Karlsruhe Institute of Technology

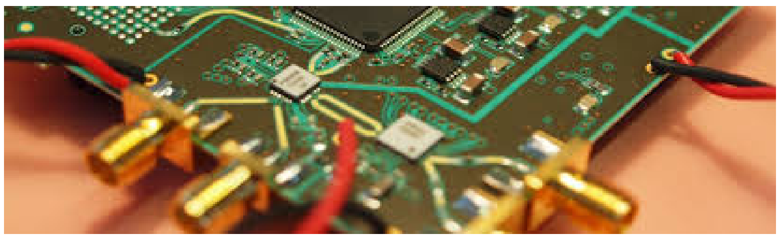
DEPARTMENT OF ELECTRICAL ENGINEERING
AND INFORMATION TECHNOLOGY
Institute for Data Processing and Electronics (IPE)

A Terabit Sampling System with a Photonics Time-Stretch ADC

Master Thesis
of

Olena Manzhura

at the Institute for Data Processing and Electronics (IPE)



Reviewer:	Prof. Dr. Anke-Susanne Müller (LAS)
Second Reviewer:	Dr. Michele Caselle (IPE)

15.11.2020 – 13.08.2021

KIT – Die Forschungsuniversität in der Helmholtz-Gemeinschaft www.kit.edu

Declaration

I hereby declare that I wrote my master thesis on my own and that I have followed the regulations relating to good scientific practice of the Karlsruhe Institute of Technology (KIT) in its latest form. I did not use any unacknowledged sources or means, and I marked all references I used literally or by content.

Karlsruhe, 13.08.2021, ____________________________

Olena Manzhura

Approved as an exam copy by

Karlsruhe, 13.08.2021, ____________________________

Prof. Dr. Anke-Susanne Müller (LAS)

Abstract

Zusammenfassung

Résumé

Contents

Acronyms **xvii**

1. Introduction **1**

2. Terabit Sampling System for Photonics Time-Stretch/Photonic time-stretch data acquisition system **3**
- 2.1. Photonic Front-End . . . 3
  - 2.1.1. Detector/Diode . . . 3
- 2.2. Analog-To-Digital Converters . . . 4
  - 2.2.1. Sampling Theory . . . 4
  - 2.2.2. Characteristics of Analog-To-Digital-Converters . . . 6
    - 2.2.2.1. Static parameters . . . 6
    - 2.2.2.2. Frequency-Domain Dynamic Parameters . . . 7
    - 2.2.2.3. Quantization Noise . . . 8
  - 2.2.3. Interleaving . . . 9

3. State Of The Art Read-Out-Systems **11**

4. Architecture of the new system **13**

5. Front-End Sampling Card - Terahertz Readout Sampling (THERESA) **15**

6. Readout Card - Xilinx ZCU216 **17**
- 6.1. General Description . . . 17
- 6.2. Features . . . 18

7. Firmware **19**
- 7.1. SoC . . . 19
- 7.2. RF Data Converter . . . 19
- 7.3. RDMA over Converged Ethernet (RoCE) . . . 19
- 7.4. System Integration (SPI) . . . 19

8. Conclusions and Outlook **21**
- 8.1. Measurements with the developed front-end card . . . 21

Acknowledgements **23**

Appendix **25**
- A. Characteristic Impedance Of Coplanar Waveguides . . . 25
- B. QuickStart Guide for Evaluation of ZCU216 Board . . . 26
- C. 3D model of front-end card . . . 26
- D. Code . . . 26

Bibliography **33**

List of Figures

2.1. Electro-Optical Time-Stretch Technique [SELP^{+}16] 3
2.2. Transfer function of ideal, 3-bit ADC 4
2.3. Aliasing 5
2.4. Track-And-Hold-Amplifier schematic and principle [Kes05] 6
2.5. Offset and Gain Error in the Analog-To-Digital-Converter (ADC) characteristic. Notice the difference between the ideal, dotted line 7
2.6. Quantization noise as function of time (redrawn from [Kes05]) 8

6.1. ZCU216 evaluation board 17
6.2. RFSoC block diagram 18

8.1. wow 21

List of Tables

Acronyms

ADC Analog-To-Digital-Converter. xiii, 3–6, 8

LSB Least Significant Bit. 4, 6, 7

SHA Sample-And-Hold-Amplifier. 4–6, 8

THA Track-And-Hold-Amplifier. 4

THERESA Terahertz Readout Sampling. xi, 13

1. Introduction

2. Terabit Sampling System for Photonics Time-Stretch/Photonic time-stretch data acquisition system

2.1. Photonic Front-End

The working principle of the optic time-stretch technique can be described in two steps (see Figure 2.1). First, a short laser pulse is propagated in a dispersive medium (optical fiber of length L_1). This results in a chirped laser pulse of the duration

$$T_1 = \Delta\lambda D_1 L_1 \tag{2.1}$$

with the optical bandwidth of the laser pulse $\Delta\lambda$ and the dispersion parameter D_1 of the fiber. The next step is the time-to-wavelength-mapping, where a temporal intensity modulation is imprinted on the chirped pulse. After that, the modulated chirped pulse propagates through another dispersive medium, a fiber of the length L_2. In this way, the temporal modulation of the pulse is further stretched to the duration T_2, which is long enough for detection with photodetectors and/or digitizing with ADCs. [Rou14]

The factor, by which the pulse is slowed down, is calculated as

$$M = 1 + \frac{L_1}{L_2}. \tag{2.2}$$

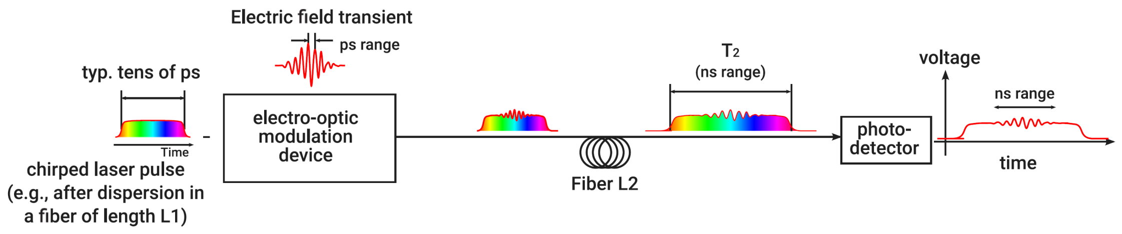


Figure 2.1.: Electro-Optical Time-Stretch Technique [SELP+16]

2.1.1. Detector/Diode

The detection and subtraction between the two stretched signals is performed using a amplified balanced photodetector (photoreceiver) from Discovery Semiconductors, with 20 GHz bandwidth and 2800 V/W gain (specified at 1500 nm). The two differential outputs of the detector are sent on a Lecroy LabMaster 10i oscilloscope with 36 GHz bandwidth, 80 GS/s sample rate on each channel and a memory of 256 Mega samples.

