IDA

INSTITUTE FOR DEFENSE ANALYSES

A Technical Review of Software Defined Radios: Vision, Reality, and Current Status

Lawrence N. Goeller
David M. Tate

May 2013
Approved for public release; distribution is unlimited.
IDA Document NS D-4878
Log: H 13-000555/1

INSTITUTE FOR DEFENSE ANALYSES
4850 Mark Center Drive
Alexandria, Virginia 22311-1882

The Institute for Defense Analyses is a non-profit corporation that operates three federally funded research and development centers to provide objective analyses of national security issues, particularly those requiring scientific and technical expertise, and conduct related research on other national challenges.

About this Publication
The views, opinions, and findings should not be construed as representing the official position of either the Department of Defense or the sponsoring organization.

INSTITUTE FOR DEFENSE ANALYSES

IDA Document NS D-4878

A Technical Review of Software Defined Radios: Vision, Reality, and Current Status

Lawrence N. Goeller
David M. Tate

A Technical Review of Software Defined Radios: Vision, Reality, and Current Status

Lawrence Goeller and David Tate
Cost Analysis and Research Division
Institute for Defense Analyses
Alexandria, VA

***Abstract*— Since the 1980s, the US military has been seeking a way to improve the ease and flexibility of communications within and between the Services. Forces deployed at that time used dozens of different radios, most of which could only communicate with other radios of the same type. Software Defined Radio (SDR) offered the promise of not only interoperating with all existing radios, but also allowing those legacy radios to communicate with each other. Just as important, SDR would enable future communications systems to be implemented on already-deployed hardware. This vision of interoperability and forward compatibility has not been realized, despite billions of dollars of investment in the Joint Tactical Radio System (JTRS). The authors reviewed JTRS program archives and studied the hardware and software architectures to determine whether there are fundamental technical reasons behind this failure. Our findings, which invoke the familiar tradeoff between performance and power consumption, are summarized in this paper. We identify three different architectural approaches used by the JTRS community over the years, and discuss why none has been able to realize the program's goals.**

Keywords*— *Software defined radio, joint tactical radio system, waveform portability, field programmable gate array, software communications architecture

I. THE SDR VISION

The Office of the Secretary of Defense (OSD) has taken a strong interest in facilitating interoperable tactical communications within and between the Services since the 1980s. At that time, more than 30 "stove piped" waveforms[1] were supported by traditional hardware-based radios throughout the Department. This was also the dawn of the personal computer (PC) era. These devices were crude but *flexible*; they could perform a wide variety of tasks, from word processing to database searches to e-mail. New capabilities could be added after purchase simply by loading new software. At the same time, tasks that had always been accomplished using analog physical processes (e.g., photography and sound recording) were being replaced by digital processes. It occurred to several people that the solution to radio interoperability might be to use a PC-like machine to *synthesize* waveforms using software-controlled digital logic, in much the same way that modern keyboards can synthesize acoustic waveforms to mimic analog musical instruments. This led to the vision of the software defined radio (SDR): a single box of software-controlled digital hardware that could emulate any radio waveform, with the groundbreaking concept that new waveforms could be added as required simply by loading new software.

After the apparent success of the Defense Advanced Research Projects Agency (DARPA) SPEAKeasy software radio, initiated in 1990 with a follow-on second phase in 1995, the Army initiated a program to develop and field tactical software radios. Originally called the Programmable Modular Communications System (PMCS), this program eventually sired the several programs of the Joint Tactical Radio System (JTRS). Both the Services and OSD intended that, over the next few decades, JTRS radios would replace all legacy radios in the Services' inventories. This would give all radio users the physical ability to communicate with any other user, resolving, at least in large part, the interoperability problem. Unfortunately, all of the JTRS programs struggled in development. Despite a series of reorganizations, the contract for the Ground Mobile Radio (GMR) JTRS program was terminated in 2011 after a Nunn-McCurdy breach, and the Airborne, Maritime, and Fixed (AMF) JTRS contract ended in 2012 without producing any fielded hardware. Those JTRS programs that are still producing hardware have had their requirements and scopes reduced drastically since 2002, and often struggle in operational tests.

