Prototyping Next Generation Wireless Systems with Software Defined Radios

Overview

Around the world, wireless consumers' insatiable demand for bandwidth has spurred unprecedented levels of investment from public and private sectors to explore new ways to increase network capacity and meet escalating demand. Software defined radios (SDRs) have emerged as a viable prototyping option for next generation wireless research by enabling researchers to quickly prototype a system, characterize it's performance, and iterate on the design.

Table of Contents

1. Insatiable Demand for Bandwidth

Around the world, wireless consumers' insatiable demand for bandwidth **(Figure 1)** has spurred unprecedented levels of investment from public and private sectors to explore new ways to increase network capacity and meet escalating demand. Industry analysts postulate demand will outpace capacity and it's simply a matter of when. Against this backdrop, wireless researchers continue to put forth new ideas to address capacity challenges. Some topical areas span low-level Physical Layer (PHY) algorithms to upper layer medium access control (MAC) and even cross layer exploration on new heterogeneous network topologies incorporating pico and femto cells, and relays. In all probability, wireless service providers may not rely on one "silver bullet" to alleviate capacity constraints, but rather employ a combination of techniques. Although there is no shortage of new concepts and theories, the time to transition from concept to simulation to prototype to deployment in a real network can take many years. In particular, transitioning from concept/simulation, which is largely a software exercise, to a working prototype with real signals and waveforms requires extensive investments in time and money, and has been an impediment to the adoption of new techniques to alleviate the wireless bandwidth crunch.

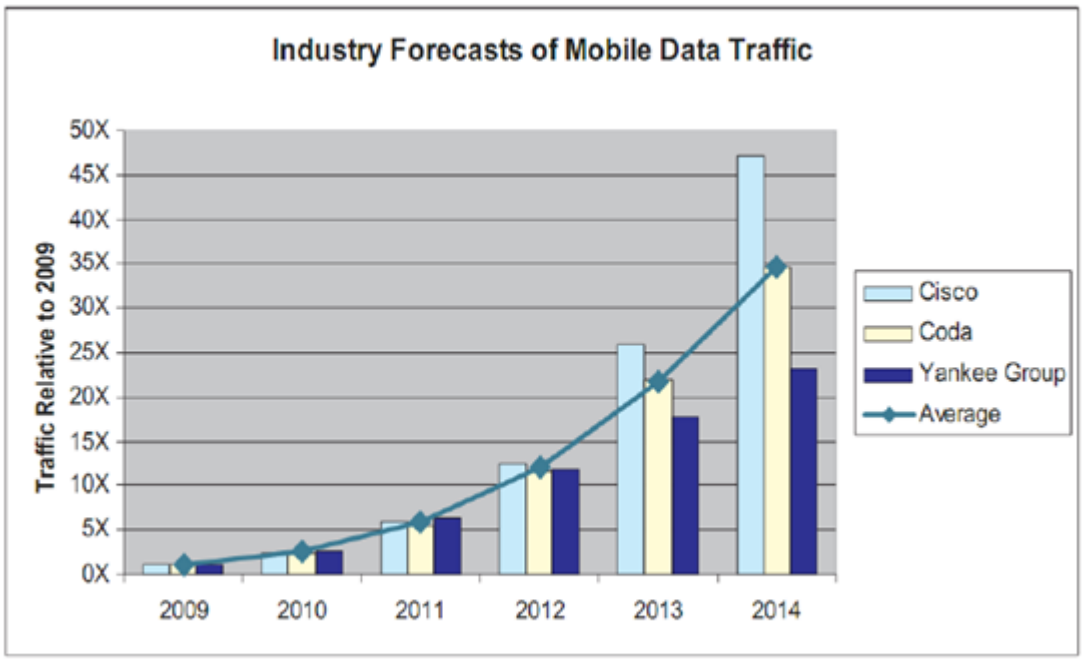

Figure 1: from Mobile Broadband: The Benefits of Additional Spectrum (FCC Report 10/2010)