2.2. Analog-To-Digital Converters

ADCs are used to translate analog quantities into digital signals, which can be processed by information processing, computing, data transmission and control systems. The translation can be seen as encoding a continuous-time analog input (voltage) into a series of discrete, N-bit words. This process, which is also called *sampling*, can be expressed as

$$V_{\mathrm{In}} = V_{\mathrm{FS}} \sum_{k=0}^{N-1} \frac{b_k}{2^{k+1}} + \epsilon \tag{2.3}$$

with V_{In} being the input voltage, V_{FS} the full-scale voltage, b_k the individual output bits and ϵ the quantization error. Figure 2.2 shows the ideal transfer function of a 3-bit ADC.

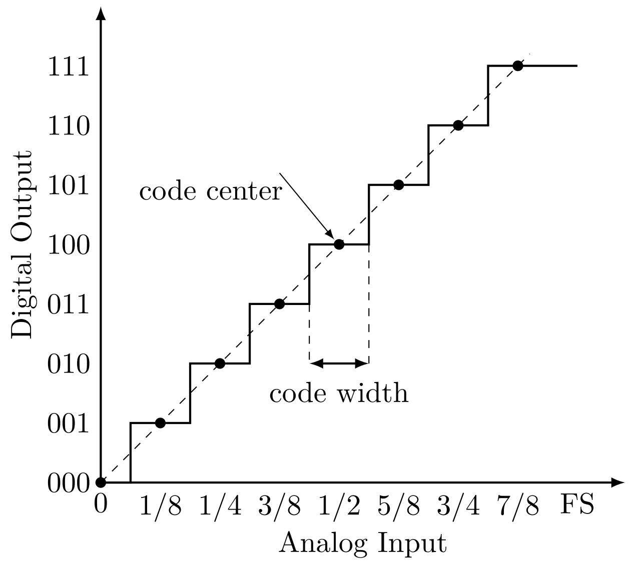


Figure 2.2.: Transfer function of an ideal, 3-bit ADC (redrawn from [LV02])

2.2.1. Sampling Theory

An ADC samples an analog signal with a sample frequency f_s. This frequency has to be chosen in such way, that the original signal can be fully reconstructed. The *Nyquist criteria* states, that in order to accurately represent a band-limited, continuous signal

$$y(t) \circ\!\!-\!\!- Y(f) \quad \text{with} \quad Y(f) = 0|_{f>B/2} \tag{2.4}$$

it has to be sampled with a frequency f_s respecting

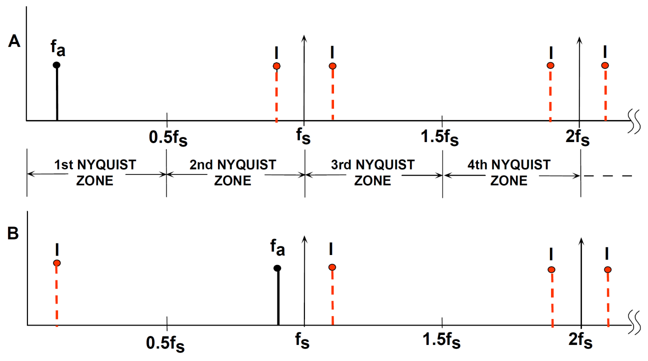
$$f_s > B \quad \text{or} \quad f_s > 2f_a \tag{2.5}$$

with f_a being the highest frequency contained in the signal. [Kes05, Pue15]

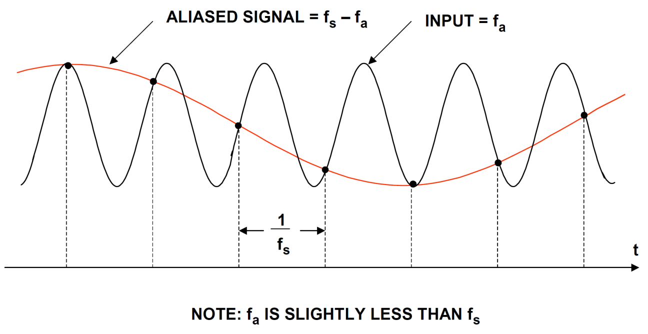
Violation of this rule leads to *aliasing*.

Sample-And-Hold-Amplifier

ADCs need a certain amount of time to sample the input signal. If the level of the analog signal changes by more than one Least Significant Bit (LSB) during this period, this can result in large errors in the output signal. Therefore, so called Sample-And-Hold-Amplifier (SHA) are used in front of the ADC to hold the input level constant for the needed amount of time. The ADC sampling time needs to be timed in such way, that the analog-to-digital



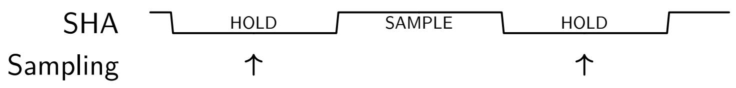
(a) Sampling in frequency domain


ALIASED SIGNAL = $f_s - f_a$
INPUT = f_a
$\frac{1}{f_s}$
t
NOTE: f_a IS SLIGHTLY LESS THAN f_s

(b) Aliasing in time domain

Figure 2.3.: Analog signal with frequency f_a sampled at f_s respecting (A) and not respecting (B) the Nyquist criteria. Figure 2.3b shows the effect of case B in time domain. [Kes05]

conversion falls into the hold period of the SHA and does not exceed into the sample period, for example like shown schematically in the diagram below. Thus, the upper frequency limitation is not determined by the ADC itself, but rather by the aperture jitter, bandwidth, distortion, etc. of the SHA. [Kes05]



In addition to the SHA, there is also the Track-And-Hold-Amplifier (THA). Instead of a sample period, the THA has a track period, where the output of the amplifier tracks the input signal (see also Figure 2.4). When switching to hold mode, the signal at this instant is held. This is opposed to the SHA, where the output during sample mode is actually not defined and is set to the value of the input signal, only when switching into hold mode.

