The Xilinx Zynq: A Modern System on Chip for Software Defined Radios

Stefan Scholl, DC9ST
Amateur Radio Research Group and Microelectronic Systems Design Research Group
University of Kaiserslautern, Germany

***Abstract*—Software defined radios can be implemented on general purpose processors (CPUs), e.g. based on a PC. A processor offers high flexibility: It can not only be used to process the data samples, but also to control receiver functions, display a waterfall or run demodulation software. However, processors can only handle signals of limited bandwidth due to their comparatively low processing speed. For signals of high bandwidth the SDR algorithms have to be implemented as custom designed digital circuits on an FPGA chip. An FPGA provides a very high processing speed, but also lacks flexibility and user interfaces. Recently the FPGA manufacturer Xilinx has introduced a hybrid system on chip called Zynq, that combines both approaches. It features a dual ARM Cortex-A9 processor and an FPGA, that offer the flexibility of a processor with the processing speed of an FPGA on a single chip. The Zynq is therefore very interesting for use in SDRs. In this paper the application of the Zynq and its evaluation board (Zedboard) will be discussed. As an example, a direct sampling receiver has been implemented on the Zedboard using a high-speed 16 bit ADC with 250 Msps.**

I. Hardware Platforms for SDR

Software defined radios (SDR) rely on software and digital signal processing to receive radio signals. SDR algorithms always run on a hardware platform: on a processor, such as a PC or microprocessor, or implemented as a digital circuit in a microchip, like an FPGA. These hardware platforms differ in processing speed, power consumption, size, flexibility and ease of design.

A. Commonly used Hardware

Today, very different platforms are available on the market, of which some examples are shown in Table I. A Raspberry PI with a Funcube Dongle SDR (FCD) with reduced sampling rate of 48 kHz is a small embedded portable device, but with very limited processing speed. An Intel Core i7 laptop processor has increased speed and can deal with a RTL DVB stick with 2 MHz bandwidth, but also requires more power and is large in size. SDRs with larger bandwidths such as the bladeRF (28 MHz) or even a direct sampling receiver like the Perseus SDR (40 MHz) require much more processing speed, that cannot be provided by a processor. In such a case FPGAs are used for high speed processing of the digitized RF signals.

B. Requirements

The required processing speed of SDR hardware is not only, but heavily, dependent on the signal bandwidth. The larger the bandwidth, and therefore the sampling rate is, the more data has to be processed. High performance direct sampling receivers often sample the analog signal with a speed of more than 50 Msps. This creates a huge requirement for processing speed - a fact that will become more apparent as direct sampling, i.e. sampling analog signals directly after the antenna, becomes common in SDR.

TABLE I. Examples of different SDR platforms

SDR	Bandwidth	Processing Speed	Hardware
Raspberry PI + Funcube Dongle	48kHz	low	ARM Processor
Laptop + RTL Stick	2MHz	medium	Intel Processor
BladeRF	28MHz	high	FPGA + ARM
Perseus SDR	40MHz	high	FPGA + μC

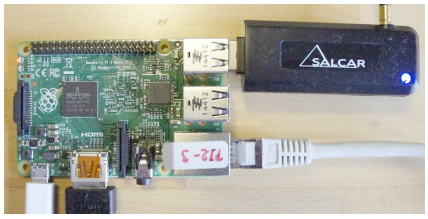

Fig. 1. A simple SDR receiver: Raspberry Pi with DVB RTL stick for low bandwidth processing

On the other hand flexibility is an issue in modern SDR receivers. One example is demodulation. Numerous digital modes have been developed by radio amateurs and also in commercial applications switching between modulation type, error correction and source coding is required.

C. Processor vs FPGA

A processor is an electronic circuit, that executes software. This execution is done in a serial way, instruction by instruction, and is therefore inherently slow and not suitable for large bandwidth processing in high end SDRs. An exception are multi-core processors, that exhibit parallelism to some extend. Especially in graphics cards powerful multi-core processors (GPUs) can be found, that are well-suited for signal processing. However, many processors have a high power consumption and in case of a PC (including graphics cards) they require large space. The advantage of the processor is, that it is very flexible, because the tasks it fulfils can be changed easily by executing a different software. Moreover, it is often possible to run an operating system with the advantage of easy access to user interfaces like a monitor, keyboard, mouse and providing access to hard disks for storage or an internet connection. In SDR applications, processors are very good for:

- demodulation of different modes
- providing a graphical user interface (GUI)
- controlling the receiver function
- recording and storing signals
- providing internet access

FPGAs are fundamentally different from processors. An FPGA is a microchip, that contains a large number of elements, that can be used to form a digital circuit, including adders, multipliers, multiplexers, registers, etc. For every algorithm, that is to be "executed" on the FPGA, a specialized digital circuit has to be designed (hardware implementation) and loaded onto the FPGA (configuration). In principle this configuration consists of the wiring between the FPGA elements to form the desired digital circuit. The FPGA can be configured over and over again, called reconfiguration, such that the circuit can later be modified to remove bugs or add extensions. This makes the FPGA a comparably cheap microchip, that can be designed by everyone having some knowledge in digital design.

The big advantage of FPGAs is that they can process data highly parallel. Therefore they are very fast and can easily process up to several hundred Msps of data. The achievable speed-ups over processors is often 100x - 1000x, even though their clock frequency is often much slower (in the order of some hundred MHz). Since the circuit in an FPGA is customly designed, the bit widths of data, memories and arithmetic units can be chosen application specific, which is very efficient. If, e.g., a bit width of 6 bits is considered sufficient for a data, the processing units are laid out to use only these 6 bits, whereas in a processor the bit width is more or less restricted to a predefined number (e.g. 24, 32 or 64 bits).

Additionally FPGAs also have a low power consumption compared to (multi-core) processor systems with similar processing speed, such as GPUs graphic cards. The drawback of FPGAs is that the hardware implementation, i.e. circuit design, of algorithms requires good knowledge of hardware description languages, digital circuit design and experience with the design tools. Also the implementation of interfaces to the outside (monitor, keyboard, network connection) is difficult. In SDR applications FPGAs are very good for high speed, repeatedly occurring calculations, like

- mixing and down conversion
- filtering
- high-speed FFT
- parallel demodulation of a large number of signals

Processors and FPGAs both have advantages and disadvantages. **In a SDR a wide range of different tasks have to be performed with different speed, that is difficult to achieve with a single processor or a single FPGA. So the ideal hardware for SDR is a combination of both.** And indeed high end SDRs already follow this conclusion.

II. The Zynq System on Chip

Recent development in microelectronic has led to a new kind of microchips, that combine an FPGA and a processor on a single chip. The FPGA manufacturer Xilinx has presented the Zynq Device, whereas its competitor Altera introduced the Altera SoC series. Both chips are very similar in system structure and performance. We focus on the Xilinx Zynq in the following, but many details are also applicable to Altera SoCs.

These new hybrid systems belong to the group of Systems on Chips (SoCs). A SoC is composed of several very different components like processors, memories, I/O blocks, accelerators etc. on a single chip. This reduces cost and size and provides close and fast interaction between the components. Microcontrollers belong to the group of SoCs, as well as the widely known Raspberry PI.

The Zynq is also a SoC, since it contains a dual ARM A9 core, an FPGA and additional blocks for different I/O (e.g. DDR3 memory interface, USB, SD card controller). There are several thousand possible internal connections between the processor and the FPGA, which are programmable and allow fast transmission of data. This is a big advantage over a separate combination of FPGA and processor on a PCB, where the number of possible connections is small and can hardly be changed after PCB design. The Zynq comes in six different sizes (Z-7010 to Z-7100). All contain a dual ARM A9, but the FPGA part has largely different sizes.

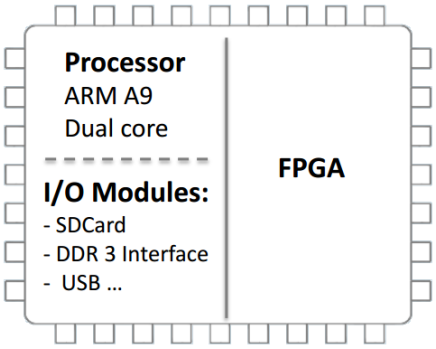

Fig. 2. Overview of the Zynq system on chip

Programming of the Zynq involves two tasks: Programming the processor and designing the digital circuit for the FPGA. These two tasks can be done nearly independently by different tools.

Programming the Processors

The processor can be programmed either bare metal or using an operating system. Bare metal refers to programming the ARM processor directly in C without any operating system using e.g. the Xilinx SDK tool. However it is strongly recommended to use an operating system like Linux, which hides many details of the processor by using a hardware abstraction layer (HAL) and provides easy access to interfaces as well as standard Linux software.

