Acta Polytechnica **60**(6):462–468, 2020
DOI:10.14311/AP.2020.60.0462

available online at https://ojs.cvut.cz/ojs/index.php/ap

VERSATILE CHIRP SINE GENERATOR ON FIXED-POINT FPGA

Jan Kunz*, Petr Beneš

Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation, Technická 3082/12, 61600 Brno, Czech Republic

* corresponding author: xkunzj00@stud.feec.vutbr.cz

Abstract. This paper deals with a logarithmic and a linear chirp sine generation on a fixed-point FPGA mainly for vibration testing, nevertheless, the generator can also be used in other areas. A basic overview of the logarithmic chirp sine signal is provided. Then, methods of software signal generation as well as different hardware platforms are briefly described and their pros and cons are mentioned. A DDS generator on FPGA needs the phase difference between samples as an input. This generation for the logarithm chirp sine signal is presented, and its resolution, errors and limitations on fixed-point arithmetic are revealed. Our implementation runs on Compact RIO 9067, uses 32-bit fixed-point and is able to generate linear and logarithm chirp signals from 10 Hz to 7 kHz with a minimum chirp speed of 1 oct/min.

Keywords: Linear chirp sine, logarithm chirp sine, FPGA, fixed-point, generation.

1. Introduction

Sinusoidal signals and their variations are, due to their properties, commonly used in engineering [1]. Their usage varies from basic, such as an impedance measurement, a system identification and a vibration analysis, to more sophisticated ones, such as a motor control, a nuclear magnetic resonance or an electron paramagnetic resonance.

Sometimes, a chirp sine signal (CS), sinusoidal signal with a continuous frequency change, is used. The most common frequency changes are linear or logarithmic, however, the change can be described by other means. In some applications, for instance the impedance measurement, the knowledge of the actual chirp sine frequency is essential, whereas other application do not require it. Furthermore, in dwell vibration testing [2], it is sometimes necessary to change the chirp speed in dependence on the previous state, therefore, a small delay is necessary. The change of the chirp speed or the transition between the chirp sine and sine has to be done with a minimal phase noise.

Nowadays, linear and logarithmic chirp sine signals can be generated very easily using, for example, a direct digital synthesis, a voltage controlled oscillator, a look-up table or a phase generation [1, 3, 4]. However, these methods do not allow a fast change of a chirp speed together with the knowledge of the actual frequency at the same time.

This can be done by a software generator based, for instance, on a direct digital synthesis. However, a point-to-point generation, which allows a fast response time, needs fast computation. Furthermore, the algorithm is relatively simple, without many branches and conditions, therefore, it is more suitable for FPGAs than processors.

To allow the fast response time, a point-to-point generation is crucial, for this type of generation the FPGA is more convenient than processors. FPGAs are becoming more and more popular because they allow true parallelism and the computation power is also sufficient [5].

Unfortunately, most of this power is available on a fixed-point arithmetic, however, modern FPGAs also contain some floating-point cores. Therefore, these cores should be used wisely for precise computation. For this reason, we have decided to use the fixed-point arithmetic for the generator.

LabVIEW 2018 was selected as the platform for programming and the generator is executed on CompactRIO 9067, which contains Zynq xc7z020 equipped with ARM cortex-A9 processor and a medium-sized FPGA.

CompactRIO is a real-time embedded industrial controller, which allows precision timing, such as STC3, or TSN. Furthermore, it is compatible with more than 100 different I/O types varying from industrial communication, via digital and analog signal input/output, to specific sensor conditioning, for instance, charge output, IEPE, thermocouple, bridge, etc. This makes the Compact RIO a versatile tool for various applications [6]. This combination of computation power and various peripherals makes the platform ideal for development and fast prototyping as well as advanced control and monitoring.

2. Chirp sine signal

Chirp sine signal is a sine signal, whose frequency is changing with time. Linear chirp sine signal is used, for example, in radars. Specifically, a Frequency-Modulated Continuous-Wave (FMCW) with frequencies as high as possible is used because the higher frequencies, the better resolution. For this reason, the

FMCW signal for radars is often generated using the FPGA and the maximal frequency of the signal is tens of GHz [7]. The generation is explained in [8, 9].

Some other papers focus on FPGA versatile chirp and sine signal generators [8, 10], nevertheless the chirp signal is also only linear.

