Synthesizing Sine Wave Signals Based on Direct Digital Synthesis Using Field Programmable Gate Arrays

Hristo Z. Karailiev[1] and Valentina V. Rankovska[2]

***Abstract* – Analysis of the design flow for creating devices and systems based on Altera's Field Programmable Gate Arrays (FPGA) has been made in the present paper. The digital part of the sine wave synthesizer based on FPGA has been designed. The synthesizer is realized using the development system TREX C1 of Terasic Technologies Inc.**

***Keywords* – Field Programmable Gate Arrays - FPGA, Design flow, Direct Digital Synthesis – DDS, Sine wave frequency synthesizer.**

I. INTRODUCTION

The method for direct digital synthesis (DDS) [1] of signals with arbitrary form is well known, but for a long time its wide implementation has been prevented by the low level of the technology development. Various methods for producing arbitrary form output are known – analogue and digital. The DDS method has some advantages: high resolution; it allows an extremely fast transition to another frequency with continuous phase; the digital implementation allows easy realization of microprocessor control. These advantages determine its growing usage in functional generators, various modulations in the communications, etc.

Various digital implementations of DDS synthesizers have been described in the literature: based on discrete components and low scale integrated circuits, such as dividers, counters, etc.; modern application specific integrated circuits, such as AD9851, AD9858, AD9857 and lately based on Field Programmable Gate Arrays (FPGA).

A drawback of the application specific integrated circuits is the fact that they produce output of certain form, for instance AD9851 produces stable frequency and phase-programmable digitized analog output sine wave. They are not so suitable for applications, where signals with arbitrary wave form have to be created, for instance functional generators. That is why in the present work field programmable gate arrays have been used. This approach will allow in a single chip to integrate many functions for creating arbitrary form output [3].

Aim of the report::

Design and implementation of a sine wave synthesizer based on the direct digital synthesis method and field programmable gate arrays.

Main problems of the report:

- Analysis of the design flow for creating devices and systems based on Altera's FPGA;
- Designing the table, including the values of the sine wave;
- Designing the digital part of the sine wave synthesizer, based on FPGA of Altera;
- Implementation of the synthesizer using the development system TREX C1.

II. DESIGN FLOW WITH QUARTUS II SOFTWARE

The characteristics, features and resources of FPGA of leading producers have been outlined in [7] and a device of Altera has been chosen. The software of Altera Quartus II Web Edition v.6.1 has been used at the design process.

The main stages of the design flow with Quartus II are shown in Fig 1 [5].

- **Entering the design** of a device or system can be implemented in one of the following ways:

➢ *As a program*, written in one of the following hardware description languages – AHDL, VHDL, Verilog HDL.

The integrated text editor of Quartus II can be used. The software allows using so called MegaWizard Plug-In Manager. It supplies the designer with high level library blocks – megafunctions, which the designer can parameterizes. MegaWizard Plug-In Manager creates automatically the necessary files to include in the project according to the chosen programming language.

➢ *As a block diagram*.

The block editor of Quartus II is used. The block diagram may include library blocks and logic gates entered and parameterized with MegaWizard Plug-In Manager and also user blocks, created with the symbol editor of Quartus II.

- **Defining requirements for the project and settings of Quartus II**

Defining in advance requirements for the project and some settings of the software allows controlling the functions and the features both the software and the created design in order to increase its effectiveness. They are made by some program parts of Quartus II. Some of the assignments and requirements refer to: design files, the device used, timing requirements, etc. Some conditions for the design optimization in relation to the resources of the selected chip, the power consumed, time intervals, maximum frequency, compilation time can be also defined.

