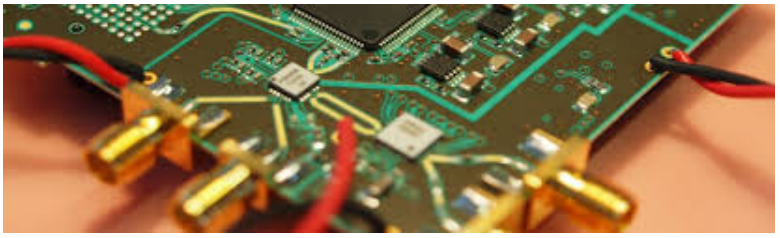
DEPARTMENT OF ELECTRICAL ENGINEERING
AND INFORMATION TECHNOLOGY
Institute for Data Processing and Electronics (IPE)

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

List of Figures

List of Tables

Acronyms

ADC Analog-To-Digital-Converter. 8–10

CSR Coherent Synchrotron Radiation. 4, 6

EO Electro-Optic. 6

LINAC linear accelerator. 3

RF Radio Frequency. 3

SHA Sample-And-Hold-Amplifier. 9, 12, 13

SR Synchrotron Radiation. 3

THA Track-And-Hold-Amplifier. 9

1. Introduction

- Microbunching instabilites → THz radiation → Potential source of brilliant radiation
- Need to be studied to be able to control it
- Time-resolution critical, as instabilites at high frequency
- EO techniques to measure such signals
- Concept: combine time-stretch with FPGA → THERESA (**T**era**He**rtz **Re**adout **Sa**mpling)

2. Theoretical Fundamentals

2.1. Coherent Synchrotron Radiation

Synchrotron Radiation (SR) is produced in synchrotron radiation facilites (like electron storage rings) by accelerating relativistic electrons. Emission of SR occurs, when electron beams are bend or deflected with dipole magnets or using an undulator[1].

Figure 2.1 shows the general scheme of an electron storage ring. Electrons, or rather electron bunches, are generated with an electron gun and are accelerated to almost the speed of light by a linear accelerator (LINAC). After being broad up to their nominal energy in a booster, the bunches are then injected into the storage ring. In the ring, the path of the electron bunches is altered by bending magnets, guiding them on a circular trajectory. Due to emission of SR at each bend, the electrons lose energy, which has to be compensated for. This is done by accelerating them with an electric field inside a Radio Frequency (RF) cavity. Not shown in the drawing are the beamlines, which lead the SR radiation, or rather chosen wavelength ranges, through an optical system to the respective user experiments. [?] [?]

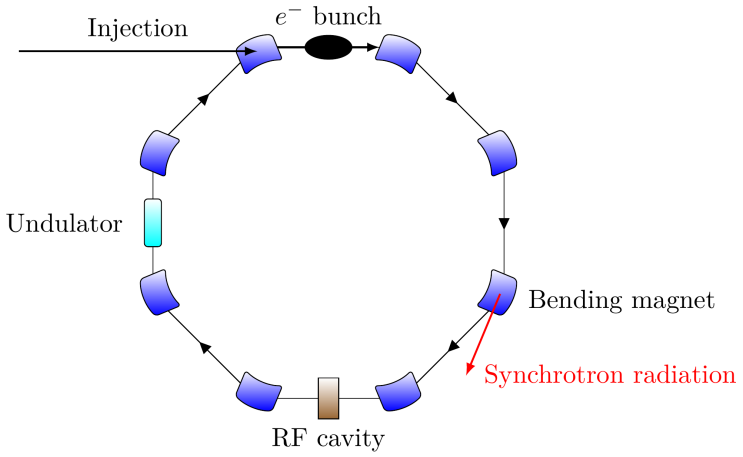


Figure 2.1.: Basic scheme of an electron storage ring (redrawn from [?])

The range of SR reaches from hard X-rays down to the infrared region of the electromagnetic spectrum (see Figure 2.2). In contrast to other sources, it has properties like:

- high intensity
- high collimation
- polarisation
- well-defined timing of pulses

[1] Undulators are used to make the electrons oscillate by generating a periodic magnetic field

Due to this properties, synchrotrons are used for microscopy, spectroscopy, and time-resolved experiments in such fields like condensed matter physics, biology, material science and many more.

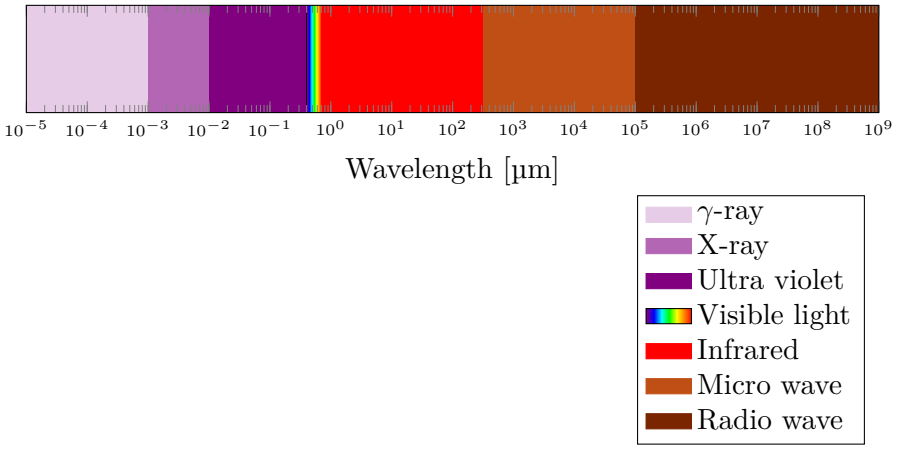


Figure 2.2.: Electromagnetic spectrum

2.1.1. Micro-Bunching Instabilities

Increasing demands in current and future accelerators call for higher brilliance of the emitted radiation. This is achieved by increased photon flux and reduction of the transverse emittance. For longitudinal coherence, the electron bunches are shortened, which results in emission of Coherent Synchrotron Radiation (CSR) at frequencies up to the THz range.

However, this introduces complex dynamics, as the electrons interact with their own radiation. This manifests into the so called micro-bunching instability. The formation of microstructures (in the sub-millimetre to centimetre range) in the longitudinal density profile of the electron bunches. Being on the one side a limitation to the stable operation of the overall system at high current density/short bunch length mode. On the other side, these instabilities can be potential sources of brilliant THz radiation. A thorough understanding of these dynamics is necessary to control the emission in this spectral domain which enables usage in experiments. [**?**, **?**]

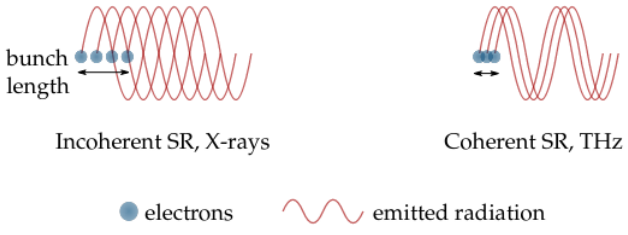


Figure 2.3.: Placeholder [**?**]

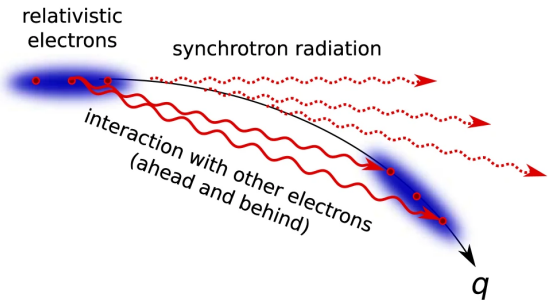


Figure 2.4.: Electrons interact with their own radiation [?]

Karlsruhe Research Accelerator (KARA)(?)

- Located at the Karlsruhe Institute of Technology (KIT)
- Up to 184 electron bunches can be filled with a distance of 2 ns (500 MHz) between two adjacent bunches
- Operated by the Institute of Beam Physics and Technology (IBPT)
- Microtron, Booster Synchrotron, and Storage Ring

Table 2.1.: KARA characteristics

Beam energy	2.5 GeV
Circumference	110 m
RF frequency	499.7 MHz
Harmonic number	184
Number of RF stations	2
Number of cavities per station	2
Accelerating voltage	1.4 MeV

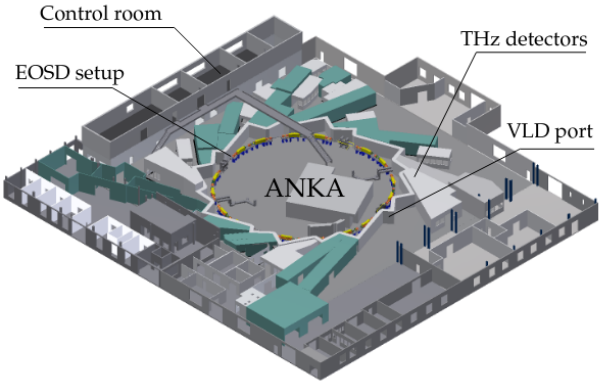


Figure 2.5.: Facility [?]

