

tone generator using different frequencies






Assignment - 1 Report

Introduction

In the evolving world of digital electronics, **Field-Programmable Gate Arrays (FPGAs)** have emerged as important elements in the design and implementation of various digital systems, due to their versatility, reconfigurability, and **high-performance** capabilities.

Among the many applications, the creation of **tone generators using different frequencies** showcases a majestic intersection of **sound engineering and digital technology**. This report deals with the design, implementation, and analysis of an **FPGA-based tone generator, a device engineered to produce audio tones by generating waveforms at specific frequencies**. This exploration not only highlights the technicalities of FPGAs in handling digital signal processing tasks but also highlights the broader implications for areas such as **music technology, audio testing equipment, and educational tools**. Through a study, this report aims to illuminate the process of using FPGA technology to synthesize sound and offering insights into the difficulties of digital sound generation and the potential for innovative applications in the realm of audio technology.

FPGA DE2	Project Photo	Application - 1 Keyboard	Application - 2 Tone Dialer
			

Abstract

This project is about creating a tool that makes **different sounds using an FPGA DE2 board**. The main goal is to show how an FPGA chip, can be **used to make various audio tones like music notes or sound effects**. We programmed the FPGA to make different kinds of sounds, such as **beeps and buzzes, which can be changed in pitch and loudness**. Further , we can also add user inputs to choose which type of sound they want .

The **project combines both hardware, the physical board and its components, and software, the code that tells the hardware what to do**. A better setup allows users to experiment with creating different sounds, which can be useful for **learning about sound, making music, or testing audio equipment**.

By completing this project, we learned **how versatile and powerful FPGAs** are for working with sound. This tool could be especially helpful for people interested in **music technology, sound design, or electronics**.

Keywords

1. **FPGA (Field-Programmable Gate Array)** : A reprogrammable circuit board that can be configured after manufacturing for various tasks, such as processing audio signals.
2. **DE2 Board** : A development and education board inserted with an FPGA chip, used for designing electronic systems.
3. **Waveform** : The shape and form of a signal wave represented graphically, such as sine, square, or triangle waves, indicating how sound varies over time.
4. **Frequency** : Refers to the number of times a wave repeats itself within a second, determining the pitch of the sound.
5. **Amplitude** : The height of the wave, which correlates to the volume or loudness of the sound produced.
6. **Digital Signal Processing (DSP)** : The use of digital computation to modify or analyze signals, such as audio.
7. **Tone Generation** : The process of creating sound electronically with specific characteristics like pitch and timbre.
8. **Polyphonic Sounds** : The ability to play multiple notes or sounds simultaneously, creating a harmonic effect.

9. **Hardware and Software Integration** : The process of making the physical components (hardware) work seamlessly with the programmed instructions (software).
 10. **Audio Signal Processing** : The techniques used to manipulate audio signals to alter their characteristics, such as filtering, compressing, or generating sounds.
 11. **Pitch** : The perceived frequency of a sound, which determines whether it sounds high or low.
 12. **Sound Synthesis** : The creation of sounds from scratch using electronic means, often involving waveforms and digital processing.
-

Literature Survey / State of the art Technology

1. <http://www.innovateasia.com/asia/download/articles/2013/cn147.pdf>
2. https://www.academia.edu/7912472/Audio_Tone_Generator_Using_Verilog_HDL_Coding_Implementation_of_Audio_Tone_Generator_on_FPGA_Using_Verilog_HDL_Coding
3. <https://www.youtube.com/watch?v=AANKGfbFYIk>

State-of-the-art technology for Polyphonic Tone Generation on FPGAs includes:

1. High-performance FPGAs with dedicated DSP blocks.
2. High-Level Synthesis (HLS) tools for C/C++ code synthesis.
3. Custom IP cores for audio processing tasks.
4. Real-time operating systems (RTOS) for task scheduling.
5. Advanced audio codecs for high-fidelity input/output.
6. Open-source audio libraries for synthesis algorithms.
7. Machine learning for sound modeling.
8. Simulation and verification tools for design validation.
9. Cloud-based FPGA platforms for scalable deployment.