[1]Hristo Z. Karailiev is with the Technical University of Gabrovo, H. Dimitar 4, 5300 Gabrovo, Bulgaria, E-mail: hkarailiev@gmail.com

[2]Valentina V. Rankovska is with the Technical University of Gabrovo, H. Dimitar 4, 5300 Gabrovo, Bulgaria, E-mail: rankovska@tugab.bg

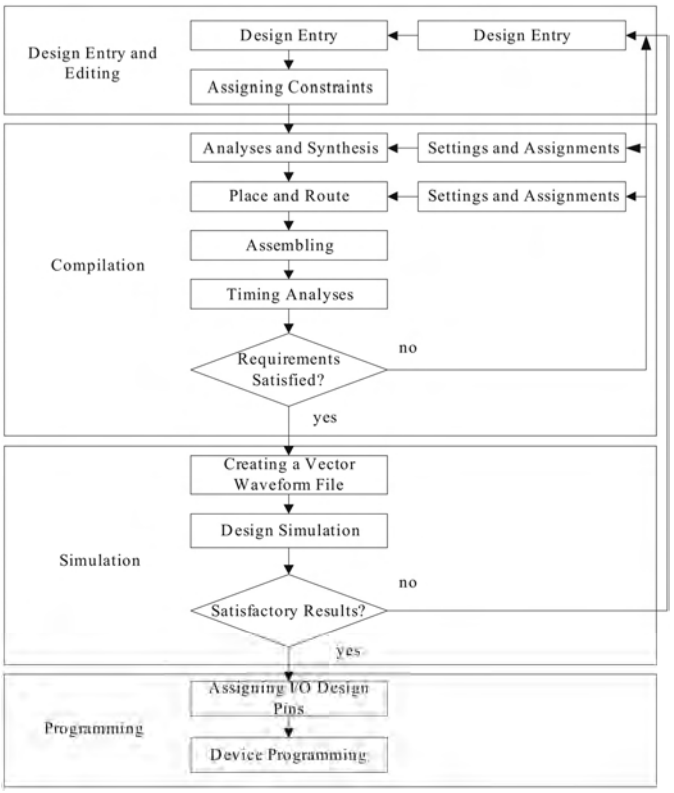

Fig. 1. Design flow with Quartus II

- **Design compilation**

Several consecutive processes take place at the compilation stage: *analysis and synthesis, place and route, assembling and timing analysis.* During every of the upper stages the design has been checked for its correctness. This stage is an iteration process – we can return to a previous one if it is necessary (if there are some errors) – till we receive a properly operating design.

At the ***Analysis and Synthesis*** the design database is created. Analysis & Synthesis performs logic synthesis to minimize the logic of the design, and performs technology mapping to implement the design logic using device resources such as logic elements. It groups register and combinational resources into individual logic cell-sized units in order to use resources efficiently. It examines the logical completeness and consistency of the project, and checks for boundary connectivity and syntax errors. It also optimizes the design for instance making choices that will minimize the number of resources as using functions, which are optimized for Altera devices.

During the ***Place and Route*** the defined timing and logic requirements are matched to the resources of the selected device. The most suitable place of the logic functions in the device logic cells is found and the most suitable interconnections and pin assignments are selected.

The Assembling completes the design processing, producing files for programming the device and information for the consumed power.

The Timing Analysis is a method of analyzing, debugging, and validating the performance of a design. Timing analysis measures the delay along the various timing paths and verifies the performance and operation of the design. These paths are the connections between the logic cells in the device as reference signals, data, etc. We can specify constraints and assignments that help the design meet timing requirements. If we specify constraints or assignments, the Fitter optimizes the placement of logic in the device in order to meet those constraints. After that timing analysis calculates the time needed the signals to reach their destination. It can also calculate signal transitions.

- **Design simulation**

At the simulation test and settings of the logic operations and timing relations in the design is made. First a file with input stimuli for the design input pins is created. Depending on the needed information we can make functional or timing simulation and to test the logical operation and timings in the worst case for the current design. We can estimate the simulation results visually. If the design needs to be corrected that can be made in the design entry. The compilation and the simulation repeat after that.

