Exponential Sine Sweep Measurement Implementation Targeting FPGA Platforms

Alexander Klemd[†], Patrick Nowak[*], Piero Rivera Benois[‡], Etienne Gerat[*], Udo Zölzer[*], Bernd Klauer[†]
[*]Dept. of Signal Processing and Communications, Helmut-Schmidt University, 22043 Hamburg, Germany
[†]Dept. of Computer Engineering, Helmut-Schmidt University, 22043 Hamburg, Germany
[‡]Signal Processing Group, University of Oldenburg, 26129 Oldenburg, Germany
Email: alexander.klemd@hsu-hh.de

***Abstract*—In this paper a field programmable gate array (FPGA) is considered as a digital signal processing platform for the implementation of an exponential sine sweep measurement algorithm. Aiming at minimizing the required computational resources, two strategies are proposed. Firstly, an oscillator implemented with the coordinate rotation digital computer (CORDIC) algorithm is used to generate the exponential sine sweep. Secondly, only the calculations are performed that lead to the linear impulse response of the system for a desired length. Furthermore, aiming at minimizing the required memory resources, the measured impulse response is stored in the memory previously allocated to the recorded signal. In order to validate the proposed implementation, measurements of an acoustical system are performed using a platform that is equipped with an FPGA and a processor. In this way, the results achieved by the FPGA fixed-point implementation can be compared to reference results achieved using a floating-point MATLAB implementation running on the processor. This comparison corroborates the validity of the proposed implementation.**

I. Introduction

Measuring the impulse response of unknown systems is an essential ability in acoustics. Most control algorithms for active noise cancellation (ANC), need to know the impulse response of the secondary path from the secondary speakers to the error microphones. For high performance ANC control algorithms, specialized architectures for FPGA or ASIC platforms usually provide superior performance over software architectures for digital signal processors (DSP) and other processors. Additionally there are several advantages to perform the control algorithm and the system identification on the same FPGA platform. Firstly, the transfer of the impulse response from the system identification component to the control algorithm can be designed more elegantly and faster, since the data is not leaving the FPGA chip. Secondly, since the peripherals are part of the secondary path, the impulse response measurement can be improved when it is conducted on the same platform as the control algorithm. Thirdly, the FPGA platform provides a hard real-time system with a constant sample delay. Thus, compared to a general-purpose computer running an operating system, no adjustments of the algorithm have to be made to compensate for not deterministic delays.
There are different approaches on how to obtain the impulse response of an unknown system. State-of-the-art methods use an exponential sine sweep to excite the system [1] [2].
The algorithm is expected to be very memory-intensive, because a convolution has to be performed with the time-mirrored sine sweep samples that were used to excite the system and the corresponding system response. To reduce the demand for block-memory resources, the proposed architecture generates the regular sine sweep and the time-mirrored sine sweeps on the fly. Thus only the system response has to be stored to perform the convolution. The architecture uses fixed-point formats of varying precision, which is more resource efficient than the floating-point number format. To save more block-memory, the incoming result of the impulse response is stored in the block-memory cells of the system response that are not needed by the convolution anymore. Hence the impulse response does not require additional memory.

II. System Identification

In discrete-time, the first K elements of the impulse response of a causal system $h(n)$ are described by

$$\hat{h}(n) = \sum_{k=0}^{K-1} h(k) \cdot \delta(n-k), \tag{1}$$

where $\delta(n)$ denotes the unit impulse. State-of-the-art methods use an exponential sine sweep instead of a unit impulse $\delta(n)$ to calculate $\hat{h}(n)$ [1]. The excitation signal used by the exponential sine sweep method can be mathematically described as

$$x(n) = A \cdot \sin\left(\frac{\Omega_{\text{start}} \cdot (L_x - 1)}{\ln(\frac{\Omega_{\text{end}}}{\Omega_{\text{start}}})} \cdot \left(e^{\frac{n}{L_x - 1} \ln\left(\frac{\Omega_{\text{end}}}{\Omega_{\text{start}}}\right)} - 1 \right) \right), \tag{2}$$

where A is the chosen amplitude, L_x is the desired signal's length in samples, $\Omega_{\text{start}} = 2\pi f_{\text{start}}/f_{\text{s}}$ and $\Omega_{\text{end}} = 2\pi f_{\text{end}}/f_{\text{s}}$ are the starting and ending angular frequencies, and f_{s} is the sampling frequency. The instantaneous angular frequency is the derivative of the argument of the sine function over time, which is given by

$$\Omega(n) = \Omega_{\text{start}} \cdot e^{\frac{n}{L_x - 1} \ln\left(\frac{\Omega_{\text{end}}}{\Omega_{\text{start}}}\right)}. \tag{3}$$