2.2. Electro-Optic Techniques and Time-Stretch

Several Electro-Optic (EO) techniques exist, to measure THz radiation with a sub-picosecond resolution. In this section, the techniques used in storage rings and the time-stretch technique are presented.

2.2.1. Scanning-Type Electro-Optic Sampling

A short laser pulse (duration typically hundreds of femtoseconds) co-propagates with a THz pulse from CSR (range of picoseconds) in an EO crystal. Due to the Pockels effect[2] the THz pulse causes a time dependent birefringence in the crystal. This modulates the polarization of the laser pulse. To sample the pulse, the delay between the laser and the THz pulse is varied. To detect the changing polarization, the polarization of the laser pulse is transformed into an intensity modulation. A general scheme of the system is shown in Figure 2.6. For this technique a stable emission of the THz pulses is crucial, as they are not measured in one acquisition. [?]

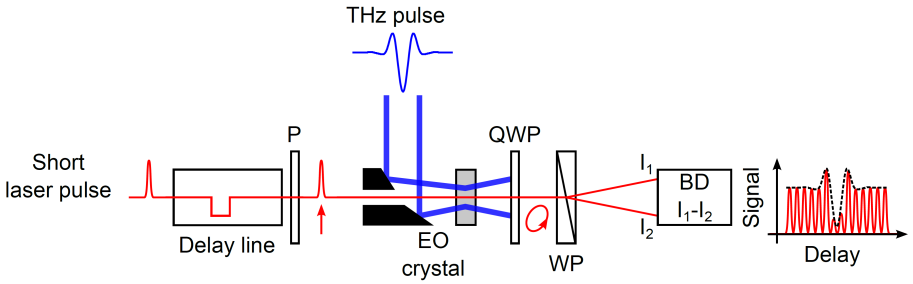


Figure 2.6.: Scheme of Scanning-Type Electro-Optical Sampling System [?]

2.2.2. Spectrally Resolved Electro-Optic Detection

In contrast to the EOS, single-acquisition is possible with the spectrally resolved electro-optic detection technique. The short laser pulse is first stretched to a duration similar to the Thz pulse in a dispersive material (stretcher). In this way the pulse is chirped, meaning the instantaneous frequency of the pulse varies over time. Together with the THz, the laser pulse propagates in a EO crystal. Again, the induced birefringence modulates the laser pulse, not only in time, but also in the optical spectrum. The polarization state of the pulse is converted into an amplitude/intensity modulation. To retrieve the THz pulse shape in time, the spectrum of the laser pulse is measured with a spectrometer. A general scheme of the system is shown in Figure 2.7. [?]

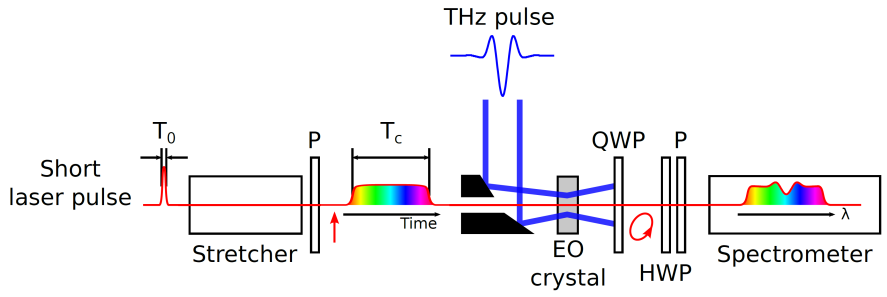


Figure 2.7.: Scheme of Spactrally Encoded Electro-Optical Detection System [?]

[2] "The Pockels effect is the name given to the occurrence of birefringence and to the change in existing birefringence phenomena in an electric field linearly proportional to the electric field strength." [?]

The temporal resolution of this method is limited due to the finite chirp rate

$$\text{chirp rate} = \frac{\text{laser bandwidth}}{\text{laser pulse duration after stretcher}}. \tag{2.1}$$

The resolution T_{min} is determined as

$$T_{\text{min}} = \sqrt{T_0 T_c} \tag{2.2}$$

with the bandwidth-limited pulse duration (before stretcher) T_0 and the duration of the chirped laser pulse T_c.

2.2.3. Photonic Time-Stretch Technique

The working principle of the optic time-stretch technique can be described in two steps (see Figure 2.8). First, like in the EOSD before, 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

$$T_1 = \Delta\lambda D_1 L_1 \tag{2.3}$$

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. [**?**]

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

$$M = 1 + \frac{L_1}{L_2}. \tag{2.4}$$

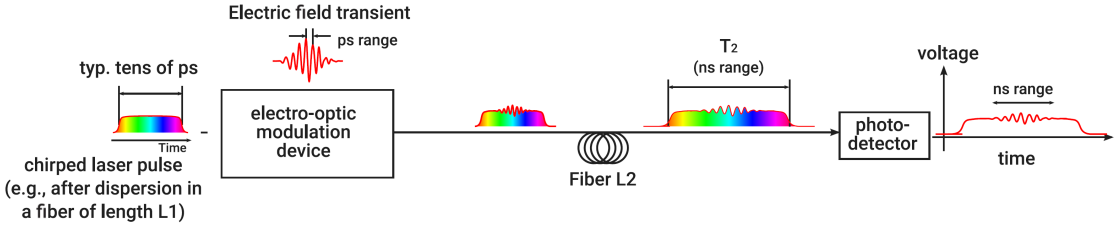


Figure 2.8.: Electro-Optical Time-Stretch Technique [**?**]

2.2.4. Time-Stretched Spectrally-Encoded Electro-Optic detection

2.3. Analog-To-Digital Converters

The following section reviews the theory and characteristics of ADCs.

2.3.1. 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

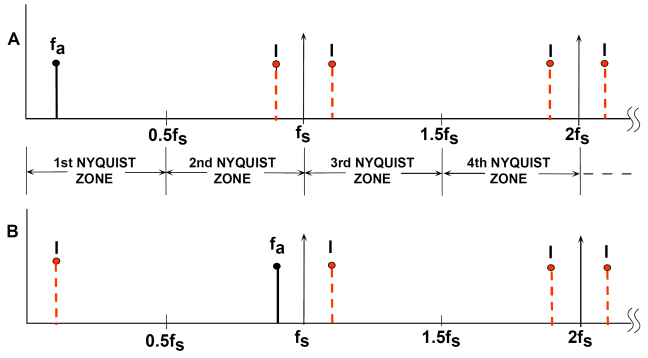
$$y(t) \circ\!\!-\!\!- Y(f) \quad \text{with} \quad Y(f) = 0,\ |f| \geq \frac{B}{2} \tag{2.5}$$

it has to be sampled with a frequency f_s respecting

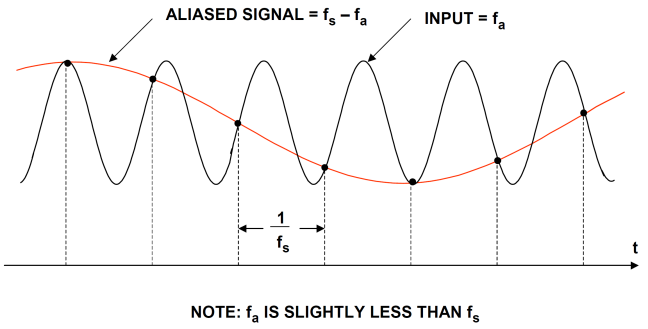
$$f_s \geq B \quad \text{or} \quad f_s \geq 2f_a \tag{2.6}$$

with f_a being the highest frequency contained in the signal. [**?**, **?**]

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 2.9.: Analog signal with frequency f_a sampled at f_s respecting (A) and not respecting (B) the Nyquist criteria. Figure 2.9b shows the effect of case B in time domain. [**?**]

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. [**?**]

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 2.10). 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.