Some Indian companies using **advanced FPGA technology** for audio processing include **Xilinx India, Intel India** (Programmable Solutions Group), **Texas Instruments India, Analog Devices India**, and various startups and research institutions focusing on audio processing with FPGAs.

Some Indian speaker making companies that may use advanced technologies like FPGA for **audio processing** include **Boat Lifestyle, JBL India** (Harman International), **Zebronics**, **Creative Technology**, and **Tranquil Audio**.

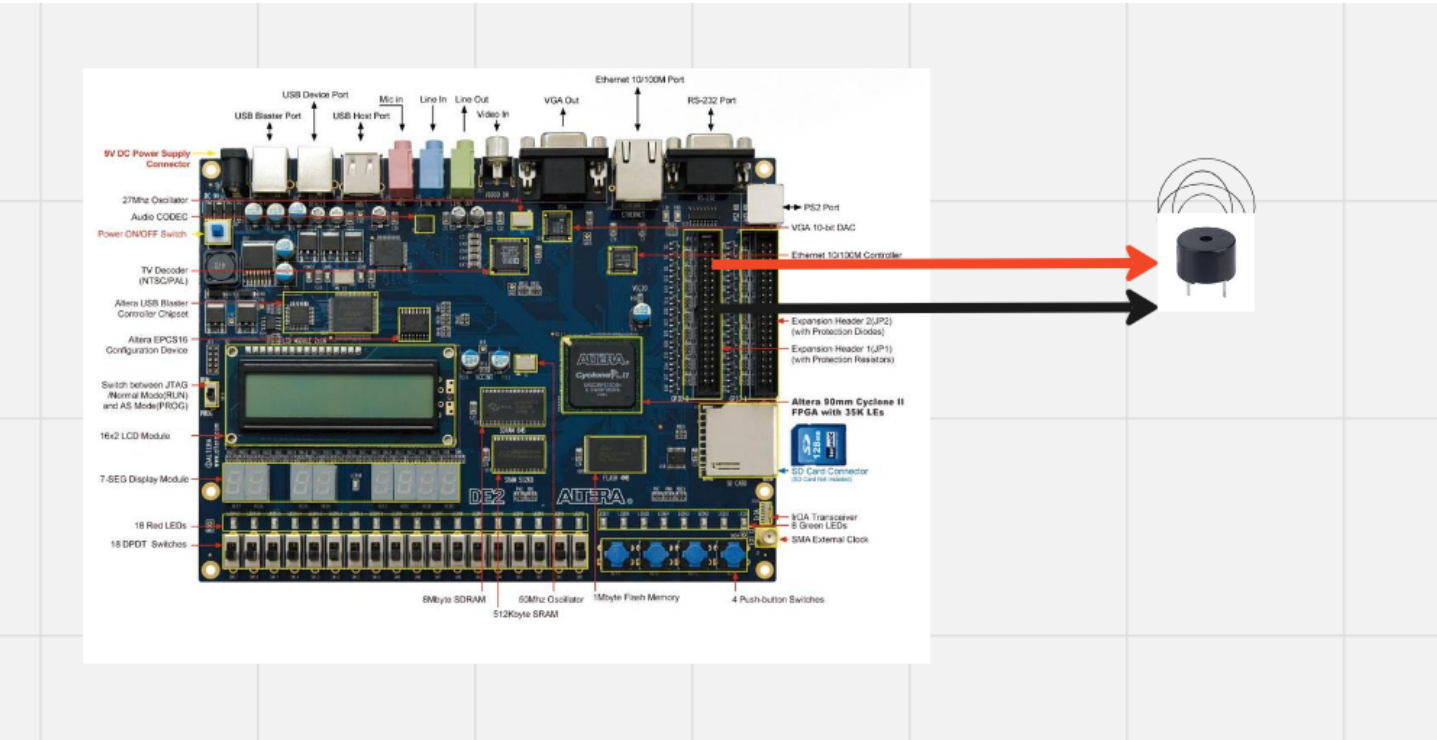
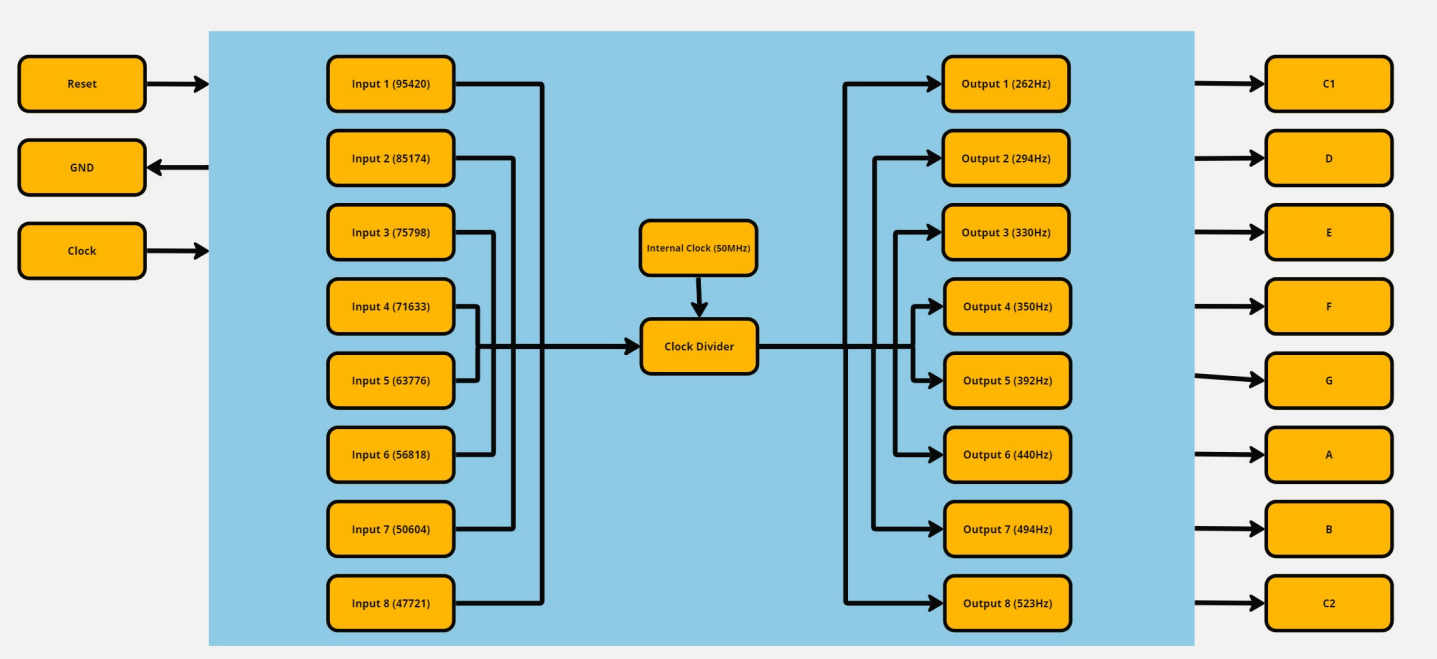
Limitations or Drawbacks of currently available technology