For this reason, this paper focuses on generation of logarithmic chirp sine.

The speed of the LCS is defined by a constant, which can be either *(dec/s)* or *(oct/s)*[1]. Because the sweep speed is usually low, the chirp constants are often defined in different time units, such as (min^{-1}) or even $(hour^{-1})$.

The chirp speed can also be defined by a start and a stop frequency and a duration of the sweep. From these, the speed of the logarithmic chirp signal can easily be calculated (eq. 1) and vice versa. In this paper, everything is demonstrated, for simplicity, on the same LCS signal with parameters $f_{start} = 10\ Hz$, $f_{stop} = 100\ Hz$, $k = 0.1\ dec/s$, so the duration is $t = 10\ s$.

$$k = \frac{log_{10}\left(\frac{f_{stop}}{f_{start}}\right)}{t} \tag{1}$$

where k (dec/s) is the chirp speed constant, f_{stop} (Hz) is the stop frequency, f_{start} (Hz) is the start frequency and t (s) is the sweep duration.

3. FPGA sine generators

There are several FPGA sine generators, some are described in the literature [8, 9, 11, 12] and some are commercially available, for instance, the NCO IP Core [13] from Intel (former Altera), or DDS [14] from Xilinx. Some of them put an emphasis on the generation speed or maximal frequency of the generated signal, whereas other on spectral purity and maximal resolution. Nevertheless, all generators use the phase difference between samples as an input, as shown on block diagram (fig. 1). Determining the phase difference for sine signal is easy, nevertheless, in the case of sweep sine signals, the phase difference changes for every sample. Moreover, the accuracy of the phase difference defines the quality of the sweep signal.

Some FPGA sine generator implementations [8, 9] use a look-up table as a source of the phase difference. Other implementations [11, 12] generate the phase for the linear chirp signal via integration. Nevertheless, neither of these methods can be effectively used for the logarithm chirp sine signal, as the phase difference changes non-linearly in dependence of the chirp speed and frequency range.

The calculation of the phase difference for the logarithm sweep signal as well as its errors and limitations caused by the fixed-point arithmetic is the aim of this paper. To generate an actual signal, a simple sine generator was created, however, it can be replaced by, for instance, some of the aforementioned solution.

[1] $1\ dec/s = \log_2 10\ oct/s \doteq 3,32\ oct/s$

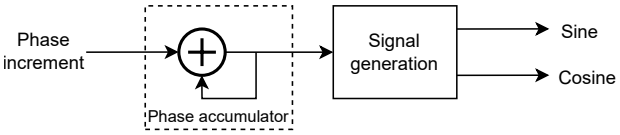

FIGURE 1. Principle of sine generation on FPGA of commercially available modules.

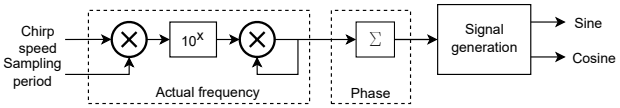

FIGURE 2. Principle of logarithm chirp sine generation on FPGA.

4. FPGA chirp signal generation

To achieve a point-to-point generation of a phase, it is necessary to integrate an angular velocity, which is the same, except the constant 2π, as a frequency. Consequently, it is more convenient to integrate frequencies, because the information about the actual frequency can be useful. Moreover, this way allows easier phase-wrapping to achieve the best available phase resolution as shown in (sec. 5.2).

The actual frequency of each sample can be calculated (eq. 2) by a multiplication of the previous frequency. In the case of the linear sweep, there is a simple addition of a frequency difference Δf and in the case of a pure sine signal, the frequency remains the same. This is the only variation of signal generation in our method. The block diagram of the generator is shown in figure (fig. 2).

$$f(n) = f(n-1) \cdot 10^{\frac{k}{f_s}} \tag{2}$$

where $f(n)$ and $f(n-1)$ are actual and previous frequencies (note that $f(0)$ is a start frequency), k is a chirp speed constant in (dec/s) and f_s is a sampling frequency.

The point-to-point phase generation from a known frequency can be done either by a numerical integration or by an integration from the analytical prescription.

