RFSoC & Python Integration and Higher Levels of Abstraction

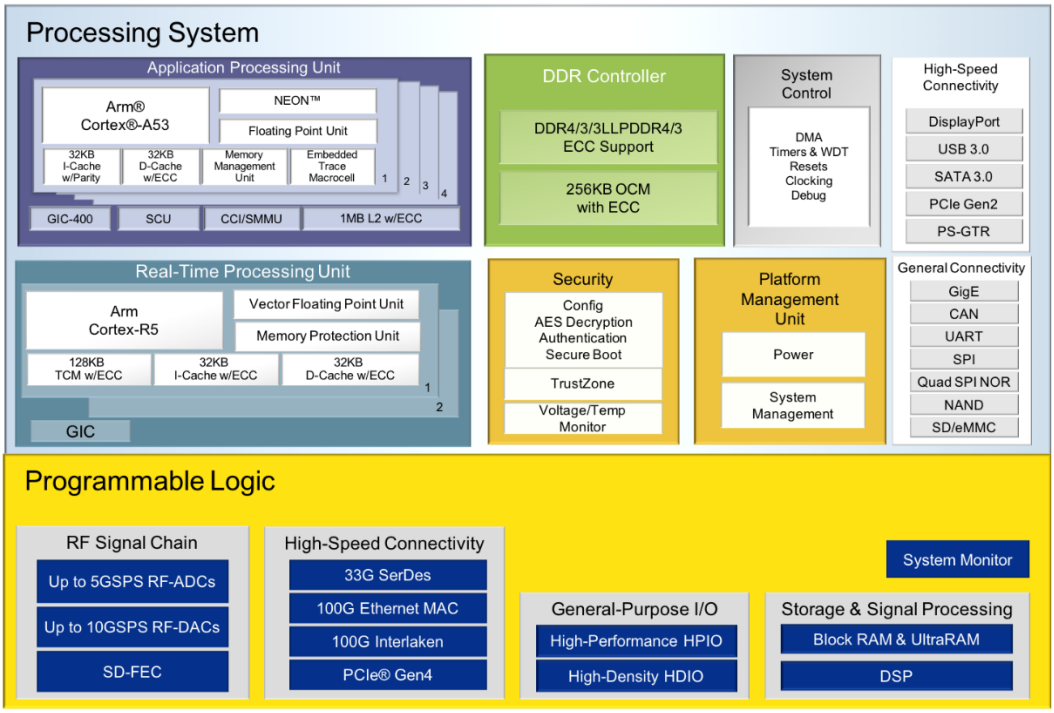
Adam Taylor & Stuart Smith Lincoln DSP

Adam@lincolndsp.com & Stuart@lincolndsp.com

www.linconldsp.com

The Xilinx RFSoC integrates for the first-time multi giga sample RF Analogue to Digital Convertors (ADC) and RF Digital to Analogue Converters (DAC) within a heterogeneous processing system. Traditionally high-performance mixed signal devices have not been implemented on CMOS processes. Instead discrete devices where used separating the processing elements from the RF data converters. However, thanks to digital assistance, it is possible to implement high performance converters with the processing elements using 16 nm CMOS FinFET technology (etal., 2017) (Christophe Erdmann et al., 2017).

In the RFSoC this fusion allows the mixed signal convertors to be tightly coupled with quad core Arm A53 64-bit processors, Arm R5 32-bit real time processors and High-Performance Programmable Logic which includes hard macro Soft Decision Forward Error Correction cores.



RFSoC Context Diagram

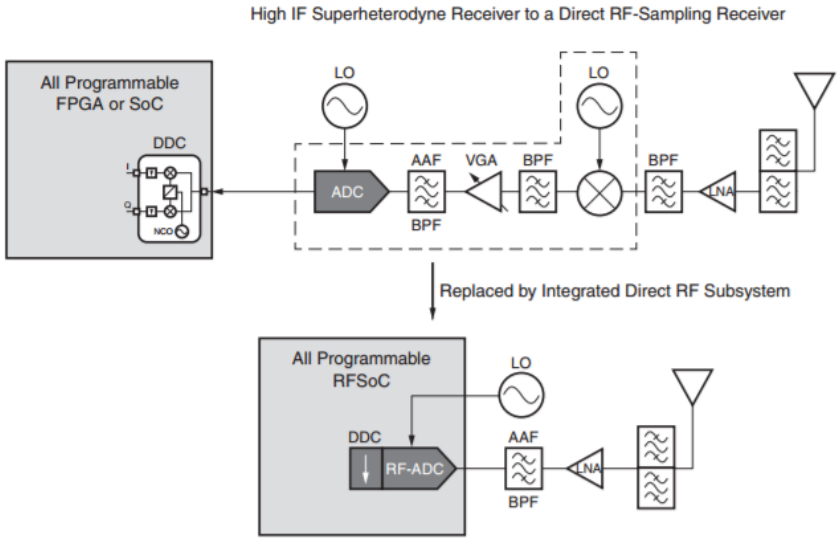
To the system developer such tightly coupled integration multi giga-sample sample RF Data Convertors with processing offers several benefits

