Experiment 6: Musical Tune Player

Prasann Viswanathan, Roll Number 190070047
EE-214, WEL, IIT Bombay
April 3rd, 2021

Overview of the experiment:

The purpose of this experiment was to use FSM theory to play a sequence of musical notes in order to create a melodious tune. This was to be done using krypton board and the basic speaker set up of a previous lab.
For this experiment, we needed three VHDL files – toneGenerator from lab 5, music.vhd and song_tb, provided to us by the TAs. Modifications needed to be made only in music.vhd We had to complete the code in order to properly map indices under each note to the correct subsequent note. Also we had to go in steps of 0.25s (4Hz) requiring some more careful mappings to play the given musical piece.

Approach to the experiment:

The musical piece to be played was given as:

Note	Sa	Ga	Sa	Ga	Sa	Ga	Ma	Ga	Re	Sa
Duration	0.5s	0.5s	0.5s	0.5s	0.5s	0.5s	0.25s	0.25s	0.25s	0.25s
Count	1, 2	3, 4					13			16
Note	Ni	Re	Ni	Re	Ni	Re	Ga	Re	Sa	Ni
Duration	0.5s	0.5s	0.5s	0.5s	0.5s	0.5s	0.25s	0.25s	0.25s	0.25s
Count										32

Table 1: Notes Table

As a result, we were to map the correct indices under each block to the correct following note.
Here is the mapping –

Sa(1, 5, 9) → Sa, Sa(2, 6, 10) → Ga, Sa(16, 31) → Ni
Re(15, 30) → Sa, Re(19, 23, 27) → Re, Re(20, 24) → Ni, Re(28) → Ga
Ga(3, 7, 11) → Ga, Ga(4, 8) → Sa, Ga(12) → Ma, Ga(14, 29) → Re
Ma(13) → Ga
Ni(17, 21, 25) → Ni, Ni(18, 22, 26) → Re, Ni(32) → Sa

Design document and VHDL code if relevant:

As mentioned in overview, we used three VHDL files to implement this musical tune player:
toneGenerator : From lab 5, maps switches to different frequencies played by speaker.
Switch 1 → Sa, …, Switch 7 → Ni
song_tb : Provided by the TAs. No modification was necessary. Used in order to test the RTL simulation of our device.

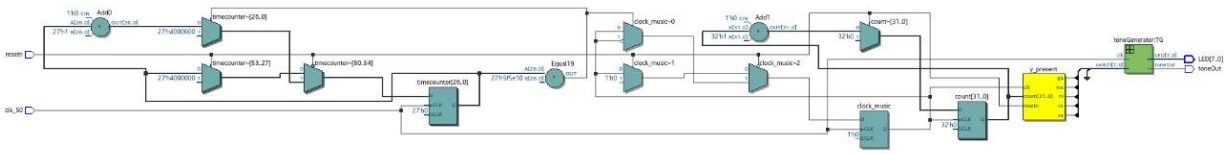
music : The main logic for this lab. We had to implement the FSM part of the design in this file. Mainly the architecture which is shown here:-

```
architecture fsm of music is

type state_type is (Silent,sa,ga,ma,re,ni);

signal y_present : state_type;
signal LED_2:std_logic_vector(7 downto 0);
signal count : integer := 0 ;
signal clock_music:std_logic :='0';

component toneGenerator is
port (toneOut : out std_logic; --this pin will give your notes output
clk : in std_logic;
LED : out std_logic_vector(7 downto 0);
switch : in std_logic_vector(7 downto 0));
end component;

begin
        process(clk_50,resetn,y_present,clock_music)    -- Fill sensitivity list
        variable y_next_var : state_type;
        variable n_count : integer := 0;
        variable timecounter : integer range 0 to 1E8 := 0;

        begin

                y_next_var := y_present;
                n_count := count;

                case y_present is
                        when Silent=>
                        y_next_var := sa;
                        LED_2 <= (0 =>'0',others => '0');

                        WHEN sa =>     --if the machine in Sa state
                                if((n_count=2) or  (n_count=6) or (n_count=10)) then
                                y_next_var := ga;
                                elsif ((n_count=16) or (n_count=31)) then
                                y_next_var := ni;
                                else
                                y_next_var := sa;
                                end if;
                                LED_2 <= (0=>'1',others=>'0');

                        WHEN re =>
                                if ((n_count=15) or (n_count=30)) then
                                y_next_var := sa;
                                elsif ((n_count=20) or (n_count=24)) then
                                y_next_var := ni;
                                elsif((n_count = 28)) then
                                y_next_var := ga;
                                else
                                y_next_var := re;
                                end if;
```

```
                        LED_2 <= (1=>'1',others=>'0');

                WHEN ga =>
                        if ((n_count=4) or (n_count=8)) then
                        y_next_var := sa;
                        elsif ((n_count=12)) then
                        y_next_var := ma;
                        elsif ((n_count=14) or (n_count=29)) then
                        y_next_var := re;
                        else
                        y_next_var := ga;
                        end if;
                        LED_2 <= (2=>'1',others=>'0');

                WHEN ma =>
                        y_next_var := ga;
                        LED_2 <= (3=>'1',others=>'0');

                WHEN ni =>
                        if((n_count=18) or (n_count=22) or (n_count=26)) then
                        y_next_var := re;
                        elsif ((n_count=32)) then
                        y_next_var := sa;
                        else
                        y_next_var := ni;
                        end if;
                        LED_2 <= (6=>'1',others=>'0');
        END CASE ;

--      generate 4Hz clock (0.25s time period) from 50MHz clock (clock_music)
        if (clk_50 = '1' and clk_50' event) then
                if (resetn = '0') then
                        if timecounter = 6250000 then -- The cycles in which clk is 1 or 0
                                timecounter := 1;
                                clock_music <= not clock_music;
                        else
                                timecounter := timecounter + 1;

                        end if;
                elsif resetn = '1' then
                        timecounter := 1;
                        clock_music <= '0';
                end if;
        end if;

--      state transition
        if (clock_music = '1' and clock_music' event) then
                if (resetn = '1') then
                        y_present <= Silent;
                        count <= 0;
                else
                        y_present <= y_next_var;
                        count <= n_count + 1;
                end if;
        end if;
```

```
        end process;

TG: toneGenerator
port map(toneOut, clk_50, LED, LED_2);

end fsm;
```

