Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
http://ens.ewi.tudelft.nl/

CAS-MS-2015-10

M.Sc. Thesis

A 1 GSa/s Deep Cryogenic, Reconfigurable Soft-core FPGA ADC for Quantum Computing Applications

ing. S. M. C. Visser

Abstract

Analog interfacing is the only way to communicate with a quantum processor, whether it is applying qubit operations or reading their quantum states. There exist other applications where analog interfacing is abundant, e.g. sensor networks, automotive, industrial control, etc. In those applications the use of FPGAs is continuously growing, however a direct link between the analog world and the digital FPGA is still missing (except for the newest generation of FPGAs, where analog-to-digital conversion is present, but limited in performance). External analog-to-digital converters (ADCs) are combined together with the FPGA to form a complete, application specific, system. This system is thus limited in compactness, flexibility and reconfigurability.

To address those issues we propose an ADC architecture, implemented entirely in a conventional FPGA, that is fully reconfigurable and easy to calibrate. This allows one to alter the design, according to the system requirements. Therefore it can be used in a wide range of operating conditions, such as a harsh cryogenic environment, where we demonstrated that the FPGA is able to operate.

This architecture employs time-to-digital converters (TDCs) and phase interpolation techniques to reach a sampling rate, higher than the clock frequency, up to 1.2 GSa/s. The resulting FPGA ADC can achieve a 8 bit resolution over a 0.6 to 1.9 V input range. The system non-linearities are less than 0.45 LSB. The main advantages of this architecture are its scalability and reconfigurability, enabling applications with changing demands on one single platform.

A 1 GSa/s Deep Cryogenic, Reconfigurable Soft-core FPGA ADC for Quantum Computing Applications

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

EMBEDDED SYSTEMS

by

ing. S. M. C. Visser
born in Dordrecht, Netherlands

This work was performed in:

Circuits and Systems Group
Department of Microelectronics & Computer Engineering
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

TU Delft

Delft University of Technology

Delft University of Technology
Department of
Microelectronics & Computer Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **“A 1 GSa/s Deep Cryogenic, Reconfigurable Soft-core FPGA ADC for Quantum Computing Applications”** by **ing. S. M. C. Visser** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 13th of November 2015

Chairman: ____________________________
prof.dr. E. Charbon

Advisor: ____________________________
prof.dr. E. Charbon

Committee Members: ____________________________
asst.dr. F. Sebastiano

____
prof.dr. K.L.M. Bertels

Abstract

Analog interfacing is the only way to communicate with a quantum processor, whether it is applying qubit operations or reading their quantum states. There exist other applications where analog interfacing is abundant, e.g. sensor networks, automotive, industrial control, etc. In those applications the use of FPGAs is continuously growing, however a direct link between the analog world and the digital FPGA is still missing (except for the newest generation of FPGAs, where analog-to-digital conversion is present, but limited in performance). External analog-to-digital converters (ADCs) are combined together with the FPGA to form a complete, application specific, system. This system is thus limited in compactness, flexibility and reconfigurability.

To address those issues we propose an ADC architecture, implemented entirely in a conventional FPGA, that is fully reconfigurable and easy to calibrate. This allows one to alter the design, according to the system requirements. Therefore it can be used in a wide range of operating conditions, such as a harsh cryogenic environment, where we demonstrated that the FPGA is able to operate.

This architecture employs time-to-digital converters (TDCs) and phase interpolation techniques to reach a sampling rate, higher than the clock frequency, up to 1.2 GSa/s. The resulting FPGA ADC can achieve a 8 bit resolution over a 0.6 to 1.9 V input range. The system non-linearities are less than 0.45 LSB. The main advantages of this architecture are its scalability and reconfigurability, enabling applications with changing demands on one single platform.

Acknowledgments

After almost a year of work and achieving fantastic results, it finally comes to an end. But I couldn't have done it without the help and support from all colleges, friends and family. First I would like to thank everyone I worked with. I enjoyed the international diversity and I learned much from and about all your cultures. Now I finally know how a respectable lunch should look like!

I would like to thank my advisor Edoardo Charbon for inspiring me. You taught me that nothing is impossible if you work hard towards your goals.

I would like to thank Harald Homulle for being my daily supervisor. You always helped me out with design problems and with reflecting on strange measurement results. When I needed more devices or components you always managed to find or order them.

In special I also would like to thank Bishou Patra for designing the PCB that I could use. You saved me a lot of time even with the debugging subtracted! Your PCB made it possible to do my measurements at 4 degrees Kelvin.

That brings me to the next persons, I would like to thank Enrico Prati and Giorgio Ferrari from Politecnico in Milano for hosting me and Harald to perform cryogenic measurements. It was amazingly well prepared when we arrived and I am very thankful that we could alter your probe, that was very generous.

But I wouldn't have met all the previous mentioned people as it weren't for my parents that supported me throughout my entire education. Without your support I would never have started my masters.

The only one left to thank now is my girlfriend, Daphne for supporting me throughout my entire master. You have helped me alot especially on the moments that I needed it the most! My apologies for all the times that I couldn't go out with you because I had to finish my coding, measurements or thesis. I will make it all up to you in the coming months!

ing. S. M. C. Visser
Delft, The Netherlands
13th of November 2015

Contents

List of Figures

List of Tables

Acronyms

ADC	Analog to Digital Converter
BRAM	Block RAM
DNL	Dynamic Non-Linearity
FIFO	First-In-First-Out
FFT	Fast Fourier Transformation
FPGA	Field-Programmable Gate Array
LUT	LookUp Table
LVDS	Low-Voltage Differential Signaling
INL	Integral Non-Linearity
MUX	MUltipleXer
PDF	Probability Density Function
SINAD	SIgnal-to-Noise And Distortion ratio
SNR	Signal-to-Noise Ratio
TDC	Time to Digital Converter
THD	Total Harmonic Distortion
UART	Universal Asynchronous Receiver/Transmitter
VHDL	Very-high-speed integrated circuit Hardware Description Language

Introduction 1

1.1 Motivation

The interaction between the analog and digital worlds has always been a challenge. Special design techniques are required to make a good analog front-end to the digital system. As analog-to-digital conversion is needed in numerous applications, ranging from sensor nodes, industrial control to (quantum) physics experiments, various ADCs with a wide range of specifications exist.
Each application has its own set of requirements, making it hard to reuse the same ADC design. The standard approach would be to combine an off-the-shelf ADC that matches the requirements with the digital system, either a system-on-chip or a reconfigurable device. After the system is set, the specifications can no longer be easily altered. Therefore the system, using an external ADC, is limited in terms of flexibility, scalability, reconfigurability and overall system size.
A better approach would be to create a more flexible system with an integrated reconfigurable ADC. This approach enables a single platform suitable for many applications that only needs to be adapted with a firm- and/or software change and a calibration in the new environment. The most suitable platform to build a flexible system is in a reconfigurable logic device, i.e. a CPLD or FPGA.
The need for interaction with the analog world and the digital reconfigurable devices has been recognised by FPGA manufacturers. Xilinx includes an on-chip ADC (XADC) in the 7 Family generation of FPGAs, Altera in the MAX-10 FPGA and Microsemi in their Fusion Mixed Signal FPGAs. Although this integration has some advantages, such as lower power, smaller system size and a simple interface, it is still not flexible. It rules out the possibility to change the conversion rate, the number of ADC channels, and above all, it takes die space solely reserved for the ADC, whether it is used or not.
In this thesis, we propose a solution using a FPGA to create a soft-core ADC architecture. Except for some small resistors on the PCB, the ADC can be completely integrated into the reconfigurable hardware blocks. Therefore the ADC can be easily interfaced with the remainder of the digital circuitry, it can be scaled to the required sampling rate or resolution and it even allows ADCs with different specifications in one system.
Above all, our approach allows calibration to each new environment the system is operating in, i.e. changes in voltage, temperature or chip can be calibrated out. We aim to show the effectiveness of our calibration techniques by operating the ADC both at room temperature and in a deep cryogenic environment at 4 Kelvin.

1.2 Related work

The idea to create an ADC in a FPGA isn't something new. Sigma delta ADCs have been implemented in [1–3]. The conversion speed of these converters is fairly low with 50KSa/s. The goal of these papers was to implement an ADC inside the FPGA to reduce the PCB size or to reduce development time.
With voltage to time converters, much higher sample rates can be achieved, in [4] multiple ADCs were created with 3 resistors and 1 capacitor. With those components a RC curve was created on multiple LVDS components. The ADC inputs were also connected to the LVDS inputs. By measering the time between start of the RC curve and the first edge in the LVDS output an analog to digital conversion could be performed. With this method a 6 bit, 22.5MSa/s conversion was reached.
The previous paper however could have created faster and more accurate ADC's if they would have used carry-chain based time to digital converters [5,6]. Although those are more expensive in therms of utilization of the FPGA.
That is exactly what has been done in [7], here they created 16, 7 bits, 62.5MSa/s ADCs with one FPGA. But they still used 4 resistors and a capacitor for every channel.
One of the latest works [8] created a 7 bits 200MSa/s ADC with only 1 resistor. Instead of adding a capacitor they used the parasitic capacitance of the LVDS input. This work is the closest competitor and will be used as basis for this thesis.
Most quantum computing applications take place at sub-kelvin temperatures. As it is hard to let electrical signals cross 300K. There has been a growing interest for cryogenic electronics. Over the years multiple papers [9–11] have been released that demonstrated functional FPGA's down to 77K. It is already proposed to bring the electronics for quantum computing to 4K [12], and very recent [13] demonstrated partial functionality of the ARTIX-7 near 4K.
If an FPGA implemented ADC could function at 4K, the flexibility of the FPGA can be used to calibrate this system to the changes at that temperature.

2 Background

This chapter summarizes the literature of this thesis. Here you will find the necessary definitions that come with characterizing an ADC or TDC. Also different ADC architectures with their FPGA implementation are discussed.

2.1 Characterization of an analog-to-digital converter

It is always a challenge to compare multiple devices. ADCs are no exception. There is a great variety of parameters on which ADCs can be rated besides range and sample rate, in this chapter the INL, DNL and ENOB will be introduced. The importance of each parameter is also varying between projects. IEEE has specified a methodology [14] to measure an ADC. This section includes some of the methods to measure the parameters of an ADC.

2.1.1 Differential non-linearity (DNL)

The differential non-linearity is defined as the difference between the actual step width and ideal step width. The official way to measure the DNL is by it's transition levels. The formula to derive it can be found in Equation 2.1. For a perfect ADC all DNL values are 0, corresponding to the ideal step width of 1 LSB. When a step is missing in the outputs it has a value of -1 and when a step is double the width, it has a value of 1.

$$DNL\left[k\right] = \frac{T\left[k+1\right] + T\left[k\right] - Q}{Q} \tag{2.1}$$

Where:

Q	is the ideal width of a code bin
$T\left[k\right]$	is the input value corresponding to the transition from k-1 to k

To properly characterize the DNL in the presence of noise (Figure 2.1 and 2.2) we use the so-called histogram mehtod. The histogram method implies that if an input signal has a known probability density function (PDF). The output of the ADC should also have a similar PDF. For the ADC we are able to use a ramp function that starts below the range and ends above the range. Thus creating a uniform PDF for all voltages. Such a system can be found in Figure 2.3 and an example of a measured PDF in Figure 2.4. The min and max output of the ADC will be more frequent, due to exceeding the range, but all values between the min and max should have the same density.

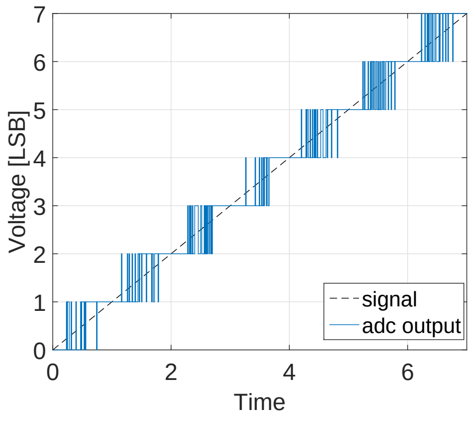


Figure 2.1: example where ADC output transition can be calculated

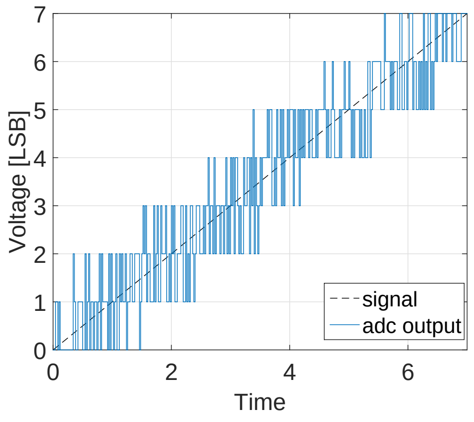


Figure 2.2: example where it is better to use the histogram method

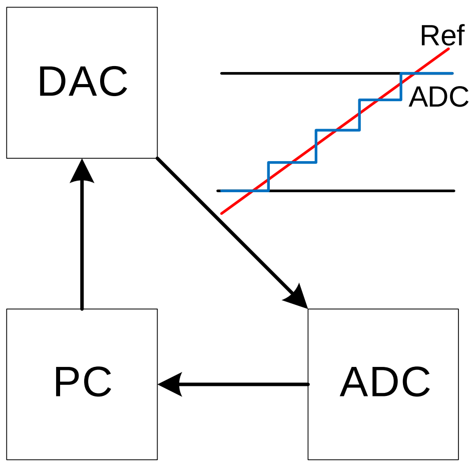


Figure 2.3: Schematic for PDF measurement

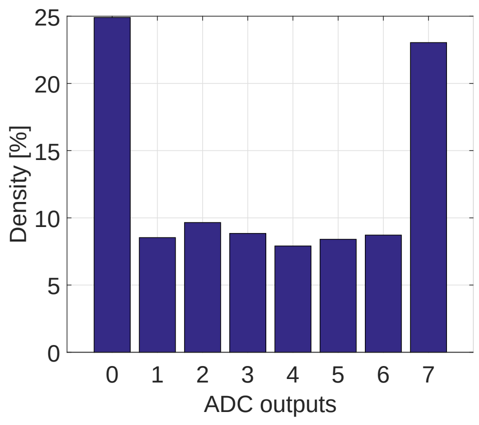


Figure 2.4: An example of a PDF from a ramp

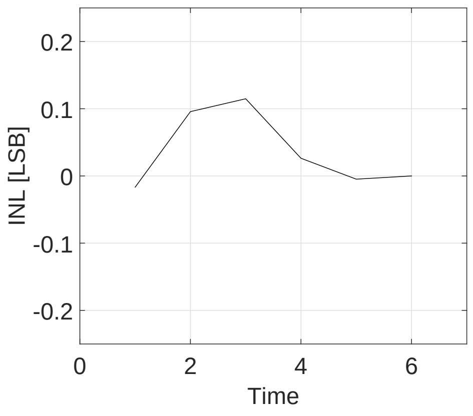


Figure 2.5: INL from the PDF

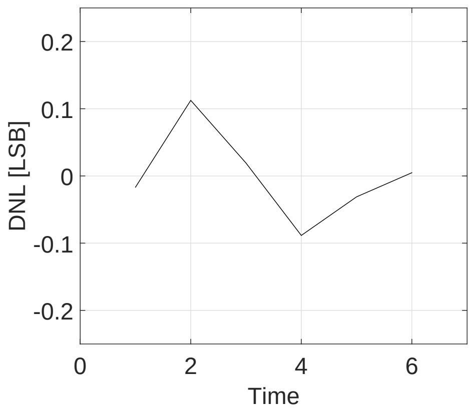


Figure 2.6: DNL from the PDF

To extract the DNL from the PDF, Equation 2.2 can be used.

$$DNL\left[k\right] = \frac{D\left[k\right]}{M} - 1 \tag{2.2}$$

Where:

$D\left[k\right]$	is the density of a code bin
M	is the mean density of all code bins between the min and max code bin

2.1.2 Integral non-linearity (INL)

The integral non-linearity is the difference between the ideal and measured code transition levels after correcting for static gain and offset. The static gain and offset correction basically say that the ideal transitions levels can be determined by a best fit line. The INL can be expressed as a function of the code bins or as worst case values. When there isn't much noise on the ADC output, such as in Figure 2.1, it is possible to calculate the transition levels. In that case Equation 2.3 can be used to calculate the INL. When there is significant noise on the output such as in Figure 2.2, it is better to use the histogram method. The INL can then be calculated by integrating the DNL as is shown in Equation 2.4.

$$INL\left[k\right] = 100\% * \frac{\epsilon\left[k\right]}{V_{FS}} \tag{2.3a}$$

$$\epsilon\left[k\right] = \frac{T\left[k\right] + V_{OS} - T_{NOM}\left[k\right]}{Q} \tag{2.3b}$$

Where:

Q	is the ideal width of a code bin
$T\left[k\right]$	is the input value corresponding to the transition from k-1 to k
V_{OS}	is the output offset in units of the input quantity, nominally equal to zero
V_{FS}	is the full-scale range of ADC input units
$T_{NOM}\left[k\right]$	is the best fit line through all codes

$$INL\left[k\right] = \sum_{i=1}^{k} DNL\left[i\right] \tag{2.4}$$

2.1.3 Signal to noise and distortion ratio (SINAD)

The SINAD is the ratio between the signal and the distortion plus noise. The SINAD can be calculated in multiple ways. For sinusoidal signals we perform the FFT of the converted digital signal. Another method is by fitting a sinusoidal signal of known frequency onto the data. Afterwards all deviations from that fitted sinusoidal signal are the noise and distortion. Fitting a frequency on the sample data can be done using the least-square-method. In Equation 2.6a we can see an equation that describes a known frequency on the left and the unknown composition of amplitude, phase and offset on the right. On the right there is a formula with the unknowns A, B and C. These parameters can be found with the least-square-method. The advantage of the fitted sinus method is that it can also work with aperiodic data. Where the FFT can only work with periodic data.

$$SINAD = \frac{P_{Signal}}{P_{noise} + P_{distortion}} \tag{2.5}$$

$$a * sin(x + p) + o = A * sin(x) + B * cos(x) + C \tag{2.6a}$$

$$a = \sqrt{A^2 + B^2}, p = atan2(B, A), o = C \tag{2.6b}$$

2.1.4 Effective number of bits (ENOB)

The ENOB gives a performance indication of your ADC. It can be calculated with Equation 2.7, here the signal to noise ratio is used to determine the maximum resolution in which a signal can be distinguished from noise.

$$ENOB \;=\; \frac{SINAD - 1.76}{6.02} \tag{2.7}$$

2.2 Analog-to-digital converters

Analog-to-digital converters are widely used. There are a great number of different ADC types. Every type has its own advantages and disadvantages. This section includes analyses of serveral ADC structures. They are compared on required resources, conversion speed, resolution and by the possibility to implement them on an FPGA.