- **Device Programming**

At the programming the files produced by the compiler are loaded into the device and it is configured. But first assignment the design pins to the physical device pins is done. The design is compiled again and the device is programmed.

III. Architecture of a DDS Sine Wave Frequency Synthesizer

An architecture of a frequency synthesizer, used for creating a frequency grid, is shown in [3], based on a study of many references. The presented block diagram can be used for producing arbitrary form signals. In the current design it is used for implementing a sine wave frequency synthesizer (Fig. 2).

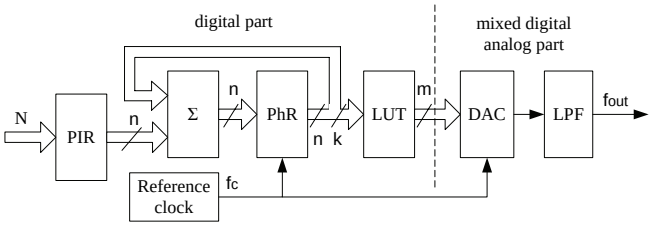

Fig. 2. DDS frequency synthesizer

Briefly the DDS synthesizer operates in the following way: The digital equivalent of the produced frequency is loaded into the phase increment register *PIR*. That value is continuously added to the value, accumulated in the adder $\sum$. The most significant *k* bits of the result address the Look-Up Table (*LUT*). In our case the LUT includes a set of values defining the form of the sinusoid. The values, derived from the table, are passed to the Digital Analogue Converter (DAC) to receive an analogue signal and after that to a low-passed filter (LPF) to reject the unwanted components of the signal and its smoothing out.

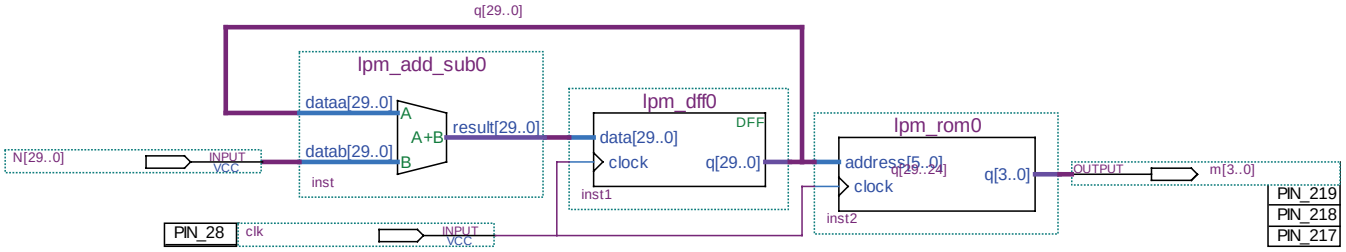

Fig. 3. Functional circuit of the digital part of the sine wave frequency synthesizer

IV. Designing a Sine Wave Frequency Synthesizer with FPGA

- *Creating the project of the sine wave frequency synthesizer*

Fig. 3 shows the functional circuit of the digital part of the sine wave frequency synthesizer, which is implemented in Altera's FPGA Cyclone EP1C6Q240C8 [2]. The registers and the adder are chosen to be 30 bits wide (n=30).

The development system TREX C1 of Terasic Technologies Inc. [6] has been used at designing and examining the frequency synthesizer. It has the following resources: f_{IT}=50 MHz, FPGA Cyclone EP1C6Q240C8, three 4-bit DACs. For the present implementation of the sine wave synthesizer one four-bit DAC (shown in Fig.3) variant and eight-bit DAC variant have been used. A standard three-tap Π-type LC filter has been used.

