

SOFTWARE COMMUNICATION ARCHITECTURE RADIO ENVIRONMENT PERFORMANCE CONSIDERATIONS

Timothy M. Schoenfelder (Rockwell Collins Inc., Cedar Rapids, IA;
tmschoen@rockwellcollins.com)

ABSTRACT

This paper addresses methodologies and applicable metrics that help achieve optimum performance for a Software Defined Radio (SDR) application. Implementers must identify and examine the critical metrics of waveforms and the operating environment(OE) processes that enter into the overall radio communications path in order to successfully determine the performance of a waveform on a Software Communications Architecture (SCA)[1] Radio.

Requirements analysis and functional decomposition yields key metrics which lead to a solid comprehension of the user to antenna data paths as well as waveform control. These key metrics are crucial to understanding performance as each individual radio processor and interface must be able to support the waveform performance requirements while also recognizing the unique attributes of that particular waveform. SCA optimization methodologies within the scope of these metrics can enhance SCA performance.

1. INTRODUCTION

With SCA becoming the *defacto* standard SDR architecture, especially in the military radio communications field, the portability that SCA implies brings into consideration the porting of new and existing waveforms to new and existing hardware. New functionality or features brought to the radio by the respective waveform under consideration may also drive hardware revisions as well as software revisions to ensure a successful implementation. Essential to this effort is ensuring that key elements of the Waveform and SCA environment are adequately characterized to make certain that any necessary hardware and software changes are captured.

Understanding the importance of the underlying waveform requirements, this paper focuses primarily on the waveform to OE interaction along with any resulting impact on associated platform hardware.

2. BACKGROUND

SCA intentionally allows design flexibility by imposing constraints on the interfaces and software structure but not on the implementation of functions. This flexibility offers the implementer many design choices and decisions. If these design choices are not adequately researched with respect to the implementer's particular application, the design may certainly be less than optimum. Less than optimum design choices prevent a design from achieving its intended potential; these, therefore, have led to a number of SCA myths[2] including the perception of SCA implementations as being heavy-weight both from a memory and processing standpoint. This in turn suggests that SCA applications are inherently slow.

While the above myths are rooted in non-optimized SCA implementations, where there is risk, there is also opportunity. The requirements analysis and functional decomposition, key metrics, and SCA optimization methodologies described in this paper seek to provide an overview for a successful design while also addressing some of the key underlying causes resulting in SCA performance deficiencies.

3. REQUIREMENTS ANALYSIS AND FUNCTIONAL DECOMPOSITION

Model-Driven Architecture (MDA) provides a process to analyze and decompose requirements. Implementers can analyze waveforms individually or collectively as a composite waveform. This process yields and further refines desired metrics via three primary models.

3.1 Computationally Independent Model (CIM)

The CIM describes a single waveform or a representative composite waveform with the most demanding attributes/requirements. A CIM from a Radio Frequency (RF) frontend-to-user perspective includes radio parameters as well as signal processing and networking protocols.

The modem level radio parameters involve the following: instantaneous dynamic range, selectivity, induced distortion, phase ripple/flatness, spur-free dynamic

range, AGC dynamics, channel skirt, linear and also non-linear (constant envelope) transmissions and receptions.

Another aspect of the frontend-to-user perspective is waveform signal processing. binary correlations, Galois-field arithmetic, FFT, CIC, digital filtering, Viterbi, Trellis, BCH, generalized convolutional codes, binary-logic as well as decimal and binary mathematics are a few of the mathematical operations which describe waveform signal processing.

The frontend-to-user facet of a CIM also can deal with network waveforms.[4] If the network uses an internet protocol(IP) then RFCs may be implemented by IP stacks. Due to the layered nature of a network, the network waveform may invoke linking and crossbanding protocols. These waveforms also encompass subnetworking layers, particularly layers 2 & 3a, which include channel/media access (e.g. MAC addressing protocols), neighbor discovery, adaptive link control, acknowledgment, retransmissions, and routing. Additionally network waveforms may also utilize multiple red enclaves, black and red processing, red and black IP or red only IP, 'link level' or 'IP sec' security.

The above radio attributes comprise an overview of the functionality that an implementer may encounter.

3.2 Platform Independent Model (PIM)

The PIM[3] involves performing a functional analysis and decomposition of the waveform to determine its anticipated target radio architecture. This characterizes the data flow, bandwidth, information assurance, and latency requirements in terms of required memory, processing power, encrypted overhead traffic, baseband throughput, IF processing, and RF front-end requirements which yields a representative hardware platform.

