DOI: 10.21917/ijme.2017.0070

IMPLEMENTATION OF INNOVATIVE LOW COST MUSIC SYNTHESIZER USING FPGA

Marmik Soni[1] and Piyush Makharia[2]
Department of Electronics and Communication Engineering, Dharmsinh Desai University, India

Abstract

Music synthesizer opens enormous possibilities of generation of new music by employing number of innumerable combinations. In this work, implementation of music synthesizer is done with the use of digital logic concepts. It is employed on FPGA platform with the use of verilog hardware descriptive language. The work is aimed to make this innovative design very adaptive, scalable and highly miniaturized. By sophisticated algorithm implementation of digital logic blocks, full octave notes (Sa Re Ga Ma Pa Dha Ni Sa) are generated and tested. Implementation is analyzed on Cyclone EP1C6Q240C8, Cyclone II EP2C20F484C7 and Stratix EP1S10F780C6ES FPGA families. Design performance is investigated with above logic devices for area, power consumption and timing constrains. Design is capable of automatic octave tone generation as well as manual key press tone generation.

Keywords

Digital Logic Design, Fourier Series, FPGA, Music Synthesizer, Octaves, SoC, Tone Generation, Verilog HDL

1. INTRODUCTION

Music synthesizer mechanism generally is based on the process of playing with the coded record produced by a musician, musical engineer, or composer with a fundamental understanding of the composition of sound. The electronic music synthesizer provides means for the production of a tone with any frequency, intensity, growth, duration, decay, vibrato, and variation [1]. Synthesizers are popular because it opens an entirely new field for the production of music, which is limited by a user with a few key combinations in conventional music player [2].

Concept of Fourier Series are used here for utilizing effortlessly available square waves for the generation of music tone frequencies. All music generating devices are based on 7 notes Sa Re Ga... Exploring richness of digital logic design for manipulation and generation of above mentioned tones can be quite handy and expedient. This system uses Programmable Logic Device.

FPGAs are current day proficient candidate in PLD family for this kind of implementation. Various frequencies can be generated and manipulated by highly densed electronics with sophisticated supportive logic. Various promising features are re-configurability, faster operation, and miniaturized low power implementation with SoC tactics. Advancements in digital logic design this days has got elevated a lot. So, that can come to help for advanced manipulation of design concept implementation.

The paper constituted of 4 sections. Section 2 discusses fundamentals of Fourier series used. In section 3 music theory fundamentals used in the implementation are used. System Design is discussed in section 4, which also discusses full design with the help of block diagrams. Implementation of the design is discussed in section 5. It also features design specifications, waveforms and comparative analysis when used different logic family devices. Last section is about conclusion of the work presented with objectives achieved.

2. FOURIER SERIES FUNDAMENTALS

Like all signals available in nature, audio signals are also analog in nature, but equally complex to deal with all the flexibility and strong suit of digital logic design. According to the concept of Fourier series any signal can be represented in terms of combination of Sine and Cosine waves. The square wave is a straightforward waveform in the time domain, as it simply switches between its maximum and minimum peaks with near infinite slope. In the frequency domain, the signal is much more complex, containing only odd harmonics of the fundamental frequency, with the subsequent higher frequencies of declining amplitude. This has inspired us to use square waves as audio signals. Eq.(1) depicts Fourier series representation.

$$g(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos 2\pi f_n t + b_n \sin 2\pi f_n t \right) \quad (1)$$

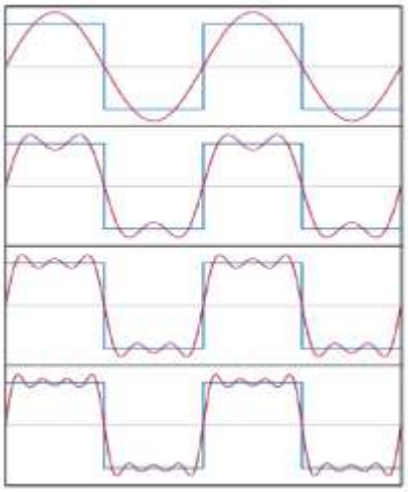

Fig.1. Fourier representation of square waves

The Fig.1 shows the realization towards square wave as we keep adding odd harmonics in fundamental frequency [5]. Relation between square and Sinusoidal wave can clearly be understood with the help of Eq.(2).

$$x(t) = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{\sin 2\pi ft(2n-1)}{2n-1} \quad (2)$$

Initially relative performance is observed just with square wave on audio transducer. Adequate performance achieved at this stage, later got enhanced with the implementation of square to sine wave converter.

