Laboratorní TestBed fyzické vrstvy V2X komunikačního systému v pásmu 5.9GHz

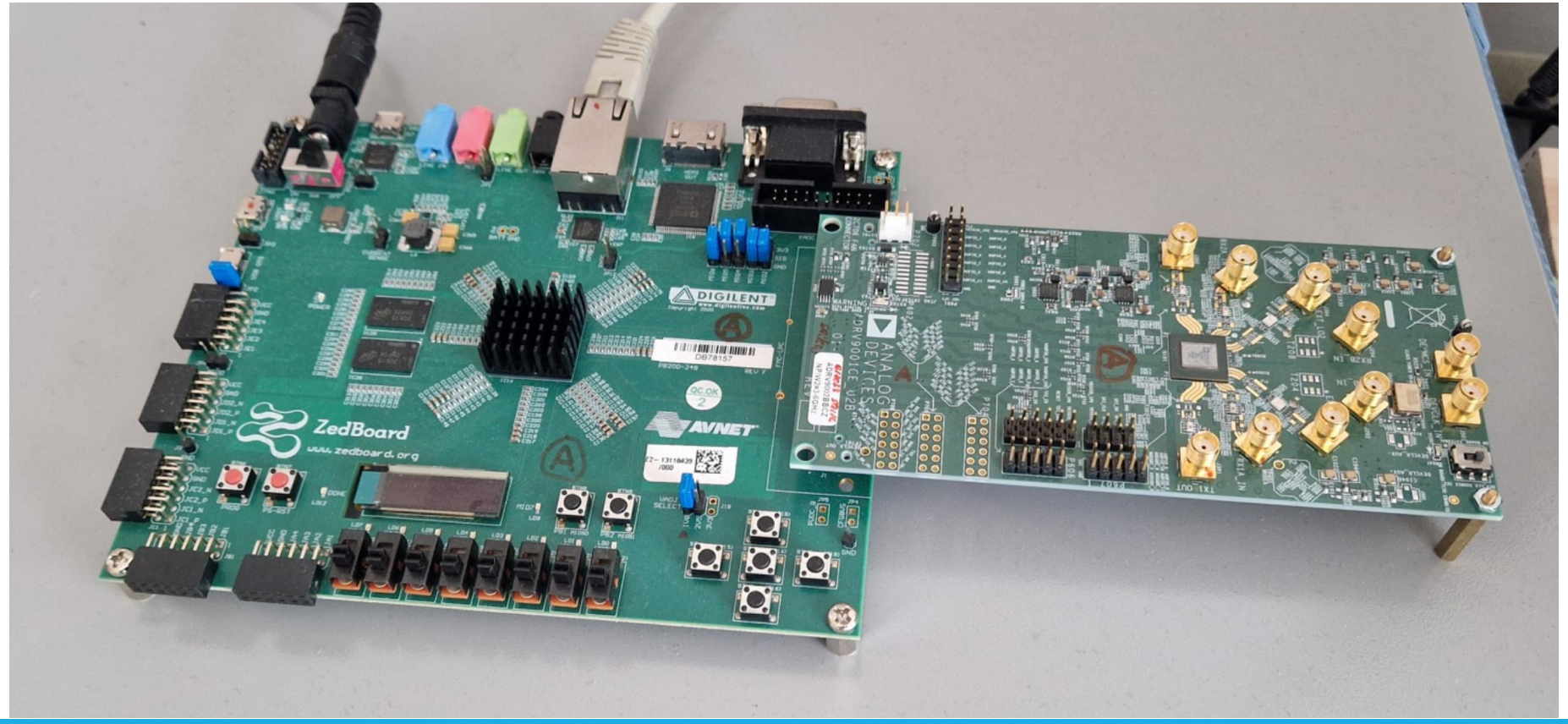
MICHAEL KIMMER

Úvod

- Realtime přijímač IEEE 802.11p
- Možnost postupného rozšiřování
- IEEE 802.11p V2X standard
  - 5.9 GHz, ad-hoc architektura
  - Vyžaduje rychlé potvrzování přijetí framů (16 nebo 32 μs)
- FPGA přístup
- GUI ovládání
- Společný projekt s firmou Hermann

Hardware

- ZedBoard + ADRV9002

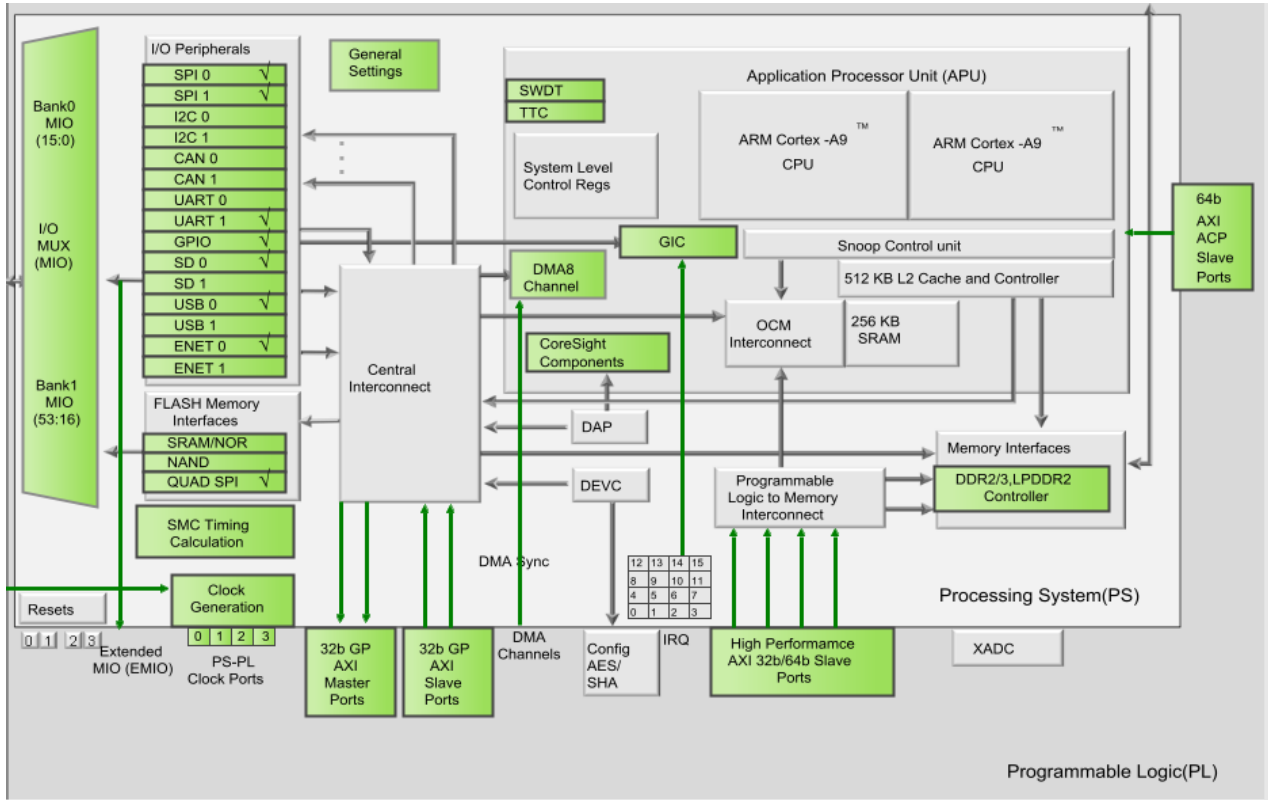


Hardware

- ADRV9002
  - 2x2 Transceiver
  - Frequency range: 30 MHz to 6 GHz
  - Bandwidths: 12 kHz to 40 MHz
- ZedBoard
  - Development board
  - SoC: Xilinx Zynq®-7000 (XC7Z020-CLG484-1)
    - FPGA (85K Programmable Logic Cells)
    - Dual-core ARM Cortex-A9 based CPU (667 MHz)
  - Many IOs
  - Analog Devices: HDL, Kuiper Linux, Libiio

Hardware

- ADRV9002
  - 2x2 Transceiver
  - Frequency range: 30 MHz to 6 GHz
  - Bandwidths: 12 kHz to 40 MHz
- ZedBoard
  - Development board
  - SoC: Xilinx Zynq®-7000 (XC7Z020-CLG484-1)
    - FPGA (85K Programmable Logic Cells)
    - Dual-core ARM Cortex-A9 based CPU (667 MHz)
  - Many IOs
  - Analog Devices: HDL, Kuiper Linux, Libiio



Celkové schéma FPGA

- ADI HDL Reference Design (pro ADRV9002 a ZedBoard)
- Vlastní IP blok IP_802_11p
- AXI4 Lite
  - Připojení IP_802_11p do adresového prostoru CPU
  - Ovládání IP_802_11p
  - Čtení dat z IP_802_11p
- ADI HDL: axi_adrv9001 blok
  - Přístup k IQ vzorkům z ADRV9002 (pro IP_802_11p)

