An FPGA-based 7-ENOB 600 MSample/s ADC without any External Components

Lukas Leuenberger
OST – Eastern Switzerland University of Applied Sciences
IMES Institute for Microelectronics and Embedded Systems
Rapperswil, Switzerland
lukas.leuenberger@ost.ch

Dorian Amiet
OST – Eastern Switzerland University of Applied Sciences
IMES Institute for Microelectronics and Embedded Systems
Rapperswil, Switzerland
dorian.amiet@ost.ch

Tao Wei
University of Rhode Island
Department of Electrical, Computer, and Biomedical Engineering
Kingston, USA
tao_wei@uri.edu

Paul Zbinden
OST – Eastern Switzerland University of Applied Sciences
IMES Institute for Microelectronics and Embedded Systems
Rapperswil, Switzerland
paul.zbinden@ost.ch

ABSTRACT

Analog to digital converters (ADCs) are indispensable nowadays. Analog signals are digitized earlier and earlier in the processing chain to reduce the need for complex analog signal processing. For this reason, ADCs are often integrated directly into field-programmable gate arrays (FPGA) or microprocessors. However, such ADCs are designed for a specific set of requirements with limited flexibility. In this paper, a new structure of an FPGA-based ADC is proposed. The ADC is based on the slope ADC, where a time-to-digital converter (TDC) measures the time from the beginning of a reference slope until the slope reaches the voltage-to-be-measured. Only FPGA-internal elements are used to build the ADC. It is fully reconfigurable and does not require any external components. This innovation offers the flexibility to convert almost any digital input/output (I/O) into an ADC. Considering the very high number of digital I/O ports available in today's FPGA systems, this enables the construction of a massive and powerful ADC array directly on a standard FPGA. The proposed ADC has a resolution of 9.3 bit and achieves an effective number of bits (ENOB) of 7 at a sample rate of 600 MSample/s. The differential nonlinearity (DNL) ranges from -0.9 to 0.9 bit, and the integral nonlinearity (INL) is in the range between -1.1 and 0.9 bit. An alternative version of the ADC operates at 1.2 GSample/s and achieves an ENOB of 5.3.

CCS CONCEPTS

• **Hardware → Reconfigurable logic and FPGAs**; **Data conversion**.

FPGA '21, February 28-March 2, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8218-2/21/02…$15.00
https://doi.org/10.1145/3431920.3439287

KEYWORDS

FPGA; Analog-to-Digital Converter; ADC; Time-to-Digital Converter; TDC

ACM Reference Format:
Lukas Leuenberger, Dorian Amiet, Tao Wei, and Paul Zbinden. 2021. An FPGA-based 7-ENOB 600 MSample/s ADC without any External Components. In *Proceedings of the 2021 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '21), February 28-March 2, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3431920.3439287

1 INTRODUCTION

In today's world, analog to digital converters (ADCs) are indispensable. ADCs exist in numerous implementations, covering a wide range of specifications to meet various application requirements. Currently, there exists a strong tendency to move the digitization process earlier in the processing chain to eliminate the need for complex analog signal processing. As a result, many ADCs have been integrated directly into digital platforms, such as field-programmable gate arrays (FPGAs) and microprocessors.

Such ADCs are often designed for a specific set of requirements with limited flexibility once built. Latest publications showed that it is possible to implement an ADC in an FPGA using reconfigurable logic blocks of the FPGA. These ADCs are based on the slope ADC concept. They measure the time from the start of a reference slope until that slope reaches the voltage-to-be-measured. The time can be measured with a time-to-digital converter (TDC). Advantages of such ADCs include low overhead, meaning that they hardly require any additional external elements, and reconfigurability, which allows them to be adapted to a broad range of requirements.

In this paper, a new structure of an FPGA-based slope ADC is proposed. The ADC requires only one pair of LVDS pads and is built using only FPGA internal elements. To the authors knowledge, this is the first FPGA-based ADC that does not require additional external resistors and capacitors. Instead of using external components, the ability to change the output impedance in the OBUFT

The code to this paper is available under https://doi.org/10.5281/zenodo.4016001.

blocks together with the parasitic pad capacitance is used to create the reference slope. A tapped-delay line TDC is used to measure the time from the start of the slope until the slope reaches the voltage-to-be-measured. Further, different linearization and correction schemes are deployed to improve the resolution of the ADC. By completely removing the need for external components, this innovation provides the flexibility to convert almost any digital input/output (I/O) into an ADC, especially considering the very high number of digital I/O ports available in today's FPGA systems. It gives rise to build a massive and high-performance ADC array directly on an FPGA.

This paper is structured as follows: Published designs of FPGA-based ADCs are reviewed in Section 2. Section 3 describes the architecture of the ADC in detail. Finally, in Section 4, the ADC is characterized and compared with other published designs of FPGA-based ADCs.

2 FPGA-BASED ADCS

The first, to our knowledge, implementation of an ADC inside an FPGA dates back to 2004 and was presented by Sousa et al. in [9]. This implementation deploys a delta-sigma ADC. The integrator of the delta-sigma ADC is made of an external resistor together with an external capacitor. An LVDS comparator from the FPGA is used as a comparator. An ENOB of 9.87 bit with a sampling rate of 48.8 kHz was achieved. The same principle was later used by Zimmermann et al. in [17]. The publication describes a first-order delta-sigma ADC with 500 kSample/s while achieving 10 ENOB. Two external resistors and one external capacitor were used.

One of the first publications describing an FPGA-based slope ADC was presented by Wu et al. in 2007 [14]. The slope is created with three external resistors and one external capacitor. The so created reference slope is brought to an LVDS comparator where it is compared to the voltage-to-be-measured. The time between the start of the slope and the crossing of the slope with the input signal is measured with a multi-phase clock TDC. This TDC uses phase-shifted clocks to measure the time [2]. The resolution of this TDC is proportional to the number of used clocks. It needs a careful routing of all signals as the time measurement relies heavily on the phase difference of the individual clocks [6]. The ADC from [14] achieved a sampling rate of 22.5 MSample/s and a resolution of 6 bit.

Instead of using a multi-phased clock TDC, a tapped-delay line TDC [1] can be used. In this structure, the input signal is delayed multiple times with equal delay elements. The delay of individual delay elements define the resolution of this TDC. This type of TDC can easily be integrated into an FPGA. There exists a fast carry chain, which offers delay elements with a delay in the order of a few picoseconds. Due to FPGA process variations, the delay of the individual elements slightly differs. Different routing delays between the individual elements of the carry chain as well as clock jitter effectively limit the resolution of the TDC [3] and are causing bubbles, e.g. discontinuities in the measured signal. However, different techniques exist to increase the linearity and resolution of the TDC.

Pałka et al. created an ADC which uses such a tapped-delay line TDC [7]. Multiple fixed reference voltages were used together with multiple LVDS comparators instead of a slope. Homulle et al. later combined the approach of the tapped-delay line TDC with the slope ADC and built a 200 MSample/s ADC with an ENOB of 6 bit [4]. Instead of using an external capacitance, the parasitic capacitance of the pad is used to create a slope. This design effectively reduced the number of external components to one resistor.

Visser and Homulle later refined the design in [11] and [5]. In the latter publication, a 1.2 GSample/s ADC consisting of three interleaved 400 MSample/s ADCs is built. An ENOB of 6 bit was achieved whereby only four external resistors and four LVDS comparators were used.