3. FUNDAMENTALS OF MUSIC THEORY

A swara or music note usually denotes the note name indicating the pitch, duration and octave. The notes are named differently according to its pitch as shadja (*sa*), rishabha (*ri*), gandhara (*ga*), madhyama (*ma*), panchama (*pa*), dhaivata (*dha*), and nishada (*ni*) and is abbreviated as given in brackets [3]. Basic music instruments like Piano, Tanpura can be understood or learnt with the fundamentals of 7 notes. Details of the frequencies which can be correlated with notes *Sa* : *Ga* : *Pa* are on the basis of ratio 4 : 5 : 6. With the achieved note frequency of Pa, here it gives the relation for remaining notes, i.e. *Sa* : *Ga* : *Pa* :: *Pa* : *Ni* : *Re* :: 4 : 5 : 6. If the frequency for Sa note if considered to be 200Hz. Frequencies of *Ma*: *Dha* : *Sa*' :: *Sa* : *Ga* : *Pa* :: 4 : 5 : 6. *Sa*' belongs to next octave which is double than that of *Sa*. Thus all he frequencies desired for generation of full octave is mentioned in Table.1. It is to be noted that here all the work is done for Shuddha notes. Principle of "octave-reduction" has also been employed here to restrict all the note frequencies of octave between *Sa* and *Sa*'.

Table.1. Octave Note Frequencies

Octave Notes	Frequency (Hz)		
	By calculation	After Octave Reduction	Obtained
Sa	200	200	198.6
Re	450	225	226
Ga	250	250	254.3
Ma	266.6	266.6	265
Pa	300	300	301
Dha	333.3	333.3	338.3
Ni	375	375	381.4
Sa'	400	400	397.2

4. SYSTEM DESIGN

The system design is divided in two parts. (i) Generation Logic (ii) Control Logic. Generation Logic is explained in Fig.2. The Fig.3 describes Control Logic.

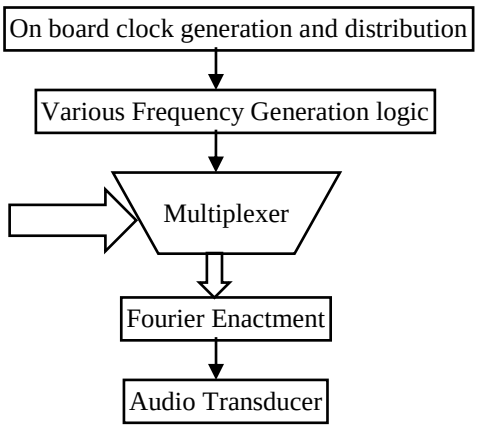

Fig.2. Generation Logic

System employs on board clock frequency generation and distribution logic which uses oscillator circuit and PLL block. Output of this block goes to frequency generation logic block. Various mathematical processes are done to achieve all different frequencies, which gives all octave notes frequencies as discussed. The processes include frequency division techniques with the help of many basic digital logic design blocks like counters, buffers, multiplexers etc. Control logic decides which octave note is to be given to the further portion of system. Control logic has the option of choosing sequence of notes or particular note to be given to next part. In next part audio transducer input conditioning is done. Which is simply the conversion of obtained signal from discrete levels to continuous levels. Which is sinusoidal wave in this case. If sequence mode is chosen then it plays one by one all the notes and if manual mode is selected than it gives output as per the key being pressed by the musician, which is similar to the instruments like piano.

Selection of manual mode or sequence mode. In manual mode, the output of the audio transducer will be as per the key being pressed by the musician, which will be one of the notes being played from the set of octave.