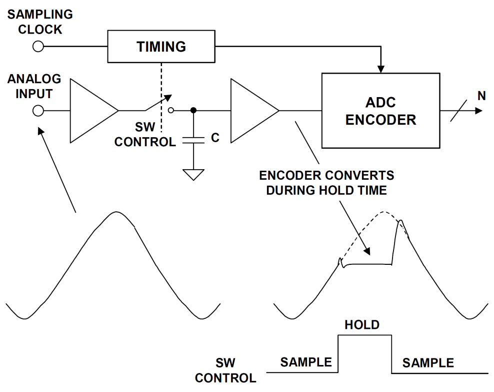


Figure 2.4.: Track-And-Hold-Amplifier schematic and principle [Kes05]

2.2.2. Characteristics of Analog-To-Digital-Converters

For an ideal converter, the number of bits would be sufficient to fully characterize it. Real ADCs however differ from the ideal behavior by introducing static and dynamic imperfections. Different applications have different requirements, which leads to a number of specifications. These can be divided into three categories [LV02]:

- Static parameters
- Frequency-domain dynamic parameters
- Time-domain dynamic parameters

This section provides an overview of these figures of merit. Which of these are needed to specify the necessary performance of the ADC has to be chosen for each application accordingly.

2.2.2.1. Static parameters

Static parameters are specifications, which can be measured at low speed/DC.

Accuracy

Accuracy is the total error at a known voltage, which includes:

- Quantization error
- Gain error
- Offset error
- Non-linearities

Resolution

Resolution is the number of bits N of the ADC. Depending from the resolution are the size of the LSB, which in its turn determines the dynamic range, code widths and quantization error.

Dynamic Range

Ratio between smallest possible output (LSB voltage) and the largest possible output (full-scale voltage). It can be calculated as

$$20 \log 2^N \approx 6N. \tag{2.6}$$

Offset and Gain Error

The *offset error* is the deviation of the first transition voltage from the ideal 1/2 LSB. *Gain Error* defines the deviation of the slope of the line going through the zero and full-scale point of the transfer function. These errors can easily be corrected by calibration. Refer to Figure 2.5

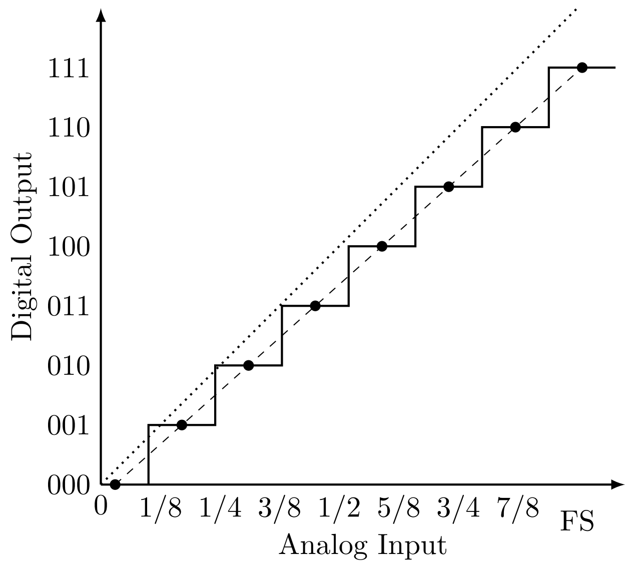


Figure 2.5.: Offset and Gain Error in the ADC characteristic. Notice the difference between the ideal, dotted line

Integral and Differential Non-linearity Distortion

Integral non-linearity in the transfer function is the distance of the code centers from the ideal line. It results from the integral non-linearities of the front-end, SHA and also the ADC itself. sThese non-linearities depend on the input signal amplitude.

Differential non-linearitiy is the deviation in code transition width from the ideal 1 LSB width. This nonlinearity stems exclusively from the encoding process in the ADC. It not only depends on the input signal amplitude, but also on the positioning along the transfer function.

2.2.2.2. Frequency-Domain Dynamic Parameters

Spurious-Free Dynamic Range (SINAD)

- SINAD
- ENOB
- SFDR
- Total Harmonic Distortion
- Intermodulation Distortion
- Effective Resolution Bandwidth
- Full-Power Bandwidth
- Full-Linear Bandwidth

Time-Domain

- Aperture Delay
- Aperture Jitter
- Transient Response
- Overvoltage Recovery

2.2.2.3. Quantization Noise

Even in an ideal N-bit converter there will be errors during the quantization, which behave like noise. The reason is that each N-bit word represents a certain range of analog input values, which is 1 LSB wide (*code width*) and centered around a *code center* (see Figure 2.2) [LV02]. The input voltage is always assigned to the code of the nearest code center. This means that there will always be a difference between the code center and the actual input. The difference betweenand the analog input $x(t)$ and its quantized signal $x_q(t)$ is called the *quantization error*. For an equidistant quantization it is

$$|e_q(t)| = |x(t) - x_q(t)| \leq \frac{q}{2} \tag{2.7}$$

[Pue15]

