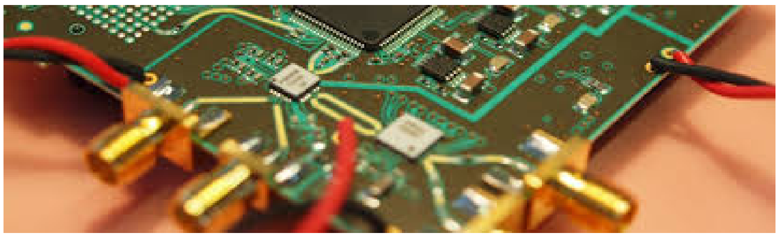
DEPARTMENT OF ELECTRICAL ENGINEERING
AND INFORMATION TECHNOLOGY
Institute for Data Processing and Electronics (IPE)

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

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**
- 1.1. Time-Stretching methods in physics . . . 1
- 1.2. Motivation . . . 1
  - 1.2.1. Karlsruhe Research Accelerator (KARA) . . . 1
  - 1.2.2. Coherent Synchrotron Radiation . . . 1
  - 1.2.3. Longitudinal Bunch Profile Diagnostics . . . 1
  - 1.2.4. Other scientific application for time-stretching methods . . . 1

2. Terabit Sampling System for Photonics Time-Stretch **3**
- 2.1. Photonic Front-End . . . 3
  - 2.1.1. Detector/Diode . . . 3
- 2.2. Analog-To-Digital Converters . . . 3
  - 2.2.1. Sampling Theory . . . 3
  - 2.2.2. Characteristics of Analog-To-Digital-Converters . . . 5
    - 2.2.2.1. Static parameters . . . 5
    - 2.2.2.2. Dynamic performance . . . 6
    - 2.2.2.3. Quantization Noise . . . 7
  - 2.2.3. Interleaving . . . 8

3. State Of The Art Read-Out-Systems **9**
- 3.1. Commercial Systems . . . 9
- 3.2. Karlsruhe Pulse Taking Ultra-fast Readout Electronics (KAPTURE) . . . 9
  - 3.2.1. General Architecture . . . 9
  - 3.2.2. Analog front-end . . . 10
  - 3.2.3. Sampling board . . . 10
  - 3.2.4. FPGA-GPU architecture . . . 13
  - 3.2.5. KAPTURE-2 . . . 13

4. Architecture of the new system **15**
- 4.1. Optical part . . . 15
- 4.2. Front-End Card . . . 15
- 4.3. Readout Card . . . 15
- 4.4. FPGA software . . . 15

5. Front-End Sampling Card - Terahertz Readout Sampling (THERESA) **17**
- 5.1. General Architecture . . . 17
- 5.2. Schematics . . . 17
- 5.3. Layout . . . 17
- 5.4. Production . . . 17

6. Readout Card - Xilinx ZCU216 **19**
6.1. General Description 19
6.2. Features 19

7. Firmware **21**
7.1. SoC 21
7.2. RF Data Converter 21
7.3. RDMA over Converged Ethernet (RoCE) 21
7.4. System Integration (SPI) 21

8. Conclusions and Outlook **23**
8.1. Measurements with the developed front-end card 23

Acknowledgements **25**

Appendix **27**
A. QuickStart Guide for Evaluation of ZCU216 Board 27
B. 3D model of front-end card 27
C. Code 27

Bibliography **33**

List of Figures

2.1. Transfer function of an ideal, 3-bit Analog-To-Digital-Converter (ADC) (redrawn from [LV02]) 3
2.2. Analog signal with frequency f_a sampled at f_s respecting (A) and not respecting (B) the Nyquist criteria. Figure 2.2b shows the effect of case B in time domain. [Kes05] 4
2.3. Track-And-Hold-Amplifier schematic and principle [Kes05] 5
2.4. Offset and Gain Error in the ADC characteristic. Notice the difference between the ideal, dotted line 6
2.5. Quantization noise as function of time (redrawn from [Kes05]) 7