4.1. Numerical Integration

The trapezoidal method appears convenient for a numerical integration, because this method needs only the current and the previously calculated frequency. Equation (3) shows the calculation. Exactly the same calculation can be used for the linear chirp sine and pure sine signals as well. However, this integration method generates an error, which is visualized on (fig. 3), where it is visible that for shorter durations, the error fades into insignificance compared with f.e. the DAC quantization error or noise.

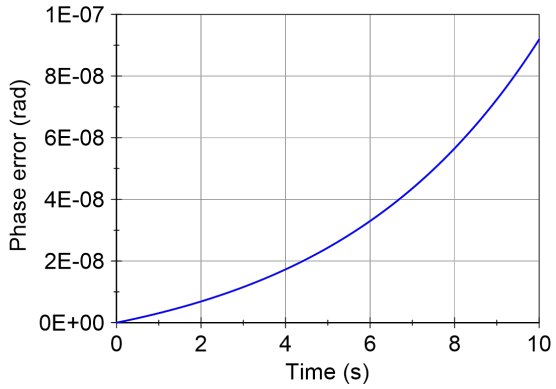

FIGURE 3. A phase error between the logarithmic chirp signal generated by a numerical integration using the trapezoidal rule and a LabVIEW built-in function. Signal parameters $f_s = 10\ kHz$, $f_{start} = 10\ Hz$, $f_{stop} = 100\ Hz$.

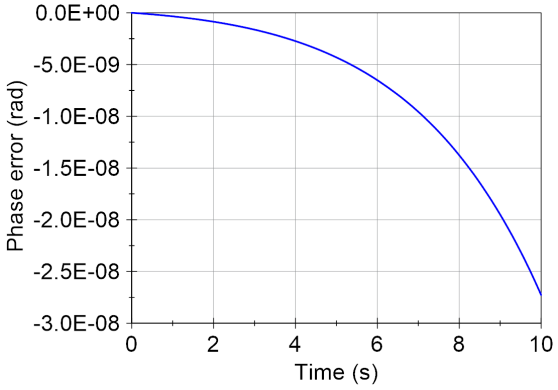

FIGURE 4. A phase error between logarithmic chirp signal generated by an analytical integration using fixed-point and floating point representation. Signal parameters $f_s = 10\ kHz$, $f_{start} = 10\ Hz$, $f_{stop} = 100\ Hz$.

$$\varphi(n) = \varphi(n-1) \ + \ 2\pi \cdot T \cdot \left(\frac{f(n) + f(n-1)}{2} \right) \quad (3)$$

where $\varphi(n)$ and $\varphi(n-1)$ are the current and previous phases, T is the sampling period, $f(n)$ and $f(n-1)$ are the actual and previous frequencies.

4.2. Analytical Integration

To calculate the phase from the angular velocity (or frequency) is analytically simple (eq. 4) due to the trivial frequency function. The calculation is very simple because the $ln(k)$ is a constant so it can be calculated beforehand.

As the calculation follows the analytical rule, the method error should be zero. However, there is an error (fig. 4) between the fixed-point implementation of this method and the floating point function. This is due to the limited fixed-point resolution.

$$\varphi(n) = \varphi(n-1) \ + \ 2\pi \cdot \frac{f(n) - f(n-1)}{ln(k)} \quad (4)$$

where $\varphi(n)$ and $\varphi(n-1)$ are the current and previous phases, $f(n)$ and $f(n-1)$ are the actual and previous frequencies and k is the chirp speed constant.

The numerical integration is less accurate, however, it is more sophisticated, as it can be used for the linear sweep and the sine generation as well. Because the method error in the presented case is significant only for long sweep durations (hours), it seems more practical to use the numerical integration method instead of the analytical one.

5. Fixed-point limitations

FPGAs are working with a fixed-point number representation to achieve the desired speed of computation and parallelism. This approach provides several differences compared to the floating-point numbers. On the one hand, there are some advantages such as numeric overflow, and on the other hand some disadvantages like a lower resolution.

10 kHz		50 kHz	
24-bit	32-bit	24-bit	32-bit
0.0988832	0.0999955	0.0957778	0.0999843
0.0994009	0.0999975	0.0983664	0.0999944
0.0999186	0.0999995	0.1009550	0.1000045
0.1004363	0.1000016	0.1035436	0.1000147
0.1009541	0.1000036	0.1061322	0.1000248
0.1014718	0.1000056	0.1087208	0.1000349

TABLE 1. Possible values of a chirp speed in (dec/s) closest to the presented chirp speed $k = 0,1\ dec/s$ for fixed-point bit lengths 24 and 32 and sampling frequencies 10 kHz and 50 kHz.