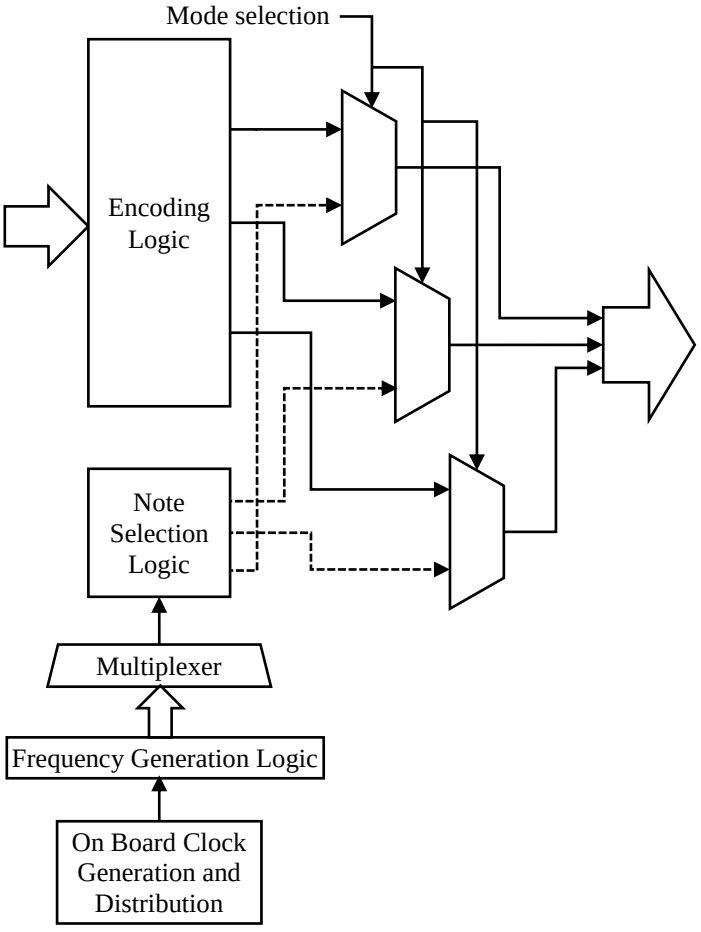

Fig.3. Control Unit

When selected this mode the all octave notes are being played in sequence. The duration for which individual octave notes are being played can also categorically be decided by the composer. Predefined finite durations for this selection is available in this design. There are four different frequencies at which the note output gets flipped, which are to be selected by the user manually.

Pause time is also provided in the system design after completion of each set of notes, for this particular mode.

Audio transducer converts electric form of energy in to acoustic energy. Here output of enactment block is directly fed to audio speaker. It makes the system low cost and less complex. Further for better output power amplifier and refined system of speakers can be employed. Here simple 8Ω, 0.5Watt speaker is used.

5. IMPLEMENTATION

Thanks to the rapid evolution of CMOS technology, the ASIC design paradigm shifted from VLSI to SoC. In parallel, programmable logic has progressed from being used as glue logic to today's well advanced FPGAs, where complete system can be implemented in a single device [4]. Being prominent player with the concern of size, speed and area in the field of digital logic design, FPGAs are very prevalent. System employment is done with the help of various families which are compared in Table.2 for a number of parameters. Implementation of the design is tested with DE-I board. The board uses Cyclone II EP2C20F484C7 family. Verilog HDL has evolved as a standard hardware description language, it also offers many useful features for hardware RTL design [5]. Features and modelling techniques of Verilog HDL are employed here extensively to achieve the design objectives [6] [8].

Various clocks are available on board FPGA. In case of Cyclone II family, on board clock used is of 50MHz. For remaining device logic families', clock of 50MHz can be given externally through GPIO pins. All note frequencies are generated with sophisticated frequency generation logic techniques. In Fig.2 this process is done by frequency generation logic block. The output here is of seven notes. With the input selection at the control logic part one of the note output will be given to Fourier Enactment block. Fourier enactment is done by using 2nd order low pass filter. The Eq.(3) shows the implementation of that filter as Fourier enactment block. This process removes higher frequency odd harmonics from square wave. The filtered output is fed to audio transducer i.e. speaker in this case being simplest of the audio output device. In case of this implementation input for enactment block is received from one of the pins available in GPIO header. Control Logic part has employed two modes as explained earlier. This mode selection is manually chosen by musician.