In [16], Xiang et al. built an 800 MSample/s ADC. The design uses one external resistor and the parasitic capacitance of the LVDS pad. A multi-phase clock TDC that uses the ISERDES blocks from the FPGA is used instead of a tapped-delay line TDC. The paper states that this allows the integration of more ADCs in an FPGA than the tapped-delay line TDC approach. This ADC achieved an ENOB of 3.8 bit for a sampling rate of 800 MSample/s.

Figure 1 shows a comparison of the mentioned ADCs including the proposed ADC from this work. Detailed performance results of our ADC are provided in Section 4.

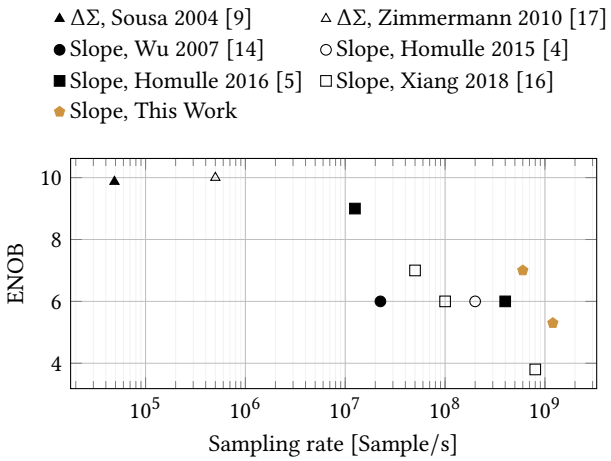


Figure 1: Our ADC compared to other FPGA-based ADCs reported in the literature.

3 ARCHITECTURE

This section describes the architecture of the proposed ADC. The structure of the ADC is explained in Section 3.1. The I/O structure and the creation of the slope is described in Section 3.2 and Section 3.3. An in-depth view of the TDC is given in Section 3.4. Necessary calibration procedures are explained in Section 3.5.

The whole ADC was integrated and tested in the Ultrascale+ FPGA XCZU7EV-2FFVC1156. For synthesis and implementation, the tool Xilinx Vivado 2019.1 is used.

3.1 Structure

The proposed ADC consists of an LVDS input buffer, a single-ended output buffer, a TDC, and a mixed-mode clock manager (MMCM).

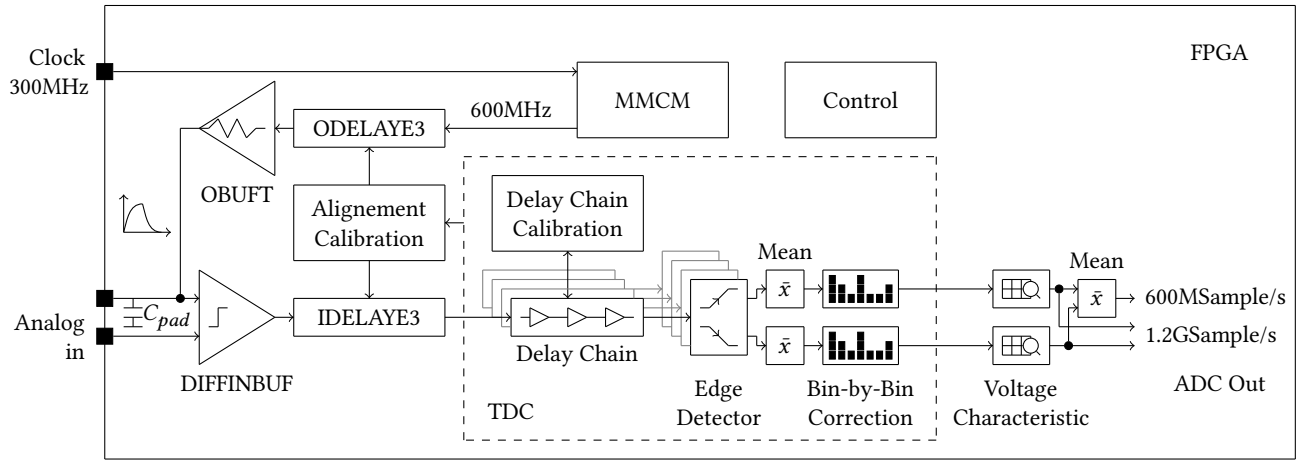


Figure 2: The proposed ADC including all linearization and correction techniques. All logic blocks in the TDC and ADC path run at a clock frequency of 600 MHz. The calibration and control logic runs with a reduced clock speed of 200 MHz.

The input and output buffer are located in the same I/O cell. Further, some calibration and control blocks are deployed. Figure 2 shows the structure of the system and Figure 3 shows an example timing diagram of the ADC. The MMCM creates a 600 MHz clock which is brought to one input of the LVDS buffer via an output buffer. The output impedance of the output buffer *OBUFT* together with the parasitic capacitance C_{pad} creates a slope on the input of the LVDS comparator *DIFFINBUF*. The slope is compared to the analog input signal. The time from the start of the pulse until the output of the comparator changes is measured by the subsequent TDC. Different correction schemes are deployed to increase the linearity of the measured values.

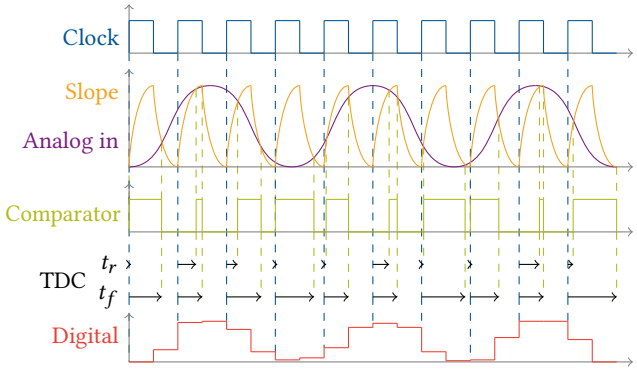
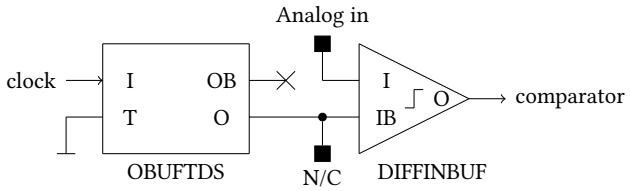


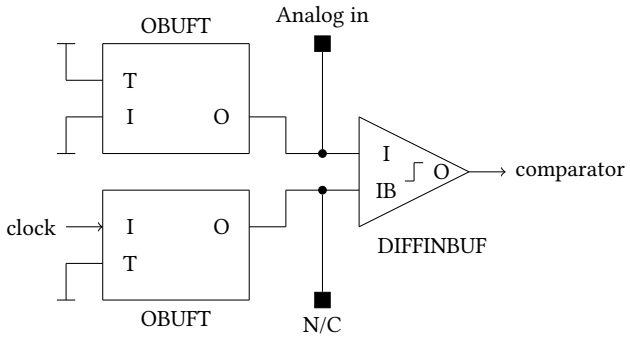
Figure 3: The reference slope (orange) is generated from a 600 MHz clock (blue). The LVDS comparator compares the analog input signal (purple) with the slope and generates an output signal (green). A subsequent TDC measures the time between the start of the pulse (dashed blue) and the two edges of the comparator output (dashed green). This results in two times t_r and t_f (black) which are a digital representation (red) of the voltage-to-be-measured.