1. **Complexity** : Implementing FPGA-based solutions for audio processing can be complex and requires expertise and designing efficient algorithms and optimizing hardware resources can be challenging.
 2. **Cost** : FPGAs and related development tools can be expensive, especially for high-performance models with advanced features.
 3. **Power Consumption** : High-performance FPGAs needs more power, especially when running complex audio processing algorithms at high frequencies.
 4. **Time Consumption** : Designing, debugging, and testing FPGA-based audio processing systems can be time-consuming.
 5. **Resource Constraints** : FPGAs have limited resources such as logic elements, memory, and DSP blocks.
 6. **Latency** : Real-time audio processing applications require low-latency solutions.
 7. **Integration Challenges** : Integrating FPGA-based audio processing projects with pre-existing audio systems can be challenging.
-

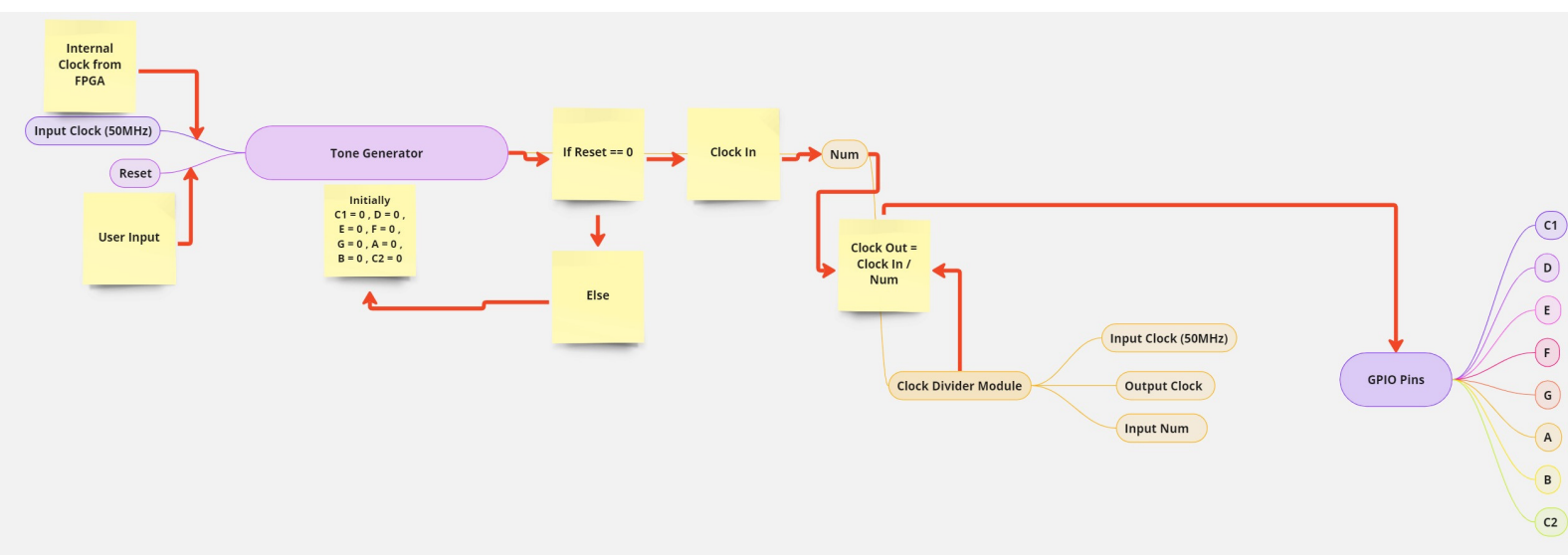
Proposed Solution / Methodology

1. **Modular Design and Simulation**: Break the system into smaller parts for easier management and use simulation tools to validate each module before hardware implementation.
 2. **High-Level Synthesis (HLS)**: Utilize HLS tools to write code in higher-level languages like C, which are then converted to FPGA logic which simplifies the design process.
 3. **Resource Optimization**: Apply techniques like time-multiplexing of DSP blocks and efficient coding practices to make the best use of FPGA.
 4. **Power Management**: Implement power-efficient designs to estimate and optimize power usage.
 5. **Use of IP Cores**: Apply pre-designed IP cores for common audio functions to reduce development time and ensure efficiency.
 6. **Latency Management**: Design with low latency in mind by employing pipelining and parallel processing to meet real-time requirements.
-