Assuming the error voltage uncorrelated and uniformly distributed, the theoretical (maximum) Signal-To-Noise-Ratio (SNR) of this *quantization noise* can be calculated. In the time domain, the quantization error can be approximated with a sawtooth signal [Kes05]:

$$e_q(t) = st, \quad -\frac{q}{2s} < t < \frac{q}{2s} \tag{2.8}$$

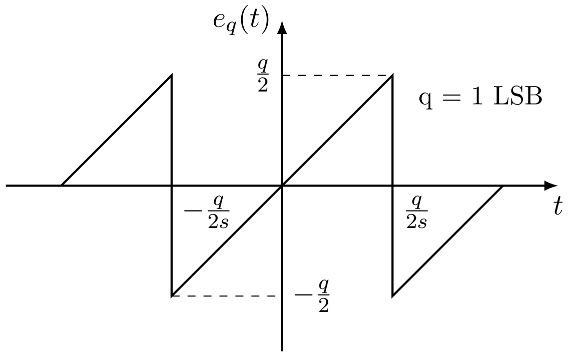


Figure 2.6.: Quantization noise as function of time (redrawn from [Kes05])

The power of the quantization noise, which is assumed to be uncorrelated and broadband, can be calculated as the mean-square of $e_q(t)$:

$$P_{\mathrm{QN}} = e_{\mathrm{rms}}^2 = \overline{e^2(t)} = \frac{s}{q}\int_{-q/2s}^{+q/2s} (st)^2 dt = \frac{s^3}{q}\left[\frac{t^3}{3}\right]_{-\frac{q}{2s}}^{+\frac{q}{2s}} = \frac{q^2}{12} \tag{2.9}$$

To calculate the maximal SNR of an ideal converter, a full-scale input sine wave is assumed:

$$u(t) = u_s \sin(2\pi f t) = \frac{2^N q}{2}\sin(2\pi f t) = 2^{N-1} q \sin(2\pi f t) \tag{2.10}$$

With the effective value of the signal amplitude

$$u_{\text{eff}} = \frac{u_s}{\sqrt{2}} = \frac{2^{N-1}q}{\sqrt{2}} \tag{2.11}$$

and the quantization noise power, the SNR can be calculated as

$$\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} = \frac{u_{\text{eff}}^2}{e_{\text{rms}}^2} = \frac{2^{2N-2}q^2/2}{q^2/12} = 2^{2N} \cdot 1.5. \tag{2.12}$$

Using the unit Decibel, this can be expressed as (see [Pue15, Kes05])

$$\text{SNR}|_{\text{dB}} = 10 \log\left(2^{2N} \cdot 1.5\right) = 6.02N + 1.76. \tag{2.13}$$

In an ADCs (with built-in SHA) there are a couple of sources, which introduce noise and distortion:

- **Input Stage:** Wideband noise, nonlinearity and bandwidth limitation
- **SHA:** Nonlinearity, aperture jitter[1] and bandwidth limitation
- **ADC:** Quantization noise, integral and differential nonlinearity

In the following the most important specifications are described, which are used to characterize the performance of ADCs.

Equivalent Input Referred Noise

Internal circuits of wideband ADCs produce rms noise due to resistor and thermal ("kT/C") noise, which is also present for DC signals. Therefore, the output of the ADC is a distribution of codes which is centered around the value of a DC input. For measuring the value of the noise, the ADC input is grounded, or held at a specific DC value, and a large amount of samples is collected and plotted as a histrogram. The noise is approximately Gaussian, thus the stndard deviation can easily be calculated.

Noise-Free Code Resolution

The input referred noise described above determines the *noise-free code resolution*, which is the number of bits, beyodn which it is not possible to resolve individual codes. [Kes05]

2.2.3. Interleaving

- Net sample rate
- Interleaving Spurs

[1] *Aperture jitter* is the sample-to-sample variation in aperture delay, with *aperture delay* meaning the time, which is needed by the SHA to disconnect the holding capacitor from the input buffer.

3. State Of The Art Read-Out-Systems

4. Architecture of the new system

5. Front-End Sampling Card - THERESA

6. Readout Card - Xilinx ZCU216

6.1. General Description

Xilinx Zynq UltraScale+ RFSoC ZCU216 Evaluation Card

Zynq UltraScale+ RFSoCs: Combine RF data converter subsystem and forward error correction with industry-leading programmable logic and heterogeneous processing capability. Integrated RF-ADCs, RF-DACs, and soft decision FECs (SD-FEC) provide the key subsystems for multiband, multi-mode cellular radios and cable infrastructure

The card, holding the sampling channels, should be mounted on a ZCU216 evaluation board. This board is the newest generation of Xilinx' evaluation cards, which has features suitable for the purpose at hand.

- Sixteen 14-bit, 2.5GSPS RF-ADC
- Sixteen 14-bit, 10GSPS RF-DAC
- I/O expansion options – FPGA Mezzanine Card (FMC+) interfaces, RFMC 2.0 interfaces, and Pmod connections