This equation can be formulated as

$$\Omega(n) = \Omega_{\text{start}} \cdot \alpha^n, \tag{4}$$

where

$$\alpha = \left(\frac{\Omega_{\text{end}}}{\Omega_{\text{start}}} \right)^{\frac{1}{L_x - 1}} \tag{5}$$

is the factor by which the instantaneous angular frequency increases with each sample time n. A companion time-series $x_{\text{inv}}(n)$ is generated, such that the convolution between both results in a scaled unit impulse

$$\sum_{k=0}^{L_x-1} x(k)\cdot x_{\text{inv}}(n-k) = C\cdot\hat{\delta}(n-L_x-1), \tag{6}$$

which is time-shifted by L_x-1 samples and band-limited by f_{start} and f_{end}. The scalar C can be calculated as

$$C = \frac{A^2\pi L_x\cdot\left(\frac{\Omega_{\text{start}}}{\Omega_{\text{end}}}-1\right)}{2(\Omega_{\text{end}}-\Omega_{\text{start}})\ln\left(\frac{\Omega_{\text{start}}}{\Omega_{\text{end}}}\right)}, \tag{7}$$

following the parameters used for generating $x(n)$ [2]. The companion time-series $x_{\text{inv}}(n)$ is achieved by applying a time-dependent exponentially-decaying amplitude correction factor given by

$$A_{\text{inv}}(n) = \alpha^{-n}, \tag{8}$$

to the time-mirrored sine sweep $x(n)$ as

$$x_{\text{inv}}(n) = A_{\text{inv}}(n)\cdot x(L_x-1-n). \tag{9}$$

All in all, the procedure using this measurement technique complies the following steps: Firstly, the generation and storage of the signals $x(n)$ and $x_{\text{inv}}(n)$ by Eq. (2), Eq. (8) and Eq. (9); Secondly, the excitation of the system by means of $x(n)$ and the recording of the system's response

$$y(n) = \sum_{k=0}^{L_x-1} h(k)\cdot x(n-k), \tag{10}$$

where $n\in\{0,...,L_h+L_x-2\}$ and L_h is the time required by the system of $h(n)$ to settle down, which can be chosen as the estimated T_{60} time in samples. Thirdly, the calculation of the convolution between $y(n)$ and $x_{\text{inv}}(n)$ as

$$\tilde{h}(n) = \sum_{k=0}^{L_x+L_h-2} y(k)\cdot x_{\text{inv}}(n-k) \tag{11}$$

where $n\in\{0,...,2\cdot L_x+L_h-3\}$ and the storage of the result; Fourthly and final, the extraction of the desired impulse response by

$$\hat{h}(n) = \frac{1}{C}\cdot\tilde{h}(n+L_x-1) \tag{12}$$

where $n\in\{0,\dots,L_h-1\}$, which discards the first L_x-1 samples, keeps the desired length of the impulse response L_h, and divides these samples by the correlation factor in Eq. (7).

III. Oscillator

To generate the sine sweep signal without using too much memory, a recursive oscillator has been used. A simple way to create an oscillator is to use a resonant filter and set it in a quasi-stable state to generate a sinusoid. The cutoff frequency f_{c} of the filter is then used to control the oscillator's frequency.

Since the frequency is aimed to change over time, a filter with good stability for time-varying parameters is required. The Gold and Rader filter is a coupled filter structure that fulfills this requirement [3][4]. The Gold and Rader filter is defined by the difference equations

$$s_1(n) = rb\cdot s_1(n-1)+ra\cdot s_2(n-1), \tag{13}$$

$$s_2(n) = rb\cdot s_2(n-1)-ra\cdot s_1(n-1)+u(n-1), \tag{14}$$

where a and b are the filter coefficients controlling the cutoff frequency Ω_{c} and the resonance. They are defined as $a=\sin(\Omega_{\text{c}})$, $b=\cos(\Omega_{\text{c}})$, where $\Omega_{\text{c}}=2\pi\cdot f_{\text{c}}/f_{\text{s}}$ and r the poles radius. The filter is stable when $r<1$.

