Whitepaper

Advanced MIMO Waveform Deployment Using GNU Radio

Nutaq® INNOVATION TODAY FOR TOMORROW®

Accelerating Advanced MIMO Wireless Waveform Deployment Using GNU Radio

Written by Martin Turgeon, Nutaq Product Manager

Development and implementation of narrowband and SISO wireless waveforms can be performed easily by feeding RF to today's high end processors, such as the new Quad Core™ i7 family of processors from Intel®.

But when it comes to wideband, multi-user, or MIMO waveforms, processors rapidly struggle to achieve real-time implementation, eventually requiring parallel processor computing. An example of this would be high-performance computing (HPC) systems using a large matrix of general purpose processors (GPPs) to share the processing load.

Including an FPGA between the RF module and the computer drastically reduces the load on the CPU by offloading high-speed and high-parallel computing PHY-related algorithms, as well as reducing the power consumption.

These are two important considerations when deploying a waveform on either a portable device (where increased power consumption reduces battery life, and a lower power CPU might not be able to cope with the high processing demands) or an infrastructure device (where increased power consumption and processing demands result in increased overall cost of the system).

Where software is concerned, waveform developers need to focus on adding value to the system rather than wasting time re-implementing standard algorithms (such as FFTs). On the communications side, customers don't want to spend time on low-value tasks such as programming FPGA interfaces, adjusting FPGA constraints, debugging drivers, and so on.

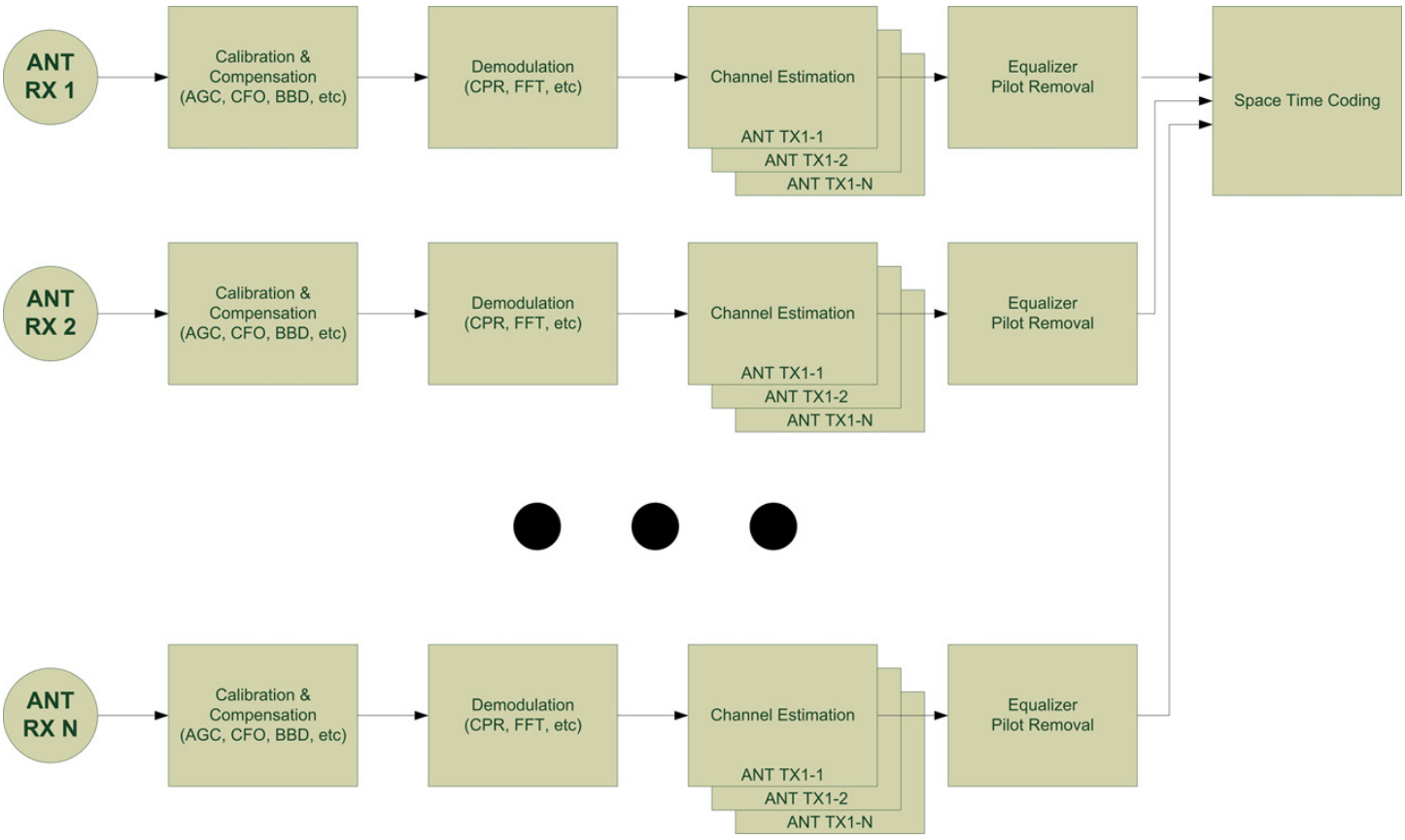
Given these constraints, how can we accelerate development and deployment of an advanced wireless waveform to a mixed PC-FPGA hardware architecture?

