ATCZ175 InterOP PROJECT

SDR Interference Emulator Essentials

SIX Research Centre

Brno University of Technology

31.12.2020

Table of Contents

Scope of the document

The goal of this document is to provide technical specification and documentation of the SDR Interference Emulator (SDR-IE) for interference analysis. The document is divided into eight main sections.

Section 1	defines the SDR-IE HW parameters and system properties.
Section 2	describes system architecture.
Section 3	insights into the FPGA firmware and describes all FPGA modules applied for signal processing.
Section 4	links to zip file containing necessary PCB and BOM files with necessary assembly instructions.
Section 5	characterizes ZYNQ operation system based on Petalinux.
Section 6	contains manual how to set Linux PC to work with SDR-IE.
Section 7	provides an overview of the SDR-IE project file structure including all necessary source files to run SDR-IE.
Section 8	describes the Gitlab project file structure.

1 SDR overview

The SDR Interference Emulator is simple but at same time a powerful and modular Software Defined Radio (SDR) platform developed by Brno University of Technology (BUT) Department of Radio Electronics (DREL) under project Interop ATCZ-175. The SDR-IE provides wireless communications designers an affordable SDR with unprecedented performance for developing communication systems. The SDR-IE refines user experience making SDR prototyping more accessible by delivering the optimum balance between simplicity and performance. It is ideal for a wide range of application areas and as an alternative for widespread SDR produced by Ettus research and National instruments (NI). The great benefit of SDR-IE is compatibility with Ettus/NI frontend transceivers such as WBX, SBX[1] and UBX[1].

[1] 2021 under development

Interface	**Power supply**	adapter 12V/4A mini power din
	PC conectivity	USB 2.0 Type B
		Ethernet 1G
	conectivity	SD
		Cannon DB 15
	RF connectors	TX1
		RX1
HW	**ADC**	250 MS/s
		14/16 bits
	DAC	250*** MS/s
		16 bits
	Tuning range	WBX/UBX: 50-2200/10-6000 MHz
	Architecture	homodyne*
	Standalone	✓ **
	one synthesis RX-TX	✓
	FPGA	ZYNQ Ultrascale+
	FPGA type	XCZU3EG-sfvc-784-1-e
	Slice LUTs used/free	18185/70560****
	Slice registers used/free	27981/141120****
	BRAM	31.5 MB
	DSP	3528
	processror	Quad-core ARM® Cortex™-A53
		MPCore™ up to 1.5GHz
		Dual-core ARM Cortex-R5
		MPCore™ up to 600MHz
SW		Partly OPEN SOURCE
		MATLAB
		C++
support		**InterOp**

*	frontend specific
**	depends on application
***	internal up-sampling to 500MS/s
****	fpga is modular can be replaced by larger module

Table 1 SDR-IE features

2 SDR architecture

The SDR-IE architecture is shown on following block diagram.

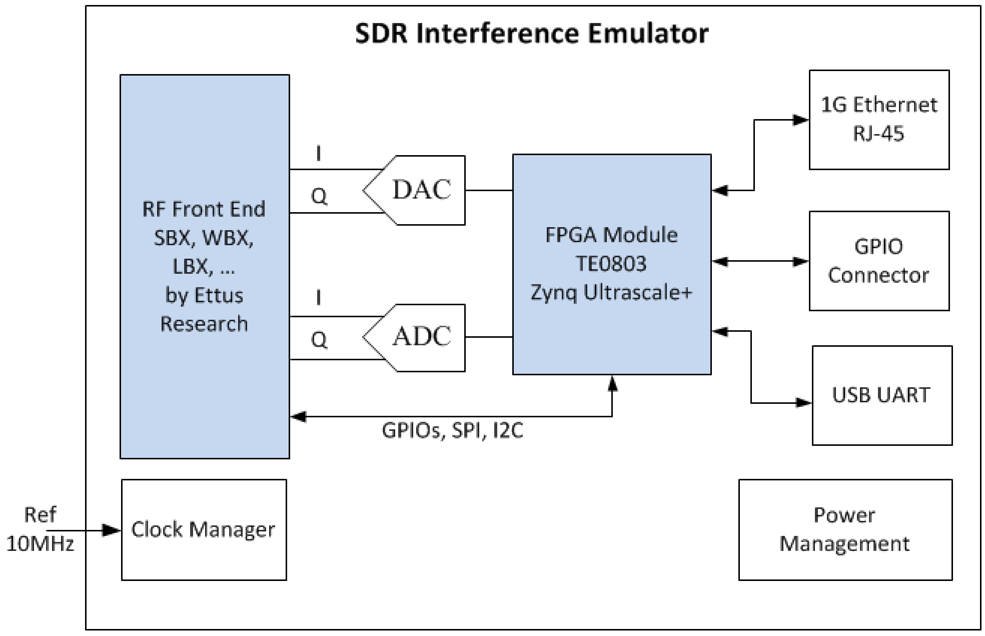
SDR Interference Emulator
RF Front End SBX, WBX, LBX, ... by Ettus Research
I
Q
DAC
I
Q
ADC
FPGA Module TE0803 Zynq Ultrascale+
1G Ethernet RJ-45
GPIO Connector
USB UART
GPIOs, SPI, I2C
Ref 10MHz
Clock Manager
Power Management