3.2 Input/Output Structure

The input and output buffer need to be connected in a very specific way. One pin of the LVDS pair needs to be configured as an input/output pin, whereas the second pin is only an input pin (see Figure 2). Both pins are connected to an LVDS input buffer. Additionally, the I/O-pin is also connected to an output buffer. Figure 4 shows two possible structures.



(a) with differential output buffer



(b) with single-ended output buffer

Figure 4: Two possibilities to implement the input/output structure.

The structure from Figure 4a uses a differential output buffer to create the slope. In a typical application, both outputs from the

output buffer would be connected to the input buffer. If instantiated by hand, it is possible to route only one signal from the output buffer and leave the other signal unconnected. This structure can be implemented in Vivado without any error or critical warning and a valid bitstream is generated. Unfortunately, when tested on the FPGA, both signals from the output buffer are disconnected from the input buffer, although the implemented design shows one connected signal.

An alternative structure is shown in Figure 4b. This structure uses two single-ended output buffers. Both buffers are connected to the input buffer, while one output buffer is set to tristate. Two output buffers are needed because both pins need to be configured as input/output pins as Vivado does not allow to have two different I/O directions within a single LVDS pair. If the *in*-port is not connected to an output buffer, an error will arise stating that an I/O-port needs to have an output buffer. Note that this structure produces a DRC error during placement. The error states that a single-ended buffer is not allowed to drive a differential input. If this error is suppressed during placement and re-enabled afterward, a valid bitstream is still generated. Our experiments further showed that this structure has no negative impact on the device.

3.3 Slope Creation

Xilinx introduced the possibility to change the output impedance of the high-performance output buffers in the Ultrascale FPGA family. Together with the ability to change the slew rate of the output buffer, nine different slopes can be generated. If the impedance of the output buffer is set to the highest value of 60 Ω and the slew rate to the lowest setting of *SLOW*, the output signal is slow enough to rise from 0 V to 1.8 V within approximately half a period of the 600 MHz clock. The slope generated this way has the advantage that it is slew rate limited and has therefore a more linear shape than a slope that uses only the charging curve of the parasitic pad capacity C_{pad}.

3.4 TDC

A tapped-delay line TDC is used to measure the time from the beginning of the slope until the slope reaches the voltage level of the input signal. The carry chain is used to form the delay chain. To capture the signal which is traveling through the carry chain, flip-flops (FFs) are used. The FFs are located very near to the individual outputs of the carry elements and thus allow an accurate capture of the delay line state. The outputs of these FFs are routed to a second FF stage to avoid metastable phases. Metastability can occur in the first FF stage whenever the setup and hold times of the FFs are not met.

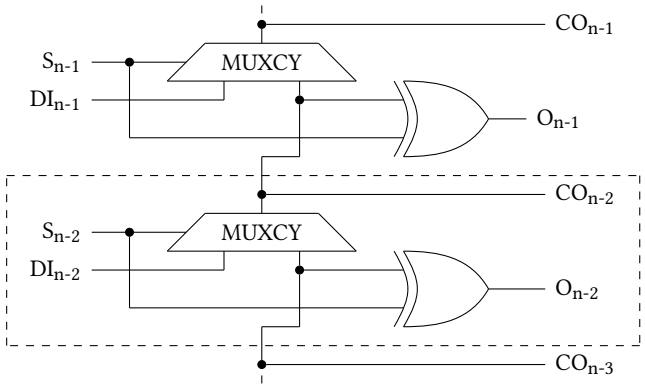


Figure 6: The used delay elements from the *CARRY8* block. The dashed box marks one carry element.

The carry chain consists of multiple *CARRY8* blocks. Each block contains eight *MUXCY* elements acting as delay elements. Figure 6 shows the structure of two delay elements. The data inputs *DI* are not used, and therefore, all select signals *S* need to be set to high.

3.4.1 Dual Sampling. Instead of using only the eight outputs of the *MUXCY* blocks, Wang and Liu proposed in [12] to use also the outputs from the eight XOR gates to double the number of measured values. Because the select signals *S* of the *MUXCY* blocks are high, the outputs from the XOR gates are inverted. This can be resolved by adding an inverter after the first FF stage.

3.4.2 Reordering based on Static Timing Analysis. Bubbles are discontinuities in the measured signal of the delay chain. These bubbles can occur due to different characteristics of the delay elements and the routing, due to clock skew and jitter of the sampling clock [3], or due to metastability of the FFs [1].

Xilinx Vivado allows to report the static timing for the *CARRY8* elements in the FPGA. The static timing analysis reveals that the delay from the input of a *CARRY8* block to the first carry output CO_0 is higher than the delay to the output CO_7. As it is unclear how the *CARRY8* block is implemented in hardware, it is expected that this discrepancy arises because of routing and parasitic elements inside the *CARRY8* element.

Experiments showed that the number of measured bubbles in the captured signal can be reduced if the *CO* and *O* outputs of the *CARRY8* elements are ordered based on the static timing analysis. Figure 5 shows the ordering for a delay chain with three *CARRY8* blocks. This reordering comes for free in the implementation, as

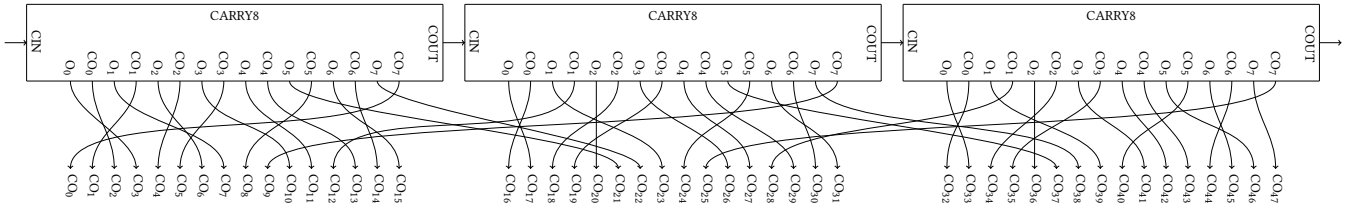


Figure 5: The outputs of *CARRY8* blocks are reordered based on the timing reported by the static timing analysis from Vivado.

simply the routing between the two first FF stages is changed. The measurement result of this experiment is provided in Figure 7.

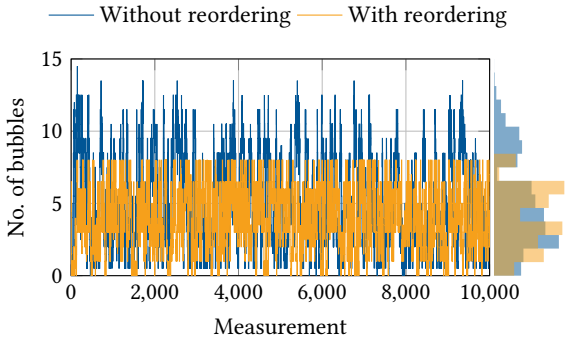


Figure 7: The blue line shows the number of bubbles if the outputs from the *CARRY8* blocks are not sorted. The orange line shows the number of bubbles if the outputs are sorted based on the reported delay from the static timing analysis. This measurement was produced by a single delay chain where the second FF stage was duplicated.