1. Reduced PCB Complexity – Integrating the RF converters, reduces significantly the tracking required between the digital processing and the converters. This reduction allows for not only a simpler routing but also a more compact solution in many cases a RFSoC solution can be 50 to 75% smaller. This reduction in size is critical when developing RF Array solutions such as MIMO Antenna where module spacing needs to be of the of the order of $\lambda/2$ to prevent interference when formed into sub arrays

2. Simplified Clocking Architecture – The majority of the clocking network is within the RFSoC this simplifies the clock distribution network, reducing the area, cost and power of the RF clocking solution.

3. Reduced Power – An integrated solution requires fewer high-speed IO as such the power required to drive those transceivers is saved. Power savings of up to 50% are achievable using direct sampling RFSoC solution compared to a discrete implementation.

4. Flexible RF front end – The high sampling rate of the RF Convertors the RFSoC allows direct RF sampling. Direct sampling offers several advantages, not only a component count reduction. But also, as the solution is implemented digitally within the programmable logic it allows adaption as algorithms, waveforms and standards evolve over the years enabling updates in the field once deployed.



Direct Sampling Vs IF Conversion

5. Interfacing – To interface with the outside world, the RFSoC PS provides support for multiple industry standard interfaces such as GigE, SATA, USB3, PCIe, CAN, I2C and SPI etc. This allows the RF front-end, signal processing, control and communication to be implemented within a single device.

6. Security – Many solutions for the RFSoC are remotely deployed or in critical applications as such security of the solution is critical. To support secure solutions the RFSoC enables both secure configurations using AES256 encryption and run time antitamper and security features using the Sysmon and SECMON IP.

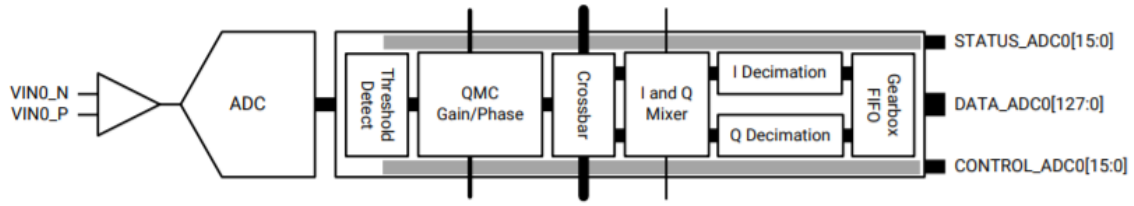
These factors enable the RFSoC to be used to create solutions which offer an optimal SWaP-C implementation for a range of applications from 5G to Industrial Internet of Things, Software Defined Radio, RADAR, Test and Measure, DOCSIS and Satellite Ground Stations.

RF Data Convertors

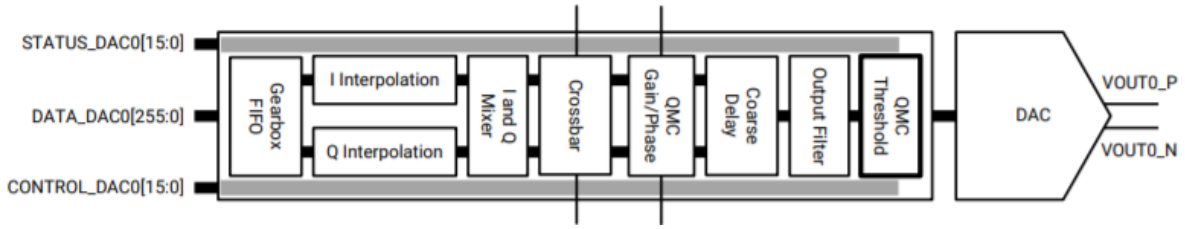
At the heart of the RFSoC is the RF Data Converter, to support the processing bandwidth necessary to effectively use the converters the RF Data Converter is located within the programmable logic.