3.3 Platform Specific Model (PSM)

The PSM[3] further maps the PIM's waveform functionality onto a specific hardware design. Allocating functionality to appropriate respective hardware maximizes portability as well as efficiency. Functionality is typically allocated by available processing time. The GPP generally executes the most time intensive and complex operations with relatively low bandwidth. A DSP, if used, generally handles less complex computations requiring faster processing or higher bandwidth while the FPGA is best suited for simple-time critical and bitwise type operations.

4. KEY WAVEFORM METRICS

The above requirements analysis and functional decomposition presents a process by which the design can be implemented successfully. This method can also provide

several key metrics or Most Important Requirements (MIRs) necessary to optimize the design in an SCA environment. The next sections further describe these key metrics.

4.1 Memory Footprint

The waveform's Static and Dynamic Component Memory Footprints provide basic metrics for determining the size of non-volatile flash and random access memory (RAM) respectively. The Static Memory Footprint dictates the amount of flash memory required to store the waveform while the Dynamic Memory Footprint is the approximate waveform RAM required to support the waveform once it is loaded as follows:

1. Static Memory Footprint \cong static uncompressed waveform image size.
2. Dynamic Memory Footprint \cong (Static Memory Footprint + dynamically created directories + dynamically created files + heap space).

4.2 SLOCs, Classes, and Threads

Counts of software lines of code (SLOCs), classes, and threads by component give developers insight into porting, integration, and testing efforts via code size, complexity, and concurrency. Concurrency and threading are discussed further in section 5.5 Concurrency and Threading Priorities.

4.3 Processor Resource Utilization

Special attention must be directed toward waveforms that utilize computationally intensive algorithms to ensure that respective processors are chosen properly and sized correctly. For instance, HF waveforms, with relatively low datarates, execute extensive floating-point based decoding on baseband signals within the DSP to extract data from noise.

The Million of Instructions per Second (MIPS) metric may be one of the most difficult metrics involved with SDR as MIPS means different things to different audiences. A relevant question is, "How many MIPS does the OE and waveform use at normal vs. at peak demand?" Time Division Multiplexed Access (TDMA) waveforms with more constant data flow naturally pose a more straightforward approach to this metric than IP based waveforms.

The implementer should verify the metrics against a given platform as specifications may appear valid merely as a result of their presentation – 'window dressing'. For example, a team may find that hardware vendor specifications are consistently low or high. To validate a performance specification, run a benchmark profile based on activities that the application is required to implement. Another example might be benchmarking a

Communications Entity (CE) only to find that it transfers data well unidirectionally but cannot turn its interfaces around quickly enough to effectively work with bidirectional data.

4.4 Security

Security plays a vital role especially in Military SDRs. It complicates SCA implementation by placing additional constraints on data and control such that many typical SCA optimizations cannot be implemented. Security in this regard, while important, is beyond the scope of this paper.

4.5 Encrypted Traffic Overhead

Encrypted overhead traffic[5] is tied into the military waveform domain. It is further decomposed into Communications Security (COMSEC) and Transmission Security (TRANSEC) traffic. This traffic is in addition to regular data throughput and decreases the amount of usable baseband throughput available to the waveform. $\text{Data throughput} \approx \text{baseband throughput} - \text{encrypted traffic overhead}$. Including this metric in the design ensures adequate baseband throughput is accounted for and available to support the waveform's bandwidth requirements.

Additional processing resources above and beyond the predicted waveform traffic may be required to perform both COMSEC and TRANSEC security functions.

4.6 Interprocessor Interface Bandwidth

Intrinsically, SCA describes a distributed system with many computational elements which may include one or more General Purpose Processors (GPPs), Digital Signal Processors (DSPs), Field Programmable Gate Arrays (FPGAs) and Embedded Cryptographic Subsystems (CSS). As with all distributed systems the inter-element communication bandwidth must be considered within the waveform and system design. These metrics include the traditional bus speeds of the interfaces between computation elements (e.g. Ethernet, PCI) as well as any overhead associated with communication transport over the bus (e.g. IP header).

Memory speed in particular plays a critical role with many processors having the ability to execute code faster than instructions and data can be fed to them. This can trump processor speed as a critical metric; thereby, assuring its place in system design[6,7].

Understanding the impact of interface and transport selection for interelement (or interprocessor) communication is instrumental in determining the overall throughput of an SCA system.