3.4.3 Edge Detector and Bubble Filter. The edge detector detects the first falling and rising transition in the captured signal from the delay chain. Further, it handles the elimination of bubbles in the signal. Figure 8 shows the functionality of this block.

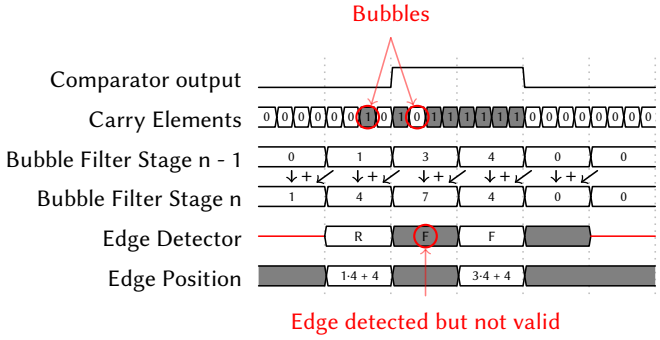


Figure 8: Timing diagram of how the bubble filter and edge detector work. First, the overlapping sum over $k = 8$ carry elements is calculated. Afterward, the edges are detected based on these sums.

First, the sum over $\frac{k}{2}$ carry elements is calculated. This happens in the bubble filter stages 1 to $n-1$. In stage n, an overlapping sum S is calculated for each block i.

$$S(n,i) = S(n-1,i) + S(n-1,i+1) \tag{1}$$

This overlapping sum makes sure that transitions, which occur exactly between $\frac{k}{2}$ carry elements, are detected. The size k of the sums determines how strong bubbles are filtered. If k is larger, bubbles in a wider neighborhood will be filtered out.

Algorithm 1 Edge Detection

```
for i ← 2 to (l_dc·2)/k − 2 do ▷ l_dc: maximum length of delay chain
    transitionValid(i) ← 0          ▷ Initialize to invalid transition
    if S(n,i) ≠ 0 and S(n,i) ≠ k then
        if S(n,i+1) > k/2 then
            transitionDir(i) ← 0                    ▷ Falling edge
            if S(n,i−1) < k/2 then
                transitionValid(i) ← 1              ▷ Valid transition
            end if
        else if S(n,i+1) < k/2 then
            transitionDir(i) ← 1                    ▷ Rising edge
            if S(n,i−1) > k/2 then
                transitionValid(i) ← 1              ▷ Valid transition
            end if
        end if
    end if
end for
```

After all sums are calculated, all edges in the signal are detected. Algorithm 1 shows the algorithm used to detect the edges.

Because it is necessary for the transition detection algorithm to check the sums before and after the current sum, it is not possible to detect edges in the first and last $\frac{k}{2}$ carry elements.

When all edges are identified, the positions of the transitions inside the delay chain are calculated. The position *Pos* for each edge can be calculated with Equation 2.

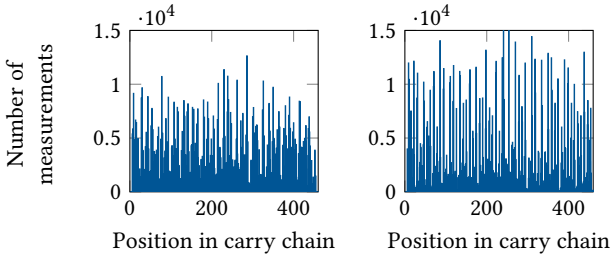
$$Pos(i) = i \cdot \frac{k}{2} + S(n,i) \tag{2}$$

Once all positions are calculated, the first rising edge and the first falling edge is determined and used for further processing. Note that the slope is generated such that precisely one falling and one rising edge should appear in the carry chain per sampling period.

3.4.4 Bin-by-Bin Correction. The carry chain is not as uniform as it would be possible with a dedicated delay chain in an application-specific integrated circuit (ASIC). Instead, some delays may have nearly no delay while other delay elements have a multiple of the delay of a single delay element. Figure 9 shows the histogram of the implemented carry chain. The histogram would be flat if all carry elements had the same delay.

Note that the histogram for the rising and falling edge is similar but not identical. This phenomenon was also observed by Szplet et al. in [10] and Wang and Liu in [12]. The second paper claims that the rising edge moves roughly 5 % faster through the carry chain in an Ultrascale FPGA than a falling edge.

For a more accurate TDC linearization, Wu and Shi propose the bin-by-bin correction scheme [15]. In this scheme, essentially the DNL and INL are calculated from the histogram and the measured value is corrected based on the INL. In a later paper [13], Wu mentions that it is better to correct the center of the bin instead of one of the edges as this reduces the root-mean-square (RMS) measurement error. The INL for every bin of the delay chain can therefore be calculated as shown in Equation 3.



(a) Falling edge **(b) Rising edge**

Figure 9: Histogram of the measured positions of the falling and the rising edge inside the carry chain. The histogram shows that the delays of the carry elements are not equal as otherwise the histograms would be flat.

$$INL(k) = \sum_{i=1}^{k-1} DNL(i) + \frac{DNL(k)}{2} \tag{3}$$

Before the actual measurement starts, the histogram, DNL, and INL are calculated in the FPGA. A TDC linearization look-up table is then calculated in the FPGA. During the measurement phase, the measured values are corrected by looking-up the corrected value from the pre-calculated look-up table. Note that this bin-by-bin correction is performed for the falling and rising edge individually.

3.4.5 Parallel Instances. Another approach to increase the resolution and linearity of the delay chain is the use of multiple parallel delay chains. Shen et al. showed in [8] that the effective bin width resolution (e.g. the time which is represented by one bin) can be nearly linearly reduced with the number of used parallel instances.

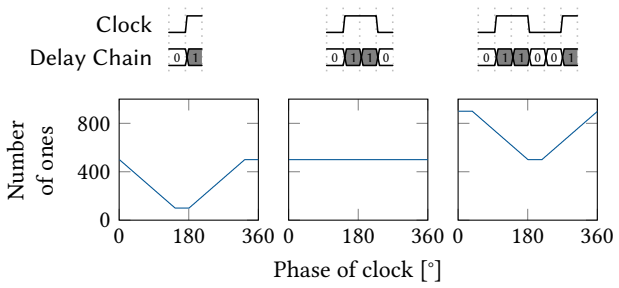
Multiple structures with different amounts of parallel instances were tested in our implementation. Experiments showed that four parallel instances of the carry chain together with four edge detectors and bubble filters offered a good trade-off between DNL, ENOB of the ADC, and required FPGA resources.

3.5 Calibration

Before the ADC is able to provide accurate results, three calibration steps need to be performed. These calibrations need to be repeated periodically to ensure continuous accurate results.

3.5.1 Length of the Delay Chain. The time needed for a signal to travel through the delay chain can vary because of many different circumstances. For example, the temperature of the silicon die directly influences the propagation delay time. Therefore, the length of the delay chain needs to be determined based on the actual delay through the delay elements. Based on the overall length of the delay chain, the mean delay through one single delay element can be calculated.

To determine the length of the delay line, a procedure as proposed by Visser in [11] or later by Homulle et al. in [5] can be applied. This method uses a phase-shifted clock, which is fed into the delay line. The clock must have the same period as the sampling clock. For each period, the total number of measured ones is determined. Based on the length of the delay line, a result as shown in Figure 10 appears.



