Four Quick Steps to Production: Using Model-Based Design for Software-Defined Radio

Part 1—the Analog Devices/Xilinx SDR Rapid Prototyping Platform: Its Capabilities, Benefits, and Tools

By Di Pu, Andrei Cozma, and Tom Hill

Abstract

There is a significant gap between the concept of a wireless system and the realization of that working design. Bridging this gap typically involves teams of engineers with a variety of different skill sets (such as RF, SW, DSP, HDL, and embedded Linux®), and in many cases projects get derailed early in the development stage because of the difficulty in coordinating the efforts of these varied design entities.

In this four part article, we will examine the advances in platforms and tools that allow developers to quickly simulate and prototype wireless systems while establishing and maintaining a deployable path to production. As a real-world example of the process, we will prototype a wireless SDR platform that receives and decodes automatic dependent surveillance broadcast (ADS-B) signals to allow us to detect and report the position, altitude, and velocity of the commercial aircraft flying in our vicinity. The resources required in this case are MATLAB® and Simulink and the skills to integrate and embed hardware/software. The hardware platform will be the Analog Devices/Xilinx® software-defined radio (SDR) prototyping system. Using MATLAB and Simulink® the following tasks will be performed:

- Design of signal processing algorithms used to decode ADS-B messages
- Simulation of the RF transceiver receiving ADS-B signals
- Generation of C and HDL code
- Verification of the HDL code with recorded and live data on the target transceiver and FPGA

The final result will be a working RF SDR design running on production-worthy hardware, which we will take to a local airport and verify its performance and functionality.

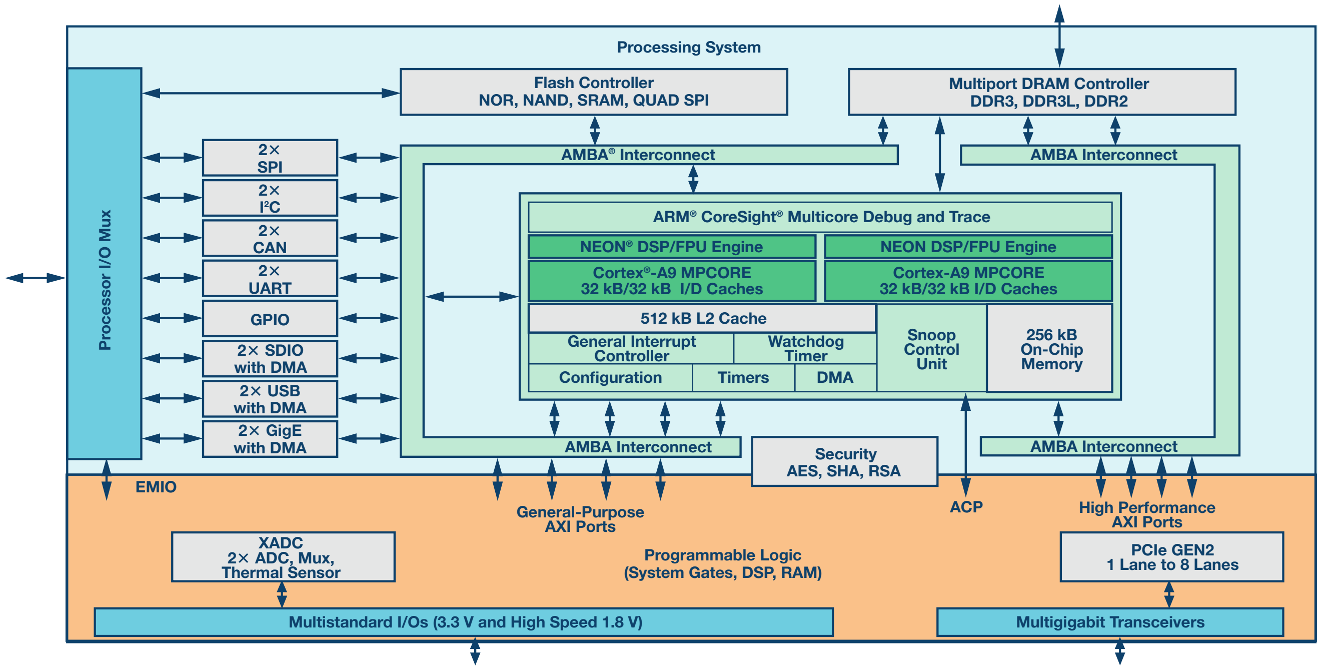
The first part of this four part article will discuss the Analog Devices/Xilinx SDR prototyping system, its capabilities and benefits, and a brief description of the tool flow. The second part will review the automatic dependent surveillance broadcast (ADS-B) signals and explain how to decode their information in MATLAB and Simulink in simulation. The third part will describe and showcase how to use hardware in the loop (HIL) and capturing signals with the target transceiver, but still doing the signal processing on the host in Simulink for verification. The fourth part will show how to take the algorithm developed in part 2, verified in part 3, and use HDL Coder and Embedded Coder from MathWorks to generate code and deploy it in the production hardware, and finally we'll operate the platform with real-world ADS-B signals at an airport.