Figure 1 SDR-IE block scheme

SDR-IE modular system has two main modules, RF front-end and digital processing module. RF front-end module connectors on the main board are pin-compatible with National instruments or Ettus research RF front-ends such as WBX, SBX, UBX etc., also custom front-end can be used, such as RFID front-end. RF and sampling frequency can be referenced to an external 10MHz signal or use internal reference. Main A/D data converters – ADC and DAC, are on the main board, supporting several options depending on production variant. ADC supports up to 250Msps dual channel with 16bit resolution. Dual channel main DAC is 16bit, with 500Msps sampling frequency and 2x oversampling (data rate 250Msps). Auxiliary slow ADC and DAC channels are available for front-end control purposes, such as output power sensing or gain control. Additional front-end digital control from FPGA is available and fully configurable as GPIO, SPI or I2C.

Digital processing module is based on FPGA board TE0803 from Trenz Electronic, housing a Xilinx Zynq Ultrascale+ FPGA. A The SDR-IE uses a 1Gbit Ethernet as main connectivity, allowing for up to 20Msps full-duplex continuous streaming. Faster sample rates can be used if the processing is done on the FPGA or burst-mode can be used for short high speed record/replay functionality. On-board DDR memory can be used as sample buffer (up to 2x512MB), providing up to 500ms of sample storage at full sample rate (250Msps). Lower sample rates can be used to obtain longer recordings. USB serial port is provided for software development and debugging. GPIO connector allows simple connection to other measurement systems, providing triggering or other control signals. The SDR-IE platform is powered from single 9-15V power supply and can be powered directly from battery for field usage.

2.1 Firmware and software

The firmware three main parts:

1. **The FPGA firmware**: The description of internal FPGA connection in the FPGA chip and settings of the Zynq processor. The firmware is used for control high-speed and low-speed Analog-to-Digital (AD) and Digital-to-Analog (DA) converters and other necessary parts used

in the mother board or daughter board. As a development tool chain, the Xilinx Vivado software in version 2018.3 [3] was used.

2. **The Zynq processor firmware**: The Zynq firmware controls Ethernet communication and translation of the Telnet command to the FPGA. The software project is written in the C language and as a tool chain we used Vivado SDK software.
3. **The demo application software**: In the current state as a PC application we used a standard command line application. The SDR device is controlled by Telnet commands.

3 FPGA architecture

This section describes the overview schematics of FPGA architecture and provides insight into Baseband signal processing. The top-module of the FPGA project is depicted by Figure 2. The SDR-IE FPGA can be divided into four functional parts: receiver (RX), transmitter (TX), triggering, clocking and sync system and ZYNQ Block Design (BD) wrapper. Each block consists of several submodules described in sections below.