(a) too short **(b) correct** **(c) too long**

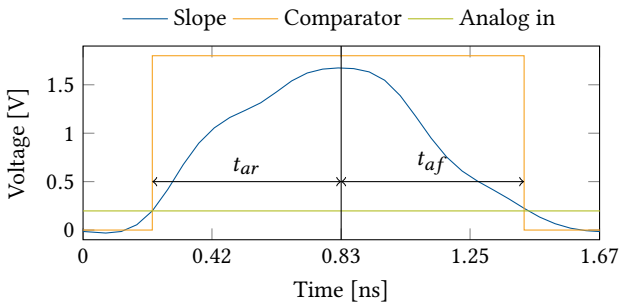
Figure 10: Influence of the length of the delay chain (adapted from [5, 11]). If the delay chain is too short, a part of the high part of the clock is cut off. This results in an underestimation of the length. If the delay chain is too long, the previous clock period is still present in the delay chain producing an overestimation of the length.

If the delay chain is too short, the high phase of the clock is cut off for certain phases. This results in a reduced number of counted ones. Conversely, if the delay chain is too long, the number of counted ones will be too high for certain phases as the high phase of the clock appears twice in the delay chain.

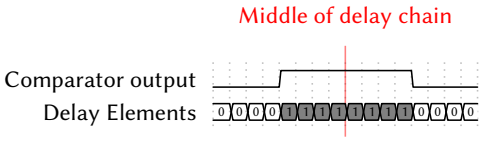
In the implemented ADC, the delay chain is implemented to be longer than needed and thus a situation as shown in Figure 10c will occur. The correct length of the delay chain is then two times the minimum value of counted ones from all measurements for a clock with a duty cycle of 50 %.

3.5.2 Alignment of the Input Signal with the Sampling Clock. To accurately measure the time between the generation of the output pulse and the actual toggle of the comparator, the output signal of the comparator needs to be aligned. If, for example, an ideal pulse is generated (rising linearly from 0V to V_{DD} during half a sampling period) and an input voltage of $\frac{V_{DD}}{2}$ is applied to the second pin of the comparator, then the rising edge of the comparator output should be measured after $\frac{t_s}{4}$ where t_s is the time of the sampling period.

In hardware, no ideal impulse will be generated. The resulting curve at the input pin of the comparator looks more like the blue curve in Figure 11a. This blue pulse is compared to a DC voltage, which produces the orange comparator output. The DC voltage must be chosen such that the time of the rising edge t_{ar} and the time of the falling edge t_{af} are equal. The produced pulse from the comparator is then aligned such that the rising and falling edge are centered around the middle of the delay chain. This is shown in Figure 11b. The pulse of the comparator can be aligned to the middle of the delay chain by changing the input delay (*IDELAYE3* block) of the comparator output or the output delay (*ODELAYE3* block) of the clock.



(a) Time domain



(b) Delay chain

Figure 11: Alignment of the comparator output to the sampling clock. The generated pulse of the comparator output must be centered around the middle of the delay chain.

Note that the alignment does not need to be perfect. The LVDS comparator in the Ultrascale+ FPGA has two deadbands where no edges will be created. This has the consequence that there will never be an edge at the start or the end of the delay chain. Further, if the alignment is not perfect, this will appear as an offset voltage. Offset voltages will be corrected with the calibration of the voltage characteristic.

However, the calibration is still necessary to ensure that both edges appear in the correct order and in the same clock period. It is also necessary to ensure that the edges are aligned such that they are always out of the range where no edge can be detected by the edge detector (see Section 3.4.3). All these conditions are automatically fulfilled if the pulse of the comparator is aligned to the middle of the delay chain.

3.5.3 Voltage Characteristic. A third calibration takes the overall ADC voltage characteristic into account. This can be done by applying a known low-frequency ramp on the ADC input. All digitized samples are matched against the input voltage. The resulting table reflects the actual characteristic of the pulse, the comparator, and the TDC. Based on this characteristic, two look-up tables can be calculated (one for the rising edge and one for the falling edge). The look-up tables are used to correct the measured values. The voltage characteristic is shown in Figure 12.

3.6 Sampling Rate

The ADC supports two different sampling rates. A 600 MHz clock is generated internally by an MMCM and is routed to the output buffer (see Figure 2). The clock generates two slopes at the input of the LVDS comparator. During the first half period of the clock, a slope that rises from 0 V to 1.8 V is created whereas a falling slope originates during the second half of the clock. This corresponds to

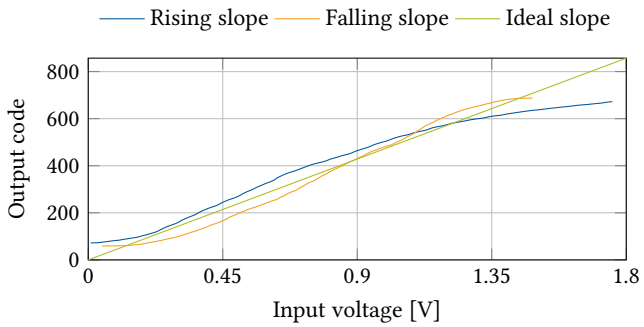


Figure 12: Voltage characteristic of the rising and falling slope.

two measurements per clock period, resulting in a sampling rate of 1.2 GSample/s. Another possibility is to calculate the mean value from these two measurements. This increases the ENOB for the price of a halved sampling rate.

Note that the sampling rate is chosen to match the slowest possible slope that can be generated with this FPGA. A slower sampling rate will not improve the ENOB, because it is not possible to generate a slower reference slope inside this FPGA. It is however possible to oversample the signal-to-be-measured. This leads to a slower effective sampling rate while at the same time a higher resolution can be achieved.

4 RESULTS

The proposed ADC is synthesized, placed and routed using Xilinx Vivado 2019.1, implemented on an Ultrascale+ FPGA XCZU7EV-2FFVC1156, and fully characterized. The first Section describes our test setup. Afterward, the utilization and different performance measurements are presented. Finally, the ADC is compared to other ADCs, which are based on a similar working principle.

4.1 Test Setup

The ADC is tested on a ZCU104 evaluation board from Xilinx. A custom printed circuit board (PCB) with an SMA connector is connected to the evaluation board using the FMC connector. The SMA connector is routed to one input of an LVDS pair while the other input is left unconnected. A function generator, namely an Agilent 33522A, is connected to the SMA connector. The generator is used to create different input signals to calibrate and characterize the ADC.

The measured data from the ADC is stored in an UltraRAM (URAM) FIFO. The ARM processor contained in the FPGA reads the data via an advanced extensible interface (AXI) and transfers the data via ethernet to the host computer, where it is evaluated with Matlab.

4.2 Utilization

The full ADC needs around 20,500 look-up tables (LUTs) and 32,000 flip-flops (FFs). Table 1 shows the total number of used elements as well as the utilization inside the used FPGA.

Table 1: Utilization results of the ADC.

Element	Used	Available	Percentage
LUT	20,472	230,400	8.88 %
FF	31,870	460,800	6.92 %
BRAM	15	312	4.81 %

Most elements are occupied by the edge detector, which is instantiated four times. The edge detector requires 4,115 LUTs and 5,566 FFs. This block requires many resources because it is heavily pipelined to handle the clock speed of 600 MHz. It would be possible to use only one carry chain together with one edge detector to save FPGA resources. However, the resulting ADC would have a smaller ENOB.