Introduction

With the exponential growth in the ways and means by which people need to communicate, modifying radio devices easily and cost effectively has become business critical. Based on this requirement, software-defined radio technology has been widely employed recently since it brings the flexibility, cost efficiency, and power to drive communications forward.[1] The purpose of an SDR system is to implement as much as possible of the modulation/demodulation and data processing algorithms in software and reprogrammable logic so that the communication system can be easily reconfigured just by updating the software and the reprogrammable logic and not making any changes to the hardware platform.
With the advent of system on chip (SoC) devices like the Xilinx Zynq.® All Programmable SoC that combine the versatility of a CPU and the processing power of an FPGA, designers have the means to consolidate the data processing functions of an SDR system into a single device while integrating additional processing tasks. Processing intensive tasks like the data modulation/demodulation algorithms are offloaded to the programmable logic of the device while tasks like data decoding and rendering, system monitoring and diagnosis and user interface are deferred to the processing unit.

At the same time, prototyping wireless systems has been a discussion topic for decades but has only in recent years evolved into a complete design flow for FPGAs—from model creation to complete implementation—due to the evolution of the modeling and simulation tools like MATLAB and Simulink from MathWorks. Prototyping wireless systems is transforming the way engineers and scientists work by moving design tasks from the lab and field to the desktop.[2] Now the entire wireless system, such as an SDR system, can be modeled, allowing the engineer to observe the system's behavior and to tune it before it is actually implemented in the field. This has several benefits, such as accelerating system integration and reducing the dependency on equipment availability. Moreover, once the Simulink model for the SDR system is complete, C and HDL code can be generated automatically for implementation on Zynq SoCs, saving time and avoiding the introduction of manually coded errors. The risk is further reduced by linking the system model to a rapid prototyping environment that allows the SDR system to be exercised under real-world conditions.

This first part of the four part article series will discuss the Analog Devices/Xilinx SDR rapid prototyping system, its capabilities and benefits, and a brief description of the tool flow. The article showcases how Analog Devices RF IC technology and reference design hardware and software require a reduced design skill subset, thus enabling customers to mitigate risk and shorten their time to market.



Processing System
Flash Controller NOR, NAND, SRAM, QUAD SPI
Multiport DRAM Controller DDR3, DDR3L, DDR2
Processor I/O Mux
2× SPI
2× I²C
2× CAN
2× UART
GPIO
2× SDIO with DMA
2× USB with DMA
2× GigE with DMA
AMBA® Interconnect
AMBA Interconnect
ARM® CoreSight® Multicore Debug and Trace
NEON® DSP/FPU Engine
NEON DSP/FPU Engine
Cortex®-A9 MPCORE 32 kB/32 kB I/D Caches
Cortex-A9 MPCORE 32 kB/32 kB I/D Caches
512 kB L2 Cache
General Interrupt Controller
Watchdog Timer
Configuration
Timers
DMA
Snoop Control Unit
256 kB On-Chip Memory
AMBA Interconnect
Security AES, SHA, RSA
AMBA Interconnect
EMIO
General-Purpose AXI Ports
ACP
High Performance AXI Ports
XADC 2× ADC, Mux, Thermal Sensor
Programmable Logic (System Gates, DSP, RAM)
PCIe GEN2 1 Lane to 8 Lanes
Multistandard I/Os (3.3 V and High Speed 1.8 V)
Multigigabit Transceivers

Figure 1. Xilinx Zynq SoC block diagram.

Zynq for SDR

Advanced SDR systems are required to execute a combination of data processing, communication, and user interface tasks that have different processing bandwidth requirements and real-time constraints. The hardware platform chosen to implement such a system must be robust and scalable at the same time allowing for future system improvements and expansion. Xilinx Zynq-7000 All Programmable SoCs fulfill these requirements by supplying a high performance processing system combined with programmable logic as shown in Figure 1.[3] The combination of programmable logic and processing system delivers superior parallel processing power, real-time performance, fast computational speeds, and connectivity versatility.

The processing system side of the Zynq SoC consists of a dual-core ARM® Cortex®-A9 processor combined with a NEON coprocessor and floating-point extensions to accelerate software execution. Embedded Linux or real-time operating systems can be deployed on the dual-core ARM processor to fully benefit from the system's capabilities. The processor is self-contained and can be used without the need to configure the programmable logic, which is a critical element for software developers who will want to start developing code in parallel to hardware developers who will design the FPGA fabric.