Block Diagram / Circuit Diagram



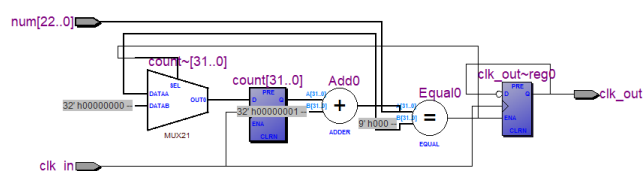
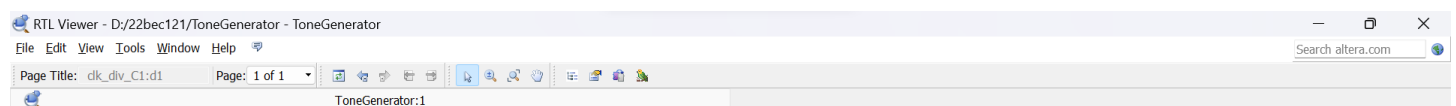
Flow Chart of the Code



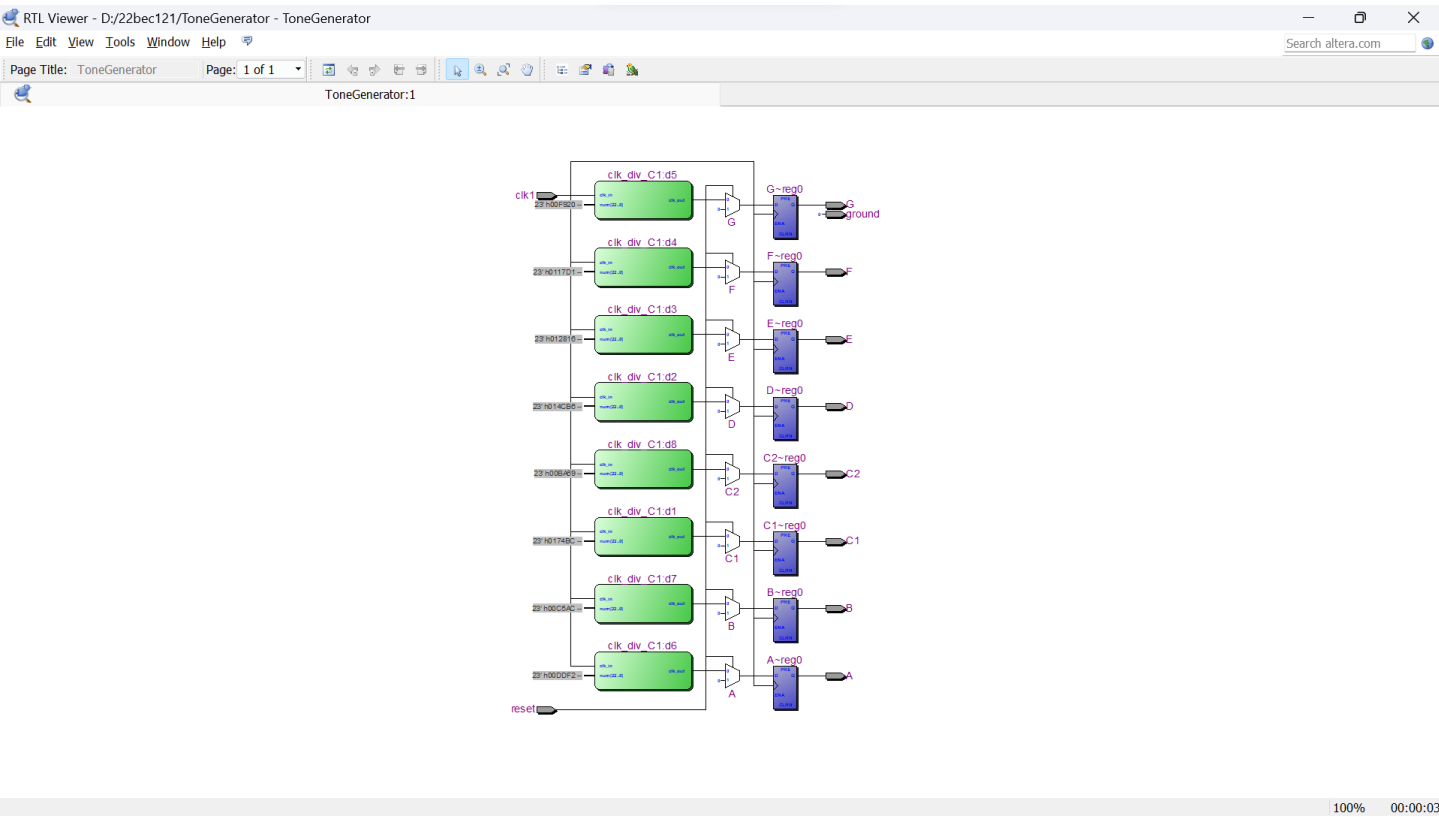
RTL , TTL and Simulation (Waveform)

