Jacek BORKO, **Grzegorz DULNIK,** Adam GRZELKA, Adam ŁUCZAK, Adam PASZKOWSKI
CHAIR OF MULTIMEDIA TELECOMMUNICATIONS AND MICROELECTRONICS
Poznań University of Technology
Polanka 3, 60-965 Poznań, POLAND

DOI 10.1000/182

Parametric synthesizer of audio signals on FPGA

Abstract

The article shows sound synthesizer dedicated to parametric sound compression, but also able to generate any signal in audible band. The synthesizer is realized on FPGA platform, and it uses additive sound synthesis. Synthesizer is characterized by high quality, faithful reproduction of sound and low power consumption.

Keywords: FPGA, Sound Synthesis

1. Introduction

Sound synthesis comprises a wide field of research, including many sound generation methods. Unfortunately, the most accurate and universal methods are characterized by high complexity and demand for power. Fortunately, dynamic development of electronics allows to elaborate more efficient implementations for complex processes, like sound synthesis.

There is many FPGA-based sound synthesizers described in literature, also in MAM (formerly PAK) magazine. Most of these projects similarly as in article [1] use piano keyboard to deliver input parameters and produces finite number of built-in tones and effects. Thus, the amount of sounds they generate is limited.

The goal of presented solution is to design a synthesizer that can generate any signal in audible band. One of the applications of such synthesizer can be a parametric sound coding system [3]. The synthesizer constitutes a part of a decoder where, based on parameters of sinusoidal components, the sound is generated.

In the presented solution additive synthesis has been used, because of its universality and fidelity. This method consists of adding together many independent sinusoidal components [2]. Each **component is described by amplitude, phase, and frequency**, as shown in Figure 1. The more components, the more realistic sound. The realism of generated sound is achieved by using **several hundred of independent components** with variable parameters. The parameters can be obtained by spectral analysis of particular sound, which is to be imitated. That rule is used in parametric sound coding [3], where parameters of sinusoidal components are coded.

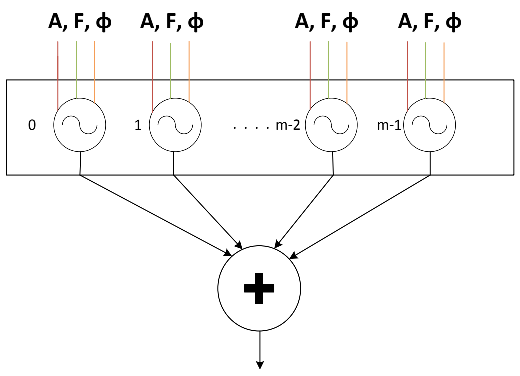

Figure 1. Additive sound synthesis

In order to implement additive synthesis, only one sine generator needs to be designed, because the rest of sinusoidal components is generated in the same way.

2. Implementation

In the discussed solution the samples of sine are generated by interpolation based on reference 16-bit samples from one period of sine, calculated in Matlab. Due to properties of sine signal, only one quarter of period has to be calculated and remembered. Sample values in other quarters are the same, just with different sign or order, as in *Figure 2*. Based on relations between quarters of sine period, we can define samples from the whole period, storing just one quarter in memory. This allows to store 4 times more referent samples in memory, so the sine can be sampled 4 times more densely, which increases final precision of generated samples. Most FPGA devices have block RAM memory size of 1MB or more, so the solution involves 1 MB of referent 16-bit samples, to be universal. It gives 512 referent samples per quarter of sinus period.