Figure 1 Reference: *FCC Report 10/2010*

2. Rapid Prototyping with Software Defined Radio

Software defined radios (SDRs) have emerged as a viable prototyping option for next generation wireless research by enabling researchers to quickly prototype a system, characterize it's performance, and iterate on the design. Today, researchers mostly rely on software simulation to test their theories employing simplified channel models (ie AWGN), that loosely emulate real-world conditions Incorporating fading models, such as Rayleigh or Ricean, may improve the simulation for mobile scenarios; however these models fall short in accurately portraying all network conditions and offer little insight into deployment feasibility. Furthermore, more sophisticated network topologies such as Multi-user MIMO (MU-MIMO) and Coordinated Multipoint (CoMP) are simply very difficult to model accurately. SDRs can accelerate prototyping as researchers utilize the inherent software re-configurability and system flexibility. Researchers using SDRs can develop, deploy, test and iterate on new signal processing algorithms and/or system software quicker and easier than conventional approaches.

SDRs functionally mirror real mobile devices and/or cellular base stations by employing parallel processing architectures with representative RF front ends. While SDRs promise software re-configurability, it does not come for free. Of particular note, the computational partitioning of signal processing blocks among parallel heterogeneous computational engines that comprise today's wireless systems can be difficult and even daunting. Without a thorough understanding of processing, latency, and throughput; researchers may fall into a seemingly endless cycle of trial and error. Careful choice of tools and SDR platforms can significantly reduce the transition from simulation to prototype.

General Purpose Processors (GPPs) have benefitted from Moore's Law **(Figure 2)** and now offer SDR developers a rich option for prototyping both PHY and even upper layer software.

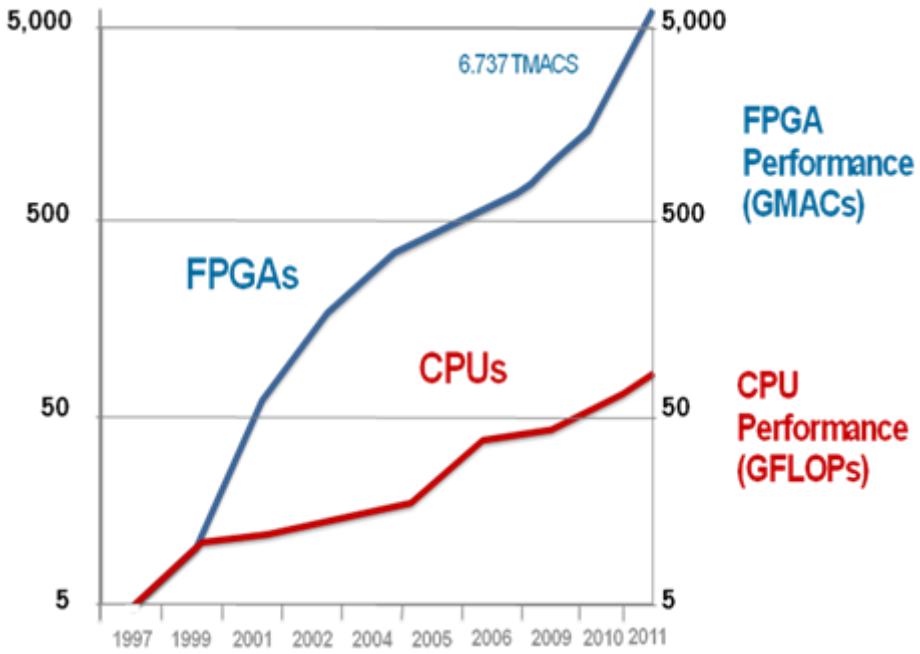

Figure 2. *Performance of CPU/GPPs and FPGAs over time.*

Rather than increasing the GPP clock rate, GPPs have become parallel execution engines in their own right as GPP manufacturers increase the number of processing cores in a single monolithic integrated circuit. Multiple cores in a GPP increase processing capability exponentially. When using SDRs with multi-core GPPs, the system developer must distribute the signal processing blocks among the cores and coherently bind them together. With this approach, researchers can avoid many system and software development pitfalls posed by heterogeneous processing architectures. Ideally, SDRs could use a single multi-core GPP to simplify development. However in many of today's applications, this approach is not practical as issues such as determinism and latency are difficult to address reliably with just a single multi-core GPP and standard operating systems. In these cases, more sophisticated and capable SDR platforms using GPPs and FPGAs provide an attractive option as FPGAs **(Figure 3)** can address the real-time signal processing requirements of wireless research.