What went wrong? Many argue that the problems were entirely management-related; others suggest that the trouble started when the Net-Centric Warfare requirements (such as the Wideband Networking Waveform, or WNW) were emphasized after 2000. However, the ubiquity of practical problems across different Services and contractors hints that there might be unresolved technical problems inherent in the SDR vision itself. What are they—if they exist—and can the still-attractive SDR vision be salvaged with newer technology or a different approach? These are the questions that the Office of the Under Secretary of Defense for Acquisition, Technology and Logistics' Performance Assessments and Root Cause Analyses (PARCA) organization asked the Institute for Defense Analyses to investigate.

This effort was sponsored in part by the Office of the Under Secretary of Defense for Acquisition, Technology and Logistics/Performance Assessments and Root Cause Analyses (PARCA).

[1] In this paper, we will define *waveform* as the techniques used to imprint user information onto a radio wave, including but not limited to modulation, channel encoding, and multiplexing.

II. What a Radio Does

With few exceptions, the function of any radio is to use a baseband signal to modulate a higher-frequency sine wave (a *carrier*), convert this modulated signal into a radio frequency (RF) wave, and then transmit it via an antenna. The receiving radio reverses the process. The baseband signal, encoding user voice, data, or video, may be analog or digital. The carrier wave can be at almost any frequency. In traditional radio technology, this modulation/ demodulation process is performed via non-linear mixing and amplification of the baseband and carrier signals; hence, such devices are often called mixers. Other operations may be performed on the baseband signal—before modulation, such as channel encoding (if the signal is digital); during modulation, such as frequency-hopping; and after modulation, such as multiplexing this signal with others transmitted on the same carrier. The combination of channel encoding, modulation, multiplexing, and other processes is the *waveform*.

Waveforms can be analog or digital. These terms refer to whether the baseband information is processed as a continuously-varying voltage (analog) or as a quantized series of discrete values at regular time intervals (digital). To demodulate an analog signal in the traditional way, the receiver uses another mixer to combine the incoming modulated signal with an internally-generated copy of the (un-modulated) carrier wave. Baseband information can be recovered from digital signals via the mixing process, albeit via a different methodology. It is a common fallacy to assume that digital signals can only be recovered with a "digital" radio, where this term is used to refer to a logic-based transceiver (discussed next). For the rest of this paper, we will refer to radios as either mixer-based or logic-based, and not by the misleading terms *analog* or *digital*; either radio type can transmit and receive both analog and digital waveforms.

III. Synthesizing Waveforms

Waveform synthesis and recovery are techniques that fall under the general heading of *digital signal processing* (DSP). Logic-based devices that convert analog to digital signals are called analog-to-digital converters, or ADCs; devices that do the reverse process are called digital-to-analog converters, or DACs. Analog-to-digital conversion is typically done by *sampling*: the amplitude of a continuously-varying signal is rapidly but carefully measured many times per second, and each value is converted into a discrete quantity. The reverse process uses a reconstruction filter to create an output voltage that is proportional to each discrete quantity. Both ADCs and DACs fall into a category of devices called *digital signal processors*. To distinguish the acronym for digital signal *processing* from that of a digital signal *processor*, we will refer to the former as DSP and the latter as a pDSP, where the lower-case "p" stand for "programmable."

In practice, most of what pDSPs do is multiply and/or add binary numbers. To multiply two numbers, high and low voltage values associated with the logical 1s and 0s propagate through a hardwired circuit. The calculation is complete when the output voltages in the product register stabilize. Dealing efficiently with the large number of "carries" that occur when binary numbers are added and multiplied leads to complex physical designs. Every change of state of a transistor in a logic circuit draws some power, and produces some heat. For tactical radios, in which the source of power is a battery, these effects drive the fundamental performance tradeoff. More complex processing requires more electrical power, which both drains the battery and produces heat that must be dissipated.

To synthesize a waveform, one begins with digital representations of both the baseband signal to be transmitted and the carrier wave. Because the carrier wave repeats exactly, sampled values can be stored in a lookup table. The radio uses a dedicated multiplier circuit and a DAC to combine the baseband and carrier, producing a signal that closely resembles *what would have been produced* by a traditional mixing circuit. Note that despite the use of a DAC in this circuit, the device can emulate both digital and analog waveforms.