5.1. Resolution limit

Since the resolution limit seems to be a great issue, it can be solved easily by increasing the number bit length. However, the higher the bit length, the higher the resources consumption is, which limits the amount of code to fit in the FPGA. For this reason, it is essential to determine the necessary bit length beforehand. In this paper, two different bit lengths, 24 and 32, are used to show the differences.

As visible from (eq. 2), the actual frequency is calculated from the previous one by multiplication by a number, which is very close to one[2]. The resolution of this multiplier affects the possible chirp speeds and it is the main limitation of the fixed-point generation. This also limits the possible chirp speeds to several discrete values (tab. 1) and leads to a frequency error. The error between the ideal (float-point calculation) and actual (fixed-point calculation) frequency is shown in (fig. 5).

[2] in presented case the number is $10^{0,00001} \doteq 1.0000230261$

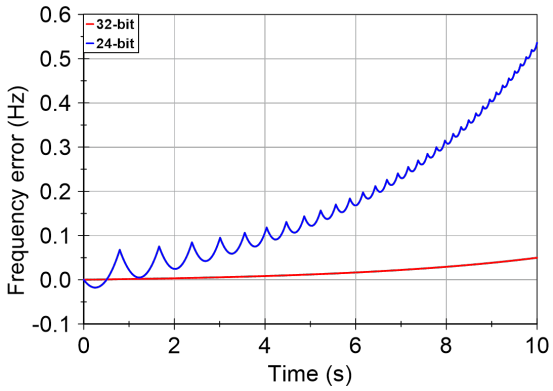

FIGURE 5. A frequency error between a logarithmic chirp signal generated by 24 and 32-bit fixed-point arithmetic with rounding and a floating-point generation. Signal parameters $f_s = 10\ kHz$, $f_{start} = 10\ Hz$, $f_{stop} = 100\ Hz$.

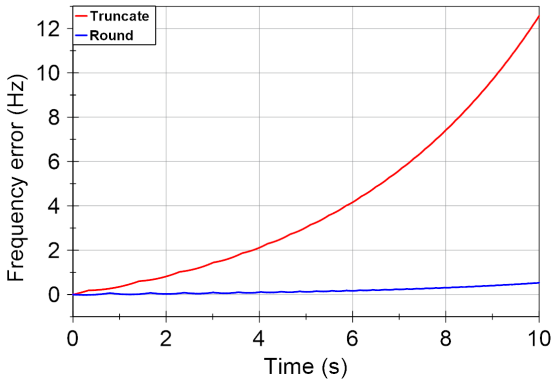

FIGURE 6. A frequency error between a logarithmic chirp signal generated by 24-bit fixed-point arithmetic with different coercing options, truncate and round and a floating-point generation. Signal parameters $f_s = 10\ kHz$, $f_{start} = 10\ Hz$, $f_{stop} = 100\ Hz$.

However, it is usually enough to keep the chirp speed within a certain limit. For example, the [2] limits the frequency error for a vibration testing to ±5 %. This can also be easily achieved with the fixed-point computation, where the errors can be much smaller (tab. 1).

More important is the accuracy of the actual calculated frequency (phase). As the calculation itself produces no error, the result has to be rounded to fit into the fixed-point range. The maximum rounding error for one calculation is one least significant bit (LSB), so it can be neglected. However, the rounding error is accumulated throughout the whole signal generation, where it can cause a significant differences as shown in (sec. 5.3). For this reason, it is necessary to handle the rounding properly. In conclusion, the calculated frequency is the actual frequency of the generated sample up to an error of the sine function, which is defined by an actual implementation. However, the frequency is different from the ideal one due to the rounding.

5.2. Number wrapping

To keep the phase accumulation error as low as possible, it is necessary to have a maximal fixed-point resolution. However, the phase of the logarithm chirp signal is exponentially rising, so it is necessary to wrap it. Phase wrapping is normally done as a remainder after division, however, this method requires a fixed-point division, which is inaccurate and time demanding. For this reason, it is better to let the phase wrap when the fixed-point overflows. If we use modified units ($\pi \cdot rad$) instead of normal phase units (rad) modified units ($\pi \cdot rad$), then the phase can easily be wrapped, when it exceeds the value 2, because it means $2\pi\ rad$, so one period of a sine function. Moreover, it is very easy to wrap around this value just by ignoring the overflow status and keeping the rest.