3.1. THz measurement 10
3.2. General schema of KAPTURE-2 (cf. [CAB$^+$17, p.2]) 11
3.3. Signal and sampled points S_1 to S_4 11
3.4. Photo of KAPTURE with highlighted main components. [Bro20, p. 61] . . 12
3.5. Comparison between KAPTURE and KAPTURE-2[CAB$^+$17] 13

8.1. wow 23

List of Tables

Acronyms

ADC Analog-To-Digital-Converter. xiii, 3–6, 8, 10

DAQ Data Acquisition System. 9

FPGA Field Programmable Gate Array. 9, 10

FWHM Full Width Half Maximum. 9

IPE Institute for Data Processing and Electronics. 9, 10

KAPTURE Karlsruhe Pulse Taking Ultra-fast Readout Electronics. xi, 9, 10

KARA Karlsruhe Research Accelerator. xi, 1, 9, 10

LNA Low-Noise-Amplifier. 9, 10

LSB Least Significant Bit. 4–7

MMIC Microwave Monolithic Integrated Circuit. 10

PCIe PCI Express. 13

PLL Phase-Locked-Loop. 10

RF Radio Frequency. 10

SHA Sample-And-Hold-Amplifier. 4–6, 8

THA Track-And-Hold-Amplifier. 4, 9, 10

THERESA Terahertz Readout Sampling. xi, 17

THz tera Hertz. 9, 10

1. Introduction

1.1. Time-Stretching methods in physics

1.2. Motivation

1.2.1. KARA

1.2.2. Coherent Synchrotron Radiation

1.2.3. Longitudinal Bunch Profile Diagnostics

1.2.4. Other scientific application for time-stretching methods

- Microbunching instabilites $\rightarrow$ THz radiation $\rightarrow$ Potential source of brilliant radiation
- Need to be studied to be able to control it
- Time-resolution critical, as instabilities have high bandwidth
- Commercial oscilloscopes don't have enough internal memory
- already developed system KAPTURE/KAPTURE-2 to solve this problem
- however: four/eight points not enough to see the microbunching instabilities
- Concept: combine time-stretch with FPGA $\rightarrow$ THERESA (**T**era**He**rtz **Re**adout **Sa**mpling)
- EO technique to stretch the signal in time, so requirements on the bandwidth of converters can be relaxed
- more converters, to have more points
- New generation FPGA/SoC for fast readout and high data throughput

Available commercial DAQ system or real-time oscilloscopes for high bandwidth (e.g. above 40 GHz) are very expensive and, due to limited internal memory and missing fast readout interfaces, are not suitable for long term bunch-to-bunch CSR measurements. To circumvent these limitations a DAQ architecture for fast and continuous sampling of the individual ultra-short THz pulses has been developed.

2. Terabit Sampling System for Photonics Time-Stretch

2.1. Photonic Front-End

2.1.1. Detector/Diode

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 can be expressed as

$$V_{\text{IN}} = V_{\text{FS}} \sum_{k=0}^{N-1} \frac{b_k}{2^{k+1}} + \epsilon \tag{2.1}$$

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 2.1 shows the ideal transfer function of a 3-bit ADC.

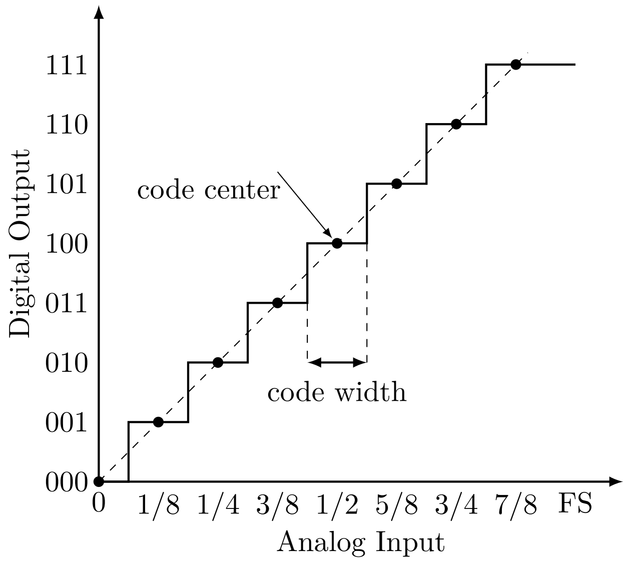


Figure 2.1.: Transfer function of an ideal, 3-bit ADC (redrawn from [LV02])

