FPGA Implementation of a Parameterized Fourier Synthesizer

Rui Yang, J.G. Wang, Benoit Clement, Ali Mansour

FPGA Implementation of a Parameterized Fourier Synthesizer

R. YANG* †, J.G. WANG*, B. CLEMENT† and A. MANSOUR†
*College of Engineering, Ocean University of China, Qingdao 266100, China
Email: yang.rui@ensta-bretagne.fr, ckyiqi@ouc.edu.cn
†Lab-STICC, UMR CNRS 6285, ENSTA Bretagne, 29806 Brest Cedex 9, France
Email: benoit.clement@ensta-bretagne.fr, Mansour@ieee.org

***Abstract*— Field-Programmable Gate Array (FPGA) offers advantages for many applications, particularly where missions are complex and time performance is critical. For small-production digital acoustic synthesizers, FPGA can achieve the above-mentioned tighter system requirements with low total system costs on single chip. In this manuscript, a real-time acoustic synthesizer is implemented using Fourier series algorithm on Altera's Cyclone II FPGA chip. This work emphasizes systematic designs and parallel computations. The proposed system includes a flexible processor and a parallel parameterized acoustic module. On one hand, the Nios II embedded processor, which is relatively low-speed component, is used to generate commands and configure high-speed acoustic module parameters. On the other hand, acoustic module which should require high-speed components contains 4 parallel architectures to gain high-speed simultaneous calculus of 4 independent digital timbres. Every timbre is equivalent to 16 parallel high-precision harmonic channels with 0.3 % frequency error. Experimental results corroborate the fact that a single FPGA chip can achieve complex missions and attain real-time performances.**

I. INTRODUCTION

Since the last two decades, FPGA has become widely used in electronic engineering applications after its first product in 1984. Comparing with Application-Specific Integrated Circuit (ASICs), FPGA is considered as the best technical solution for small-production applications. According to [1], first FPGA circuits are slower, less energy efficient and larger size than fixed ASICs. However, FPGA has many advantages: Flexibility, reconfigurability and low-cost. New technology and devices enhance these features and make FPGA more and more popular in many electronic applications.

It is well known that FPGA has the ability to implement any combinational function implemented on ASICs. Programming FPGA to form complex system logical functions can be achieved by interconnecting Look-Up Table (LUT) elements on a single chip [2].Comparing to digital computer architectures, such as Von Neumann architecture or Harvard architecture [3], FPGA can provide more flexibility and achieve better calculation performance. On one hand, FPGA allows designers to make serial and parallel mixed designs that are more flexible for complex missions. On the other hand, parallel designs can produce more simultaneous calculus for high-speed applications. In this paper, we focus on the case of digital acoustic synthesizers.

The primary aim of a synthesizer is to generate different acoustic timbres. Each timbre can be distinguished from others by the quantity of different harmonics. Acoustic synthesizers can be divided into two kinds: Analog and digital synthesizers.

Actually, analog synthesizer applications have good sound quality and are based on many well-defined algorithms, such as Frequency Modulation (FM) which is popular and simple to be implemented on electronic devices. Recently, more convenient digital approach emerges with similar ideas. Main advantages of digital synthesizers comparing analog ones are robustness, cheap prices and simple architectures.

Two popular digital synthesizers focus on Direct Digital Synthesizer (DDS) [4] and FM synthesizers [5]. The crucial technique in DDS is the phase-amplitude conversion architecture [6]. The main drawback of DDS is still the larger size of needed ROM tables for higher performance. FM approaches do not suffer from these inconvenient and they intend to modulate the frequency of carrier wave for generating the timbres. In fact, FM methods can save more hardware resources than DDS. The major drawback of FM techniques is still the need for an expert to tune the multitude parameters. To evaluate the ability of FPGA to handle a large amount of data, we select the Fourier series, namely an extended DDS application, instead of FM which can be easily implemented on FPGA. Our work challenges FPGA on special aspect of solving complex missions and parallel calculations. It is worth mentioning that the Fourier series designs don't require the knowledge of experts and parameters can be tuned automatically.

