DESIGN ABSTRACTION TOOL FOR ON-DEMAND, AUTOMATED SYNTHESIS AND IMPLEMENTATION OF SOFTWARE-DEFINED RADIOS

Doug Jaeger (Northrop Grumman IT/TASC, Chantilly, VA; douglas.jaeger@ngc.com);
Patrick Ring (Northrop Grumman IT/TASC, Chantilly, VA; patrick.ring@ngc.com);
Matt Vondal (Northrop Grumman IT/TASC, Chantilly, VA; matthew.vondal@ngc.com);
fred harris (San Diego State University, San Diego, CA; fred.harris@sdsu.edu)

ABSTRACT

The term Software-Defined Radio (SDR) refers to a class of radio devices that provide a flexible, generic hardware platform where the modulated waveforms, data format, and transport protocol are defined by configurable means, such as software and programmable logic. This configurability, if utilized, enables a radio to be modified quickly in the field to meet changing requirements or varying environmental conditions. The process of designing a specific implementation for such flexible radios, however, requires a large investment of time and effort by skilled design engineers who posses in-depth knowledge of both the signal processing and the development tools required to implement and optimize the software or logic for a given hardware platform. In this paper the authors discuss their efforts to create a tool that provides a new level of radio design abstraction – intended to provide an integrated toolset for the rapid development of new designs by an operator without an in-depth knowledge of communication theory. By raising the level of radio design abstraction and automating the process flow, this new tool enables designers to generate implementations for software-defined radios which can be quickly and easily adapted to changing signal environments and the requirements of specific systems and networks.

1. INTRODUCTION

Designing a radio requires expertise in many related disciplines. These include radio architecture, signal flow, modulation and demodulation, synchronization, digital signal processing, filter design, finite arithmetic effects, implementation budgets, platform and tool constraints, protocols and others. To assure wide acceptance and a high measure of success, the software-defined radio concept requires a formidable list of attributes. A partial list includes (i) easily upgradeable, (ii) quickly reconfigurable, (iii) operable by unskilled users, (iv) presentable via a simple, transparent human interface, and (v) capable of a high level of specification abstraction.

The operator must be able to implement system changes with minimal or no reliance on external instruments and tools as might be available in a laboratory environment. Field changes must be rapid, easy to implement, and manageable with minimal guidance by unskilled operators. The need for use by unskilled personnel means the system must respond intelligently to low level of detail requests. The system must accept low level input detail and must respond with a high level configuration consistent with interpreted tasks.

To address these issues, the authors are developing an application that will reduce the amount of time, effort, and knowledge necessary to create SDR configuration files. This application, named Hotrod, provides an integrated tool environment that simplifies and automates the process of designing a radio system. Using the efficient design process the operator can access the benefits provided by SDR and quickly reconfigure the hardware to meet the communication task within the constraints presented by the immediate environment and currently available communication systems and networks.

2. DESIGN ABSTRACTION

A necessary first step for simplifying the design process is to raise the level of abstraction at which the design process takes place. Most communication problems are stated in high level concepts that are concerned with the objectives and the environment, rather than the specifics of the implementation necessary to achieve the goals - the "what" and "where" rather than the "how." For example, the problem may be stated something like "Get this data from the aircraft to the ground," not "Encrypt this data, encode it, then send it through a QPSK modulator with an output frequency of 2.4 GHz and a link margin of 6dB." Working at a higher level of abstraction reduces the complexity of the problem by allowing the operator to describe it in more familiar terms and leaving the more obscure details of the implementation to tools specifically designed to handle them. As a result, the amount of knowledge necessary to produce a specification for the desired radio is reduced and the design process is simplified.

Hotrod supports this higher level of abstraction by allowing the designer to specify the problem using these higher level concepts. The operator can supply information such as the amount of data, the protocol or available equipment, maximum connection time, available spectrum, or other easily measurable and familiar quantities, and allow Hotrod to fill in the details.

To support this higher level of abstraction, the core of Hotrod is a knowledge base that encodes the processes and technology necessary to build a radio. The knowledge base consists of an Expert System and a Function Database.