5.3. Result rounding

A result from the fixed-point arithmetic operation has to be rounded to fit into the predefined bit length. There are two possible ways how to proceed. Unnecessary bits can be either cut off (truncate mode) or the number can be coerced. This is done by adding half of the LSB to the result and then the result is truncated. The truncate mode is very fast, however, it can produce error of up to one LSB. However, coercing requires a little bit more resources and one more adding operation, but the error is half of the LSB maximum. More information is provided in [5].

Both methods can be used in the chirp signal generation with a different impact on the result. The truncate method consumes less resources, but produces a bigger error than the other method, which is more resource demanding. The errors for the 24-bit calculation are shown in (fig. 6). The error of the rounded result is clearly visible in (fig. 5). Moreover, the actual change of rounding from lower to higher value and vice versa, which causes the non-monotony of the error curve, is also visible .

6. Limitaions

When considering appropriate bit length of a fixed-point representation, it is necessary to consider its limitations. A maximal frequency is determined by a decimal part of the fixed-point, whereas the rest, a fractional part, limits a resolution, a chirp speed and a sampling frequency.

The maximal frequency in the chirp signal has to be lower than the maximum represented value of the fixed-point. Otherwise, the frequency will be coerced or even worse, wrapped. This will result in a completely different signal. Fortunately, the maximum frequency can be easily calculated from the number of decimal bits.

The length of the fractional part indicates the frequency resolution, which has to be lower than the

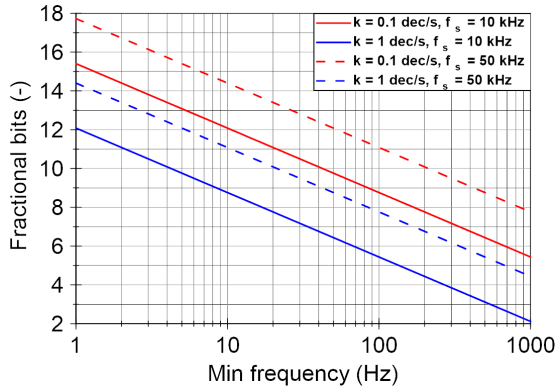

FIGURE 7. A minimal length of a fractional part of a fixed-point number in dependence on a minimal frequency in a chirp signal for different chirp speeds and sampling frequencies.

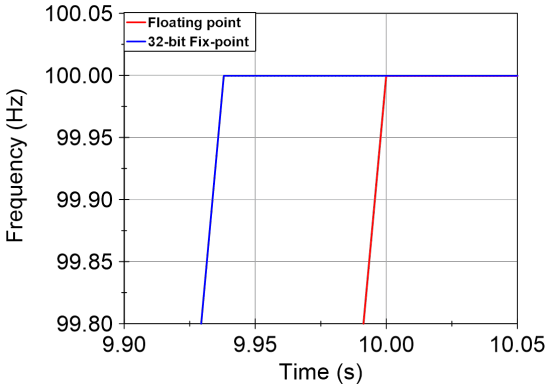

FIGURE 8. Comparison of the actual chirp speed frequency generated on 32-bit fixed-point and floating point arithmetics.

difference between two lowest frequencies in the chirp signal. Otherwise, the new frequency will be coerced to the previous one, which results in a sine signal instead of the chirp one. The difference is determined by the chirp speed and the sampling frequency. The calculation of the necessary fractional resolution is presented in (eq. 5) and it is visualized for the selected cases in (fig. 7).

$$res \geq log_2 \left(\frac{1}{f_{min} \cdot \left(10^{\frac{k}{f_s}} - 1 \right)} \right) \tag{5}$$

where res (bit) is a minimal number of fractional bits in fixed-point number, f_{min} (Hz) is a minimal frequency in signal, k (dec/s) is a chirp speed and f_s (Hz) is a sampling frequency.

