

Automating the Implementation of Software Defined Radios at Northrop Grumman

By Gus DiPierro, Doug Jaeger, Patrick Ring, and Matt Vondal, Northrop Grumman

AS COMMUNICATIONS ENGINEERS, OUR MOST CONSTRAINED RESOURCE IS time. We might have as little as a month to implement a new wireless system or multi-megabit datalink and deploy it in the field. Once a radio is operational, we often need to rapidly reconfigure it in response to evolving standards, new requirements, or changing environmental conditions. A Software Defined Radio (SDR) provides the flexibility to address these challenges, but that adaptability comes at a cost.

The design of a flexible or multi-mission SDR requires a great deal of development time, resources, and engineering expertise. Broadly speaking, SDR creation taps multiple domains involving signal processing, RF design, mixed-signal integration, FPGA development, and software engineering. Moreover, communications engineers with the necessary skills are in short supply.

At Northrop Grumman, we set out to address these challenges by building a system called Hotrod that automates the implementation of SDR devices (Figure 1). The core of this integrated framework builds on the modeling and simulation capabilities of MATLAB®, Simulink, and Xilinx® System Generator. Our main objective for this effort was to enable inexperienced users without an in-depth knowledge of the nuances of communications theory, such as junior engineers

and technicians, to specify and construct an operational radio design at a high level of abstraction using parameters such as data rate, modulation scheme, and available bandwidth. Beyond this, Hotrod has shown it can save considerable development time by generating an initial design architecture that experienced engineers can later modify. Utilizing MATLAB and Simulink in the Hotrod framework, we automated the complex SDR design process and demonstrate virtually unattended radio synthesis, reducing design time for our reconfigurable platforms by an order of magnitude.

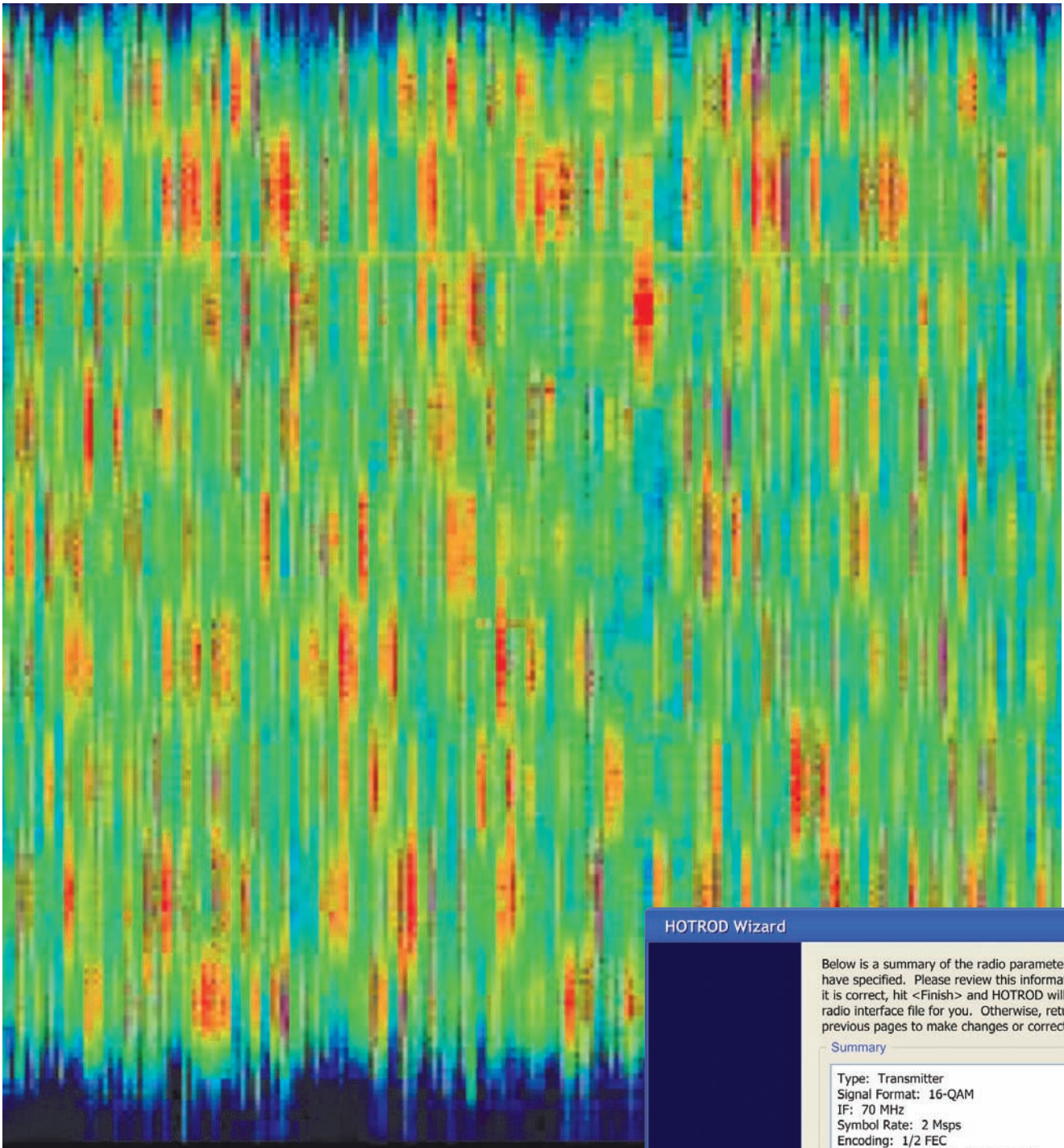
We knew that Hotrod had to accomplish much more than simply selecting predefined designs based on given specifications; it had to make intelligent adaptations—for example, by using multiple carriers to take advantage of nonadjacent spectral bands.