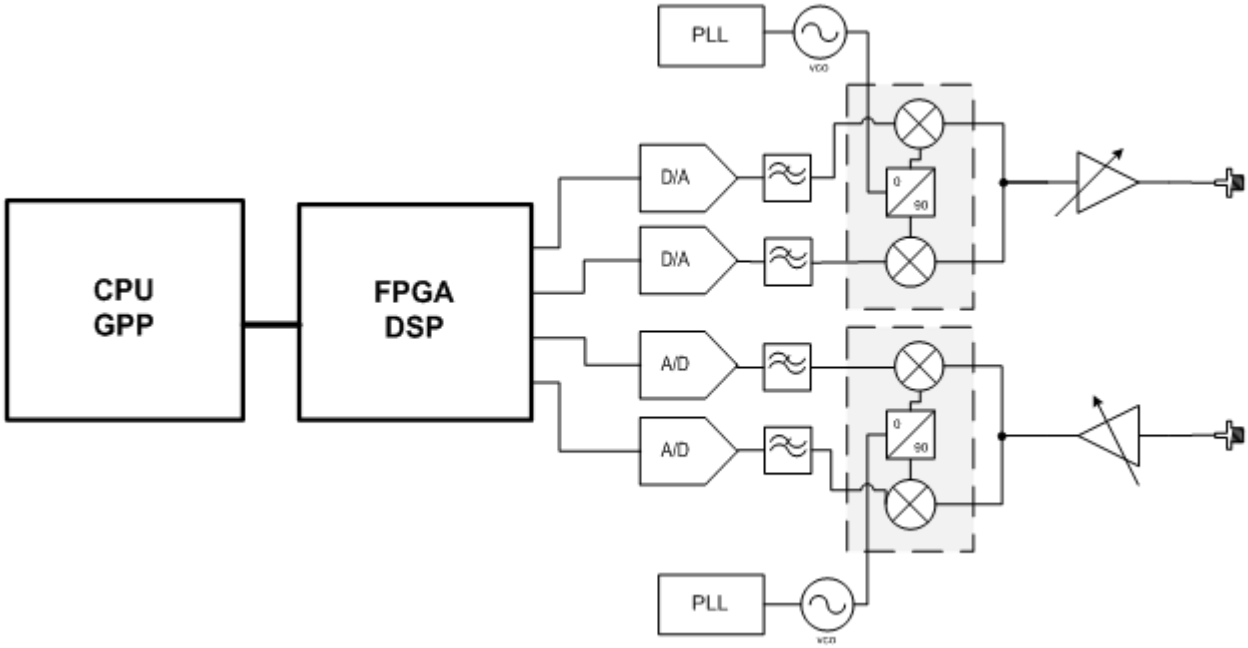

Figure 3: *Generic SDR block diagram*

3. Simplified Approaches to PHY Layer Design and Prototyping

Although SDRs promise to further and hasten the time to results for wireless researchers, the actual system development and integration can take time especially with systems that have loose hardware / software connectivity compounded by the need to use multiple yet disaggregated software development tools.

Some attributes to consider when developing a real-world prototype using an SDR are:

1. Tight hardware integration with abstraction
2. Software tools that offer simulation, target compilation, and compatibility with models of computation embracing other languages/tools
3. Heterogeneous multiprocessing capabilities to simplify algorithm deployment among the computational engines

Hardware Abstraction

Software environments that provide tight integration with the hardware may offer the benefit of hardware abstraction. By abstracting the hardware complexity, researchers can quickly transition from simulation to a prototype as simpler, easier to understand hardware APIs are presented in the software tool. Abstraction does come with a price. Ostensibly, abstraction offers faster prototyping at the expense of optimization of platform resources. Software design tools that offer multiple layers of abstraction from high to low (meaning more granular control) can enable researchers to tradeoff development time for optimization and vice versa to finely tune system performance.