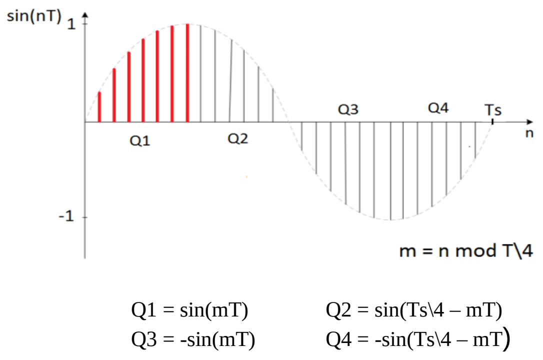

Figure 2. Samples in one period of sine

Successive samples of sinusoidal component with given parameters are generated by moving across the sine period. Specified space, dependent of frequency and phase parameters, is added to the position of the previous sample. As a result, the position of the next sample is obtained. Thus, for example, assuming 48 kHz output sampling frequency, in 12kHz sine component with 0 phase, the interspace between samples is quarter of period, as illustrated in *Figure 3*.

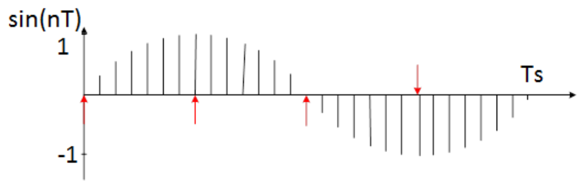

Figure 3. Location of samples in 12 kHz sine example

The actual position is the remainder from division by sine period(Ts), so it moves only across one period. Otherwise, the position would exceed the size of a referent sample table. Once the sample position is calculated, it is known in which quarter the sample is (Q1,Q2,Q3 or Q4). This information allows to define the sign of sample. Also, based on previously mentioned relations between quarters, it is possible to find the sample position in 16-bit referent samples table (quarter period size).

Densely sampled reference sine allows for using simple linear interpolation while maintaining satisfying precision of synthesized samples. Knowing the position in reference quarter, two neighboring reference 16-bit samples are used to interpolate desired sample at certain position, as in *Figure 4.*

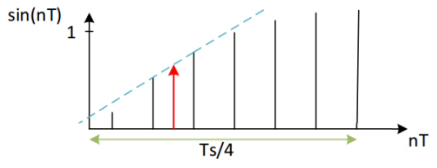

Figure 4. Linear Interpolation

In that process a 24-bit sample is obtained. Finally, taking into account amplitude parameter and previously defined sign, the sample extends to 25 bits. That is the way, a sample of sinusoidal component is generated. Next samples are calculated the same way. All there is left to do is to duplicate such generators, and add their outputs to obtain the final signal.

3. FPGA realization

It is easy to notice, that the structure of every sine generator is the same. Therefore, the system performs the same operation all the time, just with different input parameters. This allows for pipeline processing, which is the domain of FPGA devices [4].

Instead of many generators, only one sine generator has been designed but it works in pipeline. In considered solution, the process of generating one sample of one sinusoidal component takes over 20 clock cycles. With the use of pipeline processing, it is possible to obtain effectively one component sample per clock cycle, which is a huge gain. The assumed number of components, which allows for faithful sound reproduction is 256. So one output sound sample is created from 256 component samples, and one component sample is produced per one clock cycle. As the target of the synthesis are audio signals, the selected output sample frequency is 48 kHz, so system clock frequency is:

$$256 * 48000 * 1(sample\backslash cycle) = 12.288\ MHz.$$

The simplified diagram of synthesizer is shown in *Figure 5*.

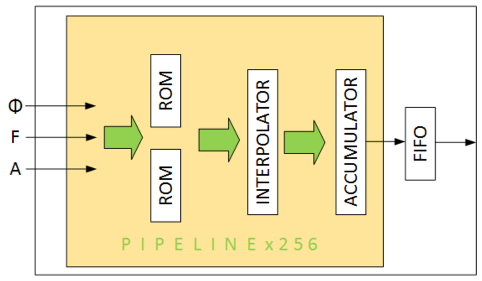

Figure 5. Simplified synthesiser diagram

Based on frequency and phase, the position in referent samples table is calculated, as it is described in the first part of the article. Then, two neighboring referent samples are read from ROM memories. Based on these two samples, the searched component sample is interpolated in the next step. The result is multiplied by amplitude parameter and modified depending on the sign. The next step is to add the obtained component sample to accumulator. This process is executed in pipeline, until all 256 component samples are added to accumulator. Then, the final output sample from accumulator is written to FIFO queue, accumulator is set to 0, and all process starts over. FIFO queue is used to send samples to other modules, also those unsynchronized with the synthesizer module.