4.7 Timing Methodology

Correct time-sourcing within a distributed architecture ensures that all waveform components and processes are synchronized. Smith, Demirbilek, and Bicer[8] divide multiprocessing waveform timing issues into three classes: Waveform Synchronization, Intercomponent/Interprocessor Communication Timing, and Security-Related Timing.

Waveform Synchronization ensures system clocks synchronization to involve both timing and carrier recovery. This allows multiple components working on different aspects of the same data to accurately merge their outputs.

Intercomponent/Interprocessor Communication Timing issues result from buffering, processor multiplexing rate, processor multiplexing overhead, context switching overhead and synchronization of multiple task completion threads.

Security-Related Timing issues involve suppressing covert channels. These covert channels result from timing leaks resulting in the potential of data with different classification levels being on the same bus simultaneously. A Multiple Independent Levels of Security (MILS) architecture suppresses covert timing channels through careful management of processor time allocation in fixed increments.

4.8 Baseband Throughput

Waveforms have a data throughput requirement. A radio's data traffic throughput potential can be determined via the baseband processing speed. All SCA application data and control traffic is eventually converted to and from baseband. The radio's theoretical throughput potential is further tempered by the RF hardware's designed transmit duty cycle. $\text{Baseband Throughput} \approx (\text{Encrypted control traffic overhead} + \text{data traffic})$. The radio's data traffic with respect to Baseband Throughput also includes message header and framing overhead and forward error correction if applicable.

4.9 Multichannel

Multichannel Radios are essentially two or more Single Channel radios with a single Human Machine Interface (HMI). However, unlike two separate radios, a Multichannel Radio has additional complexity due to inter-channel communication, and control. The decomposition of any additional functionality as a result of this channel-to-channel interaction and control early in the design cycle ensures any required hardware support as well as timely incorporation.

4.10 Multiple Waveforms per Channel

Supporting multiple waveforms can present several challenges in addition to just their basic operation. The radio may require the ability to dynamically switch between waveforms ‘on the fly’ after power-up. The SCA does not provide a convenient method of switching waveforms efficiently given some implementations can take relatively considerable amounts of time to release application resources and instantiate new ones as the defacto SCA Method of Operation implies. This capability to switch waveforms is platform specific at present so waveforms may have to implement specific functionality as a result. User perception typically drives how quickly a new waveform must be instantiated and operational which is similar to the reasoning leading to maximum waveform instantiation time as described in section 4.11.

4.11 Maximum Waveform Instantiation Time

The user’s anticipated performance perception typically drives maximum waveform load and instantiation times with handheld and non-console radios receiving the most scrutiny[9]. To this end, waveform instantiation time key ‘players’ include the RTOS/Board Support Package (BSP), CSS, Operating Environment (OE), and waveform. Whether the OE and waveform application are copied and instantiated simultaneously or separately depends on the design.

Application loading and instantiation time depends largely on the static memory footprint and the number of SCA connection and configuration in the Software Assembly Descriptor profile of the application. The application execution is directly impacted by the bus, memory, processor, and CSS (if involved) speed. All collectively determine how quickly the application can be copied from flash into RAM. The waveform instantiation time is directly related to the amount of dynamic connections and configuration specified in the waveform software assembly descriptor profile and the capabilities of the SCA implementation to process the dynamic nature of application deployment.

5. SCA OPTIMIZATION METHODOLOGIES

The flexibility of the SCA allows a number of optimizations to be considered. Linn[9] recommends several including the following:

1. Partially linking multiple components such as DomainManager, ApplicationFactory, Application, DeviceManager, Log and various file components into a single partially linked shared library or executable that is loaded on power-up.

2. Speeding up local file access by extending the core framework interface via IDL inheritance to add getNativeFilename(), with a processor ID passed as a parameter. This allows the application to perform file operations efficiently without the CORBA overhead.
3. Minimize use of device capacities by components that are always deployed on the same device by using of a non-external allocation property which does not involve performing an allocateCapacity() operation on the device. This speeds up the application’s launch.

5.1 Optimize XML Parsing

XML parsing performance will have a relatively large impact on how quickly the core framework is loaded as it has to parse the many XML files required for the core framework and Waveform Application loading. Rockwell Collins Inc. (RCI) tests reveal this to be as much as 10% of the total boot-up time. There are several ways that this parsing time can be addressed to minimize impact:

Test parsers against the implemented files to benchmark which will accomplish the most with the least overhead in both memory footprint and parsing time.

