumulators). There are also other specialized embedded blocks, such as soft-core and hard-core processors.

All of these processor architectures have been utilized in several different products, for varying market segments. We will highlight some of the most popular products, identify the processor architecture, and discuss some of the tradeoffs inherent in the product.

The Universal Software Radio Peripheral (USRP) comes in two models: the USRPv1 and USRPv2. The first uses an Altera Cyclone 1 FPGA; the second, uses a Xilinx Spartan FPGA for basic, high sample-rate processing, like digital up-and-down conversion. The GPP on the host computer is used for the more complex, low sample-rate processing operations. For this reason, we do not consider the USRPv1 to really be a FDR, since the FPGA is not directing the operation of the platform. The FPGA is just responsible for pre-processing, and it does not have the capability to do anything more. The USRPv2 can do more of both the high and low sample–rate processing operations in the FPGA, making it resemble more of a true FDR. Both devices are low cost – $700 for the USRPv1 and $1400 for the USRPv2 [13].

The Small Form Factor (SFF) SDR Platform by Lyrtech uses both a Virtex 4 SX35 FPGA and a specialized embedded DSP called a digital media processor (DMP) which consists of DSP and a GPP based on the ARM architecture. This platform has an FPGA, DSP and GPP, making it very difficult to categorize. Depending on how its processors are utilized, the SFF platform can be more SDR oriented or more FDR oriented. This is demonstrated by the SFF software development kit, which allows a user to develop C/C++ for DSP and GPP, or HDL code for FPGA. It also allows the user to generate code for the DSP and the FPGA using MATLAB and Simulink. It costs $3500 for board and $8500 for kit [14].

The wireless open access research platform (WARP) version 2 uses Virtex 4 FX100 FPGA as its primary processing unit. Although the radio can communicate with a GPP-based PC, the platform is designed to perform all radio calculation on the local FPGA. Even though it uses same Virtex 4 family as the SFF SDR by Lytrech, the FPGA in WARP is tweaked for high-performance logic applications. For example, it has features such as more logic slices (almost 3 times more slices than the SFF Virtex4) and more RAM. On the other hand, the number of DSP slices is almost the same as the SFF SDR. The WARP board costs approximately $8500 for a basic single-antenna, single radio model [15].

Although not commercially available, Kansas University has developed a SDR that uses Virtex 2 Pro P30

Table 2. Summary of Market Research

	USRP ½	Lyrtech	RICE/WARP	KUAR	BEE2/BEE3	TI DSK6416
Cost	$700 / $1400	$10,395	$3500	Not for sale	Not figured out	$495
FPGA	Cyclone 1 EP1C12 / Xilinx Spartan 3-2000	Virtex-4 SX35	Virtex-2 Pro P70 (V.1) / Virtex-4 FX100 (V.2)	Virtex-2 Pro V20	Virtex-2 Pro P70/ Virtex-5 (LX/SX)	•
Other elements (signal processing)	Host computer(GPP)	DM6446 DMP SoC from TI (DSP+MPU)	•	PowerPC 405		TMS320C6416

FPGA in conjunction with a 1.4 GHz Pentium M GPP. It has an internal power supply with a battery pack, so it can operate self-contained [16]. Since all radio functionality is handled on the GPP, we consider it a SDR.

The BEE2 uses 5 FPGAs (all Virtex 2 Pro P70s) [17]. The BEE3 uses 4 FPGAs (all Virtex 5s, of the LXT/SXT/FXT varieties). This is somewhat different from previous platforms [18], in that the original purpose of this platform is to make a FPGA base computer system. Since it has only FPGAs, a radio that utilizes a BEE would be considered a FDR.