2.2.1 Flash ADC

Flash ADC's are the fastest, they can be made with conversion speeds up to tens of gigasamples per second (GSa/s). The downside of this architecture can be found in the required resources. The number of resistors, comparators and logic complexity increases exponentially with the resolution, $(2^N - 1)$ where N is the number of bits.
Because this exponential demand for resources, flash ADCs usually have a smaller resolution than ADCs with other architectures in the same price range. When trying to make a flash ADC with more bits the chance of mismatches increases. When this chance isn't negligible anymore, the simple decoding of Figure 2.7, starts producing errors. These errors are discussed in subsubsection 2.4.2.1.

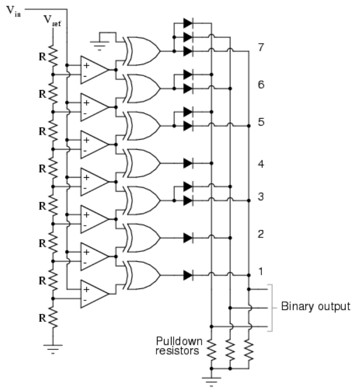


Figure 2.7: The simple decoding of a flash ADC [1]

Implementing a flash ADC on an FPGA is possible but not easy. An FPGA has LVDS inputs that could be used as comparators. The resistors could be mounted outside of the FPGA. However all FPGA pins have a capacitance around 8 pF [15]. This makes it hard to create, for example, a 6 bit ADC. Such ADC would have a 512 pF input capacitance, require 64 LVDS inputs and 65 resistors. The big input capacitance would reduce the usefulness of the high sample rate because it is basically a low pass filter.

[1]source: http://www.allaboutcircuits.com/textbook/digital/chpt-13/flash-adc

2.2.2 Sigma-delta modulation

The sigma-delta modulation is an ADC with a feedback control loop. The components of a sigma-delta modulator are an analog summing circuit, a comparator, a voltage reference, a switch and an integrator. In short, the sigma-delta works by comparing the analog input with the integrated value of the DAC. When the integrated value is lower than the analog input, the DAC will increase the value of the integrator. This continues until the integrated value is greater again.

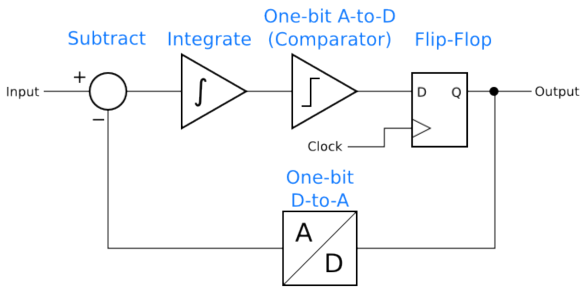


Figure 2.8: The schematic of a sigma-delta ADC [2]

In Figure 2.9 it can be seen that the digital output will follow the value of the signal with an error of at most 1 LSB, except when the signal increases faster then 1 LSB per sample, in which case a slope overload occurs.

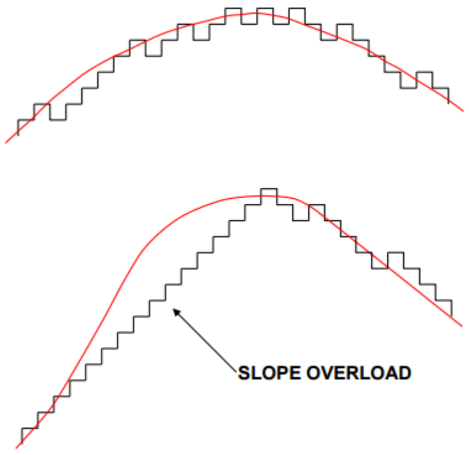


Figure 2.9: The signal of a sigma-delta ADC [3]

The sigma-delta modulation can also be implemented on an FPGA [1–3], only the time from comparator to integrator costs some time. For example, if an FPGA operates at 400MHz and takes 2 clock cycles to read the input, 1 cycle to process the input and write the new DAC output and 1 cycle to settle the new DAC value. The sample rate will be 100 MSa/s. This is still 10 times lower than the goal of 1 GSa/s.

[2]source: http://skywired.net/blog/2011/05/introducing-the-delta-sigma-modulator
[3]source: http://www.analog.com/media/en/training-seminars/tutorials/MT-022.pdf?doc=cn0354.pdf

2.3 Characterizing time-to-digital converters

TDCs are components to quantize the time between two events. These events can be clock events or incoming signals. TDCs have just as ADCs an INL, DNL. The ENOB however is impractical to measure with a TDC. So instead one can express the TDC's effective resolution with a single shot time resolution.

2.3.1 INL and DNL

As explained in subsection 2.1.1 and 2.1.2, it is possible to retrieve the INL and DNL from a PDF. For a TDC it is difficult to generate a ramp function in time length. But a randomly occurring pulse is relative easy to create. By measuring a random frequency the outcome should have a uniform PDF. So Equation 2.2 and 2.4 hold also for the TDC case.

2.3.2 Single-shot time resolution (SSTR)

To measure the single-shot time resolution it is required to have a setup where the same input value can be generated multiple times. Afterwards, the variation (σ) can be calculated of the output values, the variation of the SSTR can be calculated with Equation 2.8. When different parts are used to measure different outcomes, such as with a delay line of subsection 2.4.2. It can be useful to measure the SSTR for multiple inputs. Afterwards the SSTR can be coupled to the input to express the accuracy over the entire range.

$$SSTR = \frac{\sigma * LSB}{\sqrt{2}} \tag{2.8}$$

2.4 Time-to-digital converters

This section will introduce multiple ways to create a time-to-digital converter(TDC) on an FPGA. Because of the goal to create an GSa/s ADC with an TDC this section will aim at TDC's with a sub nanosecond resolution.

2.4.1 Multi-phased clock counting

A simple way to increase the resolution of a counter is by counting on multiple phases. In Figure 2.10 an example of an multi phase counter is shown. This schematics originates from [4], in that paper they used this TDC to create multiple ADCs. The amount of resources required to make such a TDC in an FPGA are little compared to a delay line. For FPGA TDCs with a resolution just below the nanosecond, this is the most efficient structure. But every time the resolution doubles, the number of required phase shifted clocks also doubles, making this method impractical beyond a certain resolution.

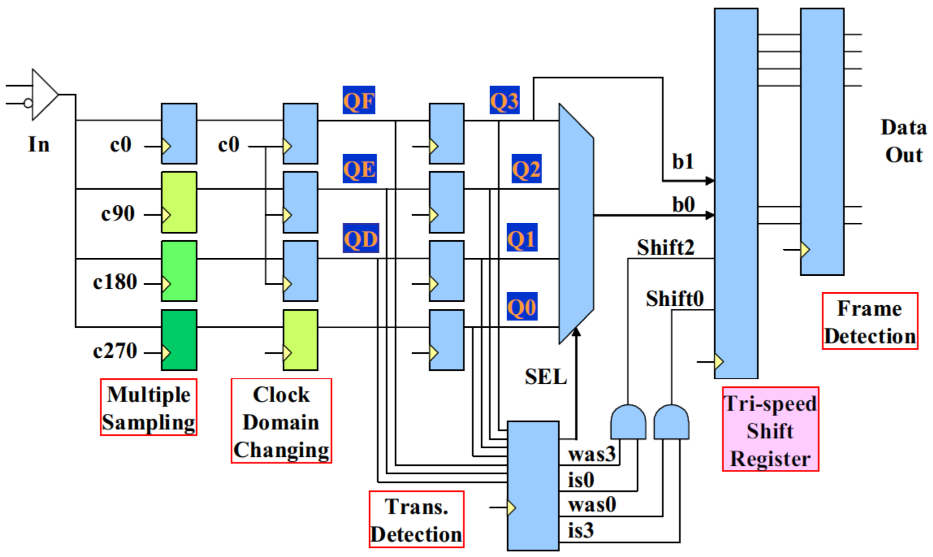


Figure 2.10: Multi phased clock TDC design, source [4]

2.4.2 Delay Line

The mechanism of the delay line is comparable with the flash ADC of subsection 2.2.1. Instead of resistors, delay components are used. In Figure 2.11 it can be seen that a start pulse enters the delay line and gets delayed by the delay blocks τ. After some time a second event (a clock or stop signal) occurs to write the delay line data into the flip flops. The time difference between start en stop determines how many flip-flops will be set to 1. The time between start and stop can be calculated with Equation 2.9.

$$\triangle t = \frac{\tau}{Q} \tag{2.9}$$

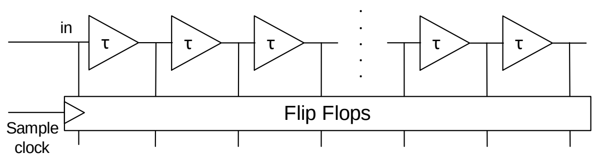


Figure 2.11: Schematic of a delay line

2.4.2.1 Bubbles in carry-chain

Because of clock skew and jitter there is a possibility that not all flip flops are set at the same time. This results in a fussy transition, as can be seen in the example of Figure 2.12. To find the best transition point some form of filtering is required. subsection 3.2.1 proposes multiple strategies to solve this problem.

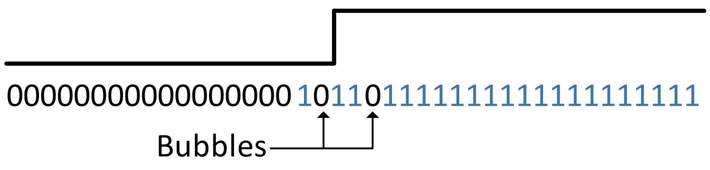


Figure 2.12: Example of bubbles

Designs 3

The goal of this thesis is to create a giga sample ADC by building on the work of Homulle [8]. In his work the analog-to-digital conversion is done by integrating time. Here we tests the hypothesis that when using a time to voltage conversion, it is possible to produce a data point that consists out of a time and a voltage. Then by using multiple of these converters the data points can be merged to create an analog-to-digital converter with a much higher sampling rate.

3.1 Single LVDS ADC

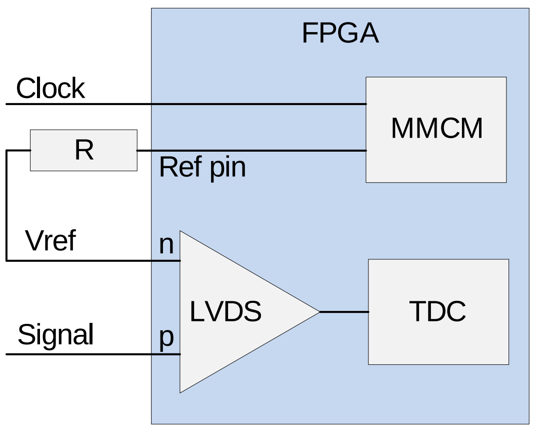


Figure 3.1: Single LVDS schematic

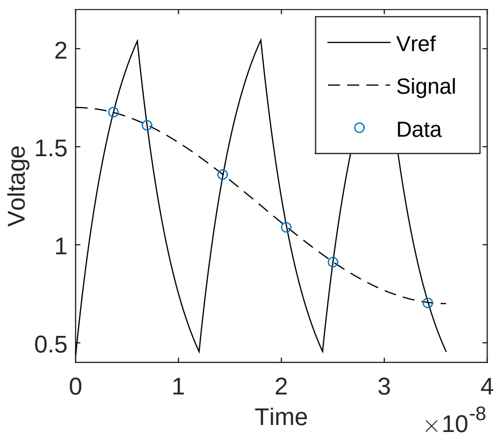
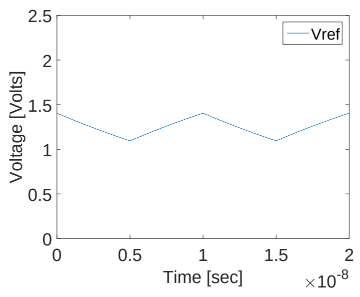


Figure 3.2: Single channel sampling

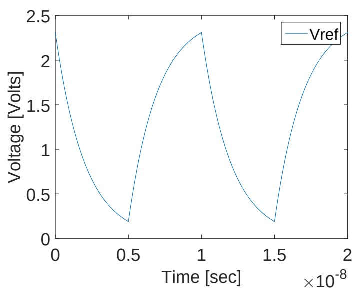
Figure 3.1 contains the schematic of a single LVDS ADC. The reference pin charges and discharges the parasitic capacitance of the LVDS input. Figure 3.2 shows the resulting RC curve. The data dots in this graph are the points where the LVDS output switches value. By measuring the time of these events we can infer the original voltage. Because the charging and discharging curves are clearly different, they both require their own calibration to convert a time to a voltage.

3.1.1 Best RC curve

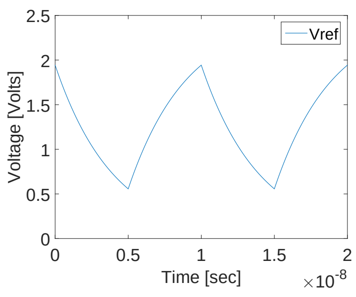
The parasitic capacitance of the LVDS input is a fixed value estimated at 9 pF (see subsection 4.3.1). But the resistor can be chosen freely (see Figure 3.1). This way we can create every RC time for the reference signal. In this section we will describe a reference signal by its number of RC periods in a reference period. The reference period is reconfigurable in the ADC we will create.
Figure 3.3 shows the dependence of the Vref time response from the ratio of the sampling period Ts to RC time constant. The RC time constant has two opposite effects on the ADC. By charging the capacitance to fast, RC distortion occurs (an example in Figure 3.3b), which results in a non-linear voltage-to-time conversion. Moreover the absolute voltage range becomes bigger.



(a) $\frac{Ts}{RC} = 0.5$



(b) $\frac{Ts}{RC} = 5$



(c) $\frac{Ts}{RC} = 1.3$

Figure 3.3: Absolute RC response in time

To make the quality of the ADC better in terms of SNDR, the influence of static noise should be reduced. Sources of this noise are for example thermal noise or interference from other signals on the PCB. The influence of that noise is least when we maximize the smallest voltage step per time unit. As that value is related to the range and RC distortion, we can find an optimum for RC. In Figure 3.4 one can see that the optimum is at Ts/RC = 2.5.

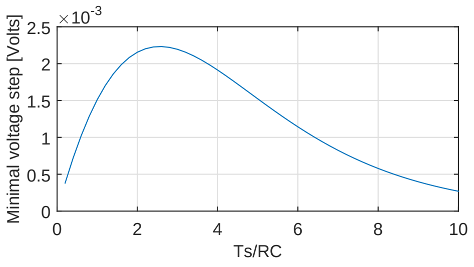


Figure 3.4: Reference periods vs minimal voltage step

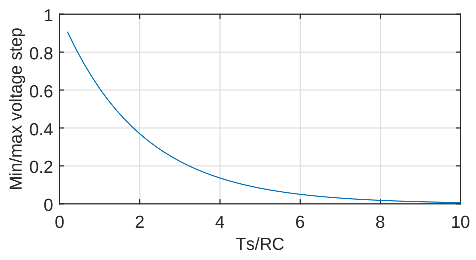


Figure 3.5: Minimum voltage step normalized to the by maximum voltage step

The second problem is the ratio between minimum and maximum voltage steps per time unit. This value is only influenced by the RC distortion. Figure 3.5 shows these ratios, already when Ts/RC = 1.3, the maximum step is 2 times bigger then the minimum step. To explain the problems of the RC distortion Figure 3.7 shows the ADC output of a sinusoid, the reference signal had a Ts/RC = 5. Here is a noticeable difference between the data acquired from the rising curve and the falling. This can be explained by the time-to-voltage conversion (see Figure 3.6). With the same time resolution, the rising curve has a higher accuracy at the top of the Vref signal.

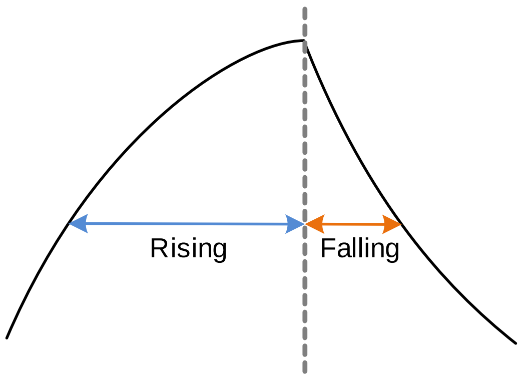


Figure 3.6: The Vref top, notice that the time-to-voltage step has a higher resolution for the rising curve

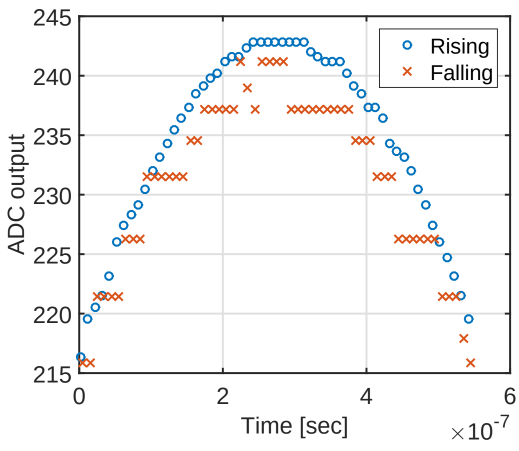


Figure 3.7: Merging measurement outputs of a Vref signal with significant RC distortion

3.1.2 Sample rate, accuracy and power consumption

The power dissipation, accuracy and sample speed are correlated with each other. To achieve an increase in the measurement accuracy, thus reducing quantization noise, the sampling period, Ts needs to be increased. When increasing the sampling period, the reference period will also increase. This results in a higher RC/Ts. To counter this effects the resistor needs to be increased as well. This increase in resistance results in a lower current and power dissipation. Figure 3.8 shows the resolution in bits of the conversion as a function of the sampling rate. This graph is based on a time resolution of 17 ps.

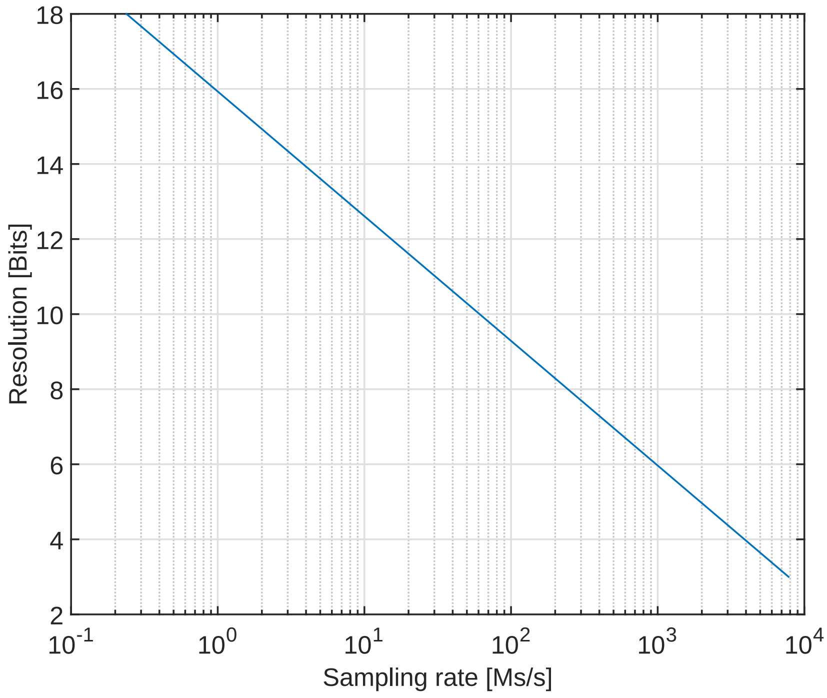


Figure 3.8: Single channel resolution vs sampling rate

3.1.3 Noise

All digital systems have margins in which they are specified to operate. Power supplies may have noise and ripples, clocks can have some jitter and the outputs will have similar impurities. That's where this project will find the borders of minimal noise with off-the-shelf components.

