P19

Direct Digital Synthesis

Kyle Harrison[†], Warren Fletcher [‡], Sean Le Roux[§] and Roan Song[¶]
EEE4084F Class of 2017
University of Cape Town
South Africa
[†] HRRKYL008 [‡]FLTWAR002 [§]LRXSEA001 [¶] SNGROA001

***Abstract*—A brief overview of Direct Digital Synthesis systems is described as well as the background to some DDS systems and DDS systems in general. A modified version of the traditional DDS system is presented for use within the audio range of frequencies. Every western note (12-tone) is designed and implemented on the Nexys4 DDR FPGA. Multiple waveforms (Sinusoidal, Square, Sawtooth, Triangle) are created with 1024 Samples, each at 9-bit resolution. The resulting waveforms are presented to the listener via PWM coupled with a Low Pass Filter. The output is compared against simulated results and was found to be successful.**

I. Introduction

This report conveys the design process and implementation of a Direct Digital Synthesizer (DDS). DDS is the process of reproducing an analog signal using digital values stored in a Look-up-table (LUT).

DDS systems are used in a multitude of fields, such as: Biomedical, to locate the resonant frequencies or compensate for temperature drift that can occur with electronic sensors and systems; RF communications systems, for fast dynamic frequency sources; A reference for a phase-lock-loop (PPL) to improve resolution and many more applications where accurate frequency generation is of top priority.

The chosen implementation is to control and sum the frequencies of multiple waveform outputs in order to reproduce high fidelity audio signals. This is achieved by aggregating a number of waveforms, at various frequencies, to form one analog signal output waveform via PWM. Control over the sinusoidal frequencies is done via the Nexys 4 board's build in switches and buttons. As the Nexys 4 DDR board has an on board PWM audio amplifier the LUT digital output is converted into a PWM signal and passed through a fourth order low pass filter [1], which filters out noise and smooths out the produced analog signal.

These outputs can be compared to their simulated equivalents as well as external function generator signals to evaluate the degree of accuracy and quality of the produced signal.

In this report, DDS is used to implement the western 12-tone music standard with support for distinct increases or decreases in pitch in the form of octave manipulation.

A. Objectives

The objective of this project is to accurately document the process of re-creating 12 frequencies that correspond to the notes found in western music in equal temperament tuning (That is, the 'distance' between each frequency note is consistent) with support for octave and waveform adjustment:

1) Input
   - User selection of desired note (frequency), octave, and waveform
   - IO de-bouncing
   - User Feedback via the on-board 7-segment displays.
2) Processing Stage
   - Storage of different preset waveforms (sine, triangle, saw, square) in lookup table
   - Application of frequency adjustment
   - Send control signals to the output stage
   - Send processed waveform to output buffer
3) Output Stage
   - Read from the summing buffer and use and convert it to analog output via PWM
   - Ensure "flat" frequency response of amplifier
4) Stretch Goals
   - User interface via UART
   - Utilize the input analog signal as a reference signal. This involves using an external ADC system
   - Digital filtering of output signal to reduce noise
   - Loading input signal into buffer to be sent to the processing stage
   - Dynamic storage of input signal in lookup table
   - Peak/zero crossing detection to determine frequency of input signal
   - Adding a DC offset to input signal before ADC
   - Dual channel Output
   - Multiplexing inputs
   - Frequency and/or amplitude modulation of two signals

B. Motivation

DDS systems are becoming increasingly important in systems that require signal sources that have no disturbances and little to no noise [2]. A DDS system provides extremely fast and accurate tuning [3] while maintaining continuous phase with no overshoot or undershoot. The tuning of the system is implemented digitally and thus can be changed "continuously" while maintaining high precision and with relative ease. The frequency resolution is usually in the micro-hertz range, although creating accurate clocks for each DDS module based off the master clock can be problematic if the accumulator that forms the sub-clock is required to tick over at a non-integer divisor of the master. The digital nature of the system also allows the system to be tested and reconfigured remotely. Many unwanted analog electronic circuit properties don't effect the digital implementation, such as temperature, dust etc [2]. DDS systems are gaining support for solving frequency and waveform generation for communication systems and industrial applications. These system are packaged in IC chips which have low power consumption and reduced cost [3].

The disadvantage of DDS systems such as spurs which are due, mainly, to truncation effects in the NCO. Crossing spurs are due to the Nyquist sampling effects. Sampling effects such as quantization noise, aliasing, filtering must be considered. Further considerations are higher order harmonics that fold back into the Nyquist bandwidth, which cannot be filtered when using a digital implementation [4], as well as a higher noise floor at large frequency offsets due to the Digital to Analog Converter (DAC) [5]. These reasons make DDS systems undesirable, but methods for improving the quality of the output waveform are being developed [6].

II. Background

Due to the widespread use of digital techniques in instrumentation and communications systems, a digitally-controlled method of generating variable frequencies from a reference frequency source was developed called Direct Digital Synthesis (DDS) [4].