As an operating system a Linux distribution (e.g. Ubuntu/Linaro) can be installed on a SD card, which also serves as a file system. Access to the processor can be gained through the network interface (e.g. via SSH or RDP) or by directly attaching monitor, keyboard and mouse (use as a standalone system). Then standard software can be used

(provided from the Linux package manager or compiled by hand). Software like SciLab (mathematical toolbox) or the powerful demodulation software Fldigi can be installed directly via package manager. Programming can be done using many languages, like C/C++, Qt, Python, Java etc. As an editor for programming the QtCreator can be used, which can directly run on the Zynq without the need for cumbersome cross compilation.

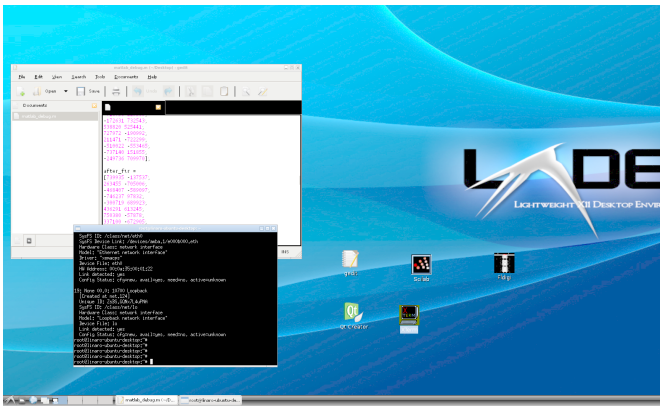

Fig. 3. Linux Desktop running on the Zynq

FPGA Design

The more challenging part is the hardware design for the FPGA. Here, Xilinx provides a design tool called "Vivado". The basic functionality of Vivado is free of cost, an additional license for the debugging functionalities (Vivado Logic Analyzer) is included with the Zedboard evaluation board. These tools are sufficient for most of the designs for the Zynq. SDR design can be done by using the Vivado Block Design, which is a top level schematic, that connects different blocks. The blocks contain the main functionality and can be designed in different ways. One is using a hardware description language like VHDL or Verilog. The other option is to rely on high level synthesis, like Vivado HLS or Matlab based tools (System Generator, HDL Coder), that basically try to convert C or Matlab code into a digital circuit.

It is worth to mention, that Xilinx offers some predefined hardware components, that are very useful to build an SDR: A sine signal generator based on direct digital synthesis (DDS) is available, that can be parametrized [1]. Frequency range, resolution, bit widths, clock speed and spurious-free dynamic range (SFDR) can be chosen. SFDRs of 150 dB can be easily obtained. Another useful hardware block is the FIR filter compiler, that allows for easy implementation of FIR filters [2]. Many hardware parameters can be chosen, the filter coefficient design itself can be done using e.g. Matlab.

Designing the hardware for FPGA requires good knowledge and experience in hardware design and digital circuits. Furthermore, experience shows that the Vivado tools sometimes seem to be quite immature and unintuitive to use.

How the interconnect works

As mentioned above, the Zynq provides several thousand interconnections between the processor and the FPGA for the

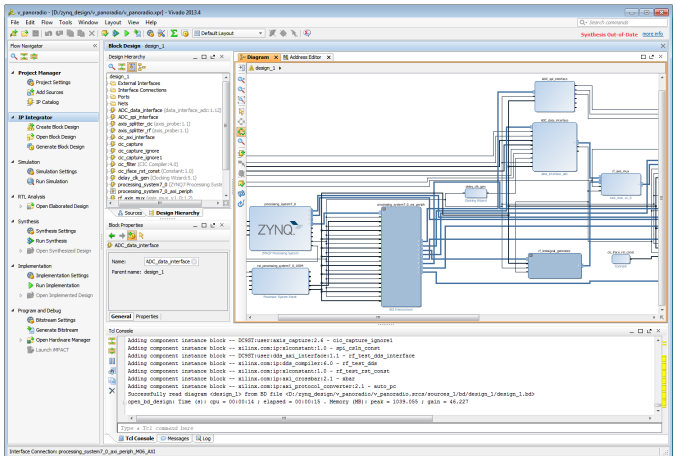

Fig. 4. The Vivado design tool, here: block desinger

exchange of data. It is worth to mention a few words on how these interconnect works.