3.1.3.1 Power supply noise

As the power consumption of the FPGA is strongly influenced by its activity. It is difficult to minimize the noise on the output voltage. With clocks running over 100MHz and a power consumption that is influenced by events. Ripples up to 20 mV (peak-peak) have been measured.

3.1.3.2 Voltage drop

As this setup will be tested on a prototype board that is powered from long cables. There is a significant voltage drop on all cables due to intrinsic resistance of the cables. Combine this with a varying power consumption and this could significantly change the RC curve or effect the logic speed.

3.1.3.3 Thermal noise

The warmer an environment the more noise is measured over the components of a system. For this system the thermal noise applies on the reference signal. This signal is actually just an RC oscillation. This means that the thermal noise is only influenced by the KTC noise [16]. For the 9 pF parasitic capacitance we expect the noise is less then 20 μV according to Equation 3.1. This becomes insignificant compared to the noise of the power supply.

$$v_n = \sqrt{\frac{k_B T}{C}} \tag{3.1}$$

3.1.3.4 Jitter on clock

Normally, jitter only influences the margins for the timing constrains. Thats is why most Xilinx components have two settings, namely high performance and low power. With the low power setting the component uses less power in exchange for more jitter. When trying to read-in a carry-line, the FPGA was only designed to do this within the timing contains. However, now it is of great importance that the jitter is as low as possible. This project crosses the boundaries within which an FPGA is designed to operate.

3.1.3.5 Jitter on interconnects

After the LVDS component, the signal needs to travel over the chip to the input of the TDC. This journey often takes more then 5 ns. When comparing this massive 5 ns to the tiny TDC resolution of 17 ps are notices that an increase or decrease of 0.3% is equal to 1 LSB. In subsection 4.3.2 it is determined that the speed of the interconnects is related to the internal voltage.

3.1.3.6 LVDS interference

LVDS works by measuring the direction of the current between the two input pins. This can only be done if there is a current between those pins and when there is a current between those pins they also influence each other. This means that the input signal can actually change the reference signal.

3.2 FPGA implemented TDC design

One of the most important features of the carry-chain is the uniformity of the delay elements. The higher the sampling frequency the shorter the carry-chain. The ARTIX-7 used for this project has a carry elements delay of 17 ps. Figure 3.9 shows a graph of the sampling frequency vs carry-chain length. The line in this graph covers the points where the total delay of the carry-chain equals the sampling period. In this condition the dead time in the raw data vanishes. When comparing the sampling frequency vs utilization, a shorter carry-chain requires less resources for the decoder and involves simpler calculations. The limit is the maximum FPGA frequency of 450 MHz.

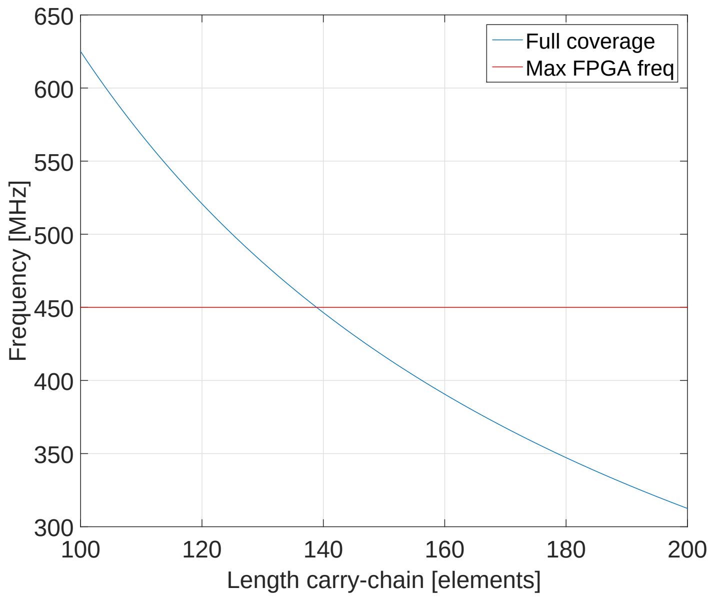


Figure 3.9: The sampling frequency vs carry-chain length for full coverage

3.2.1 Carry-chain read-out

The read-out of the carry-chain is relatively simple. One needs to connect the input signal to the input of the carry-chain. Preferably a signal that hasn't been latched into a clock domain (for example directly from a LVDS input). When reading a signal that has been clocked into a clock domain, timing violations will arise (These violations can be prevented by making the sample and signal clock asynchronous).
After the input signal is connected to the first element of the carry-chain. All carry elements need to be connected together. When that's is done all first flip-flops need to be placed on the same slice as the carry elements. Then a second flip-flop is required to eliminate metastability. A schematic overview of the carry-chain read-out can be found in Figure 3.11.
Figure 3.10 shows a Vivado implementation of a carry-chain read-out. This image also shows the structure of the XC7A100T (ARTIX-7). The FPGA is 200 by 80 slices and every slice contains 4 carry elements. Making it possible to create a carry-chain of 800 elements. However, there are 8 clock domains, within a clock domain there is minimal clock skew. So, ideally a carry-chain is routed in only 1 clock domain. Every clock domain is 50 slices high, making it possible to create a carry-chain of 200 elements in one clock domain.

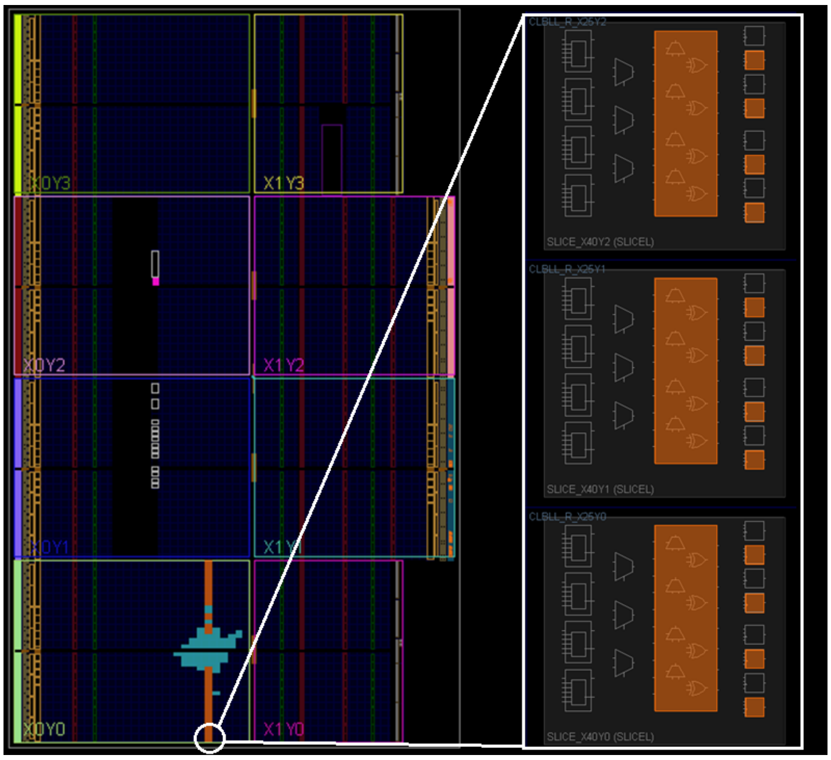


Figure 3.10: schematic view of vivado carry-chain

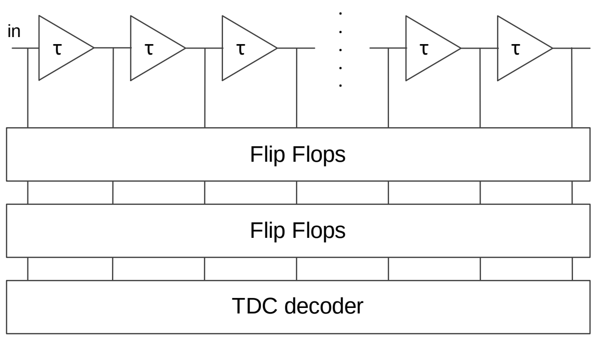


Figure 3.11: schematic view of a carry-chain read-out inside an FPGA

After the second flip-flop, the data is decoded. This is a heavy process due to the amount of data involved. For example a 200 elements long carry-chain sampled at 400 MHz, produces 80 Gb/s. Processing this much data has a big influence on the FPGAs power consumption and resources. Therefore 2 carry-chain decoder structures were tested. Afterwards we judged them on required resources and maximum frequency.

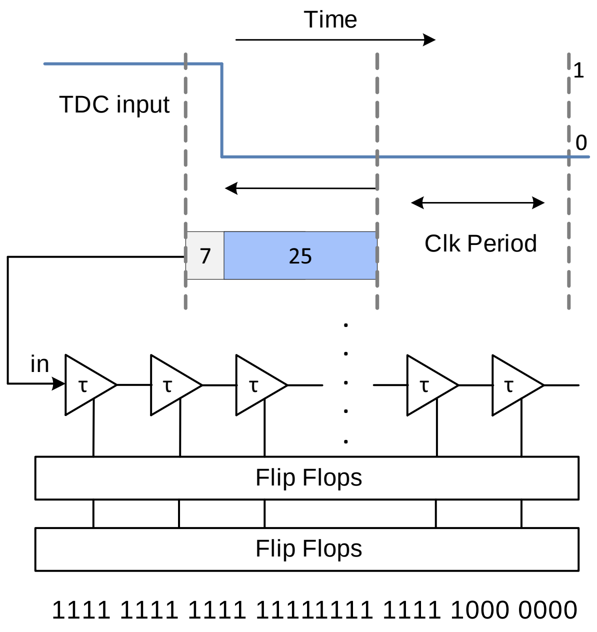


Figure 3.12: A timing explanation of the carry-chain

The read-out of the carry-chain is triggered by the FPGA clock, the start however isn't an active switch but is determined by the delay of the carry-chain (see Figure 3.38 and 3.39). In Figure 3.12 is an demo readout of a 32 element carry-chain. The data of the first carry-element is always the most recent data, making the end of the carry-chain the oldest. So, an edge, fast after the previous clock cycle, appears at the end of the carry-chain. In subsection 3.2.2, 3.2.3 and 3.2.4 we will decode this data.

Another property to judge a carry-chain decoder on is how it handles bubbles. Bubbles are created by clock-skew and clock-jitter. An example of a bubble in the carry-chain output can be found in figure 3.13. Here one can see that there are a few bumps in the transition of the signal. The bubbles can be ignored or filtered out. One method for example is a median filter implemented by counting the number of ones.

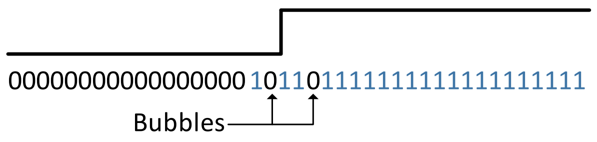


Figure 3.13: Example of bubbles

Creating a long virtual carry-chain can be required due to the physical limit of 800 carry elements, but it is also more efficient in resources. There have been multiple works [17], [18] where the range of an TDC was extended with a coarse counter and an event trigger in the carry-chain decoder. This requires, that after every measurement a decision has to be made about the usefulness of the data. For this work the decoder output will be added for a fixed number of clock cycles. An example can be found in Figure 3.14. Because input edges can occur at the same time as an clock cycle, can bubbles be filtered out by adding the results. This also prevents the problem of choosing an event in the case of multiple events.

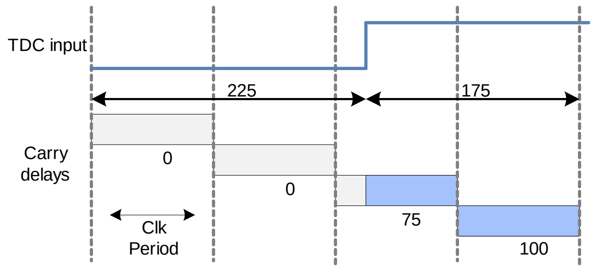


Figure 3.14: Example of a virtual carry-chain, here 4 samples of a 100 elements carry-chain are combined to create a virtual carry-chain of 400 elements

3.2.2 Count-ones carry-chain decoder

The count-ones-decoder works by counting the number of ones in the output. Figure 3.15 shows that this architecture minimizes the length over which data needs to be send, because every time two parts are added to form new data, data can stay locally in the FPGA, making this a simple and scalable architecture.
The implemented version of this decoder has 2 parameters which can be set, block size and number of blocks. The block size refers to the number of bits that is converted to a number in the first step of the decoder. In Figure 3.15 a block size of 3 is used. The number of blocks, determines out of how many counting blocks the decoder exists, thus indirect it also determines the carry-chain length.
By this counting mechanism, bubbles in are averaged out, it behaves as a median filter. Especially in terms of clock-skew this is the best solution. For the implementation of an ADC this decoder can be used to get the time of a transition from 0 to 1 or from 1 to 0, under the precondition that the direction of the transition is known.

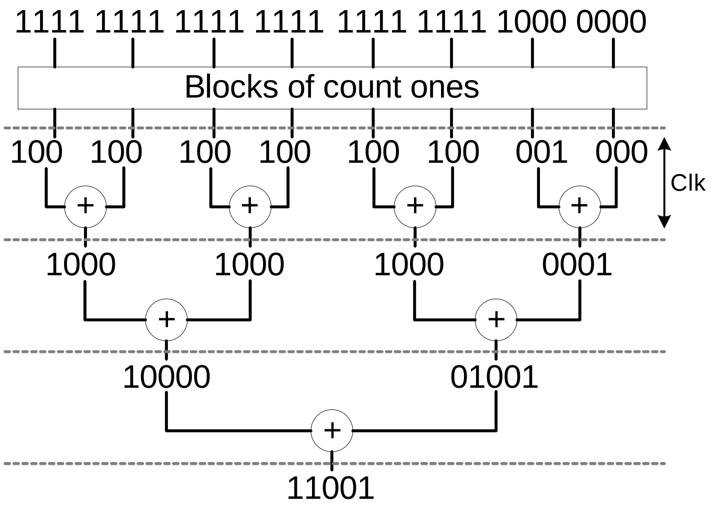


Figure 3.15: An example of the count-ones decoder

3.2.2.1 Performance

To gain some more knowledge about carry-chain decoders and FPGA routing. All possible versions of the decoders were synthesized and benchmarked on maximum frequency and used resources. All these tests were done on the ARTIX-7 speedgrade -2. In Figure 3.16 we can see that with a block size (BS) greater than 8 it becomes very difficult to overcome the barrier of 400 MHz. However in figure 3.17 we see that the small block sizes of 3 and 4 are very inefficient in therms of utilization. The block sizes 6 and 7 however performs well on both performance and utilization. Therefore, we recommend using this setting for creating decoders.

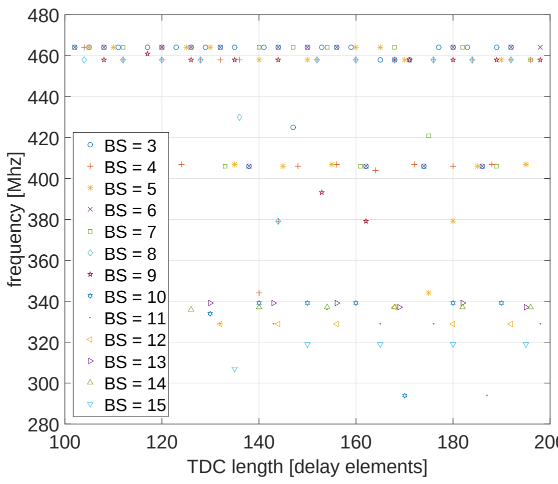


Figure 3.16: Length vs frequency with different block size

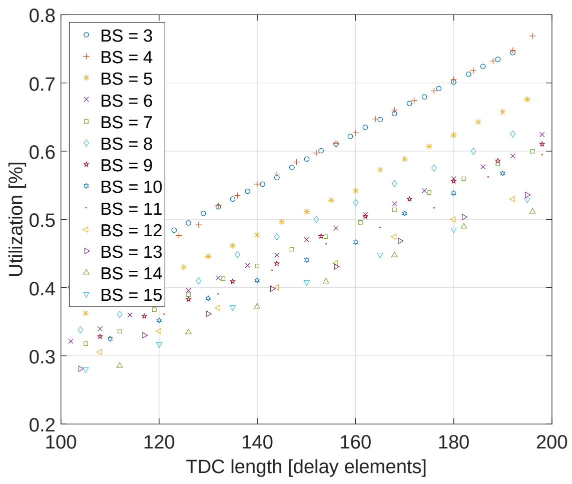


Figure 3.17: Length vs utilization with different block size

TDC len	Block size	Freq (MHz)	Regs	Luts	Utilization
192	3	464	758	371	0.74 %
196	4	458	782	385	0.77 %
195	5	407	707	300	0.68 %
198	6	464	661	262	0.62 %
196	7	458	612	297	0.60 %
192	8	458	620	345	0.63 %
198	9	458	616	316	0.61 %
190	10	339	574	291	0.57 %
198	11	329	575	359	0.60 %
192	12	329	534	275	0.53 %

Table 3.1: Count ones implementations near a length of 200 carry elements

3.2.3 Multiplexer carry-chain decoder

Where the count-ones-decoder processes the data from locally to globally. The Multiplexer implementation starts directly by processing the entire carry-chain. Every clock cycle the carry-chain gets split into 2, keeping only the useful part. The multiplexer decoder can only detect a rising or a falling edge in the carry-chain. This needs to be decided before synthesizing. Figure 3.18 shows an example of the multiplexer decoder that is set to detect a falling edge in the carry-chain.