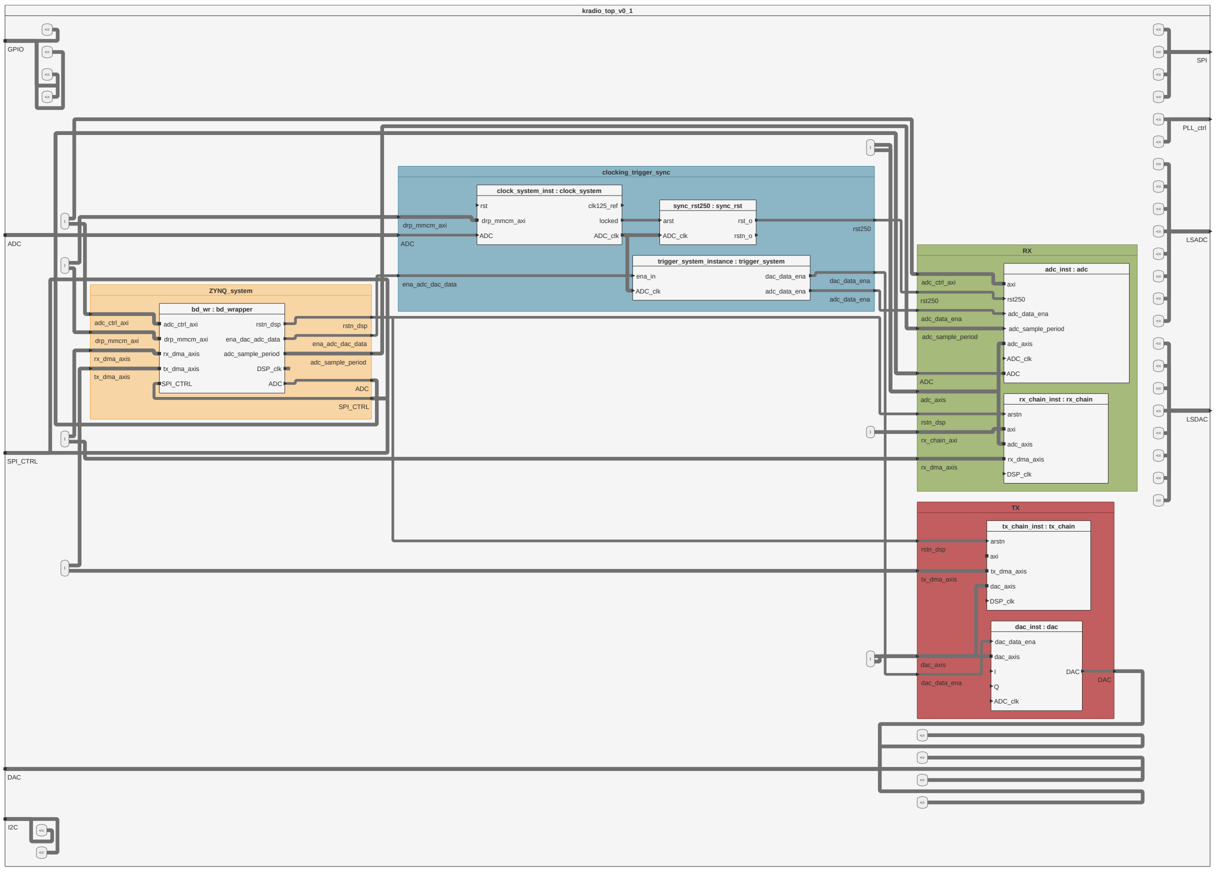

Figure 2 FPGA top module architecture

3.1 Analog to Digital Converter (ADC) interface (adc.sv)

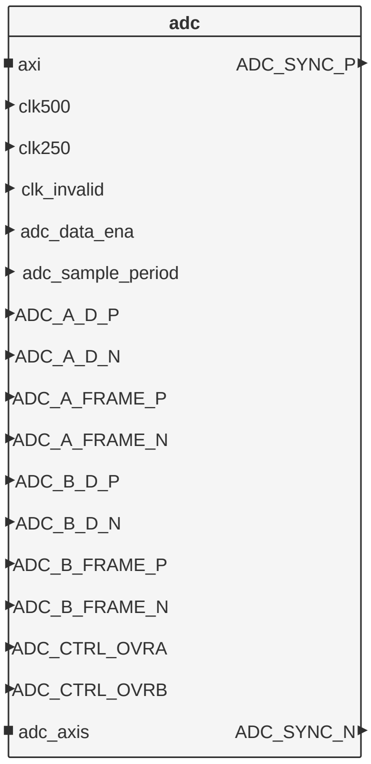

Figure 3 ADC module inputs and outputs

The module translates ADC raw data in QDR mode synchronized with ADC clock to output AXI stream bus in format of 16b Inphase (I) and 16b Quadrature (Q) component. The module contains input buffers, controlled delay blocks, input serial/deserializers, pattern checker submodule, AXI liter controller interface for configuration of entire blocks and also used for verification of data sample validity because of uncertainty of data samples (described in 3.7).

3.2 ADC AXI controller (adc_ctrl_axi.sv)

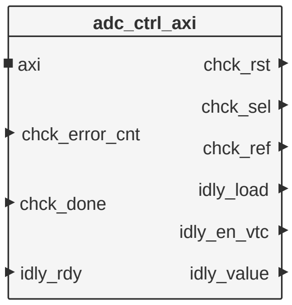

Figure 4 ADC AXI controller inputs and outputs

The module is driven by ZYNQ commands and translates AXI commands to drive ADC verification module and delay blocks for frame and data signals.

3.3 ADC pattern checker (adc_ptrn_checher.sv)

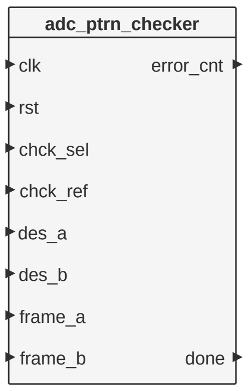

Figure 5 Pattern checker inputs and outputs

This module verifies received data with a pattern set by ADC AXI controller. The output is the number of error bits which is transmitted through ADC AXI controller into ZYNQ after checker signalizing the done flag.