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

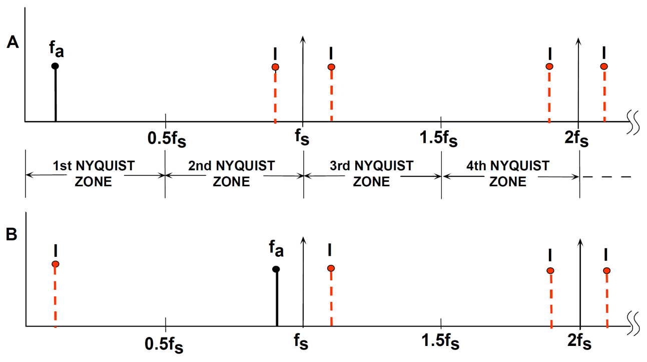
$$y(t) \circ\!\!-\!\!\!- Y(f) \quad \text{with} \quad Y(f) = 0, \, |f| \geq \frac{B}{2} \tag{2.2}$$

it has to be sampled with a frequency f_s respecting

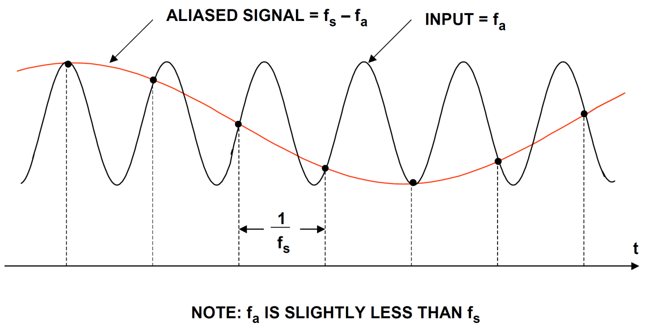
$$f_s \geq B \quad \text{or} \quad f_s \geq 2f_a \tag{2.3}$$

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

In other words, f_a must be inside of the *Nyquist bandwidth*, which is the spectrum from DC to $f_s/2$. Violation of this rule leads to *aliasing*.



(a) Sampling in frequency domain

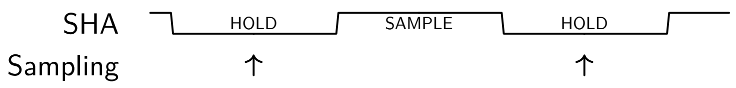


(b) Aliasing in time domain

Figure 2.2.: Analog signal with frequency f_a sampled at f_s respecting (A) and not respecting (B) the Nyquist criteria. Figure 2.2b shows the effect of case B in time domain. [Kes05]

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 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 2.3). 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.

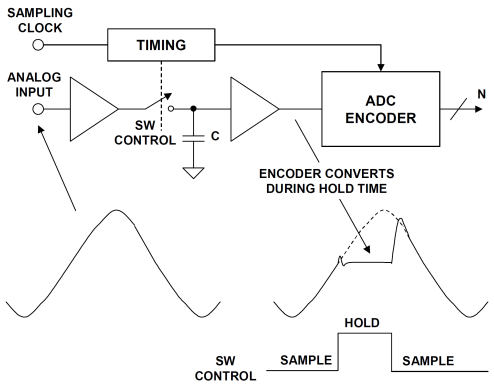


Figure 2.3.: Track-And-Hold-Amplifier schematic and principle [Kes05]