Results

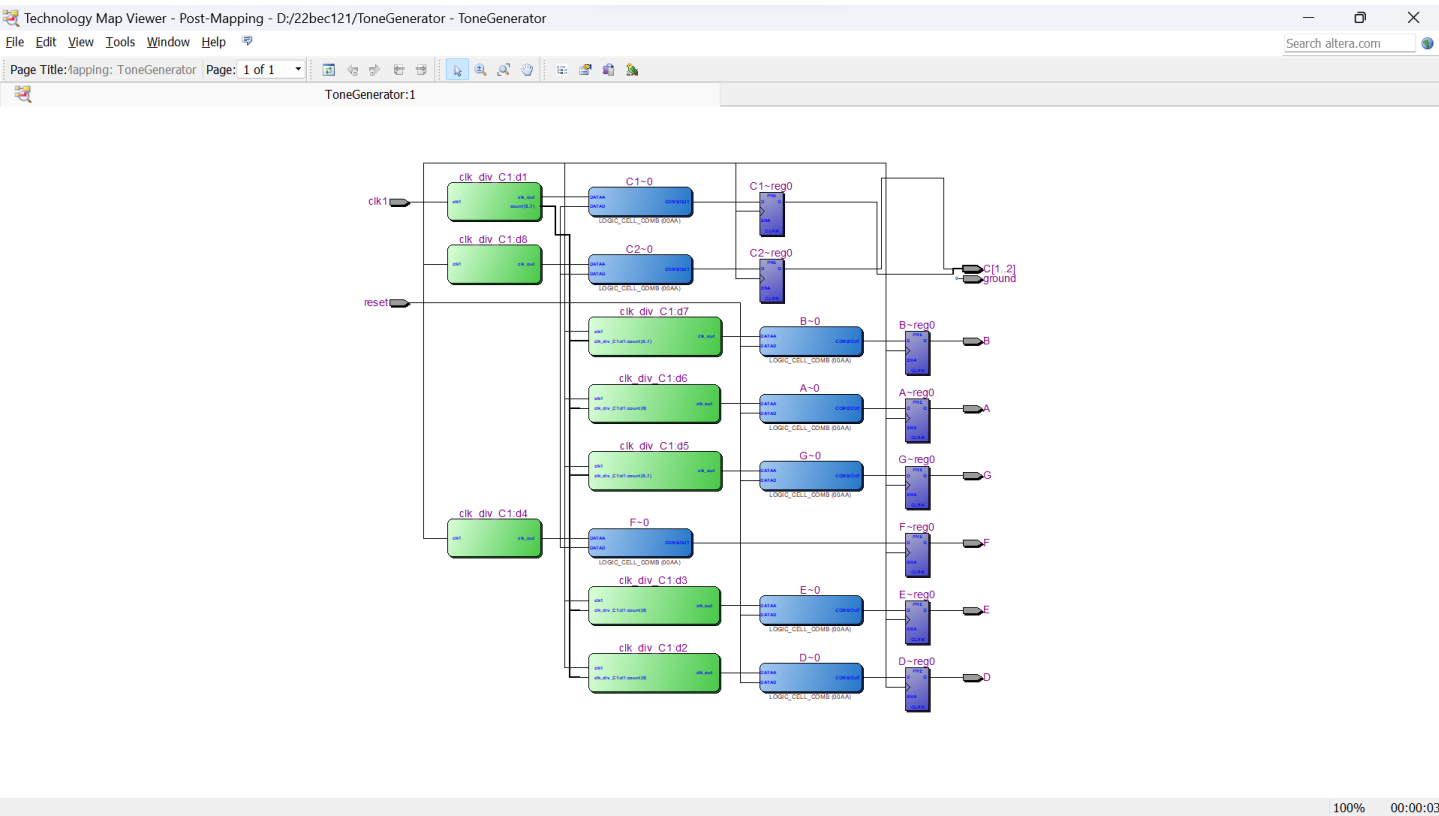
RTL :- 1) Clock Divider Module