4.3 Latency

The latency through the whole ADC, meaning the time it takes until a measured sample can be seen at the output, is 26 clock cycles. This is equal to $26 \cdot 1.67\text{ns} = 43\text{ns}$. Most clock cycles are used by the edge detector which uses 13 cycles. The bin-by-bin correction lasts seven cycles, the voltage characteristic block takes 4 cycles, and the delay chain requires two cycles.

4.4 Digital Range

The digital range of the ADC is determined by the TDC resolution. The mean TDC-length for a sampling clock of 600 MHz is 426 carry elements, e.g. the delay through these 426 carry elements is in the mean 1.67 ns. Due to the dual sampling approach, $426 \cdot 2 = 852$ sampling points are available[1]. The LVDS comparator itself has two deadbands. One deadband is located in the lower input region and the second deadband is located in the upper region. These deadbands slightly change from FPGA to FPGA. In our measurements, the deadbands were found to be from 0 V to 0.15 V and 1.45V to 1.8 V. If the rising and falling slopes are treated individually (e.g. a sampling frequency of 1.2 GSample/s is achieved), the digital range is

$$N_{1.2\text{ GSample/s}} = \log_2 \frac{426 \cdot (1.45\text{ V} - 0.15\text{ V})}{1.8\text{ V}} = 8.3\text{ bit} \quad (4)$$

On the other hand, if only a sampling frequency of 600 MSample/s is required, the digital range is

$$N_{600\text{ MSample/s}} = \log_2 \frac{852 \cdot (1.45\text{ V} - 0.15\text{ V})}{1.8\text{ V}} = 9.3\text{ bit} \quad (5)$$

As the LVDS comparator from the FPGA introduces the two deadbands, it is only possible to overcome these by using an external rail-to-rail comparator. The digital range would then be increased to $N_{1.2\text{ GSample/s}} = 8.7$ bit and $N_{600\text{ MSample/s}} = 9.7$ bit. Note that previously published papers (e.g. [4]) do not have this problem because older FPGAs with a different type of LVDS comparator were used.

[1]Note that the number of sampling points within the TDC is four times higher, as four parallel delay chains are used. Since the average of these measurements is calculated later in the signal path, the number of sampling points is then reduced again by a factor of four.

4.5 DNL and INL

The DNL and INL are determined using a code density test. A slow ramp, spanning the input range, is applied to the ADC. For each output code, the number of occurrences $C(k)$ are counted and stored. Based on the counted codes, the DNL and INL can be calculated with Equation 6 where N is the total number of output codes.

$$DNL(k) = \frac{C(k) \cdot N}{\sum_{i=1}^{N} C(i)} \quad (6a)$$

$$INL(k) = \sum_{i=1}^{k} DNL(i) \quad (6b)$$

For accurate results, the applied ramp spans only the input range. Figure 14 shows the assessed DNL and INL of the ADC. The DNL is in the range of -0.9 to 0.9 bit and the INL is in the range of -1.1 to 0.9 bit for a sampling rate of 600 MSample/s. The empty areas on the left and right side represent the two dead bands of the LVDS comparator. For a sampling rate of 1.2 GSample/s, the DNL is in the range of -1 to 1.6 bit and the INL is in the range of -1.1 to 1.2 bit.

4.6 SNR, SNDR and ENOB

The signal-to-noise-ratio (SNR), the signal-to-noise-and-distortion-ratio (SNDR), and the effective number of bits (ENOB) is determined from the measurement of a 10 kHz and a 30 MHz sine signal. The sine signal spans 90 % of the input range. Figure 13 shows the time domain representation of the 30 MHz signal. The frequency domain representation of the measured sine signals is provided in Figure 15 and is calculated from the time domain representation with the fast-fourier transform (FFT).

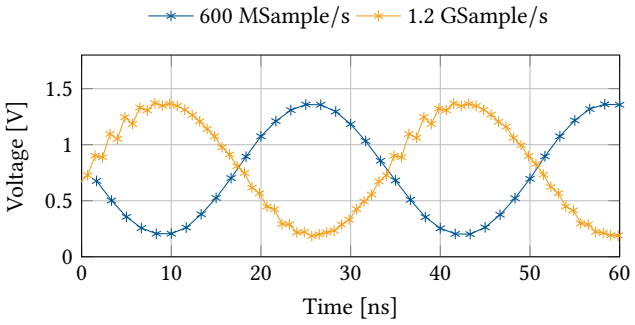
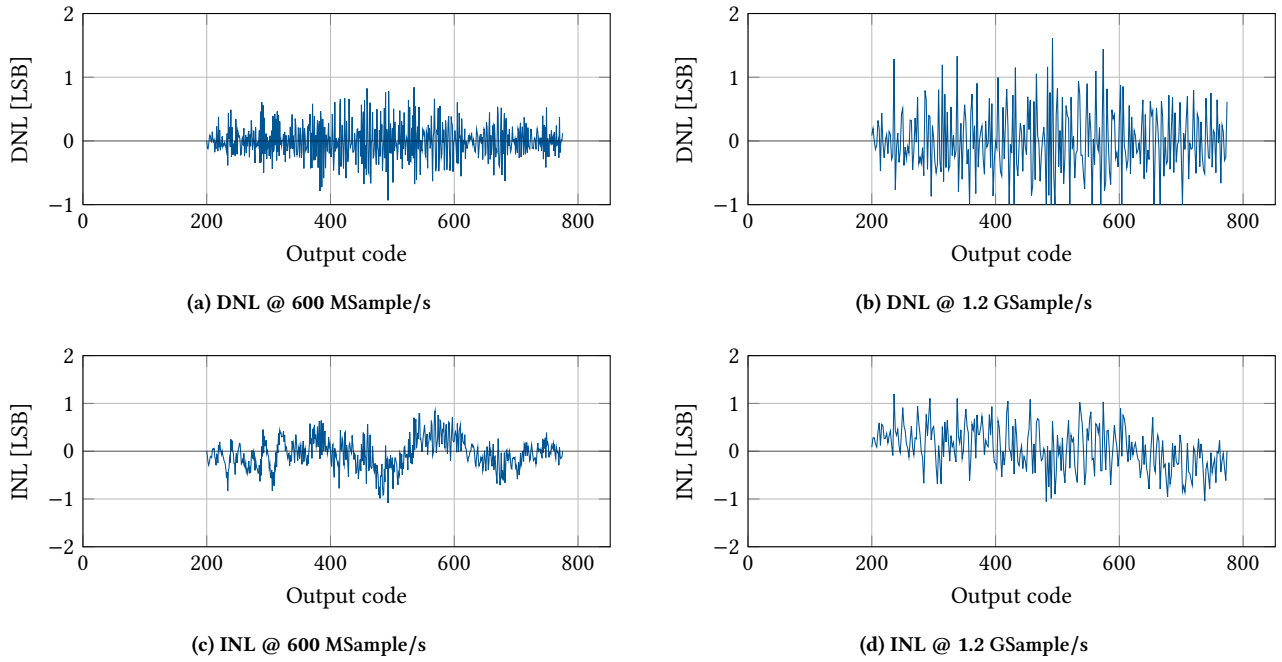


Figure 13: Time domain representation of a 30 MHz sine. The sine has been sampled with 600 MSample/s and 1.2 GSample/s. The 600 MSample/s signal is phase-shifted to provide a better view of the individual sampling points.