2.1 Expert System

The Expert System functions much as the design engineer who supplies knowledge and experience to the radio design process. It handles the lower level details of the radio design process that are not provided in the design specification. The knowledge-based system consists of a list of design rules that encodes the functions, relationships, dependencies, and procedures necessary to build a radio. These rules include the experience of experts, heuristics, and best practices that would ordinarily be used by a skilled engineer in designing the radio. The rules are executed by the Expert System step by step, as information is available, just as a design engineer would approach the process. The execution of these rules produces a model of the radio in memory that meets the specified requirements.

The Expert System provides specific processing advantages over other design programs. The underlying software for the heuristic algorithm is designed for solving problems a small piece at a time, performing specific processing tasks for a specific set of input values. This software consists of a rule-based engine rather than a procedurally based program. Unlike a procedural program, which executes statements in a linear fashion, the Expert System operates on each rule as the information on which that rule depends becomes available. As information is added or updated, the rules can be executed repeatedly. As a result, the Hotrod excels at providing the guidance and structure necessary to determine a best fit solution to a problem that is under-specified or poorly defined. As new information is added or old information is changed, derived values are automatically recomputed. This allows problems to be solved in an iterative, but non-linear manner. This is useful in situations where small adjustments can propagate through a solution in many directions. This also allows several different implementations to be calculated and compared.

The knowledge-based system provides all of the usual capabilities that are expected of design programs. It will check the input data and requirements for consistency, flagging any conflicts. Moreover, the algorithm provides estimates, average values, or commonly used values for missing data fields. It predicts the level of performance of the radio in the field based on the provided design criteria, or conversely, or even build a radio to meet a specified minimum performance level.

Since each SDR hardware platform presents a different set of interfaces to the software, the Expert System also encodes information about how to target the design to the specific hardware. This includes the hardware specific interfaces, command and control, and any necessary processing functions.

The Expert System also maintains an archive of the work that it has completed, so it can be reused in the future. Once the design processing is complete and the SDR configuration file is created, the Expert System places it in a database, along with the criteria used to produce it. When a new request to build a radio is received, it checks this database for a design that was previously implemented. If there is a match, then this previous design is reused, saving the time and processing necessary to create a new configuration file, and using a design that may have been verified by successful operation in the field.

As the Expert System processes the provided criteria and designs the radio system, it uses blocks from the Function Database to provide the necessary capabilities.

2.2 Function Database

The Function Database is a library of functional blocks that can be used to build a radio. These include all constituent functions such as encryption, differential encoding, error correction, modulation, synchronization, equalization, and others. Each block has standardized I/O interfaces, and is parameterized so that it can be used in various configurations. These blocks are developed and verified independently outside of Hotrod. The verification occurs over a range of parameter values in order to ensure that the system in which they are used will operate correctly with a high degree of confidence. Once verification is complete, the blocks are placed in a library so they can be used by Hotrod. New blocks can be added as additional functionality is required and the block is developed.

The current target SDR platform is custom hardware which uses a Xilinx FPGA for the reprogrammable processing element. As a result, the Function Database consists of blocks that can be targeted to an FPGA. As the need arises, Hotrod can be expanded to include other tools and SDR hardware platforms.