The oscillator is based on the filter structure, but the stability criterion is yet set to quasi-stability by setting $r=1$ [5]. Additionally, the input $u(n)$ is removed. The oscillation of the system will be started by initializing the states $s_1(0)$ and $s_2(0)$. The difference equations are now given as

$$s_1(n) = b\cdot s_1(n-1)+a\cdot s_2(n-1), \tag{15}$$

$$s_2(n) = b\cdot s_2(n-1)-a\cdot s_1(n-1). \tag{16}$$

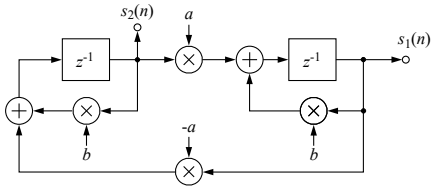


Fig. 1. Block diagram of the Gold and Rader oscillator.

In quasi-stable mode the Gold and Rader filter generates two quadrature sinusoids (shifted by 90°) $s_1(n)$ and $s_2(n)$ shown in the block diagram in Fig. 1. The initialization of the states $s_1(0)$ and $s_2(0)$ defines the starting phase ϕ_0 of the oscillator

$$s_1(0) = \sin(\phi_0), \tag{17}$$

$$s_2(0) = \cos(\phi_0). \tag{18}$$

Since two signals are generated, one of them can be selected as sine sweep. In this case, $s_1(n)$ is used. The control of the rotation frequency Ω is then done by setting the coefficients a and b, in a similar way than to calculate the cutoff frequency of the filter

$$a = \sin(\Omega), \tag{19}$$

$$b = \cos(\Omega). \tag{20}$$

The coefficients a and b are updated at every iteration following the sweeping-up frequency geometric suite $\Omega_{\text{up}}(n)$ defined by

$$\Omega_{\text{up}}(n+1) = \alpha\cdot\Omega_{\text{up}}(n), \tag{21}$$

with α being the ratio defined in Eq. (5).

The precise control over the starting phase of the oscillator is very handy in this application, because both a sine sweep $x(n)$ and the corresponding inverse sine sweep $x_{\text{inv}}(n)$ need to be generated that have to be perfect inverses of each other without being stored in memory. For that matter, as the first sine sweep is recursively generated until the last sample L_x-1. One extra iteration has to be performed such that

the states $s_1(L_x)$ and $s_2(L_x)$ are stored and will be reused to generate the inverse sweep $s_{1,\text{inv}}(n)$. The inverse sweep can be seen as the quadrature oscillator rotating in the other direction. In order to simulate this for the signal $s_{1,\text{inv}}(n)$ the initial states $s_{1,\text{inv}}(0)$ and $s_{2,\text{inv}}(0)$ are set as

$$s_{1,\text{inv}}(0) = s_1(L_x), \tag{22}$$

$$s_{2,\text{inv}}(0) = -s_2(L_x). \tag{23}$$

The coefficients a and b are updated at every iteration following the sweeping-down frequency geometric suite $\Omega_{\text{down}}(n)$ defined by

$$\Omega_{\text{down}}(n+1) = \frac{\Omega_{\text{down}}(n)}{\alpha}, \tag{24}$$

with α being the ratio defined in Eq. (5).

IV. Proposed Architecture

The here proposed architecture is implemented in VHDL and uses the fixed-point number format with custom precisions per signal. The fixed-point format needs careful consideration about the necessary number range and precision of each signal, but is more hardware-efficient compared to the floating-point format. VHDL was chosen to increase the flexibility of the design. The design is platform- and vendor-independent and makes extensive use of parameters that are evaluated at synthesis time called *generics*. Among other parameters the sampling rate, sweep duration, starting and ending frequency of the sweep, length of the system to be identified, amplitude fade length, number of CORDIC-iterations and the bitwidths of the signals from Sec. II can be set without code modifications. The fixed-point format for other internal signals are derived from these signals.