Using Fourier series, we can decompose a target timbre $F(2\pi f_0 t)$ as follows:

$$F(2\pi f_0 t) = A_0 + \sum_{i=1}^{\infty} A_i sin(2\pi i f_0 t + \varphi_i) \qquad (1)$$

where f_0 represents the frequency of the target timbre,A_i and φ_i represent respectively the amplitude and phase of i^{th} harmonic. A_0 is the average signal value. From the engineering point of view, equation (1) need to be truncated at a finite order of harmonic instead of using infinite frequency components. In our case, all the high order harmonics that have less

[1]R. Yang is a joint PhD student between ENSTA Bretagne and Ocean University of China.

[2]This work was supported by the China Scholarship Council.

energy than human hearing threshold should be ignored in the synthesizer implementation. According to [7], the sum in equation (1) can be truncated in worst case up to the 10^{th} harmonic. To get better precision, we assumed the sum truncated at the 16^{th} harmonic is rich enough to generate a single timbre.

In order to implement Fourier series on FPGA boards, we designed a complete system which includes three main parts: Human-Machine Interface, Coordinator Module (Embedded Nios II Processor [8]) and Acoustic Module. The Nios II Processor and the Acoustic Module are implemented on a single FPGA chip; further details are given in the following section.

II. System Design

As it was early mentioned, the implementation of a complex application, such as acoustic synthesizer, on hardware components is still very challenging. Serial as well as parallel structures are both required. In the following, the whole system architecture is described and the Timbre Module using parallel structure and requiring high computational efforts is emphasized.

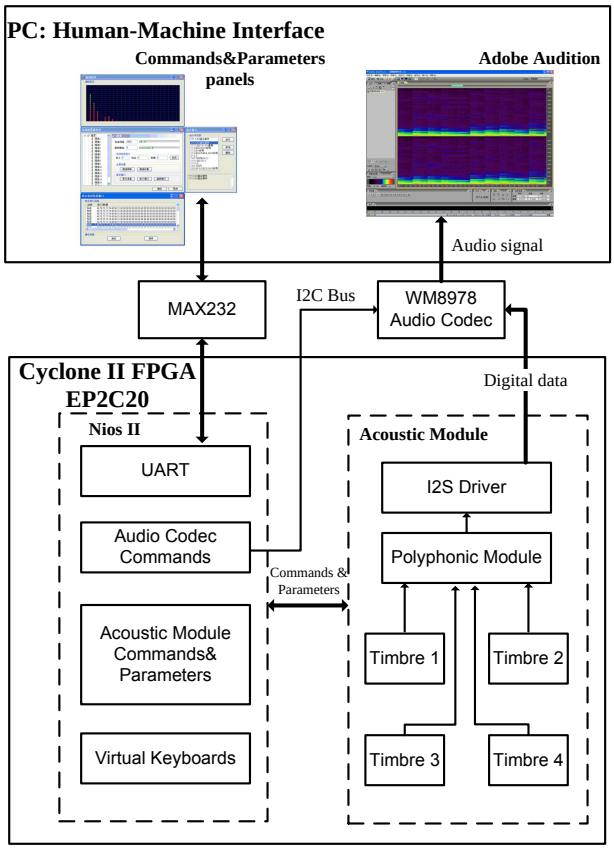


Fig. 1. System Diagram of Fourier Synthesizer

A. System Architecture

In order to generate 16 harmonics at each of the implemented timbres, our system can be divided into three major parts, (the first one is realized using a PC, the two other using FPGA), see Figure 1:

1) Human-Machine Interface: This part is done using a classic PC architecture and it should handle the human commands, sound generation and recording,sound analysis tools, etc.
   The synthesizer user interface, Commands and Parameters Panel, is developed using a VC++ software to monitor and manage the synthesizer (For example, users can modify frequency or amplitude parameters on-line via this interface). The audio analysis tools have been done using available commercial software, i.e. Adobe Audition.
2) Embedded Nios II Processor: Flexible Nios II embedded processor is relatively low-speed comparing to parallel architecture as it runs codes in serial sequence. In fact, its ultimate processing speed is limited by the CPU clock rate. However, Nios II shows better flexibility and robustness in responding to random environmental changes with interrupt mechanism which are ideally for human-machine interface tasks. In our system, Nios II is dedicated to communicate with the PC, to manage audio codec, to respond to keyboards and to configure acoustic module.
3) Acoustic Module: High-speed Acoustic Module can reach real time performance by mean of parallel architectures which can solve large calculations in short time. In the Acoustic Module we have 4 independent Timbre Modules which simultaneously generate 4 different timbres. The real time timbre data is processed in the Polyphonic Module to get a 32 bits polyphonic audio stream. The I2S Driver converts digital audio stream into serial I2S [9] format and finally analog audio signal is generated using an audio codec (such as WM8978).