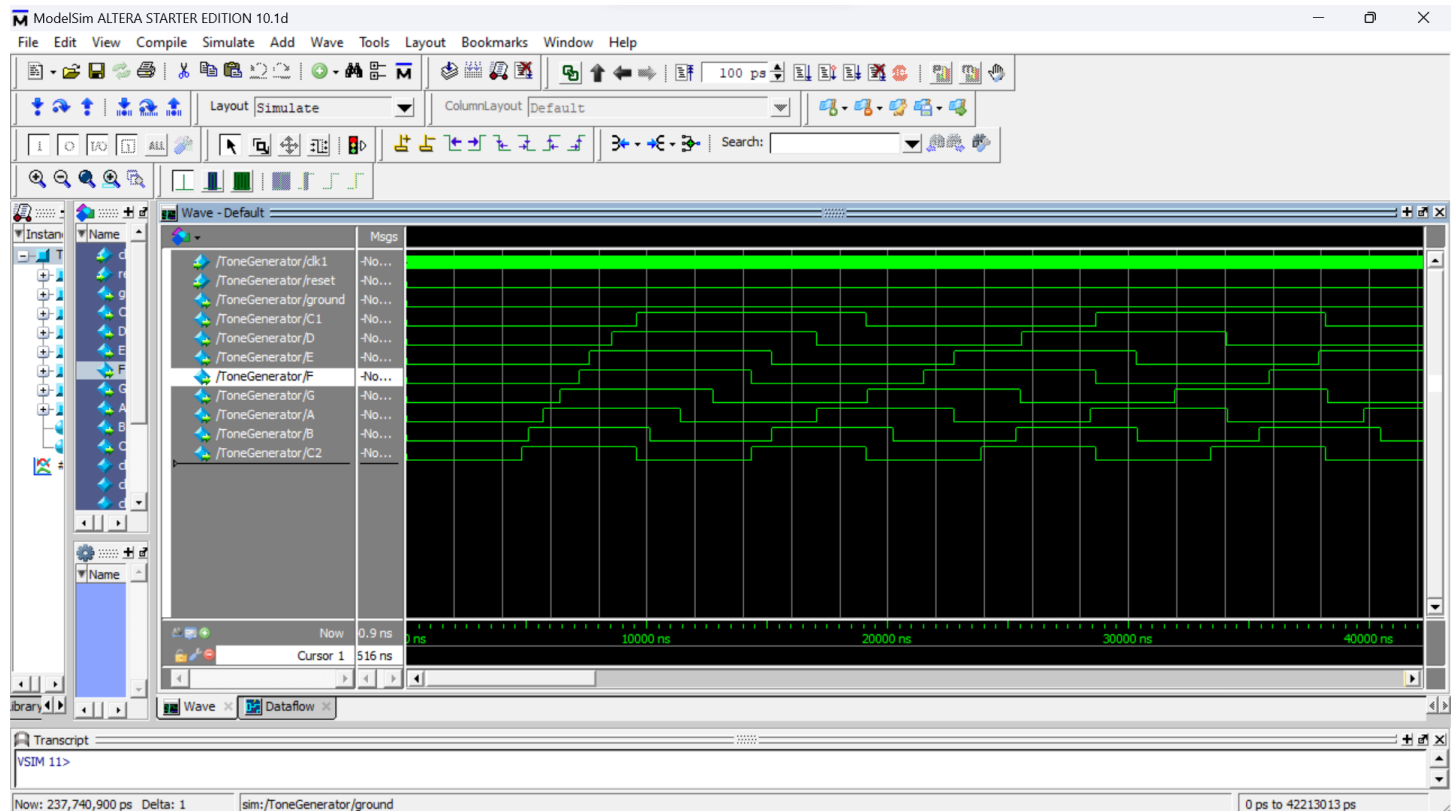
RTL :- 2) Tone Generator Module (Main)