A. Top-level architecture

The datapath of the top-level architecture is shown in Fig. 2 and can be divided into three main components which perform the sine sweep generation, the convolution and the amplitude fade of the sine sweep signal. To identify a system, the control

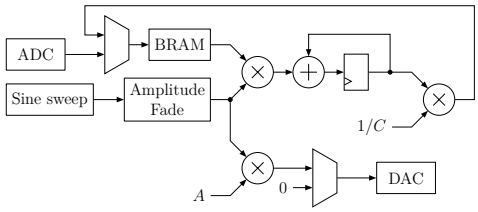


Fig. 2. Datapath of the top-level architecture.

logic first triggers the periodic generation of the sine sweep from f_{start} to f_{end} with the given sampling rate. That signal is processed by the component that performs the amplitude fade and sent to the DAC. Synchronously to the DAC, the ADC is triggered to sample the system response. After L_x samples, the control logic stops the excitation but records L_h more samples. After the completion of this sweep, the control logic sets the state of the sine sweep generator to reverse the sweep with the maximum sampling rate possible. For the convolution of the reversed sweep with the system response, the block-RAM (BRAM) is traversed in the opposite direction by the control logic in sync with the sine sweep generator. A multiply-accumulate unit (MAC) sequentially calculates the partial results of the convolution. According to Eq. (12), each result of the convolution has to be divided with the scalar C. This scalar is constant during runtime, so the division is implemented as a multiplication with the reciprocal value. When the first partial result of the convolution is accumulated in the MAC and divided by C, the control logic stores the result in the block-RAM and thus overwrites the first sample of the system response. As the reversed system response is shifted to the right for each partial result of the convolution, more samples stop to overlap with the sine sweep and thus do not contribute to the result. When the partial result of the convolution reaches the desired length of the system to be identified, the architecture signals its readiness and returns to the idle state. The second port from the block-RAM can be used by external entities to read the result.

B. Sine sweep generation

The hardware-implementation of the Gold and Rader oscillator requires the use of trigonometric functions. Among hardware-efficient algorithms for trigonometric functions, the CORDIC algorithm is a simple and fast solution. A good survey about the CORDIC equations and the processor architecture with a focus on FPGA implementations can be found in [6]. Besides of being capable to calculate the sine and cosine function, the CORDIC processor in rotational mode is also able to rotate the coordinate (x_0,y_0) by the coordinate transform

$$x_M = A_M(x_0\cos z_0 - y_0\sin z_0) \tag{25}$$

$$y_M = A_M(y_0\cos z_0 + x_0\sin z_0) \tag{26}$$

$$z_M = 0 \tag{27}$$

$$A_M = 1/\prod_{m=1}^{M}\sqrt{1+2^{-2m}} \tag{28}$$

in M iterative steps, where z_0 is the rotation angle. With the parameters of the oscillator put into the CORDIC processor

$$x_0 = s_2(n)/A_M, \tag{29}$$

$$y_0 = s_1(n)/A_M, \tag{30}$$

$$z_0 = \omega(n), \tag{31}$$

the processor output (y_M, x_M) returns the new states of the oscillator $s_1(n+1)$ and $s_2(n+1)$ and thus can be used to implement the Gold and Rader oscillator itself.

The coordinate transform of the CORDIC processor is limited to rotation angles z_0 between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ [6]. Since $\Omega(n)$ can range between 0 and π, the CORDIC algorithm needs additional logic to shift the number range of the rotation angle accordingly. Consequently $\Omega(n)$ uses an unsigned fixed-point ranging from $[0,\pi[$ and z uses a signed fixed-point format of the same bitwidth ranging from $\left[-\frac{\pi}{2},\frac{\pi}{2}\right[$. Both

parameters use a modulo-2π representation. A value from the Ω-register is fed into z_0-register without any transformation. Due to the different interpretation of the most significant bit (MSB), this effectively rotates any Ω-value larger than $\frac{\pi}{2}$ by π and thus within the valid range of z_0. To compensate for this transformation, the initial coordinates x_0 and y_0 are multiplied with $-A_M$ instead of A_M. The multiplications with A_M are necessary either way, so additional hardware is only required for the multiplexer and the constant $-A_M$.
The datapath of the sine sweep architecture is shown in Fig. 3. To begin the initial sweep with increasing frequency,