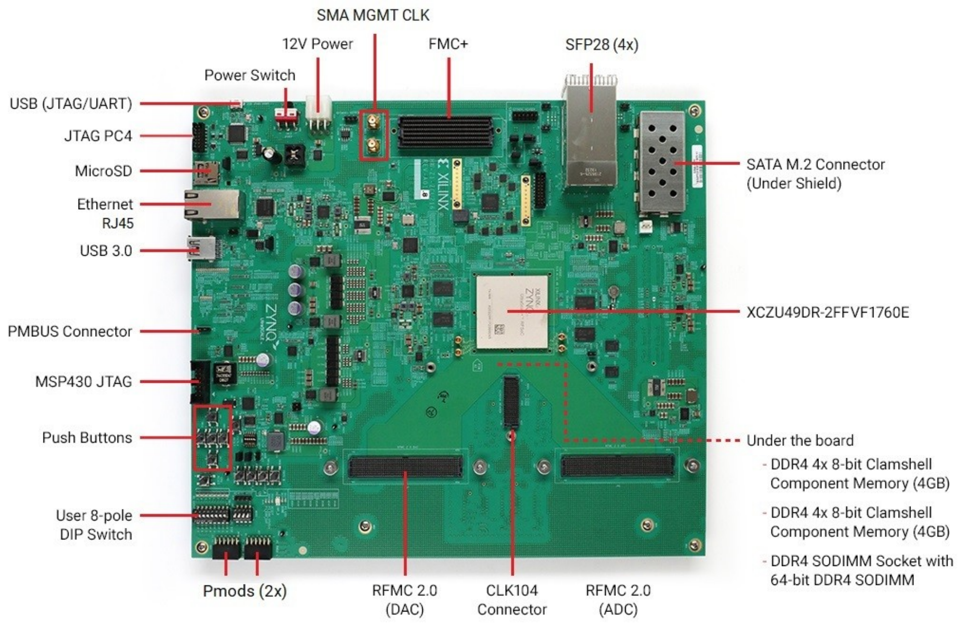


Figure 6.1.: ZCU216 evaluation board

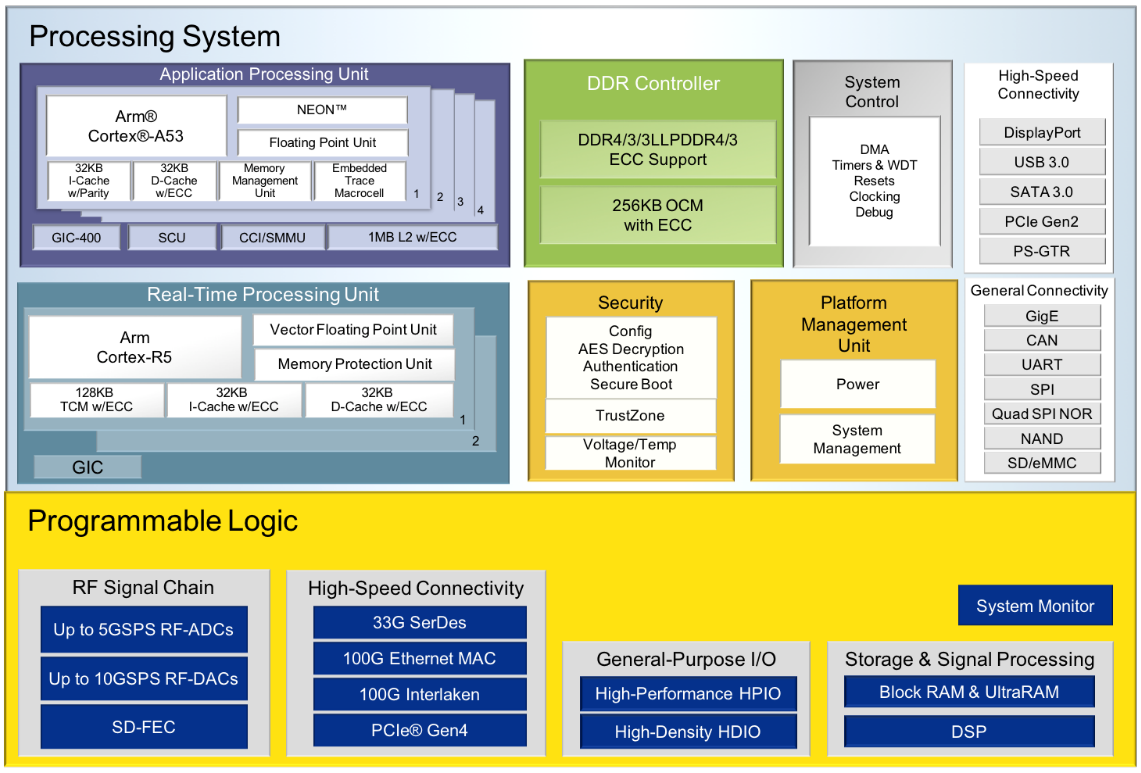


Figure 6.2.: RFSoC block diagram

6.2. Features

7. Firmware

7.1. SoC

7.2. RF Data Converter

7.3. RDMA over Converged Ethernet (RoCE)

7.4. System Integration (SPI)

8. Conclusions and Outlook

8.1. Measurements with the developed front-end card

Card is in production. Measurements will be done, but not in the scope of the thesis.