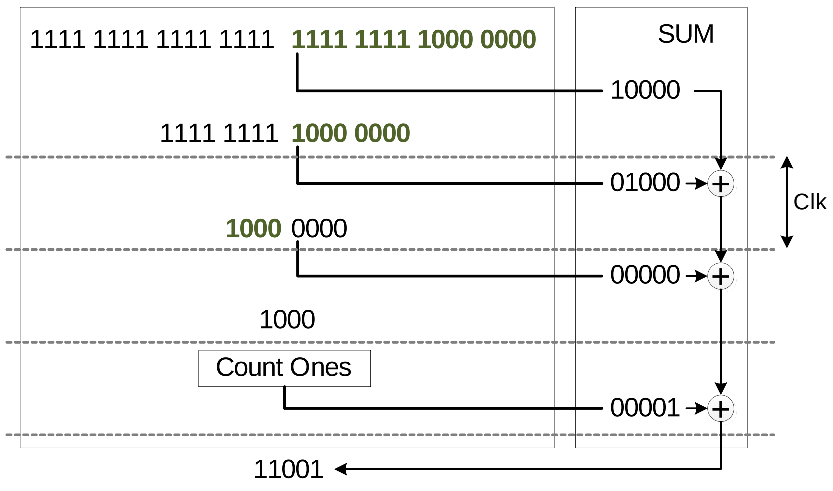


Figure 3.18: An example of the multiplexer based carry-chain decoder

3.2.4 Advanced multiplexer carry-chain

Because the multiplexer decoder from subsection 3.2.3 chooses the split of the entire carry-chain based on only one bit. There can be situations where bubbles created by clock skew can increase the non-linearity when not processed. An example of such a situation can be found in Figure 3.19.

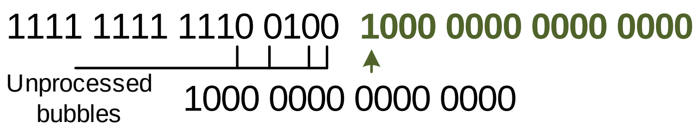


Figure 3.19: An example of unprocessed bubbles by the simple multiplexer implementation

The latency and utilization can be reduced by chopping the multiplexer into multiple pieces. Figure 3.20 shows how a carry-chain can be chopped into 5 pieces. From now on this will be called mux steps.

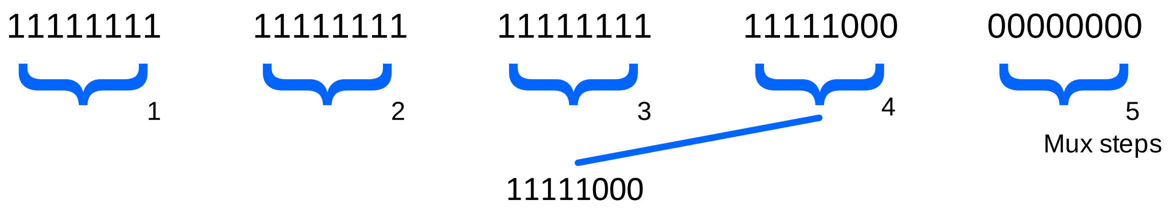


Figure 3.20: A schematic overview of the number of mux steps

As a solution for handling bubbles caused by clock skew, the advanced multiplexer doesn't pick it's useful data by just one bit. A number of bits can be set by the parametric, "filter bits". Figure 3.21 shows an example of a carry-chain with 3 filter bits and 5 mux steps. The output of the filter is determined by majority voting. Afterwards the two filters that could contains bubbles are copied to the result. Note that the outer left and right filters aren't required for this algorithm, they behave as data bits, but are required to have a fixed number of output bits.

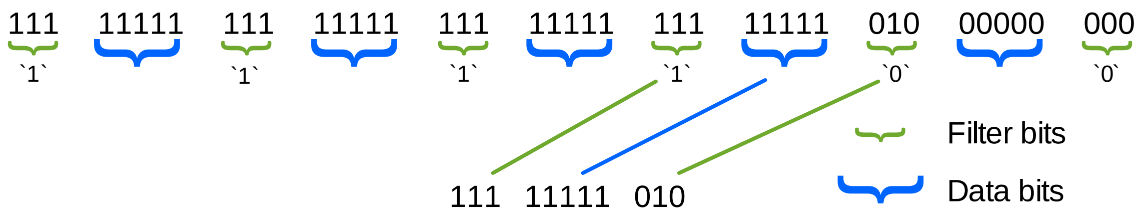
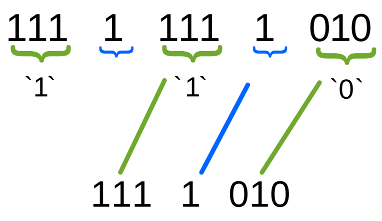
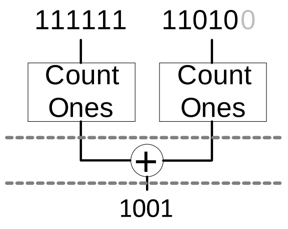


Figure 3.21: A schematic overview of the filter mux implementation

Near the end of the decoding there isn't always enough data to chop the carry-chain in the number of mux steps. When this happens, the maximum number of steps that is still possible, will be executed. Another problem could be that the data bits can't be all equal due to the carry-chain length. When this occurs the length is virtually extended with zeros at the end of the carry-chain. After every decoding step this advanced decoder checks if it's faster to process the remaining data with the count-ones-method or with another multiplexer step and then the count ones method. In Figure 3.22a you can see a multiplexer step on the remaining chain from Figure 3.21 and in Figure 3.22b the count-ones steps. Here we picked 6 bits to count in one clock cycle, from now on we will call this number of bits the block size. Because the remaining 7 bits of the multiplexer step would also take 2 clock cycles to count, it is more efficient to skip this step and directly perform the counting method on the 11 bits.



(a) An example of the next mux step



(b) An example of the counting steps

Figure 3.22: The last steps of decoding

3.2.4.1 Performance

The advanced multiplexer carry-chain decoder has 4 parameters (length, mux steps, filter bits and block size). To vary over all those values would result in an explosion of data. This is prevented by using a fixed carry-chain length of 200 elements. Figure 3.23 shows that the split that the mux steps 2 and 3 are less efficient than the 4 until 8 steps. However, the mux steps 7 and 8 don't reach the 400MHz. Figure 3.24 shows the max frequency and utilization expressed in number of filter bits. Over the entire range from 1 to 11 bits there implementations over the 400 MHz. The only thing that can be observed from the data is that the number of filter bits clearly impacts the utilization. Table 3.2 shows all filter sizes with their most efficient implementation, that also exceeds the 400MHz.

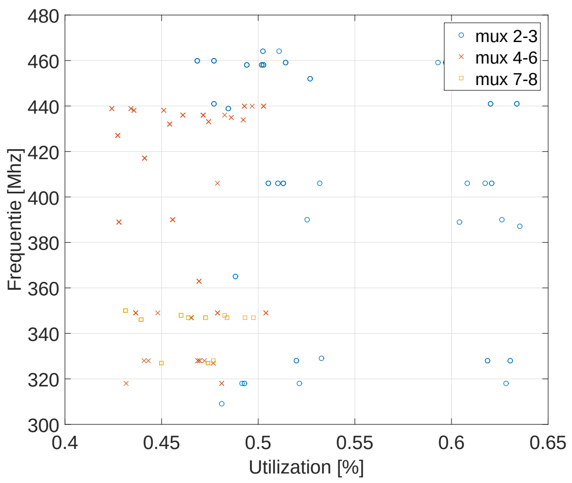


Figure 3.23: Mux steps vs utilization and max frequency

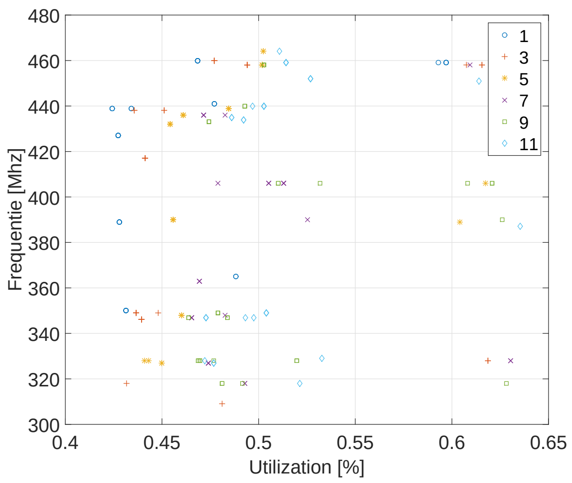


Figure 3.24: Filter bits vs utilization and max frequency

Filter	**Mux steps**	**Block size**	**Freq (MHz)**	**Regs**	**Luts**	**Utilization**
1	5	9	439	476	124	0.42 %
3	6	9	438	474	157	0.44 %
5	6	7	432	491	170	0.45 %
7	5	8	436	513	170	0.47 %
9	6	7	433	506	191	0.47 %
11	6	7	435	515	203	0.49 %

Table 3.2: Multiplexer solutions with a max with different filter sizes and there utilization

3.2.5 Discussion

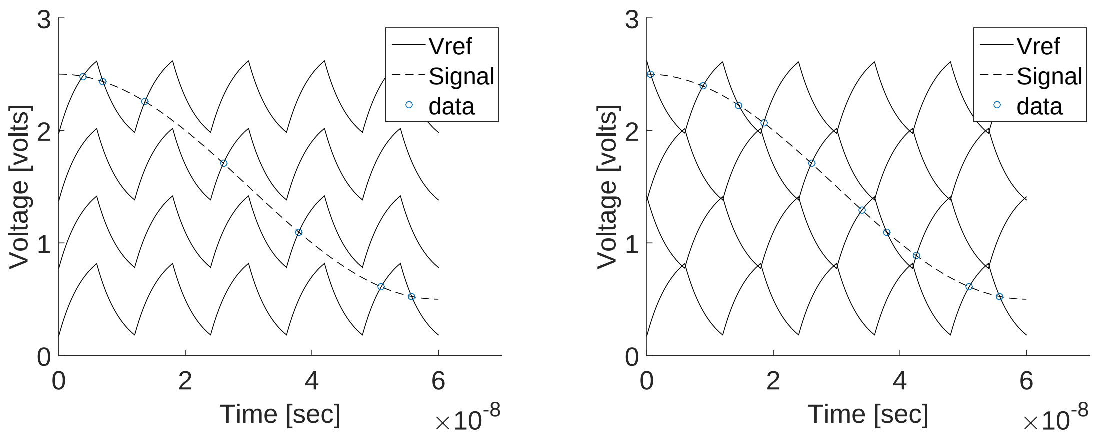
The 2 decoder structures have been implemented, tested and specified. The count-ones-decoder uses clearly more resources with an utilization of 0.60 % compared to the 0.42 % of the unfiltered multiplexer decoder. But to filter out all bubbles, maybe 11 bits filters are required, this increases the utilization of the multiplexer decoder to 0.49 %.
All these implementations were synthesized and simulated. The simulation outputs were even checked for errors in the decoder. But the power consumption is still unclear. They both process data in a different way, so most likely there will be differences in the power consumption.
For now the count-ones is the easiest, because it can detect both edges and filter out any kind of bubbles. But once the count-ones works it is a good idea to test the advanced multiplexer.

3.3 Scaling up conversion speed

As section 3.2 concludes that the maximum sampling rate of the carry-chain is around 400 MSa/s. In this section 2 designs are discussed that could increase the performance by adding resources.

3.3.1 Parallel sampling

Parallel sampling is a method to increase resolution. Parallel sampling makes multiple ADC's sample in different parts of the input range. After every measurement cycle, there should only be one part of the range that had a collision. That part can than translate its result to the global range. An example of parallel sampling can be seen in Figure 3.25a, here multiple RC curves with different offsets are merged. However the signal here can cross between 2 RC curves. To force collisions all curves need to be 180 degrees phase shifted with respect to their adjacent neighbors. This has been done in Figure 3.25b.



(a) Simple parallel sampling (b) Closed parallel sampling

Figure 3.25: Two versions of parallel sampling

3.3.2 Interlaced sampling

Where parallel sampling increases the resolution of the ADC, interlaced sampling is a good way to increase the sampling rate. It works by phase shifting multiple RC curves, so that they are all equally spread over 360 degrees. In Figure 3.26 you see an example of 4 interlaced RC curves. In section 3.2 we established that it is possible to determine the measurement time and voltage. So all outputs of the measurements can be sorted in time and combined to one output stream.

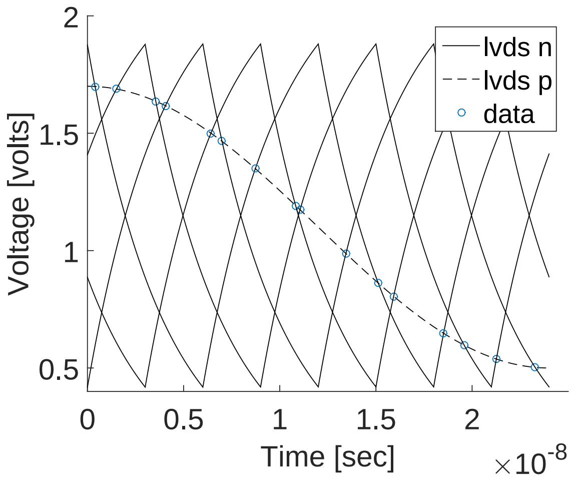


Figure 3.26: The waveforms of interlaced sampling

With all these independent sources that produce measurements, there is an increase of high frequency noise. Because each falling and rising wave has its own calibration.

3.3.3 Discussion

When comparing parallel sampling to interlaced sampling, parallel sampling has less noise because it has only one reference signal for a sub-range in the adc. This means that low input frequencies will be measured with less noise. One of the disadvantages is the limitation of the sampling rate. This is linked to the switching performance of the io pins.
The parallel sampling can only adjust its sampling rate by increasing the frequency of all RC signals. To gain more resolution it only needs more channels. The interlaced sampling however, can drop its frequency on the RC signals for more accuracy and increase the number of channels for a higher sampling rate.
Both implementation can't sample signals that cross the rising or falling part of a reference period more than once. This because the carry-chain decoders can only detect one edge. This means that the maximum input frequency one can measure is equal to the reference frequency. So even if a design would consist out of 100 interlaced phases of 100MHz, it would have a sampling rate of 20GSa/s, but it can't measure a 200MHz sinusoidal.
This problem could partly be solved by combining parallel and interlaced sampling. An example is shown in Figure 3.27. When double the accuracy is required with interlaced sampling. Instead of lowering the reference frequency and doubling the phases, it is possible to add parallel interlaced reference signals. By doing this the theoretical accuracy is doubled without reducing the maximum frequency that can be measured.
Because our goal is a sampling rate of 1 GSa/s, the interlaced sampling is the best and easiest implementable solution. The parallel sampling implementation is hard to realize with only a source of 2.5 Volts and resistors. So for practical reasons the interlaced sampling method is the method of choice in our implementations.

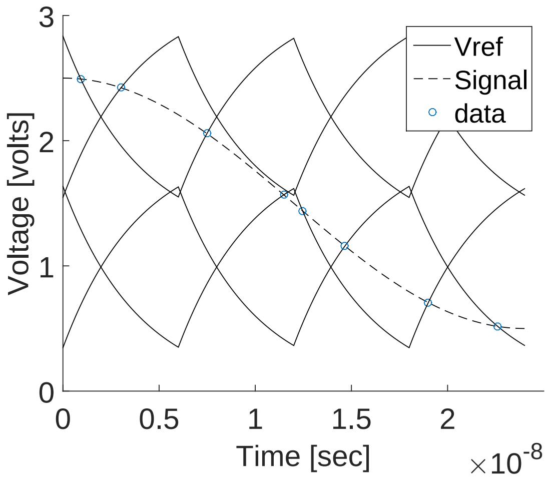


Figure 3.27: Combined parallel and interlaced sampling

3.4 FPGA ADC structures

There are some challenges before the interlaced sampling method can actually be implemented. The developed carry-chain decoders can measure only one edge every sample. To make sure that two edges don't appear in the same clock cycle. An extreme accurate calibration is necessary. This section will show the possibilities for and implementations of such a calibration procedures.

3.4.1 IODelay

The ARTIX-7 has 300 IDELAYE2 components. These delay components can add a configurable delay to a signal. Between 0 and 31 delay tabs can be added to the signal with a resolution of 78, 52 or 39ps [15]. These different resolutions can be created by offering the IDELAYCTRL respectavely 200, 300 or 400MHz clock. Unfortunately Xilinx doesn't provide information about the actual implementation. However the manual says the following:

If the IDELAYE2 or ODELAYE2 primitives are instantiated, the IDELAYCTRL module must also be instantiated. The IDELAYCTRL module continuously calibrates the individual delay taps (IDELAY/ODELAY) in its region (see Figure 2-16, page 126), to reduce the effects of process, voltage, and temperature variations. The IDELAYCTRL module calibrates IDELAY and ODELAY using the user supplied REFCLK.

We can thus deduce that the IDELAYCTRL can adjust the delay of the IDELAY components to match the control frequency. In Figure 3.28 one can see a schematic of an IDELAY component. These IDELAY and ODELAY are mainly used directly before reading or writing an IO pin. However the IDELAY can also be used from within the FPGA. A signal from the FPGA can be routed to an IDELAYE2 component and then routed back to the FPGA. This makes it an excellent addition to our carry-chain implementations.

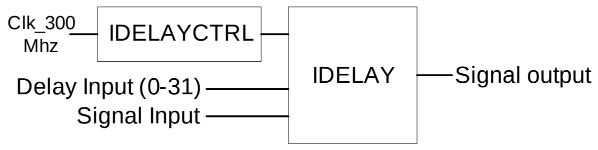


Figure 3.28: A schematic overview of IDELAYCTRL and IDELAY

3.4.2 Logic speed stabilization

Once some measurements had been done with long power supply cables, it became clear that these cables had a negative effect on the accuracy of the ADC. The observation was that fixed input signals created a repeatable distortion (see Figure 3.29). Combine these facts with the logic speed observations of subsection 4.3.2, it is presumable that this distortion was caused by longterm (milliseconds) voltage drop. The electric cables have a resistance and form a circuit as in Figure 3.30. The voltage on those cables is dependent on the current the FPGA uses. When the FPGA uses more current there is a higher voltage on the cables, thus a lower voltage on the FPGA.