The amplitude of the signal is, when an appropriate fixed-point representation is selected, determined by a used analog output card. However, especially small amplitudes can also be affected by rounding in the sine evaluation. However, this possible issue should be solved in advance by selecting necessary precision of the function used.

For example, a versatile fixed-point generator for a vibration testing should be able to generate a chirp signal from 10 Hz to 7 kHz with a chirp speed[3] 1 oct/min. In this case, a 50 kHz sampling frequency is sufficient. Then, the versatile generator requires 13 $bits$ of the decimal part and 19 $bits$ of the fractional part. So, in general, a 32-bit fixed-point representation of a frequency is enough for this generator.

7. IMPLEMENTATION

The generator was implemented in LabVIEW 2018 and executed on Compact RIO 9067. The data type used for the implementation was a 32-bit fixed-point with rounding after an arithmetic operation, because in our case, this representation can meet the aforementioned criteria.

The generator is able to generate a logarithmic and a linear chirp signal as well as a sine signal. This is possible due to a different method of the actual frequency calculation. It can be calculated by a multiplication for the logarithmic sweep, an addition for the linear sweep or remain the same as a previous frequency for the sine signal.

Parameters for the generation are sent to the FPGA from a superior control system. The start and stop frequencies and the amplitude are transmitted unchanged, but the chirp speed is recalculated according to the sampling frequency to a multiplier (eq. 2). Whereas the sampling frequency is determined by an analog output card speed so it is not necessary to transmit it.

The actual frequency is coerced to fit between the start and stop frequencies. When the stop frequency is reached, a flag about the chirp completion is set. This is due to the unknown duration of the chirp caused by the discrete chirp speeds (tab. 1) and rounding. After the completion, a sine signal with the same frequency (stop frequency) and amplitude is generated until the superior system does not change the parameters or shut down the generation.

This algorithm consumes 406 total slices, 1317 slice LUTs and 8 DSP48, which are used for the sine evaluation.

7.1. GENERATED SIGNAL

The implementation results are shown on the logarithm chirp sine with aforementioned parameters ($f_{start} = 10\ Hz$, $f_{stop} = 100\ Hz$, $k = 0.1\ dec/s$).

Due to the resolution limit, the chirp speed is a little bit higher than the selected value (sec. 5.1), therefore, the duration is shorter, as can be seen in the figure (8).

Unlike the chirp speed, where slight differences are usually tolerable, the spectral purity is essential, especially in the transition between the chirp and sine

[3] $1\ oct/min \doteq 0.005 dec/s$

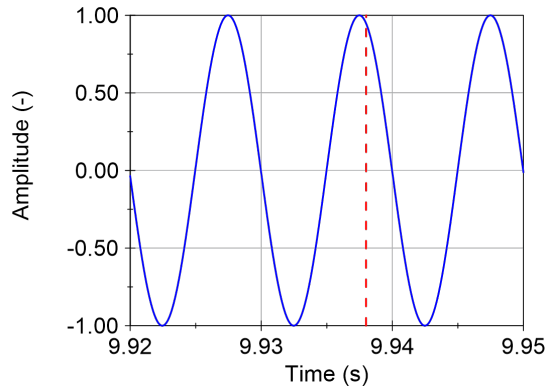

FIGURE 9. Detail of the transition between the chirp and sine signal generated by our 32-bit fixed-point generator.

signal. In our implementation, the transition seems smooth, because it changes only the phase difference calculation. The transition is shown in figure (fig. 9). Nevertheless, more sophisticated sine signal generator with this phase generator can be used to ensure a better signal purity.

One can see that the implementation of the fixed-point generator provides a trustworthy signal with only few limitations.

8. RESULTS

This paper describes the point-to-point generation of a logarithm and linear chirp as well as pure sine signal using a fixed-point number representation to use the algorithm on the FPGA. Main errors and limitations are also discussed. For the selected FPGA, the platform implementation of the point-to-point signal generation method is used. This method and its errors are explained in (sec. 4), where it is shown that the method errors fade into insignificance when compared to others.

Transferring the calculation from a floating-point to a fixed-point arithmetic comes with other limitations, such as discrete chirp speeds (tab. 1), and of course, accuracy errors (fig. 5). Moreover, the fixed-point calculation suffers from additional issues, which have to be considered, such as results rounding (fig. 6). Using a specific fixed-point length also limits the range of possible frequencies in dependence on the chirp speed and the sampling frequency. These limitations are explained and evaluated (fig. 7).