Require XML files to be validated prior to installation with in an SCA system. Including a validating XML parser in an SCA implementation significantly increases the processor resources and memory utilization. Invalid XML can cause significant parsing delays. For example, tests of a representative XML file with invalid syntax required 400ms to parse while its equivalent clean example took 50ms.

This optimization improves the system startup and waveform initialization metrics.

5.2 Offline XML Parsing

Consider making the XML parsing a part of the software installation process[9] since the SCA does not specify when it must take place. This method results in a one-time increase in boot time that could be done at software installation. This parsed information is then stored in a non-XML file for use in future boots effectively eliminating this parsing activity.

Gonzalez, Portelinha, and Reed[10] propose a two step offline domain profile XML installation method which consists of translating XML files into a simplified text format which only incorporates the most important information related to framework functionality.

This optimization improves the system startup and waveform initialization metrics.

5.3 CORBA and Interprocess Communications (IPC)

Lind and Littke [11] demonstrated that even for Military Satellite Communication waveforms the CORBA ORB is

not the major factor in waveform performance; however, CORBA does offer ample opportunity for optimization and therefore deserves a closer look.

CORBA takes on many shapes. Real-Time (RT)-CORBA has been designed with embedded systems needs in-mind. Still, several CORBA Optimizing techniques can provide real benefits to your application. TCP/IP Internet Inter-Orb Protocol (IIOP) transport has too much overhead for many embedded applications. A faster transport such as UDP, Multicast or Shared-Memory in some cases can yield an order of magnitude[2] improvement.

Also consider augmenting CORBA with Non-CORBA IPC message passing for additional performance. SCA Implementers must evaluate the CORBA interfaces which carry most of the baseband data where the benefits of CORBA data marshalling are outweighed by the need for performance. Those interfaces, which are known for passing sequences of octets or characters, are examples where a Non-CORBA IPC Message passing provides significant performance improvement over the CORBA interface.

5.4 Non-CORBA IPC

IPC is used to move information between processes or threads running on an OS. Linn[9] describes IPC between threads within the same GPP and address space as being as much as 2000x faster than identical IPC between separate address spaces (i.e. processes). IPC within the same address space leads to more portability, additional deployment options and enhanced load balancing. Additionally, collocating components within the same address space provides the opportunity to use non-CORBA methods to further enhance throughput.

Common same-address space IPC mechanisms include pipes, files, shared memory, and message queues. These non-CORBA IPC methods leave data formatting including Endianness (e.g. High Byte, Low Byte), Data/Packet Structures as well as alignment and padding to the implementation. Likewise, two way messaging such as return values, flow control, and data synchronization is also left to the implementation.

SCA 2.2.2 Application Environment Profile[1] supports POSIX message passing (`_POSIX_MESSAGE_PASSING`). Given that SCA POSIX[12] is based on IEEE Std 1003.13-2003, it also implies support of POSIX Message queues and, thereby, `mqueue`. This method provides more CORBA like behavior without the overhead providing an order of magnitude higher throughput over CORBA based on RCI tests. POSIX messaging enforces file permissions via the OS; moreover, its syntax requires a name similar to a file.

POSIX message passing provides the SCA developer with the ability to open, read, or write SCA resources (e.g. modem device) as a file. Additionally, POSIX IPC message

passing can transfer raw bytes while supporting basic flow control and priorities. This method could work well for IPC within co-located waveform components on the same GPP and address space and be could be used in addition to CORBA on devices (e.g. Modem Hardware Abstraction Layer - MHAL) to allow lower latency as well as improved throughput. Its drawbacks include not supporting interprocessor communication and not being a good medium for one to many, or many to many messages.

5.5 Concurrency and Threading Priorities

SDR applications use multi-threading to support diverse tasks such as reading a file, processing an algorithm, and monitoring I/O operations which in-turn provide timely operations and services. These threads often support larger functional tasks that must in-turn be prioritized to support critical operations relating to the communications chain involved with sending or receiving and processing the radio's messages. Additionally, to avoid unbounded priority inversion and deadlock, the SDR applications involving these functional tasks often require some form of pre-emptive multi-threading.

These larger functional tasks along with any supporting tasks must preserve their priorities across components and even across processors to provide the most optimum message processing and predictability. Processor loading and task priorities[11] with respect to these tasks can be the primary timing concerns when working with CORBA messaging.