An interesting GPP based processor architecture is found in the TI Beagleboard. The Beagleboard has an OMAP3530 application processor featuring the ARM Cortex-A8. The OMAP processor contains both a GPP (ARM) and a DSP (TI architecture). This GPP board is much cheaper than a host PC which is generally used with radio frontends such as the USRP. Even though the performance of the GPP on the Beagleboard is lower than a GPP in a laptop, the DSP can make up the deficiency. For example, it is expected that a low cost SDR can be developed by using the Beagleboard in place of a standard PC [19].

Finally, the TMS320C6416 DSP Starter Kit (DSK) is a low-cost development platform used for the development of high performance digital signal processing applications. The system uses the TMS320Cx DSP chipset. The board features USB communications for off-board connectivity. The system is compatible with TI's Code Composer Studio IDE and eXpressDSP Software. The hardware board sells for $400-500 [20].

Table 2 summarizes these products. Note that every platform except the TI DSK6416, whether considered to be an FDR or SDR, uses FPGAs in some capacity. Furthermore, some platforms use not only FPGAs, but also GPPs and/or DSPs. The only platform that uses only FPGAs is the WARP. Not surprisingly, FDR FPGAs have much more capacity such as more logic slices, larger RAM, more cores, more I/O interface, and more transceivers.

3. EXPERIMENTAL COMPARISON

This section defines the metrics that we used to characterize and evaluate SDR and FDRs. To evaluate the relative performance of three classes of digital radios (GPP, FPGA, and DSP), we investigated the performance of the hardware under various metrics. Each class of digital radio was investigated in the context of a commercial platform that contained a RF front end, consisting of digital and analog components, in addition to the processor being evaluated.

To eliminate cost disparities arising from the front end, we investigated the price of the processing chipset independently. Therefore, the *price of hardware* is defined as the approximate retail cost of a single instance of the processing chip in U.S. dollars.

We defined performance in the context of the spectrum sensing application. *Performance* is defined as the wall clock time required (in seconds) to complete one sense cycle (filter, FFT, and detect), once the A/D samples are in local registers (and assuming the sense cycle is not interrupted by another thread or process). In this case, a lower clock time represented a higher level of performance.

Working with fixed point resolution versus floating point has issues. Each mathematical operation leads to a loss of fidelity as rounding and truncation errors build. *Fidelity* captures the issues associated with these errors. We measured fidelity in dB lost over the SNR of floating point operation.

Hardware cost is not the only expense. The *price of software* metric measures the approximate commercial single seat license price in U.S. dollars of the standard development kit for the platform. It does not take into account support, maintenance or development hardware costs.

Power is measured as the number of Watts consumed by the processing chip alone. It excludes the front end power and support hardware power requirements.

The amount of time required to build a standard project on the processing platform given only a background in a standard, functional programming language, such as C or Java, is its *development difficulty*, or its learning curve. It is qualitatively measured based on our experience in developing systems on the platforms as low, medium, or high.

Figure 1 details the configuration of our experimental comparison. We used a RF capture board to collect radio samples of the environment. These samples were then passed to three architectures, which executed our FFT-based PSD sensor algorithm. These architectures were: 1.) a Linux PC laptop which used an Intel Core2Dou GPP,

Table 3. Architecture Evaluation

Evaluation	Price of Hardware (Dollars)	Performance (ns)	Fidelity (dB)	Price of Software (Dollars)	Power (Watts)	Development Difficulty (opinion)
GPP	2,000	100-200	-302	2,000	20-50	Low
DSP	231	406.111	-48	445	1.5	Medium
FPGA	2,188	7.4322	-302	4,000	1.44	High

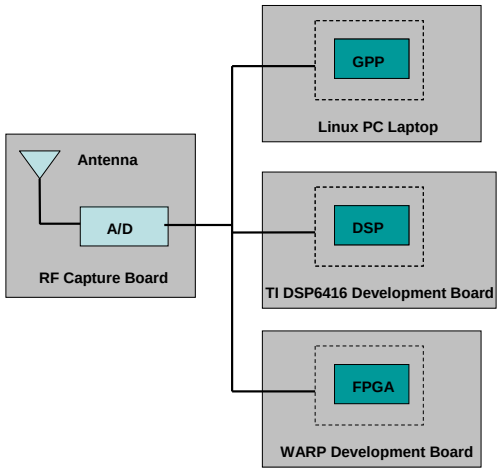

Figure 1. Experimental Setup

2.) the Texas Instruments 6416 DSP development board with a TMS320C6416 DSP processor, and 3.) the WARP development board with Virtex 4 FX100 FPGA.