Demodulation of a transmitted waveform in a logic-based radio is similar. The received signal is first sampled and digitized; note that this sampling occurs even if the received waveform is "digital." The digitized samples are then fed into another multiplier circuit, where the other input is once again a series of numbers that represent the value of the carrier wave at various points in its cycle. The output is equivalent to what would have come out of the multiplier-based demodulator in the mixer-based design.

Virtually all modern military radios (as well as commercial smart phones) use a logic-based, rather than a mixer-based approach for both analog and digital waveforms. However, such devices are not software defined radios; the logic core is simply a specially-designed pDSP. This device can be changed via software only to a limited degree; for example, a different set of values can be loaded into a lookup table to represent a different carrier frequency. These devices are customized for specific waveforms, and this constraint is hardwired into the digital logic itself. In general, it is not possible to install a new or different waveform onto such a device after it has been fabricated. (One can download "apps" to a smart phone, but one cannot convert a 3G phone to a 4G phone via a software upgrade alone.)

IV. Technical Approaches to SDRs Over Time

In trying to understand why all of the JTRS programs have struggled, the authors have pored over a great deal of archival documentation on the early years of the program. We have found strong circumstantial evidence that the fundamental technological path that was being used to implement the SDR vision within the JTRS programs changed twice, reflecting three fundamentally different architectural approaches. The first approach assumed that waveform processing would be performed by general purpose processors (GPPs), similar to the core chips in desktop computers. The second approach was based on an early generation of field-programmable gate arrays (FPGAs) comprising large numbers of small, identical logic blocks. The third approach took advantage of a later generation of FPGAs with extensive embedded proprietary intellectual property (IP), often referred to as System-on-a-Chip (SOC). Both the GMR and AMF JTRS programs were planning to use this third approach at the time their contracts were terminated.

A. SDR via General Purpose Processor

Archived JTRS-related documents from the late 1990s through at least 2001 explicitly refer to the processor at the heart of the proposed SDR as "Pentium," "PowerPC," or other examples of well-known GPPs of the time. Early JTRS program briefings describe waveform development purely in terms of high-level programming languages and commercial middleware designed to run on GPPs. The software archiving and maintenance plan for the waveforms to be shared among programs made no provision (at first) for lower-level code. Early estimates of the amount and type of software that would be required to implement the waveforms did not include any additional effort for low-level programming. By the time the Cluster 1 (later JTRS GMR) contract was awarded, the program's Operational Requirements Document (ORD) [1] defined a waveform application as "a re-useable, portable, executable software application that is independent of the JTR System operating system, middleware, and hardware." The incremental proof-of-concept activities funded by the JTRS program[2] were all implemented using GPPs for the waveform synthesis. These references strongly suggest that it was originally assumed that the JTRS programs would implement their SDR designs using GPP technology.

The advantage of using a GPP is that any program can run on any computer, once a suitable compiler has been implemented. In the case of software radio waveforms, this creates the potential for *waveform portability*. Portable waveforms would be written like other applications in a high-level language such as Ada or C++. These high-level programs would then be translated into a mid-level assembler language by a compiler. The resulting "object code" is specific to the underlying hardware, but the "source code" is not—hence its designation as *portable*. GPPs all use a similar architecture: an arithmetic-logical unit (ALU) supported by a data path that includes a number of memory elements for storing instructions and data (called *registers*), as well as an instruction-decoder circuit and a control store that holds the information needed to process each instruction. The entire system is tied to a clock that maintains synchronization across all of the elements of this *central processing unit*, or CPU.

CPU functionality includes simple operations such as "add" and "shift," but not more complex arithmetic operations such as "multiply"—the most common task in digital signal processing. When the CPU is directed to multiply two numbers, it performs it via an extended sequence of shifts and adds that requires many clock cycles. This is very different from the fast but inflexible pDSP approach described earlier. Further, the pDSP, unlike a GPP, performs multiplications in the same amount of time every time, a critical factor in real-time processing. Not only is there a risk that a GPP might not always be fast enough to handle the real-time requirements of waveform synthesis and interpretation; there is also a risk that the GPP will require too much power and/or generate too much heat. These problems exacerbate each other, since more powerful (faster) processors draw more power and generate more heat.

[2] These activities are referred to in program documents as Step 1, Step 2A, Step 2B, and Step 2C.