3.4 RX Chain (rx_chain.sv)

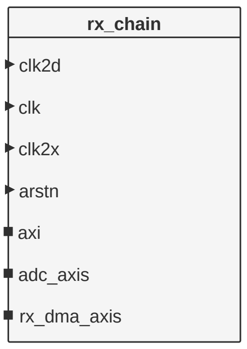

Figure 6 RX chain inputs and outputs

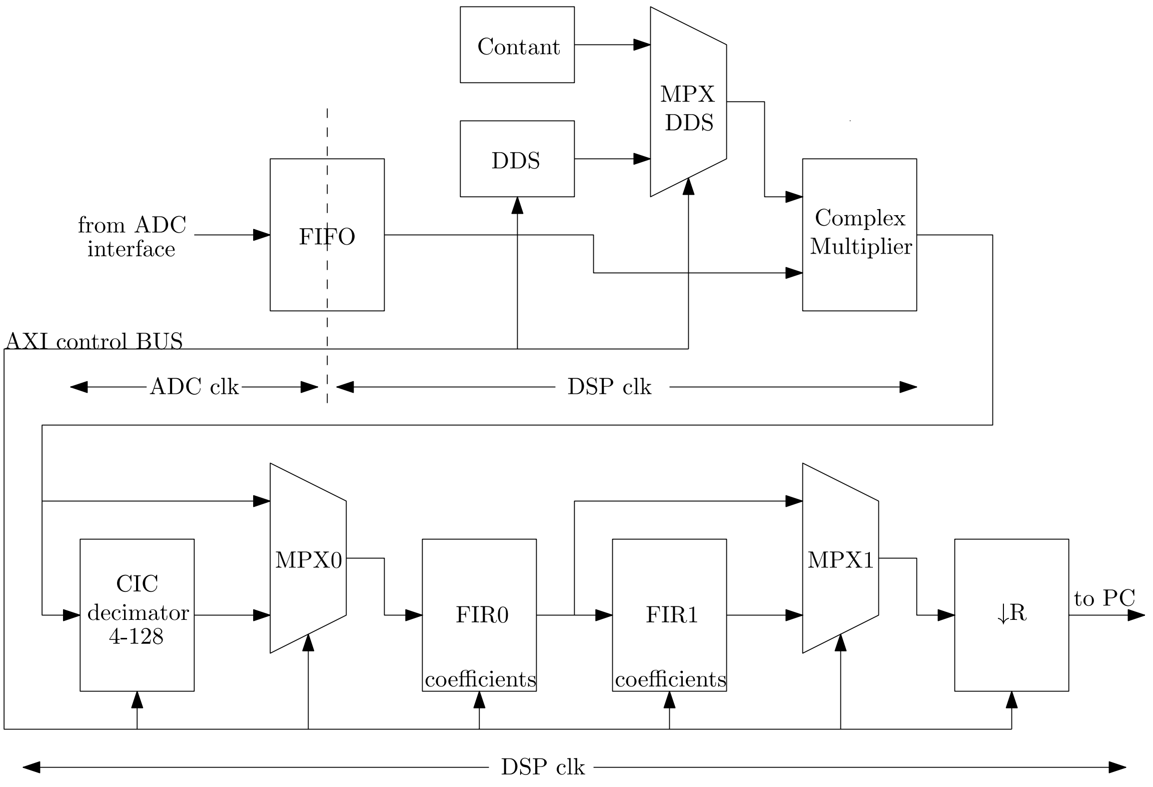

Figure 7 RX chain block scheme

This module synchronizes ADC data from ADC clock domain (clk125, clk125_ref, clk250, clk500) to Digital Signal Processing (DSP) clock domain (clk2d, clk, clk2x) and integrates several baseband processing blocks described by Figure 7. The ADC AXI stream signal inputs into synchronization block (adc_axis_sync.sv). Synchronized signal inputs into complex multiplier (cmpy_0.sv) to be multiplied by constant or sinewave generated by Direct Digital Synthesis (DDS) to process frequency shift (dds_compiler_0.sv). In next stage two configurable Cascaded Integrator–Comb (CIC) filters for I and Q path are implanted (cic_compiler_0.sv), then signal is filtered in first stage configurable Finite Impulse Response (FIR) filter (fir_compiler_0.sv) following by second stage configurable FIR filter (fir_compiler_1.sv). Filtered signal is downsampled by configurable downsampler (downsampler.sv). The last block of RX chain module is packetizer providing translation between RX chain module and Direct Memory Access (DMA) layer. Whole RX chain DSP is verified testbench (tb_rx_chain.sv) and compared with simulations in MATLAB (rx_chian.m, plot_results.m), for more details see 8.