The RF ADC are capable of sampling rates at up to 5 GSPS, while the RF DAC can achieve sampling rates at up to 10 GSPS.

Of course, it is not only the ADC or DAC converters which are included within the RF Data Converter element but also includes Phase Locked Loops, Complex Mixers, Decimators and Interpolators, Quadrature Modulation Correction along with support for the synchronisation across devices.



ADC Signal Path

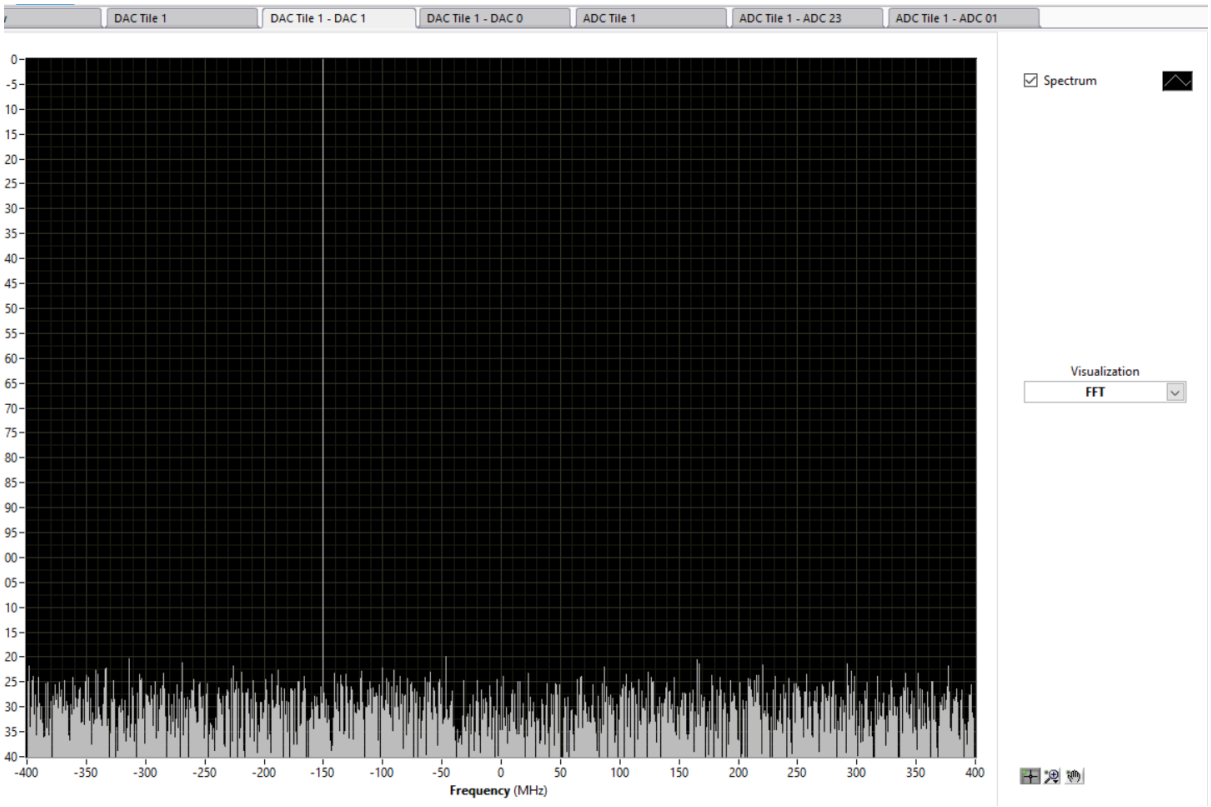


DAC Signal Path

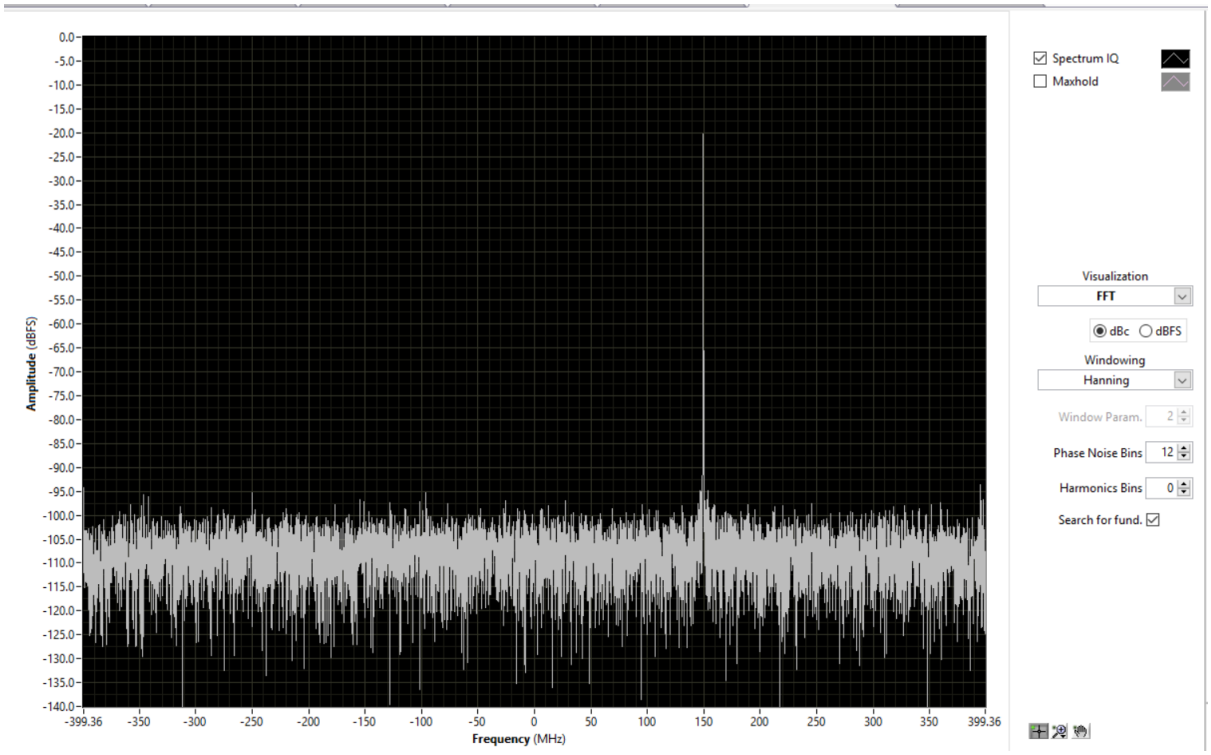
When it comes to performance of the RF DAC and RF ADC, they are comparable to those offered by discrete solutions offering:-

- ADC Full Power Bandwidth – 4 GHz
- ADC 2 GSPS – Fin 3.5 GHz Spurious Free Dynamic Range (SFDR) is 80 dBC
- ADC 4 GSPS – Fin 3.5 GHz Spurious Free Dynamic Range (SFDR) is 72 dBC
- Cross Talk between ADC channels -70 dBC
- DAC Full Power Bandwidth – 4 GHz