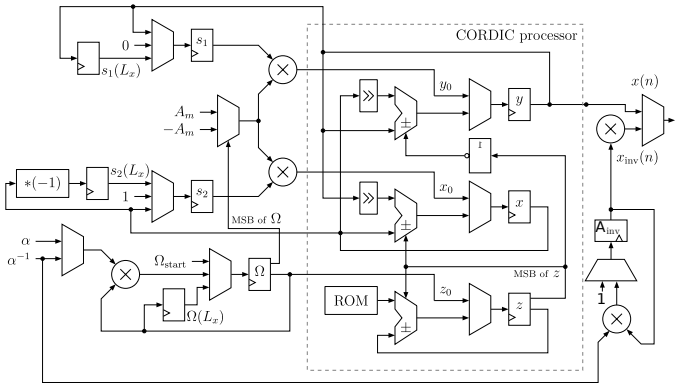


Fig. 3. Datapath for the generation of the sine sweep using an iterative CORDIC processor.

the registers of the CORDIC processor need to be intialized with the initial state of the oscillator according to Eq. (17) and (18) using a starting phase of $\phi_0 = 0$. After M cycles of the CORDIC processor, the state of the oscillator $s_1(n)$ and $s_2(n)$ is updated and the succeeding angular frequency $\Omega(n)$ is calculated. After L_x samples are generated, one additional CORDIC-iteration is started and the state is stored in $s_{1/2}(L_x)$. The initial rotation frequency is stored in the register $\Omega(L_x)$. The generation of the repeated, reversed sweep runs in an analogous manner. One difference is the concluding multiplication of the CORDIC output with the amplitude correction factor $A_{\text{inv}}(n)$ to obtain the reversed excitation signal $x_{\text{inv}}(n)$.

C. Amplitude fade

An amplitude fade of the excitation signal is necessary to eliminate unwanted ripple effects in the frequency response near the starting and ending frequencies of the sweep [2]. Thus, the first and the last N_{fade} samples of the sweep with the length L_x get attenuated. The amplitude fade component multiplies the excitation signal with

$$A_{\text{fade}}(n) = 1 - \beta(n) \tag{32}$$

where $\beta(n)$ is a recursive factor defined as

$$\beta(n) = \begin{cases} 1 & \text{if } n = 0 \\ \beta(n-1)2^{-12/N_{\text{fade}}} & \text{if } 0 < n \leq N_{\text{fade}} \\ \beta(n-1)2^{12/N_{\text{fade}}} & \text{if } L_x - N_{\text{fade}} \leq n < L_x \\ \beta(n-1) & \text{otherwise.} \end{cases} \tag{33}$$

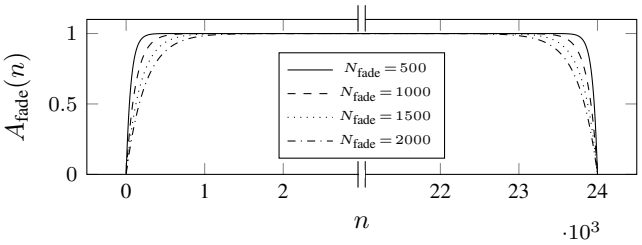


Fig. 4. The factor $A_{\text{fade}}(n)$ for the amplitude fade is plotted for a selection of N_{fade} over the first and last 3000 samples. The total sweep length in samples is $L_x = 24001$.

This algorithm was chosen, because it is more hardware efficient compared to common algorithms, such as a quadratic equation. The architecture is plain and consists of two sequential multiplications, one substraction and a register for $\beta(n)$. An exemplary progression of $A_{\text{fade}}(n)$ over 24001 samples with different N_{fade} values can be seen in Fig. 4.

V. Evaluation

In order to evaluate the proposed architecture, the dSpace MicroLabBox is used which contains both an FPGA and a processor. It contains a *Freescale QorlQ P5020* CPU with 2 cores up to 2 GHz, a *Xilinx Kintex-7 XC7K325T* FPGA clocked at 100 MHz, a 16 bit ADC and DAC. In this way, in addition to the proposed architecture on the FPGA, also a reference measurement can be implemented on the processor of the same prototyping hardware to obtain more comparable results. In this reference measurement, the exponential sine sweep is generated using a sample-based floating-point MATLAB implementation of Eq. (2). After exciting the system under test, the recorded sine sweep is stored by the processor and exported into MATLAB, where the impulse response of the system under test is calculated according to Eq. (11). Thus, both the generation of the exponential sine sweep and the convolution of the recorded signal with the inverse sweep can be evaluated using the same measurement setup.