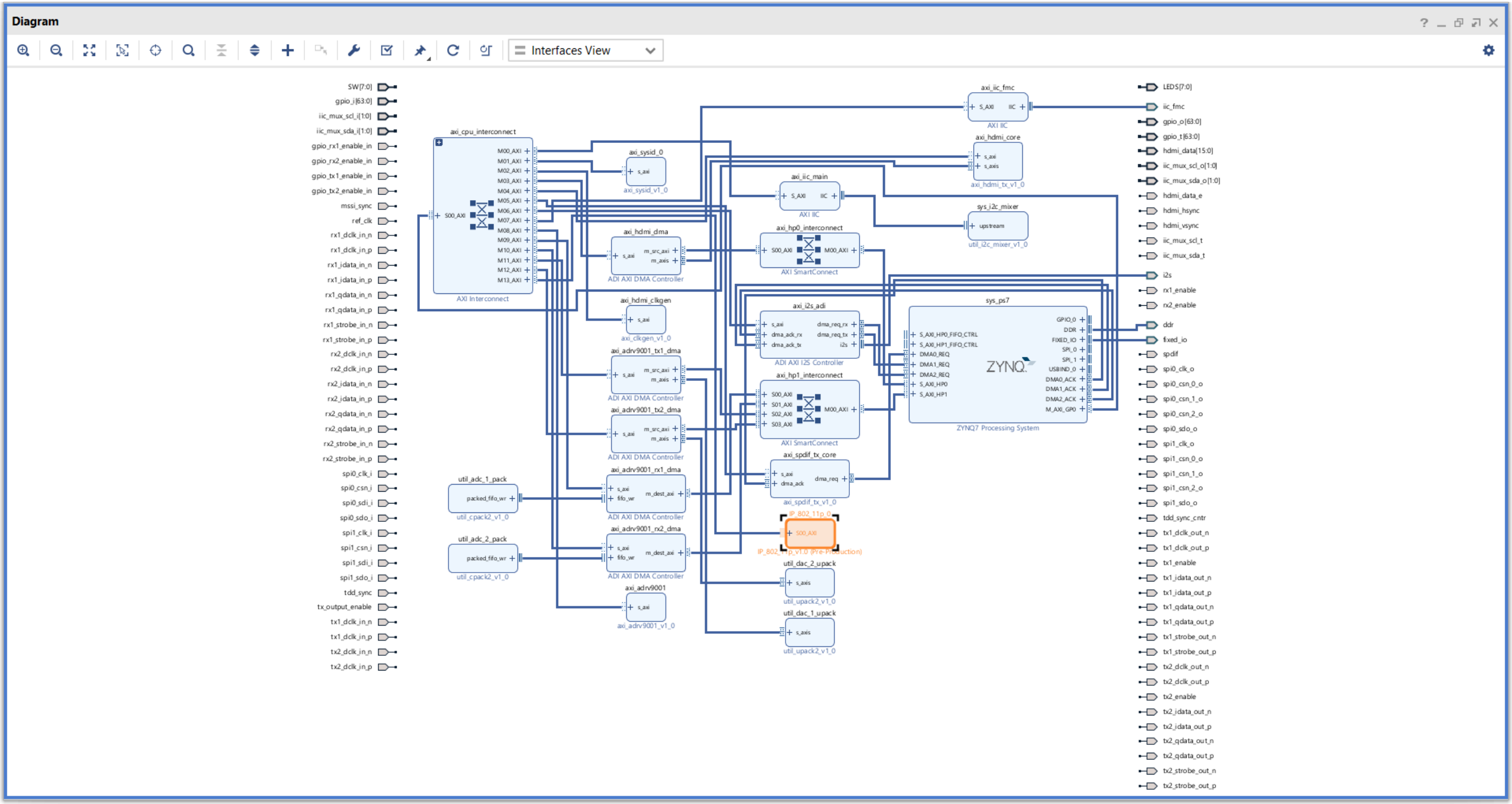


Diagram
Interfaces View
SW[7:0]
gpio_i[63:0]
iic_mux_scl_i[1:0]
iic_mux_sda_i[1:0]
gpio_rx1_enable_in
gpio_rx2_enable_in
gpio_tx1_enable_in
gpio_tx2_enable_in
mssi_sync
ref_clk
rx1_dclk_in_n
rx1_dclk_in_p
rx1_idata_in_n
rx1_idata_in_p
rx1_qdata_in_n
rx1_qdata_in_p
rx1_strobe_in_n
rx1_strobe_in_p
rx2_dclk_in_n
rx2_dclk_in_p
rx2_idata_in_n
rx2_idata_in_p
rx2_qdata_in_n
rx2_qdata_in_p
rx2_strobe_in_n
rx2_strobe_in_p
spi0_clk_i
spi0_csn_i
spi0_sdi_i
spi0_sdo_i
spi1_clk_i
spi1_csn_i
spi1_sdi_i
spi1_sdo_i
tdd_sync
tx_output_enable
tx1_dclk_in_n
tx1_dclk_in_p
tx2_dclk_in_n
tx2_dclk_in_p
axi_cpu_interconnect
S00_AXI
M00_AXI
M01_AXI
M02_AXI
M03_AXI
M04_AXI
M05_AXI
M06_AXI
M07_AXI
M08_AXI
M09_AXI
M10_AXI
M11_AXI
M12_AXI
M13_AXI
AXI Interconnect
axi_sysid_0
s_axi
axi_sysid_v1_0
axi_hdmi_dma
m_src_axi
m_axis
ADI AXI DMA Controller
axi_hdmi_clkgen
axi_clkgen_v1_0
axi_adrv9001_tx1_dma
axi_adrv9001_tx2_dma
axi_adrv9001_rx1_dma
axi_adrv9001_rx2_dma
util_adc_1_pack
packed_fifo_wr
util_cpack2_v1_0
util_adc_2_pack
axi_adrv9001
axi_adrv9001_v1_0
axi_iic_main
S_AXI
IIC
AXI IIC
axi_hp0_interconnect
axi_hp1_interconnect
S00_AXI
S01_AXI
S02_AXI
S03_AXI
M00_AXI
AXI SmartConnect
axi_i2s_adi
dma_req_rx
dma_req_tx
dma_ack_rx
dma_ack_tx
i2s
ADI AXI I2S Controller
axi_spdif_tx_core
dma_req
dma_ack
axi_spdif_tx_v1_0
IP_802_11p_0
util_dac_2_upack
util_dac_1_upack
util_upack2_v1_0
axi_iic_fmc
axi_hdmi_core
axi_hdmi_tx_v1_0
sys_i2c_mixer
upstream
util_i2c_mixer_v1_0
sys_ps7
ZYNQ7 Processing System
LEDS[7:0]
iic_fmc
gpio_o[63:0]
gpio_t[63:0]
hdmi_data[15:0]
iic_mux_scl_o[1:0]
iic_mux_sda_o[1:0]
hdmi_data_e
hdmi_hsync
hdmi_vsync
iic_mux_scl_t
iic_mux_sda_t
i2s
rx1_enable
rx2_enable
ddr
fixed_io
spdif
spi0_clk_o
spi0_csn_0_o
spi0_csn_1_o
spi0_csn_2_o
spi0_sdo_o
spi1_clk_o
spi1_csn_0_o
spi1_csn_1_o
spi1_csn_2_o
spi1_sdo_o
tdd_sync_cntr
tx1_dclk_out_n
tx1_dclk_out_p
tx1_enable
tx1_idata_out_n
tx1_idata_out_p
tx1_qdata_out_n
tx1_qdata_out_p
tx1_strobe_out_n
tx1_strobe_out_p
tx2_dclk_out_n
tx2_dclk_out_p
tx2_enable
tx2_idata_out_n
tx2_idata_out_p
tx2_qdata_out_n
tx2_qdata_out_p
tx2_strobe_out_n
tx2_strobe_out_p

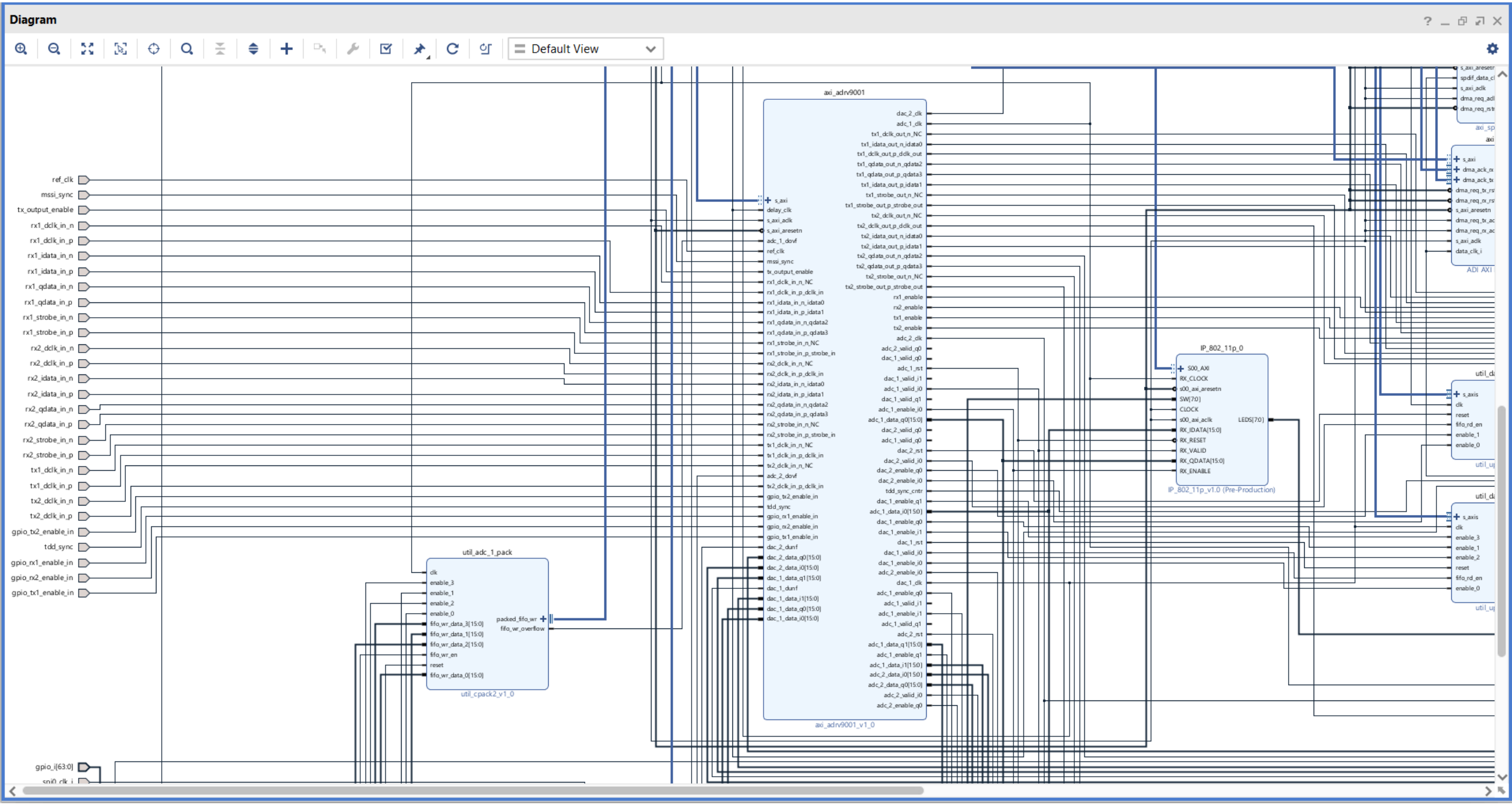
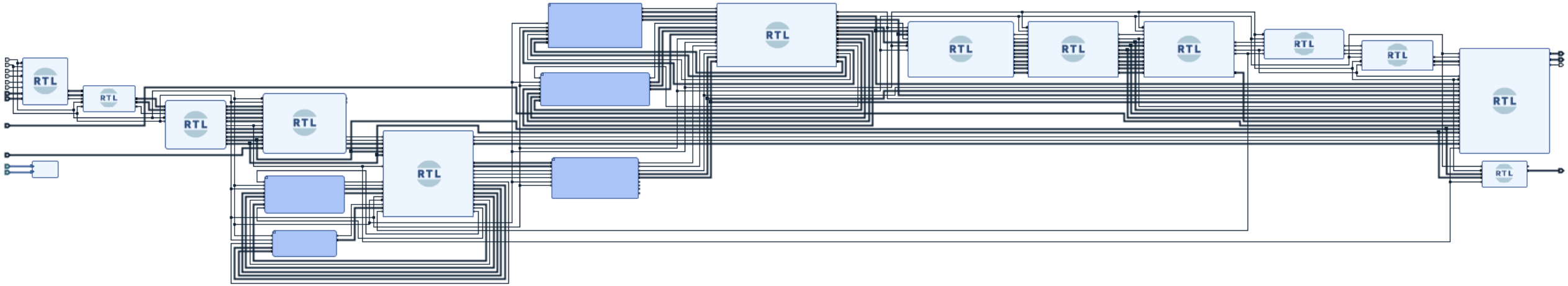