The interconnect is a memory mapped interface. That means that a slice of 32 bit of the processor's memory gets virtually connected to a 32 bit register in the FPGA. The processor can write data to this special memory slice, that is then immediately transferred to the FPGA registers. Or it can read the registers in the FPGA by reading data from the memory. From an FPGA point of view data transfer is done by just reading and writing to the 32 bit registers.

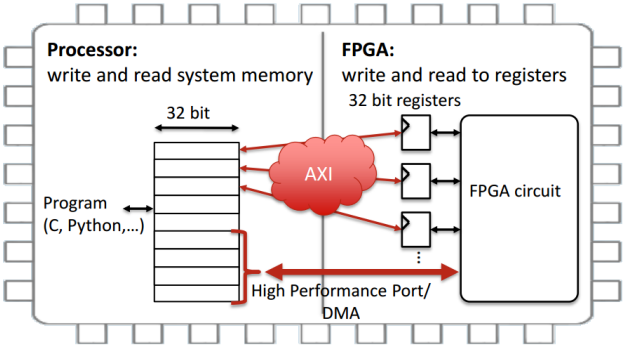

Fig. 5. Memory mapped interconnect between processor and FPGA

Although moving data sounds not too complicated and is for a programmer, it involves a quite complex process based on a standard called AXI, which is fortunately mostly hidden to the user.

If a large amount of data is to be transferred, direct memory access (DMA) can be used. This allows much faster transmission of data by using the Zynq's high performance (HP) or accelerator coherency (ACP) port.

Reconfiguration

One additional advantage of Zynq comes into play if reconfiguration is used. As already mentioned above, the digital circuits in an FPGA can be changed. When the digital circuit is designed with Xilinx Vivado tools, the design is stored in a file (.bit or sometimes .bin), called the bit stream. After power on the FPGA is empty and the bit stream for a the designed

circuit can be loaded onto the chip to make the FPGA do its job. It is possible to design different hardware circuits in advance and store them in different bit stream files. These bit streams can be exchanged *during runtime*. This process is called reconfiguration. Reconfiguration takes just a few ms to completely change the circuit in the FPGA.

A standalone FPGA needs quite a large overhead for reconfiguration. The design files first have to be transferred somehow to an additional memory on the PCB. And afterwards, whenever a reconfiguration had been triggered transferred again to the FPGA, making additional logic (e.g. a microcontroller) necessary. The Zynq eliminates this overhead since the bit stream files can be stored in the Linux file system on the SD Card or on a remote computer and are transferred to the FPGA by using the processor. In Linux the whole process of reconfiguration reduces to the execution of a single command:

```
cat new_bitstream.bin > /dev/xdevcfg
```

This opens up new possibilities for SDRs, since the high speed digital circuits can be changed during operation. They can be adapted to the required radio operation modes, to the spectral environment or the used transmission modes, that is also considered in modern radio technologies like cognitive and adaptive radio.

III. The Zedboard

The Zedboard is a low-cost evaluation board for the Xilinx Zynq device. It contains many peripherals that unleash the power of the Zynq, such as power supplies, memory and many interfaces.

In detail the Zedboard features:

- 512 MB DDR3 Memory
- Gigabit Ethernet
- USB OTG and USB UART
- HDMI interface
- VGA connector
- OLED display
- SD card slot
- GPIO pins, buttons and LEDs
- FMC connector
- Audio interface ADAU 1761
- onboard clock generators

These peripherals make the Zedboard a fully standalone system. Monitor, mouse, keyboard and headphone can be connected, as well as an internet connection established.

The Zedboard is ideal for experimenters, because it provides a lot of possibilities. It is an ideal platform for many different applications like image processing, control systems, acceleration of algorithms - and SDRs. Moreover, there is a large online community at www.zedboard.org. Tutorials, example projects and a forum are available to support the Zedboard beginner. Also the schematics of the board have been published and can be downloaded at no cost.

