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

A parametric synthesizer of audio signals on FPGA

Abstract

The paper presents a sound synthesizer dedicated to parametric sound compression, but also able to generate any signal in the audible band. The synthesizer is realized on an FPGA platform, and it uses additive sound synthesis. The synthesizer is characterized by a 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. However, the dynamic development of electronics allows elaborating more efficient implementations for complex processes, like sound synthesis.

There is many FPGA-based sound synthesizers described in literature and web portals. There are projects offering basic functionality, e.g. [1], other are more advanced and complex, e.g. [2], some of them introduce new capabilities, like an innovative way to control a device [3]. However, the common feature of most of them, is the use of an MIDI interface and a piano keyboard to deliver input parameters. Thus the amount of sounds they can generate is limited.

The goal of the presented solution is to design a synthesizer that can generate any signal in the audible band. A parametric sound coding system is the dedicated application of such a synthesizer [4, 5]. This efficient technique relies on decomposition, which divides the sound into sine components and other parts. The parameters of these components are coded. The synthesizer would be a part of a decoder in which the sound is generated based on the decoded parameters.

In the presented solution, the additive synthesis is used because of its universality and fidelity. This method consists in adding together many independent sinusoidal components [6]. Each **component is described by amplitude, phase, and frequency**, as shown in Figure 1. The more components, the more realistic sound. The realism of the generated sound is achieved by using **several hundred of independent components** with variable parameters.

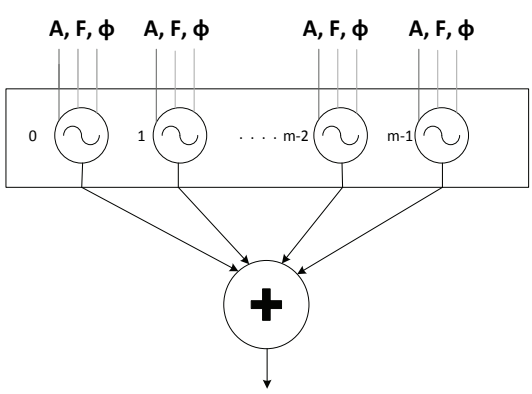

Fig. 1. Schematic diagram of additive sound synthesis

In order to implement additive synthesis, only one sine wave 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 a sine wave are generated by interpolation based on the reference 16-bit samples from one period of sine, calculated in Matlab. Due to the properties of the sine signal, only one quarter of the 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 the relations between the quarters of sine period, we can define samples from the whole period, storing just one quarter in a memory. This allows storing 4 times more referent samples in the memory, so the sine can be sampled 4 times more densely, which increases the final precision of the generated samples. Most FPGA devices have the block RAM memory size of 1 MB or more, so the solution involves 1 MB of referent 16-bit samples, to be universal. It gives 512 referent samples per a quarter of the sinus period.

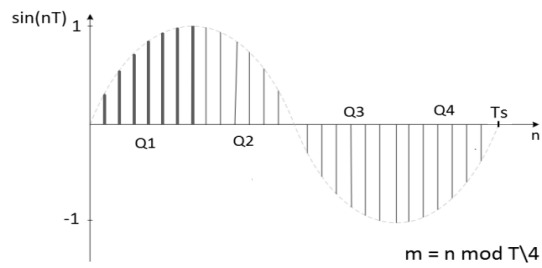

Fig. 2. Samples in one period of the sine wave

Successive samples of a sinusoidal component with given parameters are generated by moving across the sine period. The specified space, dependent on the 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 a 12 kHz sine wave component with 0 phase, the interspace between the samples is a quarter of the period, as it is shown in Figure 3.

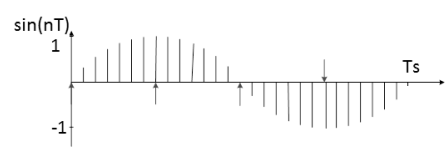

Fig. 3. Location of the samples in the 12 kHz example sine wave

The actual position is the remainder from division by the 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 defining the sign of the sample. Also, based on the previously mentioned relations between the quarters, it is possible to find the sample position in the 16-bit referent samples table (quarter period size).

A densely sampled reference sine wave allows using simple linear interpolation while maintaining the satisfying precision of the synthesized samples. Knowing the position in the reference quarter, two neighboring reference 16-bit samples are used to interpolate the desired sample at a certain position (see Figure 4).

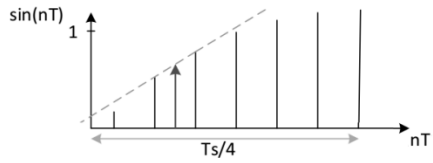

Fig. 4. Linear interpolation

In that process a 24-bit sample is obtained. Finally, taking into account the amplitude parameter and the previously defined sign, the sample extends to 25 bits. That is the way a sample of the 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 [7].