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

$$20 \log 2^N \approx 6N. \tag{2.4}$$

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 2.4

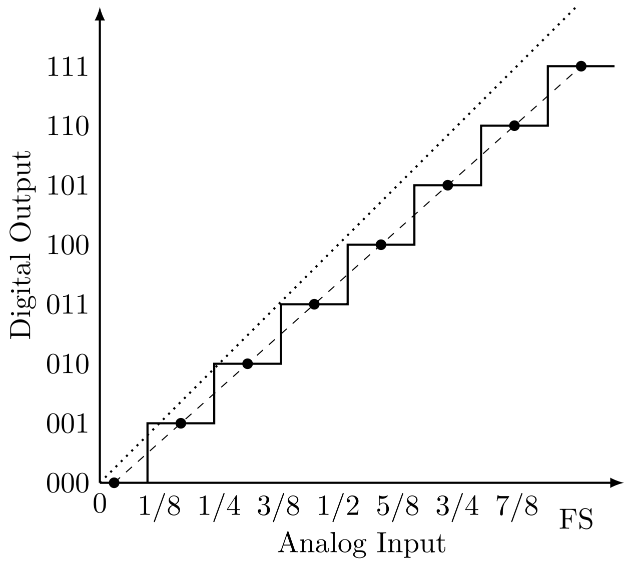


Figure 2.4.: Offset and Gain Error in the ADC characteristic. Notice the difference between the ideal, dotted line

Integral and Differential Non-linearity Distortion

Integral non-linearity in the transfer function of data converters results from the integral non-linearities of the front-end, SHA and also the ADC itself. These non-linearities depend on the input signal amplitude. distance of the code centers in the A/D converter characteristic from the ideal line *Differential non-linearities* stem exclusively from the encoding process in the ADC. They not only depend on the input signal amplitude, but also on the positioning along the transfer function. The deviation of the code transition widths from the ideal width of 1 LSB.

2.2.2.2. Dynamic performance

Frequency-Domain

- 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 2.1) [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

$$|e_q(t)| = |x(t) - x_q(t)| \leq \frac{q}{2} \tag{2.5}$$

with $x_q(t)$ being the quantized/discrete signal, $x(t)$ the input signal and q the width of the quantization stage. [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]:

$$e_q(t) = st, \quad -\frac{q}{2s} < t < \frac{q}{2s} \tag{2.6}$$

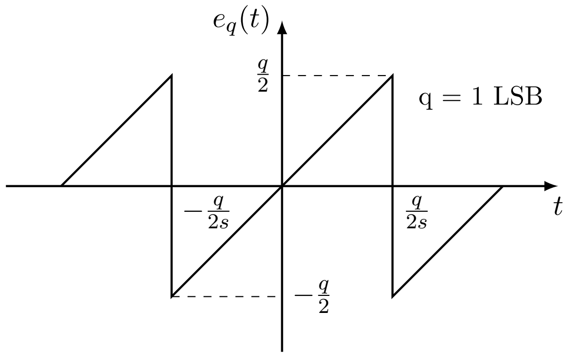


Figure 2.5.: Quantization noise as function of time (redrawn from [Kes05])

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

$$P_{\text{QN}} = e_{\text{rms}}^2 = \overline{e^2(t)} = \frac{s}{q} \int_{-q/2s}^{+q/2s} (st)^2 dt = \frac{s^3}{q} \left[\frac{t^3}{3} \right]_{-\frac{q}{2s}}^{+\frac{q}{2s}} = \frac{q^2}{12} \tag{2.7}$$

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

$$u(t) = u_s \sin(2\pi f t) = \frac{2^N q}{2} \sin(2\pi f t) = 2^{N-1} q \sin(2\pi f t) \tag{2.8}$$

With the effective value of the signal amplitude

$$u_{\text{eff}} = \frac{u_s}{\sqrt{2}} = \frac{2^{N-1}q}{\sqrt{2}} \tag{2.9}$$

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

$$\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} = \frac{u_{\text{eff}}^2}{e_{\text{rms}}^2} = \frac{2^{2N-2}q^2/2}{q^2/12} = 2^{2N} \cdot 1.5. \tag{2.10}$$

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

$$\text{SNR}|_{\text{dB}} = 10 \log \left(2^{2N} \cdot 1.5\right) = 6.02N + 1.76. \tag{2.11}$$

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

3.1. Commercial Systems

3.2. KAPTURE

KAPTURE is a fast readout system developed at the Institute for Data Processing and Electronics (IPE) for Tera Hertz (THz) diagnostics at KARA. It is designed to continuously digitize the pulses generated by THz detectors at each revolution, only sampling the pulse shapes without the "empty" signal inbetween. The system is able to sample pulses with a Full Width Half Maximum (FWHM) between a few tens to a hundred picoseconds with a minimal sample time of 3 ps. [CAB+17]

The concept and architecture of KAPTURE is described in the following, as it will serve as base for the newly developed readout system.