Diagram
Default View
ref_clk
mssi_sync
tx_output_enable
rx1_dclk_in_n
rx1_dclk_in_p
rx1_idata_in_n
rx1_idata_in_p
rx1_qdata_in_n
rx1_qdata_in_p
rx1_strobe_in_n
rx1_strobe_in_p
rx2_dclk_in_n
rx2_dclk_in_p
rx2_idata_in_n
rx2_idata_in_p
rx2_qdata_in_n
rx2_qdata_in_p
rx2_strobe_in_n
rx2_strobe_in_p
tx1_dclk_in_n
tx1_dclk_in_p
tx2_dclk_in_n
tx2_dclk_in_p
gpio_tx2_enable_in
tdd_sync
gpio_rx1_enable_in
gpio_rx2_enable_in
gpio_tx1_enable_in
gpio_i[63:0]
util_adc_1_pack
clk
enable_3
enable_1
enable_2
enable_0
fifo_wr_data_3[15:0]
fifo_wr_data_1[15:0]
fifo_wr_data_2[15:0]
fifo_wr_en
reset
fifo_wr_data_0[15:0]
packed_fifo_wr
fifo_wr_overflow
util_cpack2_v1_0
axi_adrv9001
s_axi
delay_clk
s_axi_aclk
s_axi_aresetn
adc_1_dovf
ref_clk
mssi_sync
tx_output_enable
rx1_dclk_in_n_NC
rx1_dclk_in_p_dclk_in
rx1_idata_in_n_idata0
rx1_idata_in_p_idata1
rx1_qdata_in_n_qdata2
rx1_qdata_in_p_qdata3
rx1_strobe_in_n_NC
rx1_strobe_in_p_strobe_in
rx2_dclk_in_n_NC
rx2_dclk_in_p_dclk_in
rx2_idata_in_n_idata0
rx2_idata_in_p_idata1
rx2_qdata_in_n_qdata2
rx2_qdata_in_p_qdata3
rx2_strobe_in_n_NC
rx2_strobe_in_p_strobe_in
tx1_dclk_in_n_NC
tx1_dclk_in_p_dclk_in
tx2_dclk_in_n_NC
adc_2_dovf
tx2_dclk_in_p_dclk_in
gpio_tx2_enable_in
tdd_sync
gpio_rx1_enable_in
gpio_rx2_enable_in
gpio_tx1_enable_in
dac_2_dunf
dac_2_data_q0[15:0]
dac_2_data_i0[15:0]
dac_1_data_q1[15:0]
dac_1_dunf
dac_1_data_i1[15:0]
dac_1_data_q0[15:0]
dac_1_data_i0[15:0]
dac_2_clk
adc_1_clk
tx1_dclk_out_n_NC
tx1_idata_out_n_idata0
tx1_dclk_out_p_dclk_out
tx1_qdata_out_n_qdata2
tx1_qdata_out_p_qdata3
tx1_idata_out_p_idata1
tx1_strobe_out_n_NC
tx1_strobe_out_p_strobe_out
tx2_dclk_out_n_NC
tx2_dclk_out_p_dclk_out
tx2_idata_out_n_idata0
tx2_idata_out_p_idata1
tx2_qdata_out_n_qdata2
tx2_qdata_out_p_qdata3
tx2_strobe_out_n_NC
tx2_strobe_out_p_strobe_out
rx1_enable
rx2_enable
tx1_enable
tx2_enable
adc_2_clk
adc_2_valid_q0
dac_1_valid_q0
adc_1_rst
dac_1_valid_i1
adc_1_valid_i0
dac_1_valid_q1
adc_1_enable_i0
adc_1_data_q0[15:0]
dac_2_valid_q0
adc_1_valid_q0
dac_2_rst
dac_2_valid_i0
dac_2_enable_q0
dac_2_enable_i0
tdd_sync_cntr
dac_1_enable_q1
adc_1_data_i0[15:0]
dac_1_enable_q0
dac_1_enable_i1
dac_1_rst
dac_1_valid_i0
dac_1_enable_i0
adc_2_enable_i0
dac_1_clk
adc_1_enable_q0
adc_1_valid_i1
adc_1_enable_i1
adc_1_valid_q1
adc_2_rst
adc_1_data_q1[15:0]
adc_1_enable_q1
adc_1_data_i1[15:0]
adc_2_data_i0[15:0]
adc_2_data_q0[15:0]
adc_2_valid_i0
adc_2_enable_q0
axi_adrv9001_v1_0
IP_802_11p_0
S00_AXI
RX_CLOCK
s00_axi_aresetn
SW[7:0]
CLOCK
s00_axi_aclk
RX_IDATA[15:0]
RX_RESET
RX_VALID
RX_QDATA[15:0]
RX_ENABLE
LEDS[7:0]
IP_802_11p_v1.0 (Pre-Production)
ADI AXI

Blokové schéma práce (blok IP_802_11p)

- Vlastních 21 VHDL bloků, 6 Vivado IP bloků
- Příjem 802.11p PHY framů (povinné modulace a kódové rychlosti)
- Ukládání dat do 4 KB BRAM
- Python GUI
  - AXI4: konfigurace bloku a čtení dat z BRAM
  - Libiio: jednoduché vysílání



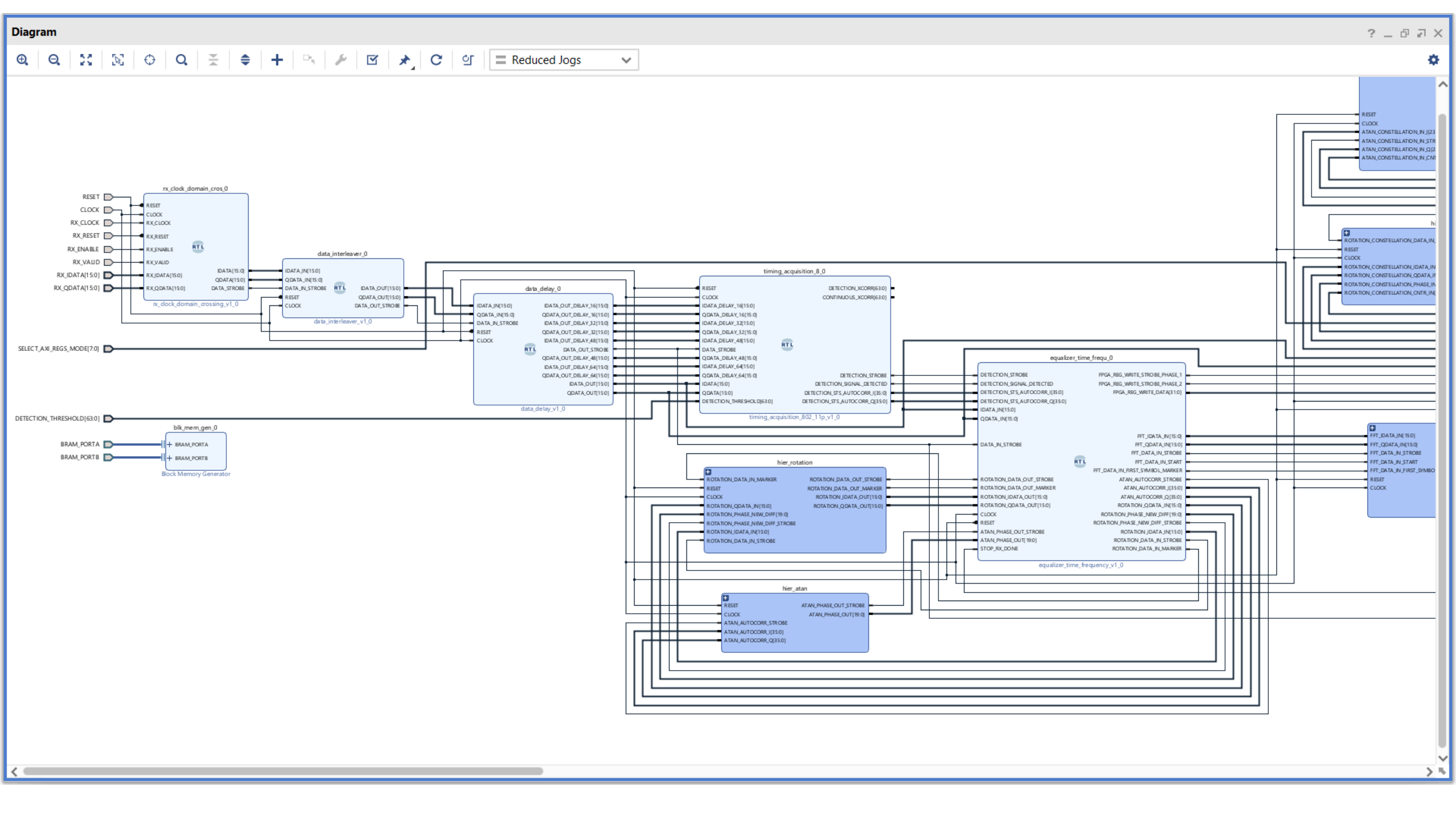


Diagram
Reduced Jogs
RESET
CLOCK
RX_CLOCK
RX_RESET
RX_ENABLE
RX_VALID
RX_IDATA[15:0]
RX_QDATA[15:0]
rx_clock_domain_cros_0
rx_clock_domain_crossing_v1_0
data_interleaver_0
data_interleaver_v1_0
SELECT_AXI_REGS_MODE[7:0]
data_delay_0
data_delay_v1_0
timing_acquisition_8_0
timing_acquisition_802_11p_v1_0
DETECTION_THRESHOLD[63:0]
blk_mem_gen_0
BRAM_PORTA
BRAM_PORTB
Block Memory Generator
hier_rotation
hier_atan
equalizer_time_frequ_0
equalizer_time_frequency_v1_0