Design Goals

Our target was not to create the same absolutely optimized solution that a team of experienced engineers might if given the same input parameters or design specifications. Rather, it was to incorporate the knowledge and thought processes of engineers with years of radio design experience into an automated, rules-based expert system. We saw that we could swiftly obtain a conservative implementation by pairing a library of proven design architectures with a framework for reconfiguring each functional module to meet the user's requirements.

Hotrod's Library of Functional Blocks

The first iteration of Hotrod exclusively targeted FPGA-based solutions. To that end we developed a Simulink block library that exactly modeled the hardware characteristics



A waterfall plot shows that radio channel occupancy varies over time. Each mission therefore encounters unique wireless environments even in the same physical location. Unlike a conventional radio, a software defined radio (SDR) device can be reconfigured dynamically to accommodate changing requirements or environmental conditions.

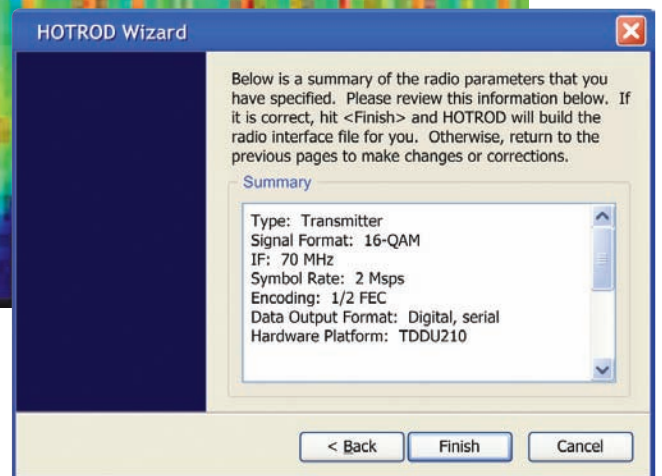


FIGURE 1. A Hotrod Wizard summary page reports the key characteristics of the radio design capture process.

for functions such as encryption, differential encoding, error correction, modulation, synchronization, and equalization for FPGA structures. Implemented primarily using the Xilinx Blockset, each functional block was optimized for implementation on a Xilinx FPGA and formatted with a standardized I/O interface, as well as with parameterized inputs for use in multiple configurations. This step was not as onerous as it might appear because we had a large library of previous digital designs to draw from. Much like software engineers, most of our digital designers maintain a cookbook of useful structures in generic language from which they draw common functions as needed. However, we have found the need to establish a process to maintain quality control. Before adding a functional block to the library, our designers verify its correct synthesis both through hardware definition language (HDL) simulations and through building hardware prototypes and testing over a range of representative parameter values.

How Hotrod Automatically Builds a Simulink Model

An expert communications engineer with hardware design knowledge could use Simulink with System Generator blocks to create a Xilinx-optimized SDR either block-by-block or one subsystem at a time. For less experienced engineers and technicians, SDR design is still a significant challenge, even when they can draw on a functional block library. Hotrod's rules-based engine was created to select blocks from the library, determine their placement in the system, and set each parameter to meet the user's specifications.