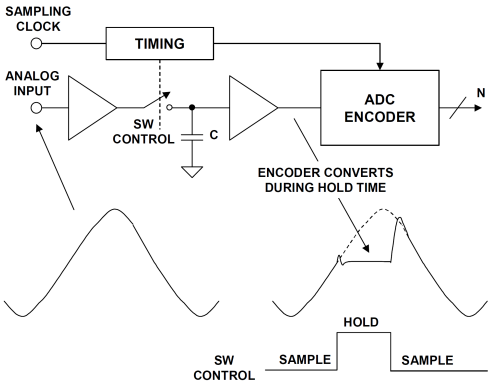


Figure 2.10.: Track-And-Hold-Amplifier schematic and principle [**?**]

2.3.2. Characteristics of 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

$$V_{\mathrm{IN}} = V_{\mathrm{FS}} \sum_{k=0}^{N-1} \frac{b_k}{2^{k+1}} + \epsilon \tag{2.7}$$

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

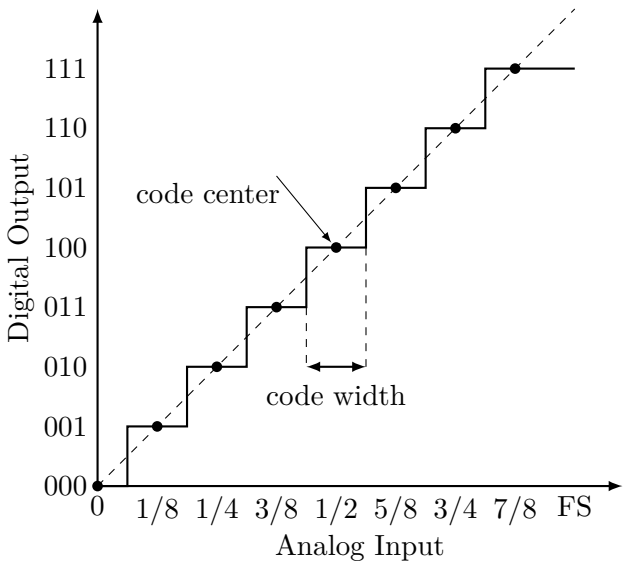


Figure 2.11.: Transfer function of an ideal, 3-bit ADC (redrawn from [**?**])

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 [**?**]:

- 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.

2.3.2.1. 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 2.11) [**?**]. The input voltage is always assigned to the word of the nearest code center. This means that there will always be a difference between the code center and the actual input. The difference between the corresponding voltage of the respective code center and the analog input is called the *quantization error*. For an equidistant quantization it is

$$|e_q(t)| = |x(t) - x_q(t)| \leq \frac{q}{2} \tag{2.8}$$

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

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:

$$e(t) = st, \quad -\frac{q}{2s} < t < \frac{q}{2s} \tag{2.9}$$

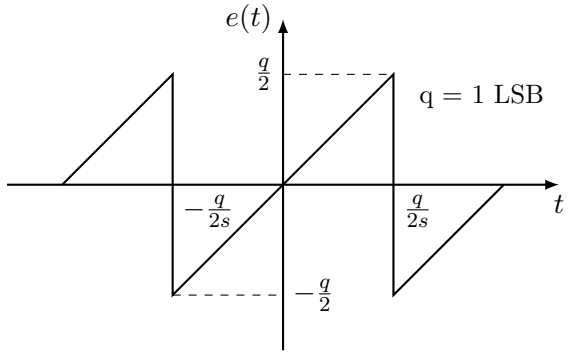


Figure 2.12.: Quantization noise as function of time (redrawn from [**?**])

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

$$P_{\mathrm{QN}} = e_{\mathrm{rms}}^2 = \overline{e^2(t)} = \frac{s}{q}\int_{-q/2s}^{+q/2s}(st)^2 dt = \frac{s^3}{q}\left[\frac{t^3}{3}\right]_{-\frac{q}{2s}}^{+\frac{q}{2s}} = \frac{q^2}{12} \tag{2.10}$$

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

$$u(t) = u_s \sin(2\pi f t) = \frac{2^N q}{2}\sin(2\pi f t) = 2^{N-1} q \sin(2\pi f t) \tag{2.11}$$

With the effective value of the signal amplitude

$$u_{\mathrm{eff}} = \frac{u_s}{\sqrt{2}} = \frac{2^{N-1}q}{\sqrt{2}} \tag{2.12}$$

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

$$\mathrm{SNR} = \frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}} = \frac{u_{\mathrm{eff}}^2}{e_{\mathrm{rms}}^2} = \frac{2^{2N-2}q^2/2}{q^2/12} = 2^{2N}\cdot 1.5. \tag{2.13}$$

Using the unit Decibel, this can be expressed as (see [**?**, **?**])

$$\mathrm{SNR}|_{\mathrm{dB}} = 10\log\left(2^{2N}\cdot 1.5\right) = 6.02N + 1.76. \tag{2.14}$$

2.3.2.2. 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

$$20 \log 2^N \approx 6N. \tag{2.15}$$

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 2.13

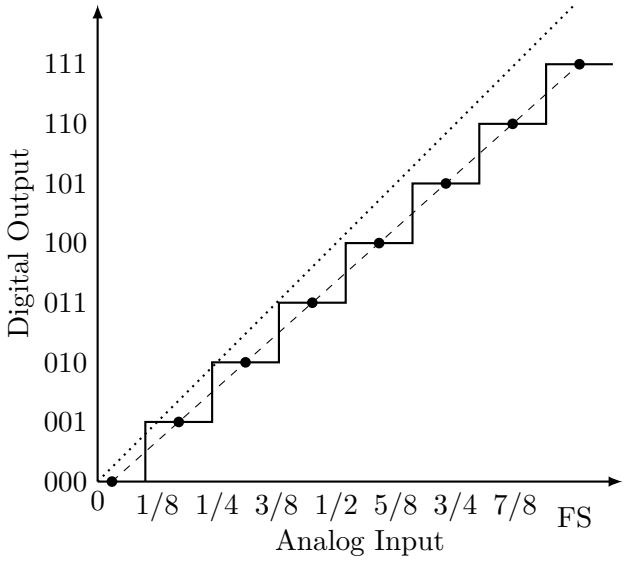


Figure 2.13.: Offset and Gain Error in the ADC characteristic. Notice the difference between the ideal, dotted line

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.

2.3.2.3. 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

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[3] 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. [**?**]

2.3.3. Interleaving

- Net sample rate
- Interleaving Spurs

[**?**]

[3] *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.

2.4. RF/Microwave Design Basics

2.4.1. General Techniques/Strategies?

2.4.2. Coplanar Waveguides

Surface Coplanar Waveguide with Ground

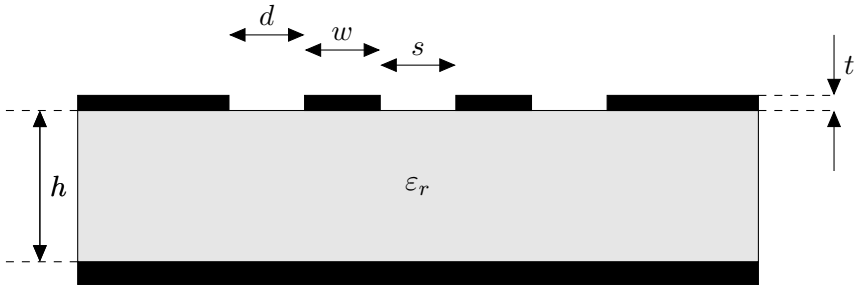


Figure 2.14.: Edge-Coupled Coplanar Waveguide

The corresponding equations are [**?**, p197-198]:

$$Z_{0,o} = \frac{\eta_0}{\sqrt{\epsilon_{eff,o}}} \left(\frac{1.0}{2.0\frac{K(k_o)}{K'(k_o)} + \frac{K(\beta_1)}{K'(\beta_1)}} \right) \tag{2.16}$$

$$Z_{0,e} = \frac{\eta_0}{\sqrt{\epsilon_{eff,e}}} \left(\frac{1.0}{2.0\frac{K(k_e)}{K'(k_e)} + \frac{K(\beta_1 k_1)}{K'(\beta_1 k_1)}} \right) \tag{2.17}$$

$$\epsilon_{eff,o} = \frac{2.0\epsilon_r \frac{K(k_o)}{K'(k_o)} + \frac{K(\beta_1)}{K'(\beta_1)}}{2.0\frac{K(k_o)}{K'(k_o)} + \frac{K(\beta_1)}{K'(\beta_1)}} \tag{2.18}$$

$$\epsilon_{eff,e} = \frac{2.0\epsilon_r \frac{K(k_e)}{K'(k_e)} + \frac{K(\beta_1 k_1)}{K'(\beta_1 k_1)}}{2.0\frac{K(k_e)}{K'(k_e)} + \frac{K(\beta_1 k_1)}{K'(\beta_1 k_1)}} \tag{2.19}$$