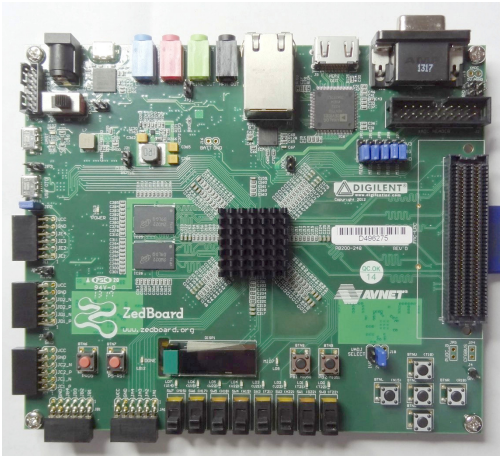

Fig. 6. The Zedboard

Since the Zedboard itself does not contain any ADC or DAC suitable for SDR, a radio module needs to be connected. The FMC connector is ideal for that. Many companies like Analog Devices, Texas Instruments or Linear Technology offer daughter board with an FMC interface. These boards can be easily connected to the Zedboard. An advantage of the Zedboard over many other SDR platform is, that the radio module is not fixed. It can be chosen according to the user's requirements and available budget. It is easy to change the FMC board and create a radio with a completely new ADC/DAC system.

Besides the "classical" Zedboard more Zynq boards appeared on the market. The MicroZed and the Zybo should be mentioned, since they are much cheaper and offer similar functionality. But it needs to be pointed out that these boards do not have a full FMC connector. This makes it harder to attach ADCs and DACs to create a full SDR platform.

IV. Examples of Modern SDRs using Zynq

As mentioned before, the Zynq fits very well to the requirements of SDR applications. And that is the reason, why it is already used in modern SDRs, of which some will be presented in the following.

Red Pitaya

The Red Pitaya has been developed as a measurement device (oscilloscope, spectrum analyzer, signal generator) [3]. It features a dual channel input with a 14 bit ADC and a dual channel output with a 14 bit DAC. It is a direct sampling receiver with 125 Msps and captures signals with up to 50 MHz. For signal processing it relies on the Zynq 7010, which is the smallest device of the Zynq series. Linux is running on the chip, which allows to connect a PC or tablet via LAN and provides interfaces to C++, Matlab or Labview. It is claimed to be open source, but this is so far only true for the software part. Although designed for measurement applications, the Red Pitaya has been converted into a full SDR system [4].

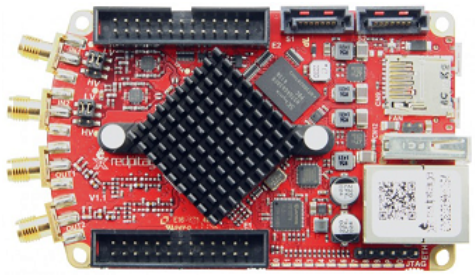

Fig. 7. Red Pitaya: Zynq based ADC/DAC board

Zepto SDR

The Zepto SDR has been developed by Nutaq and consists of a Zedboard combined with a FMC extension (Radio420S FMC) [5]. It operates in the frequency band from 0.3 to 3 GHz using 12 bit ADC and DAC converters. The maximum processing bandwidth is 28 MHz. It is proposed to run GNU radio on the Zynq.

USRP E310

The USRP E310 is a device of the Embedded Series of the well-known USRPs from Ettus [6]. Thanks to the Zynq 7020 (same size as in the Zedboard) it is a standalone device. The dual channel analog frontend supports 70 Mhz to 6 GHz with a maximum bandwidth of 56 MHz.

R2T2

The R2T2 is a direct sampling transceiver with two channels for receiving and two for transmitting [7]. It features two 14 bit ADCs with 125 Msps and 14 bit DACs with 250 Msps. It provides similar processing features as the Zedboard: A Zynq 7020 with DDR3 memory, Gigabit Ethernet, HDMI, USB, SD Card and the Audio Codec ADAU1791. Via web interface up to 8 users can use the SDR remotely [7].

V. The Panoradio SDR: A Tech Demo for a High-Speed Direct Sampling Receiver

A. Overview

To show today's possibilities for an experimenter, a tech demo, the *Panoradio SDR* has been developed. It features a Zedboard with an attached FMC card with the high-speed AD9467 ADC, that samples analog signals with 16 bit resolution and 250 Msps [8]. Because of its accuracy and high sample rate, it can pick up signals directly after the antenna without any analog down conversion, working as a direct sampling receiver. Due to the high sampling rate, a bandwidth of up to 125 MHz can be picked up and displayed instantaneously (If an anti-aliasing filter is required, this bandwidth reduces to approximately 100 MHz).