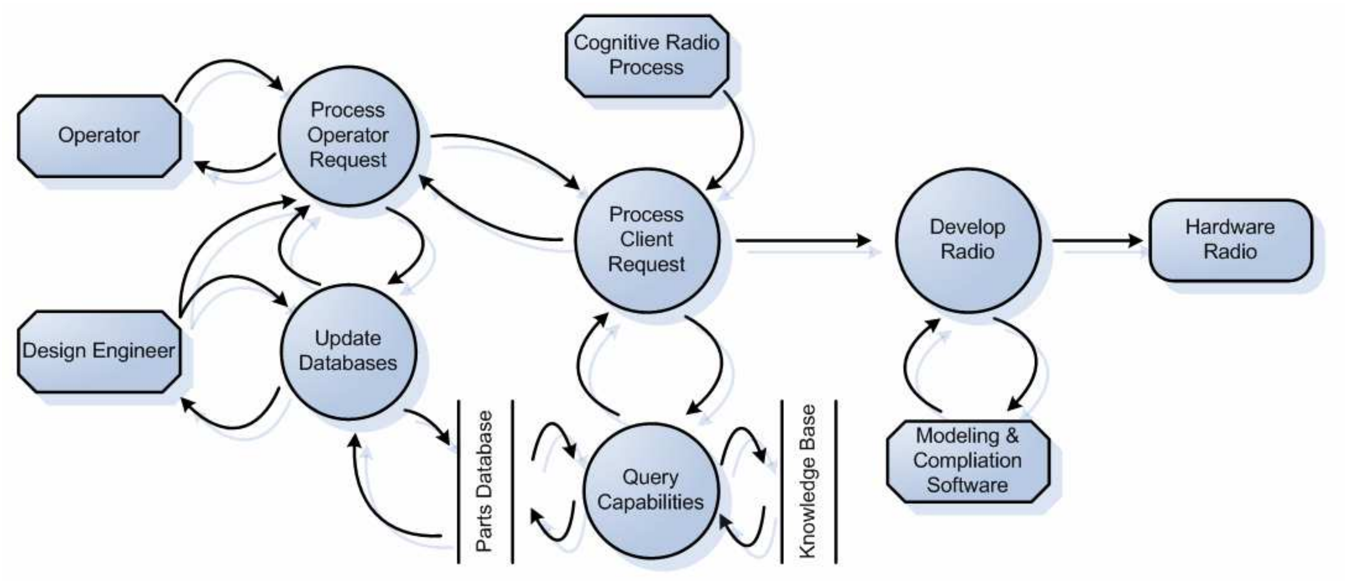


Figure 1. Hotrod an On-Demand, Automated, Software Defined Radio Design Process

3. AUTOMATED IMPLEMENTATION

Another step for simplifying the radio design process is to reduce the knowledge that is necessary to skillfully operate the various tools that must be used to perform calculations and implement the design. Typically, an analysis tool, such as Matlab, is used to calculate design parameters such as filter coefficients and to perform other complex calculations during the design process. Once the design process is complete, the design, which exists as a list of interconnections between functional blocks, must be converted to a form that can be executed on the hardware platform. For FPGA-based hardware platforms, the design must be converted to a netlist (a list of interconnections of logic gates) and then converted to an FPGA-specific configuration file. Hotrod currently targets hardware that uses Xilinx FPGAs, and the set of software tools that produce configuration files for these FPGAs are complex and take considerable effort to operate effectively. To minimize this burden on the operator, Hotrod automates the operation of these tools and integrates them into one development environment.

3.1 Matlab

During the design process, Hotrod uses Matlab to perform complex calculations for which it is designed and well suited. Chief among these are determining filter coefficients for the various filters in the signal processing path. Hotrod calls the functions in the Matlab API to execute scripts and retrieve the results.

For the Xilinx FPGA-based SDR hardware, Matlab is used as the first step in the implementation process. The function blocks are designed using System Generator from Xilinx inside of Simulink. The radio design, as produced by the Expert System, is saved as a Simulink model file and loaded into Simulink. If desired, the operator can utilize the graphical Simulink environment to view and explore the system design, or to provide stimulus and simulate it. The design can also be modified, if necessary, although changes at this stage can no longer be checked by the Expert System. Once the operator is satisfied with the design, Hotrod invokes the System Generator executable to create an FPGA netlist of the design as it appears in the Simulink model.

3.2 Xilinx Implementation Tools

Once the netlist is complete, Hotrod places it in a project for final implementation by the Xilinx tool set. This project provides the I/O interfaces expected by the radio, mapping them to the hardware-specific interfaces on the target device. Hotrod provides option files and invokes the Xilinx implementation tools to produce the final device configuration file. It monitors progress and checks for error conditions and exceptions.

After compilation, Hotrod has a valid device configuration file that can be loaded on the target hardware. The configuration file, the design criteria, and the model are saved to a project database and the operator is notified that the configuration file is ready for use. In hardware platforms which support it, Hotrod can send the configuration to the hardware to automatically configure it.

At this point in the design process, the configuration file is complete and can be loaded onto the target hardware.