For validation purposes, the parameters of configuration 1 are chosen as $L_x = 24001$, $f_{\text{s}} = 48\,\text{kHz}$, $f_{\text{start}} = 20\,\text{Hz}$, $f_{\text{end}} = 20\text{kHz}$ and $L_h = 1024$. According to

$$T_x = \frac{L_x - 1}{f_{\text{s}}}, \tag{34}$$

this results in a sweep duration of $T_x = 0.5\,\text{s}$. This sweep length is chosen as it represents the limit that the platform's processor can process in real-time. The exponential sine sweeps are faded-in and faded-out for $N_{\text{fade}} = 500$ samples. The hardware implementation requires additional parameters to set the precision of the different fixed-point formats and to set the number of iterations of the CORDIC component. The number of iterations of the CORDIC processor is set to 12. Together with a bitwidth of 24 bit for the CORDIC states $s_{1/2}(n)$ and 32 bit for $\Omega(n)$, these parameters provide a relative error of less than 1% for the sine sweep component compared to the floating-point reference over the sweep length in configuration 1. The bitwidth of the excitation signal and

TABLE I
HARDWARE RESOURCES PRESENT ON THE *Xilinx Kintex-7 XC7K325T* FPGA AND THE USED HARDWARE ELEMENTS FOR THE PRESENTED CONFIGURATIONS.

Config.	CLB-Slices Used	(%)	BRAM Tiles Used	(%)	DSP-Slices Used	(%)
Config. 1	4530	(8.89)	120	(26.96)	36	(4.28)
Config. 2	5007	(9.82)	368	(82.69)	36	(4.28)
Total	50950		445		840	

system response is 16 bit due to the underlying platform. The bitwidth of $A_{\text{fade}}(n)$ is set to 24 bit and the bitwidth of the coefficients of the impulse response $\hat{h}(n)$ is set to 32 bit.

A. Hardware usage

As all $L_x + L_h$ samples of the system response need to be stored in block-RAM, the amount of memory becomes quickly a limiting resource. Thus configuration 2 differs from configuration 1 only in the increased sweep duration to 10 seconds. Above a sweep length of 10 seconds, the synthesis time increases exponentially which makes it difficult to determine the exact maximum sweep length for this FPGA that the synthesis tool is still able to place and route. When longer sweep durations are required, it might be an option to reduce the bitwidths of the system response and impulse response. Naturally this decreases the quality of the measured impulse response. The synthesis results of configurations 1 and 2 are listed in Tab. I. The synthesis results show that an increase in sweep length almost solely effects the BRAM usage. Besides that, scaling the sampling rate f_s or the length of the impulse response L_h, which is very small compared to L_x, does not show a significant effect on hardware usage. Thus these configurations are not listed in Tab. I. The maximum sampling rate is rather determined by the time the CORDIC processor requires to calculate each sample. With the configurations above, the sine sweep component takes 15 clocks to initialize, operate the CORDIC-processor for 12 iterations and to resize fixed-point format. The FPGA operates at a frequency of $f_\text{clk} = 100\text{MHz}$, which results in a maximum sampling rate of $f_\text{s} = 6.67\text{MHz}$.

B. Generated Exponential Sine Sweeps

In a first step, the accuracy of the oscillator implementation is evaluated by comparing the generated exponential sine sweeps on the FPGA and the processor. Here, parameter configuration 1 is used for the exponential sine sweeps. The short duration of the exponential sine sweeps results from real-time processing issues on the processor.

Figure5 illustrates the beginning ($t \leq 0.1\text{s}$) of the generated exponential sine sweeps $x_\text{Proc}(n)$ and $x_\text{FPGA}(n)$. From this, mainly a phase-shift between the two exponential sine sweeps is visible. In order to further investigate the differences, the magnitude spectra $|X_\text{Proc}(f)|_\text{dB}$ and $|X_\text{FPGA}(f)|_\text{dB}$ of the entire exponential sine sweeps are shown in Fig.6. Additionally, the difference between the two magnitude spectra

$$\Delta X_\text{dB}(f) = |X_\text{Proc}(f)|_\text{dB} - |X_\text{FPGA}(f)|_\text{dB} \tag{35}$$