A. DDS Overview

Figure 1 shows an abstracted block diagram implementation of a DDS system. Samples of a periodic waveform to be output are stored in a LUT and an address for the LUT is stored in register, in Figure 1 it is the address counter but it is also known as a phase accumulator. This phase accumulator's value is incremented every clock cycle in order to traverse through the entire block of memory. The output of the LUT is stored in a register in order to be output through a DAC, where it is then filtered in order to minimize the effects of quantization that occur during the conversion from a digital waveform.

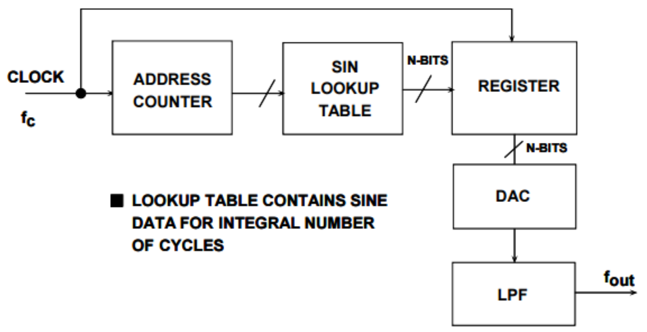

Fig. 1: Simple DDS Block Diagram [4]

This basic DDS system outputs a single frequency which is dependent on the clock rate and the size of the LUT. In order to dynamically adjust the frequency which is necessary for DDS, the size of the increment of the address each clock cycle requires adjustment. This is done through a tuning word M. Figure 2 shows how the nature of an overflowing address register can be viewed as a phase wheel traveling from 0 to 360 degrees at set intervals, where each interval is a sample's address. By adjusting the jump size or tuning word the rate at which the LUT is traversed is increased, as fewer samples are being read out.

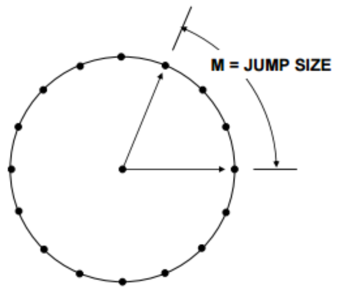

Fig. 2: Phase Wheel [4]

For an n-bit phase accumulator, there are 2^n possible phase points on the wheel. If f_c is the clock frequency, then the frequency, f_o, of the output waveform is equal to:

$$f_o = \frac{M f_c}{2^n} \tag{1}$$

An issue with this simple DDS system is that attaining certain frequencies is not possible, due to the phase wheel operating at fixed integer increments. If a frequency requires a fractional tuning word it will automatically be truncated due to the integer nature of registers, this introduces a phase truncation error causing a reduction in SNR and the requires frequency to not be obtained. The other major issue is a loss of fidelity of a high frequency output signal due to quantization introduced by leaving out a large number of samples from the large tuning word.

B. Types of DDS

DDS architectures fall into 6 major categories: pulse output, sine output, triangle output, phase interpolation, jitter injection, and fractional divider or pulse snatching DDS's [7].

1) Pulse Output DDS: This is the simplest DDS architecture. A tuning word is added to an accumulator register once every clock period. The frequency of the output is the frequency at which a carry occurs in the accumulator for a pulse output or the most significant bit, MSB, for a square wave. Where n is the bit size of the accumulator and M is the tuning word.

$$F = \frac{M}{2^n} \tag{2}$$

$$f_o = F f_c \tag{3}$$

2) Sine Output DDS: This DDS system adds a sine-wave LUT to the Pulse output DDS system and a DAC to convert the digital output to an analog output. This is the system described in the Introduction of this paper. The input of the DAC is determined by the value stored in the sine LUT at the address that the accumulator points to. Many methods of truncating the LUT have been developed, while still maintaining acceptable accuracy. Where x_th is the specified sample at an instance in time.

$$V_o = sin(2\pi F f_c x_t h T_c) \tag{4}$$

3) Triangle Output DDS: The output of a Pulse output DDS system passes through a bit compliment logic circuit after which it is converted to an analog signal using a DAC.

4) Fractional Divider or Pulse Snatching DDS:

This is a modification to the pulse output DDS. Every time a carry occurs in the accumulator block the value stored in the divider block, initially the clock frequency f_c, is divided by $n + 1$. The output of the divider block determines the rate at which the accumulator is clocked at, and the rate at which a carry occurs for a constant tuning word.

$$f_o = \frac{f_c}{n + F} \tag{5}$$

5) Phase Interpolation DDS: Whenever an output transition occurs in the Pulse output DDS or Fractional divider DDS, the value stored in the accumulator register R is proportional to the time or phase difference between the output transitions of the DDS and that of an ideal frequency generator. Therefore R is used to phase shift or delay the output of a pulse output or fractional divider DDS, but will result in lower phase jitter and spurs. A simplified version of this system, is called a microstepper and can produce small variations in the output frequency with extremely high resolution.

6) Jitter Injection DDS: A jitter signal is injected into a Pulse DDS system which reduces the size of the spectral spurs the output.

III. Methodology

This section outlines the spiral design approach used in order to define the system. It begins by broadly outlining what needs to be achieved through the systems specifications and goes on to define the functional requirements to better understand the possible implementation of these functions.

The final design relies on these key decisions defined in the functional requirements which are then used to confirm the system achieves the necessary goals. Throughout the process the advantages and disadvantages of each possible decision will be discussed in order to supplement the final design choice and provide an explanation for the systems functions.