However, the complete radio has not been tested to ensure that it works correctly and that it meets the design criteria. As in any automatically generated configuration, validating the system is a key step in ensuring that the system will function as intended. There are several approaches to verification that can be used on an automatically designed system.

4. VERIFICATION

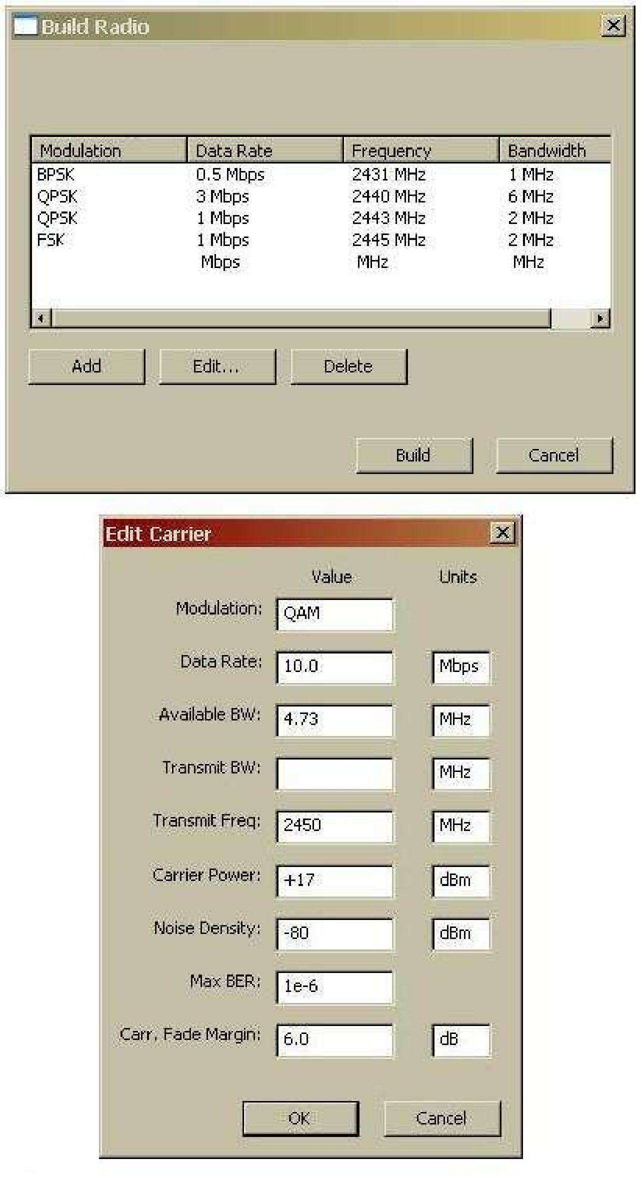
There are four primary potential sources of error that arise during automatic synthesis of radio configurations: (i) the modular components are flawed in a manner not previously discovered; (ii) the desired connections and routing at the system design level are flawed; (iii) the hardware configuration and synthesis tools inaccurately report successful completion; (iv) the real world implementation adds impairments not accounted for in the system design. Each of these sources can have serious or fatal effects on the performance of the generated system, and should be addressed in any thorough verification scheme.

All four sources of error can be mitigated to an extent through dual-graph loopback testing. For instance, a digital loopback after the modulator block back into the demodulator verifies that the components are functional in the current configuration, that the system routing is correct, and that the design synthesized correctly. Digital loopback costs little in resource overhead because a full modem implementation is usually required for data link configurations.

Analog loopback testing is only practical to a certain extent. Performing a loopback test at the I/Q baseband, sampled IF, or even RF level can verify that noise and distortion effects at the digital interface and in the radio front end are not limiting the performance. However, analog loopback does require additional hardware investment.

Straightforward loopback testing does not localize the radio errors to either the modulator or demodulator side, as both maybe unverified, and certainly cannot test the actual channel effects of the environment. Future work may look at modes in the Expert System that deal with automatic recovery from flawed radio synthesis. System choices could include synthesizing a known good configuration or iterating the design with a loopback test further back in the system processing chain.