Comprehensive Software Development Environments

SDR system development tools must not only comprehend heterogeneous processing elements, but also different models of computation and other design languages where initial concepts may be developed. The system design software should embrace software IP blocks developed with other tools as well as code created in different languages or encapsulated as a binary object (ex Dynamic Linked Library, VHDL or other). By accommodating alternative models of computation and design languages, comprehensive software development environments, and system design tools can leverage code developed with other tools maximizing software reuse.

System design tools that offer communications and signal processing libraries also facilitate rapid prototyping. For example, a researcher may be only interested in prototyping a single PHY algorithm, such as a new modulation technique or coding scheme in a signal chain. However to prototype with real signals and waveforms, the other blocks in the chain must be present during execution to assess performance of the modified or new block. Without the basic communications IP blocks, the researcher would have to build these blocks from scratch; a very expensive and time-consuming process.

A final point regarding comprehensive software development environments and system design tools refers to simulation. Simulation is a critical step in the design and development process as transitioning to the hardware prototype takes time even for optimized flows. FPGAs, in particular, suffer from long compile, synthesize, and place-and-route times that delay time to result. Consider the inevitable iteration – trial and error – on algorithm development where each cycle incurs this delay. As discussed above, the transition from the floating point model to a fixed point model must be addressed for the researcher to gain confidence in the fixed point implementation prior to FPGA deployment. Software environments that provide the tools to iterate data types and evaluate the algorithm at a bit-by-bit level (bit exact) can streamline the development by providing immediate visibility into a design. These tools can produce metrics for evaluating the quality of result (QOR) along with providing a test bench to automate the QOR analysis and confirm functionality.

Simulation and even emulation where the hardware environment can be co-simulated with the algorithms in a system are particularly valuable in expediting designs and building rapid prototypes. Tools that ultimately enable simulation on the host GPPs are critically important to the researcher.

Heterogeneous Multiprocessing

Most SDR software tools focus on a specific hardware target encompassing a wide range of GPPs, FPGAs, or DSPs. With various processing elements and multiple cores available on many SDRs, a common design flow with heterogeneous multiprocessing capabilities is desired to shorten the learning curve compared to multi-tool approaches, to streamline software development in a unified design flow, and to simplify system integration. Alternatively, an approach that employs several software tools significantly extends the system integration timeline as each dedicated tool focuses on a specific task and

specific core, and the user must integrate the individual components into a real working system using a variety of debugging features offered by each tool. True heterogeneous multiprocessing (**Figure 4**) capabilities enable users to build systems using a common language and design flow to multiple targets, and can reduce the learning curve associated with having to master multiple tools with the additional benefit of simplifying system integration.

In addition to heterogeneous multiprocessing, tools that comprehend parallel execution, pipelining, and multiple data flows match well with the parallel execution SDR paradigm. Sequential languages and compilers do not expose parallelism to the user nor do they present methods to realize true parallel execution, whether the processing target is an FPGA, DSP or a multi-core GPP.

A common design flow facilitating the transition from concept to simulation and to prototype is rare but necessary to reduce the time to build the prototype and facilitate rapid iteration. In fact, communications systems design tools that present a common, integrated flow with hardware abstraction enable communications and signal processing domain experts to rapidly prototype and iterate on their ideas reducing development time and expense.