The system must be designed in a modular fashion, this assists the nature of the verilog programming necessary to implement the FPGA system. Along with this, designing the system in such a manner using the spiral approach allows for each component to be independent and its performance and functionality to be tested and analysed. Finally this approach allows for easy expansion and added functionality to be integrated seamlessly.

A. Design Specifications

This section details the broad requirements the final implementation much accomplish in order to be successful. The functional requirements will be used to determine the methodology for possible implementations.

The following list defines the necessary functional requirements:

- User control over system behavior ie: which outputs to generate and when to do so
- Multiple fundamental notes (12)
- Octaves of fundamental notes
- Audio output
- Visual feedback ie: 7-Segment Display

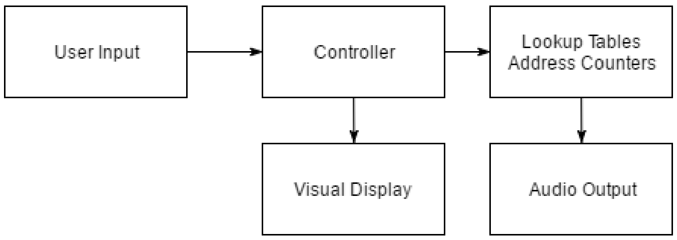

Fig. 3: Block diagram of necessary components

Figure 3 shows the block diagram form of each individual and independent component that the system is required to implement in order to function. By abstracting the required functionality into independent but linked blocks it simplifies the modeling of constraints as well as creating modularity of design.

B. Functional Requirements

This section outlines the more specific modules which the design will employ. It details how each specification can be achieved and the advantages and disadvantages of each possible implementation. These requirements will be used to decide on a final design implementation based on the pros and cons of each possible modules implementation.

1) User input:
   In order for the system to be dynamically modified once functioning on the FPGA and the full range of its abilities to be displayed it requires input from the user which can be done in the following ways.
   Design Decisions:
   - Ease of use
   - Intuitive interface
   - Ease of implementation

   Implementations:
   a) *Piano Keyboard:* As the DDS system is generating audio waveforms like an audio synthesizer the ideal input would be an audio keyboard.

  - Advantages: Automatically debounced, easy and intuitive interface intrinsically linked to music
  - Disadvantages: Expensive or difficult to obtain, requires advanced interfacing and communication through either and ADC, UART or Midi

   b) *Buttons:* The nexys 4 board has 5 built in buttons on the board.
  - Advantages: On-board, easy to implement
  - Disadvantages: Requires debouncing, only 5 available therefore limited abilities

   c) *Switches:* The nexys 4 board has 16 built in switches on the board.
  - Advantages: On-board, easy to implement
  - Disadvantages: Always on/always off. Not much customisablity for advanced envelope funtions

2) Lookup tables:
The nature of DDS is that it requires a wave table lookup stored digitally, this can be achieved in a variety of ways.
Design Decisions:
- Resolution
- Types of waveforms, fundamentals would be Sin, Saw, Triangle, Square

Implementations:

   a) *Pre-generated LUT:* A LUT can be generated using MatLab, output to a Xilinx .coe file, a .txt file or a .dat depending on the chosen implementation.
  - Advantages: Can be plotted and viewed/analyed for correctness
  - Disadvantages: Simple, fundamental waveforms

   b) *ADC Input:* An ADC can read in a sampled signal for a period of time and this can be the fundamental LUT.
  - Advantages: Customizable, complex
  - Disadvantages: Difficult to implement, added complexity

3) Memory:
Once the LUT have been generated it needs to be accessed from internal memory, there are multiple options for this.
Design Decisions:
- Access speeds, clock delays
- Size limitations
- Resolution, number of samples and bits per sample ie: width and depth

Implementations:

   a) *ROM:* Read only memory can be generated from an input file.
  - Advantages: Easy implementation using Xilinx IP Cores, design via Wizard, input via a MatLab generated .coe file.
  - Disadvantages: Less customisation, design is based on options avalialbe in the Xilinx ROM IP Core. Limited to Single/Dual input/output. This would require a different LUT for each note.

   b) *Registers:* The verilog synthesiser allows for the function readmemb which reads a block of data from a file into a two dimensional block of registers.
  - Advantages: Easy to implement, concurrent memory access
  - Disadvantages: Limited in size

4) Changing note frequency:
Once the LUT has been generated it needs to be accessed from internal memory, there are multiple options for this.
Design Decisions:
- Ease of expansion and integration
- Accuracy
- Modularity

Implementations:

   a) *Fractional address counter:* Implementing a fixed clock rate with a large LUT allows for development of an advanced address counter which uses a bit format implementing fractional bit representations.
  - Advantages: Allows for a single clock frequency for all notes, no chance of clock timing inaccuracies
  - Disadvantages: Difficult to implement, difficult to integrate the rest of the system to fractional bit format

   b) *Different clock frequencies per note:* By iterating over the same size LUT for each note at different speeds the output waveforms frequency is determined.
  - Advantages: Simplicity, each note would have a different downclock counter triggering increment, creates an indepdent module
  - Disadvantages: May introduce clock inaccuracies, if two note clocks are fast and close to the 100MHz clock their posedge triggers may create different timings

