Embedded GNU Radio running on Zynq/PlutoSDR

G. Goavec-Merou[1], P.-Y. Bourgeois[1], J.-M Friedt[1]
[1] FEMTO-ST Time & Frequency, Besançon, France

Abstract

GNU Radio has been ported to the `buildroot` environment and hence can be run on any platform supported by this development framework, including the Zynq. We extend the frozen version of `buildroot` used by Analog Devices (2018.02) to the `BR2_EXTERNAL` mechanism allowing to use the latest release of `buildroot` and hence the latest added packages, including GNU Radio and associated packages to run on the Zynq as found on the PlutoSDR. We demonstrate running the demodulation scheme on the PlutoSDR itself, and streaming the resulting audio file, as well as providing custom FPGA bitstream for embedded RF frontend processing.

1 Introduction

Current embedded platforms and associated electronics frontends exhibit on the one hand increasing flexibility and on the other hand increasing embedded computational power, with a bandwidth bottleneck at the data transfer from one processing unit to another. A demonstration of this evolution is Analog Device's (ADI) PlutoSDR combining on a same board an AD9363 radiofrequency frontend streaming digital data to the Zynq 7010 System on Chip providing both FPGA (PL) and general computational (PS) functionalities on the same chip. The resulting complex I/Q data are then streamed to a personal computer through a USB connection for further processing. The architecture provided by ADI, in which the Zynq is only used to collect the data and stream them to the personal computer on the one hand restricts the available bandwidth due to the USB bus, and prevents using fully the PS capability of the Zynq. In order to run GNU Radio on the PS and take advantage of the processing power as well as the huge communication bandwidth between PL and PS, we have ported the development framework provided by ADI to the `BR2_EXTERNAL` framework providing a homogeneous, fully consistent development framework. Hence, the latest release of the `buildroot` framework shall be used on the embedded software, including the latest packages such as the necessary extensions to GNU Radio needed to collect data from the AD9363 and process such data on the embedded board. Since the processing requiring most bandwidth is run on the PS, the resulting decimated stream becomes consistent with USB bandwidth when streamed to the personal computer.

In addition to providing the Buildroot framework to add custom software, including GNU Radio blocks, to the PlutoSDR, we provide the ability to tune the bitstream configuring the Zynq PL with custom processing blocks, such as a sound output, making the PlutoSDR a fully autonomous, embedded radiofrequency transceiver.

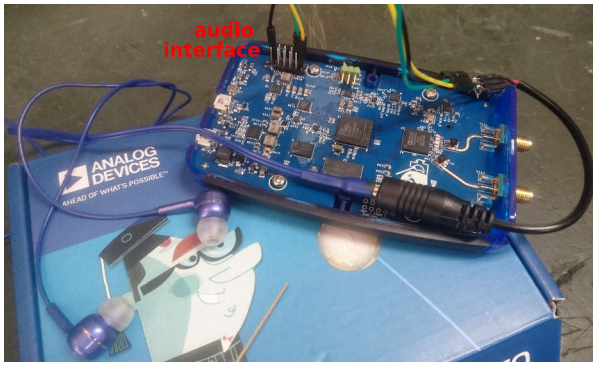

Figure 1: Adding an audio interface to the PlutoSDR through EMIO driven by a PWM added to the original bitstream. In this example, the PWM datapath is independent of the data stream coming from the AD9363 radiofrequency frontend providing the raw I/Q signals needed to demodulate a broadcast FM station and playing sound on the headphones.

2 Experimental setup

ADI's development framework provided at `github.com/analogdevicesinc/plutosdr-fw` uses a `Makefile` configuration to run multiple tools in order to generate all the files needed to generate the embedded firmware. The kernel is compiled out of the buildroot environment, which is itself a version frozen at the time of the release. While the generation of the image is functional, its long term evolution is dependent on porting the updates to the current kernel to the latest Linux release. Furthermore, the version of buildroot is frozen to a version not yet supporting GNU Radio.

Various efforts aimed at leveraging the processing power of the Zynq [1] include running a web server on the embedded target (`github.com/unixpunk/PlutoWeb`) or updating the PL bitstream

`github.com/timcardenuto/testPlutoSDR`. All these projects still rely on the official ADI framework whose long term stability is questionable since buildroot and the linux kernel will keep on evolving. In order to avoid freezing features of a given `buildroot` release, we have extracted the modifications brought by ADI to buildroot and included them in an external branch designed to be merged with the latest buildroot release as provided through the `BR2_EXTERNAL` mechanism.