- *Creating and Filling in the LUT Table*

The output signal Y_{ROM}, passed to the input of the DAC can be expressed by Eq. (1):

$$Y_{ROM}=\{\sin[\pi.(N-2^{k-1})/2^{k-1}]\}.(-1).(2^{m-1}-1)+(2^{m-1}-1) \quad (1)$$

where: k – the number of the address inputs of LUT; 2^k – the number of the cells in the LUT; 2^{k-1} – the number of the cells for the positive (negative) part of the sinusoid; $N \in 0 \div 2^k-1$ – the current number of the cell in the LUT, matching the current number of a point of the sinusoid; m – the length of the cells, represented in a number of bits (the resolution of the DAC); $M=2^m$ – a level scaling factor; $2^{m-1}-1$ – an offset of the sinusoid on the ordinate, in order to receive positive values of the function.

The values Y_{ROM} are mixed fractions and can't be loaded in that way into the LUT. That is why they must be rounded and that operation is a source of errors.

The limited number of address inputs k of the LUT and the length p of the cells, define the non-linearity of the output sinusoid. To increase the linearity using the DDS method it is necessary k and m to be relatively big numbers. On the other hand to increase the spurious-free performance it is necessary to take into account the Eq. (2) [6]:

$$k=m+2 \quad (2)$$

In the case of 4-bit DAC $m=4$, $k=6$, $N_{max}=2^k=64$, $M=2^m=16$. The values received for Y_{ROM} and Y^r_{ROM} in relation to the current number of the cell $N \in 0$ -63, are shown in Fig. 4, and Table I includes the exact and the rounded values, loaded into the MIF file in Quartus II.

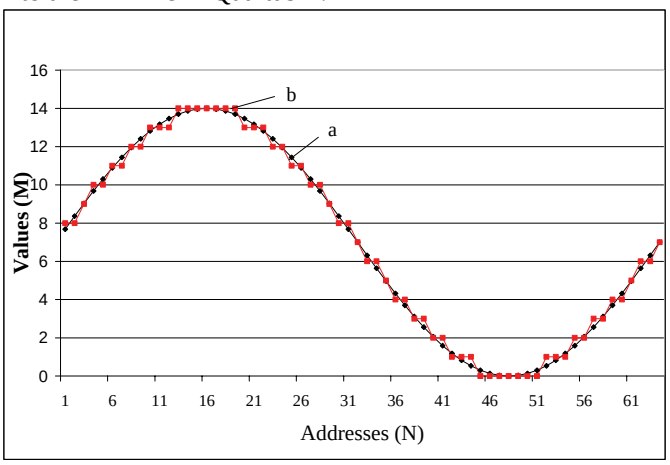

Fig. 4. Output signal, defined by the calculated (a) and rounded (b) values for the amplitude at N_{max}=64

In the case of 8-bit DAC $m=8$, $k=10$, $N_{max}=1024$, $M=256$. The values loaded in the MIF file (Fig. 5) are too much that is why they are not shown in a table.

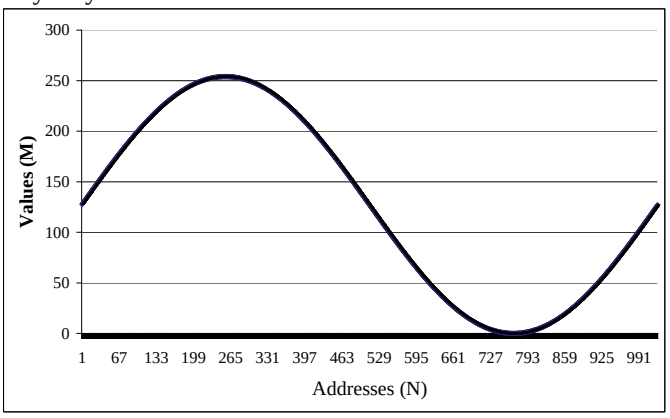

Fig. 5. Output signal, defined by the calculated and rounded values for the amplitude at N_{max}=1024

In Quartus II a new file is opened with *File/New/Other Files/Memory Initialization File*. We define the size of the MIF file, in our case 64 addresses, 4-bit cells. An empty MIF