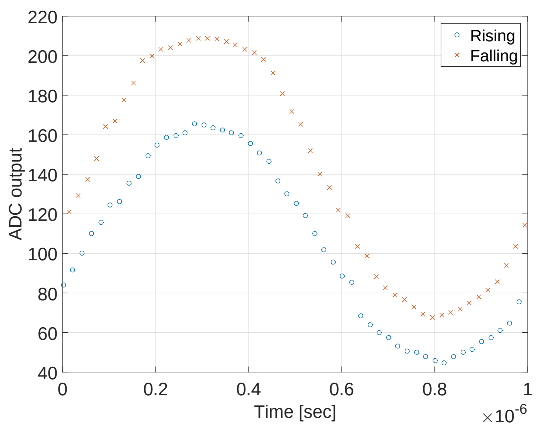


Figure 3.29: The distortion of an 500KHz sinus

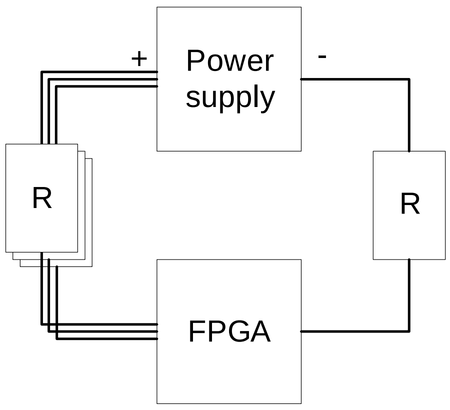


Figure 3.30: Equivalent electric circuit

Ideally we would like to adjust the voltage on the FPGA to create an energy efficient solution. However with the PCB already designed and produced there isn't any other way than adjusting the voltage drop by regulating the power consumption. As far as we could observe, changes the TDC outcome slowly, roughly 50ps after 5 microseconds. This gives us a hints that the decoupling works fine.
To counter the logic speed variations a self measuring clock analyzer was created (see Figure 3.32). Here a 400MHz clock is routed through multiple delay components and than connected to a carry-chain. Where it first goes through 152 carry elements (38 slices) before it reaches a short, 48 carry elements long decoder.
So the clock signal is significantly influenced by the logic speed. When the FPGA is put into a stable condition the delay components can be adjusted to measure the falling clock edge in the middle of the carry-chain decoder. Now with the assumption that the delayed clock is influenced more by delay then the clock, an increase in decoder output means the logic has become faster, and a decrease means that the logic has become slower.

Now to compensate for this behaviour, a fast 6.25 MHz control loop has been designed (see Figure 3.32). This control loop sums 4 results and brings them to an 100MHz clock domain. Then an infinite impulse response filter is applied. Every 160 ns, the output is analyzed to check if it corresponds to a higher or slower logic speed. Given that outcome an array of oscillators is controlled.
In total 512 oscillators are used. All implemented with 6 LUTS as in Figure 3.46. These oscillators were grouped in 4 oscillator farms (see Figure 3.33). Making it possible to enable and disable them in quantities of 4.

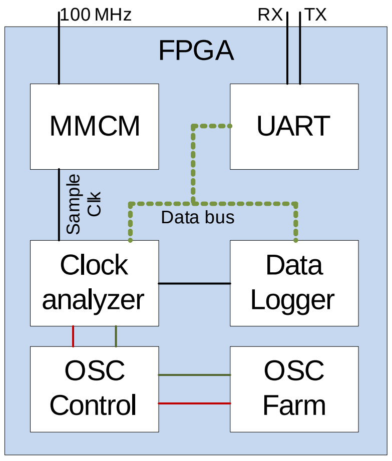


Figure 3.31: FPGA Schematic of a voltage stabilizer

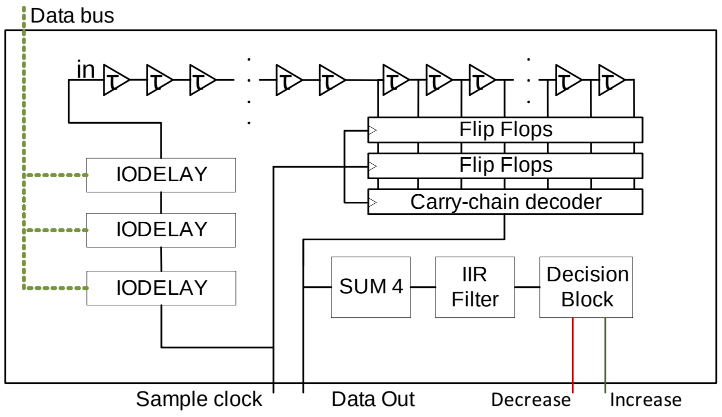
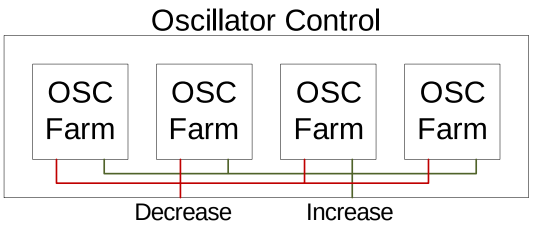
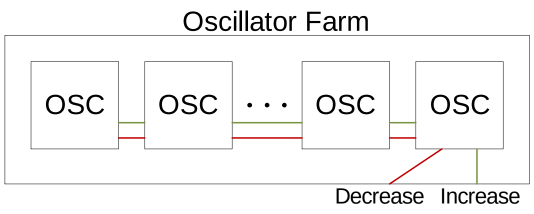


Figure 3.32: Schematic of the Clock analyzer with its control loop



(a) Oscillator Control



(b) Oscillator Farm

Figure 3.33: The two oscillation components

3.4.3 IODelay with Counting

As the goal of this project is to create a reconfigurable ADC, it would be a waste of time to directly implement all prototype algorithms on the FPGA. To save some time only the most essential data processing is done on the FPGA. Figure 3.34 contains a schematic of a 6 interlaced phases sampling ADC. The MMCM creates the reference pulses and the sample clock. The UART receives and sends data to the PC.

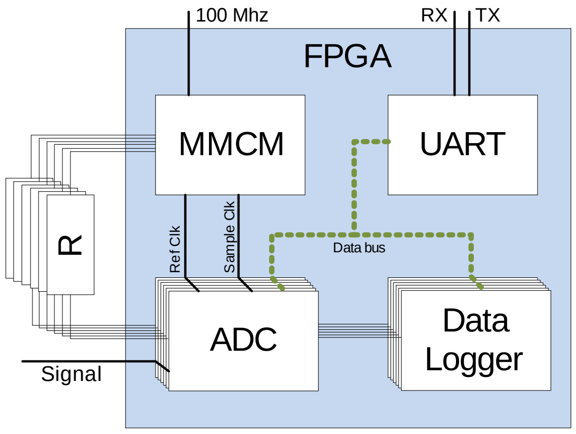


Figure 3.34: Schematic of a GSa/s ADC

Figure 3.35 shows a zoom of the ADC block. Here the data enters through a LVDS port and goes through 3 DELAY components before it reaches a multiplexer. By this configurable delay the input signal can be aligned with the sample clock (this is further explained in section 3.5). In this section is also explained why the ref clocks needs to be connected to the carry-chain.

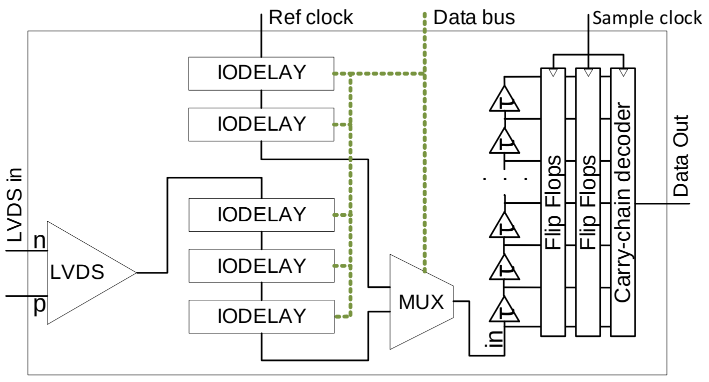


Figure 3.35: ADC implementation count-ones decoder

3.4.4 IODelay with multiplexer read-out

The setup for the multiplexer decoder read-out has an additional XNOR gate compared to the count-ones version (see Figure 3.36). This gate is necessary because the multiplexer carry-chain decoder can only detect rising edges. Other than the XNOR gate there are no differences with Figure 3.35. In Figure 3.37 is an example of the signals. The carry-chain decoder here samples every 100 time units. The falling edges of the reference signal only generates falling edges in the LVDS output. This is why the XNOR is added, by using a XNOR on the reference output all falling edges turn in rising edges.

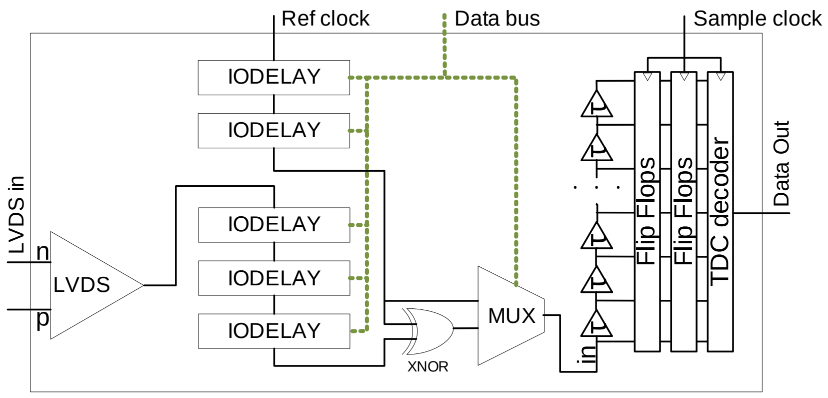


Figure 3.36: ADC implementation mux decoder

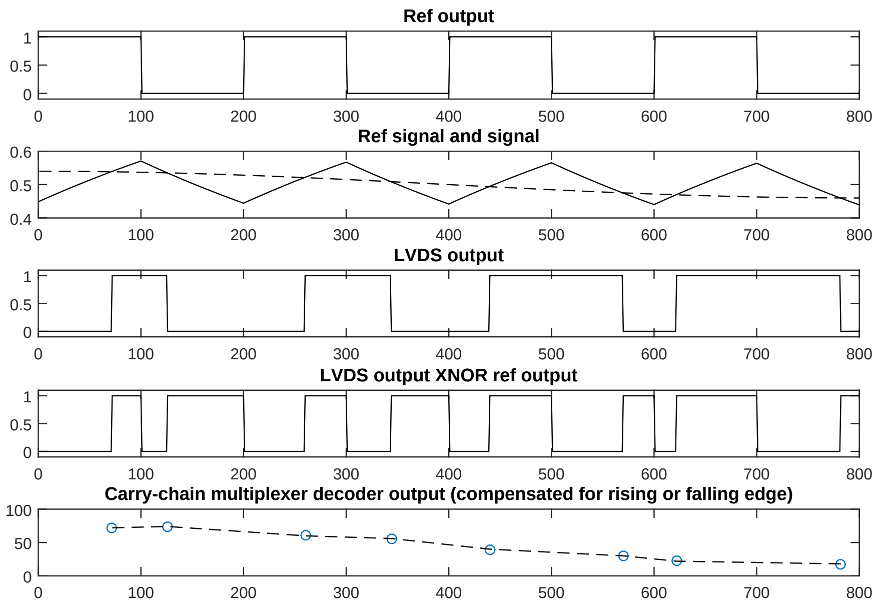


Figure 3.37: Internal signals ADC based on mux decoder

3.4.5 Discussion

While the only difference between the two ADC implementation is a XNOR gate and the carry-chain decoder, the calibration procedures are different. However, the count-ones version is simpler. For the first GSa/s FGPA ADC it is better to first have a clear design. That when errors occur it is easier to pinpoint the error.

3.5 System Calibration

In contrast to ASIC ADCs, that only need a small auto calibration or no calibration at all, our ADC needs a full calibration. By having a full calibration, we can compensate for a wide temperature range and can use components with higher tolerance. The calibration consists of 4 steps and needs a total of 3 different input signals on the ADC. Two sawtooth and one sinusoidal signal, whose frequencies are proportional to the sapling rate, are applied to the ADCs input during this phase.

3.5.1 TDC Length Calibration

The wide temperature range brings significant changes to the timing inside the FPGA. To deal with timing differences the used carry-chain has been made longer than the sampling period, which is 2.5 ns. The implemented carry-chain consists out of 200 delay elements and has a delay of 3.2 ns at room temperature. Figure 3.38 and 3.39 show the problems that are created when the carry-chain has more or less delay than the sampling period. In the case of a line covering more time than the sampling period, the last part of the chain will be a representation of data from the previous period. Data will be double sampled, leading to an overestimation of the correct time. A line that is too short will loose a part of the required measurable range, leading to an underestimation of the correct time in this lost range.
To equalize these values, the TDC has an option to disable elements at the end of the chain. The calibration of the TDC length is done by phase shifting a clock signal through the TDC, as can be seen in Figure 3.40, only if the length of the TDC exactly matches the clock period, the output values will stay the same while shifting a clock through the line. Too long TDCs will have an overestimation of the time in a certain range, too short TDCs will have an underestimation of the time in a certain spot.
By repeating the same measurement with increasingly more TDC blocks disabled, we can find the number of blocks for which the output of the TDC is always the same, i.e. the smallest standard deviation over one complete 360 degrees rotation of the clock. Now the carry-chain covers exactly one clock period, it is possible to virtually increase the carry-chain. This is achieved by adding multiple outputs together. In this way we can reduce the sampling rate and increase the resolution for more accuracy.

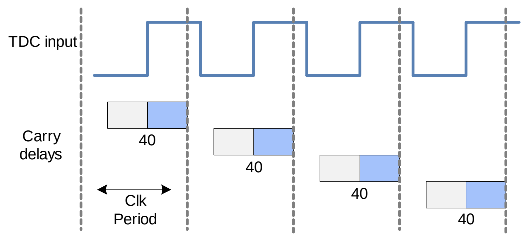


Figure 3.38: Carry-chain too short

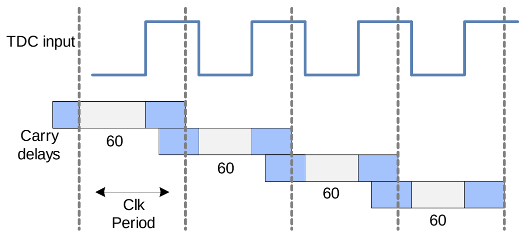


Figure 3.39: Carry-chain too long

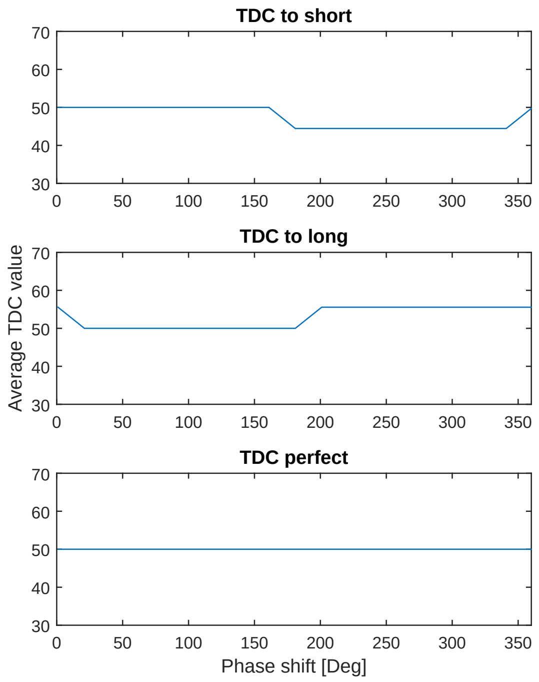


Figure 3.40: Different average TDC values with a too short, too long and a perfect TDC, while rotating the clock input by 360 degrees through the carry-chain. Only in the perfect TDC, the average is constant.

3.5.2 TDC Alignment Calibration

After calibration of the TDC length, the TDC needs to be aligned with the reference period. We have to take into account the fact that one LVDS generates two outcomes in one reference period, one for the rising and one for the falling edge. Hereafter, we will refer to these properties as even, respectively odd, parity.
Figure 3.41 shows on the first line the RC-curve and a DC input signal on the bottom of the range. These two signals compared in the LVDS buffer lead to the result shown below for an unaligned TDC. The measured values are 0 and 20 for the first and second half of the clock period, which is not aligned to the reference period. Consequently one of the parities can't measure the current input voltage. After alignment of TDC clock and reference period, the values are properly measured to be 5 respectively 15. Both parities can measure input voltages over the complete analog range.
By applying a slow ramp that exceeds the minimum and maximum input voltage, we can find the best alignment of the TDC, i.e. when the TDC parities have a maximum span. At a certain time after starting the ramp, the TDC values of the parities go from 0 to 1, indicating the start of the TDC span. The span ends as soon as the parities go from max-1 to max. By calculating the time between these events for both parities we can calculate the span. When doing this span measurement for all possible alignments we can find the alignment where both parities have the biggest span. That is the configuration in which the alignment is optimal.

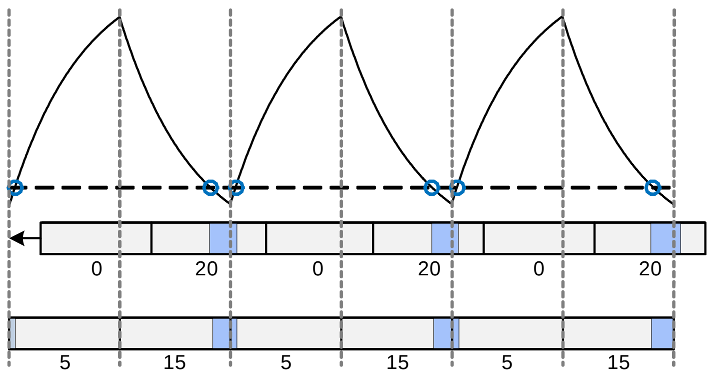


Figure 3.41: A timing diagram of TDC alignment process. The measurement period of the TDC doesn't match with the reference period. The TDC is shifted to match both periods to one another.

3.5.3 TDC to ADC Calibration

After completing the calibration of the TDC, the voltage characteristics of the ADC is calibrated. The transfer of input voltage to digital output is measured by applying a sawtooth on the input of the ADC. As there is a direct relation between input voltage and time after starting the sawtooth, the sample time can be converted to a voltage. For every LVDS input this calibration generates two look-up tables that convert TDC values to ADC values, one for the falling and one for the rising RC-curve. An actual conversion graph can be seen in Figure 3.42. Rising and falling curves are monotonic. This can be explained by the duty cycle measurement. As shown in Figure 3.43 the outcome of the TDCs is positively correlated to time of measurement for the rising curve and negatively correlated to the falling curve.

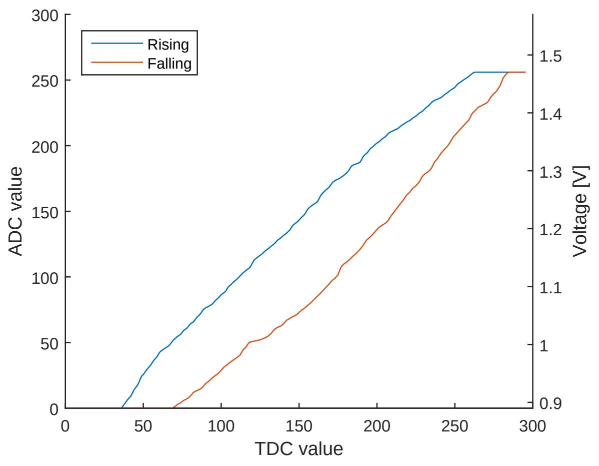


Figure 3.42: The conversion graph from TDC value to ADC value for the rising and falling edge of the RC-curve. The ramps are monotonically rising and in the calibration corrected for their RC distortion.