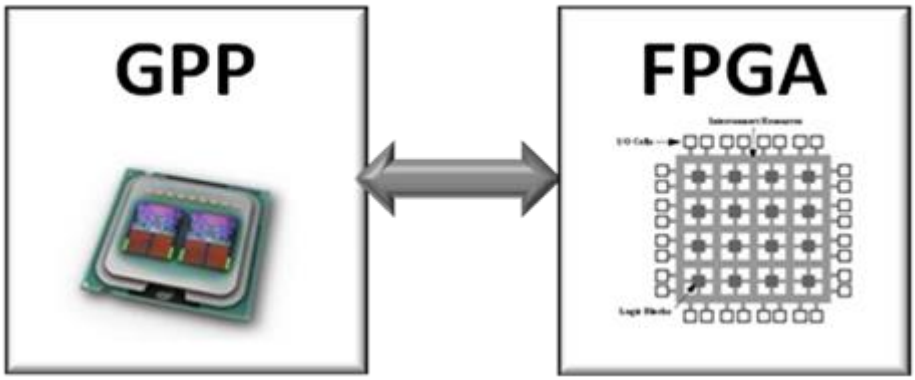

Figure 4. *Heterogeneous multi-processing architecture*

4. Host-based SDRs

LabVIEW graphical design system software from National Instruments offers many of the attributes noted above for rapid prototyping with SDRs. The tight integration between LabVIEW and SDR platforms, such as the Universal Software Radio Peripheral (USRP ™), abstracts the low-level hardware complexities and enables the researcher to focus on improving and evolving their algorithms with a flexibility of system partitioning.

LabVIEW offers parallel constructs spanning multiple cores, threads and targets providing a seamless software development flow from host to real-time embedded processors and also to FPGAs. Researchers can incorporate their design code in the LabVIEW environment with full emulation on the host and then migrate the code algorithms to faster targets or multiple GPP cores to meet timing constraints.

The NI USRP SDR architecture provides both GPP (via a host computer connected through a Gigabit Ethernet connection) and FPGA processing capabilities. The FPGA executes basic channelization, pulse shaping, and downsampling / upsampling signal processing functions in real-time with limited run-time configuration options. The researcher develops and executes the PHY code on the Host GPP and can iterate quickly using the inherent floating point capabilities. There are advantages and disadvantages to this approach. A brief summary is presented below:

Advantages

1. Single target – multi-core GPP, which reduces system design complexity
2. No floating point to fixed point math transition
3. Tight hardware/software integration with abstraction
4. Comprehensive development platform for multi-core software design and implementation
5. Rich communications software libraries available to speed development
6. Compatibility with other software design tools and languages

Disadvantages

1. Limited FPGA real-time processing capacity

The combination of NI USRP and LabVIEW with multi-core GPPs **(Figure 5)** offer advantages to SDR developers by providing multiple processing engines running in parallel to effectively increase processing capacity higher than increasing the clock rate alone. The developer can partition signal processing tasks among the available cores so that the code runs in parallel. In effect, the greater number of cores, the more capable the prototype. Considering this architecture, more of the user code executes on the Host GPP. Unfortunately, code written targeting standard off-the-shelf operation systems does not execute predictably or deterministically. However as many researchers have seen, pedestrian GPP technology continues to advance overcoming many latency constraints to address more and more SDR and wireless prototyping applications. Furthermore, as GPPs incorporate more cores, faster clocks rates and more sophisticated caching schemes, SDRs such as the NI USRP with LabVIEW inherently become more powerful.

Should the NI USRP lack the computational horsepower to facilitate more sophisticated prototyping, other options are available.