As part of the risk mitigation process for the JTRS program, the Army funded development of a series of prototype radios. These radios demonstrated portable software implementing relatively simple RF waveforms and networking capabilities using GPPs. However, the actual military waveforms required by the JTRS program were significantly more complex and demanding in their processing requirements, and the radio sets on which they were to be deployed were strictly limited in their permitted size and thermal emissions. No JTRS radio uses a GPP as the modulator/demodulator device today.

B. SDR via Field Programmable Gate Arrays

In the 1980s, a company called Xilinx had developed what they called a Field Programmable Gate Array (FPGA) as a test bed for large circuits composed of pDSPs and other design elements. Its architecture was quite different from a conventional pDSP; instead of logic gates wired together on a chip to form hardware multipliers or shift registers, an FPGA is composed of many small identical logic elements, each typically consisting of a small lookup table and some memory. These logic blocks do not connect to each other, but rather to one or more of a series of parallel wires (called *routing channels*) that surround each block. Connections between the internal logic of each block and the routing channels are controlled by transistors that can be opened or closed on command.

The essential SDR-related feature of the FPGA was that, by configuring the logic blocks and properly selecting values for the lookup tables, the resulting circuit could emulate virtually any other digital logic element. Best of all, these connecting transistors could be controlled by a software program. Mid-level programming languages called *hardware description languages* (HDLs), originally used to emulate circuits in computer-aided design tools, could also be used to "write circuits" on a given FPGA. In principle, these configured circuits could emulate a hardware pDSP that itself emulates a mixer-based radio waveform. Then, when desired, that FPGA could be reconfigured to emulate a different waveform.

By 2003, it was clear to the developers of the JTRS radios that it would be impossible to implement all waveform features on GPPs, for the reasons discussed above. FPGAs seemed like the ideal alternative: they could preserve the necessary programmability of the radio set and most of the portability, while using less power than a GPP. However, there were two significant consequences of this approach. The first is that the FPGA-based emulated circuit used many times the number of transistors that a dedicated pDSP would have used to perform the same function; one study in 2006 [2] found the difference to be more than a factor of ten. As a result, they still draw considerably more power (and generate more heat) than pDSP implementations. The second is that HDL code is not really portable the way C++ is. Since the resulting "circuit diagram" is completely dependent on the physical layout of the underlying hardware, the HDL "source code" only works on one specific FPGA type. This conflicts with the fundamental vision of the SDR as a system in which the software (the waveform) is independent of the hardware. In principle, the HDL code could be "recompiled" separately for each

individual FPGA type. In practice, the limitations of the HDL circuit layout capabilities, coupled with the demands of real-time processing, meant that time-consuming hand-tweaking of the code was invariably required when moving from one hardware platform to another.

The SDR community maintained its commitment to the separation of waveform software and underlying hardware by proposing to expand the standard for the interface between them. These efforts are discussed in Section V.

C. SDRs via System on a Chip

The "sea of logic blocks" FPGAs performed poorly compared to a pDSP. Manufacturers realized that they would have to improve their performance if they were to be successful in the commercial marketplace, where portability and forward-compatibility are less important. Xilinx (and their new competitors) chose to add more configurable logic blocks to their designs, and also to embed a large number of proprietary special-purpose hardware memory circuits, hardware multipliers, other pDSPs, and even microprocessors into the newer generations of chips. Collectively, these embedded elements are referred to as *Intellectual Property*, or IP. The trend has been for more and more DSP functionality to be offloaded to specialized IP, leaving the configurable logic blocks to implement generic memory and "glue logic" among the IP modules. While these devices are still technically FPGAs, they are also starting to be referred to as *System on a Chip* (SOC).

SOCs are much more capable and, in general, more power-efficient than their "sea of logic blocks" precursors. Even so, the previous problems remain. The embedded IP elements are no more flexible than other pDSPs; only the ability to change how they are connected to each other has been added. Despite all the advancements, SOCs still draw significantly more power and produce more heat than non-reconfigurable pDSPs. Embedded IP is in fundamental conflict with the goal of waveform portability; code written to take advantage of the efficient IP of a specific SOC will be useless when porting to a radio that uses a different SOC. Both the GMR and AMF JTRS designs were using the SOC approach when their contracts ended.