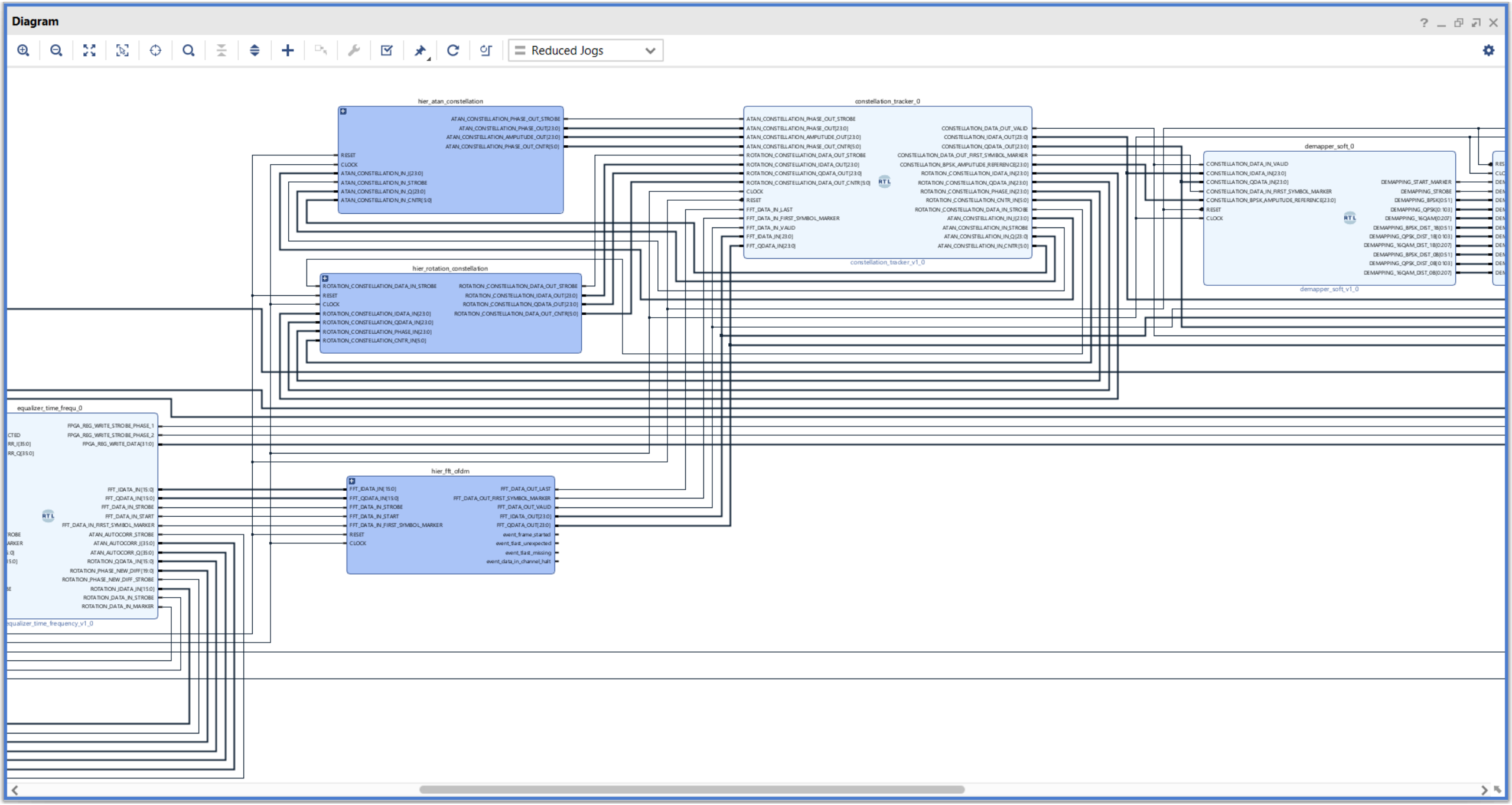


Diagram
Reduced Jogs
hier_atan_constellation
ATAN_CONSTELLATION_PHASE_OUT_STROBE
ATAN_CONSTELLATION_PHASE_OUT[23:0]
ATAN_CONSTELLATION_AMPLITUDE_OUT[23:0]
ATAN_CONSTELLATION_PHASE_OUT_CNTR[5:0]
RESET
CLOCK
ATAN_CONSTELLATION_IN_I[23:0]
ATAN_CONSTELLATION_IN_STROBE
ATAN_CONSTELLATION_IN_Q[23:0]
ATAN_CONSTELLATION_IN_CNTR[5:0]
constellation_tracker_0
ATAN_CONSTELLATION_PHASE_OUT_STROBE
ATAN_CONSTELLATION_PHASE_OUT[23:0]
ATAN_CONSTELLATION_AMPLITUDE_OUT[23:0]
ATAN_CONSTELLATION_PHASE_OUT_CNTR[5:0]
ROTATION_CONSTELLATION_DATA_OUT_STROBE
ROTATION_CONSTELLATION_IDATA_OUT[23:0]
ROTATION_CONSTELLATION_QDATA_OUT[23:0]
ROTATION_CONSTELLATION_DATA_OUT_CNTR[5:0]
CLOCK
RESET
FFT_DATA_IN_LAST
FFT_DATA_IN_FIRST_SYMBOL_MARKER
FFT_DATA_IN_VALID
FFT_IDATA_IN[23:0]
FFT_QDATA_IN[23:0]
CONSTELLATION_DATA_OUT_VALID
CONSTELLATION_IDATA_OUT[23:0]
CONSTELLATION_QDATA_OUT[23:0]
CONSTELLATION_DATA_OUT_FIRST_SYMBOL_MARKER
CONSTELLATION_BPSK_AMPLITUDE_REFERENCE[23:0]
ROTATION_CONSTELLATION_IDATA_IN[23:0]
ROTATION_CONSTELLATION_QDATA_IN[23:0]
ROTATION_CONSTELLATION_PHASE_IN[23:0]
ROTATION_CONSTELLATION_CNTR_IN[5:0]
ROTATION_CONSTELLATION_DATA_IN_STROBE
ATAN_CONSTELLATION_IN_I[23:0]
ATAN_CONSTELLATION_IN_STROBE
ATAN_CONSTELLATION_IN_Q[23:0]
ATAN_CONSTELLATION_IN_CNTR[5:0]
constellation_tracker_v1_0
demapper_soft_0
CONSTELLATION_DATA_IN_VALID
CONSTELLATION_IDATA_IN[23:0]
CONSTELLATION_QDATA_IN[23:0]
CONSTELLATION_DATA_IN_FIRST_SYMBOL_MARKER
CONSTELLATION_BPSK_AMPLITUDE_REFERENCE[23:0]
RESET
CLOCK
DEMAPPING_START_MARKER
DEMAPPING_STROBE
DEMAPPING_BPSK[0:51]
DEMAPPING_QPSK[0:103]
DEMAPPING_16QAM[0:207]
DEMAPPING_BPSK_DIST_1B[0:51]
DEMAPPING_QPSK_DIST_1B[0:103]
DEMAPPING_16QAM_DIST_1B[0:207]
DEMAPPING_BPSK_DIST_0B[0:51]
DEMAPPING_QPSK_DIST_0B[0:103]
DEMAPPING_16QAM_DIST_0B[0:207]
demapper_soft_v1_0
hier_rotation_constellation
ROTATION_CONSTELLATION_DATA_IN_STROBE
RESET
CLOCK
ROTATION_CONSTELLATION_IDATA_IN[23:0]
ROTATION_CONSTELLATION_QDATA_IN[23:0]
ROTATION_CONSTELLATION_PHASE_IN[23:0]
ROTATION_CONSTELLATION_CNTR_IN[5:0]
ROTATION_CONSTELLATION_DATA_OUT_STROBE
ROTATION_CONSTELLATION_IDATA_OUT[23:0]
ROTATION_CONSTELLATION_QDATA_OUT[23:0]
ROTATION_CONSTELLATION_DATA_OUT_CNTR[5:0]
equalizer_time_frequ_0
FPGA_REG_WRITE_STROBE_PHASE_1
FPGA_REG_WRITE_STROBE_PHASE_2
FPGA_REG_WRITE_DATA[31:0]
FFT_IDATA_IN[15:0]
FFT_QDATA_IN[15:0]
FFT_DATA_IN_STROBE
FFT_DATA_IN_START
FFT_DATA_IN_FIRST_SYMBOL_MARKER
ATAN_AUTOCORR_STROBE
ATAN_AUTOCORR_I[35:0]
ATAN_AUTOCORR_Q[35:0]
ROTATION_QDATA_IN[15:0]
ROTATION_PHASE_NEW_DIFF[19:0]
ROTATION_PHASE_NEW_DIFF_STROBE
ROTATION_IDATA_IN[15:0]
ROTATION_DATA_IN_STROBE
ROTATION_DATA_IN_MARKER
equalizer_time_frequency_v1_0
hier_fft_ofdm
FFT_IDATA_IN[15:0]
FFT_QDATA_IN[15:0]
FFT_DATA_IN_STROBE
FFT_DATA_IN_START
FFT_DATA_IN_FIRST_SYMBOL_MARKER
RESET
CLOCK
FFT_DATA_OUT_LAST
FFT_DATA_OUT_FIRST_SYMBOL_MARKER
FFT_DATA_OUT_VALID
FFT_IDATA_OUT[23:0]
FFT_QDATA_OUT[23:0]
event_frame_started
event_tlast_unexpected
event_tlast_missing
event_data_in_channel_halt