$$f_c = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}} \quad (3)$$

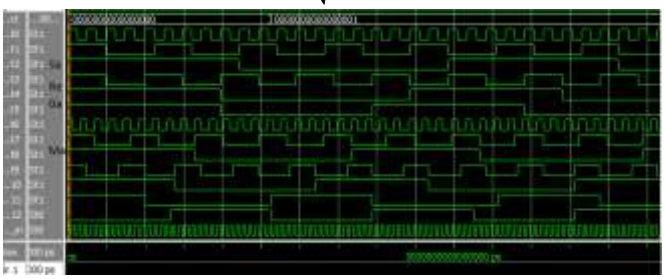

Fig.4. Generation of the Sa Re Ga Ma notes

The Fig.4 shows the output of *Sa Re Ga Ma* note frequencies, which can be verified from the time scale. Similarly, Fig.5 shows output of *Pa Dha Ni* octave frequencies obtained from the Generation Logic block.

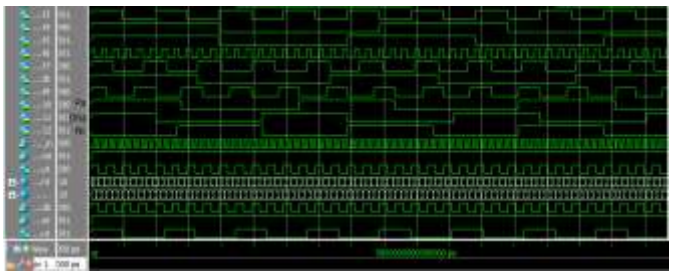

Fig.5. Generation of the Pa Dha Ni notes

5.1 MANUAL MODE

For manual mode key pressed is controlling the selection of particular note at that instance. Encoder is used for doing it, where k_1 to k_7 are keys for the music instruments and output becomes selection lines for the note to be played.

5.2 SEQUENCE MODE

In this mode being selected, all notes are played in sequence after a fixed duration depending on the time interval fed by user. The Table.2 explains timing interval options for which single note will be played for.

Table.2. Timing Interval Selection in Sequence Mode

Interval Selection Control		Time Interval Selected (s)
A_0	A_1	
0	0	0.67
0	1	1.34
1	0	2.68
1	1	5.36

The Table.3 shows the result analysis of the design implementation with different logic families. Cyclone is the first low-cost product produced in 0.13µm, there are no other low-cost products to which we can fairly compare. To highlight the achievements of the Cyclone architecture, we are providing a comparison with Stratix, a full-featured FPGA produced on the same process technology. The Stratix has far more memory, I/Os and other high-end features. So it definitely becomes more power consuming. If speed is the constraint than Cyclone is better as shown in Table.3, due to same reason [7]. It is to be noted that Cyclone II is based on 90nm technology. It also has highest logic density as compared to other two discussed. Also the detailed comparison of performance is shown in Fig.6, when implemented on three different FPGA platforms for total thermal power dissipation, core dynamic thermal power dissipation, core static thermal power dissipation and I/O thermal power dissipation.

Table.3. Result analysis of design with different logic families

Device Logic Family	tpd (ns)		Logic Element usage	Total Power Consumption (mW)
	Sequence Mode	Manual Mode		
Cyclone EP1C6Q240C8	10.993	27.474	3% 184/5980	60.18
Cyclone II EP2C20F484C7	7.282	21.845	< 1% 193/18752	70.02
Stratix EP1S10F780C6ES	8.103	21.561	2% 175/10570	187.50

Use of FPGA proves the design to be highly scalable, low power, high speed and low cost implementation. Design idea is very innovative where such synthesis of music is a novice experiment as it employs square waves for the music generation. PLD output being connected to Fourier enactment block. Testing of such prototype systems on FPGAs make the designing process more reliable and highly cost effective as compared to ASIC implementation [9]. With any of the modern developments in algorithms, FPGAs can be reprogrammed even on board, whereas ASIC is needed to be replaced. This makes the choice of FPGA over ASIC very cost effective [10]. This design is implemented with three different FPGA logic families, where the consumption is below 3%. Design complexity can be increased with the remaining capability to attain various features.