3.2.1. General Architecture

The system consists of two parts: the sampling front-end card and a Field Programmable Gate Array (FPGA) readout card. In Figure 3.1 the setup for THz radiation measurements with KAPTURE is shown.

The incoming radiation is fed into a detector, which converts the incident photons into an electrical signal. This signal is then amplified in a wide-band Low-Noise-Amplifier (LNA). The latter splits the detector signal into four identical signals, which are then propagated to the sampling front-end card. The card consists of four parallel sampling channels with adjustable sample time, each containing a THA and an ADC. This card is connected to a read-out card, which has two tasks: programming the components on the front-end card (FPGA-part) and sending the acquired data to a PC/Data Acquisition System (DAQ) system. [CBC+14]

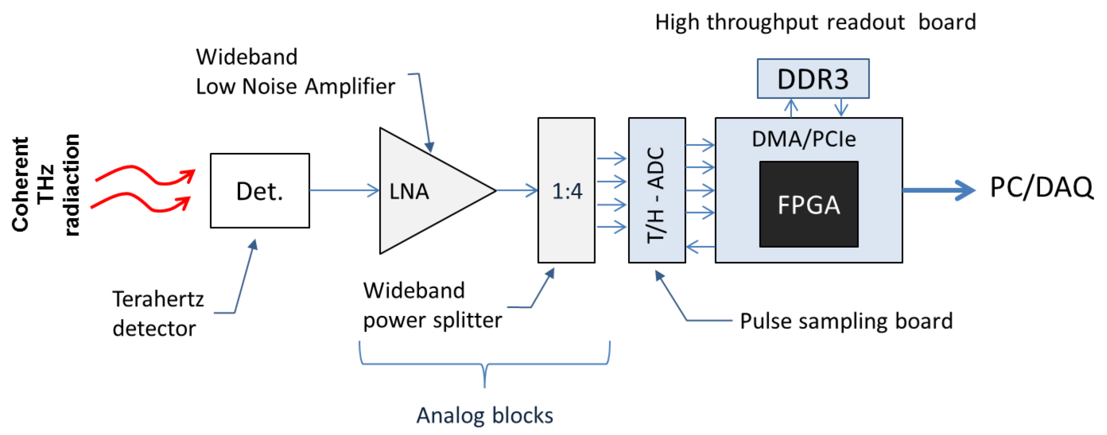


Figure 3.1.: Measurement setup for THz radiation with KAPTURE(cf. [CBC+14])

3.2.2. Analog front-end

Due to the high bandwidth nature of the detector signal, the analog front-end of the system has to be wideband as well to be able to sample the signal with picosecond resolution.

The used LNA is based on a commercial GaAs Microwave Monolithic Integrated Circuit (MMIC) which operates from DC to 50 GHz. It is needed to compensate the insertion loss[1] due to the following power splitter.

Classical dividers are not intrinsically wideband ([CBC+14]). For that reason, an own wideband power-splitter was developed at IPE to fulfill the bandwidth requirements. The designed power-splitter works up to 100 GHz with an insertion loss of 8 dB and a return loss of about 20 dB at 50 GHz.[CBC+14]

3.2.3. Sampling board

The general structure of the board with the power splitter is shown in Figure 3.2.

Four identical signals from the power-splitter are fed into four channels, consisting of a respective THA unit and a 12-bit ADC sampling at 500 MS/s. The sampling time of each unit can be adjusted individually with a delay chip with a resolution of 3 ps (maximal delay range: 100 ps). The delay chips are programmed with the FPGA on the readout card. The clock signal is provided by KARA, which is cleared from jitter by a Phase-Locked-Loop (PLL). This ensures the synchronization of the ADCs with the Radio Frequency (RF) system. The cleaned clock signal is distributed to the delay chips via fan-out buffer. [CAB+17] In this way, the pulse can be ”locally sampled” by adjusting the different delay with a maximum rate of 330 GS/s possible. A simplified representation of the local sampling of the signal is shown in Figure 3.3.