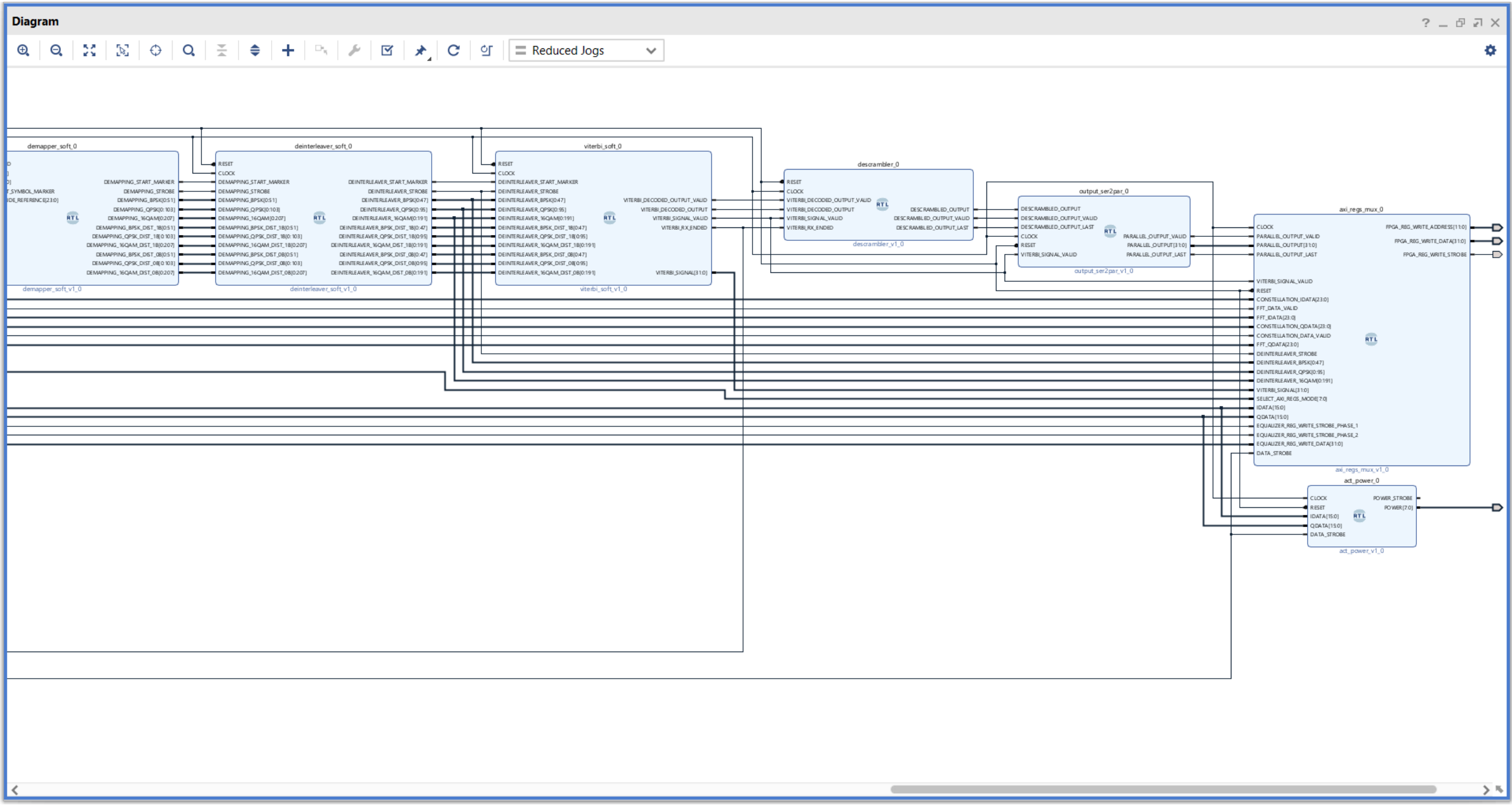
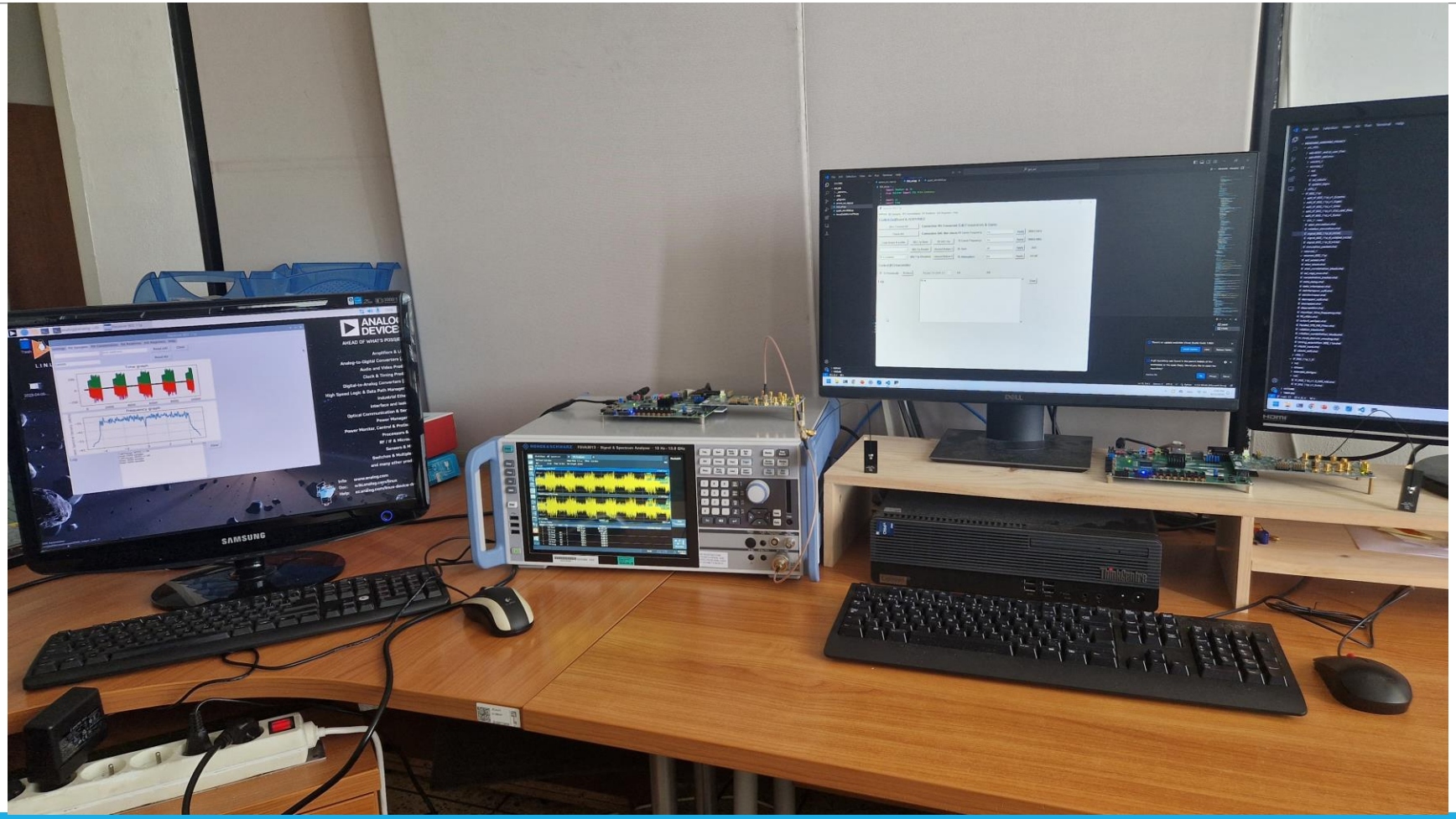
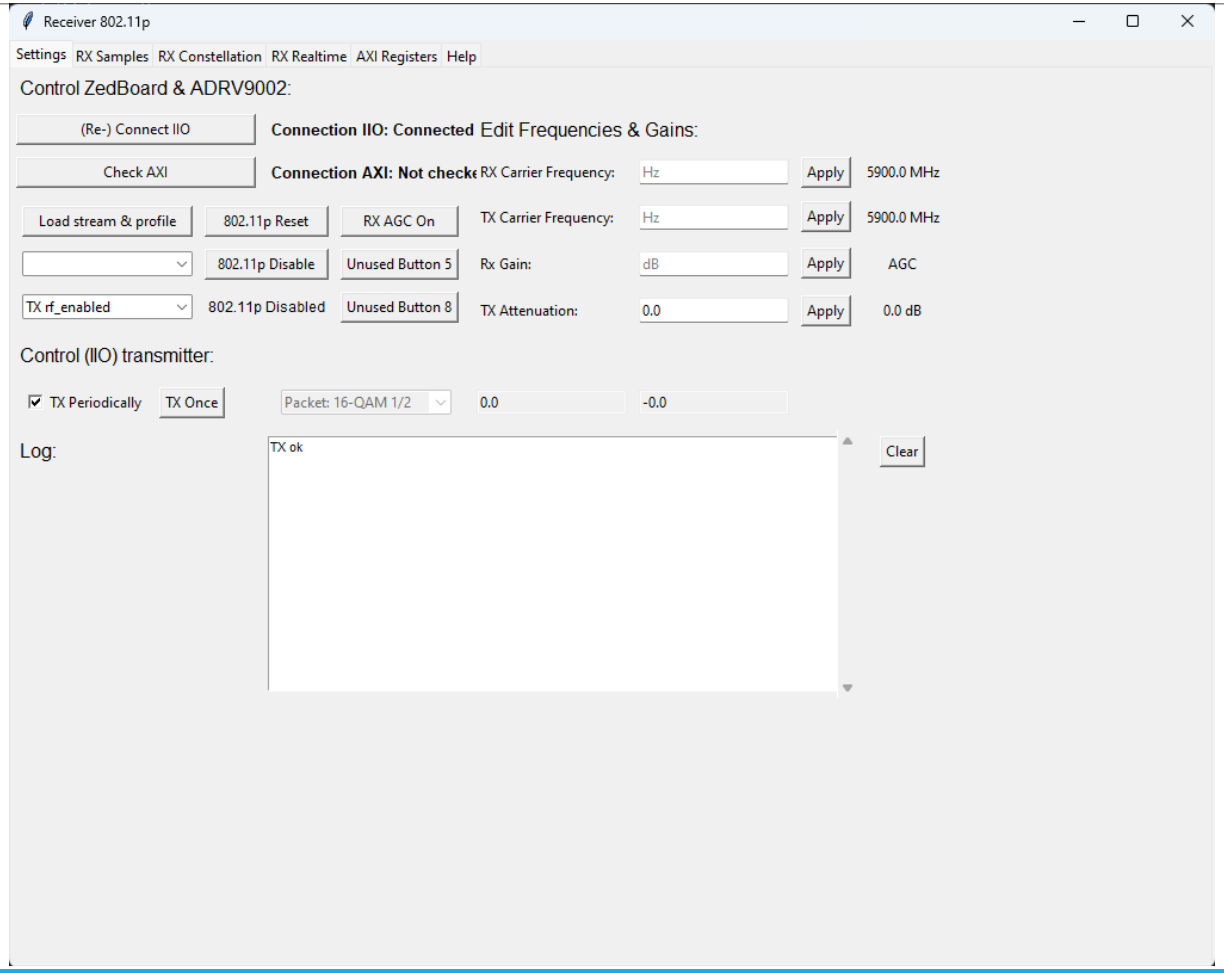


Diagram
Reduced Jogs
demapper_soft_0
DEMAPPING_START_MARKER
DEMAPPING_STROBE
DEMAPPING_BPSK[0:51]
DEMAPPING_QPSK[0:103]
DEMAPPING_16QAM[0:207]
DEMAPPING_BPSK_DIST_1B[0:51]
DEMAPPING_QPSK_DIST_1B[0:103]
DEMAPPING_16QAM_DIST_1B[0:207]
DEMAPPING_BPSK_DIST_0B[0:51]
DEMAPPING_QPSK_DIST_0B[0:103]
DEMAPPING_16QAM_DIST_0B[0:207]
demapper_soft_v1_0
deinterleaver_soft_0
RESET
CLOCK
DEINTERLEAVER_START_MARKER
DEINTERLEAVER_STROBE
DEINTERLEAVER_BPSK[0:47]
DEINTERLEAVER_QPSK[0:95]
DEINTERLEAVER_16QAM[0:191]
DEINTERLEAVER_BPSK_DIST_1B[0:47]
DEINTERLEAVER_QPSK_DIST_1B[0:95]
DEINTERLEAVER_16QAM_DIST_1B[0:191]
DEINTERLEAVER_BPSK_DIST_0B[0:47]
DEINTERLEAVER_QPSK_DIST_0B[0:95]
DEINTERLEAVER_16QAM_DIST_0B[0:191]
deinterleaver_soft_v1_0
viterbi_soft_0
VITERBI_DECODED_OUTPUT_VALID
VITERBI_DECODED_OUTPUT
VITERBI_SIGNAL_VALID
VITERBI_RX_ENDED
VITERBI_SIGNAL[31:0]
viterbi_soft_v1_0
descrambler_0
DESCRAMBLED_OUTPUT
DESCRAMBLED_OUTPUT_VALID
DESCRAMBLED_OUTPUT_LAST
descrambler_v1_0
output_ser2par_0
PARALLEL_OUTPUT_VALID
PARALLEL_OUTPUT[31:0]
PARALLEL_OUTPUT_LAST
output_ser2par_v1_0
axi_regs_mux_0
FPGA_REG_WRITE_ADDRESS[11:0]
FPGA_REG_WRITE_DATA[31:0]
FPGA_REG_WRITE_STROBE
CONSTELLATION_IDATA[23:0]
FFT_DATA_VALID
FFT_IDATA[23:0]
CONSTELLATION_QDATA[23:0]
CONSTELLATION_DATA_VALID
FFT_QDATA[23:0]
SELECT_AXI_REGS_MODE[7:0]
IDATA[15:0]
QDATA[15:0]
EQUALIZER_REG_WRITE_STROBE_PHASE_1
EQUALIZER_REG_WRITE_STROBE_PHASE_2
EQUALIZER_REG_WRITE_DATA[31:0]
DATA_STROBE
axi_regs_mux_v1_0
act_power_0
POWER_STROBE
POWER[7:0]
act_power_v1_0