B. Timbre Module

Previously, we mentioned that a timbre can be synthesized as a truncated sum of equation (1). In this section, we focus on the realization of the j^{th} timbre by electronic circuits. By adjusting A_0 to be zero, adding the shape of ADSR ($\mu_j(t)$, Attack-Decay-Sustain-Release envelope techniques are used to reshape signal in time domain to mimic the acoustic sounds of real instruments.) and multiplying by the strength parameters (σ_j), equation (1) can be rewritten as:

$$V_j = \mu_j(t)\sigma_j \sum_{i=1}^{16} \alpha_{ij} sin(2\pi f_{ij} t + \varphi_{ij}) = \mu_j(t)\sigma_j T_j \quad (2)$$

In this case, each timbre is characterized by three main features:

a) Harmonics: Up to 16 harmonics.
b) Strength (σ_j) is equivalent to volume parameters.
c) Attack-Decay-Sustain-Release envelope $\mu_j(t)$.

Each of the above features is implemented in different stage of the pipelined Timbre Module, see Figure 2.

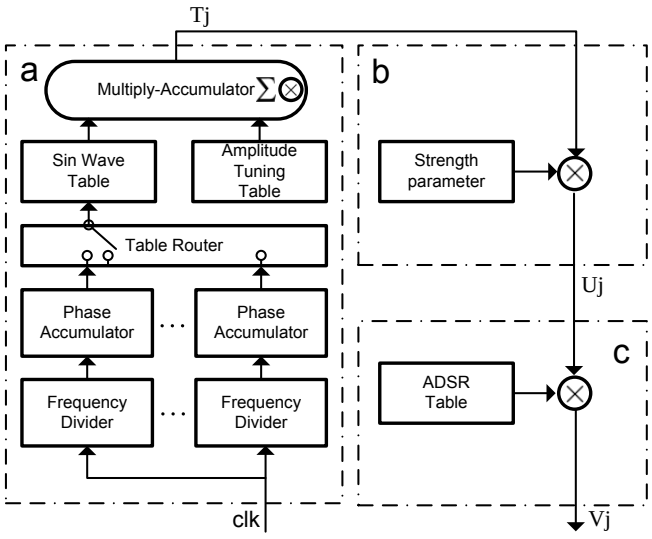


Fig. 2. Diagram of Timbre Module

Concerning the first feature, we need to generate T_j, as described in equation (2). To generate the main sinusoidal wave, we may use paralleled conventional DDS concept as shown in Figure 3.

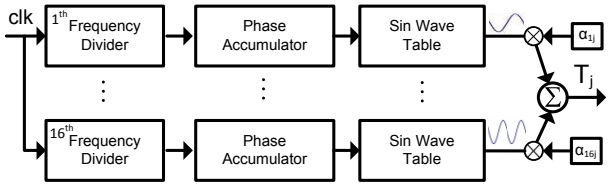


Fig. 3. Simple Fourier Synthesis by Sinusoidal DDS

In conventional DDS method, Frequency Divider is a digital counter that divides system clock by a chosen factor into desired frequency. Phase Accumulator is another digital counter that generates addresses to activate the RAM table which used to generate Sin Wave Table (SWT). The output T_j of the DDS component is obtained as the sum of all partial outputs. It is worth mentioning that the above structure becomes very expensive as it used 16 SWT and 16 multipliers per Timbre Module. This problem is a major one because the FPGA resources are limited.