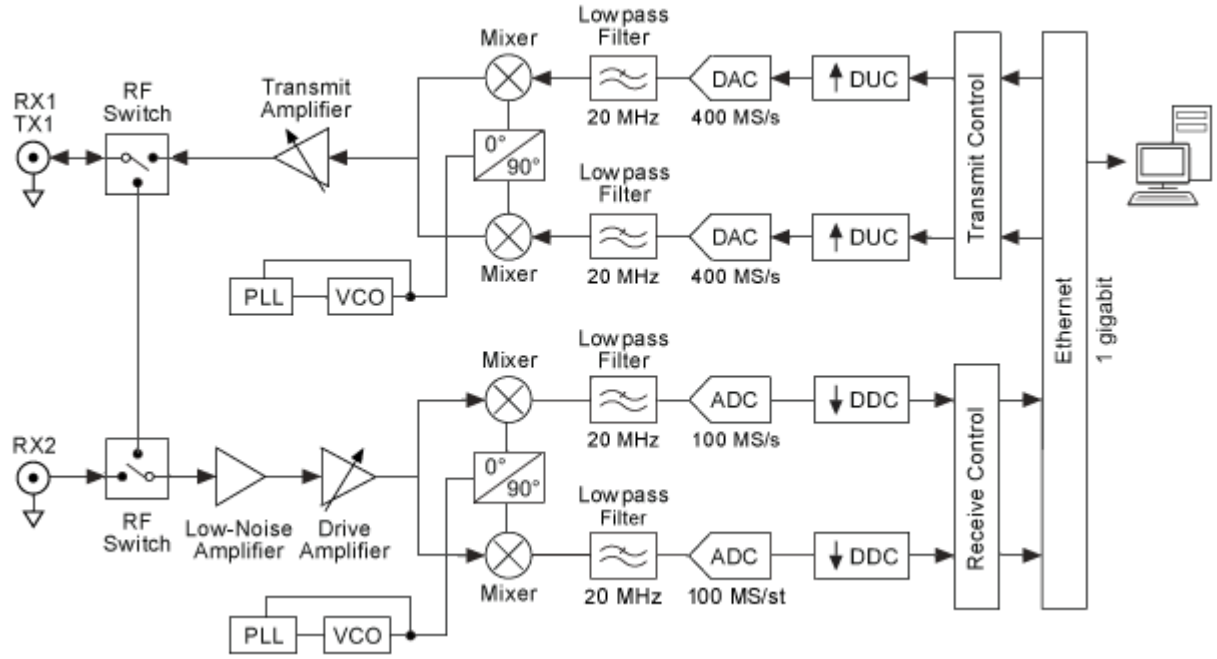

Figure 5. *NI USRP-2920 System Block Diagram*

5. Heterogeneous Processing SDRs

Research that requires significant computational capabilities, including dedicated real-time signal processing, may require FPGAs and GPPs, and perhaps multiple instances of each **(Figure 6)**. As discussed above, developing software for a heterogeneous multiprocessing system prototype can be challenging and may require "tool specialists" to implement the algorithms and software honed to a specific processing target.

The researcher develops and executes the PHY code on the Host GPP and then partitions the Rx and/or Tx chain among the processing elements. There are advantages and disadvantages to this approach. A brief summary is presented below:

Advantages

1. Rich processing environment to address demanding wireless applications
2. Tight hardware/software integration with abstraction
3. Comprehensive development platform for multi-core software design and implementation
4. Rich communications software libraries available to speed development
5. Compatibility with other software design tools and languages

Disadvantages

1. Increased system complexity

LabVIEW graphical system design software enables users to develop software for each target using a common language. Algorithms can be developed on the host GPP in a simulation environment and then ported to any of the available processing elements as required by the system. In the example diagram, the amount of processing power available to the developer is significant – a multi-core Windows PC, a multi-core RT GPP, and a number of FlexRIO FPGAs modules – yet the system integration aspects are simplified because the software and hardware integration is abstracted. True software re-configurability can be achieved a number of ways, but with an integrated system design tool such as LabVIEW, the path to prototyping is shorter.