For each system, we evaluated our metrics and listed our results in Table 3. Prices were based on actual acquisition costs for the software and hardware systems. Power statistics were computed via system specifications, but tailored to our particular processor configuration. We approximated performance results using software profiling and debugging tools, which tallied the clock cycles and run time consumed by the FFT-based sensor algorithm. We assigned the development difficulty rating as a qualitative judgment of the overall ease of use of the hardware and software design flow tools.

These numbers enumerate the strengths and weaknesses of each processor's architecture. For example, the GPP system had high fidelity, high ease of development, but high power constraints and moderately high costs. The DSP system had lower fidelity and performance, but was inexpensive, had low power requirements, and moderate ease of use. The FPGA had the highest performance, nearly highest fidelity, and low power constraints, but it had high costs and low ease of development.

Next, we plotted these metrics in a series of spider plots in Figures 2-4, to visually represent the trends. Here, the better, a more desirable metric value is represented by a larger quantity on the plot. For example, a large acquisition cost was a disadvantage and scored low on the plot, while a short execution time was favorable and scored high. These figures reveal that each processor tended to best cover their own unique region of the metric space, and the optimum choice for a processor may depend on the specific application.

4. CONCLUSIONS

The results in Figures 2-4 establish how processor architectures each have their own strengths and weaknesses, and how the optimum choice of a processor for a system may depend on that system's application. For example, the GPP architecture may be best suited for ground-based communication and spectrum sensing applications, where power is plentiful, and where high ease of development, high fidelity, and moderate costs are desired. On the other hand, in mobile or vehicle mounted devices where power is a constraint, and high performance and fidelity is a requirement, then FPGAs may be best suited. Or, in a battery-powered sensor network application, where low cost, low power is a must, at the expense of performance or fidelity, then a DSP system may be the best choice. Better understanding these design and performance tradeoffs will assist the designer when selecting a processor architecture for his/her specific system application.

5. ACKNOWLEDGEMENT

This work funded in part by the Air Force Research Labs, Sensors Directorate and the Air Force Office of Scientific Research. The views expressed in this paper are those of the authors, and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

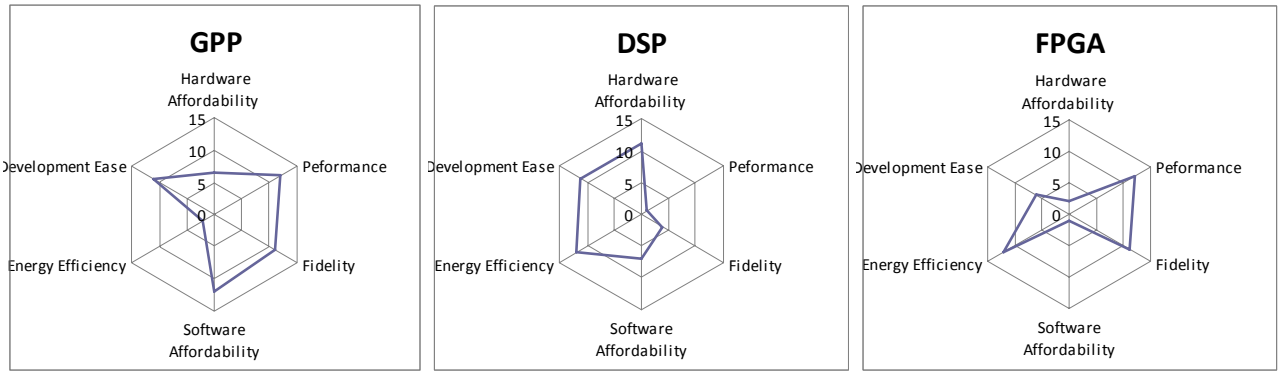

Figure 2-4. Experimental Results

6. REFERENCES

[1] SDR Forum, 2010. http://www.wirelessinnovation.org.