Where

$$k_o = \Lambda \frac{-\sqrt{\Lambda^2 - t_c^2} + \sqrt{\Lambda^2 - t_B^2}}{t_B\sqrt{\Lambda^2 - t_c^2} + t_c\sqrt{\Lambda^2 - t_B^2}} \tag{2.20}$$

$$k_e = \Lambda' \frac{-\sqrt{\Lambda'^2 - t_c'^2} + \sqrt{\Lambda'^2 - t_B'^2}}{t_B'\sqrt{\Lambda'^2 - t_c'^2} + t_c'\sqrt{\Lambda'^2 - t_B'^2}} \tag{2.21}$$

$$\Lambda = \frac{\sinh^2\left(\frac{\pi(s/2.0+w+d)}{2.0h}\right)}{2} \tag{2.22}$$

$$t_c = \sinh^2\left(\frac{\pi(s/2.0+w)}{2.0h}\right) - \Lambda \tag{2.23}$$

$$t_B = \sinh^2\left(\frac{\pi s}{4.0h}\right) - \Lambda \tag{2.24}$$

$$\Lambda' = \frac{\cosh^2\left(\frac{\pi(s/2.0+w+d)}{2.0h}\right)}{2} \tag{2.25}$$

$$t'_c = \sinh^2\left(\frac{\pi(s/2.0+w)}{2.0h}\right) - \Lambda' + 1.0 \tag{2.26}$$

$$t'_B = \sinh^2\left(\frac{\pi s}{4.0h}\right) - \Lambda + 1.0 \tag{2.27}$$

The parameters have to be chosen according to

$$s + 2.0w + 2.0d \leq h \tag{2.28}$$

to guarantee coplanar propagation. [**?**]

Surface Coplanar Waveguide with Ground

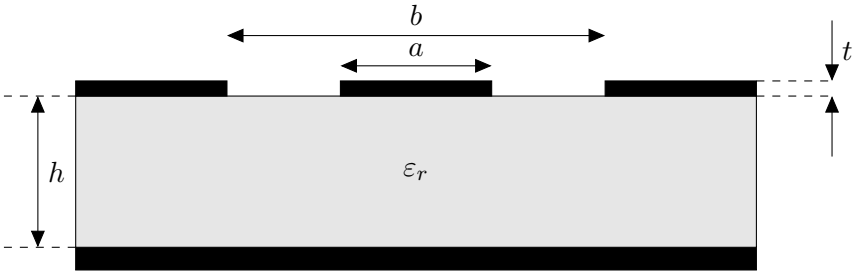


Figure 2.15.: Coplanar Waveguide with Ground

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

$$Z_0 = \frac{60.0\pi}{\sqrt{\epsilon_{eff}}} \frac{1.0}{\frac{K(k)}{K(k')} + \frac{K(k_1)}{K(k'_1)}} \tag{2.29}$$

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

$$k = a/b \tag{2.30}$$

$$k' = \sqrt{1.0 - k^2} \tag{2.31}$$

$$k'_1 = \sqrt{1.0 - k_1^2} \tag{2.32}$$

$$k_1 = \frac{tanh(\frac{\pi a}{4.0h})}{tanh(\frac{\pi b}{4.0h})} \tag{2.33}$$

$$\epsilon_{eff} = \frac{1.0 + \epsilon_r \frac{K(k')}{K(k)} \frac{K(k_1)}{K(k'_1)}}{1.0 + \frac{K(k')}{K(k)} \frac{K(k_1)}{K(k'_1)}} \tag{2.34}$$

3. Design and Development of the System

This chapter covers the architecture and design of the system.

3.1. General architecture

In this section the general architecture of the Terahertz Readout Sampling (THERESA) system is described. The idea for the new system is to reuse the concept of the already existing sampling system Karlsruhe Pulse Taking Ultra-fast Readout Electronics (KAPTURE), which is using four ADCs, and expanding it to 16. Therefore, also the architecture of the latter is explained briefly.

3.1.1. KAPTURE-2

KAPTURE, developed at Institut für Prozessdatenverarbeitung und Elektronik (IPE), is a system designed to continuously sample ultra-short pulses generated by Terahertz detectors. It consists of a daughter card, holding four sampling channels, mounted on a Field-Programmable Gate Array (FPGA). The FPGA is connected to a computer via PCI Express (PCIe) for further processing of the acquired data. [**?**] The newer version, KAPTURE-2, was designed for more accurate sampling for pulse repetition rates up to 2 GHz. The acquired data is processed with a FPGA and GPU architecture. [**?**]

The general structure of the board is shown in Figure 3.1.

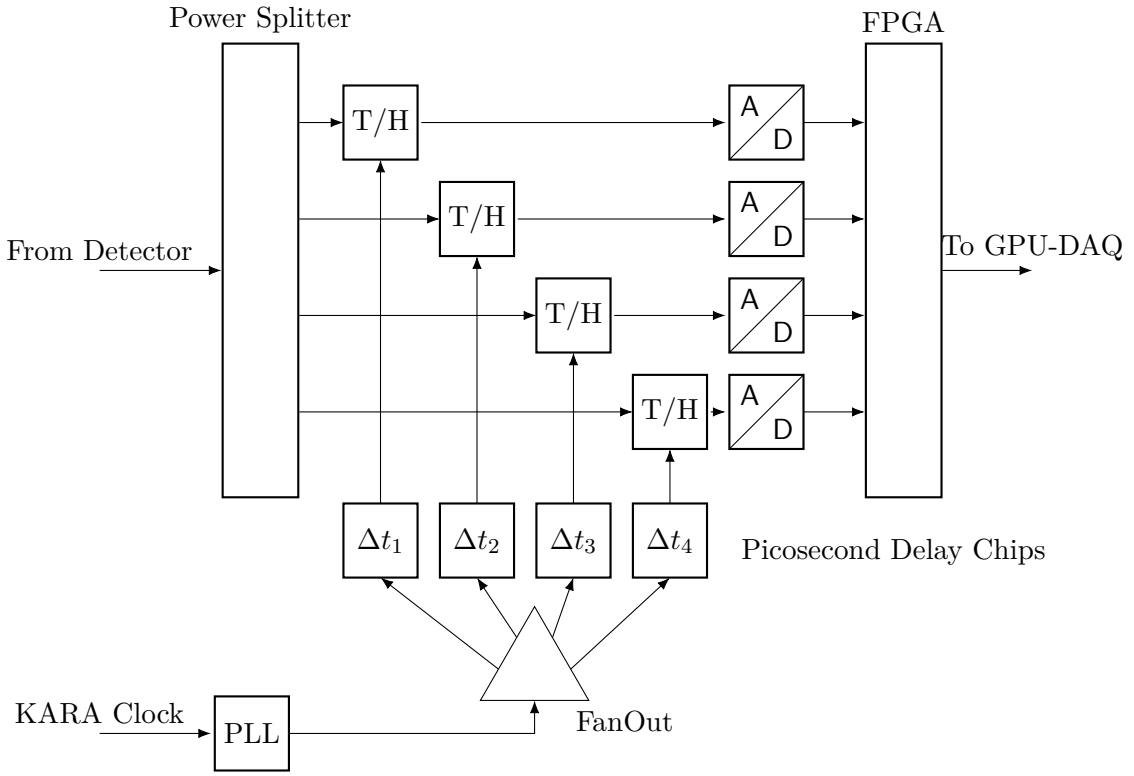


Figure 3.1.: General schema of KAPTURE-2 (cf. [**?**, p.2])

The pulse from the tera Hertz (THz) detector is fed into a power splitter, which splits the signal into four identical pulses and distributes them to 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 Picosecond Delay Chip with a resolution of 3 ps (maximal delay range: 100 ps). The clock signal is provided by Karlsruhe Research Accelerator (KARA), which is cleared from jitter by a Phase-Locked-Loop (PLL). This ensures the synchronization of the ADCs with the RF system. The cleaned clock signal is distributed to the delay chips via fan-out. [**?**]

The resulting sampling of the detector signal is shown Figure 3.2 (simplified representation).