3.5 TX chain (tx_chain.sv)

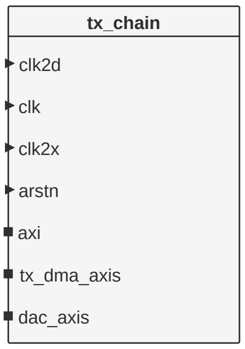

Figure 8 TX chain module inputs and outputs

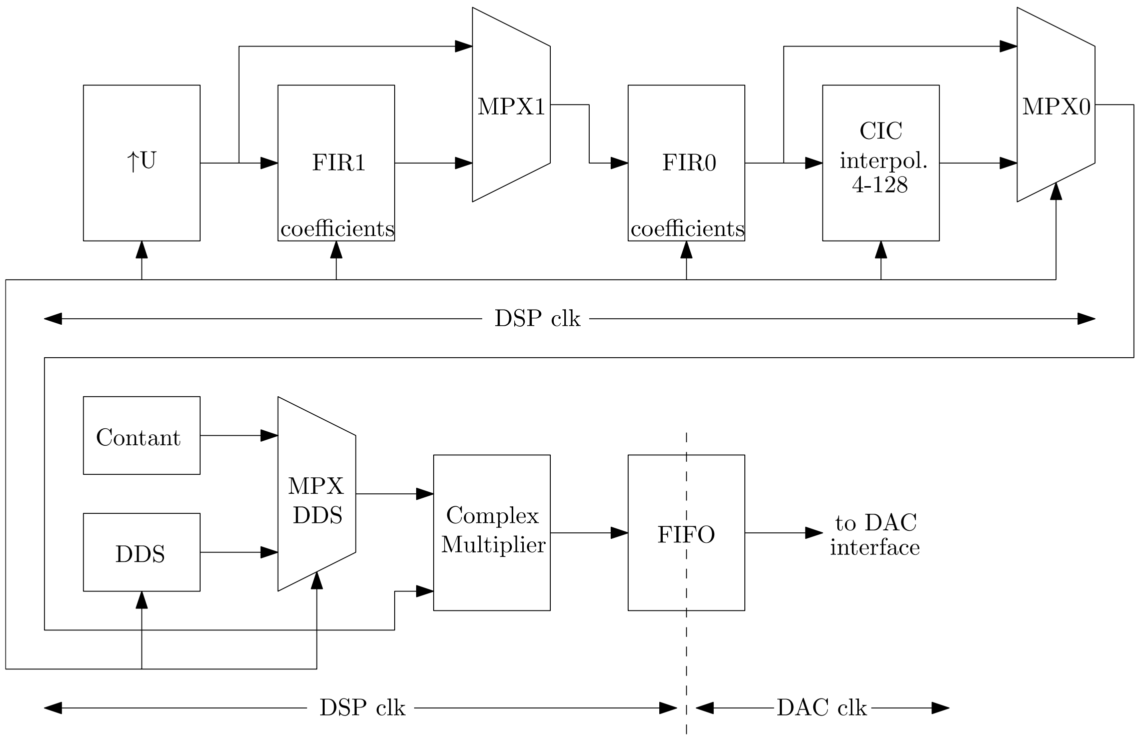

Figure 9 TX chain block scheme

This module proceeds transmitter DSP basically reversed to the RX chain. The data inputs from DMA layer to upsampler (upsampler.sv) with variable upsample factor. The upsampled signal is synchronized to 500MHz DSP clock domain (clk2x) and filtered by two stages of configurable FIR filters (fir_compiler_1.sv and fir_compiler_0.sv). The filtered signal is splitted into I and Q component and separately filtered by configurable CIC filters. Filtered signal inputs into complex multiplier (cmpy_0.sv) to be multiplied by constant or sinewave generated by DDS to process frequency shift (dds_compiler_0.sv). Then the signal is synchronized with clock DSP signal (clk). Whole TX chain DSP is verified testbench (tb_tx_chain.sv) and compared with simulations in MATLAB (tx_chian.m, plot_results.m), for more details see 8.

3.6 Digital to Analog Converter (DAC) interface (dac.sv)

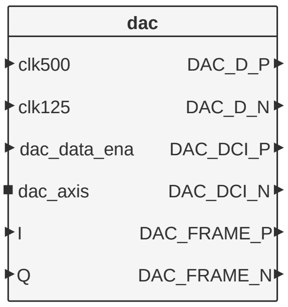

Figure 10 DAC interface module inputs and outputs

This module translates digital AXI stream bus in format of 16b I and 16b Q component data samples into form to be accepted by the DAC. The module contains output buffers, output serial/deserializers, axis data checker (axis_checker.sv) to verify the consistency of output data during development and input FIFO proceeding synchronization of data from DSP clock domain into ADC clock domain.