Furthermore, thanks to the availability of the PlutoSDR HDL firmware, a bitstream can be generated to configure the PL with custom functionalitites. We here promote the compatibility with the OscimpDigital PL/PS co-design framework as documented at `https://github.com/oscimp/oscimpDigital/tree/master/doc/tutorials/plutosdr/`.

3 Results

In order to demonstrate the embedded signal processing using GNU Radio, we tune the AD9363 (whose configuration was updated [2] to match an AD9364 to allow reaching the 100 MHz commercial FM broadcast band) to a broadcast FM station, stream the data to an embedded command-line python script generated from GNU Radio Companion, and transfer the resulting audio stream to the personal computer through the ZeroMQ framework. The personal computer then sends the stream to the sound card to assess the demodulation quality.

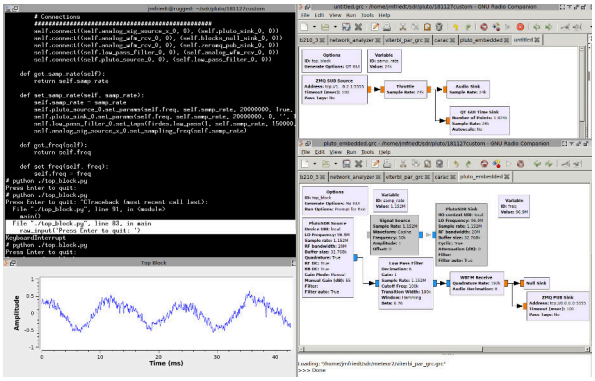

Figure 2: Flowgraphs running on the Zynq target (bottom right) and on the host PC (top-right), streaming the FM signal demodulated on the PS of the target (top-left) and using the host as a sound card.

Fig. 2 exhibits the flowcharts running on the embedded target and the host computer, as well as the resulting oscilloscope output.

Beyond allowing for processing datastreams at the PS side of the Zynq, accessing the bitstream generated to configure the PL allows for including basic preliminary processing steps at the FPGA level. Fig. 3 exhibits the initial FPGA configuration provided by ADI to fetch data from the AD9363 and stream them to the PS memory through the AXI DMA interface. While the basic design only includes FIR decimator and interpolator blocks between the AD936{1,3} block and the AXI DMA, any additional processing block complying with the interfaces might be included to preprocess the data at the FPGA level, hence removing the bandwidth limitation introduced by the PL to PS communication.

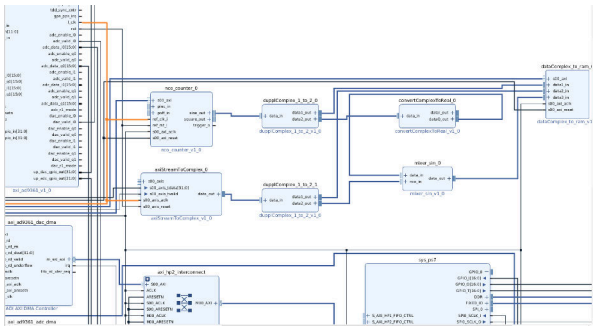

Figure 3: FPGA processing chain, from AD9363 to PS memory through the Direct Memory Access (DMA) AXI stream, and collecting the AXI stream to feed the custom processing chain here made of an NCO and a mixer to demonstrate an additional frequency transposition.

4 Conclusion

A fully functionnal extension to buildroot supporting the PlutoSDR to run embedded processing software is proposed. All development files are released at `github.com/oscimp/PlutoSDR`. The demonstration of the operational framework is achieved by streaming the sound demodulated from the incoming commercial broadcast FM radiofrequency signal onboard the Zynq processor.

References

[1] *PlutoSDR: enable 2nd CPU core for better performance* at `www.reddit.com/r/RTLSDR/comments/7h2hh2/plutosdr_enable_2nd_cpu_core_for_better/`

[2] *Updating to the AD9364* at `wiki.analog.com/university/tools/pluto/users/customizing`