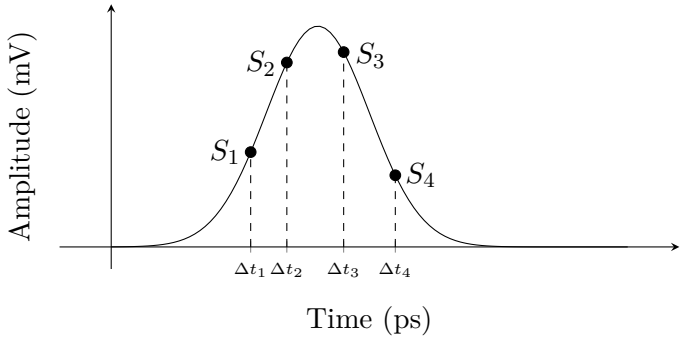


Figure 3.2.: Signal with sample points

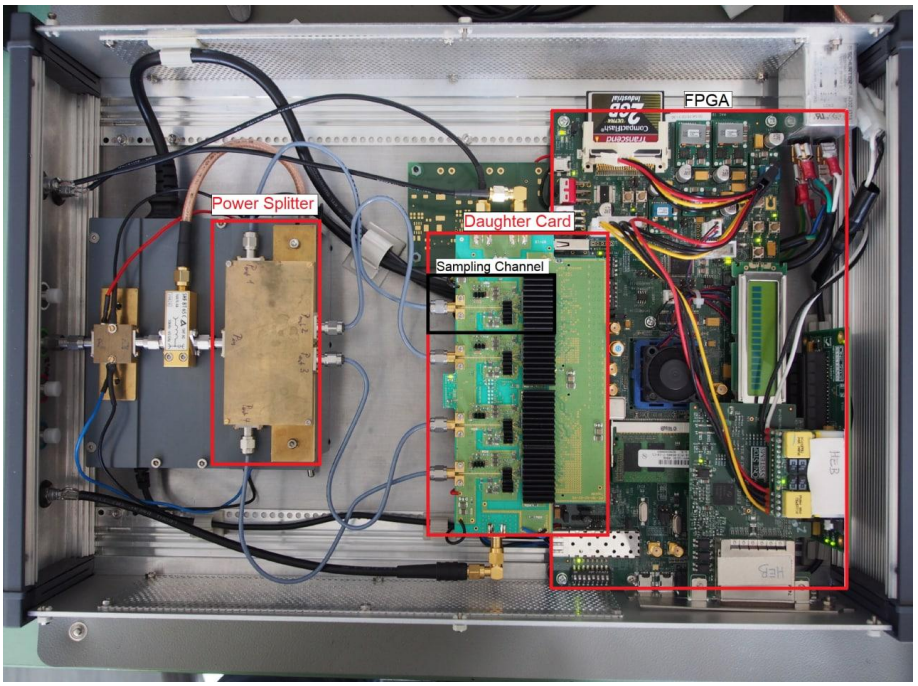


Figure 3.3.: Photo of KAPTURE with highlighted main components. [**?**, p. 61]

3.1.2. New System THERESA

In principle, the new system has the same structure, as KAPTURE. Notable differences are firstly the number of ADCs, which is increased up to 16. Secondly, the latter are not located on the daughter card anymore, but inside the Xilinx Zynq UltraScale+ RFSoC ZU49DR on the ZCU216 Evaluation Kit on which the front-end card is mounted. Figure 3.4 shows the general schema of the sampling system, reduced to four channels for presentation purposes.

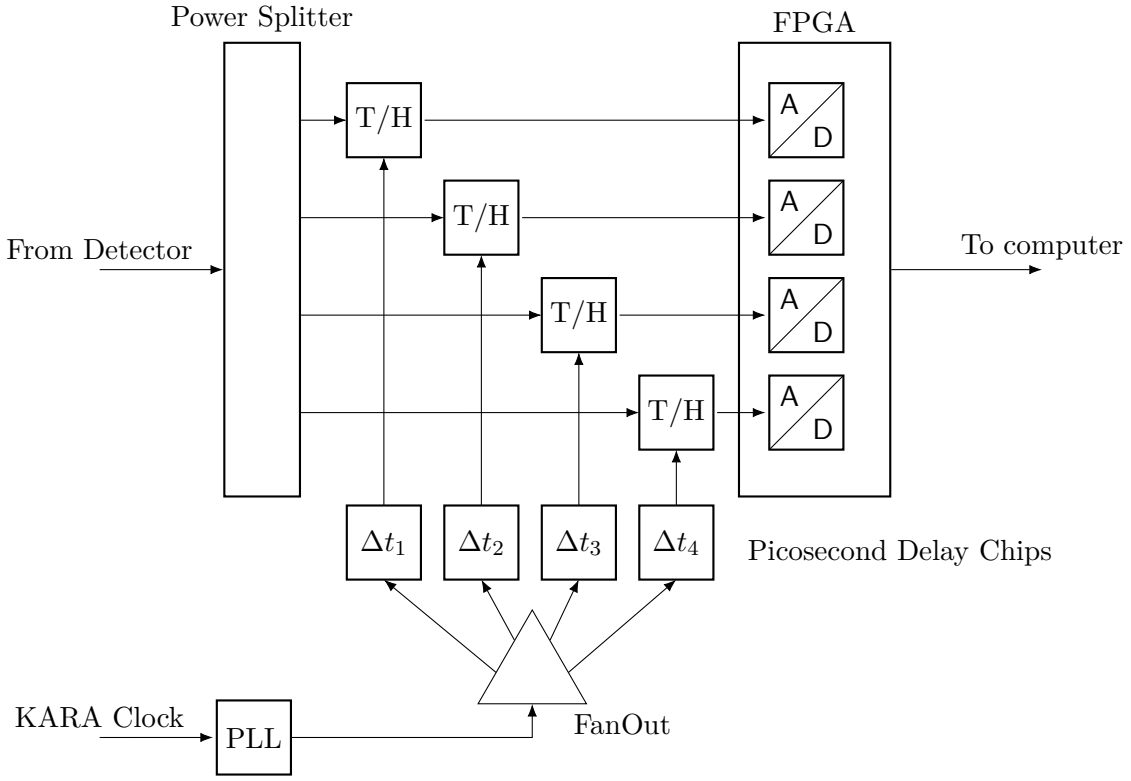


Figure 3.4.: General schema of THERESA. For presentation purposes only four channels are shown.