5) Audio Output:
In order for the synthesised waveform to be audible to the user an audio output is required.
Design Decisions:
- Fidelity of signal
- Ease of implementation

Implementations:

   a) *High Resolution DAC:* The ideal audio interface would be a high quality DAC.
  - Advantages: High fidelty audio limited only by resolution of LUT and quantisation
  - Disadvantages: Difficult implementation, no on-board DAC, would require digital output and possibly external circuitry

   b) *PWM:* The Nexys 4 has an on-board PWM audio amplifier.
  - Advantages: PWM is on-board and simply requires an enable signal via the constraints. Built in low pass filter at 10kHz
  - Disadvantages: 10kHz low pass filter is lower that 20kHz audible frequency

6) User display output:
In order for feedback to be provided to the user a display of the systems current parameters is necessary.
Design Decisions:
- Customisabilty/Possible outputs
- Ease of implementation and cost

Implementations:

a) *Screen:* The nexys 4 has a VGA/HDMI output that can be used for an external screen.
   - Advantages: Highly customisable and ability to display a lot of information
   - Disadvantages: Difficulty implementing

b) Seven Segment Display: The Nexys 4 has 8 seven segment displays available.
   - Advantages: Easy implementation as its on the board, all notes can be displayed in 7 seg format.
   - Disadvantages: Limited to seven segment format for advanced displays. Only 8 avaliable Difficulting to translate multiple unit binary format to hex for decoding

c) *LEDs:* The Nexys 4 has 8 seven segment displays available.
   - Advantages: On-board, simple interface and easy to implement
   - Disadvantages: Not much customisability, limited to 16

C. Measuring Results

Each independent module can be tested and verified to ensure its accuracy. Vivado IDE allows for behavioral simulations to test clock timing accuracy of inputs and outputs and verify the systems functioning. It allows for a register to be viewed in analog mode to show waveform outputs, this can be used to verify the frequency of our output waveforms and confirm they match the necessary notes. The inputs and outputs can be tested individually, buttons and switches can provide feedback via LED's and the seven segment display, their indented functions can be viewed via the simulation.

IV. DESIGN

Based on the functional requirements discussed in the previous section the advantages and disadvantages of each individual component were evaluated to determine which implementation would be chosen for the final design. The design was developed in an iterative modular fashion where each component of the system was designed, tested, and compared to the necessary requirements. This was repeated until each module met its necessary requirements.

1) Design Choices:

- User Input: Buttons and switches
- Look-up tables: MatLab scripts are used to generate fundamental sin, saw, triangle and square waveforms. The choice was made to have each complete waveform (1-period)consist of 1024 samples with an amplitude resolution of 9 bits.
- Memory: Two dimensional registers are used to store each waveform and enable concurrent memory access.
- Note frequency control: The frequency of each note is done by varying the clock speed to each note module which determines the rate the LUT is iterated at. The octave control is done by incrementing the address pointer at different multiples of two, leaving out a logarithmic number of samples each octave increase.
- Audio output: A PWM module running at the 100MHz clock transforms the required output into a PWM signal to the on-board PWM audio output.
- User display: The LED's and seven segment displays are used to provide the user with feedback on the current parameters of the system.

A. System Overview

Figure 8 shows an overview of the system's functional blocks. The core system is powered by the 100MHz clock of the Nexsys 4. Twelve "note blocks" adjust this clock to create a base frequency for each block. This new clock determines the rate at which a look-up table (LUT) is iterated over, sending values to be outputted by the system. The system's button inputs light corresponding LEDs and are fed into a control core, which in turn sets the parameters of each note block: LUT, octave, and waveform. The selected LUT determines which waveform will be produced, the octave determines the size of the steps through the LUT (Increments). The outputs of all of the note blocks are fed into an aggregator - a block which ensures that simultaneous notes do not overload the output, and are instead rescaled into an acceptable voltage range. Finally, the output is converted to an analog signal through the FPGA's pulse-width modulation (PWM) output.

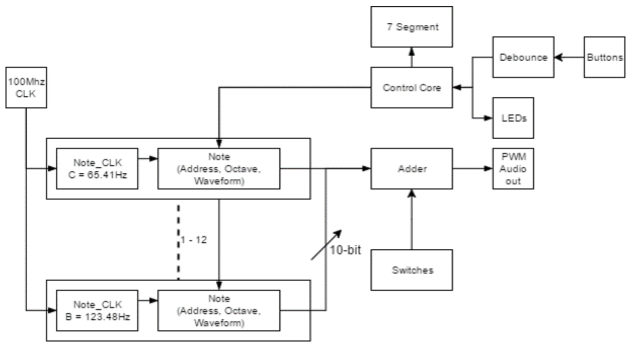

Fig. 4: Block diagram of entire system

B. User interface

The input and output for the user is displayed in Figure 5, the user's inputs are provided by the switches and buttons and the systems current parameters are displayed on the seven segment displays. The notes are always continuously being generated but are only sent to the adder module when the user selects a switch. This is done intentionally to prevent clock skew from constantly starting, halting, and restarting the note modules.