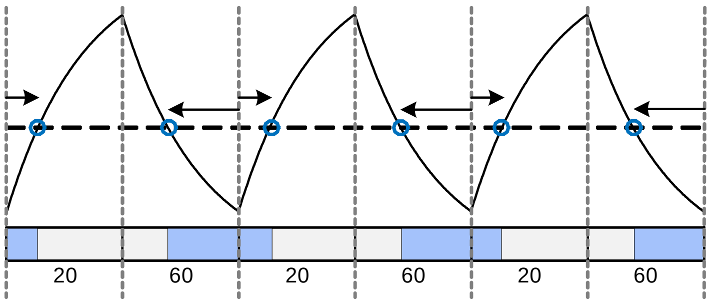


Figure 3.43: The timing diagram of the TDCs. The principle of counting ones in the carry-chain decoder makes the generated time stamps to appear for the rising edge to be between the start of the reference period and the crossing point of the signal and ramp. For the falling edge, the calculated time is between the falling edge crossing point and the end of the reference period.

3.5.4 ADC Synchronization

The entire design of this ADC depends on the accuracy of the ADC synchronization. After all, the results of the 6 interleaved channels can only be combined if the channels are accurately synchronized to one another. With the previous calibration steps, each LVDS input behaves as a standalone ADC, generating two values in one period of the reference signal. To properly combine the values of six different channels, the time difference of each of the six phases has to be known.
To the inputs of all six channels, a sinusoid with a period covering 32 reference periods is applied. All ADCs measure this signal and a least-square-fit is done on the result. This least-square-fit produces a phase of the measured signal. This phase can be converted to a time and than every ADC calculates his offset from the mean off all phases. By applying all these offsets the ADCs produce ADC values with timestamps that can be sorted and then produce an output seen in figure 3.44.

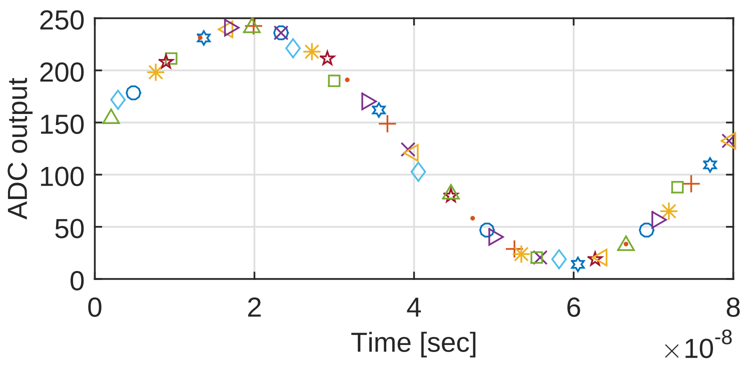


Figure 3.44: The timestamps of 12 different sources, every source has its own marker

3.6 Cryogenic Design

Before creating an ADC at 4 Kelvin, it was important to test the basic parts of the FPGA independently, so to pinpoint potential problems during the cooling procedure. For the first test we wanted to see if a programmed and functioning FPGA keeps operating when cooled down. This can be done by reading an io pin and writing the input (asynchronously) to an output pin. If this operation is succesful we can try to reprogram the FPGA at 4 Kelvin.

3.6.1 Measurement Setup

The setup consists out of a tank filled with liquid helium as can be seen in Figure 3.45. On top of this tank a probe can be inserted. The probe is moved vertically to adjust the FPGA temperature. Through the probe run 16 cables that are connected to a pin header. Additional cables, such as power cables or cables with high frequency connectors, can be added as well.

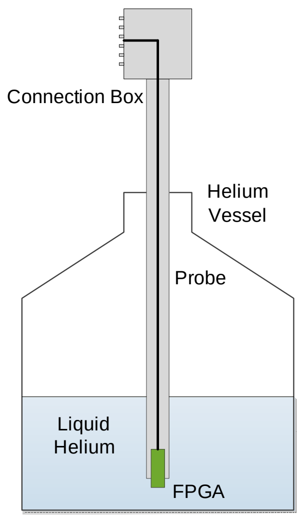


Figure 3.45: The helium setup

3.6.2 Oscillators

A first test to obtain a speed indication is an oscillator test. This is a friendly test to check the speed of the logic. In Figure 3.46 one can see the schematic of an FPGA implemented oscillator, the inverter and the multiplexer are both implemented with 1 LUT5. The buffer to create the delay of the oscillator is implemented in 3 ways: with LUTS, carry-elements and IDELAY components.

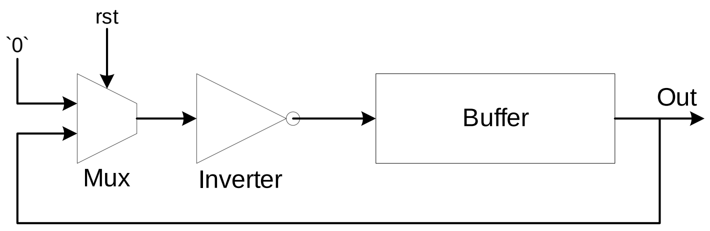


Figure 3.46: The schematic of an oscillator

3.6.3 Serial and block ram

Two important components for testing the TDC and ADC are serial communication and block ram. The communication is necessary to get the data to the PC and the block ram modules can be used to store the massive amount of data created in a short measurement. To test these two we first test the serial communication by implementing an echo program. Once this program works the echo is extended by adding a FIFO as buffer between receiving and sending data. The schematic of this test can be found in Figure 3.47.

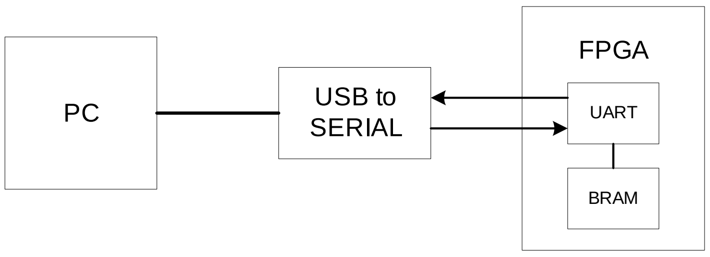


Figure 3.47: The schematic of the block ram test

Results 4

The measurements that can be done on a reconfigurable device are endless. The designed ADC has 4 adjustable variables: temperature, sample rate, range and resources. Therefore a set of configurations has been tested at room temperature. Afterwards we did measurements of a 1.2GSa/s ADC on multiple identical PCBs. To test our FPGA at deep cryogenic temperatures we first checked the basic functions. Afterwards we tested the 1.2 GSa/s ADC in liquid helium. For getting more knowledge about logic operating speed we tested an oscillator at multiple voltages and temperatures.

4.1 The PCB

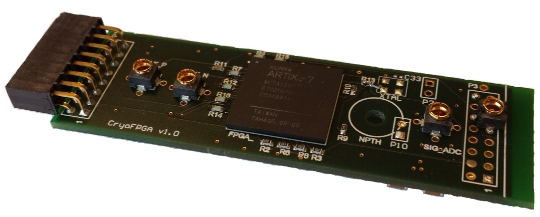


Figure 4.1: PCB front

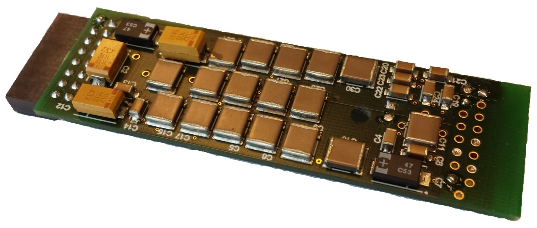


Figure 4.2: PCB back

The used PCB for the FPGA was extremely minimalistic. Every component that could be placed more than a meter away from the FPGA isn't on the PCB. This reduced the chance of failing parts at 4 degrees Kelvin. The only parts that remained on the PCB are the decoupling caps, some resistors, and the FPGA itself. The capacitors are all picked by their potential to keep operating at 4 Kelvin, making this a truly unique board. Thanks to the minimalistic design this device can't work alone. The FPGA needs 3 power supplies with voltages of 1.0V, 1.8V and 2.5V, it requires a differential clock and it needs to be programmed with the JTAG pins. The used equipment is mentioned in section 4.2.
The PCB was created by Bishnu Patra as an internship for his master. There were some design faults in the PCB. In total 2 were found, the differential clock wasn't connected to a differential input and the polarized capacitors were all reverse mounted. Both these faults could be fixed with some simple modifications. Figure 4.1 shows that an additional MMCX connector is soldered on the spare 16 pins bank. By adding this connector we could put the clock on a differential buffer.

4.2 Equipment

For the 3 voltages, 2 TTi EL302RT Triple power supplies were used. The differential clock was provided with a SP605 (spartan 6) development board and we could program the FPGA with the digilent USB-JTAG programmer. The scope we used to measure jitter on the clocks and read the output of the signal generator is a LeCroy WavePro 700Zi-A scope. This together with LeCroy ZS2500 probes. Although the differential clock pin is connected to a non dedicated clock buffer, when we copied the clock inside the FPGA and then connected it to the oscilloscope, we measured a jitter of only 8 ps. The function generator that was used is the Rohde & Schwarz HMF2550 function generator. This device was connected to the PC with a RS232 connection, to create a closed loop between PC, FPGA and function generator. Calibration steps could be automated to prevent user errors and to decrease measurement- and development time. The function generator works with a 250MSa/s, 14 bits resolution DAC, making it difficult to create a clean sinusoids at high frequencies. Leading to a harmonics suppression greater than -40 dBc for signals greater then 10 MHz, which means that if we measure this signal with a perfect ADC it will appear to have only 6 bits of ENOB.

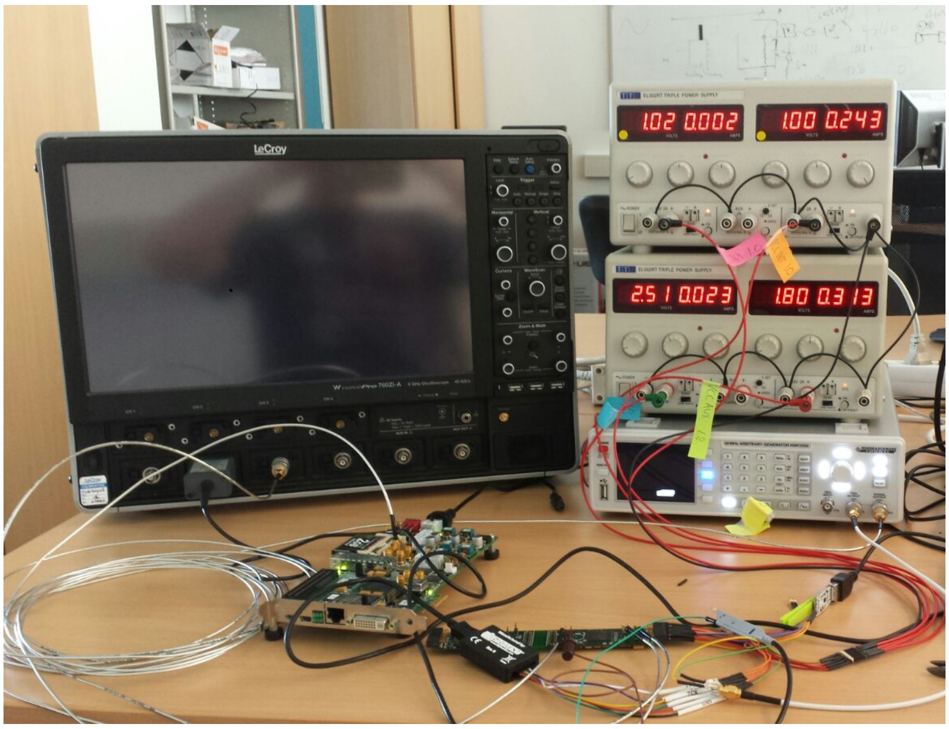


Figure 4.3: Measurement setup

4.3 Room temperature results

4.3.1 Capacitance measurement

To measure a capacitance of about 10 pF is difficult. But we wanted to have an estimation of the LVDS input capacitance. The designed measurement method is to charge and discharge the capacitance with an FPGA pin through a 500Ω resistor. The charging signal and the LVDS input voltage were measured with an active probe. This measurement data was saved on the oscilloscope and afterwards processed. A simulated LVDS input voltage could be calculated from the measured charging signal. Here all possible capacitance values could be simulated, and the best fit could be determined. This best fit turned out to be at 10 pF, in Figure 4.4, the simulation of the 10 pF capacitance is shown together with the measured LVDS input. The active probe has a capacitance of 0.9 pF, this could be subtracted from the measured value, Therefore the LVDS input capacitance is estimated to be 9 pF.

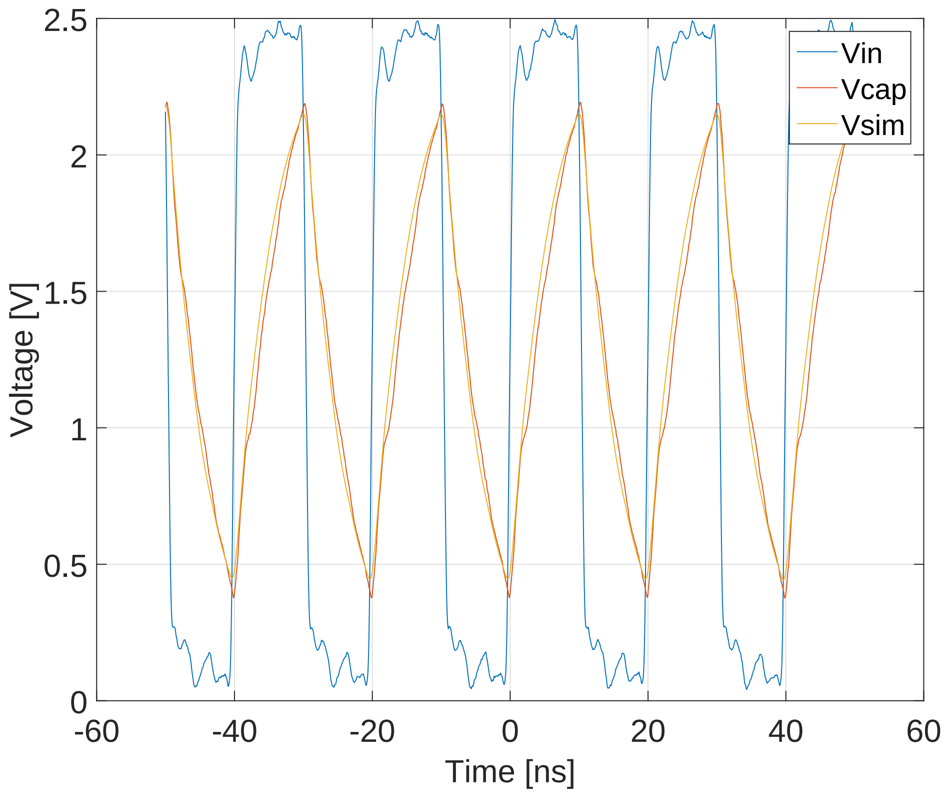


Figure 4.4: LVDS capacitance measurement ans simulation result

4.3.2 Logic speed vs power supply

The carry-chain is a promising tool to create high performance sensors. However the carry-chain isn't an official component of the FPGA. Xilinx doesn't optimize the carry-chain for stability but for power consumption and speed. To get an idea of the correlation between the speed of the logic and the voltages of the VCCINT and VCCAUX. Multiple measurements have been done, 2 for the carry chain resolution and 4 with an IODELAY based oscillator. All these measurements have been executed on 6 implementations on different locations inside the FPGA. The measurements on the carry-chain resolution can be found in Figure 4.5 and 4.6. Here can be seen that over the supported VCCINT from 950mV to 1050mV, the resolution changes with 23 percent! This means that the ADC is also very sensitive to these changes.

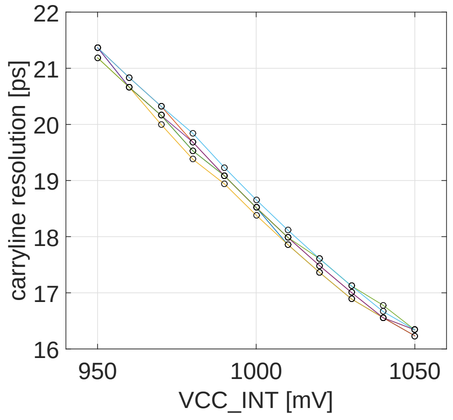


Figure 4.5: Carry resolution vs VCCINT (VCCAUX = 1.8V)

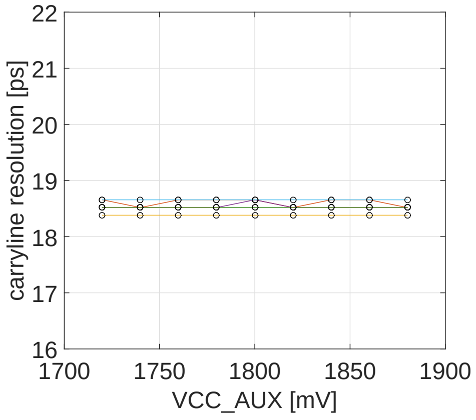


Figure 4.6: Carry resolution vs VCCAUX (VCCINT = 1.0V)

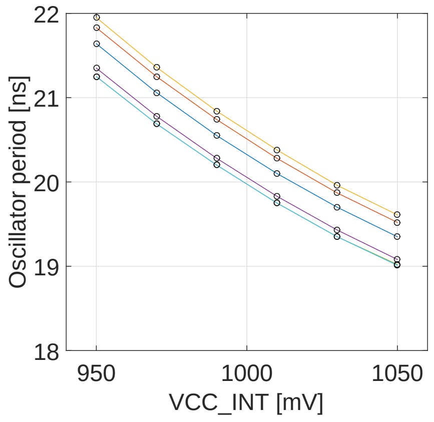


Figure 4.7: Oscillator period with 77 delay tabs vs VCCINT (VCCAUX = 1.8V)

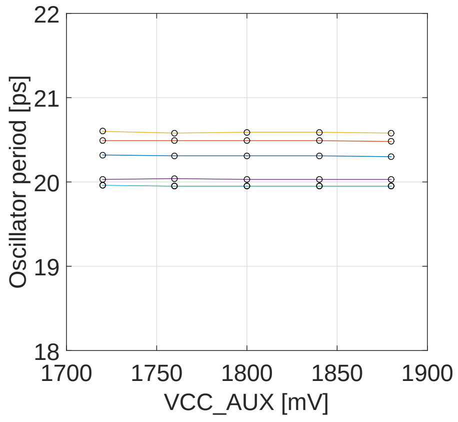


Figure 4.8: Oscillator period with 77 delay tabs vs VCCAUX (VCCINT = 1.0V)

Because the IODELAY is supposed to be continuously calibrated as discussed in subsection 3.4.1. An oscillator was made with this delay component. Afterwards it was measured over the VCCINT and VCCAUX range with 2 delay configurations. The results of these oscillator measurements can be found in Figure 4.7, 4.8, 4.9 and 4.10. The VCCAUX seems to have no influence on the logic speed but the VCCINT has a significant influence on the oscillator period. However it is still possible that the logic and the interconnects are influenced by the VCCINT but the delay component not. To prove that the delay component is stable over the VCCINT range, the oscillator periods from the 2 delay configurations where subtracted from each other. The result of this can be found in Figure 4.11. Here you can observer that the delay component stays stable over the entire range.