Hotrod relies extensively on MATLAB and Simulink to automate this complex process. Many functions within the library have a number of alternate structures, defined as parameters for the block itself. Hotrod configures these parameters to meet the radio's specifications. For example, it will configure a modulator block to use a BPSK, QPSK, or 16-QAM modulation type, and call the

MATLAB signal processing methods to calculate filter coefficients.

After gathering design specifications, an engineer will typically create a new Simulink model, add blocks and then wire the blocks together. Hotrod uses the MATLAB Application Programming Interface (API) to perform these same steps faster to automatically construct an operational Simulink model. The flexibility of the API and scripting capabilities made it easy to create an interface between Simulink and CLIPS, a free public-domain expert system development environment used to create the rules-based engine. Without user intervention, Hotrod can generate and execute a MATLAB script containing all the commands necessary to build the Simulink model.

This section of automatically generated script uses the `set_param()` function to adjust key block parameters as needed. It also exploits the autorouting capabilities of Simulink, calling `add_line()` to programmatically add signal lines and buses between components.

```
...
src = ['STC_Library/Modulator IQ'];
src = hruGetBlkPath(src, LibList);
dst = [syspath '/Mod_A'];
add_block(src, dst);
set_param(dst, 'Position',...
           [120 60 150 90]);
set_param(dst, 'MaskValueString',...
           'BPSK 1.0 Mbps');

src = ['xbsIndex_r4/Gateway In'];
src = hruGetBlkPath(src, LibList);
dst = [syspath '/Freq_A'];
add_block(src, dst);
set_param(dst, 'Position',...
           [60 60 90 90]);
set_param(dst, 'arith_type',...
           'Unsigned');
set_param(dst, 'bin_pt', '0');
set_param(dst, 'n_bits', '8');

add_line(syspath, 'Freq_A/1',...
         'Mod_A/1');
...
```

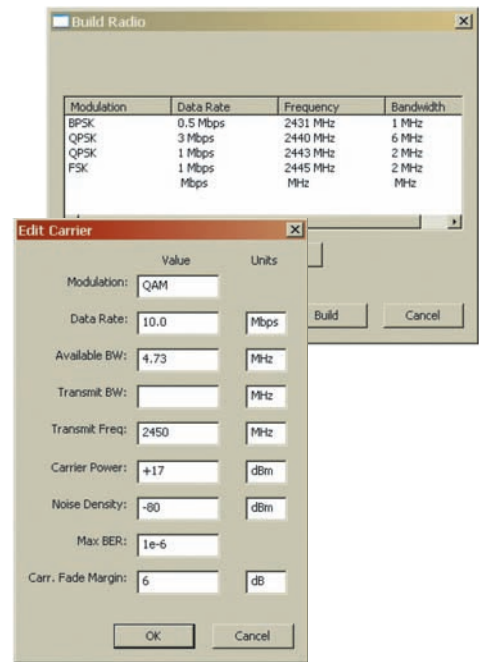


FIGURE 2. Hotrod dialog boxes.

Working with Hotrod

In practice, Hotrod lets users describe a radio in high-level terms and use the language of operators to specify system parameters such as capacity, type, and target. The user does not even need to specify a value for every parameter of the system. For example, a designer might declare a desired data rate, the maximum tolerable bit error rate (BER), and the noise density for a datalink, and leave Hotrod to find the best match for the other parameters.

Figure 2 shows two dialog boxes that a user might encounter while specifying parameters such as modulation, data rate, available bandwidth, and possible frequencies. If Hotrod cannot produce a design that meets the requirements on a given platform, it provides a best estimate of achievable performance. It might report, for instance, that while the user specified 100 megabits per second, the best that it can achieve given other constraints is 73 megabits per second.

While Hotrod provides additional guidance and hides design details for less experienced operators, it allows experts a high degree of control. Of course, the Simulink model created

by Hotrod can simulate the system before it is committed to hardware. Engineers can use Simulink to generate stimulus signals and monitor the system output by using scopes or by plotting simulation results with MATLAB. At this stage in the process, engineers can also modify the design in Simulink, perhaps adding some further patch logic in HDL. They might perform multiple design iterations, using Simulink to make a change, simulate the new design, and then visualize the results.

Implementing the Design on an FPGA

When the user is satisfied with the design, Hotrod invokes Xilinx System Generator to create an FPGA netlist from the model. The user must provide a hardware platform-specific wrapper template so that Hotrod might use Xilinx ISE to produce a viable device configuration file. Our latest reconfigurable platforms, based on the Xilinx Virtex-4 and Virtex-5 devices, have met with considerable success.