4. Validating

To verify the design in a real time, a runtime environment has been set up, as shown in *Figure 6*. FPGA modules are launched on Xilinx Spartan 6. Communication with computer or other external device is provided by UART PORT using RS232 protocol. Parameters sent from a computer are written to 3 RAM modules, corresponding to 3 input parameters. The synthesizer reads parameters from RAM, calculates samples, and sends them to I^{2}S module. There the conversion to I^{2}S interface serialized form takes place, used in D/A converter.

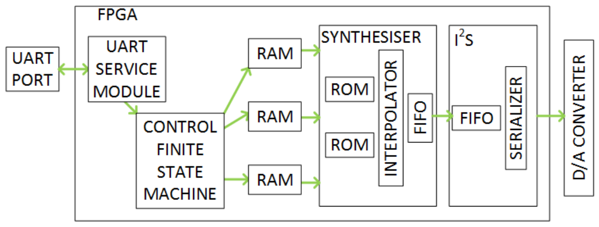

Figure 6. Runtime environment diagram

The system arranged this way, allows delivering input parameters to synthesizer and hearing the synthesized sound. The board with Xilinx Spartan 6 and D/C converters, used for real time tests is shown in *Figure 7*.

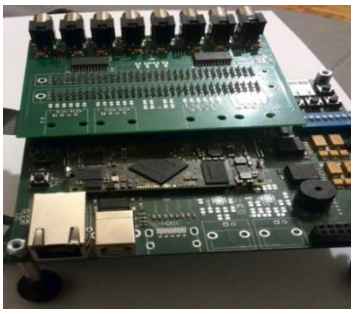

Figure 7. Board with FPGA and D/C converters

The use of logic resources by the synthesizer on Xilinx Spartan 6 is shown in *Figure 8*.

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	594	54.576	1%
Number of Slice LUTs	612	27.288	2%
Number of DSP48A1s	3	58	5%
Number of RAMB8BWERs	2	232	1%

Figure 8. Resources usage on Spartan 6 *XC6SLX45T*

The precision of synthesizer was measured by comparing 2048 synthesized samples in a quarter of sine period with samples calculated in Matlab. Peak sine value compared to error was at least 43,01 dB (0.005% of peak sine amplitude), as shown in *Figure 9*.

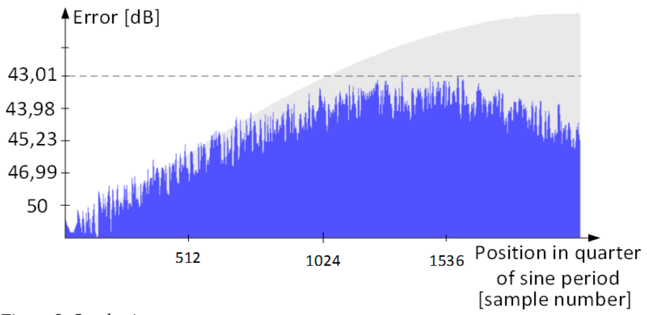

Figure 9. Synthesis error

There is a non-negligible error, caused by simple linear interpolation. In order to reduce the error, the simplest way is to sample the reference sine more densely, if there is more memory in FPGA device. Other way is to use more accurate type of interpolation than linear one. Such modification should not enlarge the synthesizer module dramatically.