On the programmable logic side, the device has up to 444,000 logic cells and 2,200 DSP slices that supply massive processing bandwidth, allowing the Zynq device to tackle a variety of challenging signal processing applications. Five high throughput AMBA®-4 AXI high speed interconnects tightly couple the programmable logic to the processing system with the equivalent of more than 3,000 pins of effective bandwidth.[4]

AD9361 Agile Wideband RF Transceiver IC for SDR

In recent years, Analog Devices has brought to market revolutionary SDR products to support increasingly evolving SDR requirements and system architectures. Some of the most important Analog Devices products in this field are the AD9361/AD9364 integrated RF agile transceivers. The AD9361 (2 × 2)[5] and AD9364 (1 × 1)[6] are high performance, highly integrated RF transceiver ICs intended for use in SDR architectures in applications such as wireless communications infrastructure, defense electronics systems, RF test equipment and instrumentation, and general software-defined radio platforms. The devices combine an RF front end with a flexible, mixed-signal baseband section and integrated frequency synthesizers, simplifying design-in by providing a configurable digital interface to a processor or FPGA. The chips operate in the 70 MHz to 6 GHz range, covering most licensed and unlicensed bands, and support channel bandwidths from less than 200 kHz to 56 MHz by changing the sample rate, digital filters, and decimation, all programmable within the AD9361 and AD9364 devices.[7] Figure 2 shows the block diagram of a AD9361 device.

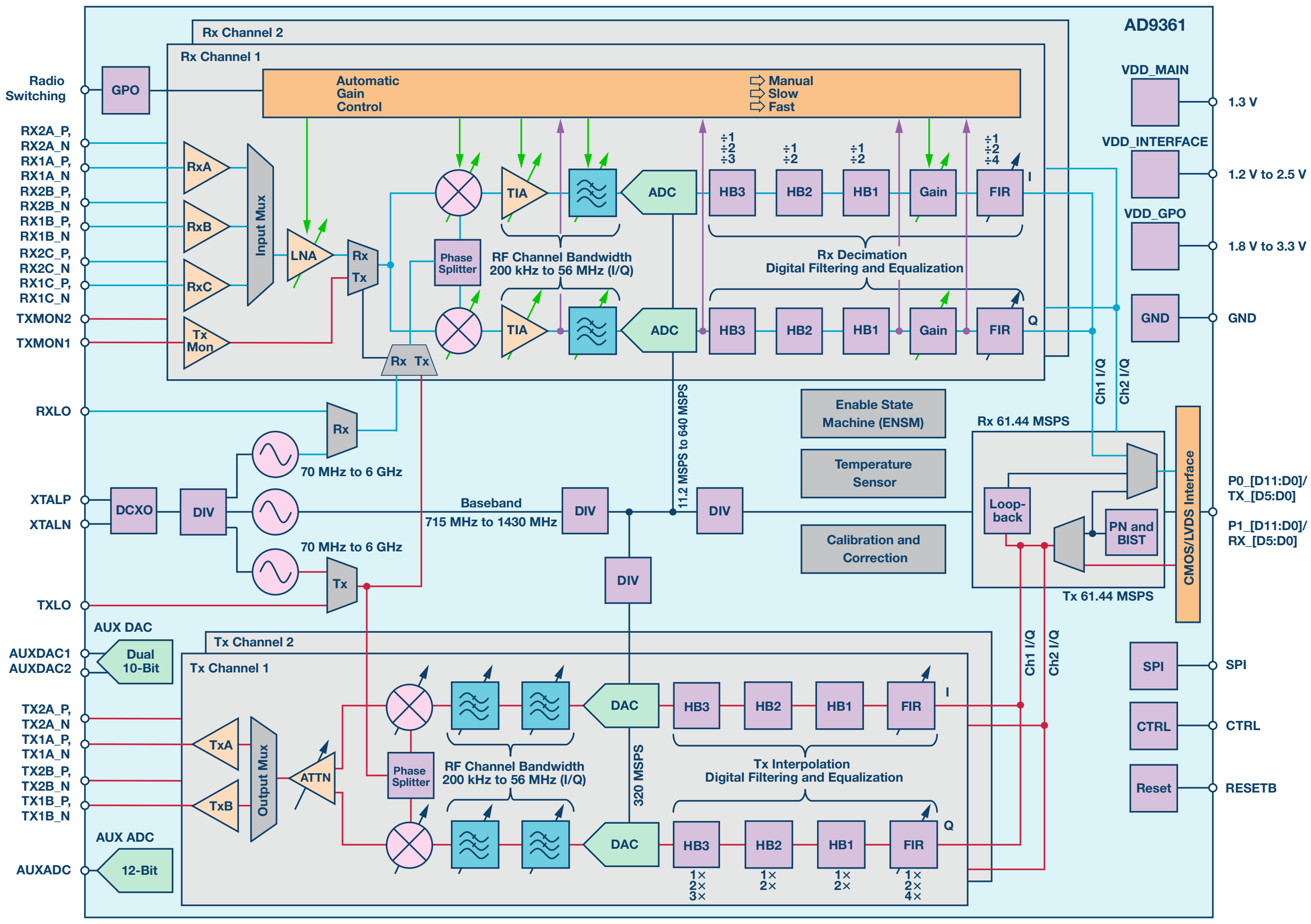


Figure 2. AD9361 block diagram.