The actual implementation of the method is described in (sec. 7). The implementation was done to meet the vibration testing criteria according to [2], so the parameters are selected accordingly. The generator is able to generate not only the logarithmic chirp sine signal, but also the linear chirp and sine signal and is able to switch between them instantaneously and with minimal phase noise. Furthermore, the actual frequency of the sample is always known. The implementation also deals with some fixed-point generated issues, such as the discrete values of the chirp speed or the frequency outside limits, which makes it a versatile tool for different engineering areas.

9. CONCLUSION

This paper presents an universal point-to-point method of chirp sine signal generation on a fixed-point FPGA. Differences between the floating and the fixed-point generation as well as the most significant error sources are described. Moreover, the evaluation of the minimal fixed-point resolution and overall error of the method are presented. The generator is able to instantaneously switch between chirp sine and sine signal without any additional phase noise. The generator has been realized on Compact RIO 9067 and is used for vibration testing from 10 Hz to 7 kHz with a minimal chirp speed of 1 oct/min. Nevertheless, the generator is versatile, therefore it can be used in other areas with a different frequency and speed range. In the future, we would like to implement a vibration control to the fixed-point arithmetic, so the whole process will be on FPGA.

ACKNOWLEDGEMENTS

The completion of this paper was made possible by the grant No. FEKT-S-17-4234 - "Industry 4.0 in automation and cybernetics" financially supported by the Internal science fund of Brno University of Technology.

REFERENCES

[1] R. L. Allen, D. W. Mills. *Time, Frequency, Scale and Structure*. John Wiley & Sons, Inc., 2004.

[2] IEC:60068-2-6. *Environmental testing: Tests – Test Fc: Vibration (sinusoidal)*, 2nd edn., 2007.

[3] J. Vankka, K. A. Halonen. *Direct digital synthesizers: theory, design and applications*, vol. 614. Springer Science & Business Media, 2013.

[4] E. Murphy, C. Slattery. Ask the application engineer—33 all about direct digital synthesis. *Analog Dialogue* **38**(3):8 – 12, 2004.

[5] U. Meyer-Baese. *Digital signal processing with field programmable gate arrays*, vol. 2. Springer, 2004.

[6] National Instruments. CompactRIO Systems. `https://www.ni.com/cs-cz/shop/compactrio.html`.

[7] S. Srivastava, P. Hobden. 5Ghz Chirp Signal Generator for Broadband FMCW Radar Applications. In *2018 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, pp. 152 – 155. 2018. DOI:10.1109/iSES.2018.00041.

[8] H. Yang, S.-B. Ryu, H.-C. Lee, et al. Implementation of DDS chirp signal generator on FPGA. In *2014 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 956–959. IEEE, 2014. DOI:10.1109/ICTC.2014.6983343.

[9] H. Yang, S.-B. Ryu, H.-C. Lee, et al. Implementation of DDS chirp signal generator on FPGA pp. 956–959, 2014. DOI:10.1109/ICTC.2014.6983343.

[10] M. Y. Chua, J. T. Sri Sumantyo, Y. Q. Ji. An 8-Channels FPGA-Based Reconfigurable Chirp Generator for Multi-Band Full Polarimetric Airborne/Spaceborne CP-SAR. In *2018 Progress in Electromagnetics Research Symposium (PIERS-Toyama)*, pp. 876–881. 2018. DOI:10.23919/PIERS.2018.8597713.

[11] J. An, H. Jung, H. Yang, et al. Development of chirp signal generator for micro satellite on-board synthetic aperture radar. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3663–3666. IEEE, 2015.

[12] H. Yang, Y. Izumi, A. Hendra, J. T. S. Sumantyo. Novel chirp phase error compensation algorithm using polynomial chirp modeling for high resolution synthetic aperture radar. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 6012–6015. 2017. DOI:10.1109/IGARSS.2017.8128380.

[13] Intel. Nco ip core: User guide. `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_nco.pdf`, 2017. Accessed: 16 September 2020.

[14] XILINX. Dds compiler v6.0. `https://www.xilinx.com/support/documentation/ip_documentation/dds_compiler/v6_0/pg141-dds-compiler.pdf`, 2017. Accessed: 16 September 2020.