5. Conclusion

The presented project effectively implements additive sound synthesis. Additionally can also be used to generate any signal in audible band. The input parameters can be changed every sample, which allows a fully controlled output signal. Due to pipeline processing one component sample is generated per one clock cycle. The system is universal, the number of components and output sampling frequency can be easily changed, and the system clock is described by the equation:

$$CLK = c * f * 1$$

1 - clock cycles per component sample, c - number of components, f – output sample frequency

In the tested scenario, with 256 components and 48 kHz output sample frequency, the clock is only 12.288 Mhz. Low clock frequency causes low power consumption. 25 bit wide output signal provides high quality. Very low use of logic resources on Spartan 6 *XC6SLX45T* (low-medium size FPGA) means that the synthesizer module can fit in almost every currently manufactured FPGA with DSP blocks, including the smallest (cheapest) ones.
Thanks to additive synthesis method and many component sines used, the device is able to generate any kind of sound, unlike most of other FPGA-based synthesizers, which often have limitations.
This is why the presented design can be used in efficient audio coding, e.g. as a part of decoder in parametric audio coding system [5].

The only issue to improve, is the error level. In order to minimize it, the reference sine signal should be sampled more densely, or a more accurate interpolation method should be used.

6. Acknowledgements

The presented work has been funded by the Polish Ministry of Science and Higher Education within the status activity task “Signal processing and antenna optimization for acquisition, processing, analysis and presentation in 3D television systems” in 2015.

7. References

[1] M.Łazoryszczak:” An experimental reconfigurable platform for sound processing applications”, PAK 2014 nr 07, s. 420-422

[2] T.Tolonen, V.Välimäki, M.Karjalainen: „Evaluation of Modern Sound Synthesis Method”, Helsinki University of Technology, March 1998

[3] R. McAulay, Th. Quatieri: “Speech Analysis/Synthesis Based on a Sinusoidal Representation”, in IEEE Transactions on Acoustics, Speech, and Signal Processing, August 1986

[4] S. Kilts: “Advanced FPGA Design – Architecture, Implementation and Optimization”, Spectrum Design Solutions, June 2007

[5] M.Bartkowiak: “State-of-the-art in audio coding”, Poznan University of Technology, 2006

Received: 00.00.2014 Paper reviewed Accepted: 05.01.2015

Ph.D. Adam ŁUCZAK

Received his M.Sc. and Ph.D. degrees from Poznan University of Technology in 1997 and 2001, respectively. In 1997 he joined the image processing team at Poznan University of Technology. Member of of Polish Society Theoretical and Applied Electrical Engineering (PTETiS). His research activities include video coders control, MPEG-4/H.264 systems and hardware implementations of digital signal processing algorithms. He received Annual Fellowship for Young Scientist from the Foundation for Polish Science (FNP) and also a Group Award of the Minister of the National Education for research in the National Education for research in image compression. Currently he is involved in some project on video coding and video delivery.

e-mail: aluczak@multimedia.edu.pl

M.Sc. Adam GRZELKA

Received M.Sc. degree from Poznan University of Technology in 2014. He is a Ph.D. student at the Chair of Multimedia Telecommunications and Microelectronics. The main area of his professional activities are image processing, FTV (Free Viewpoint Television) and FPGA – especially implementation of compression algorithms and communication interfaces.

e-mail: agrzelka@multimedia.edu.pl

Eng. Grzegorz DULNIK

He is a M.Sc. student at the Chair of Multimedia Telecommunications and Microelectronics. The main area of his interests are FPGA devices and microcontroller programming.

e-mail: dulnik.g@wp.pl

Eng. Adam PASZKOWSKI

He is a M.Sc. student at the Chair of Multimedia Telecommunications and Microelectronics. The main area of his interests are FPGA boards and microcontroller programming.

e-mail: adam9205@wp.pl

Eng. Jacek Borko

He is a M.Sc. student at the Chair of Multimedia Telecommunications and Microelectronics. He is interested in microcontrollers, FPGA boards and high level programming.

e-mail: jacekborko@gmail.com