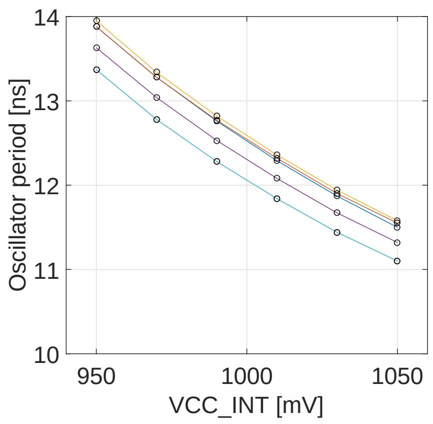


Figure 4.9: Oscillator period without delay vs VCCINT (VCCAUX = 1.8V)

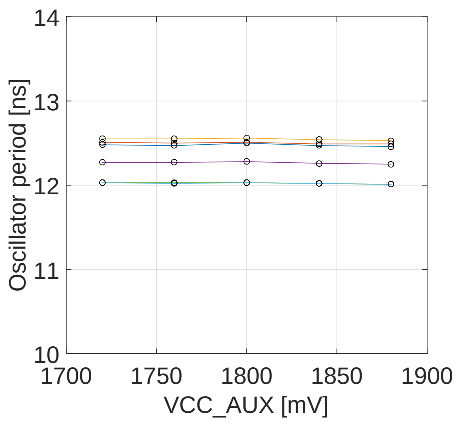


Figure 4.10: Oscillator period without delay vs VCCAUX (VCCINT = 1.0V)

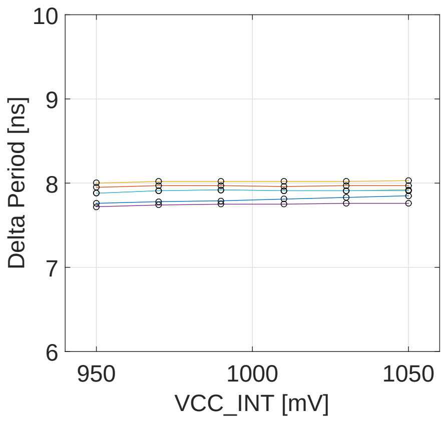


Figure 4.11: Period difference between 77 tabs and 0 tabs delay vs VCCINT (VCCAUX = 1.8V)

4.3.3 1.2 GSa/s ADC Characterization

While the goal was 1 GSa/s we made a 1.2 GSa/s because of the chosen frequencies inside the FPGA. As far as we could find, this has never been done before thus we believe that this fully programmable FPGA based ADC is the state-of-the-art.

4.3.3.1 INL, DNL and single shot

The INL and DNL of this ADC are determined by 12 sources, each source has its own calibration. To obtain these graphs a slow ramp was applied to the ADC. By sampling this ramp multiple times the INL could be determined by calculating the difference between the sample mean and the best fitted line. In total 5.5 million samples were used to create these graphs. The data was averaged over 1 LSB to determine the non linearity. The DNL was then calculated by counting the number of occurrences of an output and then dividing it by the mean.

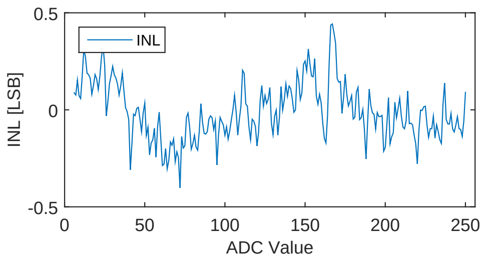


Figure 4.12: The INL of the 1.2 GSa/s ADC

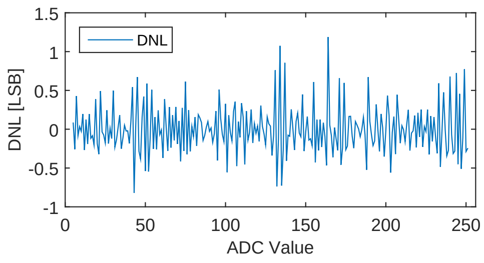


Figure 4.13: The DNL of the 1.2 GSa/s ADC

To get an estimation of the noise as a function of the ADC value. The single shot accuracy is also determined with the ramp data. By calculating the mean of the variance between the mean and the individual values for each bin. It was possible to sketch a good impression of the accuracy over the entire range. Figure 4.14 shows the single shot accuracy.

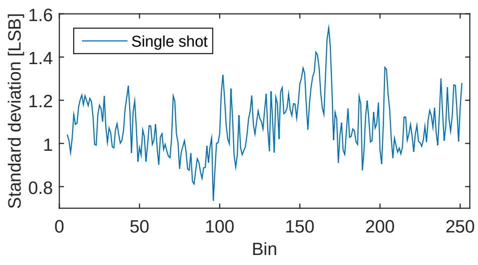


Figure 4.14: The single shot accuracy over the output range

4.3.3.2 Spectrum and samples

To rate the performance in effective number of bits (ENOB), a sinusoid was inputted to the ADC. By analyzing the output the signal-to-noise and distortion ratio SINAD and the total harmonic distortion THD could be determined. For this kind of measurements the function generator has to outperform the ADC. But for our application we could not access such a device. The used signal generator could only suppress it harmonics by > 37 dB for signals > 25MHz. In Figure 4.15 and 4.16 two examples of digitized sinusoidal signals are shown: one at 1 and one at 25 MHz input frequency. For reference, a best fit sinewave is drawn through the measurement points. Comparing a 1 and 25 MHz input, sampled at 1.2 GSa/s, we observe slightly higher noise at 25 MHz.

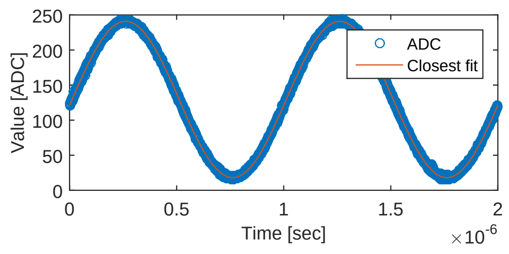


Figure 4.15: 1 MHz sinus signal

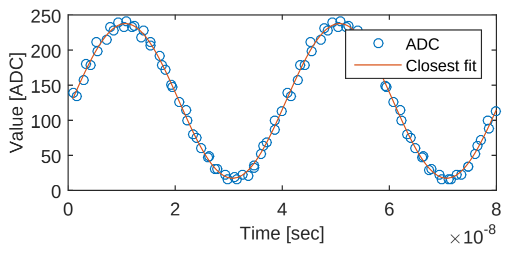


Figure 4.16: 25 MHz sinus signal

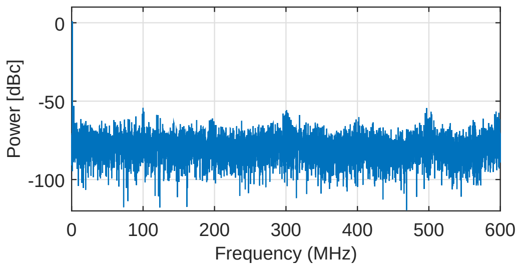


Figure 4.17: Spectrum of a 1 MHz sinus

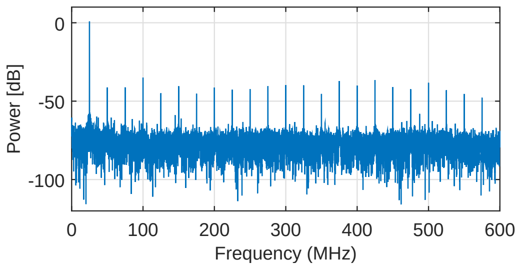


Figure 4.18: Spectrum of a 25 MHz sinus

Transferring the sampled sinusoid to the Fourier domain with an FFT, leads to the frequency domain plots of Figure 4.17 and 4.18. For these spectra 48000 samples were used. Again the 1 and 25 MHz signals are taken as an example. The SINAD is defined as the power ratio of the signal to the sum of the remainder of the spectrum. The SINAD was found to be 37 dB, respectively 25 dB. Furthermore the THD, defined as the ratio of the sum of all signal harmonics by the power of the signal frequency itself, was found to be -52 dB, respectively -34 dB. The noise floor in contrast was roughly -60 dB below the main signals power, showing that the harmonic distortion is a major contributor to the lower SINAD especially at higher input frequencies.

From the SINAD, the ENOB is calculated with Equation 4.1 to be 6 respectively 4 bits. Again it has to be noted that the performance is limited for a large part by the function generators harmonics and noise.

$$ENOB = \frac{SINAD - 1.76}{6.02} \tag{4.1}$$

Finally the effective number of bits ENOB over frequency is shown in Figure 4.19. The expected behaviour for the SINAD is to drop over the increasing input frequencies and gives an indication of the effective resolution bandwidth ERBW of our ADC. The ERBW is the input frequency at which the SINAD drops by 3 dB or the ENOB by 0.5 LSB. The ERBW is in the order of 5 MHz.

4.3.4 Range vs ENOB

As already discussed in subsection 3.1.1, the RC curve can influence the noise and range. To test the impact of the RC distortion and static noise, 3 different resistor sizes were soldered on the PCB. Which resulted in a different range and ENOB, see Table 4.1. Surprisingly all the results are close. What indicates that the static noise and RC distortion have about the same impact on the system. It could be that with a better function generator the smallest range would perform better. For an arbitrary function generator it is hard to create a high resolution sinusoidal with a range between 1.12 and 1.45 volts.

Resistors (Ohm)	**Range low (V)**	**Range high (V)**	**Range total (V)**	**ENOB**
200	0.47	2.02	1.55	6.1
500	0.88	1.66	0.78	6.3
1000	1.12	1.45	0.33	6.0

Table 4.1: Results of the 1.2GSa/s ADCs with different range

4.3.5 Cross device performance

As the application is so close to the maximum capability of an FPGA and PCB, it is likely that there are changes in performance from board to board. We had 3 boards at our disposal and tested the 1.2 GSa/s ADC on all the boards. The results are depicted in Figure 4.19, the performance of the ADCs on all three boards is very similar thanks to the advanced calibration of our ADC. The calibration can smooth out the performance over different devices making this ADC usable as a soft-core of which the performance can be well regulated.

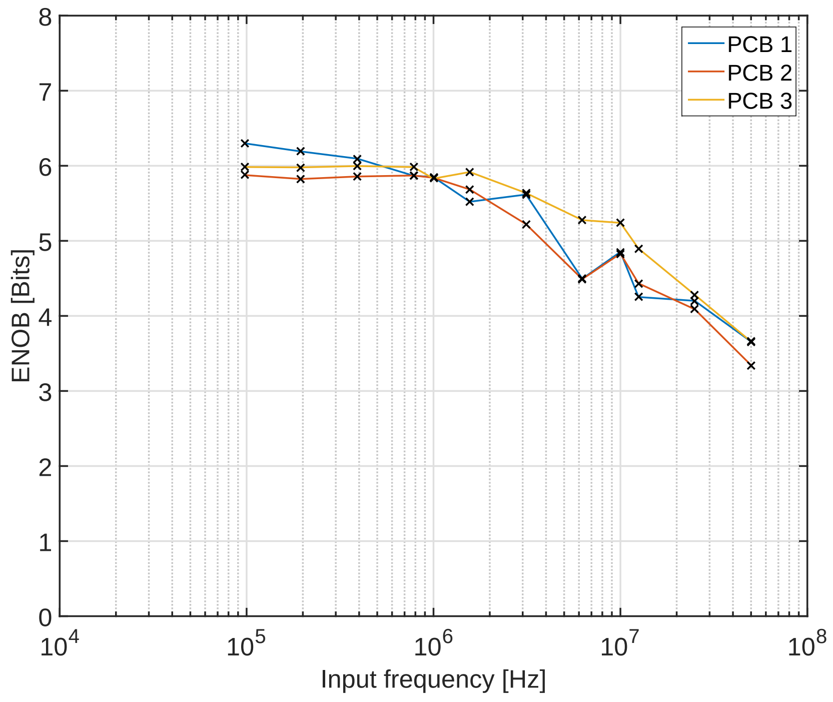


Figure 4.19: The ENOB over input frequency tested on 3 identical boards. The ENOB rolls of at roughly 5 MHz, which is the ERBW of the system (a drop in ENOB of 0.5 LSB compared to the maximum).

4.3.6 Performance over multiple configurations

An interesting aspect of this ADC is the possibility to switch between a number of configurations. The only requirements are a small firmware change and, for optimal range, a change in reference resistor for the creation of the RC-ramp. The performance of ADCs, capable of sampling from 12.5 MSa/s up to 2.4 GSa/s, is shown in Figure 4.20. For a single channel, being only the rising or falling edge of one LVDS comparator, we can reach a performance of over 9 bits (ENOB) with a sampling rate of 12.5 MSa/s. The highest possible sampling rate is achieved with a reference period of 200 MHz on the 6 phase interleaved channels, leading to an impressive 2.4 GSa/s on an FPGA. With each pin added to the ADC, the performance can be improved, however the performance doesn't scale linearly with the number of pins added to the system. The performance at 2.4 GSa/s is slightly less than 4 bits, especially due to distortion and also due to interference between the different RCs of 200 MHz.
With our ADC we can achieve a wide variety of both sampling rates, a factor of 192 between highest and lowest, and effective resolution, a factor of 5 bits difference. This makes the ADC useful in many applications, whatever fits in the range of possible configurations.

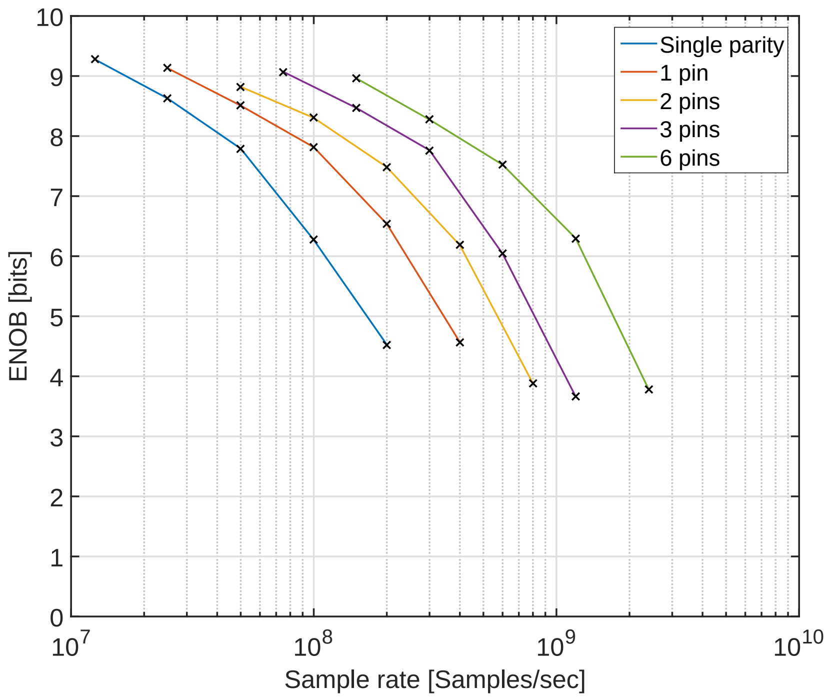


Figure 4.20: The resources vs sample rate vs enob

4.3.7 Logic speed stabilizer

The logic speed stabilizer from subsection 3.4.2 is tested by offering multiple VCCINT voltages. The data is measured at multiple oscillator enable percentages. Table 4.2 the results of the measurements. The VCCINT current shows a 30 mA increase for every 10 percent of the oscillators. This increase of current increases the voltage drop as well, by that increase the VCCINT voltage on the PCB is kept stable, thus we can conclude that the voltage drop can be controlled by steering the FPGA's power consumption. The VCCO current increases because an FPGA pin was used to output the enable percentage of the oscillators.

Oscillators enable(%)	**20**	**30**	**40**	**50**	**60**	**70**	**80**
VCCINT supply(mV)	1037.0	1040.0	1043.5	1046.8	1050.2	1053.9	1057.4
VCCINT PCB(mV)	999.1	999.3	999.5	999.7	999.9	1000.0	1000.0
VCCINT cable(mV)	28.5	31	33.5	36.3	38.7	41.8	44.6
GND cable(mV)	9.3	9.8	10.4	11	11.5	12.2	12.8
VCCINT(mA)	309	336	364	392	419	451	481
VCCAUX(mA)	222	222	222	222	222	222	222
VCCO(mA)	26	29	33	36	40	44	47

Table 4.2: Results of the logic speed stabilizer

To put the logic speed stabilizer to test, the FPGA was connected with 3 meter long power cables, and programmed with the 1.2GSa/s FPGA ADC that included the logic speed stabilizer. Without the logic speed stabilizer, the ADC failed to calibrate. With this stabilizer the ADC could be successfully calibrated. Afterwards a 10MHz sinusoid was measured that is shown in Figure 4.21.

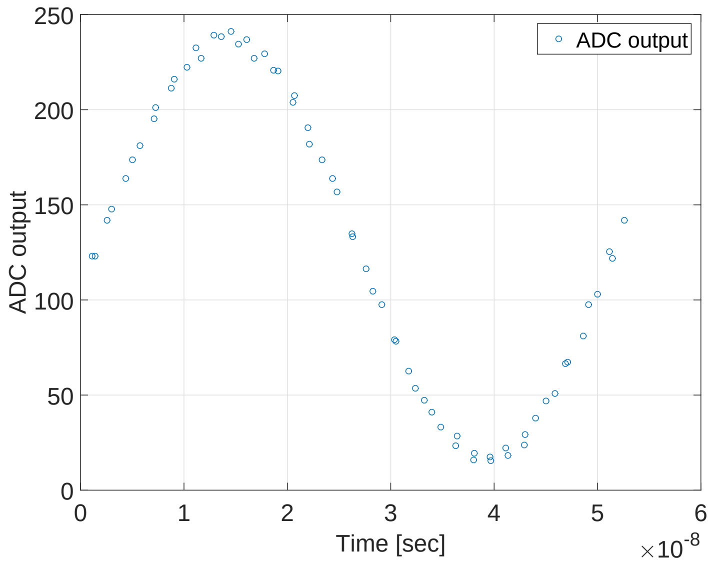


Figure 4.21: A 10MHz sinusoid measured with the 1.2GSa/s ADC, while the FPGA was powered with 3 meter long cables and the logic speed stabilizer was enabled

4.4 Cryogenic Testing

This section outlines the results of the measurement campaign in Milan. Here we could use a probe to cool the FPGA in a helium vessel, (as can be read in section 3.6). These measurements were done before we discovered that all polarized capacitors were mounted with reverse polarity, and before the logic speed stabilizer was designed.