3.7 Clock system (clock_system.sv)

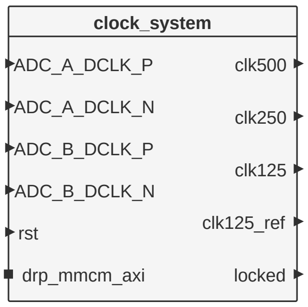

Figure 11 Clock system inputs and outputs

The base of this block is Mixed-Mode Clock Manager (MMCM) providing a method to dynamically change the clock output frequency, phase shift. The block is implemented because of uncertainty of data samples (DA0, DB0, …DA3, DB3) and frame signal (DAFRAMEM, DBFRAMEM, DAFRAMEP, DBFRAMEP) synchronization. The effect is caused by different length of data and frame paths resulting in different delay between signals. However, the ADC interface contains the configurable delay blocks, the uncertainty is introduced. Thus, the phase control of the frame signal is necessary. The clock system block is driven by ZYNQ system in order to reach zero error bits indicated by ADC data checker output. The data-frame signaling is depicted by Figure 12.

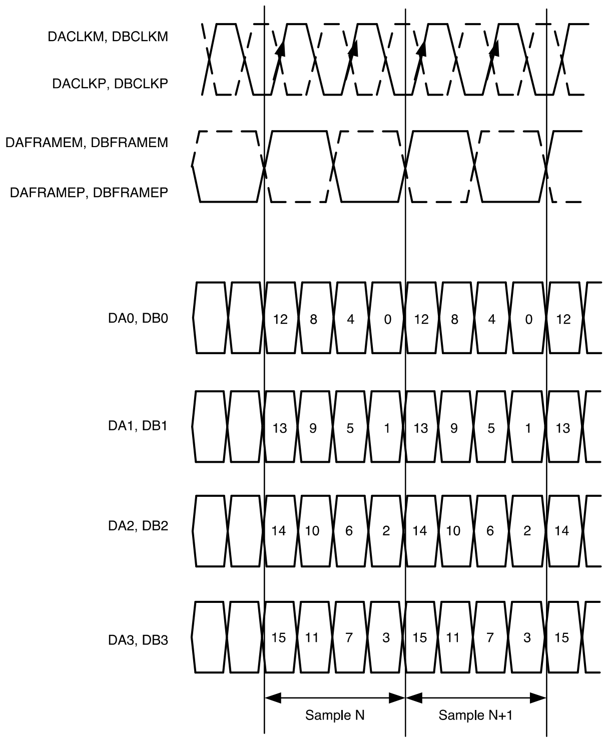

Figure 86. QDR LVDS Interface Timing: ADS42LB69

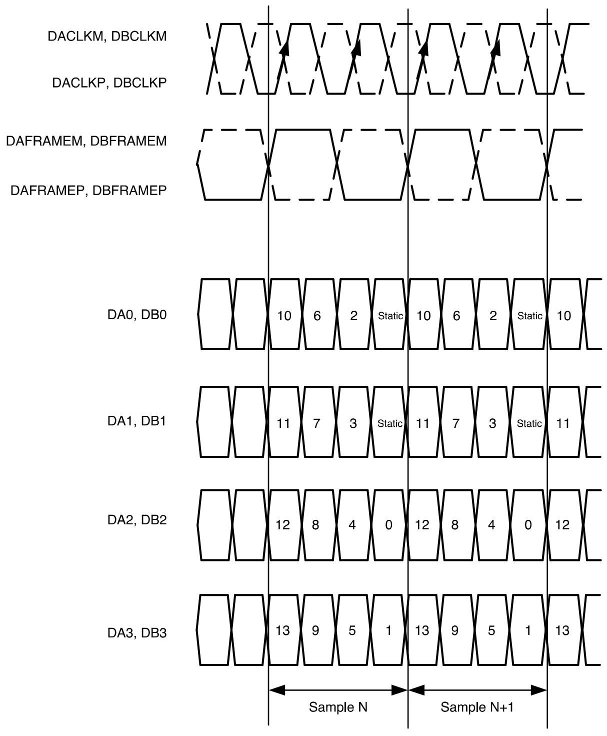

Figure 87. QDR LVDS Interface Timing: ADS42LB49

Figure 12 Data - Frame signaling

3.8 Reset synchronization (sync_rst.v)

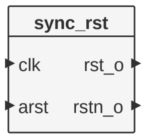

Figure 13 Reset synchronization block inputs and outputs

The MMCM and PLL Dynamic Reconfiguration datasheet is not specific if the locked output of the MMCM signal is synchronous with MMCM clock. Therefore, the synchronization of the lock signal into clk125 domain is necessary. The synchronization routine operates with asynchronous set and synchronous release. The reset signal can occur without running clock.