There are several data-aided and non-data-aided statistical metrics that can always be embedded at several levels within a synthesized modem like an automated design-for-test. These may help locate the error source, but still not require many resources. Further, by inserting digital channel models into the loopback test configuration during verification, the real world performance might be more accurately predicted. Finally, testing to an arbitrary percentage confidence level can require a long period of time, especially for relatively low data rates. The capability to accelerate the verification time by running multiple parallel models, perhaps at an increased clock rate, could serve to approximate real performance in a fraction of the required time, provided enough resources are available.



Modulation	Data Rate	Frequency	Bandwidth
BPSK	0.5 Mbps	2431 MHz	1 MHz
QPSK	3 Mbps	2440 MHz	6 MHz
QPSK	1 Mbps	2443 MHz	2 MHz
FSK	1 Mbps	2445 MHz	2 MHz
	Mbps	MHz	MHz

	Value	Units
Modulation:	QAM	
Data Rate:	10.0	Mbps
Available BW:	4.73	MHz
Transmit BW:		MHz
Transmit Freq:	2450	MHz
Carrier Power:	+17	dBm
Noise Density:	-80	dBm
Max BER:	1e-6	
Carr. Fade Margin:	6.0	dB

Figure 2. Semi-Expert Mode Enables Directed Design

5. ADDITIONAL ADVANTAGES

Hotrod currently supports two interfaces, a graphical user interface (GUI) and an Ethernet interface. The primary interface for most operators is the GUI. It gives the operator access to all of the capabilities of the software, and can be customized to meet the needs of the situation and the operator by varying the amount of information and the level of abstraction that is presented. The Ethernet interface provides these same capabilities to third-party software that may want to take advantage of the design guidance or tool automation that Hotrod provides.

Although Hotrod is designed to simplify the design process for use by less knowledgeable operators, it is not constrained to provide this level of assistance. It also supports more experienced users by allowing them to enter as much specific information as they know. For these users, it will offer reduced levels of design guidance or possibly none at all. This allows these users to design radios with various combinations of parameters and function blocks that Hotrod may not have been programmed to construct, thereby permitting design trade-offs and comparisons to be performed, while still automating the computational tasks.

The tool provides great flexibility in the design process supporting simple and complex link configurations as necessary to meet design objectives or guidance given by the operator when Hotrod is used in a semi-expert mode.

To support single and multi-carrier configurations within the design tool, data stream multiplexers and demultiplexers have been developed. Incoming data is broken into packets that are diverted to individual sub-carriers. Hotrod automatically instantiates multiplexers as needed to reassemble packet streams in the corresponding receiver sections. Each sub-carrier is independently configured for specific bandwidths, amplitudes, modulation, and internal structure.

In similar fashion, Error Correction, Interleaving, Data Framing, Protocol Support, and other functional components are used to construct link architectures which meet or best fit channel capacities and design requirements. The authors are exploring expanded capability through BER tracking, SNR estimators, and other blocks to enable intelligent link layer management including dynamic carrier re-assignment.

This approach to radio synthesis provides several benefits. It speeds up the design process. It supports the creation of modems that use multiple carriers as easily as those that use only one. It will also permit different waveforms on each carrier to effectively use the available spectrum.

6. SUMMARY

Software-defined radios provide a flexible and generic radio platform that can be reconfigured quickly in the field to meet the changing requirements dictated by variations in the signal environment and the capabilities of other accessible communication systems. Creating new configuration files for these radios can be a slow and demanding process because of the amount of knowledge and expertise needed to design the new radio and operate the tools necessary to implement it. As a result, the advantages provided by reprogrammable hardware are not fully realized. This paper presented a software package that provides an integrated design tool set that simplifies and expedites the radio design process. This software uses Expert System technology and process automation to raise the level of abstraction at which the desired radio is specified, designed, and implemented. As a result, SDR configuration files can be created more quickly and easily by operators who do not have the knowledge or experience necessary to design a radio using the current design processes. In addition, these capabilities allow experienced users to create and compare variations of a design in ways that were not previously possible. By simplifying the design process and designing at a higher level of abstraction, the capabilities and flexibility of software-defined radios can be more effectively exploited.