B. Features

The Panoradio provides three zoomable panorama or waterfall displays showing up to 100 MHz of bandwidth, including proper anti aliasing filtering. The panorama displays can be independently zoomed down to a bandwidth of 6 kHz and thus deliver a minimum resolution down to 7 Hz. The three

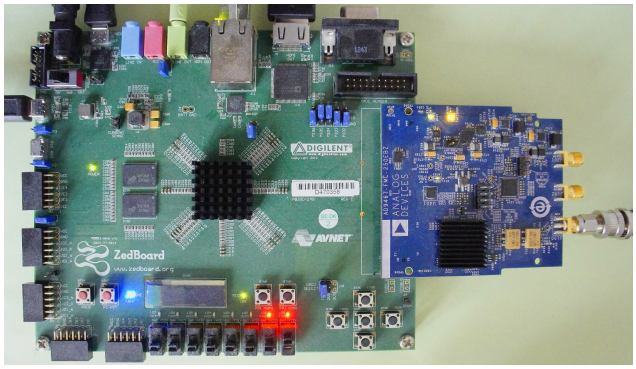

Fig. 8. The basic Panoradio hardware: Zedboard + AD9467 FMC board

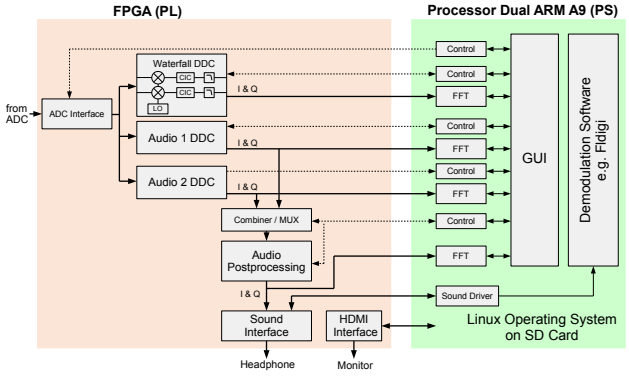

Fig. 9. Panoradio block diagram

panorama displays allow the observation of different bands in a very large frequency range simultaneously. For example it is easy to watch signals in the 80m, 10m and 4m band at the same time. With the operating system running on the Zedboard, the software for controlling the radio directly runs on the receiver and thus makes it a standalone embedded system, where also mouse, monitor and keyboard can be attached directly. In addition to that demodulation software, such as the popular Fldigi [9], can also run directly on the radio.

The Panoradio can also be used as FFT spectrum analyzer. Its frequency range easily allows the examination of short wave signals. Since the ADC's SFDR is up to 100 dBFS [8] spurs are expected to be very low.

In addition to the panorama display two independent audio receivers are available. Each audio receiver covers a bandwidth of 22 kHz (optional 6 or 3 kHz), whose signal spectrum is displayed in additional waterfall plots. The audio outputs can directly be multiplexed or mixed to the headphone output of the Zedboard's audio codec and the processor's audio interface (and shows up as a sound card device in Linux). Also the audio receivers can be tuned to any frequency between 0 and 100 MHz independently and instantaneously.

C. Description

The structure of the Panoradio is shown in Fig. 9. The design is separated into high-speed processing, which is done on the FPGA and software processing which is done on the processor part of the Zynq.