In order to help customers shorten time to market and overall development effort, Analog Devices has gone a step further by providing SDR solutions within a complete ecosystem of seamless FPGA connectivity, enabling a rapid prototyping and development environment for complete radio system design. The AD-FMCOMMSx-EBZ rapid development and prototyping boards are a family of high speed analog FMC modules, incorporating AD9361 or AD9364 agile RF transceiver ICs or a discrete signal chain that seamlessly connects to the Xilinx FPGA development platform ecosystem. These boards are fully customizable by software without any hardware changes and come with downloadable Linux drivers and bare metal software drivers, schematics, board layout, and design aid reference materials, all contained on their respective Analog Devices wiki sites. Table 1 summarizes the features of the different FMCOMMSx platforms.

Table 1. FMCOMMSx Platforms

Platform	Features
AD-FMCOMMS5-EBZ	Integrating two AD9361 2 × 2 agile transceiver ICs, this SDR rapid prototyping board provides full synchronization capability for four receiver channels and four transmitter channels, enabling any subset of a 4 × 4 MIMO system to be created. Wideband 70 MHz to 6 GHz and 2.4 GHz tuned ports are accommodated. AD-FMCOMMS5-EBZ resource wiki page: http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms5-ebz
AD-FMCOMMS4-EBZ	Integrating the AD9364 agile RF transceiver IC, this 1 × 1 SDR rapid prototyping board can be software configured for highest RF performance in the 2400 MHz to 2500 MHz region, or can be software configured to operate over the AD9364's complete RF tuning range of 70 MHz to 6 GHz for system prototyping and development purposes. AD-FMCOMMS4-EBZ resource wiki page: http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms4-ebz
AD-FMCOMMS3-EBZ	Integrating the AD9361 agile RF transceiver IC, this 2 × 2 version of SDR rapid prototyping board supports the AD9361's full RF tuning range of 70 MHz to 6 GHz. This kit is ideal for the wireless communications SDR system architect seeking a unified development platform with wide tuning capabilities. AD-FMCOMMS3-EBZ resource wiki page: http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms3-ebz
AD-FMCOMMS2-EBZ	Integrating the AD9361 agile RF transceiver IC, this 2 × 2 SDR rapid prototyping board is tuned for highest RF performance in the 2400 MHz to 2500 MHz region. This kit is ideal for the RF engineer seeking optimized system performance meeting AD9361 data sheet specifications within this defined range of RF spectrum. AD-FMCOMMS2-EBZ resource wiki page: http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz

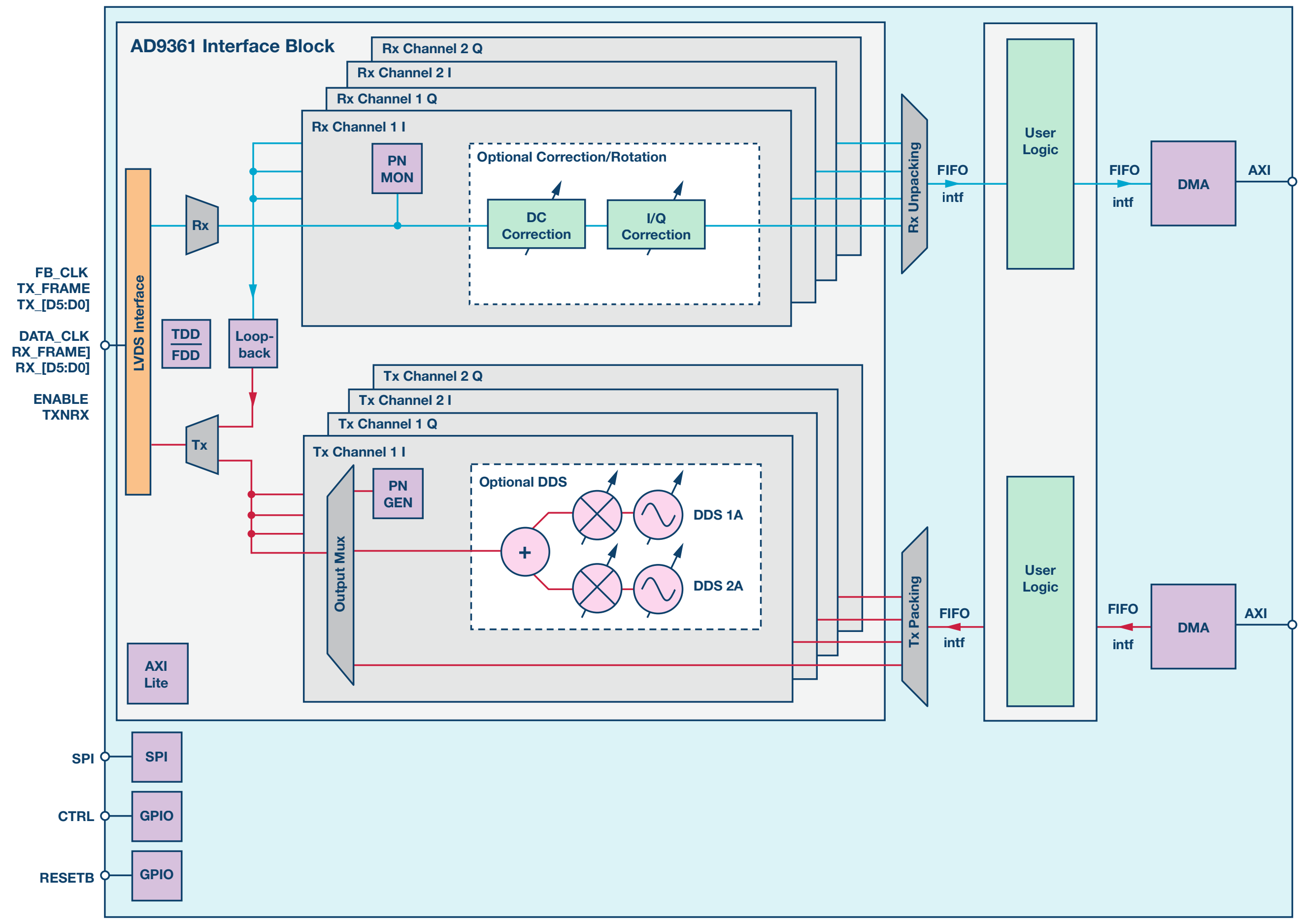


Figure 3. ADI HDL and software infrastructure.

Zynq SDR Rapid Prototyping Platform

Reference Design

Together with the FMCOMMSx platforms, Analog Devices provides a complete Vivado framework, with a Linux and bare metal software infrastructure that can be used both for prototyping purposes as well as a part of the final production system. Figure 3 shows the Analog Devices Zynq Infrastructure to support the FMCOMMSx boards.

This high level diagram shows how the ADI reference design is partitioned on a Xilinx Zynq SoC. An HDMI output is used to display the Linux interface on a monitor while a keyboard and mouse can be connected to the system on a USB 2.0 port. The ARM Cortex-A9 processing system runs Ubuntu Linux provided by Analog Devices. This includes the Linux IIO drivers needed to interface with the Analog Devices FMCOMMS hardware, the IIO Oscilloscope (Scope)[8] user space application for monitoring and control, a libiio server[9] that allows real-time data acquisition and system control over TCP together with clients running on a remote computer, and optional user applications that incorporate C code generated by the Embedded Coder for the controller's Simulink model.

Software Infrastructure

All ADI Linux drivers are based on the Linux Industrial I/O (IIO) subsystem, which is now included in all mainline Linux kernels. The IIO Scope is an open-source Linux application developed by Analog Devices that runs on the dual ARM Cortex-A[9] cores inside the Xilinx Zynq and has the ability to display real-time data acquired from any Analog Devices FMC card connected to the Xilinx Zynq platform. The data can be displayed either as a time domain, frequency domain, or constellation plot. Different popular file formats like comma separated values or .mat MATLAB data files are supported to save the captured data for further analysis. The IIO Scope provides a graphical user interface for changing or reading back the configuration of the Analog Devices FMC cards. The libiio server allows real-time data acquisition and system control over transmission control protocol (TCP) together with clients running on a remote computer.[10] The server runs on an embedded target under Linux and manages real-time data exchange over TCP between the target and a remote client. This library abstracts the low level details of the hardware, and provides a simple yet complete programming interface that can be used for advanced projects. Its modular architecture, well designed API, and built-in network capabilities allow the users to create applications that will run on the system not only where the IIO devices are connected, but also remotely through the network. At first targeted at Linux, it can now be used under Windows as well by using the remote back end of the library. Written in C and licensed under the LGPL, it features bindings for C#, Python, and MATLAB. A MathWorks IIO client[11] is available as a system object to be integrated in native MATLAB and Simulink applications. It is designed to exchange data over Ethernet with an ADI hardware system connected to a FPGA/SoC platform running the ADI Linux distribution, which enables a MATLAB or Simulink model to perform the following functions:

- Stream data to and from a target
- Control the settings of a target
- Monitor different target parameters