To ensure that messages are given proper priority from creation until their task is completed, the Waveform, Platform, and CSS should ideally use similar priorities. RT-CORBA's support of thread-pools provides a method of achieving this. Pyarali, Spivak, Cytron, and Schmidt[13] describe a leader-follower pattern for thread priority pools that can be implemented in RT-CORBA.

Thread pooling allows threads to be reused without being invoked again, thereby, reducing the overhead of threading. Their empirical benchmarks showed this pattern to outperform its rival Half-Sync/Half-Async pattern in practice by as much as 2800% with the improvement reducing to ~8% as the number of threads increased and amount of work per request increased.

Thread banding allows the ORB to bound and minimize priority inversion by setting up persistent dedicated connections to a given pool of thread priorities with one per address space.

5.6 Messaging

Waveform implementers can dramatically influence SCA performance with a thorough understanding of the key metrics and optimizations. The messaging strategy should

include messaging needs coupled with appropriate interface and transport selection for interelement (or interprocessor) communication and consider the traditional bus speeds of the interfaces between computation elements (e.g. Ethernet, PCI) as well as any overhead associated with communication transport over the bus (e.g. IP header).

To maximize data throughput use low overhead transports such as Non-CORBA IPC for small messages and messages not requiring data marshalling and tailor message traffic. For instance, MHAL data by nature is already marshaled as a self contained on-the-wire format; therefore, CORBA messaging is not explicitly required for these messages.

Additionally, the size of the data message significantly impacts CORBA throughput more than the number of messages due to CORBA overhead. Thereby, taking advantage of larger CORBA message sizes even with other optimizations will be instrumental in optimizing throughput.

6. SUMMARY

SDR describes a distributed system in which throughput and overall performance are the result of many hardware and software components and interfaces collectively functioning as a radio.

Requirements analysis and functional decomposition provides a process to identify and examine key metrics that are captured and incorporated in the overall design. These key metrics serve to enhance comprehension of the user to antenna data path in addition to waveform control.

Applicable SCA optimization methodologies when coupled with an understanding of system throughput via these key metrics team up to optimize overall radio and application performance.

7. REFERENCES

- [1] Joint Tactical Radio System (JTRS) Joint Program Executive Office (JPEO), Software Communications Architecture(SCA) version 2.2.2, <http://jtrs.spawar.navy.mil/sca/>, 2006
- [2] S. Bernier, C. Bélisle, Taking the SCA to New Frontiers, SDRF'06 Technical Conference, 2006.
- [3] OMG, "Software Defined Radio Use Cases for PIM and PSM for SWRADIO Component Submission, <http://www.omg.org/docs/swradio/03-05-02.pdf>, May 2003.
- [4] Department of Defense, Wireless Communications Waveform Development and Management DoDI 4630.09, www.dtic.mil/whs/directives/corres/pdf/463009p.pdf, November 3rd, 2008
- [5] Joint Staff, Information Security Guidelines for the Deployment of Deployable Switched Systems CJCSI 6511.01, http://www.dtic.mil/doctrine/jel/cjcsd/cjcsi/6511_01.pdf, Feb 2001
- [6] R. Sites, "It's the Memory, Stupid!" Microprocessor Report, vol. 10, no. 10, Aug. 1996
- [7] B. Jacob, A Case For Studying DRAM Issues at The System Level, IEEE Computer Society, Volume 23, Issue 4, July-Aug. 2003
- [8] J. Smith, T. Demirbilek, M. Bicer, "Homogeneous Middleware for Advanced Interoperable Communications", Government Microcircuit Applications & Critical Technology Conference (GOMAC) 2004, March 2004.
- [9] C. A. Linn, Designing JTRS Core Frameworks for Battery-Powered Platforms: 10 Techniques for Success, SDR'02 Technical Conference, 2002
- [10] C. R. Aguayo Gonzalez, F. Portelinho, J. Reed, Design and Implementation of an SCA Core Framework for a DSP Platform, SDRF'06 Technical Conference, 2006
- [11] G. Lind, C. Littke, Software Communication Architecture (SCA) For Above 2 GHZ SATCOM, SDR'04 Technical Conference, 2004
- [12] Open Group, IEEE Std 1003.1-2001 Standard for Information Technology- Portable Operating System Interface (POSIX), IEEE Computer Society, 2004
- [13] I. Pyrali, M. Spivak, R. Cytron, D. C. Schmidt, Evaluating and Optimizing Thread Pool Strategies for Real-Time CORBA, <http://www.cse.wustl.edu/~schmidt/PDF/OM-01.pdf>, 2001