Figure 3.4 shows a photo of the system setup at KARA

[1] *Insertion loss* is the loss of signal power which occurs, when a signal passes through a component.

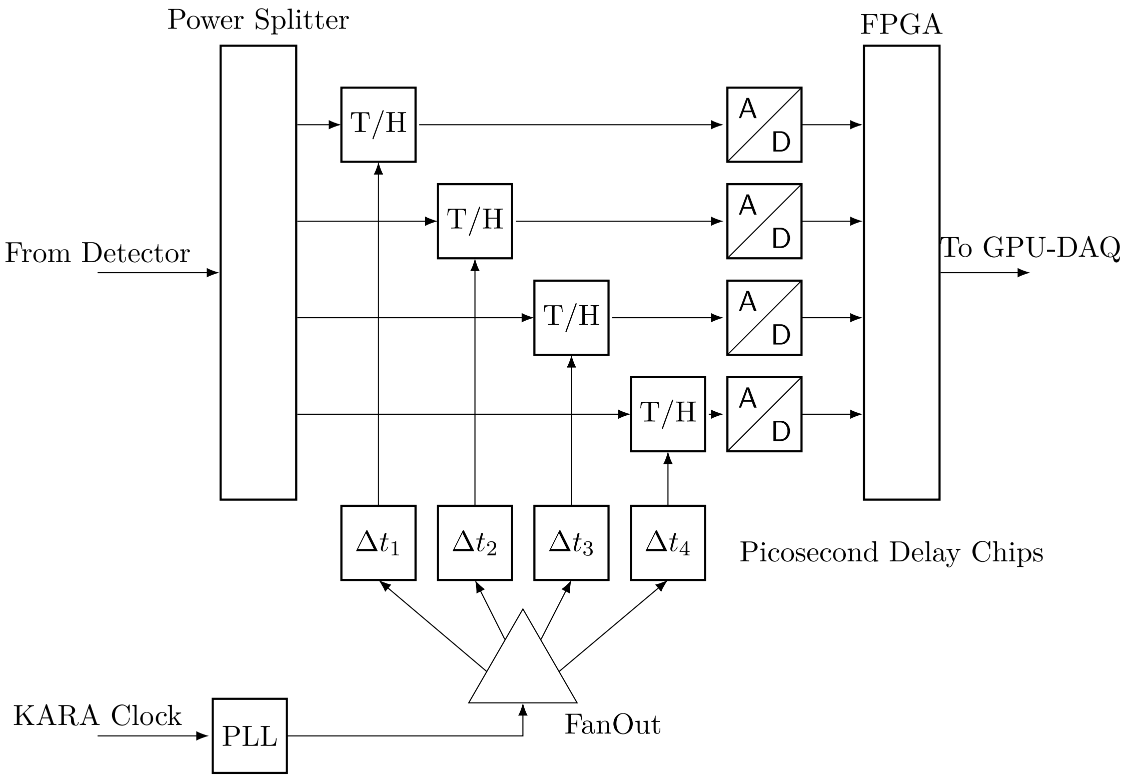


Figure 3.2.: General schema of KAPTURE-2 (cf. [CAB+17, p.2])