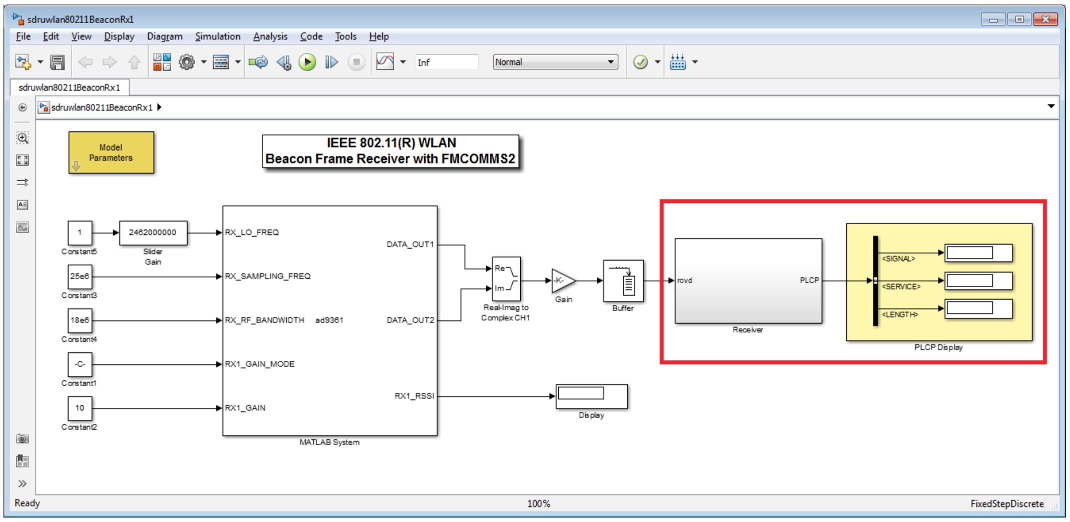


Figure 4. Screen capture of the beacon frame receiver example.

The IIO System Object is available in both MATLAB and Simulink, depending on whether the user calls it from a MATLAB script or incorporates it into a MATLAB System Block. The Linux software and HDL infrastructure provided by ADI for the FMCOMMS platforms is a great environment for prototyping SDR applications together with the tools provided by MathWorks and Xilinx, and it also contains production ready components that can be integrated into the SDR system—helping to reduce the time and cost needed to move from concept to production.

In order to help customers ramp up quickly and easily with the IIO System Object, we provide several MATLAB and Simulink examples based on this interface, such as a beacon frame receiver,[12] QPSK transmitter and receiver,[13] as well as a LTE transmitter and receiver.[14] In these examples, FMCOMMSx platforms are configured by IIO System Object, and are used as RF front ends, which transmit or receive the analog signals over the air. These signals are streamed to or from the target via the IIO System Object. All the other signal processing happens in MATLAB or Simulink. Figure 4 is a screen capture of the beacon frame receiver example, which shows a typical connection between the IIO System Object and the other Simulink blocks.

MathWorks Support for Zynq

MathWorks support for Zynq-based SDR comes from the following four aspects:

1. AD9361 Simulink Model

Since the AD9361 is an integrated RF transceiver chip, signal probing and internal operation monitoring is not really possible. For this reason, MathWorks and Analog Devices have codeveloped a SimRF™ model of the AD9361 that allows a simulation of the chip's operation so that customers can see exactly what's going on under the hood and how the chip performs under different test conditions that are hard to replicate in real life. SimRF provides a component library and simulation engine for designing RF systems using equivalent baseband or circuit envelope blocks, such as amplifiers, mixers, and S-parameter blocks. It is a useful and appropriate tool to model the AD9361 RF transceiver. The system-level AD9361 Agile RF Transceiver model, shown in Figure 5, replicates exactly the functionality of the AD9361 and is available to the users as a MathWorks hardware support package.[15]

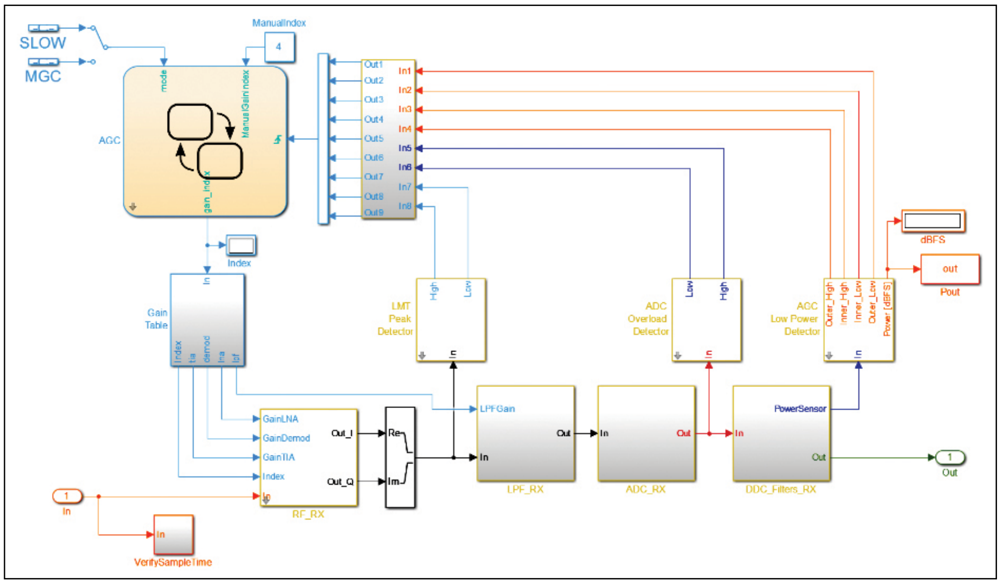


Figure 5. MathWorks SimRF model of AD9361 Agile RF receiver.

The SimRF models have been validated in a lab with power spectral measurements. The characterization of the transceiver's noise and nonlinearity at different frequencies and power levels are identified. The models are then designed to generate the same characterizations, which validates them across the range of design.