Instead of many generators, only one sine generator was designed but it worked in pipeline. In the 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 a clock cycle, which is a huge gain. The assumed number of components, which allows the 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 the system clock frequency is:

$$256 * 48000 * 1(\text{sample\cycle}) = 12.288 \text{ MHz}.$$

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

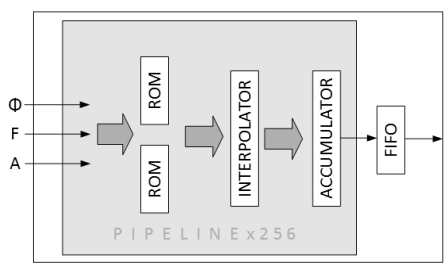

Fig. 5. Simplified diagram of the synthesizer

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

4. Validation

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

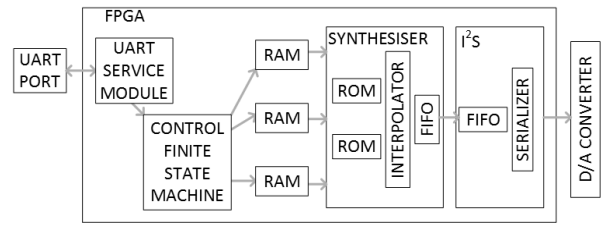

Fig. 6. Diagram of the runtime environment

The system arranged this way, allows delivering input parameters to the 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.

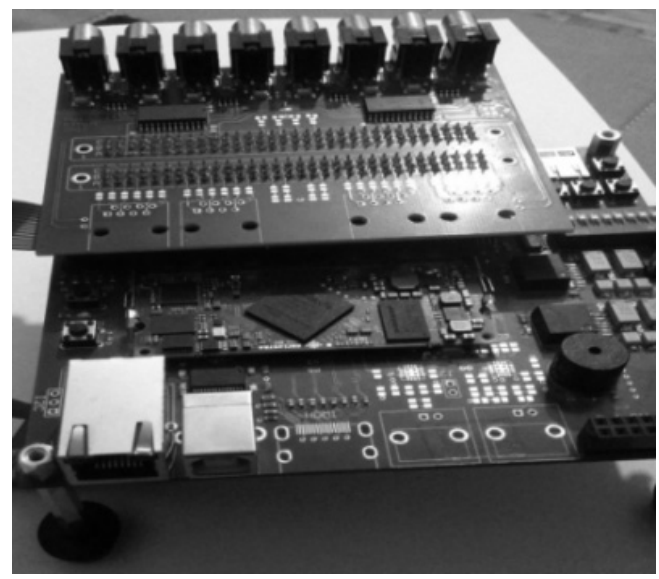

Fig. 7. The board with FPGA and D/C converters

The use of logic resources by the sythesizer 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%

Fig. 8. Usage of resources on Spartan 6 XC6SLX45T

The precision of the synthesizer was measured by comparing 2048 synthesized samples in a quarter of the sine period with the samples calculated in Matlab. PSNR was calculated for each sample. There is a non-negligible error, caused by simple linear interpolation. The simplest way to reduce the error is to sample the reference sine more densely, if there is more memory in the FPGA device. Other way is to use a more accurate type of interpolation than the linear one. Such modification should not enlarge the synthesizer module dramatically. It was at least 86.02 dB, as shown in Figure 9.

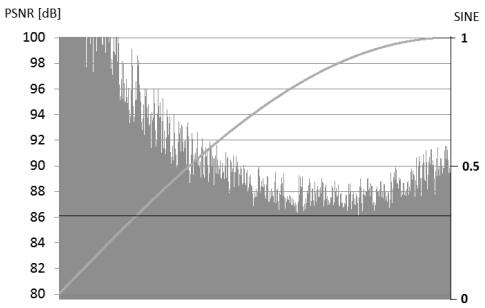

Fig. 9. The synthesis error

5. Conclusions

The presented project implements additive sound synthesis effectively. It can also generate any signal in the audible band. The input parameters can be changed for each 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 the output sampling frequency can be easily changed. In the tested scenario, with 256 components and 48 kHz output sample frequency, the clock is only 12.288 Mhz. The low clock frequency causes low power consumption. The 25 bit wide output signal provides the high quality. The use of logic resources on Spartan 6 XC6SLX45T (low-medium size FPGA) is very little, which means that the synthesizer module can fit in almost every currently manufactured FPGA with DSP blocks, including the smallest (cheapest) ones.

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.

6. References

[1] Schuhmacher J.: Altera based music workstation. 2006 [online], [access: 08-09-2015]. Available in Internet: http://www.96khz.org/htm/audiomusicworkstationvhdlcyclone2.htm
[2] Schuhmacher J.: Audio & music workstation with 4096 polyphonic voices. 2011 [online]. [access: 08-09-2015]. Available in Internet: http://www.96khz.org/htm/fpgamusicsynthesizercyclone4.htm
[3] Łazoryszczak M.: An experimental reconfigurable platform for sound processing applications. PAK 2014 no 07, p. 420-422.
[4] McAulay R., Quatieri Th.: Speech Analysis/Synthesis Based on a Sinusoidal Representation. In IEEE Transactions on Acoustics, Speech, and Signal Processing, August 1986.
[5] Bartkowiak M.: State-of-the-art in audio coding. Poznan University of Technology, 2006.
[6] Tolonen T., Välimäki V., Karjalainen M.: Evaluation of Modern Sound Synthesis Method, Helsinki University of Technology, March 1998.
[7] Kilts S.: Advanced FPGA Design – Architecture, Implementation and Optimization. Spectrum Design Solutions, June 2007.

Received: 17.04.2015 Paper reviewed Accepted: 02.06.2015

Adam ŁUCZAK, PhD

Received his MSc and PhD 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. Currently he is involved in some project on video coding and video delivery.

e-mail: aluczak@multimedia.edu.pl

Adam GRZELKA, MSc

Received MSc degree from Poznan University of Technology in 2014. He is a PhD 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

Grzegorz DULNIK, eng.

He is a MSc 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

Adam PASZKOWSKI, eng.

He is a MSc 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

Jacek BORKO, eng.

He is a MSc 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