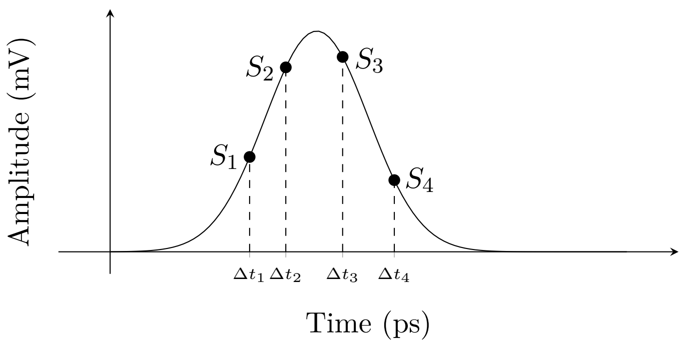


Figure 3.3.: Signal and sampled points S_1 to S_4

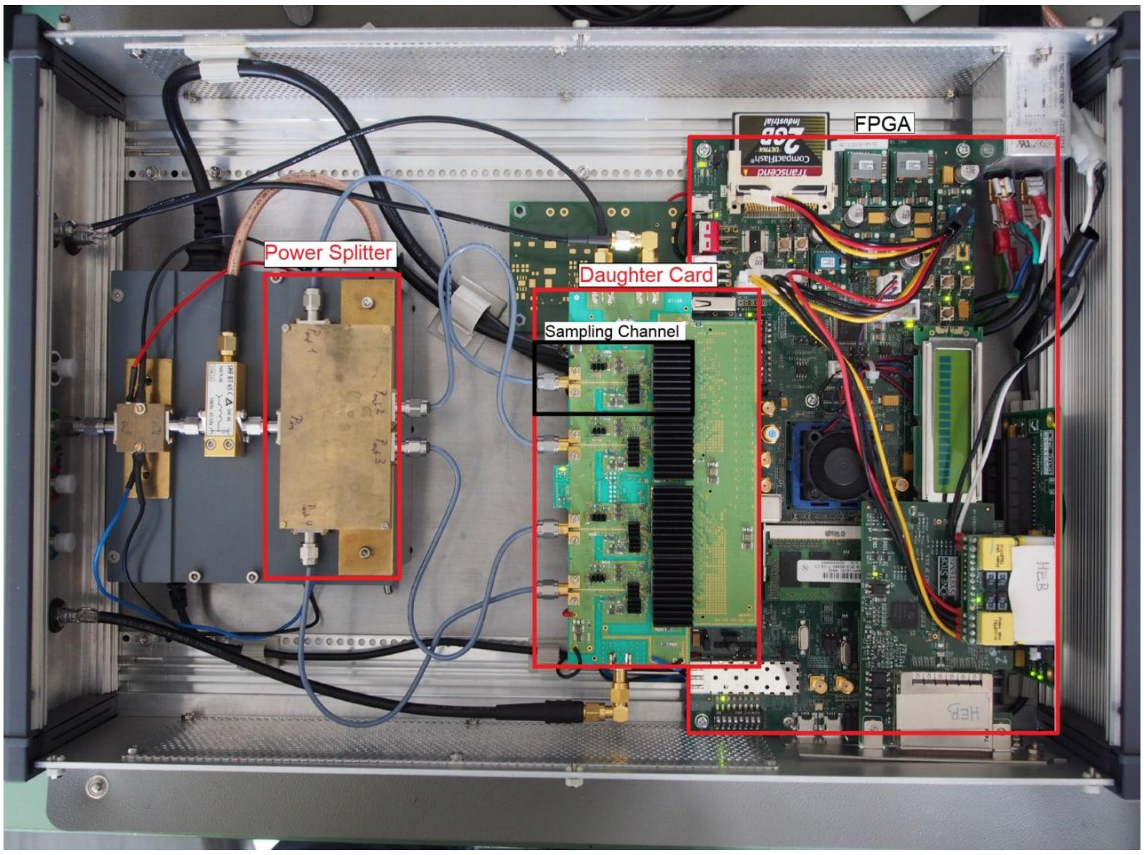


Figure 3.4.: Photo of KAPTURE with highlighted main components. [Bro20, p. 61]

3.2.4. FPGA-GPU architecture

In order to keep a continuous data acquisition the necessary bandwidth is

$$12\text{bits} \cdot 4\,\text{samples} \cdot 500\,\text{MHz} = 24\text{Gb/s} \tag{3.1}$$

To ensure high data throughput, the readout board is based on a bus master DMA architecture which is connected to PCI Express (PCIe) end-point logic. This ensure a throughput of up to 2 GByte/s. [CAB+17]