With the AD9361 transceiver SimRF models, the users can do the following:

- Predict the impact of the RF imperfections on the test signals
- Use reference tones and LTE signals
- Generate or import test vectors and evaluate the effects of nonlinearity, noise, gain, and phase imbalance, spectral leakage, and other imperfections introduced by the RF transmitter and receiver
- Add interfering signals and evaluate the results in the time or frequency domains

2. Communications and DSP System Toolbox Functions

MathWorks products such as the *Communications System Toolbox,*™16 Signal Processing Toolbox,™17 DSP System Toolbox,™18 and SimRF19 provide industry-standard algorithms and apps for systematically analyzing, designing, and tuning SDR systems. All of these tools provide the means to create high fidelity SDR models that can be used to verify the behavior and performance of the *communications* system before moving to the actual physical implementation.

3. Simulink Workflow for Zynq

MATLAB and Simulink from MathWorks are environments for multidomain simulation and model-based design that are well suited to simulating SDR systems with communication algorithms. Communication algorithms adjust gain, frequency offset, timing offset, and other performance variables, often for better synchronization between transmitter and receiver systems. Evaluating communication algorithms using simulation is an effective way to determine the suitability of SDR designs and reduce the time and cost of algorithm development before committing to expensive hardware testing. Figure 6 depicts an efficient workflow for designing a communication algorithm by following these steps:

- Build accurate SDR models using the libraries provided by the model-based design environment.
- Simulate system behavior to verify that the system is performing as expected.
- Generate C code and HDL for real-time testing and implementation.
- Test communication algorithms using prototyping hardware.

Once the performance of the SDR system is proven to be satisfactory through simulation and testing on the prototyping hardware, it is safe to take the system implementation and deploy it onto the final production system.

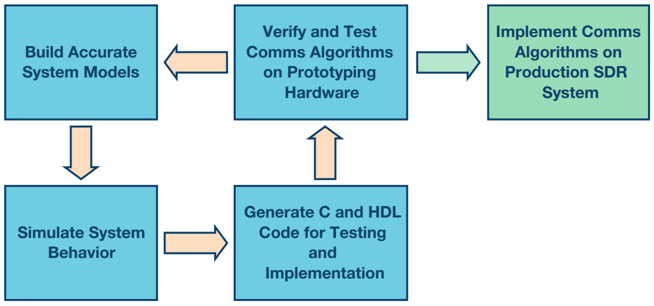

Build Accurate System Models
Verify and Test Comms Algorithms on Prototyping Hardware
Implement Comms Algorithms on Production SDR System
Simulate System Behavior
Generate C and HDL Code for Testing and Implementation

Figure 6. Workflow for communication algorithm design.

4. Simulink Platform Integration to Zynq SDR Kit

Once the SDR system is fully verified in the simulation environment using tools like the Embedded Coder®20 and the HDL Coder™21 from MathWorks, the user can generate C code with Embedded Coder and VHDL or Verilog using HDL Coder, and then deploy the code to prototyping hardware for testing, and afterward, onto the final production system. At this point, software and hardware implementation requirements are specified, such as fixed-point and timing behavior. Automatic code generation helps to reduce the time needed to move from concept to actual system implementation and avoids the introduction of manual coding errors, ensuring that the actual SDR implementation matches the model. Figure 7 depicts a real-life process of the steps needed to model a SDR system in Simulink and transfer it onto the final production system based on a Xilinx Zynq SoC.

The first step is to model and simulate the SDR system in Simulink. At this stage, the communication algorithm is partitioned into blocks that will be implemented in software and blocks that will be implemented into the programmable logic. Once the partitioning and the simulation are complete the SDR model is converted into C code and HDL using Embedded Coder and HDL Coder. A Zynq-based prototyping system is used to verify the performance of the communication

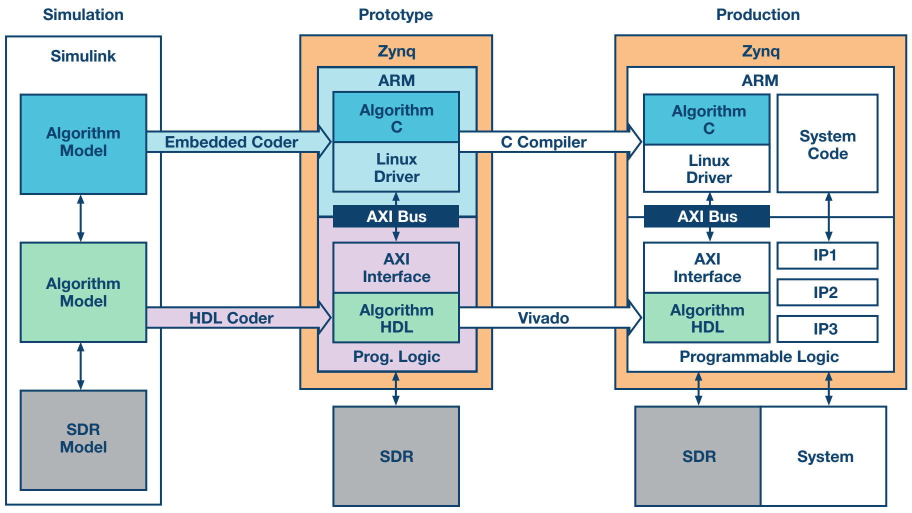


Figure 7. Path from simulation to production.