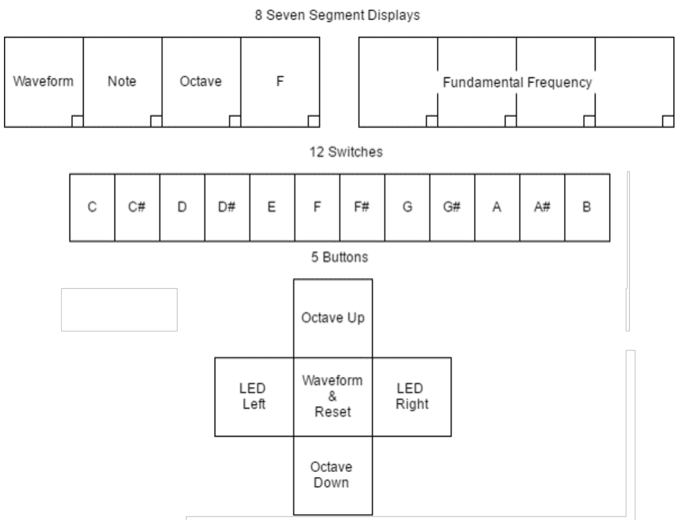

Fig. 5: User interface

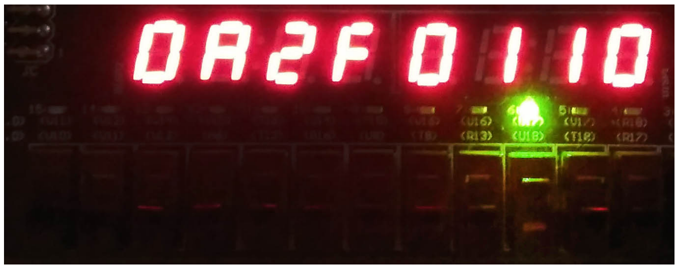

Fig. 6: Display from implementation

The LED's are used as a feedback for which note the current user is interacting with when they interface with the buttons. As shown in Figure 5, the left and right buttons increment and decrement which LED is currently illuminated, the switch the LED is above (when lit) is the current note being modified. The note can have its octave increased or decreased with the up and down buttons and the waveform type changed with the center button.

Additionally, if the center button is held in for about 8 seconds, the system will reset back to the initial state where every note is on the lowest octave (2) and outputting a sin waveform. Figure 6 shows the final implementation demonstrating the display. The current waveform is 0 for sin, the note is A, the octave is 2 and the frequency is 110Hz.

C. DDS_top

This is the top instantiation module that is used for implementing all of the submodules interconnects as well as the generation of LUT's from .dat files.

It holds the parameters for each note:

- NoteClock: The fixed note counter target value
- NoteOctave: The current octave of a note
- NoteWave: The current waveform of a note
- NoteFreq: Hard coded value of the notes fundamental frequency
- NoteName: Hard coded hex value for the seven segment display

The top module handles various controls during its clock trigger.

- Seven Segement display based on current note
- Adder for-loop
- Button debounce and their effect

D. Look-Up Table

A MatLab script is used to generate the four fundamental waveforms shown in Figure 7. Each LUT has 1024 samples with each sample comprised of 9 bits equating to a resolution of of 6.14^{-3}

$$Resolution = \frac{2\pi}{2^{10}-1} \tag{6}$$

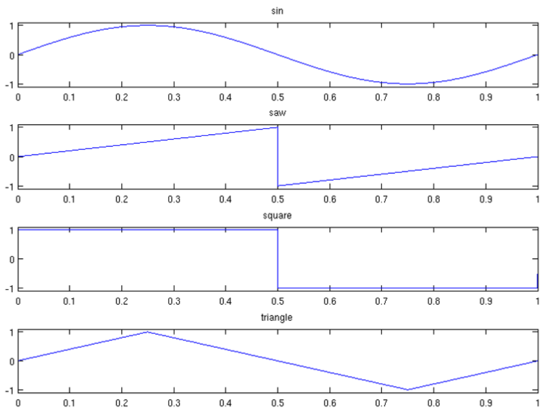

Fig. 7: Fundamental Waveforms

This produced a sampled waveform is stored in 1024 9-bit registers on the FPGA representing a LUT. Using registers allows for multiple Note modules, as discussed further on, to access the same LUT value, which would not be possible when ROM is used due to its nature. This utilizes the space of the FPGA more efficiently while improving the design of the system.

E. Note Module

A module generation loop is used to generate 12 note modules. Each module implements a simple counter, with a dynamically adjustable target value. The counter increments during every clock from the system clock f_c until it reaches the target value. Arriving at the target value triggers an output pulse and the counter value is reset. This effect divides the system clock by a value V, the target value. The target value is hard coded into the notes' instantiation.

Each note outputs an individual address which is incremented each time its specified clock target overflows. This address is not linked to any specific waveform, if the waveform is changed the address remains the same and is now linked to a different LUT at the same point. This overflow trigger represents the relative output frequency of the note.