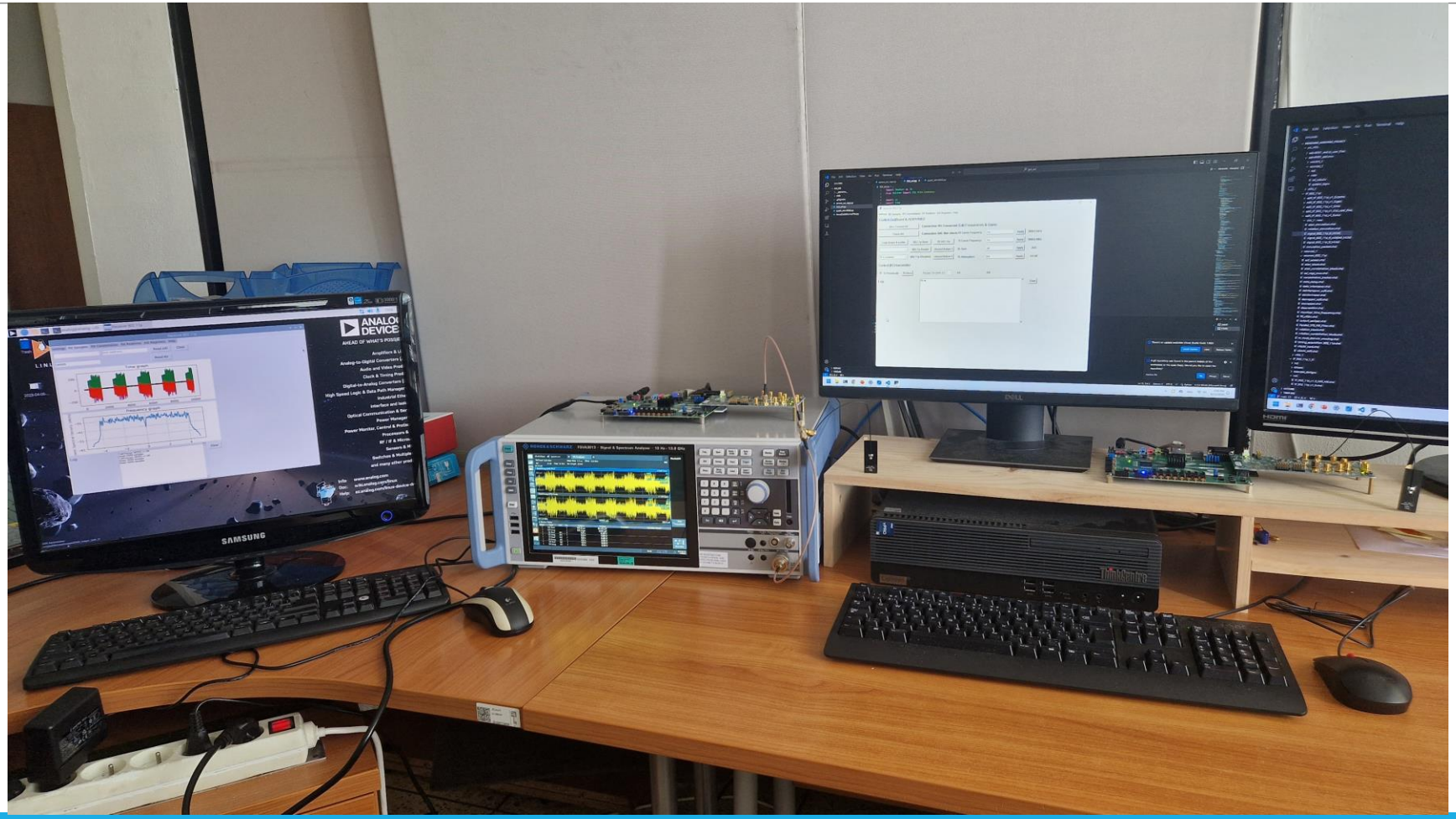
Ukázka



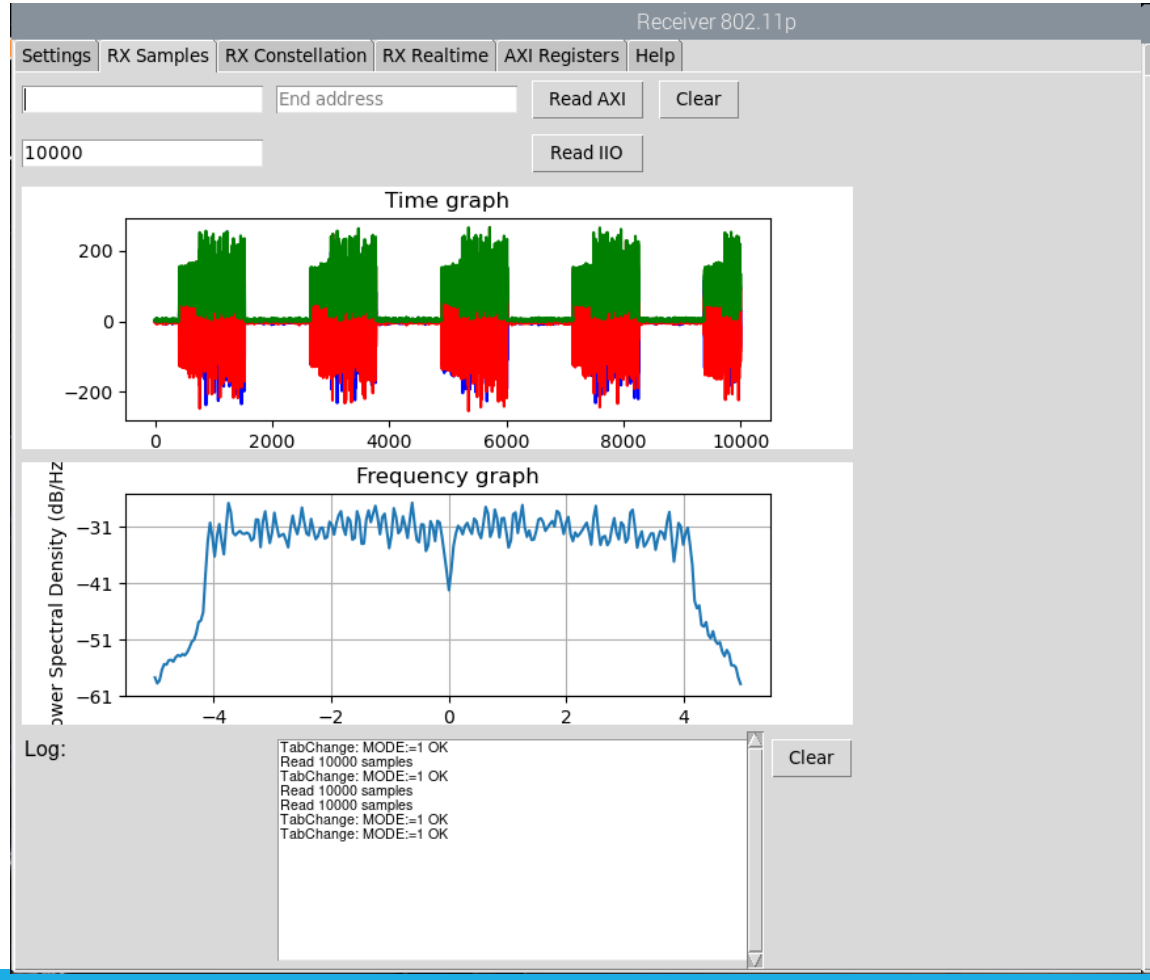
Ukázka – GUI vysílač

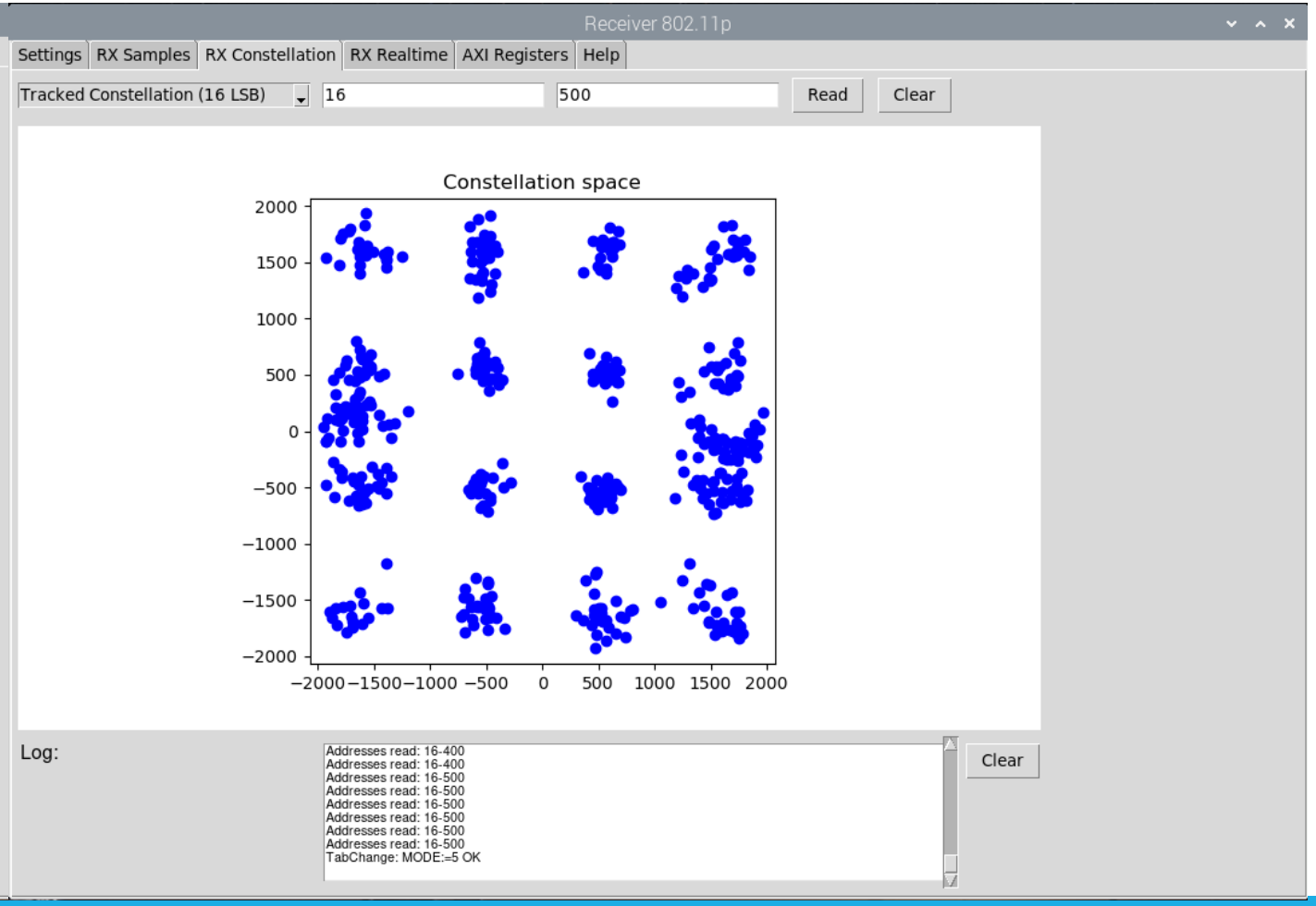


Ukázka

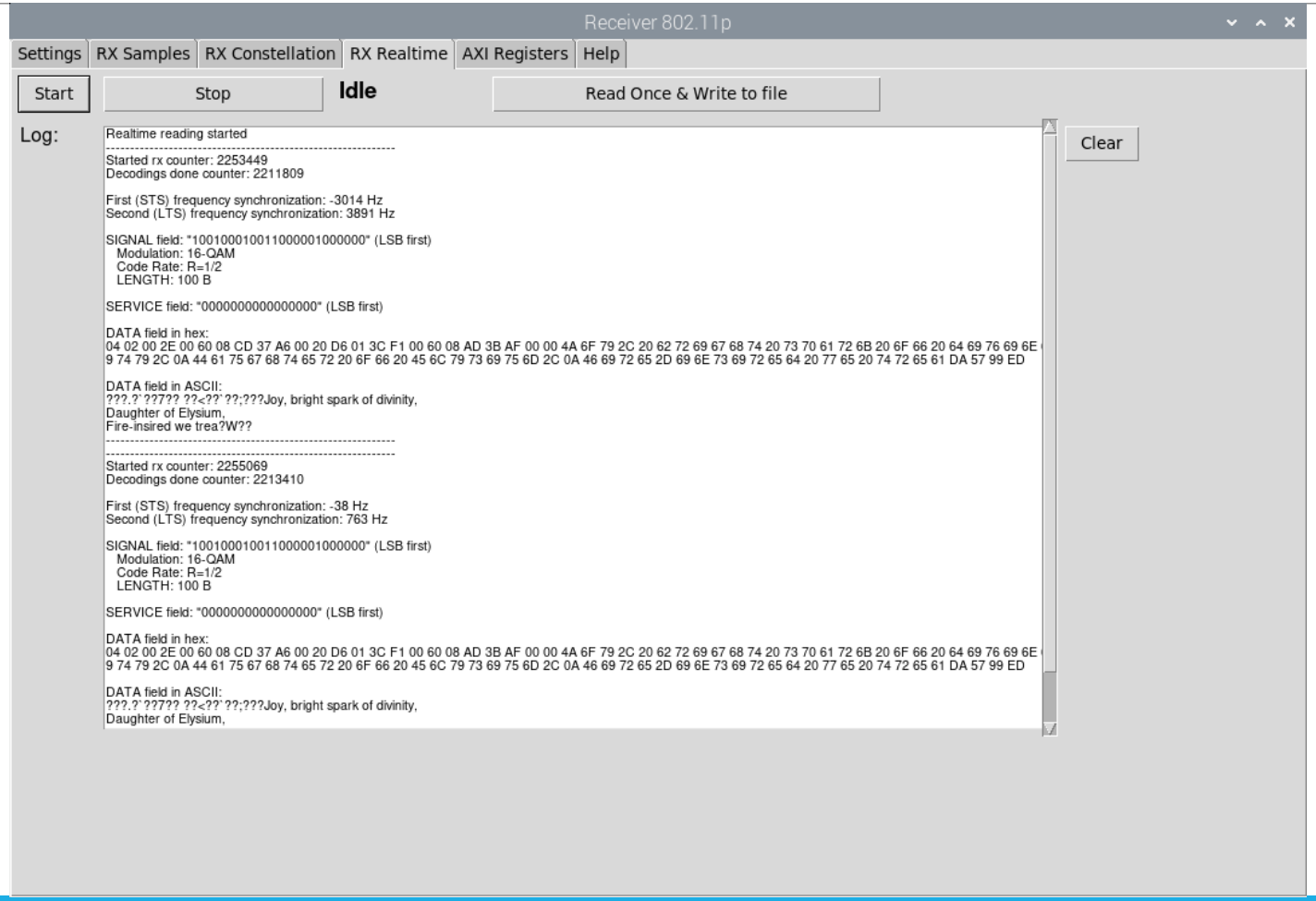


Ukázka – GUI přijímač



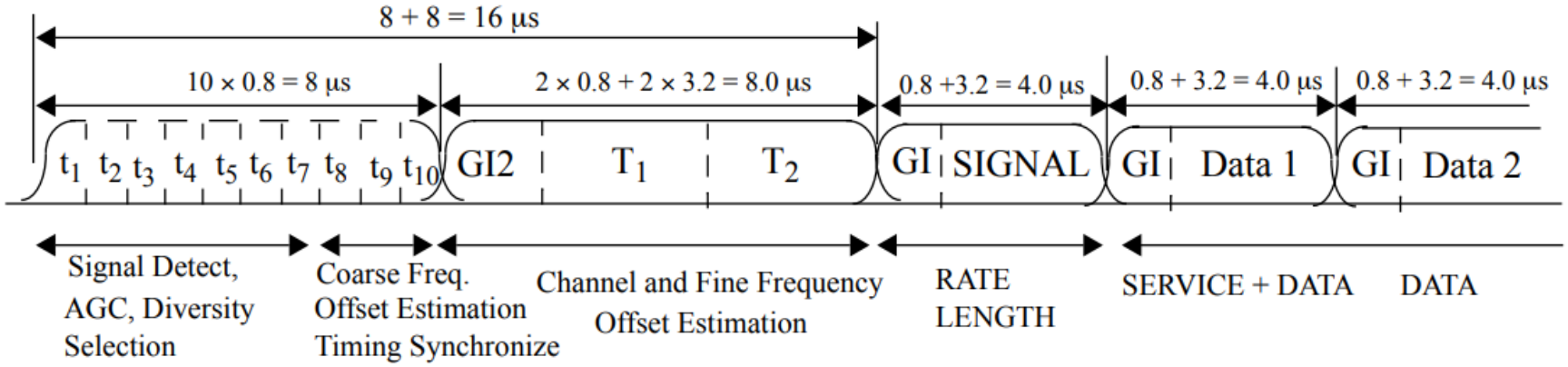


Ukázka – GUI přijímač



Ukázka teorie – detekce framu

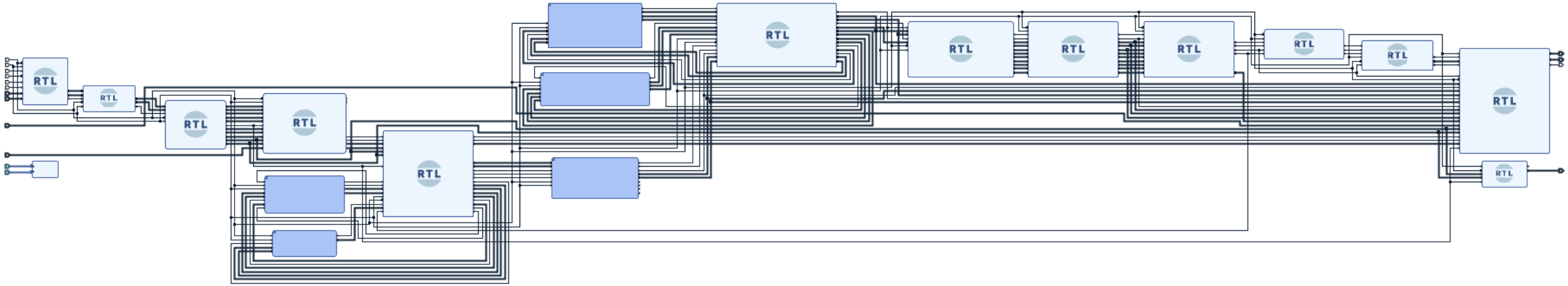
- Detekce framu za přítomnosti frekvenčního offsetu
- 802.11p preambule:



- Crosskorelace s celou synchronizační sekvencí: $|\mathcal{R}_{160}^{\mathrm{STS}}[n]|^2 = |\sum_{k=0}^{159} x[n-159+k]\bar{s}^{\mathrm{STS}}[k]|^2$
- Crosskorelace s jednou periodou: $|\tilde{\mathcal{R}}_{160}^{\mathrm{STS}}[n]|^2 = \sum_{i=1}^{10} |\mathcal{R}_{16}^{\mathrm{STS}}[n-16i]|^2$, kde $|\mathcal{R}_{16}^{\mathrm{STS}}[n]|^2 = |\sum_{k=0}^{15} x[n-15+k]\bar{s}^{\mathrm{STS}}[k]|^2$
- Rozdíl odolnosti cca: 15 kHz a 150 kHz

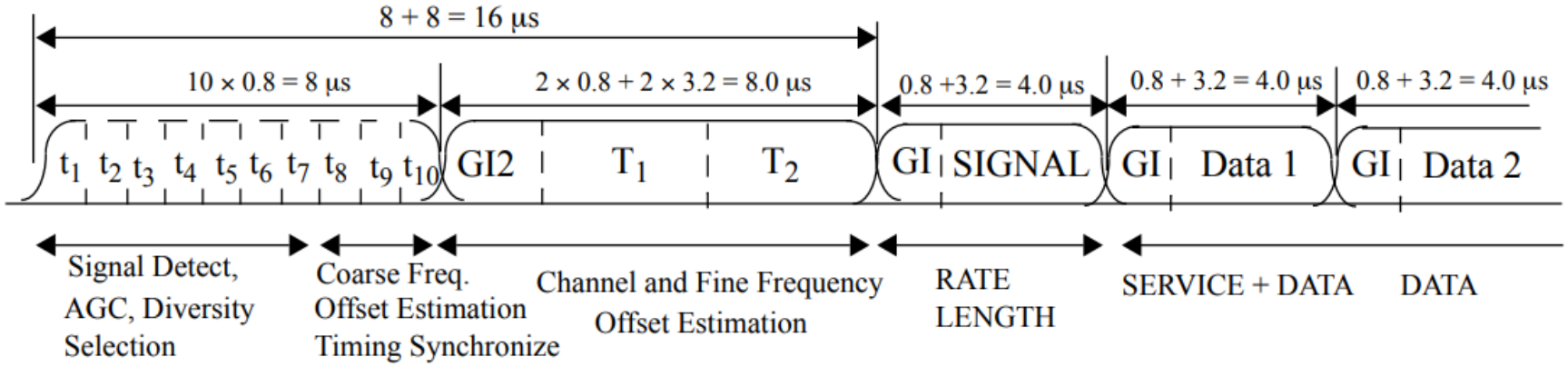