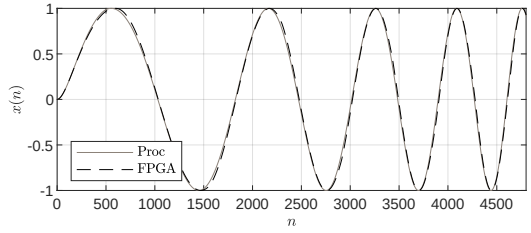


Fig. 5. Comparison of the generated exponential sine sweeps on the processor and the FPGA for the first 0.1 seconds, with $T_x = 0.5$ s, $f_\text{s} = 48\text{kHz}$, $f_\text{start} = 20\text{Hz}$, $f_\text{end} = 20\text{kHz}$, and $N_\text{fade} = 500$ samples.

is represented on a secondary y-axis in the same figure. As can be seen, both generated exponential sine sweeps show similar magnitude spectra with a difference that does not exceed $\Delta X_\text{dB}(f) < 0.25\,\text{dB}$ in the main part of the exponential sine sweep. However, also a bias of roughly $0.03\,\text{dB}$ can be seen in $\Delta X_\text{dB}(f)$ for frequencies between approximately 2kHz and 18kHz. Although small differences can be seen between the generated exponential sine sweeps, in the following, these signals are used to measure the impulse responses of real systems in order to evaluate whether the differences influence the measurement results.

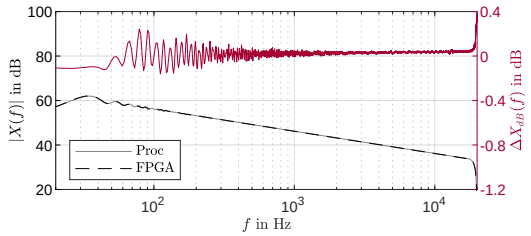


Fig. 6. Comparison of the magnitude spectra of the generated exponential sine sweeps from Fig. 5. Additionally, the difference between them is plotted at the y-axis on the right.

C. Measurements

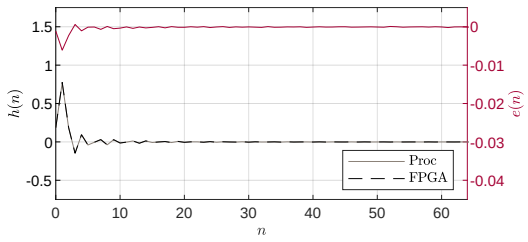
In order to evaluate the whole FPGA implementation, the impulse responses of two different systems are measured. For comparing the measurement results, two different error metrics

$$e(n) = h_\text{Proc}(n) - h_\text{FPGA}(n), \tag{36}$$

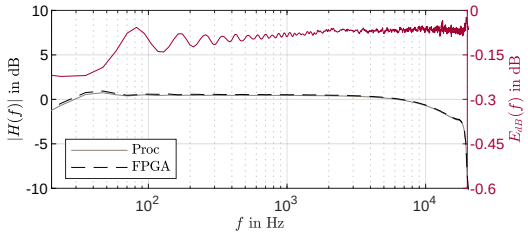
$$E_\text{dB}(f) = |H_\text{Proc}(f)|_\text{dB} - |H_\text{FPGA}(f)|_\text{dB}, \tag{37}$$

are defined in time- and frequency-domain, respectively, where $h_\text{Proc}(n)$ indicates the impulse response measured by the processor, $h_\text{FPGA}(n)$ the impulse response measured on the FPGA, and $|H_\text{Proc}(f)|_\text{dB}$ and $|H_\text{FPGA}(f)|_\text{dB}$ the corresponding magnitude responses.

As a first system, a simple input-output loop is considered by connecting an output and an input of the dSpace MicroLabBox. In this way, the system consists of a DAC, a wire connection, and an ADC. The measurement results can be seen in Fig.7. As can be seen in Fig. 7(a) both impulse responses $h_\text{Proc}(n)$ and $h_\text{FPGA}(n)$ show similar characteristics including a



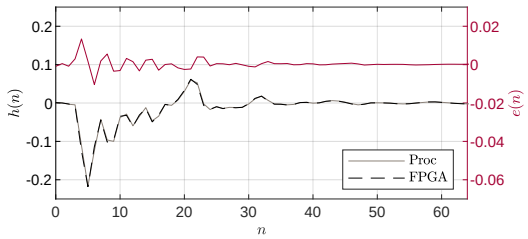
(a) Impulse response



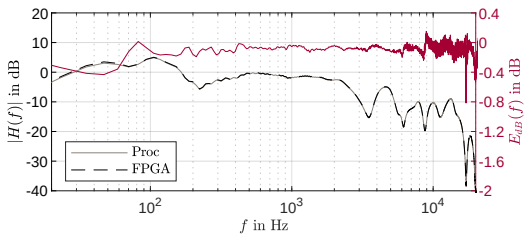
(b) Magnitude response

Fig. 7. Comparison of the measured (a) impulse and (b) magnitude responses of the input-output loop on the processor and the FPGA. Additionally, the difference between them is plotted at the y-axes on the right.