Benefits of targeting a mixed PC-FPGA hardware architecture

Using an FPGA between the computer and the RF module enables very efficient MIMO radio development. FPGA processors are good at parallelism and high speed logical operations.

In the diagram above, which shows an example of a MIMO RX PHY processing chain, such as MIMO OFDM application, the MIMO processing chain of each antenna can easily be replicated within an FPGA architecture. Of course, a bigger FPGA is required to replicate all the necessary logic, but the performance of the system and latency would not be impacted (as opposed to an implementation done entirely within the computer). Furthermore, using an FGPA reduces the computer CPU usage, freeing it up to process the upper waveform layer protocols (L2-L3) and more. These upper layer protocols are better suited for the RISC processor architecture found on many modern personal computers and tablets.

Another benefit to migrating PHY-related and parallel computing algorithms to an FPGA is that this approach can ease and validate the transition to real fabric. In real life applications, MIMO radios that sell in lower volumes can take advantage of the newly introduced low-power and low-cost FPGAs (for example, the Artix™-7 and Zynq® processors from Xilinx®) while high-volume radios could make the switch to ASIC PHY external chips.



Benefits of targeting a mixed PC-FPGA hardware architecture *cont'd*

Another important point to consider when targeting a mixed PC-FPGA architecture is the bandwidth and latency link quality between the two processors. Communication interfaces between the FPGA and PC may use various industry standards that may provide advantages or limitations, depending on the waveform application development. Let's compare two of the main interfaces used in the industry:

	PCIe	GigE
Latency	In microseconds	In milliseconds
Data Throughput	More than 5 Gbps	Less than 1 Gbps
Network management	Point-to-point	Easy network integration

Tools to accelerate waveform development

Accelerating the development of radio processing using a mixed PC-FPGA architecture is not trivial. There really isn't any one tool that can leverage the advantages of each processor in this type of architecture, given that these processors require very different programming languages. Instead, what we can use are individual tools that take advantage of each processor's characteristics as they apply to radio waveform development.

When considering tools, we need to keep in mind the following feature requirements for supporting accelerated development:

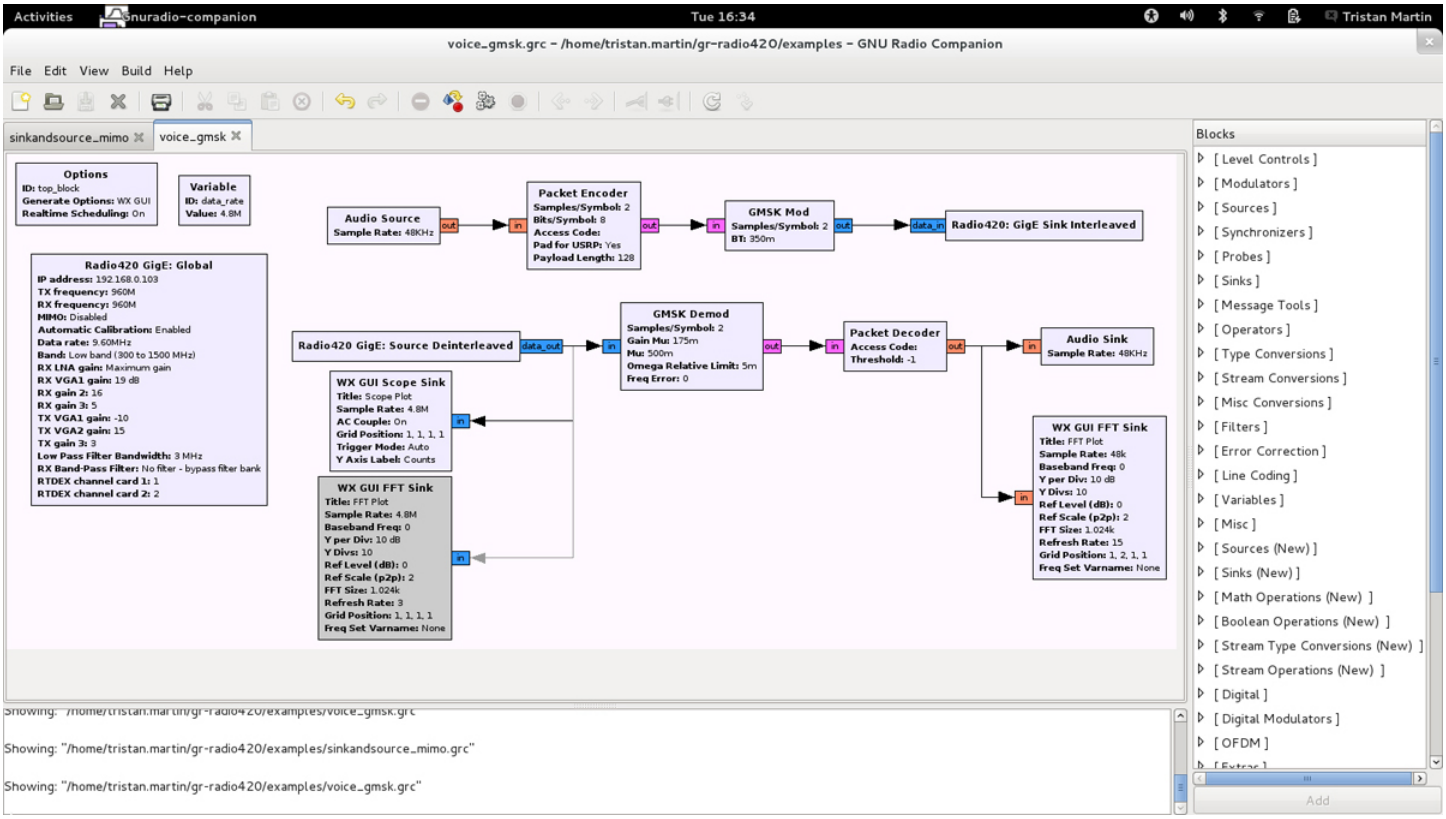
- Reusability of existing resources, for example, IP cores or other code from user communities or open source libraries
- Model-based design or high level system design approach
- Automatic code generation
- Debug/simulation capabilities

Using GNU Radio to target the PC

As a tool for the PC side, GNU Radio is rapidly gaining popularity and is now used almost exclusively by the radio community to target RISC processors. GNU Radio, being open-source, is really attractive because of the benefits of sharing resources from a worldwide community of users (our first requirement). GNU Radio also takes advantage of Linux, a very efficient and high-performance open source operating system. GNU Radio user applications are designed in two different ways:

- Using Python scripts. Python is an object-oriented language that can be combined with C++. It's free and is run using simple scripts in text files (.py files). Python scripts make it simple to interconnect GNU Radio core signal processing blocks and configure their parameters, providing an efficient and easy way to design and map the radio processing chain.

- Using a model-based design approach (our second requirement). The GNU Radio Companion is a graphical interface that represents links between GNU Radio blocks and allows the user to play with these connections, browse the available libraries, and so on. Using the GNU Radio Companion, the radio processing chain can easily be represented and understood by many users. An example is shown here:



Using GNU Radio to target the PC *cont'd*

GNU Radio users are also free to create their own signal processing blocks and associated value-added source files (C or precompiled libraries). A GNU Radio block can also be designed by creating a new block that contains one or many existing blocks and their interactions (our third requirement).