V. Software Communications Architecture

In early 1999, the program office contracted with a consortium of radio vendors for a series of proof-of-concept activities in support of the JTRS ORD. This Modular Software-defined Radio Consortium[3] (MSRC) was tasked to develop a JTRS program management plan, design an architecture for the JTRS radios and software, demonstrate software-based waveform implementation and porting, and demonstrate software-based wireless networking. The ORD at that time specified certain shortcomings of existing systems that were to be addressed by the JTRS program:

- [Current systems] "do not employ an open systems architecture.
- require extensive depot level equipment and/or component changes to implement new capabilities in installed platforms.
- do not allow incremental or modular upgrades to increase the choices of waveforms and the bandwidth within those waveforms, or modify message system standards."

The vision for remedying these shortcomings promoted by OSD was that any existing or future waveform could be implemented in software without reference to the radio hardware. Waveforms would be implemented as portable software modules, reusable with minimal reconfiguration on any JTR radio with the proper power amplifiers, user interface, and antenna. The mechanism for achieving this portability was to be a software architecture standard that would enforce defined interfaces between the waveform software and the radio hardware. This standard, the Software Communications Architecture (SCA), was developed with input from OSD, the MSRC, and the commercially-focused Software Radio Forum (SWRF). The goal was to have a specification that ensured that any SCA-compliant waveform could be adapted to run successfully on any SCA-compliant radio set with only minimal customization. As Raytheon senior vice president Frank Marchilena said, "It works like a laptop—point, click, and download a waveform. The black box no longer limits battlespace communication."[3]

The SCA as implemented used the commercial standards CORBA (for object/device virtualization) and POSIX (for real-time control) as its key interface standards. This choice effectively assumed that all waveform software would be running on GPPs, or (at minimum) that pDSPs and FPGAs were, or would become, as flexible as GPPs. This turned out to be an invalid assumption, with terrible consequences for the programs. The space and power restrictions, thermal constraints, and real-time processing requirements of military radios made it impossible to process even simple waveforms without widespread use of hand-coded FPGA and pDSP modules. This hand-coded software was not SCA compliant; it had to bypass the CORBA middleware in order to directly control the low-level hardware. As a result, the code was not portable. Since this low-level coding was the hardest part of implementing a waveform, having working waveform software for one radio set was not much of a head start toward implementing that waveform on a different radio set. All parties—government, contractors, and SWRF—were aware of these problems by 2004.

In the mid-2000s, some in the JTRS community proposed extensions to the SCA that would establish standards for direct control of low-level hardware, with the intent of making portable waveform software possible again. By this time, however, several of the JTRS programs had already committed to radio designs that would not be compliant with this new standard, and those programs were under intense external schedule pressure already; they could not start over with new designs. It is possible that this approach could have succeeded, but it must be recognized that any choice of interface standard would impose significant constraints on future generations of

[3] The original members of the MSRC were Raytheon Systems, ITT Aerospace/Communications, Rockwell-Collins, Marconi Aerospace Systems, and Rooftop Communications.

hardware, which the commercial community might not be interested in supporting. Standards are generally only successful in performance domains that are no longer cutting-edge.

VI. Conclusions

The SDR vision is a single box of software-controlled digital hardware that can emulate any radio waveform, including new and ported waveforms, simply by updating software. The three key goals are:

- Forward compatibility of radio hardware, so that new waveforms or capabilities can be implemented through software upgrades alone.
- Portability of waveform software, so that new software does not have to be developed for every new radio set.
- Open architecture, so that third-party vendors can bring new waveforms to market without the traditional high barriers to entry.

This vision remains very alluring. However, it appears safe to conclude at this point that achieving this vision for military radios will be very difficult, if not impossible, especially given the cutting-edge performance requirements of military systems. A review of history shows that at least three different technical approaches have been tried—the first two were quietly abandoned by the industry some time ago; the third approach has produced some useful radios, but does not realize the goals of forward-compatibility and waveform portability.

Can the SDR vision ever be implemented? Perhaps, but it must be concluded that the programmability vs. power consumption tradeoff remains huge. SDR advocates should also acknowledge that, since FPGA/SOCs are inherently commercial products and will only be on the market for a few years, dependence on them for interoperability will require porting legacy waveforms to new systems (as the old ones become obsolete and unsupported), *independent of whether porting gets any easier*.