This automated process lends itself to some interesting potential applications. We have demonstrated an automated spectral scavenging capability in which a scanning receiver determines the frequency and periodicity of signals in a given band and communicates this information

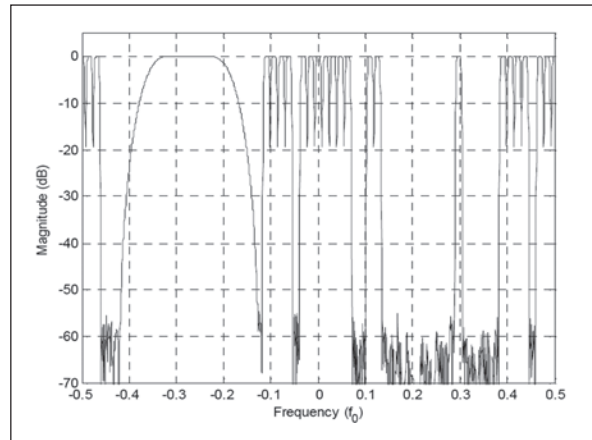


FIGURE 4. Plot showing that synthesized radios can adapt to a congested spectrum.

to a field designer location running Hotrod on a laptop. Given a datalink throughput figure to achieve, the software then designs, synthesizes, and loads a completely new radio design on a platform without user intervention. Clearly, the ability to respond to environment “white space” in both frequency and time without having to permanently deploy a complicated radio system could be a considerable advantage for quick re-action deployments.

Accelerating SDR Development

By automating the design process with Simulink and MATLAB, Hotrod greatly

reduces SDR design time. Depending on the size of the design, an engineer using Hotrod can implement a new radio on an FPGA in less than an hour. Designing a radio to the same specifications by hand would often require at least ten times the amount of effort. For even if the engineer knew exactly what steps were needed, a substantial amount of time would be required to manually select blocks, adjust parameters, connect blocks, and shepherd the intermediate data files through the design synthesis process, not to mention the time inevitably lost to debugging.

The Hotrod designer is capable of producing millions of possible single- and multi-carrier configurations. Moreover, as we continue to capture successful designs in its database Hotrod will be able to reuse them to further accelerate the process.

Our continued success as communications engineers depends on responding swiftly to emerging requirements as they appear. We think that Hotrod will be a critical piece of that capability. ■

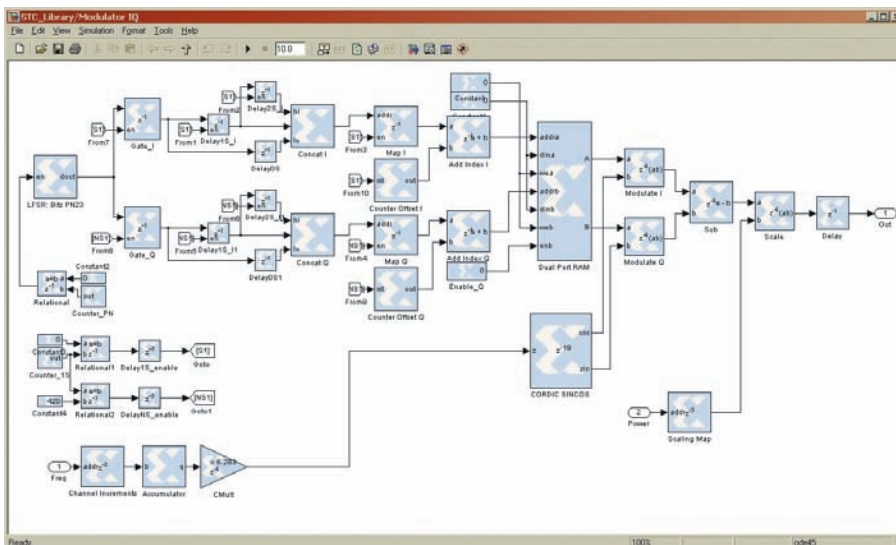


FIGURE 3. A sample modulator built with Xilinx blocks in Simulink, one of many components in the library used by Hotrod.

Resources

NORTHROP GRUMMAN
www.northropgrumman.com
 contact: matthew.vondal@ngc.com

WEBINAR: Model-Based Design and FPGA Implementation with Simulink
www.mathworks.com/nn8/wbnr30807