TABLE I
CONTENTS OF THE LUT

N	Y_{ROM}	Y^r_{ROM}	N	Y_{ROM}	Y^r_{ROM}
0	7.68612	8	32	6.31388	6
1	8.365632	8	33	5.634368	6
2	9.031993	9	34	4.968007	5
3	9.678784	10	35	4.321216	4
4	10.29978	10	36	3.700223	4
5	10.88899	11	37	3.111008	3
6	11.44075	11	38	2.559247	3
7	11.94975	12	39	2.050253	2
8	12.41107	12	40	1.588927	2
9	12.82029	13	41	1.179713	1
10	13.17345	13	42	0.826551	1
11	13.46716	13	43	0.532843	1
12	13.69858	14	44	0.301418	0
13	13.8655	14	45	0.134503	0
14	13.96629	14	46	0.033707	0
15	14	14	47	0	0
16	13.96629	14	48	0.033707	0
17	13.8655	14	49	0.134503	0
18	13.69858	14	50	0.301418	0
19	13.46716	13	51	0.532843	1
20	13.17345	13	52	0.826551	1
21	12.82029	13	53	1.179713	1
22	12.41107	12	54	1.588927	2
23	11.94975	12	55	2.050253	2
24	11.44075	11	56	2.559247	3
25	10.88899	11	57	3.111008	3
26	10.29978	10	58	3.700223	4
27	9.678784	10	59	4.321216	4
28	9.031993	9	60	4.968007	5
29	8.365632	8	61	5.634368	6
30	7.68612	8	62	6.31388	6
31	7	7	63	7	7

file appears, in which we fill in the calculated with the program Excel values, and we save the file.

- *Simulation and experiments with the design*

The simulation of the digital part of the design has been made (Fig. 6), and also experimental study of the DDS synthesizer operation as a whole (including DAC and LFP), proving its proper operation.

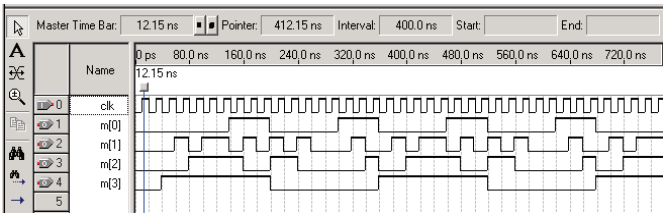

Fig. 6. Output waveforms passed to the DAC

V. CONCLUSION

The contributions of the present work are the following:

- Analysis of the design flow for creating devices and systems, based on FPGA;
- Designing the digital part of the sine wave frequency synthesizer;
- Implementing the synthesizer using the development system TREX C1.

The design is to be expanded as follows:

- Examining the noise sources and reducing their influence [4];
- Producing signals with various modulations – FSK, PSK, I-Q, etc.

REFERENCES

[1] A Technical Tutorial on Digital Signal Synthesis, Analog Devices, Inc., 1999.
[2] *Cyclone Device Handbook*, vol. 1, Altera Corp., 2005.
[3] H. Karailiev and V. Rankovska, "DDS Method for Generating a Frequency Grid at Systems for Test Control and Automated Regulation", ICEST 2006, Conference Proceedings, pp. 300-303, Sofia, Bulgaria, 2006.
[4] H. Karailiev and V. Rankovska. "Error Sources at Direct Digital Synthesis Signals", ICEST 2007, Conference Proceedings, Ohrid, Macedonia, 2007. (forthcoming)
[5] *Quartus II Version 5.0 Handbook*. Vol. 1: Design & Synthesis, Altera Corp., 2005.
[6] *TREX C1 Development Kit Getting Started User Guide*. Terasic Technologies Inc., 2005.
[7] V. Rankovska, "FPGA Families, Features, Resources, and Devices and Systems Design Technology", Unitech 2006, Conference Proceedings, pp. I-202 – I-207, Sofia, Bulgaria, 2006 (in Bulgarian).