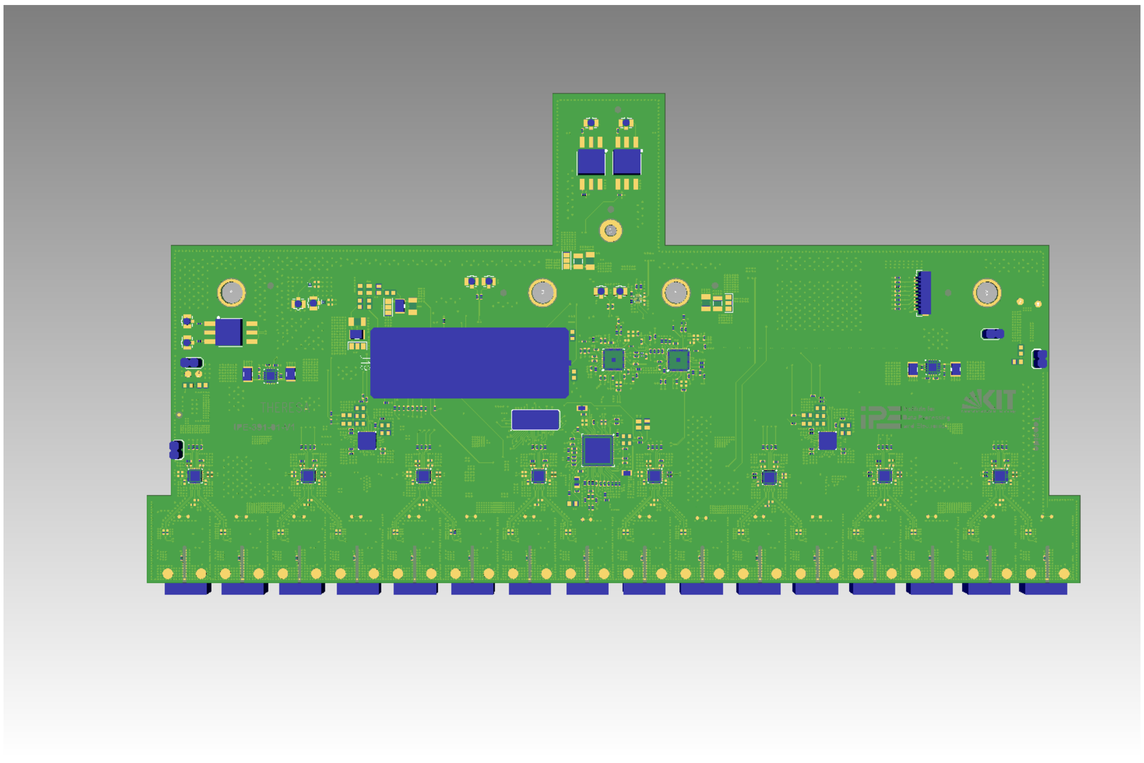


Figure 8.1.: wow

Acknowledgments

Appendix

A. Characteristic Impedance Of Coplanar Waveguides

Surface Coplanar Waveguide with Ground

Characteristic impedance[Wad91, p197-198]:

$$Z_{0,o} = \frac{\eta_0}{\sqrt{\epsilon_{eff,o}}} \left(\frac{1.0}{2.0\frac{K(k_o)}{K'(k_o)} + \frac{K(\beta_1)}{K'(\beta_1)}} \right) \tag{A.1}$$

$$Z_{0,e} = \frac{\eta_0}{\sqrt{\epsilon_{eff,e}}} \left(\frac{1.0}{2.0\frac{K(k_e)}{K'(k_e)} + \frac{K(\beta_1 k_1)}{K'(\beta_1 k_1)}} \right) \tag{A.2}$$

$$\epsilon_{eff,o} = \frac{2.0\epsilon_r \frac{K(k_o)}{K'(k_o)} + \frac{K(\beta_1)}{K'(\beta_1)}}{2.0\frac{K(k_o)}{K'(k_o)} + \frac{K(\beta_1)}{K'(\beta_1)}} \tag{A.3}$$

$$\epsilon_{eff,e} = \frac{2.0\epsilon_r \frac{K(k_e)}{K'(k_e)} + \frac{K(\beta_1 k_1)}{K'(\beta_1 k_1)}}{2.0\frac{K(k_e)}{K'(k_e)} + \frac{K(\beta_1 k_1)}{K'(\beta_1 k_1)}} \tag{A.4}$$

Where

$$k_o = \Lambda \frac{-\sqrt{\Lambda^2 - t_c^2} + \sqrt{\Lambda^2 - t_B^2}}{t_B\sqrt{\Lambda^2 - t_c^2} + t_c\sqrt{\Lambda^2 - t_B^2}} \tag{A.5}$$

$$k_e = \Lambda' \frac{-\sqrt{\Lambda'^2 - t_c'^2} + \sqrt{\Lambda'^2 - t_B'^2}}{t_B'\sqrt{\Lambda'^2 - t_c'^2} + t_c'\sqrt{\Lambda'^2 - t_B'^2}} \tag{A.6}$$

$$\Lambda = \frac{\sinh^2\left(\frac{\pi(s/2.0+w+d)}{2.0h}\right)}{2} \tag{A.7}$$

$$t_c = \sinh^2\left(\frac{\pi(s/2.0+w)}{2.0h}\right) - \Lambda \tag{A.8}$$

$$t_B = \sinh^2\left(\frac{\pi s}{4.0h}\right) - \Lambda \tag{A.9}$$

$$\Lambda' = \frac{\cosh^2\left(\frac{\pi(s/2.0+w+d)}{2.0h}\right)}{2} \tag{A.10}$$

$$t'_c = \sinh^2\left(\frac{\pi(s/2.0+w)}{2.0h}\right) - \Lambda' + 1.0 \tag{A.11}$$

$$t'_B = \sinh^2\left(\frac{\pi s}{4.0h}\right) - \Lambda + 1.0 \tag{A.12}$$

The parameters have to be chosen according to

$$s + 2.0w + 2.0d \leq h \tag{A.13}$$

to guarantee coplanar propagation. [Wad91]

Surface Coplanar Waveguide with Ground

The characteristic impedance of a coplanar waveguide is given as follows [Wad91]:

$$Z_0 = \frac{60.0\pi}{\sqrt{\epsilon_{eff}}} \frac{1.0}{\frac{K(k)}{K(k')} + \frac{K(k_1)}{K(k'_1)}} \tag{A.14}$$

It comprises of the following components, with K(k) being an elliptical integral of the first kind (see also [BSMM99, p. 430]):

$$k = a/b \tag{A.15}$$

$$k' = \sqrt{1.0 - k^2} \tag{A.16}$$

$$k'_1 = \sqrt{1.0 - k_1^2} \tag{A.17}$$

$$k_1 = \frac{tanh(\frac{\pi a}{4.0h})}{tanh(\frac{\pi b}{4.0h})} \tag{A.18}$$

$$\epsilon_{eff} = \frac{1.0 + \epsilon_r \frac{K(k')}{K(k)} \frac{K(k_1)}{K(k'_1)}}{1.0 + \frac{K(k')}{K(k)} \frac{K(k_1)}{K(k'_1)}} \tag{A.19}$$