3.9 Trigger system for DAC and ADC

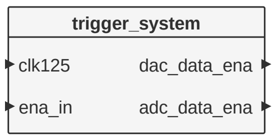

Figure 14 Trigger system inputs and outputs

This module provides triggering of the DAC and the ADC interface.

3.10 ZYNQ system (bd_wr.sv)

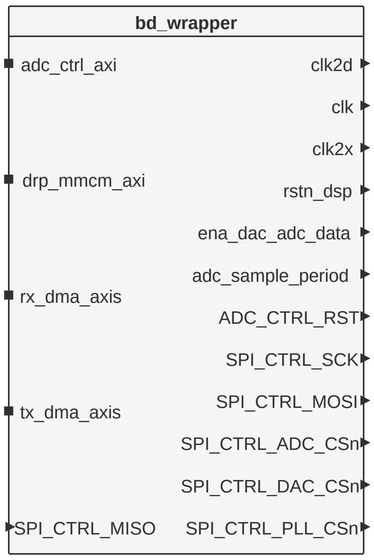

Figure 15 Block design ZYNQ subsystem input and outputs

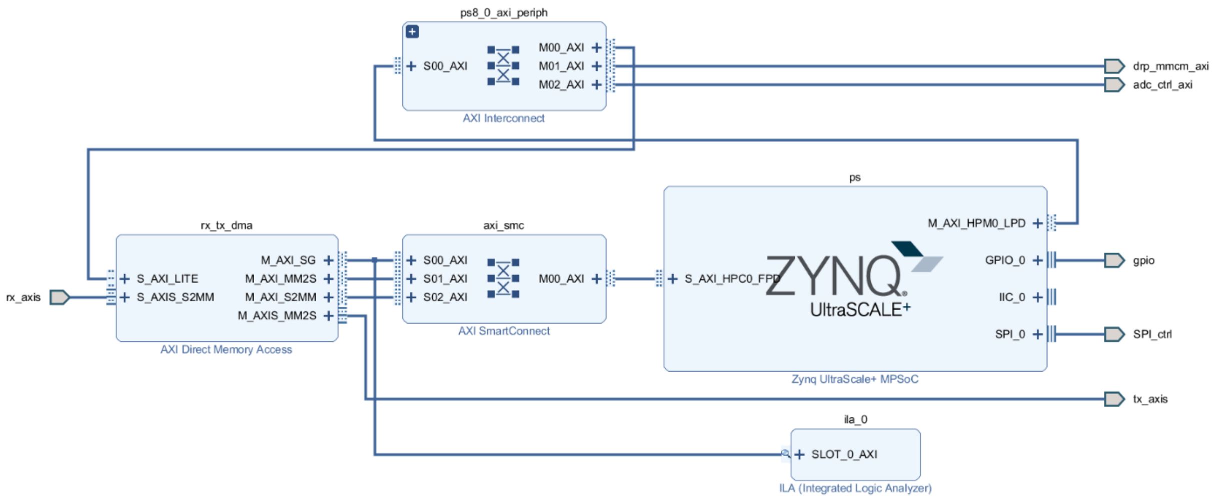

Figure 16 Block design scheme of ZYNQ subsystem

ZYNQ Block Design (BD) system is the core of the entire SDR system. The ZYNQ control all configurable modules in the FPGA (see Table 2) and also control other peripherals such a SPI, I2C, Low Speed (LS) ADCs, LSDACs, GPIO. The Block Design also implements DMA peripheral to direct memory access between Petalinux operating system running on ZYNQ processor and FPGA. For more details see 5.

Configurable modules		
Module name	AXI lite controller name	Description
clock_system	drp_mmcm	**MMCM** system for generation main system clocks
adc	adc_ctrl_axi	**IDELAYE3** ports
		write and read to **adc_ptrn_checker** ports
		read **IDELAYCTRL** port
rx_chain	rx_chain_ctrl_axi	**DDS** control
		CIC control port
		FIR0 control port
		FIR0 reload port
		FIR1 control port
		FIR1 reload port
		Downsampler factor port
		DDS enable
		CIC enable
tx_chain	tx_chain_ctrl_axi	**DDS** control
		CIC control port
		FIR0 control port
		FIR0 reload port
		FIR1 control port
		FIR1 reload port
		Upsampler factor port
		DDS enable
		CIC enable

Table 2 Configurable modules

4 PCB

All sources of the PDB design, BOM, and assembly instructions are available at in the file http://www.radio.feec.vutbr.cz/interop/sdr/sdr_ie_fabrication.zip. Inputs for manufacturing at the provider of your choice are available in the file yyyyy.zip. For options on getting an assembled device, please contact us at urel@feec.vutbr.cz to assist you with further steps.