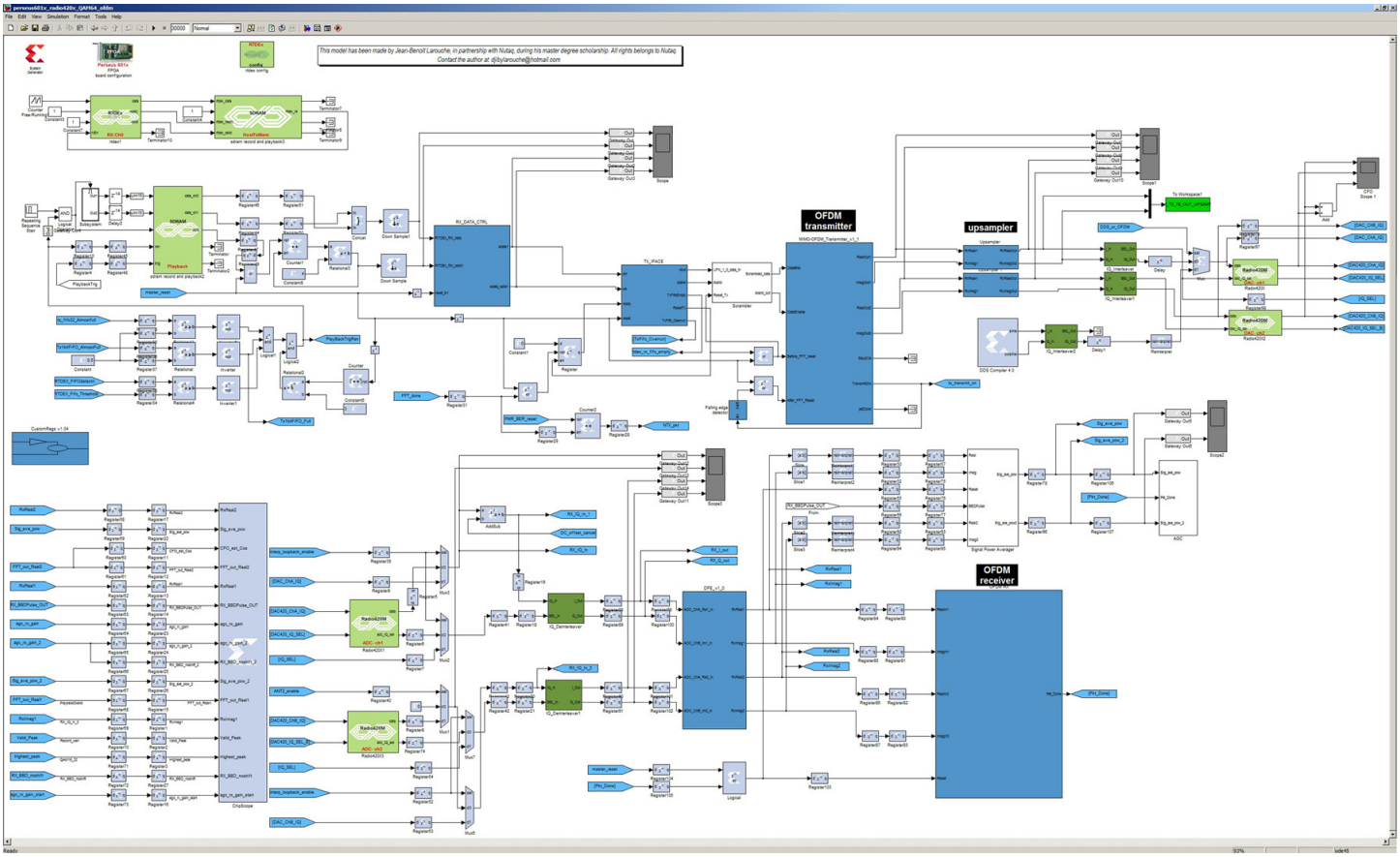
The GNU Radio block library is pretty extensive and already includes almost all necessary primitive functions such as adders, multipliers, FFTs, and so on. GNU Radio also includes a variety of radio examples—pre-built waveforms that are provided by the user community—again accelerating development through the concept of reuse.

As for debugging (our fourth requirement), GNU Radio offers a number of options, such as examining error codes, using the graphical interface sink debuggers, and, for experts, using the GNU debugger, where step-by-step coding can be implemented.

Using System Generator to target the FPGA

As for FPGA programming, System Generator for DSP™ from Xilinx can really accelerate development. Given that C code is much easier to work with than the HDL code required for FPGA programming, many waveform developers may be reluctant to learn how to program FPGAs. In addition, developers want to focus on their value-added algorithm development.