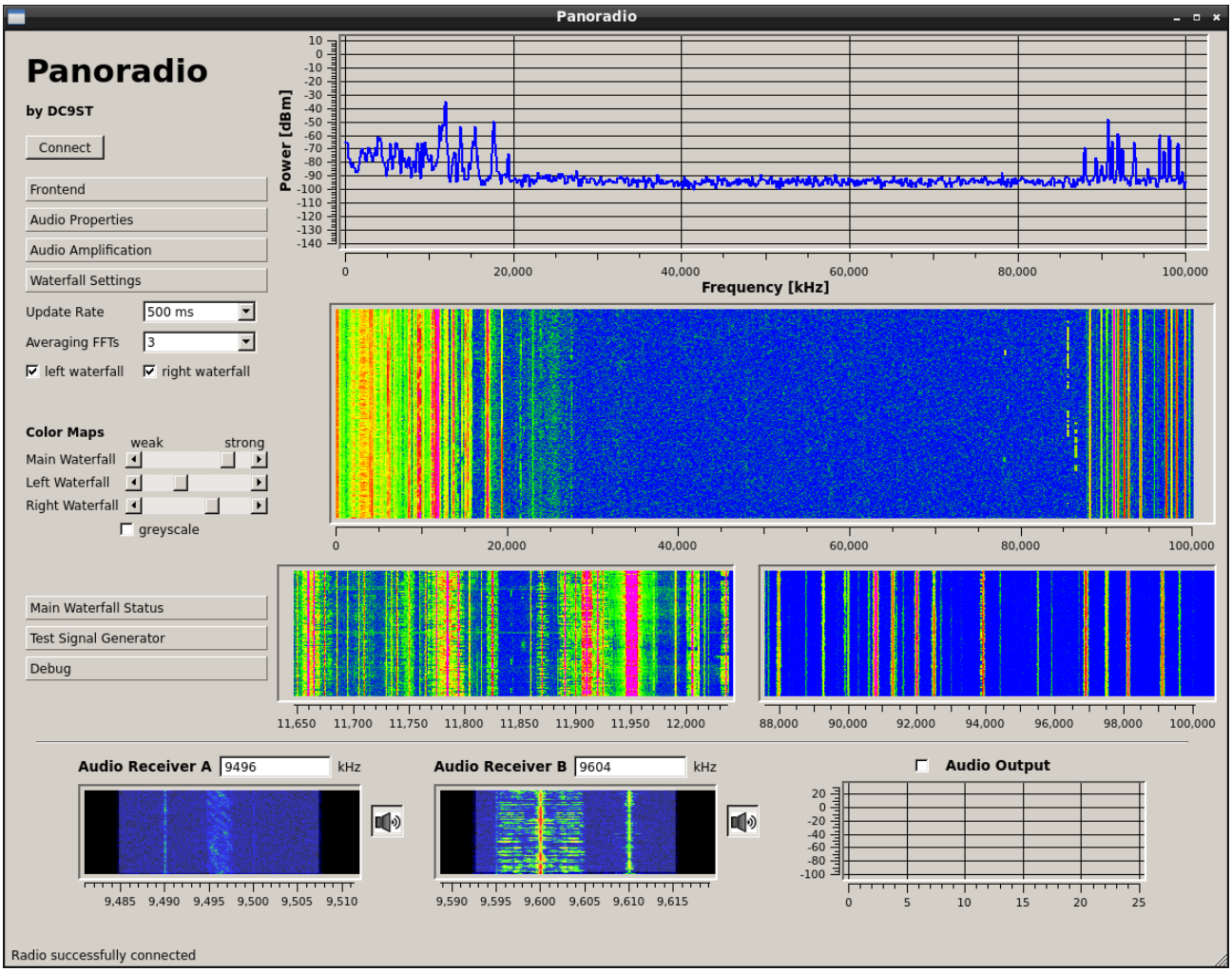

Fig. 10. Panoradio software and GUI: The main waterfall shows the full span from 0 to 100 MHz. Short wave signals occupy only a fraction at the left of the complete span. Signals above 88 MHz are FM broadcast stations and the sporadic emissions below 88 MHz belong to emergency services. The small left waterfall shows the 15 m broadcast band. The small right waterfall zooms on the VHF broadcasting signals.

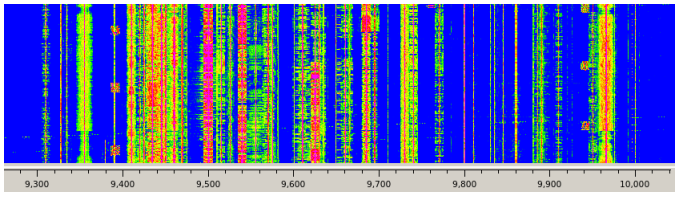

Fig. 11. Panoradio's spectrum showing signals the 31m band.

FPGA Processing: Data samples from the ADC are put into the FPGA via the ADC interface module. It performs some ADC tests, takes care of line delays and converts the differential double data rate signals to single-ended data with single data rate, that can be processed in the FPGA. The samples are then fed into three digital down conversion (DDC) blocks, one of the waterfall displays and two for the audio receivers. Each DDC consists of a complex multiplier as IQ mixer, DDS local oscillator plus a pair of CIC and FIR filters for decimation and bandwidth reduction. The waterfall DDC provides the zoomable panorama function. The zoom function is realized by rate programmable CIC filters. For all three waterfall displays only a single receiver is used, that is multiplexed between the displays. Its receiving frequency is then switched several times per second to each of the waterfall displays' center frequency. The audio DDCs are implemented with a output IQ bandwidth of 22 kHz. Afterwards, the audio data is further processed in the audio post processing block, which consists of filters for further bandwidth reduction (from 22 to 6 or 3 kHz) and a Weaver demodulator for SSB reception. Finally, an audio interface sends audio IQ data to the headphone out and to the processor.