- DAC 6.5 GSPS – Fout 3.5 GHz Spurious Free Dynamic Range is 72 dBC with a 20 mA output drive.
- Cross Talk between DAC channels is -70 dBC

Spectral Plots of RF ADC and RF DAC performance below show the RF DAC output and RF ADC input using the RFSoC development board the ZCU111. The RF DAC was configured to generate a sine wave at 1350 MHz. The RF ADC samples this and uses the mixer and decimation to place the output sine wave at 150 MHz.



RF DAC Output



RF ADC Input

Configuring the RF Data converter for an application can be achieved using traditional development tools including Vivado and SDSoC. The RFSoC can also be programmed using System Generator which supports Matlab and Simulink based approaches.

PYNQ Framework

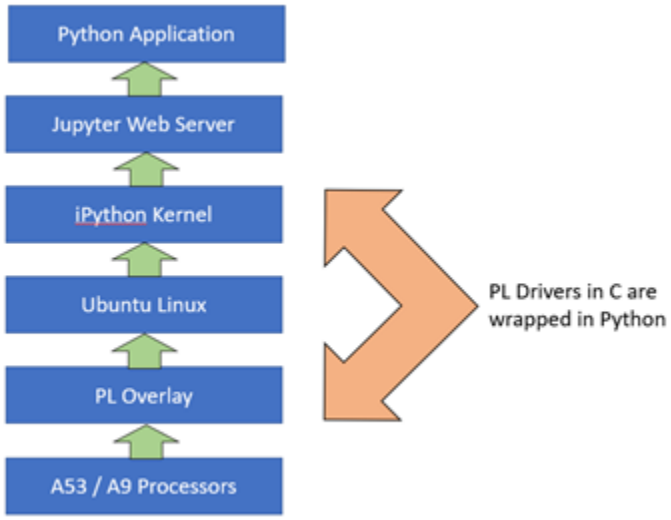
PYNQ (www.Pynq.io) is an open source project started by Xilinx, which fuses the productivity of Python with the acceleration provided by programmable logic within the Zynq / Zynq MPSoC / Zynq RFSoC.