This is where System Generator for DSP really shines: developers can implement very complex design algorithms without the need to know how to program FPGAs. The figure below shows an example of a MIMO OFDM PHY reference FPGA implementation:



Using System Generator to target the FPGA *cont'd*

System Generator for DSP is a plug-in to the Simulink® tools from The MathWorks, Inc. The plug-in installs a complete library of Simulink blocks that, when used, creates an FPGA consisting of a mix of HDL code and pre-compiled, efficient netlists for Xilinx FPGA devices (our second requirement; Simulink is a well-known model-based design tool).

The power of System Generator for DSP is that most of the blocks provide highly efficient code by directly mapping Xilinx IP cores (FFTs, FIFOs, etc.) during the build process (our third requirement).

The Xilinx System Generator for DSP tool also includes an extensive library set containing all necessary primitive functions as well as many advanced pre-built IP cores (DDS, FFTs, FIRs, and so on) necessary for waveform development (our first requirement). It also provides the option to translate basic M-code to VHDL, has plug-and-play integration with ModelSim® deep FPGA simulator from Mentor Graphics®, and it also integrates features from Xilinx ChipScope™(here's our fourth requirement again).

Using System Generator for DSP, developers benefit from simulation interaction with standard Simulink source, sink and processing block sets. The result is that design and implementation of algorithms can be performed prior to any FPGA synthesis, and without waiting hours for the FPGA image to generate (our fourth requirement), thereby accelerating the development cycle.

By interfacing with the extensive Simulink communications library in the simulation phase, developers can take advantage of the available telecom pattern generated test vectors to drive their FPGA PHY model, for example. They can add noise and simulate multipath interference using various channel emulators, so that they can deeply test their implementation with almost real-life situations. Sink scatter plots, FFT scopes, and many data visualization interfaces can also be used to identify errors early in the development process, avoiding trial and error and the long wait for hardware validation.

Nutaq offers a fully integrated MIMO waveform development system

The combined system of GNU Radio for PC plus System Generator for FPGA provides the ideal environment to accelerate radio waveform development. However, we need to keep in mind that to really take advantage of these tools, hardware-specific plug-ins must either be designed by the user or provided by the development board vendor.

For example, within GNU Radio, plug-ins add specific blocks that interface with the appropriate communication drivers to exchange data between the real radio-to-FPGA interface and the PC. Parameter configuration blocks must also be provided to configure radio parameters in real time.

On the FPGA side, plug-ins must include hardware-specific I/O block libraries that map all FPGA interface peripherals to System Generator processing function blocks (GigE interface, PCIe interface, Radio I/Q data I/Os, external RAM, and so on).

Nutaq offers SDR solutions that support GNU Radio, as well as a Model-Based Design Kit for FPGA.

Both of these plug-ins enable rapid building of implementation code and images that can directly target processors, avoiding the need to manually create interfaces between the algorithm-processing generated code and the hardware-vendor provided APIs and VHDL-related FPGA interface reference designs. The end result is that modeling, simulation, compilation and deployment of a radio waveform can all be implemented in the same design environment.

Accelerate waveform deployment by up to 60% with model based design approach

Using the right mixed PC-FPGA radio platform that offers full integration with both GNU Radio and Xilinx System Generator for DSP can radically accelerate advanced MIMO wireless waveform deployment.

These sophisticated model-based design tools and automatic code generation tools do away with the need for a traditional multi-disciplinary team of algorithm engineers, FPGA engineers, and PC engineers. Instead, the radio waveform can be deployed directly by the algorithm developer, avoiding errors that come from interface interaction and code-translation.

To gain some numerical insight on the potential time savings that can be realized by using a model-based design approach, see this article from The MathWorks, Inc., which states that ***"organizations that adopt Model-Based Design realize savings ranging from 20-60%, when compared to traditional methods."***

Nutaq products are constantly being improved; therefore, Nutaq reserves itself the right to modify the information herein at any time and without notice.

Nutaq®

INNOVATION TODAY
FOR TOMORROW®

2150 Cyrille-Duquet, Quebec City (Quebec) G1N 2G3 CANADA
T. 418-914-7484 | 1-855-914-7484 | F. 418-914-9477
info@nutaq.com