Xilinx Zynq UltraScale+ RFSoC ZCU216 Evaluation Card

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

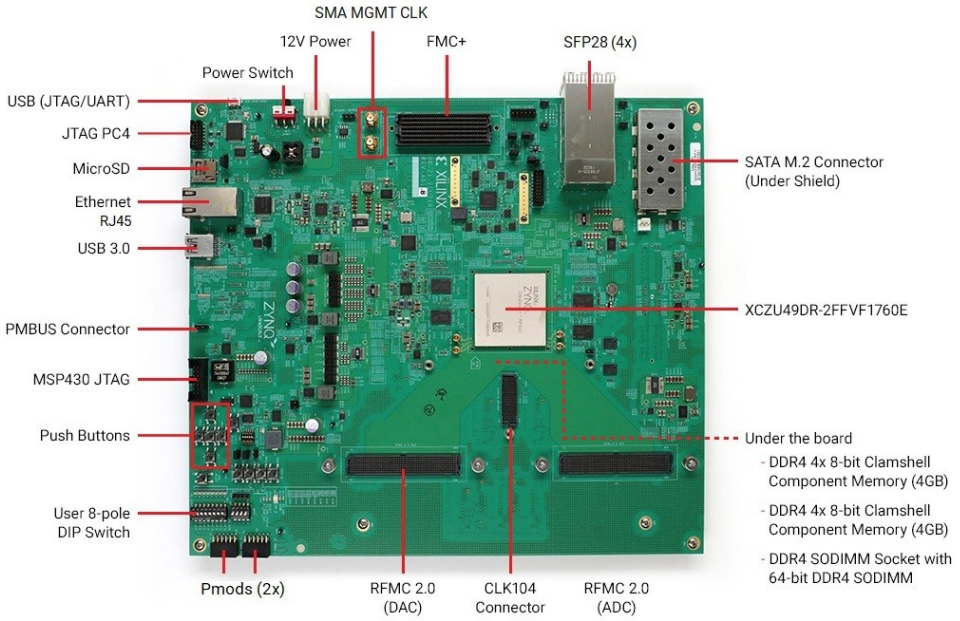


Figure 3.5.: ZCU216 evaluation board

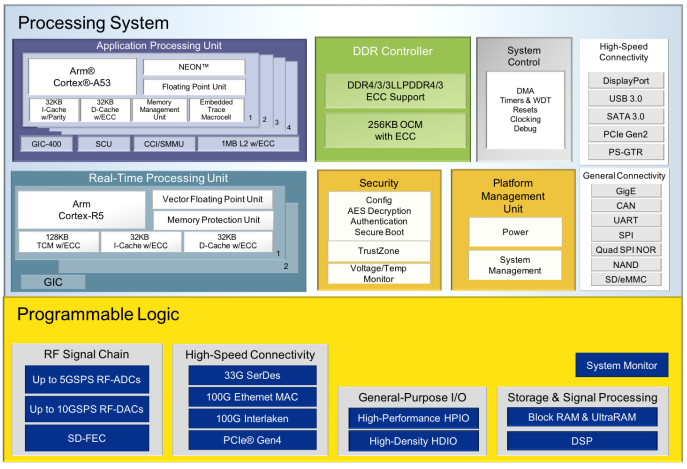


Figure 3.6.: RFSoC block diagram

3.1.3. Requirements

Step size delay chip

The necessary step size for the delay chips, when using 16 ADC@2 GS/s in time-interleaving mode, is: $\frac{2\,\mathrm{GS/s}}{16} = 31\,\mathrm{ps}$

Frequency

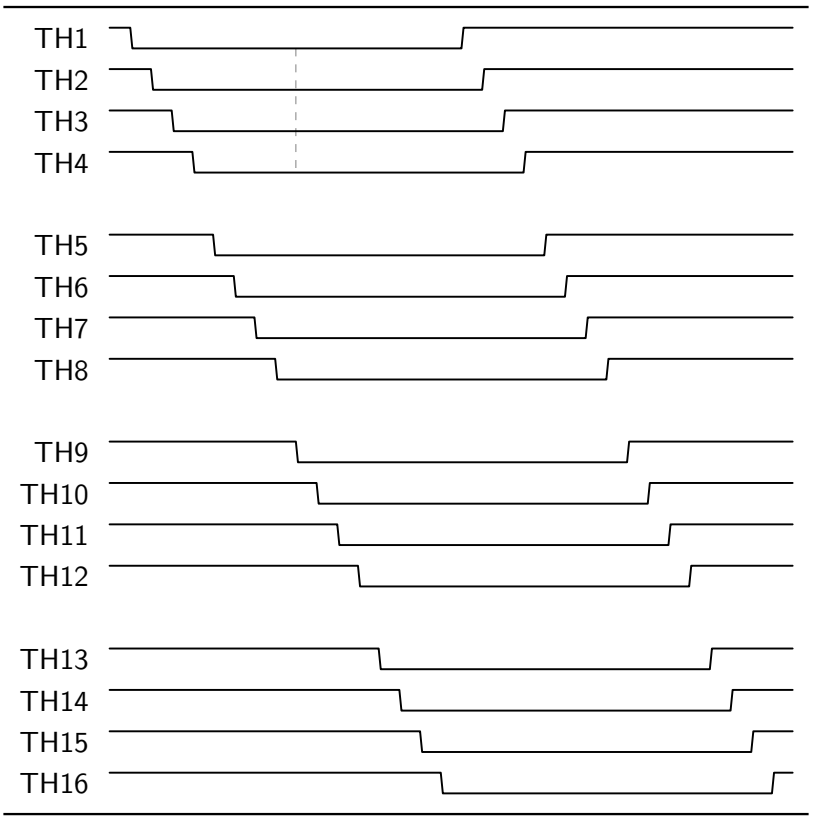


Figure 3.7.: Track-And-Hold Timing diagram

Data Rate

Visualization/GUI

3.2. Design of the front-end card

In this section, the design of the front-end card is covered.

3.2.1. Sampling-Channel

Track-And-Hold-Amplifier

Explain why better than Sample-And-Hold-Amplifier.

Delay Chip NB6L295

Dual Channel Programmable Delay Chip.

- Two individual variable delay channels
- Dual Delay: minimal delay 3.2 ns
- Total Delay Range: 3.2 ns to 8.8 ns per Delay Channel
- 11 ps Increments in 511 steps
- 100 ps Typical Rise and Fall Times

3.2.2. Clocking

On Xilinx CLK104 add-on board the LMX2594 is used to generate additional, high-frequency clocks for the ADCs/DACs. → reuse in front-end card

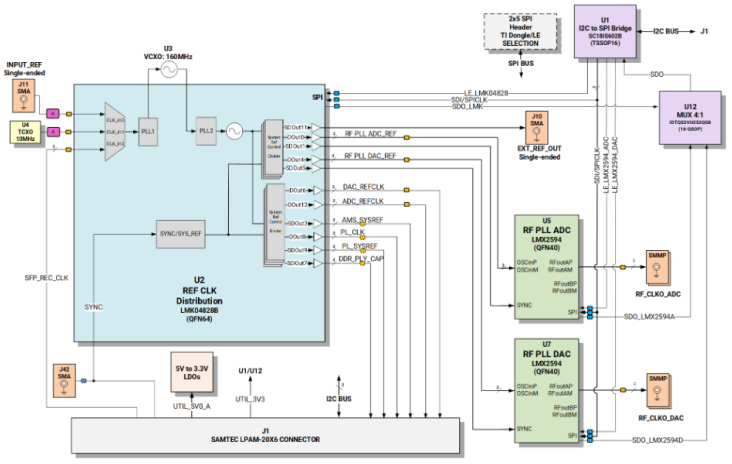


Figure 3.8.: CLK104 Add-on board clocking scheme

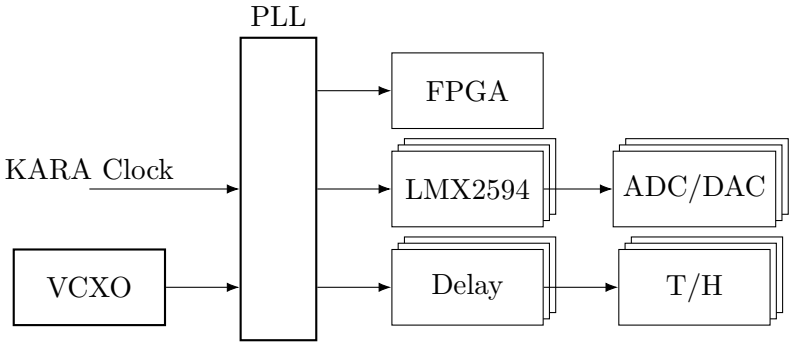


Figure 3.9.: Clocking scheme on front-end card

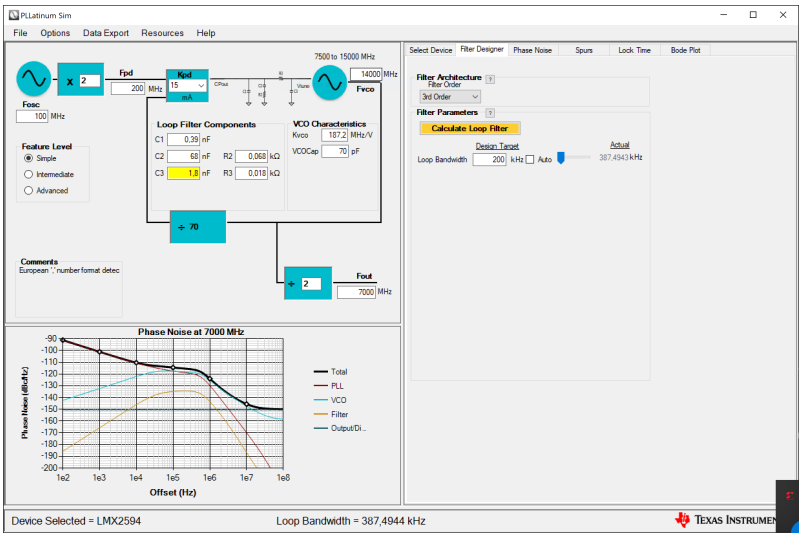


Figure 3.10.: Placeholder

3.2.3. Power Supply

For the Track-And-Hold amplifiers a new power supply unit – the ADP1741 (Analog Devices) – should be used. It is necessary to think about the amount of power supply chips needed. As a rule of thumb, the power supply should provide twice the maximum power needed by the components it drives. The power consumption/maximum current for the respective components on the THERESA board is listed in Table 3.1.