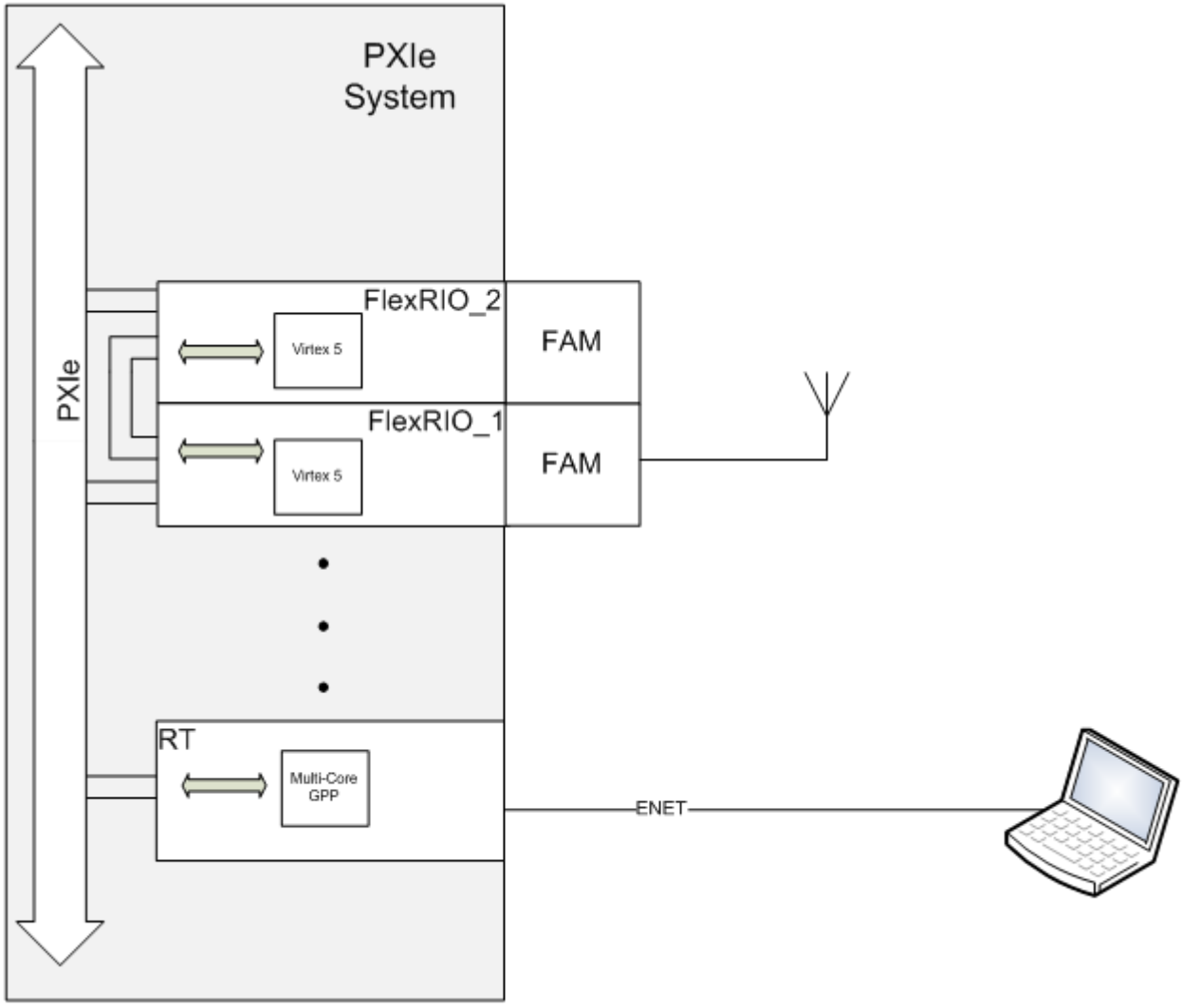

Figure 6: NI FlexRIO and LabVIEW RT SDR System Diagram

6. Conclusion

Wireless bandwidth demands are quickly reaching the capacity of the available spectrum. Therefore, new technologies are needed to extract more bits from the same amount of spectrum. As wireless researchers struggle to transition their ideas to standardization for mass deployment, SDRs offer software re-configurability, which in turn empowers researchers to quickly transition from simulation to real working prototypes using live signals and waveforms and iterate on their designs ultimately expediting deployment. In this way, SDR platform using graphical design tools such as LabVIEW deliver a path to faster time to results and validation of new ideas to address the world's wireless bandwidth challenges.