Blokové schéma práce (blok IP_802_11p)

- Vlastních 21 VHDL bloků, 6 Vivado IP bloků
- Příjem 802.11p PHY framů (povinné modulace a kódové rychlosti)
- Ukládání dat do 4 KB BRAM
- Python GUI
  - AXI4: konfigurace bloku a čtení dat z BRAM
  - Libiio: jednoduché vysílání



Ukázka teorie – detekce framu

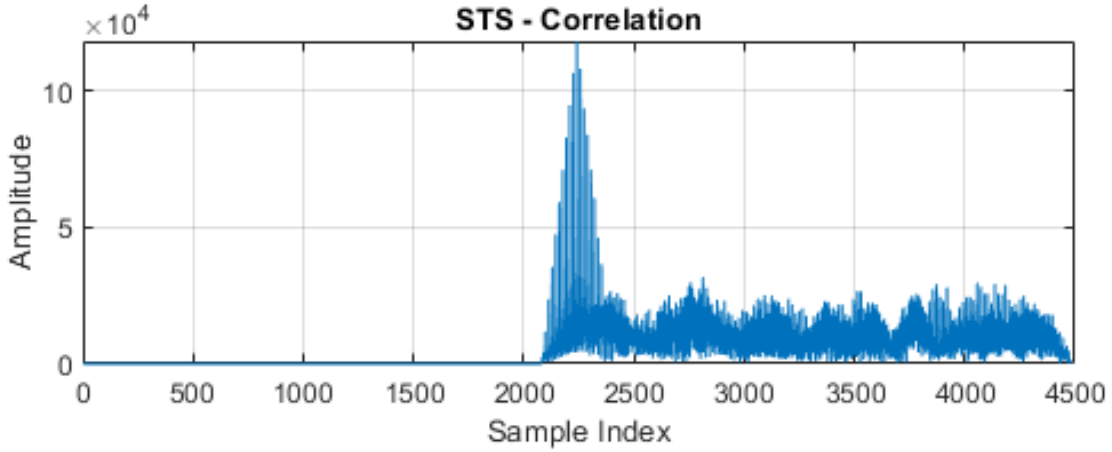
- Detekce framu za přítomnosti frekvenčního offsetu
- 802.11p preambule:

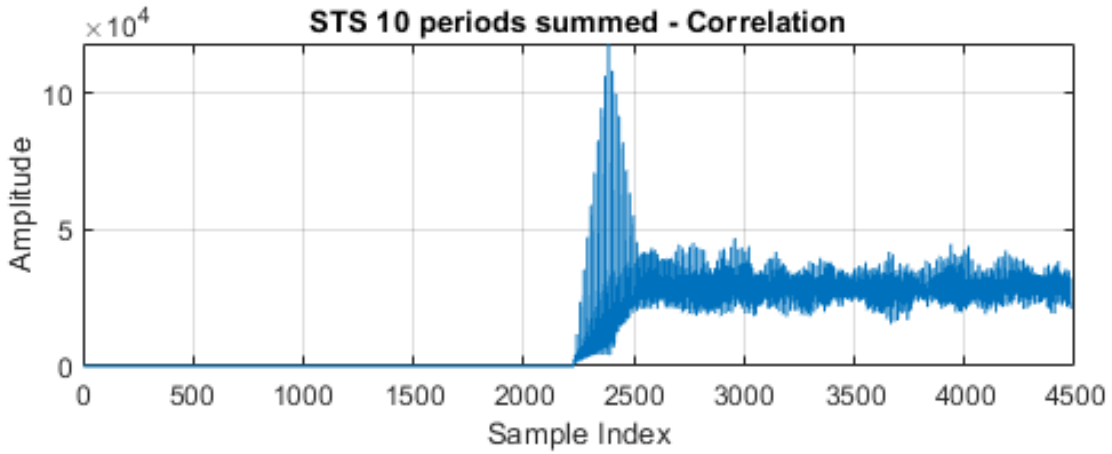


- Crosskorelace s celou synchronizační sekvencí: $|\mathcal{R}_{160}^{\mathrm{STS}}[n]|^2 = |\sum_{k=0}^{159} x[n-159+k]\bar{s}^{\mathrm{STS}}[k]|^2$
- Crosskorelace s jednou periodou: $|\tilde{\mathcal{R}}_{160}^{\mathrm{STS}}[n]|^2 = \sum_{i=1}^{10} |\mathcal{R}_{16}^{\mathrm{STS}}[n-16i]|^2$, kde $|\mathcal{R}_{16}^{\mathrm{STS}}[n]|^2 = |\sum_{k=0}^{15} x[n-15+k]\bar{s}^{\mathrm{STS}}[k]|^2$
- Rozdíl odolnosti cca: 15 kHz a 150 kHz