The sine sampled with 1.2 GSample/s shows small distortions. It is not clear where these distortions originate. One possible reason might be a non-linearity in the LVDS comparator. However, when the two samples from one clock period are averaged, the distortions are widely canceled out. This is the reason why the 600 MSample/s ADC performs significantly better than the 1.2 GSample/s ADC.



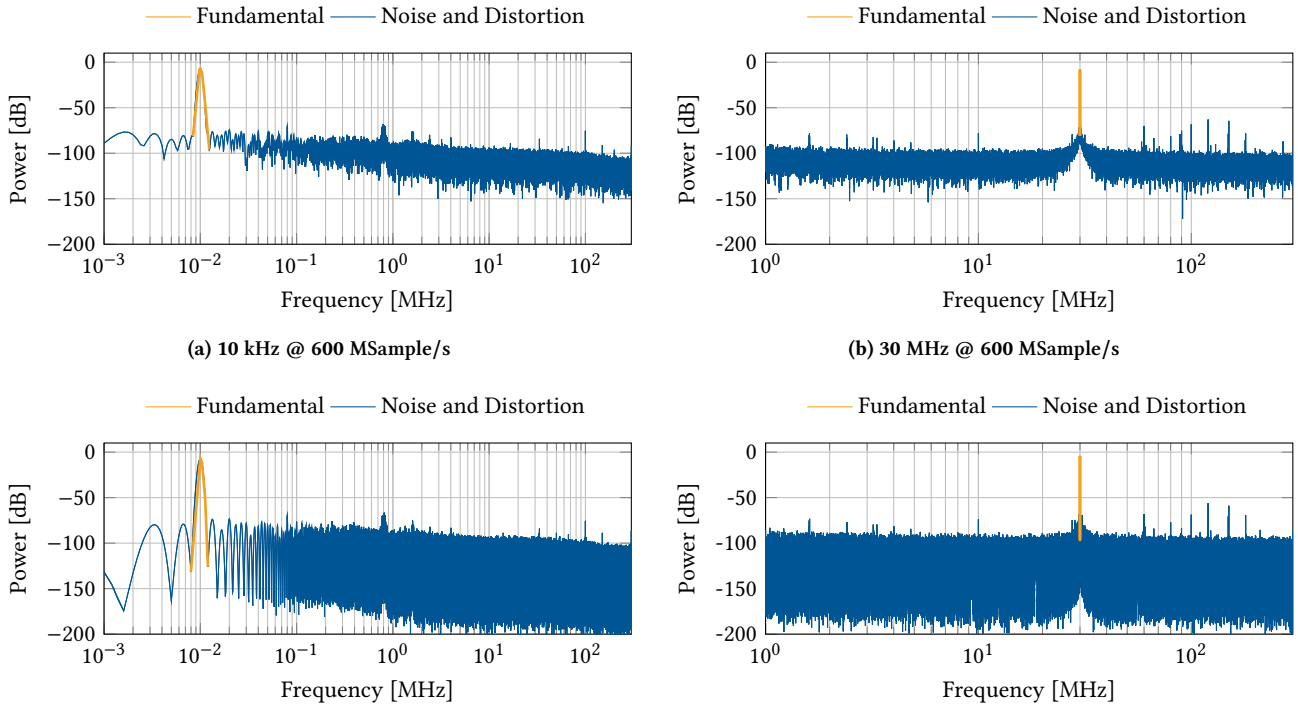
(a) DNL @ 600 MSample/s

(b) DNL @ 1.2 GSample/s

(c) INL @ 600 MSample/s

(d) INL @ 1.2 GSample/s

Figure 14: DNL and INL of the ADC. The DNL and INL are determined with a code density test. The empty areas represent the two dead bands of the LVDS comparator.



(a) 10 kHz @ 600 MSample/s

(b) 30 MHz @ 600 MSample/s

(c) 10 kHz @ 1.2 GSample/s

(d) 30 MHz @ 1.2 GSample/s

Figure 15: Frequency domain representation of a 10 kHz and 30 MHz sine. The signals have been sampled at 600 MSample/s and 1.2 GSample/s.

Table 2: Performance of the proposed ADC compared to other ADCs based on similar working principle. Unclear values are shown as N/A.

Reference	Device	Sampling rate MSample/s	Voltage range V	Digital range bit	ENOB bit	Resolution (LSB) mV	DNL LSB	INL LSB	External components	FPGA pins
This work	Ultrascale+	600	0.15 - 1.45	9.3	7.0	2	-0.9 to 0.9	-1.1 to 0.9	0	2
This work	Ultrascale+	1200	0.15 - 1.45	8.3	5.3	4	-1.0 to 1.6	-1.1 to 1.2	0	2
[14]	Altera Cyclone	22.5	0 - 3.3	6.0	N/A	52	N/A	N/A	4	4
[4]	Spartan-6	200	0 - 2.5	7.2	6.0	17	-0.9 to 1.4	-1.1 to 1.6	1	3
[5]	Artix-7	400[1]	0.9 - 1.6	8.2	6.0	3	-0.75 to 1.0	-0.36 to 0.52	1	3
[16]	Artix-7	800	0 - 3.0	N/A	3.9	N/A	-0.5 to 0.6	-0.2 to 0.5	1	3

[1] The authors present a 1.2 GSample/s ADC in their paper. However, this sampling rate is achieved by interleaving three 400 MSample/s ADCs.

The ENOB can be calculated from the SNDR with Equation 7.

$$ENOB = \frac{SNDR - 1.76}{6.02} \tag{7}$$

All measured SNR, SNDR and the corresponding ENOB values are listed in Table 3.

Table 3: SNR, SNDR, and ENOB measured at different input frequencies and sampling rates.

Sampling rate MSample/s	Input frequency Hz	SNR dB	SNDR dB	ENOB bit
600	10 k	44.3	43.9	7.0
600	30 M	41.4	36.2	5.7
1200	10 k	34.9	33.7	5.3
1200	30 M	25.3	25.0	3.9

4.7 ERBW

The effective resolution bandwidth (ERBW) is the input frequency at which the SNDR has dropped by 3 dB or the ENOB is reduced by 0.5 bit. Figure 16 shows the ENOB of the ADC in dependence of the input signal frequency.

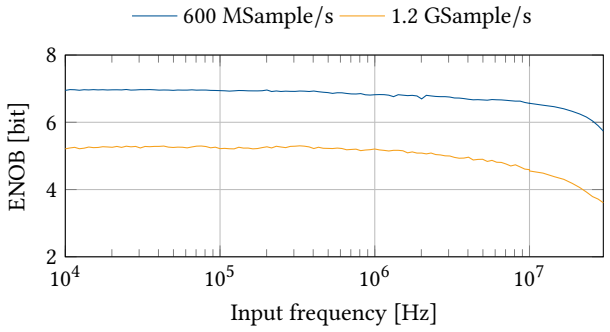


Figure 16: ENOB in dependence of the input frequency. Sines with different frequencies have been sampled with 600 MSample/s and 1.2 GSample/s and the corresponding ENOB has been calculated.

The ERBW is 14 MHz for the 600 MSample/s version and 6 MHz for the 1.2 GSample/s ADC.