F. Adder Module

The module is implemented by a for-loop which triggers on the value of each switch. If a switch is activated it uses the note's address it is related to and the current waveform the note uses in order to fetch the LUT value at that point and add it to a temporary summing value. A counter is used to increment based on the number of current notes activated, this count is used as divisor for the value of the adder in order to normalise it to a 9-bit value representing the summed waveforms. This adder value is then sent to the PWM module.

G. PWM Module

This module consists of a counter that produces a binary output, PWM, waveform that depends of the value supplied at its input register. The value is scaled by the maximum value of the LUT and an appropriate duty cycle is created. This is done by setting the target value to the scaled value. The binary output is set high, until the counter reaches its target value, then it is held low. This high to low step occurs only once per input clock.

V. Proposed Development Strategy

Commercial development of this product could be beneficial for the fields outlined in the introduction where accurate waveform synthesis is required. The major difference in the requirements between our audio based implementation and in fields such as radar and biomedical is the degree of accuracy and the scope of frequencies required. With our current implementation we can achieve the 7th octave of notes up to a frequency of 4kHz with high fidelity. In commercial applications the range of frequencies required is often in the MHz and GHz range at which point our output frequency would be quantized away.

In terms of audio applications many electronic instruments and audio synthesizers could benefit from a variable external oscillator yet the major drawback of our implementation is the forced use of a PWM output due to the Nexys4.

Converting our FPGA design into a self contained ASIC would only be cost effective for mass production of a product that would be successful in its niche environment. For this reason two implementations for an audio device from our project are proposed:

A. High End Audio Synthesiser

In order to produce a viable high end product the unachievable functional requirements outlined in section three would be implemented. For the scope of this project many of these implementations would be overly expensive and unnecessary but using components such as a high resolution audio DAC, piano keyboard and screen for visualizing outputs and parameters would need to be built in to the final product.

Converting an elaborate implementation of our design into an ASIC to be sold as a commercial product would also require advanced functionality such as audio effects, ADC's for potentiometer inputs and a UART interface for a mobile app or computer application.

B. Low Cost Audio Device

The cheaper alternative that would require less development, no added functionality and cost effective components is a proposed children's audio toy or entry level audio synthesizer. The scope of our implementation encompasses the basics of fundamental audio, it could be used educationally for understanding how music works to beginners. By adding a low cost screen that could visualize output waveforms similar to an oscilloscope it could be used educationally as an introduction to signals and the science of music.

VI. Experimentation

This section details the experiments that where undergone to ensure the integrity and fidelity of the system.

A. Systems-level Test

The final system successfully managed to be synthesized, implemented and written to the FPGA with no critical errors. Key Note modules where tested using the simulation environment found within the Vivado IDE package.

B. Audio Performance

The accurate synthesis of digital signals is the main focus of this project, and the results section shall reflect this by giving precedence to this set of results.

Integration tests were performed live and using oscilloscopes and simulated matlab results to ensure the produced waveform is as expected. The pre-recorded waveform that was generated by matlab was found to be indiscernible from the FPGE generated waveform to our testing team and differences on the oscilloscope were found to be negligible. Multiple waveforms were created and tested with the overall results being consistently accurate.

At higher octaves, the number of samples per waveform is significantly decreased. This causes artifacts to be present in the form of jagged parts of the wave. The intensity of the error is largely waveform dependent, but the SNR of the produced waveform is still high enough for the human ears to not detect any significant loss of audio quality.

Due to the requirement for integer downscaling when determining each note's effective clock. Some truncation is done. this effectively quantizes the value. At lower frequencies this is not a problem, but as the octaves are increased, this can be noticeable to individuals who have perfect-pitch. Initially this was of some concern in the design phases. In implementation, however, it was found for the note of C_8 was 4213.3Hz while the A440 standard for C_8 is 4186Hz leading to an error of 0.6%.

VII. Results

This section details the results of the experimentation and how the compared to the ideal waveforms.

A. Waveform Specifications

Each waveform will consist of 1024 9-bit values per period. These are created in matlab and are used as the LUT values. These values were chosen to ensure the processing can complete within the 100Mhz Clock speed of the FPGA Development Board. For each note to be correctly replicated. Each sample, of which they are 1024, needs to undergo a 9-bit audio PWM creation process. For our highest base frequency (B_3 at 246.94Hz). This would mean 246.94 * 1024 * 512 = 129467678.72Hz or roughly 130Mhz. This is beyond the speed of the FPGA and hence we lowed the entire initial range by 1 octave. This adjustment causes B_2 to be our highest frequency at 123.47Hz. This effectively halves the minimum required clock cycle down to 65Mhz. A speed obtainable on the FPGA with sufficient headroom.

These decisions do not come without costs however. Lowering the base octave relieves pressure on the FPGA's clock rate but cuts down on the number of samples made available to higher octaves. Table I shows the number of effective samples if C_3 is chosen as our base note.

Note	Frequency	Number of Samples
C_3	130.8	1024
C_4	261.6	512
C_5	523.3	256
C_6	1047	128
C_7	2093	64
C_8	4186	32

TABLE I: Table of Effective Samples per note at different harmonics with C_3 as lowest frequency

As we increase the octave of the note, we can observe the drastic degradation of the wave by observing the smaller amount of samples available to represent the full wave.