The tight coupling of Processing System (PS) and Programmable Logic (PL) in the Zynq / Zynq MPSoC / Zynq RFSoC allows for the creation of systems that are more responsive, deterministic, and power efficient when compared to traditional CPU or GPU-based applications. This increase in performance is due to the ability to leverage the parallel nature of the programmable logic to move accelerate the application from the sequential software world to the parallel world enabled by programmable logic. However, developing the programmable logic solution requires advanced digital design skills limiting accessibility to software developers.

The PYNQ framework really is game changer in the programmable logic world. It enables the performance of programmable logic provided by Xilinx heterogeneous SoCs to be leveraged using Python one of the hottest programming languages, without the need to learn the digital design skills.

PYNQ framework uses iPython and Juypter notebooks enabling a browser-based development flow. To be able to run the iPython kernel, and Jupyter webserver PYNQ images run an Ubuntu-based Linux image containing Ubuntu's root file system, package manager and repositories on the processing system.

The PYNQ build flow itself is based upon the standard Xilinx PetaLinux build flow. This allows the PYNQ build to access all the Xilinx Kernel patches, board support packages, and crucially add in new drivers for processor system / programmable logic interfacing.

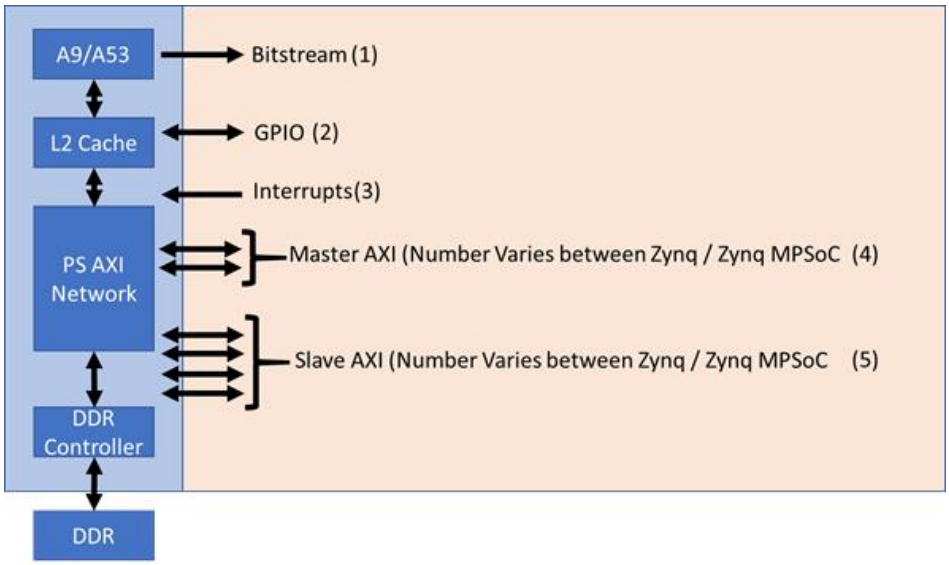


PYNQ framework stack

To transfer data between IP within the programmable logic and Jupyter notebooks, C-based drivers associated with the IP are encased in Python wrappers.

Communication between the processor system and the programmable logic depends on the interface used. In our PYNQ implementations, there are five different PS / PL interfaces which are used:

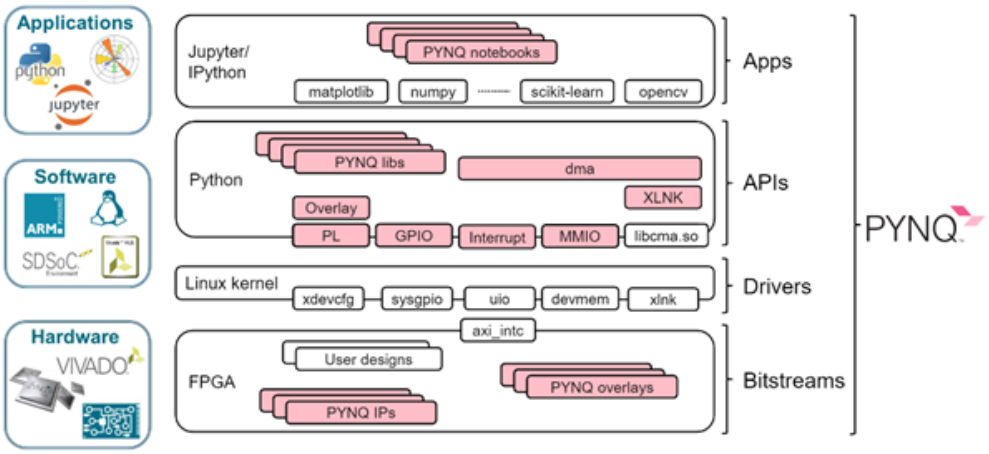
1. Bitstream — This configures the programmable logic for the desired application. In the PYNQ framework, the xdevcfg driver is used.
2. GPIO — This provides simple IO in both directions. In the PYNQ framework, this is supported by the sysgpio driver.
3. Interrupts — Support interrupt generation from the programmable logic to the processing system. In the PYNQ framework, this is supported by the Userspace IO driver.
4. Master AXI Interfaces — These are used to transfer data between the PS to the PL when the PS is the initiator of the transaction. The PYNQ framework uses devmem when employing master AXI interface.
5. Slave AXI Interfaces — These are used to transfer data between the PS and PL when the PL is the initiator of the transaction. The PYNQ framework uses xlnk to enable these transfers.



Interfacing between the PS and the PL in the PYNQ framework

The PYNQ framework builds upon these Linux Kernel drivers and offers a range of PYNQ specific APIs, which provide specific PYNQ libraries and drivers. These APIs let the Jupyter notebooks bridge the gap, accessing the programmable logic.

As many of the applications require transfer of large quantities of data between the PS and PL, these APIs include support for Direct Memory Access (DMA) using the xlnk driver.



PYNQ Framework

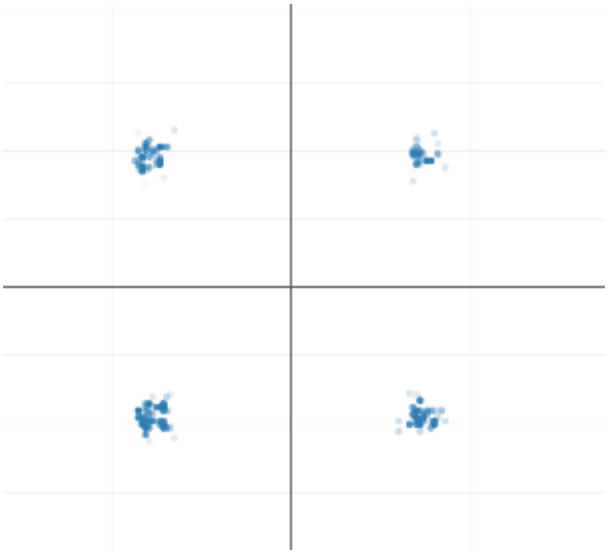
As developers, we leverage the PYNQ framework by connecting to the Jupyter server over a wired ethernet link and developing our application in a browser-based interface.

PYNQ overlays for the RFSoC therefore enable rapid prototyping of solutions using python libraries and frameworks. To support the RF Data Convertor within the RFSoC the PYNQ image contains additional libraries

- xrfdc: A Python driver for the RF data converters.
- xrfclk: A Python driver for the onboard clock synthesizers.

Of course, to get the best from the RFSoC PYNQ implementation, overlays are needed for the programmable logic which use the RF Data Convertor. One of the important aspects of PYNQ is the open source community who provide overlays including for the RFSoC. Using the open source community and sharing overlays in this manner opens the RF Data Convertor and programmable logic up to a wider range of developers than traditionally is the case.

One example overlay available for the PYNQ RFSoC Image is the University of Strathclyde QPSK overlay which demonstrates QPSK modulation.



QPSK Example

Conclusion

The RFSoC is a game changing device which enables system developers to address the increasing performance and SWaP-C challenges solutions face. Combining this capability, the productivity increases which comes when we use Python enables rapid application development and prototyping, further accelerating development.

Bibliography

[1] B. V. etal., "A 13b 4GS/s Digitally-Assisted Dynamic 3-Stage Asynchronous," *ISSCC,* 2017.

[2] Christophe Erdmann et al., "A 330mW 14-bit 6.8GS/s dual-mode RF-DAC in 16nm FinFET achieving -70.8dBc ACPR in a 20MHz channel at 5.2GHz," *ISSCC,* 2017.

[3] A. Collins and A. Taylor, "Addressing the 5G Challenge with Highly Integrated RFSoC," *Signal Interity Journal,* 2017.