To solve this problem we proposed another time-division multiplexing (TDM) structure to share one SWT for all the 16 parallel channels. As shown in Figure 2 (a), Frequency Dividers and Phase Accumulators still work in parallel mode, because they require high speed processing and their operating clock rates are usually comparable to the Nios II processor clock. SWTs of the conventional DDS on the contrary run at comparatively lower speed. They are reduced into one sequential access table in our TDM architecture. In order to prevent access conflict among channels, a Table Router was added to control the access priority. The added router can solve half of the problem, because using the router we can't obtain the total sum of the sinusoidal waves as described in equation (2). The total sum can be however obtained by using the Multiply Accumulator (MAC), seeing in previous Figure 2 (a). In our architecture, each Timbre Module will only consume one SWT , one extra small size Amplitude Tuning Table (ATT) and one MAC.

To provide the details of this parallel and sequential mixed structure, we separate the TDM architecture into Process Control Channel (PCC) and Data Processing Channel (DPC), seeing in Figure 4.

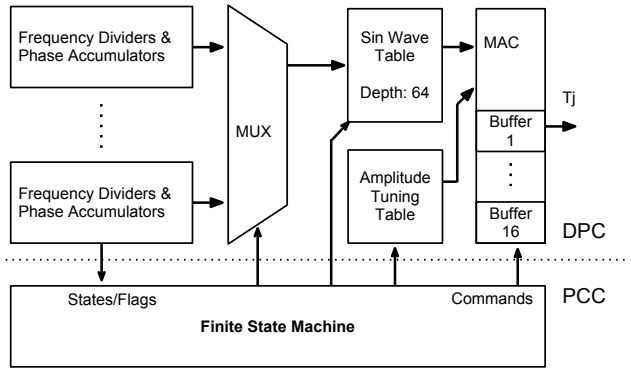


Fig. 4. SWT and MAC Shared Fourier Synthesis

PCC consists of a finite state machine, its function is detecting and coordinating the workflow of DPC. It gathers states information of the DPC, especially the Flags posted by the parallel counting channels. This Flag is triggered to apply for access of Sin Wave Table. When PCC recieve a Flag, PCC send coordinating orders to MUX, SWT, ATT and MAC in the piplined DPC to make a calculation cycle.

DPC is used to receive orders from PCC and process data as fast as possible. It separates into a parallel region and sequential region, they deal with high speed missions and low speed missions respectively. 16 parameteried channels run simultaneously and independently. This design gains more speed in calculating the sum in equation (2). Whenever a channel finishes counting with a result of table address, it will require PCC to make a sequential calculation cycle. PCC is the coordinator and core mechanism connecting parallel with serial.

The PCC follows the mechanism that low frequency channels get higher priority to Sin Wave Table. In each calculation cycle, MUX connect one channel to SWT, and corresponding Phase Accumulator provide an address pointing to the SWT memory unit, which is current normalized sinusoidal amplitude. Because every harmonic channels share the same sine wave that discreted into 64 points in length and 16 bits in amplitude. Therefore, ATT is essential to provide different amplitude for different harmonics. This coefficient from ATT is multipied with SWT result and storaged in the corresponding buffer in MAC. Then PCC ask MAC to update the sum to the following circuits. After the PCC clear the

responded Flag, it will be ready for the next cycle. This kind of pipeline structure is also used to add Strength and ADSR parameters.

.

III. Experimental Results

Many experiments have been conducted in order to evaluate operational properties and performances of the proposed Fourier synthesizer (such as high-speed, high-precision, real-time and flexibility). We also implemented a test board according to Figure 1 and Tab I . FPGA design software Quartus II 9.0 [10] is used to make a logic synthesis.

TABLE I
SYNTHESIS PARAMETERS

Experimental parameters	Quantity
Logic elements consumption	60%
total Memory consumption	27%
9 bits Multiplier	44
System clock	50 Mhz
Frequency error	$< 0.3\%$
Output bandwidth	20-20kHz

Step-by-step experiments were carried out to test frequency-amplitude accuracy and the functionality of our ADSR. In the test of frequency-amplitude accuracy,we compared the output signal frequency spectrum with an off-line pre-prepared target signal. The match between the two signals is accepted if they share the same first five harmonics, as shown in Figure 5 that indicates a good accuracy in both frequency and amplitude of each harmonic.