B. QuickStart Guide for Evaluation of ZCU216 Board

C. 3D model of front-end card

D. Code

```
`timescale 1ns / 1ps

module SDI_Delay_NB6L295(

    input [10:0]        In_1, In_2, In_3, In_4, In_5, In_6, In_7, In_8,  //
        data for respective delay chips
    input               Clk,
    input               Reset,
    output reg          [7:0] EN, // enable signal for delay chips, active LOW
    output reg          SDIN, // configuration data
    output reg          SLOAD, // signals delay chip to load previously sent
        data
    output              SCLK // clock for serial communication with delay chips
);

    reg                 start_clk;
    assign SCLK = start_clk & (!Clk);
```

```
reg [21:0]          In_1_reg, In_2_reg, In_3_reg, In_4_reg, In_5_reg,
    In_6_reg, In_7_reg, In_8_reg; // registers to intermediately store the
    inputs

reg [7:0]           select; // register used by Priority Encoder to detect
    which input changed

parameter           DATA_SHIFT_WIDTH = 11; // number of bits to be shifted
    during transmission, 1 Data word = 11 bits
reg [4:0]           clk_cnt;


reg [DATA_SHIFT_WIDTH-1:0]  Data_reg;  // register for storing data for
    state machine

reg                 start; // signal for state machine to start sending
    data
reg                 dataSent; // flags if transmission for one delay chip
    is finished

parameter           dly = 1; // delay control

reg                 delayReady;


always @ (posedge Clk)
begin
    if (select == 'd0)      delayReady <= #dly 'b1;
    else                    delayReady <= #dly 'b0;
end

// Priority Encoder
// Check if any input has changed, select which data should be sent
    accordingly
always @ (posedge Clk)
    begin
        if (Reset)
            begin
                In_1_reg        <= #dly 'd0;
                In_2_reg        <= #dly 'd0;
                In_3_reg        <= #dly 'd0;
                In_4_reg        <= #dly 'd0;
                In_5_reg        <= #dly 'd0;
                In_6_reg        <= #dly 'd0;
                In_7_reg        <= #dly 'd0;
                In_8_reg        <= #dly 'd0;
                Data_reg        <= #dly 'd0;

                select          <= #dly 'd0;

                start           <= #dly 1'b0;;
            end
        else
            begin
                if (~start & delayReady)
                  begin
                        select[7]    <= #dly In_1_reg != In_1;
                        select[6]    <= #dly In_2_reg != In_2;
                        select[5]    <= #dly In_3_reg != In_3;
                        select[4]    <= #dly In_4_reg != In_4;
                        select[3]    <= #dly In_5_reg != In_5;
                        select[2]    <= #dly In_6_reg != In_6;
                        select[1]    <= #dly In_7_reg != In_7;
                        select[0]    <= #dly In_8_reg != In_8;
    end
else
   begin
      if (clk_cnt == 4'd12 & ~start_clk) // = end of
         sequence
            start                  <= #dly 1'b0;
         else
            start                  <= #dly 1'b1;
   end

  casex (select)
      8'b1???????:
         begin
           if (~dataSent)
             begin
                  In_1_reg            <= #dly In_1;
                  Data_reg            <= #dly In_1;
                  EN                  <= #dly
                      8'b01111111;
                  start               <= #dly 1'b1;
                  end
             else
               begin
                  start               <= #dly 1'b0;
                  select[7]           <= #dly 1'b0;
               end
           end
      8'b01??????:
           begin
             if (~dataSent)
               begin
                  In_2_reg              <= #dly In_2;
                  Data_reg              <= #dly In_2;
                  EN                    <= #dly
                      8'b10111111;
                  start                 <= #dly 1'b1;
                end
              else
                begin
                  select[6]             <= #dly 1'b0;
                  start                 <= #dly 1'b0;
                end
           end
      8'b001?????:
           begin
           if (~dataSent)
              begin
                 In_3_reg               <= #dly In_3;
                 Data_reg               <= #dly In_3;
                 EN                     <= #dly
                     8'b11011111;
                 start                  <= #dly 1'b1;
              end
            else
                 begin
                     select[5]          <= #dly 1'b0;
                     start              <= #dly 1'b0;
                  end
           end
      8'b0001????:
           begin
           if (~dataSent)
              begin
                 In_4_reg               <= #dly In_4;
                Data_reg                <= #dly  In_4;
                EN                      <= #dly
                    8'b11101111;
                start                   <= #dly  1'b1;
            end

          else
                 begin
                     select[4]          <= #dly  1'b0;
                     start              <= #dly  1'b0;
                  end
        end
8'b00001???:
        begin
        if (~dataSent)
            begin
                In_5_reg                <= #dly  In_5;
                Data_reg                <= #dly  In_5;
                EN                      <= #dly
                    8'b11110111;
                start                   <= #dly  1'b1;
            end

          else
                 begin
                     select[3]          <= #dly  1'b0;
                     start              <= #dly  1'b0;
                  end
        end
8'b000001??:
        begin
        if (~dataSent)
            begin
                In_6_reg                <= #dly  In_6;
                Data_reg                <= #dly  In_6;
                EN                      <= #dly
                    8'b11111011;
                start                   <= #dly  1'b1;
            end

          else
                 begin
                     select[2]          <= #dly  1'b0;
                     start              <= #dly  1'b0;
                  end
        end
8'b0000001?:
        begin
          if (~dataSent)
            begin
                In_7_reg                <= #dly  In_7;
                Data_reg                <= #dly  In_7;
                EN                      <= #dly
                    8'b11111101;
                start                   <= #dly  1'b1;
            end
           else
                 begin
                     select[1]          <= #dly  1'b0;
                     start              <= #dly  1'b0;
                  end
        end
8'b00000001:
        begin
```

```
                        if (~dataSent)
                          begin
                            In_8_reg          <= #dly In_8;
                            Data_reg          <= #dly In_8;
                            EN                <= #dly
                               8'b11111110;
                            start             <= #dly 1'b1;
                          end
                         else
                             begin
                               select[0]      <= #dly 1'b0;
                               start          <= #dly 1'b0;
                             end
                      end
                  default:
                      begin
                         EN                   <= #dly
                            8'b11111111;
                         start                <= #dly 1'b0;
                      end
              endcase
           end
  end


// State Machine for Sending Configuration Data to Delay Chip NB6L295
/*
    State                    Description
    ______________________________________________________________
    RESET                    Resetting all parameters and registers ->
        if (reset): stay; else: to IDLE
    IDLE                     Waiting for start signal from priority
        encoder -> if (start): to LOAD_P0; else: stay
    LOAD_P0                  Load first half of Delay_X - which
        corresponds to data for Delay PD0 on delay chip - into
        temporary register -> to LOAD_P1
    LOAD_P1                  Load second half of Delay_X - which
        corresponds to data for Delay PD1 on delay chip - into
        temporary register -> to SHIFT
    SHIFT                    Shift bits for sending serial bitstream to
        SDIN, assert SLOAD -> to END
    END                      End transmission, deassert SLOAD, inform
        priority encoder about end of transmission -> to IDLE

*/
parameter RESET       = 3'd0;
parameter IDLE     = 3'd1;
parameter LOAD     = 3'd2;
parameter SHIFT    = 3'd3;
parameter END      = 3'd4;
reg [2:0] STATE;
reg [DATA_SHIFT_WIDTH-1:0]      tmp;

always @ (posedge Clk)
begin
    if (Reset)
        begin
            STATE       <= #dly RESET;
            tmp         <= #dly 'd0;
            dataSent    <= #dly 1'b0;
            start_clk   <= #dly 1'b0;
            SLOAD       <= #dly 1'b0;
            clk_cnt     <= #dly 1'b0;
        end
        else
            begin
                case (STATE)
                    RESET:
                        begin
                            if (Reset)
                                STATE     <= #dly RESET;
                            else
                                STATE     <= #dly IDLE;
                        end // RESET
                    IDLE:
                        begin
                            SDIN          <= #dly 1'b0;
                            clk_cnt       <= #dly 5'd0;
                            dataSent      <= #dly 1'b0;
                            SLOAD         <= #dly 1'b0;

                            if (start & ~dataSent)
                                STATE     <= #dly LOAD;
                            else
                                STATE     <= #dly IDLE;
                        end // IDLE
                    LOAD:
                        begin
                            tmp           <= #dly Data_reg;
                            STATE         <= #dly SHIFT;
                        end // LOAD_W1
                    SHIFT:
                        begin
                            if (clk_cnt < 4'd12) // number of bits to be
                                shifted //
                                begin
                                    start_clk        <= #dly 1'b1;
                                    clk_cnt          <= #dly clk_cnt +1;
                                    tmp              <= #dly
                                        {tmp[DATA_SHIFT_WIDTH-2:0], 1'b0};
                                    SDIN             <= #dly
                                        tmp[DATA_SHIFT_WIDTH-1];
                                end
                            else
                                begin
                                    SLOAD            <= #dly 1'b1;
                                    clk_cnt                     <= #dly
                                        clk_cnt;
                                    start_clk        <= #dly 1'b0;
                                    STATE            <= #dly END;
                                    SDIN             <= #dly 1'b0;
                                end
                        end // SHIFT
                    END:
                        begin
                            SLOAD              <= #dly 1'b0;
                            start_clk          <= #dly 1'b0;
                            dataSent           <= #dly 1'b1;
                            clk_cnt                <= #dly clk_cnt;
                            SDIN               <= #dly 1'b0;
                            STATE              <= #dly IDLE;
                        end // END
                    default:
                        STATE     <= #dly RESET;
                endcase
            end
    end
```