[2] Wikipedia, "Firmware," 2010. http://en.wikipedia.org/wiki/Firmware.

[3] M. D. Silvius, F. Ge, A. Young, A. B. MacKenzie, and C. W. Bostian, "Smart Radio: Spectrum Access for First Responders," in Society of Optical Engineering (SPIE): Wireless Sensing and Processing III, Orlando, FL, 2008.

[4] R. Rangnekar, F. Ge, A. Young, M. D. Silvius, A. Fayez, and C. W. Bostian, "A Remote Control and Service Access Scheme for a Vehicular Public Safety Cognitive Radio," in IEEE 70th Vehicular Technology Conference (VTC), Anchorage, AK, 2009.

[5] F. Ge, R. Rangnekar, A. Radhakrishnan, S. Nair, Q. Chen, A. Fayez, W. Ying, and C. W. Bostian, "A Cooperative Sensing Based Spectrum Broker for Dynamic Spectrum Access," in Military Communications Conference (MILCOM), Boston, MA, 2009, pp. 1-7.

[6] R. K. Martin and R. Thomas, "Algorithms and Bounds for Estimating Location, Directionality, and Environmental Parameters of Primary Spectrum Users," IEEE Transactions on Wireless Communications, vol. 8, pp. 5692-5701, 2009.

[7] A. A. Honore, R. W. Thomas, R. K. Martin, and S. H. Kurkowski, "Implementation of Collaborative RF Localization Using a Software-Defined Radio Network," in Military Communications Conference (MILCOM), Boston, MA, 2009, pp. 1-7.

[8] Xilinx, "Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet," November 5, 2007. http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf.

[9] Xilinx, "Virtex-4 Family Overview," September 28, 2007. http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf.

[10] Xilinx, "Virtex-5 Family Overview," February 6, 2009. http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf.

[11] Xilinx, "Virtex-6 Family Overview," January 28, 2010. http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf.

[12] P. H. W. Leong, "Recent Trends in FPGA Architectures and Applications," in 4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA), Hong Kong, China, January 23-25, 2008, pp. 137-141.

[13] Ettus Research LLC, June 11, 2010. http://www.ettus.com/.

[14] Lyrtech, "Small Form Factor SDR Development Platforms," 2010. http://www.lyrtech.com/Documents/product_sheets/Reference%20sheet%20-%20SFF%20SDR%20DP%20(hi_res).pdf.

[15] "WARP FPGA Board Overview," 2010. http://warp.rice.edu/trac/wiki/HardwareUsersGuides/FPGABoard_v2.2.

[16] G. J. Minden and the University of Kansas/ITTC, July 18, 2004. http://www.csg.ethz.ch/education/lectures/sdrn/KURadio_Overview_A50718.pdf.

[17] Berkeley Wireless Research Center, University of California, Berkeley, "Berkeley Emulation Engine," 2007. http://bee2.eecs.berkeley.edu/.

[18] BEEcube Inc, "BEEcube," 2010. http://www.beecube.com/.

[19] C. R. Anderson, G. Schaertl, and P. Balister, "A Low-Cost Embedded SDR Solution for Prototyping and Experimentation," in Software Defined Radio (SDR) Technical Conference, Washington, DC, 2009.

[20] Texas Instruments, "TMS320C6416 DSP Starter Kit (DSK)," 2010. http://focus.ti.com/docs/toolsw/folders/print/tmdsdsk6416.html.