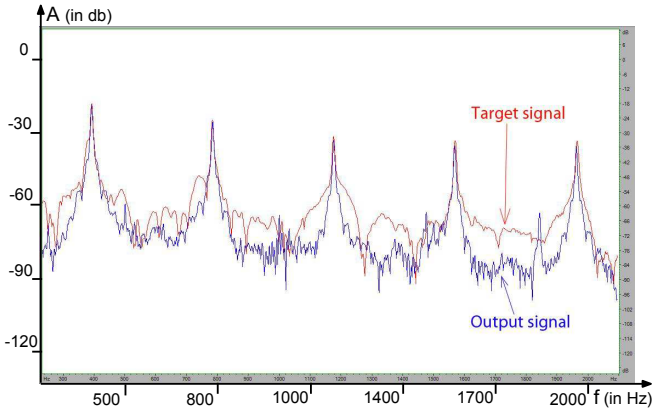
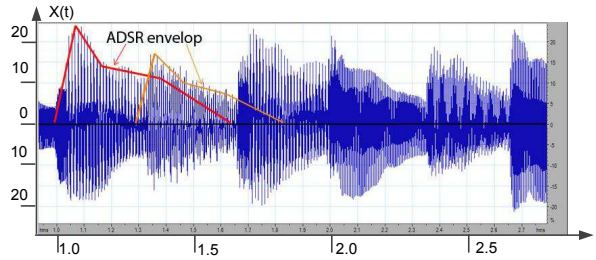
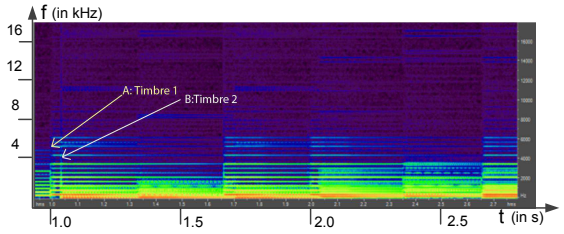


Fig. 5. Frequency accuracy analysis

To confirm the functionality of every component, we played several music compositions on our synthesizer. The output signal in time-domain and its frequency domain are shown in Figure 6. In Figure 6(a), the time-domain data show an obvious shape of ADSR envelop and strength vary which prove the functionality of the proposed Timbre Module. Meanwhile, its time-frequency representation, Figure 6(b), shows the overlap of 2 timbres (A, B). The experimental studies show that the proposed synthesizer can perform all tasks in real-time manner.



(a) Time-domain representation, $x(t)$ in % of maximum audio output level



(b) Time-frequency domain

Fig. 6. Time domain signal and its frequency spectrum

IV. Conclusions

In this paper a systematic design of Fourier Synthesizer is proposed and implemented on a single FPGA chip. The proposed circuit, especially the parallel design of Timbre Modules, shows efficient flexibility and large calculation capabilities in achieving real-time, high-speed and high-precision applications, as the implementation of digital acoustic synthesizer in FPGA field. We should highlight the fact that the small-size and low-cost FPGA boards are efficient to perform small-producion applications where tasks are complex and time performance is critical.

References

[1] I. Kuon and J. Rose,"Measuring the Gap between FPGAs and ASICs," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 26, NO. 2, Feb. 2007, pp. 203 - 215.

[2] K. Compton and S. Hauck, Reconfigurable computing: A survery of system and software, ACM Computing Surveys, vol. 34, no. 2, pp. 171210, Jun. 2002.

[3] Steven W. Smith. The Scientist and Engineer's Guide to Digital Signal Processing, 2nd ed, California Technical Publishing, San Diego, CA, 1999.

[4] V. Kroupa, Direct Digital Frequency Synthesizers, IEEE Press, Piscataway, NJ, 1999.

[5] J. Chowing, The synthesis of complex audio spectra by means of frequency modulation, J. Aud. Eng. Soc., pp. 526529, Sept.1973.

[6] J. Vankka, Digital Synthesizers and Transmitters for Software Radio.Dordrecht, Springer, 2005.

[7] L. Qiwen, L. Qiwu. Principle of Keyboard Maintenance. Electronic Industry Press, China, 1991.

[8] Altera Inc. Nios II Software Developer's Handbook, Document Version: 11.0, May 2011.

[9] I2S bus specification, Phillips Semiconductors, 1996.

[10] Altera Inc. Quartus II Handbook, Document Version: 12.1.0, Nov 2012.