Software Processing: The processor runs a Linaro/Ubuntu Linux operating system on a SD card. Application programming has been done in C++. Some additional libraries have been used: Qt plus Qwt [10] for the graphical user interface and FFTW [11] for fast FFT processing. The tasks of the software and the processor are mainly to control the functionality of the FPGA design and to provide an interface to the user. FPGA interfacing includes: reading IQ data from the DDCs for the FFT, initializing the ADC and audio interface, setting the DDC frequencies and filter properties, loading the FPGA design at system start-up, determining amplifications in ADC

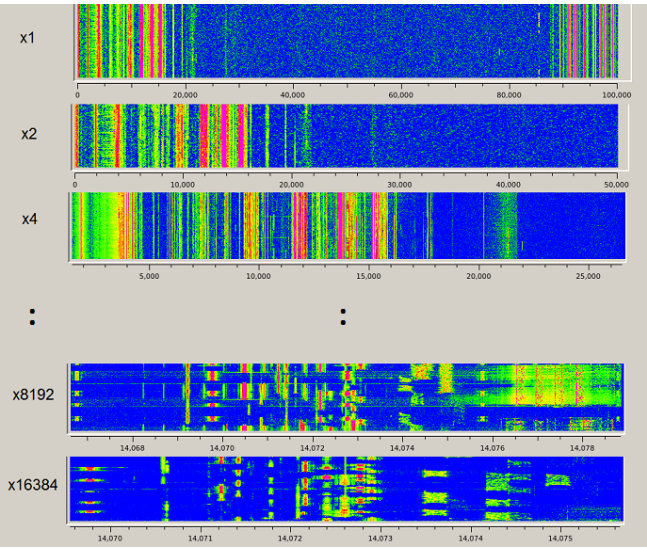

Fig. 12. The zoom functionality allows to inspect signals of small bandwidth, here shown for digital signals in the 20m amateur radio band, including some PSK31 transmissions.

and audio interface. Furthermore, the FFT calculations (including windowing, averaging, decimation etc.) for the waterfall displays are done in software with the processor. Additionally, the processor provides a Ethernet and UART as interface to the user, shows the GUI and handles user inputs.

D. Future Possibilities

Since the analog interface of the AD9467 can operate at frequencies beyond 1 GHz, undersampling can be used to receive frequencies far above the Nyquist frequency of 125 MHz, if proper analog filtering is applied. This provides new opportunities for direct sampling of much higher frequencies as considered today in amateur radio. However phase noise degradation has to be taken into account.

The Panoradio is still work in progress and is continuously developed further. More information can be found on the project website **www.panoradio-sdr.de**.

VI. Conclusion

High-performance SDRs have challenging demands on their underlying hardware platform. The hardware is required to provide a high processing speed in combination with flexibility and user interfaces. Modern SoCs, such as the Xilinx Zynq and the Altera SoC series offer both features on a single chip. In fact, the Zynq is already used in several SDR projects and with the availability of the affordable Zedboard it is also interesting for experimenters. As a tech demo the Panoradio SDR has been presented. It is a 250 Msps direct sampling receiver based on the Zedboard and shows the capability of these new devices.

References

[1] Xilinx, *DDS Compiler v6.0 LogiCORE IP Product Guide*, PG141 ed., June 2015.

[2] ——, *FIR Compiler v7.2 LogiCORE IP Product Guide*, PG149 ed., June 2015.

[3] (2015, August) Red Pitaya Website. [Online]. Available: http://redpitaya.com/

[4] P. Demin. (2016, 02) http://pavel-demin.github.io/red-pitaya-notes/.

[5] Nutaq, *Zepto Software Defined Radio Datasheet*. [Online]. Available: http://nutaq.com/sites/default/files/zeptoSDR-datasheet-lowres.pdf

[6] (2015, August) Ettus E310 Product Website. [Online]. Available: http://www.ettus.com/product/details/E310-KIT

[7] (2016) www.r2t2.de.

[8] Analog Devices, *AD 9467 Datasheet*, Rev D ed.

[9] (2016) fldigi: Ham radio digital modem application. [Online]. Available: https://sourceforge.net/projects/fldigi/

[10] U. Rathmann. (2015, August) Qwt - Qt Widgets for Technical Applications. [Online]. Available: http://qwt.sourceforge.net/

[11] M. Frigo and S. Johnson, "FFTW," MIT, Tech. Rep. 3.3.3, November 2012.