We also note that the current FPGA/SOC product line is highly diverse; each vendor produces dozens of different configurations optimized for different needs. The SDR vision was implicitly based on the assumption that one underlying piece of hardware could be used to support any waveform; buy any FPGA-based system now, and you could use it forever (so the original vision went), keeping it current via software upgrades only. The actual technological trend clearly shows that this "one size fits all" paradigm does not apply, at least in the commercial world.

The current generation of SCA is, by all accounts, not up to the task. Proposed extensions to standardize low-level hardware control have never been ratified by all parties. Discussions have taken place about replacing the current version of SCA with a more specialized approach designed from the ground up to focus on communications, with system calls optimized accordingly. This may be worth pursuing, but it remains an open question as to whether the separation of software from hardware in a non-GPP-based system with real-time requirements will ever be practical.

Moore's Law also remains in effect; a vision that sees hardware staying in service for years or decades, upgraded only via software, is nothing like the path that personal computers and cellular telephones have taken. In light of this, we note that a different approach to the problem of waveform interoperability has grown up in the past few decades. New generations of radios have taken advantage of improved hardware to fit a dozen or more different hardwired waveforms into a box. Each waveform has its own pDSP, or a portion of a larger device. These radios are operationally effective and relatively power efficient, but cannot support additional waveforms via a software upgrade alone; to add a waveform, new digital electronics must be inserted. If the vision of waveform upgradability is to be realized via this strategy, the relevant standards will not be software architecture standards like the SCA, but rather hardware interface standards describing the connections between the digital processing segment (the "motherboard") and the rest of the radio. In this vision, a waveform would be more like a graphics or network card in a personal computer—proprietary hardware performing well-specified processing functions that the CPU cannot do efficiently, according to a well-defined interface standard. This would be nothing like SDR, but it might be the only way to realize the vision of SDR.

References

[1] Operational Requirements Document (ORD) for Joint Tactical Radio (JTR), 23 March 1998.

[2] Kuon, I. and J. Rose. "Measuring the gap between FPGAs and ASICs." *Proceedings of the 2006 ACM/SIGDA 14th international symposium on field programmable gate arrays* ('FPGA '06), 21–30.

[3] Raytheon Company press release, 23 February 1999. Retrieved from http://www.thefreelibrary.com/Raytheon+Consortium+to+Define+21st+Century+Digital+Radio+Architecture.-a053930229 on 15 April 2013.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
xx-05-2013	Final	

4. TITLE AND SUBTITLE

A Technical Review of Software Defined Radios: Vision, Reality, and Current Status

5a. CONTRACT NUMBER

DASW01-04-C-0003

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Goeller, Lawrence, N.
Tate, David M.

5d. PROJECT NUMBER

5e. TASK NUMBER

C7112

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandra, VA 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

IDA Document NS D-4878

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandra, VA 22311-1882

10. SPONSOR/MONITOR'S ACRONYM(S)

IDA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Since the 1980s, the US military has been seeking a way to improve the ease and flexibility of communications within and between the Services. Forces deployed at that time used dozens of different radios, most of which could only communicate with other radios of the same type. Software Defined Radio (SDR) offered the promise of not only interoperating with all existing radios, but also allowing those legacy radios to communicate with each other. Just as important, SDR would enable future communications systems to be implemented on already-deployed hardware. This vision of interoperability and forward compatibility has not been realized, despite billions of dollars of investment in the Joint Tactical Radio System (JTRS). The authors reviewed JTRS program archives and studied the hardware and software architectures to determine whether there are fundamental technical reasons behind this failure. Our findings, which invoke the familiar tradeoff between performance and power consumption, are summarized in this paper. We identify three different architectural approaches used by the JTRS community over the years, and discuss why none has been able to realize the program's goals.

15. SUBJECT TERMS

Software defined radio, joint tactical radio system, waveform portability, field programmable gate array, software communications architecture

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Goeller, Lawrence, N.
Unclassified	Unclassified	Unclassified	Same as Report	9	19b. TELEPHONE NUMBER *(Include area code)* (703) 575-4678

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18