3.2.5. KAPTURE-2

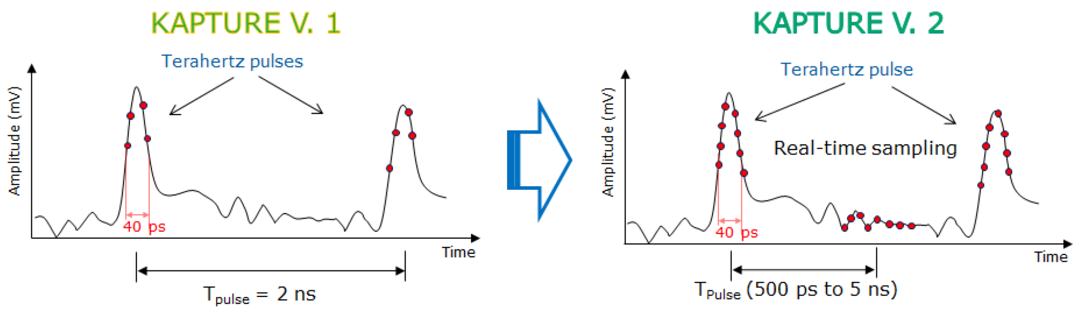


Figure 3.5.: Comparison between KAPTURE and KAPTURE-2[CAB+17]

4. Architecture of the new system

4.1. Optical part

4.2. Front-End Card

4.3. Readout Card

4.4. FPGA software

5. Front-End Sampling Card - THERESA

5.1. General Architecture

5.2. Schematics

5.3. Layout

5.4. Production

6. Readout Card - Xilinx ZCU216

6.1. General Description

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.

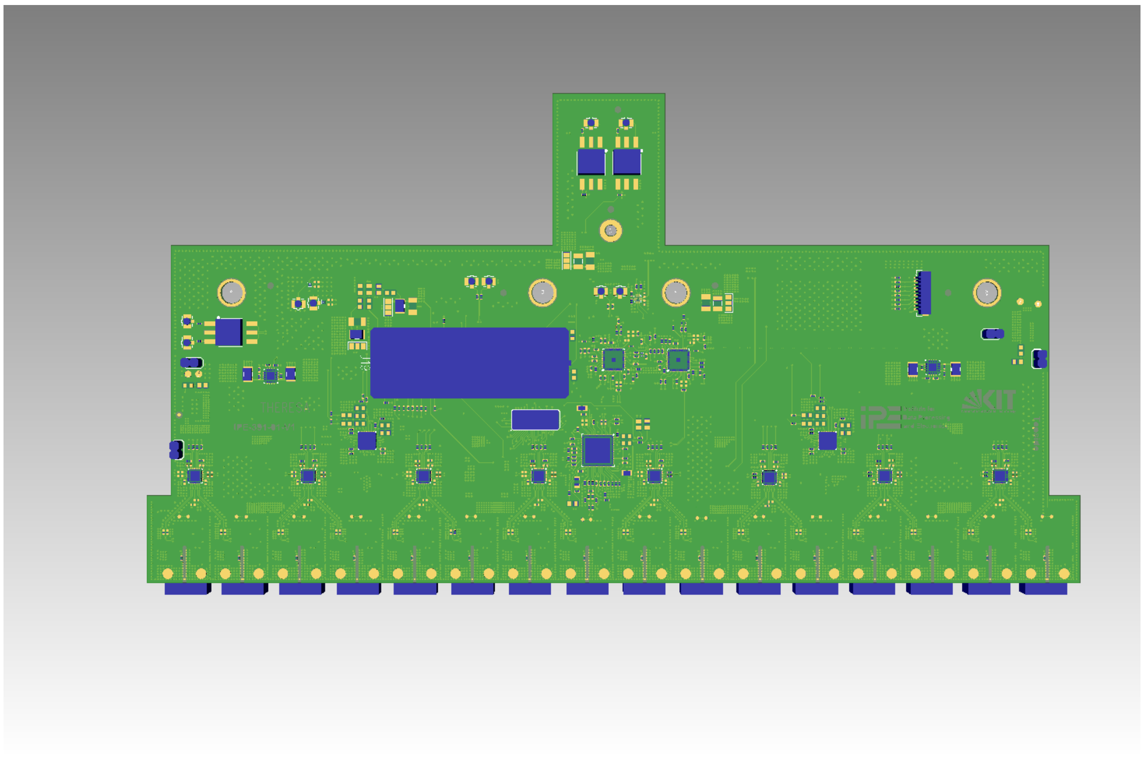


Figure 8.1.: wow

Acknowledgments

Appendix

A. QuickStart Guide for Evaluation of ZCU216 Board

B. 3D model of front-end card

C. 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

[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.

[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.