Table 3.1.: Power consumption of components on the board

Component	V_{cc} (V)	I_{max} (A)	P_{max} (W)	$\#_{parts}$	I_{tot}[1] (A)
HMC5649 (T/H-Amplifier)	2	0.221	0.442	16	3.536
	−5	−0.242	1.21		3.872
HMC856 (Delay)	−3.3	0.185	−0.611	16	2.96
HMC987LP5E (Fan-Out)	3.3	0.234[2]	0.772	2	0.468
LMC0480 (PLL)	3.3	0.590[3]	1.947	1	0.590
VCXO	3.3	0.03	0.198	1	0.03

[1] for 16 ADCs
[2] All Outputs and RF-Buffer
[3] All CLKs

The maximal current which the ADP1741 can provide @2 V is 2 A. This means, with one Track-And-Hold amplifier requiring a maximal current of 0.221 A, one ADP1741 can handle four units according to the rule mentioned beforehand ($I_{max\_ADP1741} = 2\,\text{A} > 2 * I_{tot}, I_{tot} = 4 \times 0.221\,\text{A} = 0.884\,\text{A}$).

3.3. PCB-Layout

3.3.1. Floor Planning

3.3.2. Transmission lines

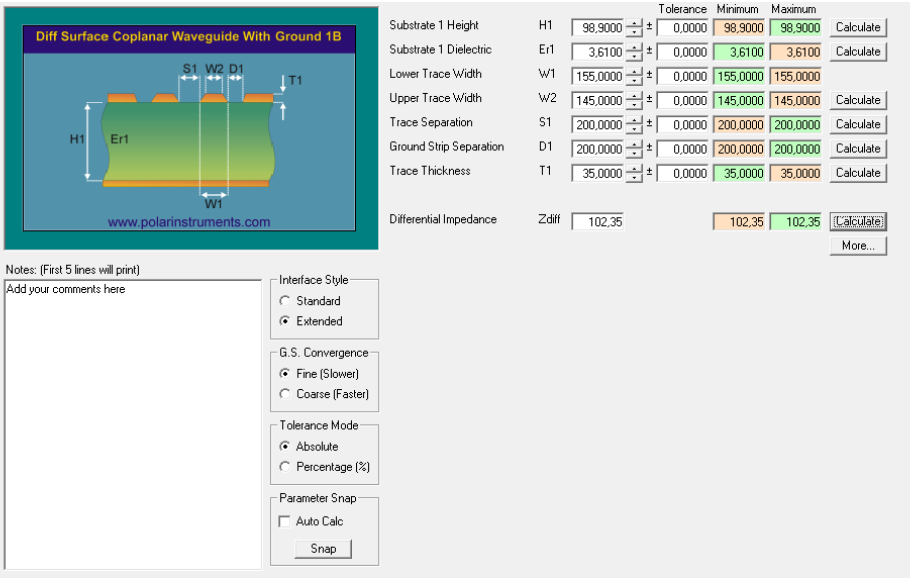


Figure 3.11.: Polaris Solver

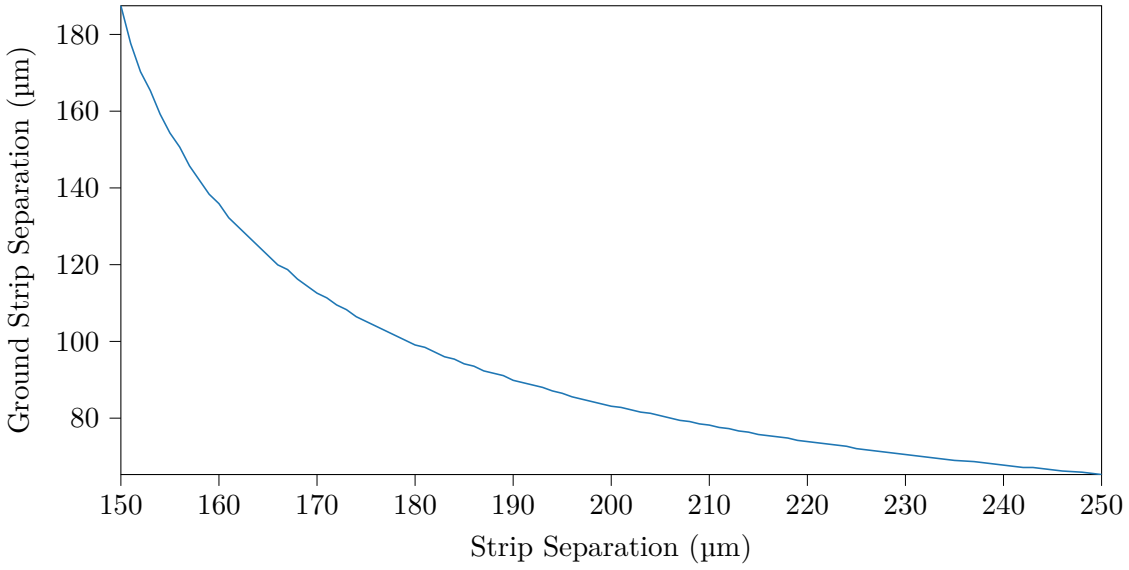


Figure 3.12.: Test Graph

3.4. Firmware

3.4.1. General Design

3.4.1.1. Firmware for Front-End Card

Clocking

SPI-Interface

3.4.1.2. Data Capture

Figure 4: **RF-ADC DDR Block Architecture for Quad RF-ADC Tiles**

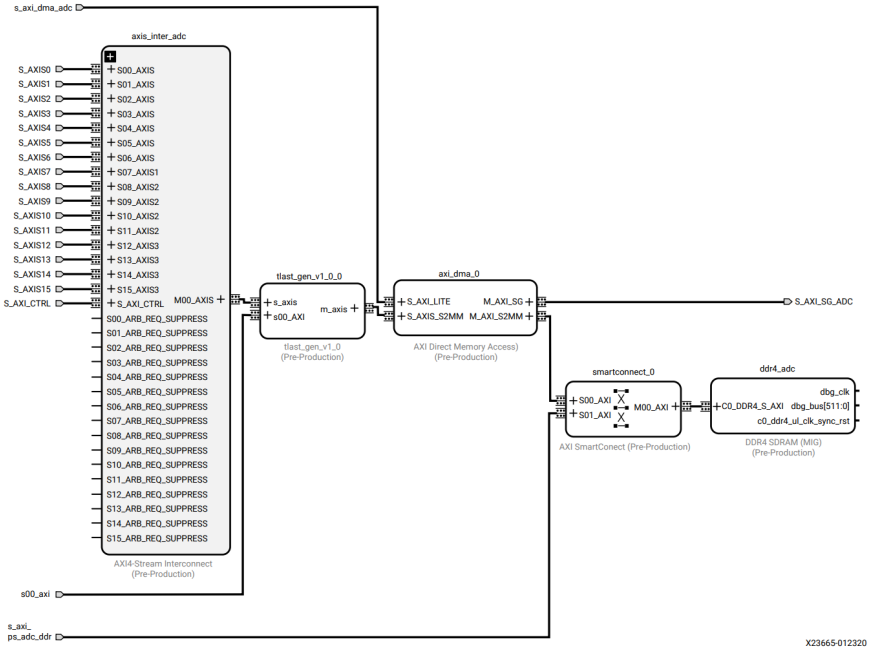


Figure 3.13.: Placeholder

3.4.1.3. Visualization

4. Characterization

4.1. Evaluation of the ZCU216 Board/ADCs

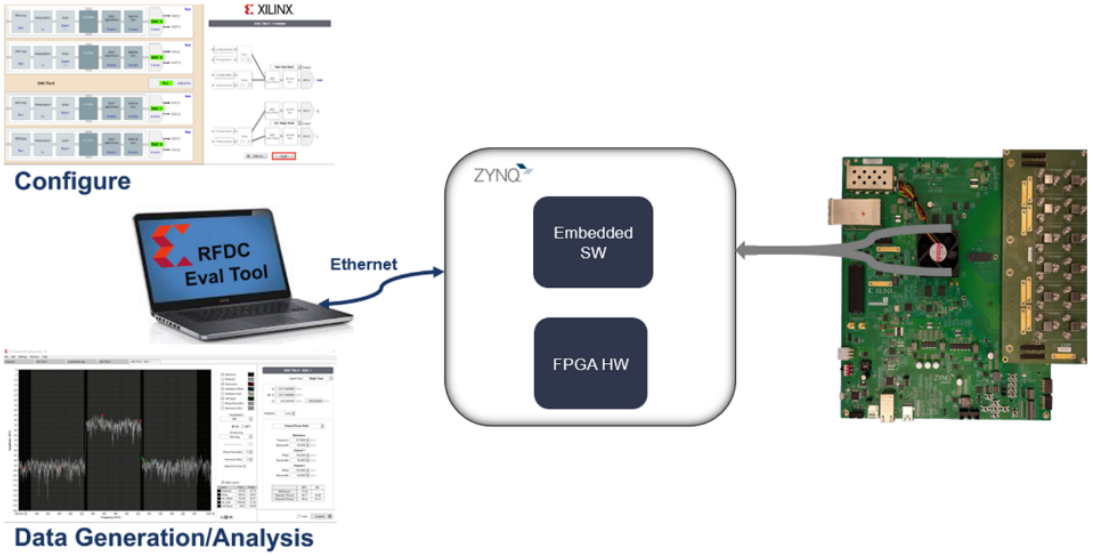


Figure 4.1.: RF DC Evaluation Tool architecture [**?**]