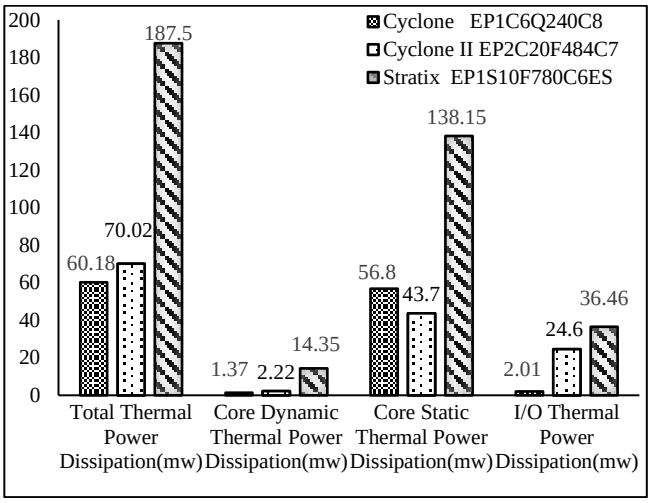

Fig.6. Power Dissipation analysis across FPGA families

6. CONCLUSION

Use of FPGA proves the design to be highly scalable, low power, high speed and low cost implementation. Design idea is very innovative, where such synthesis of music is a novice experiment as it employs square waves for the music generation. PLD output is connected to Fourier enactment block. Testing of such prototype systems on FPGAs make the designing process more reliable and highly cost effective as compared to ASIC implementation. The design implementation with FPGA board has given real time output and also gave clear idea about perception of notes highly similar to music instruments. Being scalable implementation this design is capable of increasing to multiple octave implementation instead of single octave within same design. In sequence mode adding extra clock intervals for increasing scope of music types can be easily done. Thus, it has the possibility of inclusion of extra notes in both sequence mode and manual mode of operation. Marginally the design performance gets better with Cyclone II. If considered Stratix family than power consumption increases by more than two folds. This also proves that even the very old low cost, classic FPGAs Cyclone or Cyclone II, also suffice the requirement of this design. Thus, the objective of implementation of music synthesizer to provide the musician, musical engineer, and composer with a new musical tool with no inherent physical limitations is successfully achieved. Lower consumption of logic capabilities on FPGA in this design also has the possibility to increase complexity to accommodate sophisticated and advanced functionality.

REFERENCES

[1] David Sarnoff, "New Developments in Electronics", *Electrical Engineering*, Vol. 74, No. 3, pp. 179-183, 1995.

[2] Hary F.Olson and Herbert Belar, "Electronic Music Synthesizer", *The Journal of the Acoustical Society of America*, Vol. 27, No. 3, pp. 595-612, 1995.

[3] Stanly Mammen, Ilango Krishnamurthi, A. Jalaja Varma and G. Sujatha, "iSargam: Music Notation Representation for Indian Carnatic Music", *EURASIP Journal on Audio, Speech, and Music Processing*, Vol. 2016, No. 1, pp. 1-12, 2016.

[4] Amara Amara, Frederic Amiel and Thomas Ea, "FPGA Vs ASIC for Low Power Applications", *Microelectronics Journal*, Vol. 37, No. 8, pp. 669-677, 2006.

[5] Samir Palnitkar, "*Verilog HDL: A Guide to Digital Design and Synthesis*", Vol. 1, Prentice Hall Professional, 2003.

[6] Hilary Weller, "Fourier Analysis", Available at: http://www.met.reading.ac.uk/~sws02hs/teaching/Fourier/Fourier_2_student.pdf.

[7] Paul Leventis *et al.*, "Cyclone/Spl Trade: A Low-Cost, High-Performance FPGA", *Proceedings of IEEE Conference on Custom Integrated Circuits*, pp. 49-52, 2003.

[8] Donald Thomas and Philip Moorby, "*The Verilog Hardware Description Language*", 5th Edition, Springer, 2002.

[9] Giovanni De Micheli, "*Synthesis and Optimization of Digital Circuits*", 1st Edition, McGraw-Hill Higher Education, 1994.

[10] Ian Kuon and Jonathan Rose, "Measuring the Gap between FPGAs and ASICs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 2, pp. 203-215, 2007.