If we change down the base note to be C_2, we observe that C_8 is now at 16 effective samples. a 50% drop.

Note	Frequency	Number of Samples
C_2	65.9	1024
C_3	130.8	512
C_4	261.6	256
C_5	523.3	128
C_6	1047	64
C_7	2093	32
C_8	4186	16

TABLE II: Table of Effective Samples per note at different harmonics with C_2 as lowest frequency

B. Combination of DDS Modules

We successfully managed to add two waveforms together. We created two independent DDS Modules. One running at C_3 = 130.8hz and the other running at G_3 = 196.0Hz. In music terms, the G is a perfect 5th away from the C. The output of each module was added together and normalized. The output was shown to be correct by comparing it to a MatLab Simulation.

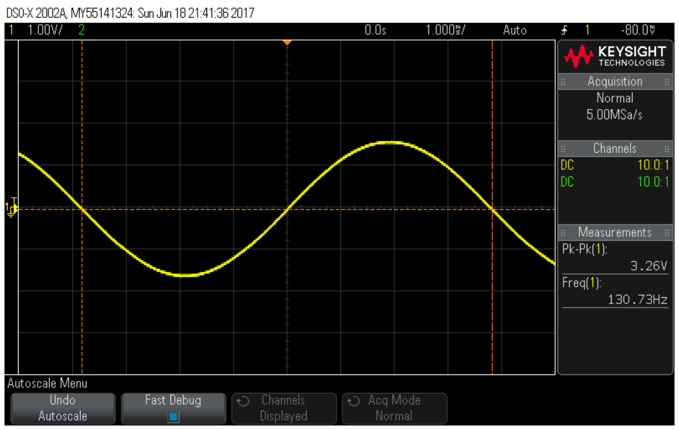

Fig. 8: Oscilloscope output at 130.8Hz at 1024 Samples

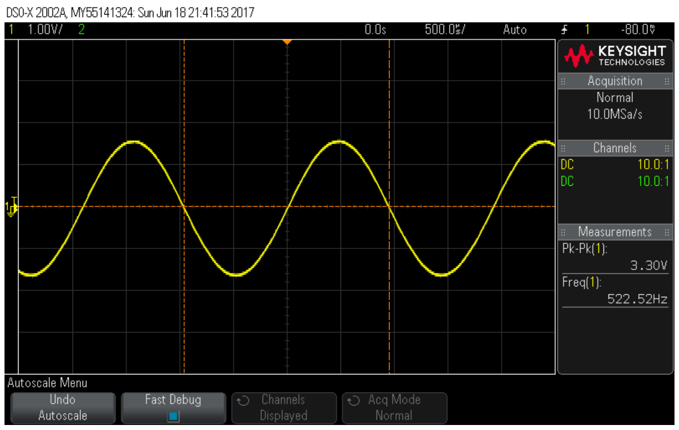

Fig. 9: Oscilloscope output at 523.3Hz at 256 Samples

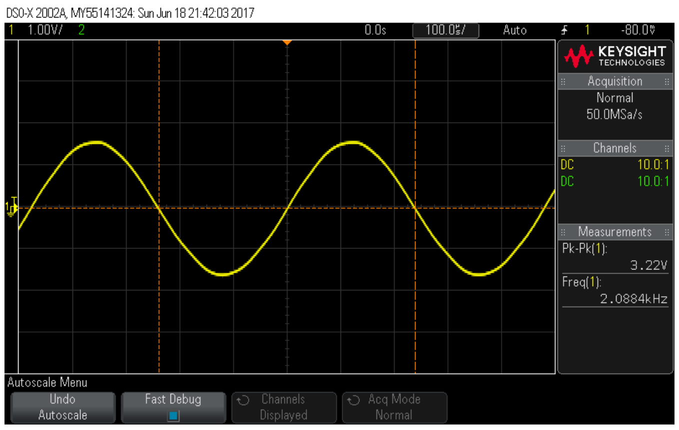

Fig. 10: Oscilloscope output at 2093Hz at 64 Samples

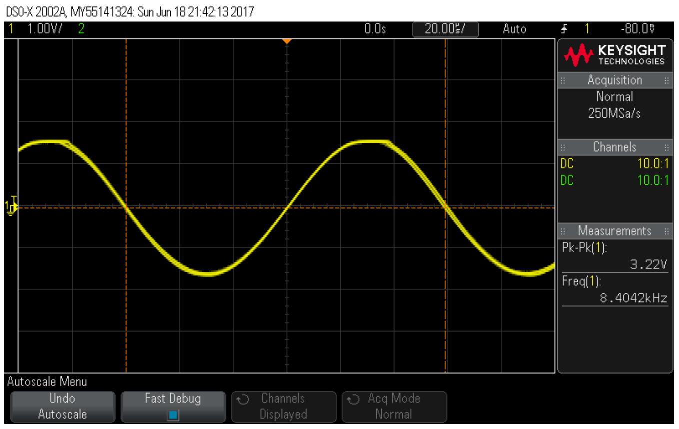

Fig. 11: Oscilloscope output at 8372Hz at 16 Samples

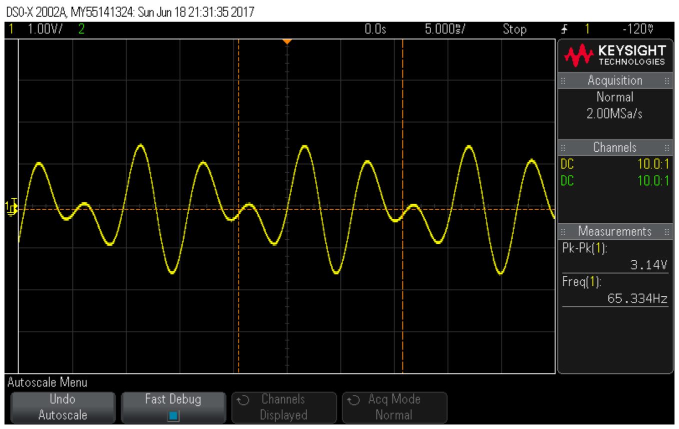

Fig. 12: Oscilloscope output of C_3 added to G_3 and normalized

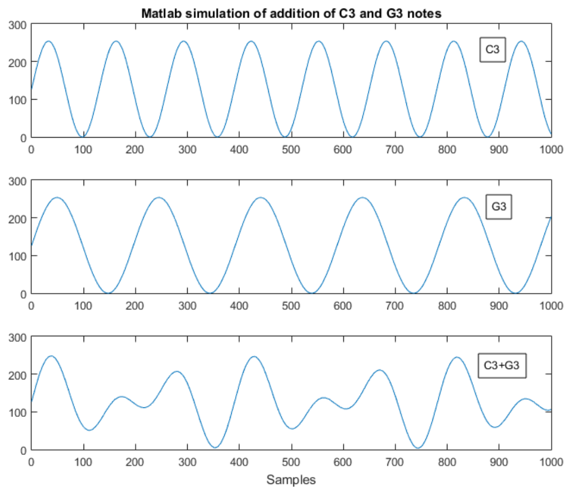

Fig. 13: MatLab Simulation of C_3,G_3, and C_3 + G_3

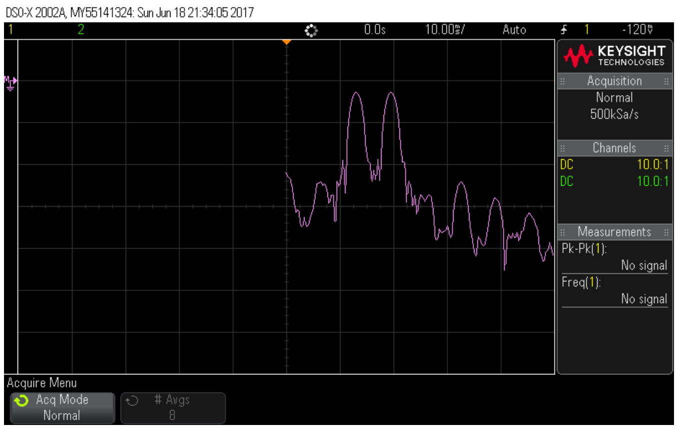

Fig. 14: Fourier Analysis showing the two Peaks: P_1 centered at 130.8Hz. P_2 centered at 196.0Hz.

VIII. Conclusion

In conclusion, we successfully managed to build a system from the ground up to allow individuals to play all 12 notes of the western music system. We created the system using equal temperament tuning with the modern A=440Hz standard. Each note was allowed a dynamic