TTL :- Tone Generator Module (Post Mapping)



Simulation :- ModelSim (Waveforms)



Conclusion

In this project , I learnt a lot of new things , i.e. Clock Divider , Square Wave Generator , Generate different frequencies and finally Polyphonic Signals and their generation .

This project also helped me to gain knowledge about the FPGA board and its power to do many different things .

References

1. https://www.academia.edu/7912472/Audio_Tone_Generator_Using_Verilog_HDL_Coding_Implementation_of_Audio_Tone_Generator_on_FPGA_Using_Verilog_HDL_Coding

2. <https://www.fpga4fun.com/MusicBox.html>

3. Textbooks and Professors

Appendix

Verilog Code

```
module ToneGenerator (  
    input clk1 ,  
    input reset ,  
    output ground ,  
    output reg C1 , D , E , F , G , A , B , C2  
);  
  
    wire clk2 ; wire clk3 ; wire clk4 ; wire clk5 ; wire clk6 ; wire clk7 ; wire clk8 ; wire clk9 ;  
  
    clk_div_C1 d1(.clk_in(clk1) ,.num(95420), .clk_out(clk2)) ; //SA 0  
    clk_div_C1 d2(.clk_in(clk1) ,.num(85174), .clk_out(clk3)) ; //RE 1  
    clk_div_C1 d3(.clk_in(clk1) ,.num(75798), .clk_out(clk4)) ; //GA 2  
    clk_div_C1 d4(.clk_in(clk1) ,.num(71633), .clk_out(clk5)) ; //MA 3  
    clk_div_C1 d5(.clk_in(clk1) ,.num(63776), .clk_out(clk6)) ; //PA 4  
    clk_div_C1 d6(.clk_in(clk1) ,.num(56818), .clk_out(clk7)) ; //DHA 5  
    clk_div_C1 d7(.clk_in(clk1) ,.num(50604), .clk_out(clk8)) ; //NI 6  
    clk_div_C1 d8(.clk_in(clk1) ,.num(47721), .clk_out(clk9)) ; //SAA 7  
  
    initial  
        begin  
            C1 = 0 ; D = 0 ; E = 0 ; F = 0 ; G = 0 ; A = 0 ; B = 0 ; C2 = 0 ;  
        end  
  
    always @(posedge clk1)  
        begin  
            if(reset)  
                begin  
                    C1 = 0 ; D = 0 ; E = 0 ; F = 0 ; G = 0 ; A = 0 ; B = 0 ; C2 = 0 ;  
                end  
            else if (!reset)  
                begin  
                    C1 = clk2 ; D = clk3 ; E = clk4 ; F = clk5 ; G = clk6 ; A = clk7 ; B = clk8 ; C2 = clk9 ;
```

```

        end
    end

    assign ground = 0 ;

endmodule

module clk_div_C1 (
    input clk_in ,
    input [22:0]num ,
    output reg clk_out
);

    reg [31:0]count ; //17

    initial
        begin
            count = 0 ; clk_out = 0 ;
        end

    always @(posedge clk_in)
        begin
            count = count + 1 ;
            if (count == num)
                begin
                    clk_out = ~clk_out ;
                    count = 0 ;
                end
            end
        end
endmodule
//GND 12

```

*Thank
You*