4.1.1. Measurements with Xilinx XM655 add-on card

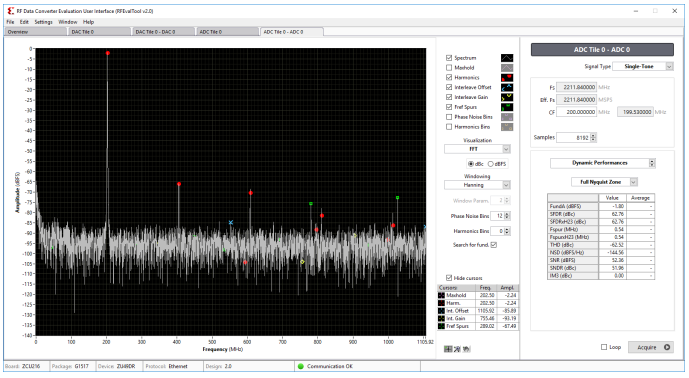


Figure 4.2.: RF DC Evaluation Tool GUI [**?**]

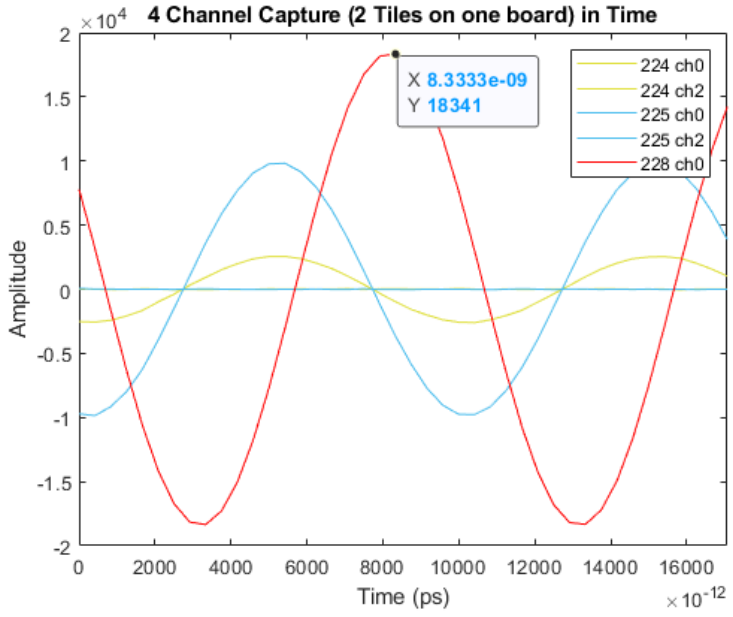


Figure 4.3.: Placeholder

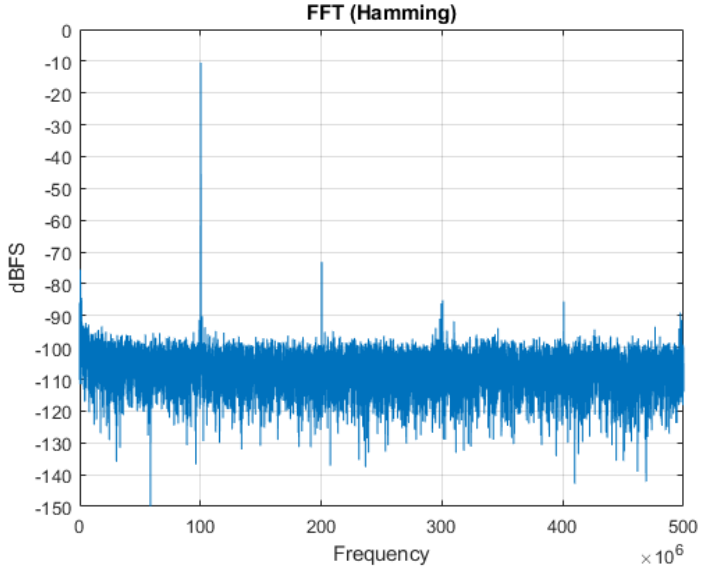


Figure 4.4.: Placeholder

5. Conclusions and Outlook

5.1. Measurements with the developed front-end card

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

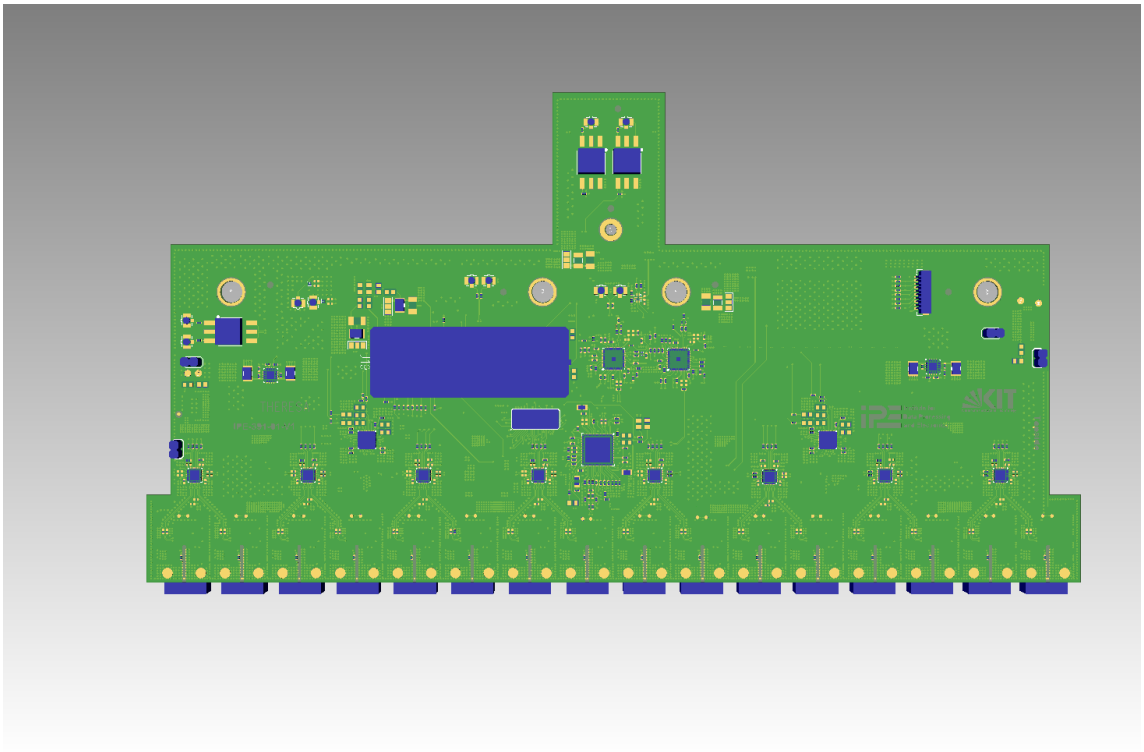


Figure 5.1.: wow

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(
    // data for respective delay chips
    input [10:0]        In_1, In_2, In_3, In_4, In_5, In_6, In_7, In_8,
    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: s
    LOAD_P0                     Load first half of Delay_X - which corresponds to data for Delay PD0 on delay chi
    LOAD_P1                     Load second half of Delay_X - which corresponds to data for Delay PD1 on delay ch
    SHIFT                       Shift bits for sending serial bitstream to SDIN, assert SLOAD -> to END
    END                         End transmission, deassert SLOAD, inform priority encoder about end of transmissi

*/
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
```