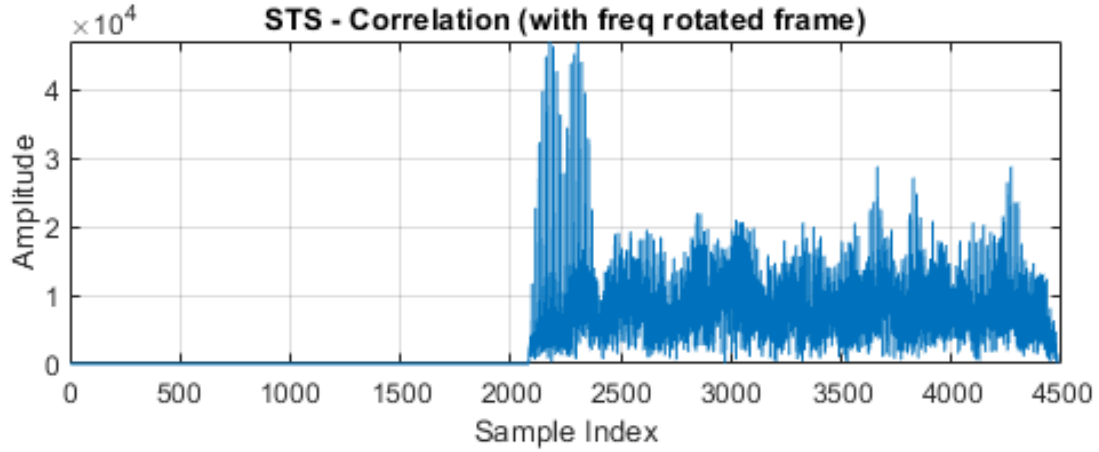
Ukázka teorie – detekce framu

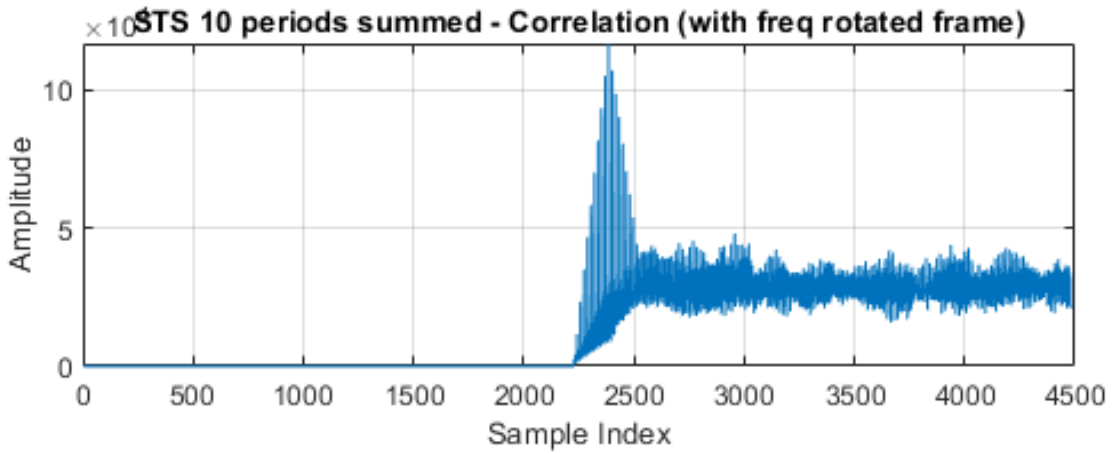
- Nulový frekvenční offset



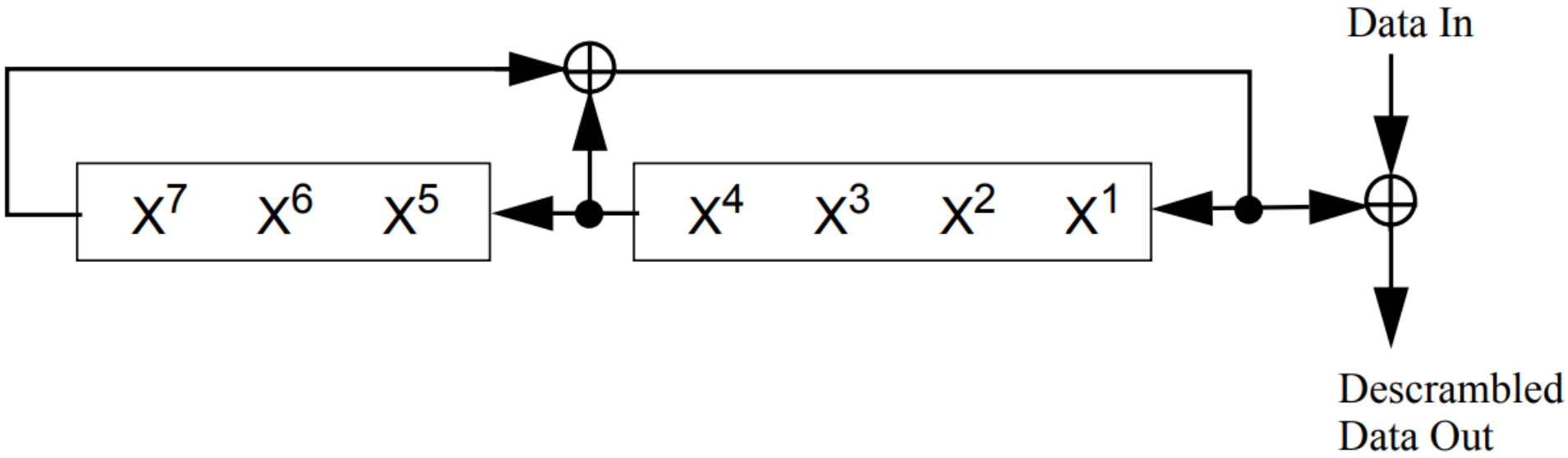


- 50 kHz frekvenční offset



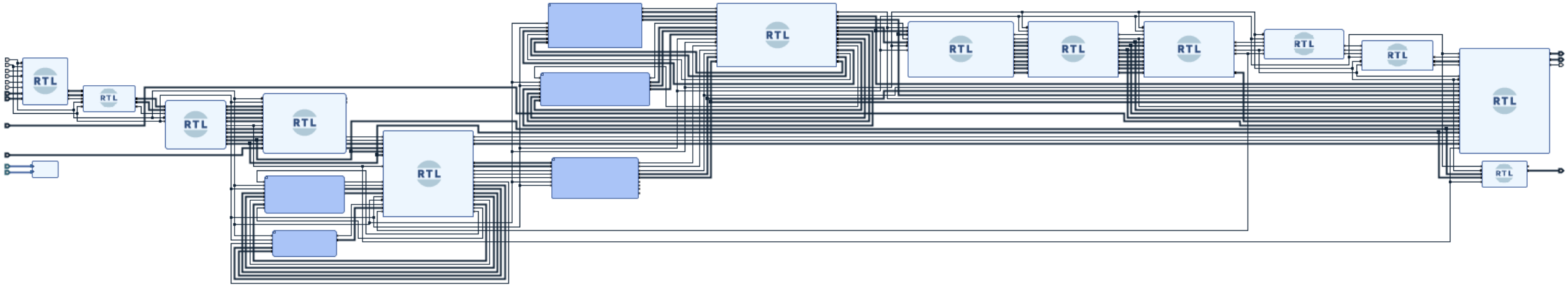


Ukázka implementace - descrambler



Blokové schéma práce (blok IP_802_11p)

- Vlastních 21 VHDL bloků, 6 Vivado IP bloků
- Příjem 802.11p PHY framů (povinné modulace a kódové rychlosti)
- Ukládání dat do 4 KB BRAM
- Python GUI
  - AXI4: konfigurace bloku a čtení dat z BRAM
  - Libiio: jednoduché vysílání



Ukázka implementace - descrambler

```
76  architecture Behavioral of descrambler is
77    ...
78  begin
79
80    state_update_process : process(CLOCK)
81      variable VAR_DESCRAMBLER_OUTPUT : std_logic := '0';
82    begin
83
84      if rising_edge(CLOCK) then
85
86        if RESET = '1' or VITERBI_SIGNAL_VALID = '1' then
87          STATE <= INIT;
88          COUNTER <= 0;
89          DESCRAMBLED_OUTPUT_VALID <= '0';
90          DESCRAMBLED_OUTPUT_LAST <= '0';
91
92        else
93          case STATE is
94            -- initialize the descramble by first 7 bits
95            when INIT=>
96              if VITERBI_DECODED_OUTPUT_VALID = '1' then
97                DESCRAMBLE_MEMORY <= (VITERBI_DECODED_OUTPUT & DESCRAMBLE_MEMORY(0 to DESCRAMBLE_MEMORY'LENGTH-2));
98                COUNTER <= COUNTER + 1;
99                if COUNTER = 6 then
100                 STATE <= DESCRAMBLE;
101               end if;
102             end if;
103             -- output 'descrambled' zeros
104             DESCRAMBLED_OUTPUT <= '0';
105
106           -- descramble all following data
107           when DESCRAMBLE =>
108             if VITERBI_DECODED_OUTPUT_VALID = '1' then
109               VAR_DESCRAMBLER_OUTPUT := DESCRAMBLE_MEMORY(6) xor DESCRAMBLE_MEMORY(3);
110               DESCRAMBLE_MEMORY <= (VAR_DESCRAMBLER_OUTPUT & DESCRAMBLE_MEMORY(0 to DESCRAMBLE_MEMORY'LENGTH-2));
111               -- output descrambled data
112               DESCRAMBLED_OUTPUT <= VITERBI_DECODED_OUTPUT xor VAR_DESCRAMBLER_OUTPUT;
113             end if;
114
115           when others =>
116             STATE <= INIT;
117
118         end case;
119         DESCRAMBLED_OUTPUT_VALID <= VITERBI_DECODED_OUTPUT_VALID;
120         DESCRAMBLED_OUTPUT_LAST <= VITERBI_RX_ENDED;
121       end if;
122     end if;
123
124   end process state_update_process;
125 end Behavioral;
```

Možná pokračování

- Oprava chyb a nepřesností (synchronizace, depuncturing , Viterbi, BRAM)
- Implementace MAC vrstvy
- Informování Kuiper Linuxu o příjmu pomocí přerušení
- Možná úprava PHY vrstvy pro podporu 802.11a a 802.11g (20 MHz pásmo)

Otázka 1

Jedním ze základních parametrů přijímače je jeho citlivost. Byla stanovena citlivost realizované implementace? Pokud ano, je v souladu s očekáváním? Pokud ne, stručně popište, jak byste její hodnotu pro daný kódový poměr a modulaci subnosné frekvence stanovil.

- Nebyla stanovena
- Viz sekce 4.4, synchronizace selhává na IQ hodnotách cca 30/2^15 (-61 dBFS, -72 dBm)
- Nutné vylepšení v budoucnu
- Měření chybovosti dekódovaných dat: pomocí známého paketu a různých útlumů na vysílači
- Možná implementace v Pythonu (GUI)

Otázka 2

V úvodu kapitoly 4 je uvedeno, že důvodem realizace příjmu v PL je především striktní požadavek na rychlé vyslání potvrzení příjmu. Umožňuje vyvinutý TestBed vysílat tato potvrzení? V textu práce toto není zdokumentováno.

- Neumožnuje
- Potvrzování příjmů je řešeno v MAC vrstvě
- TestBed je schopen přijmout poslední data cca 15 µs po skončení framu (simulace)
- Možná implementace v budoucnu

Děkuji za pozornost