endmodule

Bibliography

[BBB$^+$19] Bielawski, Serge, Edmund Blomley, Miriam Brosi, Erik Bründermann, Eva Burkard, Clément Evain, Stefan Funkner, Nicole Hiller, Michael Nasse, Gudrun Niehues und et al.: *From self-organization in relativistic electron bunches to coherent synchrotron light: observation using a photonic time-stretch digitizer*. Scientific Reports, 9(1), July 2019. `http://dx.doi.org/10.1038/s41598-019-45024-2`.

[Bro20] Brosi, Miriam: *In-Depth Analysis of the Micro-Bunching Characteristics in Single and Multi-Bunch Operation at KARA*. Dissertation, Karlsruher Institut für Technologie (KIT), 2020. 54.01.01; LK 01.

[BSMM99] Bronstein, Semendjajew, Musiol und Mühlig: *Taschenbuch der Mathematik*. Verlag Harri Deutsch, 1999.

[CAB$^+$17] Caselle, M., L.E. Ardila Perez, M. Balzer, A. Kopmann, L. Rota, M. Weber, M. Brosi, J. Steinmann, E. Bründermann und A. S. Müller: *KAPTURE-2. A picosecond sampling system for individual THz pulses with high repetition rate*. Journal of Instrumentation, 12, 2017.

[CBC$^+$14] Caselle, M, M Balzer, S Chilingaryan, M Hofherr, V Judin, A Kopmann, N J Smale, P Thoma, S Wuensch, A S Müller, M Siegel und M Weber: *An ultra-fast data acquisition system for coherent synchrotron radiation with terahertz detectors*. Journal of Instrumentation, 9(01):C01024–C01024, jan 2014. `https://doi.org/10.1088/1748-0221/9/01/c01024`.

[Kes05] Kester, Walt: *The Data Conversion Handbook*. Newnes, 2005.

[LV02] Lundberg, Kent H. und Vin Vq: *Analog-to-Digital Converter Testing*, 2002.

[Pue15] Puente León, Fernando: *Messtechnik - Systemtheorie für Ingenieure und Informatiker*. Vieweg Verlag, 10. aufl. Auflage, 2015, ISBN 978-3-662-44820-5.

[Rot18] Rota, Lorenzo: *KALYPSO, a novel detector system for high-repetition rate and real-time beam diagnostics*. Dissertation, Karlsruher Institut für Technologie (KIT), 2018. 54.02.02; LK 01.

[Rou14] Roussel, Eleonore: *Spatio-temporal dynamics of relativistic electron bunches during the microbunching instability: study of the Synchrotron SOLEIL and UVSOR storage rings*. Dissertation, September 2014.

[SELP$^+$16] Szwaj, C., C. Evain, M. Le Parquier, P. Roy, L. Manceron, J. B. Brubach, M. A. Tordeux und S. Bielawski: *High sensitivity photonic time-stretch electro-optic sampling of terahertz pulses*. Review of Scientific Instruments, 87(10), 2016.

[Wad91] Wadell, Brian C.: *Transmission Line Design Handbook*. Artech House, 1991.