4.1 Hardware schematics

Schematics of the SDR emulator are available at http://www.radio.feec.vutbr.cz/interop/sdr/schematics.pdf. Table 3 describes each page of the schematics.

Table 3: Structure of the schematics documentation

page	Description
1	SDR mother board system block schematics
2	TE0803 FPGA module – FPGA data connection
3	Block schematics of daughter board subsystem
4	Daughter board Tx connectors
5	Daughter board Rx connectors
6	High-speed DAC
7	High-speed ADC
8	Low-speed DAC

9	Low-speed ADC
10	Clocks and PLL
11	TE0803 FPGA module – processor data connection
12	Ethernet PHY
13	General purpose inputs and outputs
14	USB universal serial receiver and transmitter
15	TE0803 FPGA module – FPGA configuration and control
16	TE0803 FPGA module – power connections
17	Block schematics of powering subsystem
18	Power input
19	Main 12 V to 3.3 V step-down regulator
20	Main digitally-controlled power switch
21	Step-down regulator 12 V to 6 V for powering Daughter board
22	Step-down regulator 12 V to 3.8 V for powering linear regulators
23	Step-down regulator 12 V to 1.8 V for powering digital circuits
24	Low noise linear regulator 3.3 V for powering sensitive analog circuits
25	Universal linear regulator – specified by assembly variant
26	Power filter
27	Ferrite bead power filters

5 PetaLinux

OS Petalinux source code is available at:
http://www.radio.feec.vutbr.cz/interop/sdr/petalinux_srcs.zip.

6 Software source code

Software source code is available at: http://www.radio.feec.vutbr.cz/interop/sdr/sw_srcs.zip.

7 Getting started

To get started with SDR-IE check document Getting started available at:
http://www.radio.feec.vutbr.cz/interop/sdr/SDR-IE_getting_started.pdf. The Linux USB live image to your PC is compressed in http://www.radio.feec.vutbr.cz/interop/sdr/kradio-live-x86_64.zip.

8 Gitlab project file structures

This section describes the main project trunk (https://gitlab.com/jakral/kradio_fw.git) including documentation of entire HW and SW part used to assembly SDR-IE, source files to generate FPGA image, source files to compile Petalinux operation system for ZYNQ, source files to compile PC app to communicate with the SDR-IE. MATLAB files to filter design of the Xilinx FIR IP cores and scripts for results evaluation of the interference rejection test based on the ISO/IEC 18046-3:2019(E). (see INTERFERENCE REJECTION MEASUREMENT REPORT document).

```
├── agenda
├── docs_______________________________________Documentations, datasheets and schematics
│   ├── acrios
│   ├── datasheets
│   │   ├── Ettus
│   │   │   ├── SBX
│   │   │   └── WBX
│   │   └── Xilinx
│   ├── drafts
│   ├── schematics
│   └── workshop_elinux
├── matlab_____________________________________Measurements with SDR-IE according to ISO/IEC 18000-63:2015
│   ├── rfid_response
│   └── rx_tx_analysis____________________Measurements results with analysis scripts
├── pc_apps
│   └── udp_tester
└── vivado
    ├── ip_repo
    │   └── axi_drp
    │       ├── rtl
    │       └── xgui
    └── kradio_fw______________________________Main Vivado FPGA project
        ├── kradio_fw.sdk_________________SDK projects
        │   ├── ether_test
        │   │   └── src
        │   ├── ether_test_bsp
        │   ├── fsbl
        │   │   └── src
        │   ├── fsbl_bsp
        │   ├── gpio_test
        │   │   └── src
        │   ├── gpio_test_bsp
        │   ├── kchdr_server
        │   │   └── src
        │   ├── libscpi
        │   │   ├── dist
        │   │   ├── inc
        │   │   ├── src
        │   │   └── test
        │   └── scpi_server
        │       └── src
        └── kradio_fw.srcs________________FPGA sources
            ├── constrs_1__________Constraint XDC files
            │   └── new
            ├── sim_models________________Simulation sources
            ├── sim_parts__________Simulation sources
            ├── sim_top___________________Simulation sources
            └── sources_1
                ├── bd____________________Block design sources
                ├── hdl___________________HDL sources
                └── ip____________________IP core sources
```