one sample delay, which arises from the simultaneous triggering of DAC and ADC. In addition to the measured impulse and magnitude responses of the system, also the two error metrics given in Eq. (36) and Eq. (37) are represented on secondary y-axes in the corresponding plots. As can be seen, both measurements show similar impulse and magnitude responses with only small deviations in amplitude ($|e(n)| \leq 0.006$) and magnitude ($|E_{\mathrm{dB}}(f)| \leq 0.22\mathrm{dB}$), respectively.



(a) Impulse response



(b) Magnitude response

Fig. 8. Comparison of the measured (a) impulse and (b) magnitude responses of the acoustical path on the processor and the FPGA. Additionally, the difference between them is plotted at the y-axis on the right.

As a second system, an acoustical path between a headphone's loudspeaker and a microphone placed inside the ear cup of the headphone is measured. Thus, the acoustical system contains the DAC of the dSpace MicroLabBox, a headphone pre-amplifier, a headphone's loudspeaker, a microphone, a microphone amplifier, and the ADC of the dSpace MicroLabBox. The measured impulse and magnitude responses are shown in Fig. 8. In addition to the measured impulse and magnitude responses, also the error metrics from Eq. (36) and Eq. (37) are represented on the secondary y-axes at the right. Similar as for the results of the input-output loop in Fig.7, only small differences in amplitude and magnitude are visible between the two measurements. The highest absolute difference in magnitude ($|E_{\mathrm{dB}}(f)| \approx 0.8\,\mathrm{dB}$) can be seen at roughly 17.1kHz, where a notch inside the acoustical path reduces the SNR of the measurement. Furthermore, frequencies close to the starting and ending frequencies of the exponential sine sweeps show higher differences in the magnitudes due to the influence of the amplitude fade. For the other frequencies, the error in magnitude does not exceed $|E_{\mathrm{dB}}(f)| \leq 0.24\mathrm{dB}$.

VI. Conclusions

In the present work an exponential sine sweep measurement targeting FPGA platforms is proposed. The implementation is based on two strategies that minimize the required computational resources. Firstly, the regular and inverse exponential sine sweeps are generated using an oscillator implemented via the CORDIC algorithm. Secondly, only the part of the convolution of the recorded signal with the inverse sweep is calculated that leads to the linear impulse response of the system for a desired length. Furthermore, by storing the measured impulse response in the memory previously allocated to the recorded signal, the memory requirements can be reduced, too. In order to evaluate the proposed implementation, impulse response measurements are performed on an FPGA and a processor. The FPGA implementation shows similar results compared to the reference measurement on the processor of the same prototyping hardware, corroborating the validity of the proposed implementation. When evaluating the hardware usage of the proposed implementation, it was observed that the memory consumption is the bottleneck of the proposed implementation. Future work can be focused on the optimization of the proposed implementation, such as allocating multiple MACs to accelerate the processing of the convolution.

References

[1] A. Farina, "Simultaneous measurement of impulse response and distortion with a swept-sine technique," in *Audio Engineering Society Convention 108*, February 2000.

[2] M. Holters, T. Corbach, and U. Zölzer, "Impulse response measurement techniques and their applicability in the real world," in *Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09)*, September 2009.

[3] C. Rader and B. Gold, "Effects of parameter quantization on the poles of a digital filter," *Proceedings of the IEEE*, vol. 55, no. 5, pp. 688–689, 1967.

[4] U. Zölzer, "Entwurf Digitaler Filter für die Anwendung im Tonstudiobereich," Ph.D. dissertation, Hamburg University of Technology, Hamburg, Germany, Juni 1989.

[5] J. Laroche, "Using resonant filters for the synthesis of time-varying sinusoids," *Journal of the Audio Engineering Society*, September 1998.

[6] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, 1998, pp. 191–200.