algorithm and to help further tune the SDR model before moving to the actual production stage. In the production stage, the automatically generated C code and HDL are integrated into the complex production system framework. This workflow ensures that once the communication algorithm reaches the production stage it is fully verified and tested and provides a lot of confidence in the system's robustness. Zynq Hardware Support Packages for Embedded Coder and HDL Coder make it easier to program the Zynq platform by providing a framework for integrated hardware/software design, simulation, and verification that integrate model-based design into the workflow, enabling rapid design iteration cycles and helping to detect and correct design and specification errors early.[22]

Conclusions

This article illustrated the requirements and trends of modern SDR systems and the tools and systems that MathWorks, Xilinx, and Analog Devices bring to the market in order to meet these requirements and help drive toward more performant SDR solutions. By combining the model-based design and automatic code generation tools from MathWorks with the powerful Xilinx Zynq SoCs and Analog Devices integrated RF transceivers, SDR systems design, verification, testing, and implementation can be more effective than ever, leading to higher performance radio systems and reducing the time to market. Analog Devices FMCOMMS platforms paired with the Avnet Zynq-7000 AP SoC provide a great prototyping environment for the SDR algorithms designed using MATLAB and Simulink from MathWorks. The FMCOMMS platforms are accompanied by a set of open source reference designs intended to give a starting point for anyone who wants to evaluate the system and help kick-start any new SDR project.

In the next article in this series, we will advance down the SDR design process as we review the characteristics of automatic dependent surveillance broadcast (ADS-B) signals and explain how to decode their information in MATLAB/Simulink in simulation.

For more information about the topics presented in this article, documentation, videos, and reference designs, check out the References section.

References

1 "What is Software-Defined Radio?" Wireless Innovation Forum.

2 Model-Based Design. MathWorks.

3 Zynq-7000 All Programmable SoC. Xilinx.

4 Hill, Tom. "Motor Drives Migrate to Zynq SoC with Help from MATLAB." *Xcell Journal*, Issue 87, Second Quarter, 2014.

5 AD9361.

6 AD9364.

7 "Software-Defined Radio Solutions from Analog Devices." Analog Devices.

8 IIO Oscilloscope. Analog Devices Wiki.

9 Simulink Libiio. Analog Devices Wiki.

10 What Is Libiio? Analog Devices Wiki.

11 IIO System Object. Analog Devices Wiki.

12 Beacon Frame Receiver Example. Analog Devices Wiki.

13 QPSK Transmitter and Receiver Example. Analog Devices Wiki.

14 LTE Transmitter and Receiver Example. Analog Devices.

15 AD9361.

16 "Communications System Toolbox." MathWorks.

17 "Signal Processing Toolbox." MathWorks.

18 "DSP System Toolbox." MathWorks.

19 SimRF. MathWorks.

20 "HDL Coder." MathWorks.

21 "Embedded Coder." MathWorks.

22 "Xilinx Zynq Support from Simulink." MathWorks.

Di Pu

Di Pu [di.pu@analog.com] is a system modeling applications engineer for ADI, supporting the design and development of software-defined radio platforms and systems. She has been working closely with MathWorks to solve mutual end customer challenges. Prior to joining ADI, she received her B.S. degree from Najing University of Science and Technology (NJUST), Nanjing, China, in 2007 and her M.S. and Ph.D. degrees from Worcester Polytechnic Institute (WPI), Worcester, MA, U.S.A., in 2009 and 2013—all in electrical engineering. She is a winner of the 2013 Sigma Xi Research Award for Doctoral Dissertation at WPI.

Andrei Cozma

Andrei Cozma [andrei.cozma@analog.com] is an engineering manager for ADI, supporting the design and development of system level reference designs. He holds a B.S. degree in industrial automation and informatics and a Ph.D. in electronics and telecommunications. He has been involved in the design and development of projects from different industry fields such as motor control, industrial automation, software-defined radio, and telecommunications.

Also by this Author:

FPGA-Based Systems Increase Motor-Control Performance

Volume 49, Number 1

Tom Hill

Tom Hill, system generator product manager, Xilinx, Inc. [tom.hill@xilinx.com]Tom Hill has over 18 years experience in the EDA industry. Hill oversees all products, strategic, and corporate marketing activities related to Xilinx's Target Design Platforms for DSP. Hill was most recently at AccelChip, Inc where he was technical marketing manager responsible for product direction and application of high level design methodologies and tools to DSP applications. Prior to AccelChip Hill held positions as product manager, technical marketing manager, technical marketing engineer, and field applications engineer for various FPGA and ASIC synthesis tools. Hill began his career as a hardware and ASIC design engineer at Allen-Bradley and Lockheed. Hill holds a B.S. in electrical engineering from Cleveland State University.