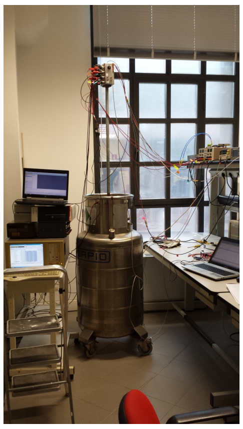


Figure 4.22: The measurement setup in Milan

4.4.1 Basic functionality

From our tests we could conclude that the FPGA is fully functional at 4 Kelvin. We successfully programmed the FPGA at 4 Kelvin, then tested the serial connection at 650 kb/s. This test was extended by buffering the serial communication in a BRAM implemented FIFO. This was done 10 times with 8191 Bytes and all tests were successful. Then the IDELAYE2 components were tested followed by the MMCM and PLL. Table 4.3 shows a summary of all tested components and if they worked at 4 Kelvin. The FPGA had been tested on two different days and spend 2 times, 5 hours in the liquid helium.

Component/functionality	**Works at 4 Kelvin**
PROGRAMMING	Yes
MMCM	Yes
PLL	Yes
LUTS	Yes
BRAM	Yes
LVDS INPUT	Yes
IO PINS	Yes
IDELAYE2	Yes
400 MHz TDC decoder	Yes
1.2 GSa/s ADC	Half, see subsection 4.4.5

Table 4.3: The parts that were tested at 4 Kelvin

4.4.2 Power consumption

The power consumption was measured at multiple temperatures for different programs. To our surprise the power consumption increased when lowering the temperature. The VCCAUX had the biggest increase of power consumption. This might have a relation to the reverse mounted polarized capacitors. Another phenomena, experienced at both room temperature and 4 kelvin, was the VCCO shorting after the supply went past the 2.8 Volts. This problem was gone after the capacitors were mounted correct. Figure 4.23 shows the idle power consumption, Figure 4.24a shows the MMCM power consumption and Figure 4.24b shows the PLL power consumption.

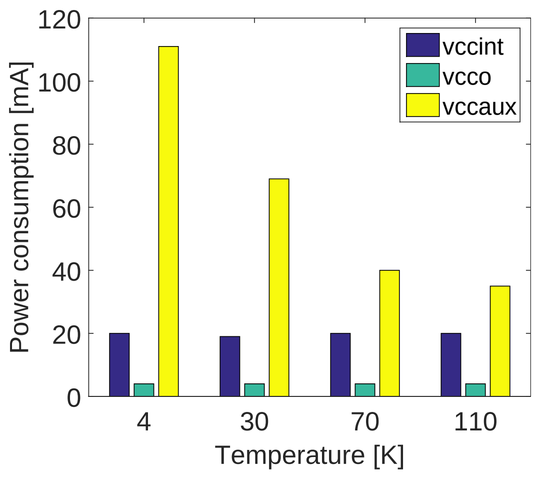
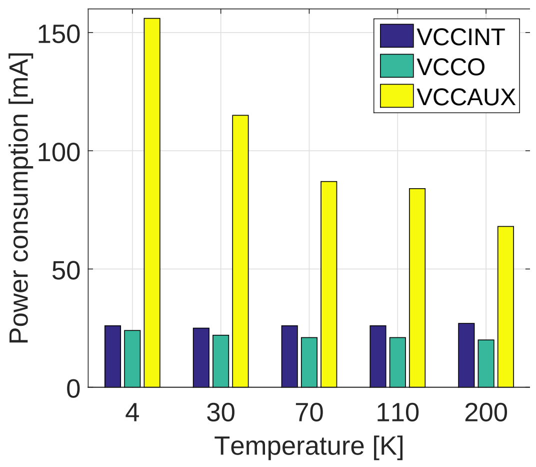
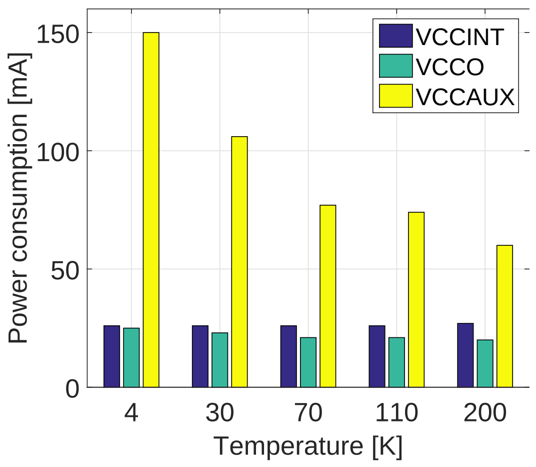


Figure 4.23: The idle power consumption of an unprogrammed FPGA



(a) Power consumption of a MMCM 100 to 50MHz

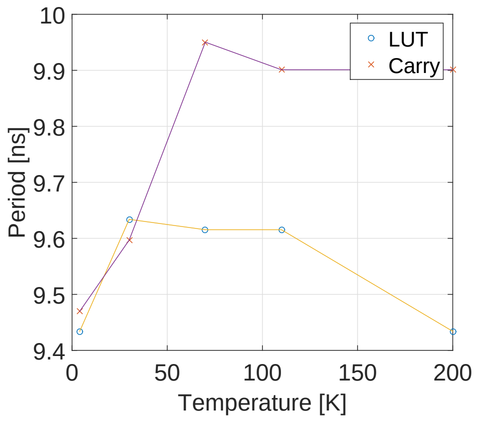


(b) Power consumption of a PLL 100 to 50MHz

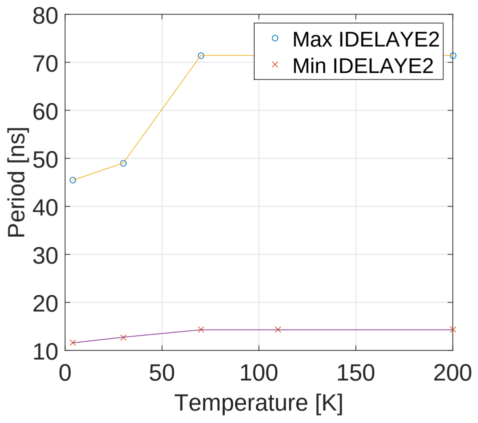
Figure 4.24: The last steps of decoding

4.4.3 Logic speed

To get an estimation of the logic and interconnect speed, multiple oscillators were created. Oscillators were used because they output an frequency that is directly related to the speed of their components. By measuring that frequency, even small changes in logic speed can be detected. Figure 4.25a shows two oscillators, one created with carry-elements and one with LUTS. Figure 4.25b shows an oscillator designed with IDELAYE2 blocks. The temperature has little influence at the oscillators until the temperature goes below 70K. The drop in the delay oscillator period could have been caused by the unsupported 100 MHz to IDELAYCTRL, this frequency was used to prevent the usage of a MMCM or PLL.



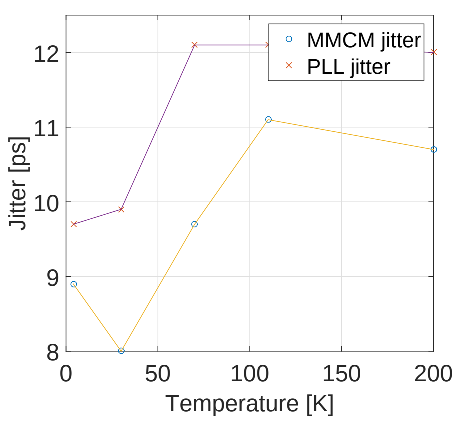
(a) Oscillators with LUTS and Carry elements



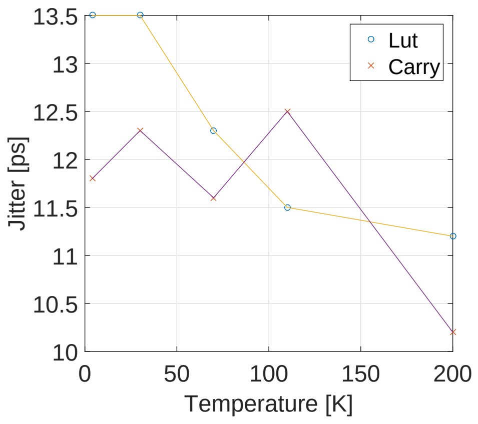
(b) Oscillator with IDELAYE2 components

Figure 4.25: The power consumption of the MMCM and PLL

4.4.4 Jitter analysis



(a) Jitter on the PLL and MMCM



(b) Jitter on the logic oscillators

Figure 4.26: The last steps of decoding

Figure 4.26 shows the jitter of the clocks and logic oscillators vary slightly. There isn't a pattern or significant change. Thus its safe to assume that this wouldn't influence the FPGA timing constraints. Figure 4.27 shows an shocking increase of the IDELAYE2 jitter. This could be caused by the unsupported IDELAYCTRL frequency. But for the ADC it would be terrible to have a jitter higher then 80 ps, as that stands for roughly 5 LSB.

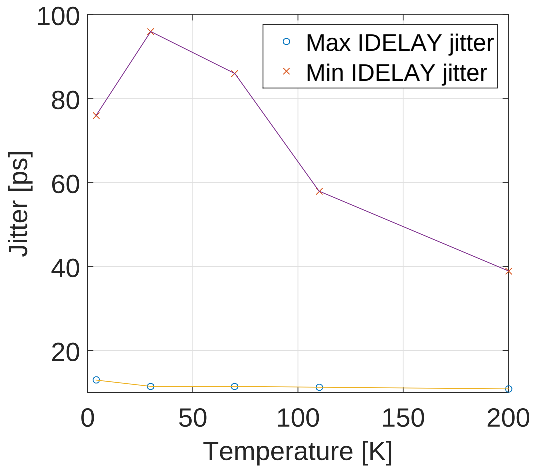


Figure 4.27: The jitter of the IDELAYE2 oscillator

4.4.5 Cryogenic FPGA ADC

The 1.2GSa/s ADC was successfully programmed into the ARTIX-7 at 4 Kelvin. But the voltage drop caused to much distortion, to perform the calibration (This was also the case at roomtemperature). To still get any results, the 600MSa/s ADC was programmed in the ARTIX-7. With this setting the FPGA could execute all calibration steps. Resulting in a measurements that confirmed that the ADC was fully functional. Figure 4.28 shows the measurement results of a 500KHz sinusoidal.

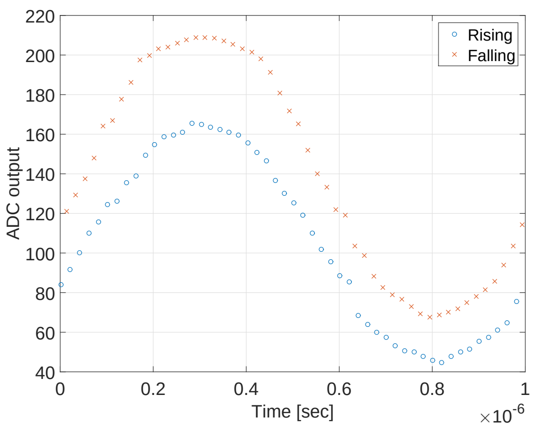


Figure 4.28: The measured 500KHz sinusiodal

5 Recommendations / future work

The results of this thesis proved that it is possible to create a 1.2GSa/s ADC on an FPGA. It also showed full functionality of the FPGA at 4 Kelvin. However there are still remaining problems before this work could be implemented in a quantum computer error correction loop.

5.1 Range

Multiple applications don't work with a range that is between 0.5 and 2.0 Volts, they need to be able to measure small positive and negative voltages. A range between -1 and 1 volt would be ideal. For that kind of applications an operational amplifier could offer a solution. By using an amplifier the input capacitance can be reduced (now estimated at 54 pF, see subsection 4.3.1). Even when the operational amplifier deforms the output at low temperatures, these effects might be automatically compensated for in the calibration procedure.

5.2 Power dissipation

The current power dissipation of 800 mW is a serious problem for operating at cryogenic temperatures. By changing algorithms and by optimizing the VHDL code the power consumption can be reduced. However, the fact that 6 carry-chain decoders need to process 80Gb per second, 480Gb in total, is a problem.

5.3 Scalability

For now the FPGA uses 20% of its resources with the 1.2GSa/s ADC. Probably 3 or 4 of those ADCs could fit into the ARTIX-7 100k. All of those ADCs could be connected to the same RC curves. With some effort the efficiency of the TDC decoders and other components can be increased. Combine this with the usage of a bigger FPGA, it can be possible to make an ADC with 16 or even 32 channels.

5.4 Auto calibration

One of the bottlenecks of this design is the requirement of calibration signals. Preferably the FPGA can calibrate itself completely. This would make the ADC more user friendly and would make it possible to create a development board that directly supports this ADC. Basically the calibration of the ADC could be implemented by adding one resistor (50kΩ for example), to the board and connecting the 50Ω closing resistor to an FPGA pin. Given that the FPGA pins are tristate pins, we could put the 50Ω resistor to high impedance and charge the parasitic capacitance of the ADC with the 50kΩ resistor. This could produce a RC charging curve which can be used to calibrate the time-to-voltage conversion. Afterwards the 50kΩ resistor can be put in high impedance and the 50Ω resistor could produce a fast RC pulse for the synchronization of the multiple carry-chains. After the calibration the 50Ω can serve as a closing resistor.

Conclusion 6

A completely reconfigurable design was presented, using a low cost Artix-7 FPGA. The design has a sampling rate as high as 2.4 GSa/s and it achieves a resolution of 9 bits (ENOB). A full characterization has been done for the 1.2 GSa/s variant. The measurements resulted in an ENOB of 6 bits, a single shot of 1.1 LSB, and a high linearity (DNL [-0.8 1.1] LSB and INL [-0.4 0.45] LSB). To demonstrate the functionality of this design, measurements were performed on 3 PCBs, showing minimal differences between the boards.
The reconfigurable design was demonstrated to work from 12.5 MSa/s up to 2.4 GSa/s. This work showed that the ADCs can be calibrated in terms of the TDC length, TDC alignment and ADC synchronization. Thanks to the synchronization of single ADC channels, higher sampling rates can be achieved without losing much of the ENOB.
This ADC is not only reconfigurable, but it can also be calibrated for any change in operating environment, compensating for either changes in temperature, voltage or in between devices. The system is completely soft-core and can be implemented in an FPGA, only 7 additional resistors are needed (6 for the RC-curves and one 50 Ω closing resistor).
The Artix-7 was tested at 4 Kelvin to demonstrate usefulness for quantum computing applications, during these tests the FPGA was fully functional. The logic of the ADC worked as well, but the output was distorted (this was also the case at room temperature), caused by a voltage drop. To counter these effects, a logic speed stabilizer, that stabilizes the power consumption, was designed and showed to prevent distortions.
To the best of our knowledge this is the fastest fully reconfigurable sof-core FPGA ADC reported to date. Thanks to the reconfigurability and calibration feature, this ADC can be used in many applications with specifications ranging from low sampling speed and high accuracy to very high sampling rates as 1.2 GSa/s.

Bibliography

[1] “Integrated ADC for Altera Cyclone-IV devices,” Apr. 2011.

[2] J. Acero, D. Navarro, L. Barraga, I. Garde, J. Artigas, and J. Burdio, “FPGA-based power measuring for induction heating appliances using sigma delta A/D conversion,” *IEEE Transactions on Industrial Electronics*, vol. 54, p. 18431852, Jun 2007.

[3] F. Sousa, V. Mauer, N. Duarte, R. Jasinski, and V. Pedroni, “Taking advantage of LVDS input buffers to implement sigma-delta A/D converters in FPGAs,” *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, 2004.

[4] J. Wu, S. Hansen, and Z. Shi, “ADC and TDC implemented using FPGA,” in *Nuclear Science Symposium Conference Record, 2007. NSS ’07. IEEE*, pp. 281–286, Oct. 2007.

[5] M. Pałka, T. Bednarski, P. Białas, E. Czerwiński, Ł. Kapłon, A. Kochanowski, G. Korcyl, J. Kowal, P. Kowalski, T. Kozik, *et al.*, “A novel method based solely on fpga units enabling measurement of time and charge of analog signals in positron emission tomography,” *arXiv preprint arXiv:1311.6127*, 2013.

[6] C. Favi and E. Charbon, “A 17ps time-to-digital converter implemented in 65nm FPGA technology,” *Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays - FPGA 09*, 2009.

[7] J. Wu, J. Odeghe, S. Stackley, and C. Zha, “Improving single slope ADC and an example implemented in FPGA with 16.7 GHz equivalent counter clock frequency,” in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE*, pp. 2183–2187, IEEE, 2011.

[8] H. Homulle, F. Regazzoni, and E. Charbon, “200 MS/s ADC implemented in a FPGA employing TDCs,” in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA ’15, (New York, NY, USA), pp. 228–235, ACM, 2015.

[9] A. A. A. B. N. A. M. S. Douglas Sheldon, Gary Burke, “Cryogenic operation of field programmable gate arrays,” 2011.

[10] A. Bakhshi, “Fpga-based instrumentation withstands the chill of deep space,” *Xcell journal*, vol. 80, pp. 24–27, 2012.

[11] T. Liu, D. Gong, S. Hou, C. Liu, D.-S. Su, P.-K. Teng, A. Xiang, and J. Ye, “Cryogenic digital data links for the liquid argon time projection chamber,” *Journal of Instrumentation*, vol. 7, no. 01, p. C01091, 2012.

[12] J. Hornibrook, J. Colless, I. C. Lamb, S. Pauka, H. Lu, A. Gossard, J. Watson, G. Gardner, S. Fallahi, M. Manfra, *et al.*, "Cryogenic control architecture for large-scale quantum computing," *Physical Review Applied*, vol. 3, no. 2, p. 024010, 2015.

[13] D. J. Reilly, "An fpga-based instrumentation platform for use at deep cryogenic temperatures," *arXiv:1509.06809*, 2015.

[14] A. to Digital Converters Working Group, "Ieee standard for terminology and test methods for analog-to-digital converters," *IEEE Std*, 2011.

[15] Xilinx, "Artix-7 fpgas data sheet: Dc and ac switching characteristics (ds181)," September 2015.

[16] K. H. Lundberg, "Noise sources in bulk cmos," vol. 3, 2002.

[17] T. C. C. Herve, J.Cerrai, "High resolution time-to-digital converter (TDC) implemented in field programmable gate array (FPGA) with compensated process voltage and temperature (PVt) variations," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 682, p. 1625, Aug 2012.

[18] J. Song, Q. An, and S. Liu, "A high-resolution time-to-digital converter implemented in field-programmable-gate-arrays," *IEEE Transactions on Nuclear Science*, vol. 53, p. 236241, Feb 2006.