waveform with on-the-fly octave adjustment. We speculated and showed that higher octaves would produce waveforms that deviated from their analog counterparts due to less samples being used, but that the general shape is maintained and only minor distortion would be observed by the listener.

The system successfully managed to produce these higher frequency waveforms, while itself did not need any form of dynamic clock adjustment. This was all achieved in realtime using bitshifting.

This project can further be expanded by allowing parallel communication with a desktop computer via the UART port and by displaying the produced waveforms to the VGA output.

References

[1] Digilent, "Nexys 4 fpga board reference manual," 2016. [Online]. Available: https://reference.digilentinc.com/_media/nexys:nexys4:nexys4_rm.pdf

[2] K. Bhagat, "Tutorial on designing and implementing a direct digital synthesizer (dds) on a field programmable gate array (fpga)," 2012.

[3] C. S. Eva Murphy, "All about direct digital synthesis," *Ask The Application Engineer33*, 2004. [Online]. Available: http://www.analog.com/media/en/analog-dialogue/volume-38/number-3/articles/all-about-direct-digital-synthesis.pdf

[4] A. Devices, "Fundamentals of direct digital synthesis (dds)," *Analog Devices*, 2009. [Online]. Available: http://www.analog.com/media/en/training-seminars/tutorials/MT-085.pdf

[5] Wikipedia, "Direct digital synthesizer," 2017. [Online]. Available: https://en.wikipedia.org/wiki/Direct_digital_synthesizer

[6] L. Cordesses, "Direct digital synthesis: A tool for periodic wave generation (part1)," *dsp tips & tricks*, 2004. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1311140

[7] V. S. Reinhardt, "Direct digital synthesizers."