4.8 Comparison

Table 2 compares our ADC with other ADCs based on similar working principles. Homulle et al. [5] presented a 400 MSample/s ADC that achieved an ENOB of 6. Xiang et al. [16] increased the sampling rate to 800 MSample/s while achieving 3.9 ENOB.

The ADC presented in this paper achieves a higher sampling rate than the two ADCs from [5] and [16] while also achieving a higher ENOB. In addition, our implementation needs no external resistor to create the slope for the LVDS comparator. Note that our implementation benefits from the newer FPGA technology compared to the FPGAs used in other publications.

5 CONCLUSION

We presented an FPGA-based ADC, which is fully reconfigurable and does not require any external elements. The slope ADC creates the required slope inside the FPGA and compares it to an analog input signal with an LVDS comparator. A tapped-delay line TDC is used to measure the time from the start of the slope until the slope crosses the input signal level. Different linearization and correction techniques are implemented to increase the precision of the TDC. The final ADC is able to operate at 600 MSample/s or 1.2 GSample/s.

For 600 MSample/s, the ADC achieves a resolution of 9.3 bit, an ENOB of 7, a DNL of -0.9 to 0.9 bit, and an INL of -1.1 to 0.9 bit. A resolution of 5.3 bit, an ENOB of 5.3, a DNL of -1 to 1.6 bit, and an INL of -1.1 to 1.2 bit is accomplished at 1.2 GSample/s.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their accurate reviews and valuable comments.

REFERENCES

[1] Claudio Favi and Edoardo Charbon. 2009. A 17ps Time-to-Digital Converter Implemented in 65nm FPGA Technology. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays* (Monterey, California, USA) *(FPGA '09)*. Association for Computing Machinery, New York, NY, USA, 113–120. https://doi.org/10.1145/1508128.1508145

[2] Mark D. Fries and John J. Williams. 2002. High-precision TDC in an FPGA using a 192 MHz quadrature clock. In *2002 IEEE Nuclear Science Symposium Conference Record*, Vol. 1. 580–584 vol.1. https://doi.org/10.1109/NSSMIC.2002.1239380

[3] Stanisław Grzelak, Marcin Kowalski, Jarosław Czoków, and Marek Zieliński. 2014. High Resolution Time-Interval Measurement Systems Applied to Flow Measurement. online. *Metrology and Measurement Systems* No 1 (2014), 77–84. https://doi.org/10.2478/mms-2014-0008

[4] Harald Homulle, Francesco Regazzoni, and Edoardo Charbon. 2015. 200 MS/s ADC Implemented in a FPGA Employing TDCs. In *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (Monterey, California, USA) *(FPGA '15)*. Association for Computing Machinery, New York, NY, USA, 228–235. https://doi.org/10.1145/2684746.2689070

[5] Harald Homulle, Stefan Visser, and Edoardo Charbon. 2016. A Cryogenic 1 GSa/s, Soft-Core FPGA ADC for Quantum Computing Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers* 63, 11 (Nov 2016), 1854–1865. https://doi.org/10.1109/TCSI.2016.2599927

[6] Rui Machado, Jorge Cabral, and Filipe S. Alves. 2019. Recent Developments and Challenges in FPGA-Based Time-to-Digital Converters. *IEEE Transactions on Instrumentation and Measurement* 68, 11 (Nov 2019), 4205–4221. https://doi.org/10.1109/TIM.2019.2938436

[7] Marek Pałka, Paweł Moskal, Tomasz Bednarski, Piotr Białas, Eryk Czerwiński, Łukasz Kapłon, Andrzej Kochanowski, Grzegorz Korcyl, Jakub Kowal, Paweł Kowalski, Tomasz Kozik, Wojciech Krzemień, Marcin Molenda, Szymon Niedźwiecki, Monika Pawlik, Lech Razyński, Zbigniew Rudy, Piotr Salabura, Neha Gupta-Sharma, Michał Silarski, Artur Słomski, Jerzy Smyrski, Adam Strzelecki, Wojciech Wiślicki, Marcin Zieliński, and Natalia Zoń. 2014. A novel method based solely on field programmable gate array (FPGA) units enabling measurement of time and charge of analog signals in positron emission tomography (PET). *Bio-Algorithms and Med-Systems* 10, 1 (2014), 41 – 45. https://doi.org/10.1515/bams-2013-0104

[8] Qi Shen, Shubin Liu, Binxiang Qi, Qi An, Shengkai Liao, Chengzhi Peng, and Weiyue Liu. 2014. A multi-chain measurements averaging TDC implemented in a 40 nm FPGA. In *2014 19th IEEE-NPSS Real Time Conference*. 1–3. https://doi.org/10.1109/RTC.2014.7097550

[9] Fabio Sousa, Volker Mauer, Neimar Duarte, Ricarado P. Jasinski, and Volnei A. Pedroni. 2004. Taking advantage of LVDS input buffers to implement sigma-delta A/D converters in FPGAs. In *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, Vol. 1. I–1088. https://doi.org/10.1109/ISCAS.2004.1328388

[10] Ryszard Szplet, Dominik Sondej, and Grzegorz Grzeda. 2014. Subpicosecond-resolution time-to-digital converter with multi-edge coding in independent coding lines. In *2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*. 747–751. https://doi.org/10.1109/I2MTC.2014.6860842

[11] Stefan Visser. 2015. *A 1 GSa/s deep cryogenic, reconfigurable soft-core FPGA ADC for quantum computing applications*. Master's thesis. Delft University of Technology. http://resolver.tudelft.nl/uuid:04fadfcd-fb54-47c3-b564-661aefa40dda

[12] Yonggang Wang and Chong Liu. 2016. A 4.2 ps Time-Interval RMS Resolution Time-to-Digital Converter Using a Bin Decimation Method in an UltraScale FPGA. *IEEE Transactions on Nuclear Science* 63, 5 (Oct 2016), 2632–2638. https://doi.org/10.1109/TNS.2016.2606627

[13] Jinyuan Wu. 2010. Several Key Issues on Implementing Delay Line Based TDCs Using FPGAs. *IEEE Transactions on Nuclear Science* 57, 3 (June 2010), 1543–1548. https://doi.org/10.1109/TNS.2010.2045901

[14] Jinyuan Wu, Sten Hansen, and Zonghan Shi. 2007. ADC and TDC implemented using FPGA. In *2007 IEEE Nuclear Science Symposium Conference Record*, Vol. 1. 281–286. https://doi.org/10.1109/NSSMIC.2007.4436331

[15] Jinyuan Wu and Zonghan Shi. 2008. The 10-ps wave union TDC: Improving FPGA TDC resolution beyond its cell delay. In *2008 IEEE Nuclear Science Symposium Conference Record*. 3440–3446. https://doi.org/10.1109/NSSMIC.2008.4775079

[16] Zikun Xiang, Tianqi Wang, Tong Geng, Tian Xiang, Xi Jin, and Martin Herbordt. 2018. Soft-Core. Multiple-Lane, FPGA-based ADCs for a Liquid Helium Environment. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*. 1–6. https://doi.org/10.1109/HPEC.2018.8547550

[17] Axel Zimmermann, Endric Schubert, and Christian Grumbein. 2010. Integration of Analog-Digital and Digital-Analog Converters into FPGA-Based Microcontrollers. (2010). https://www.missinglinkelectronics.com/files/papers/MLE-Altera-DuE-Entwicklerforum2011.pdf