RTL View:

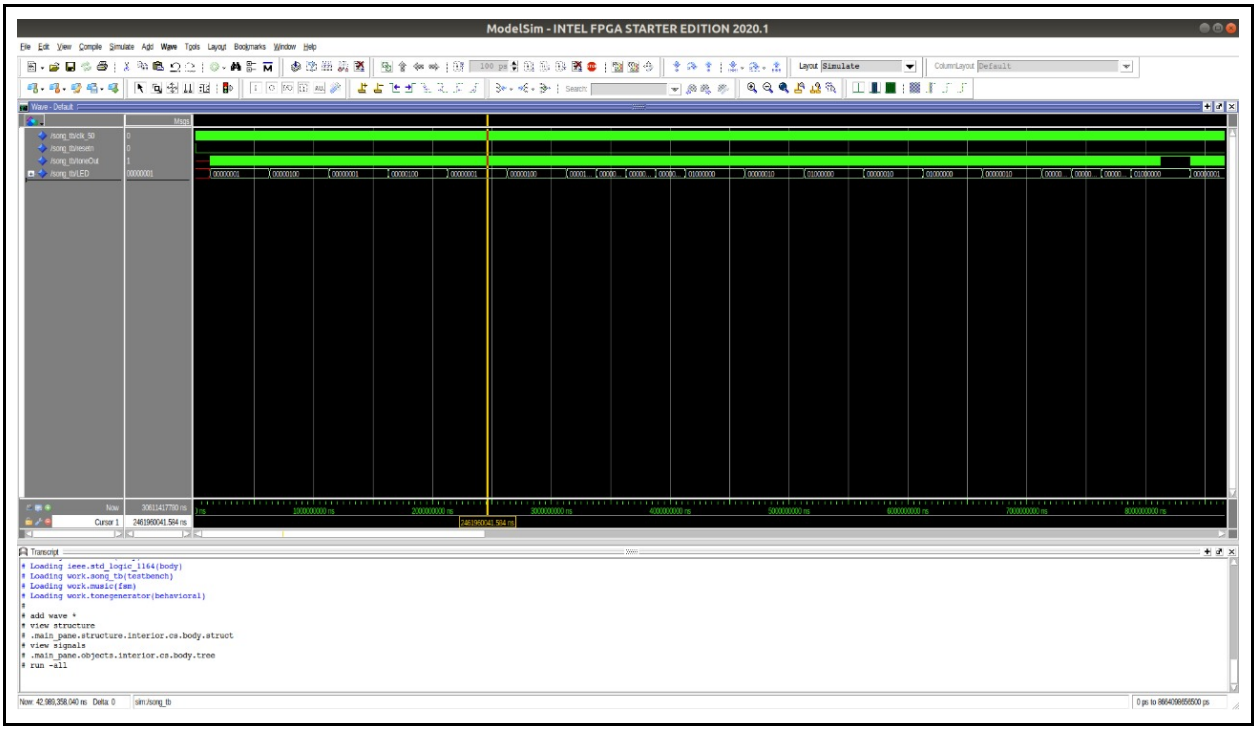


State Transition Diagram:-

DUT Input/Output Format:

NA

RTL Simulation:

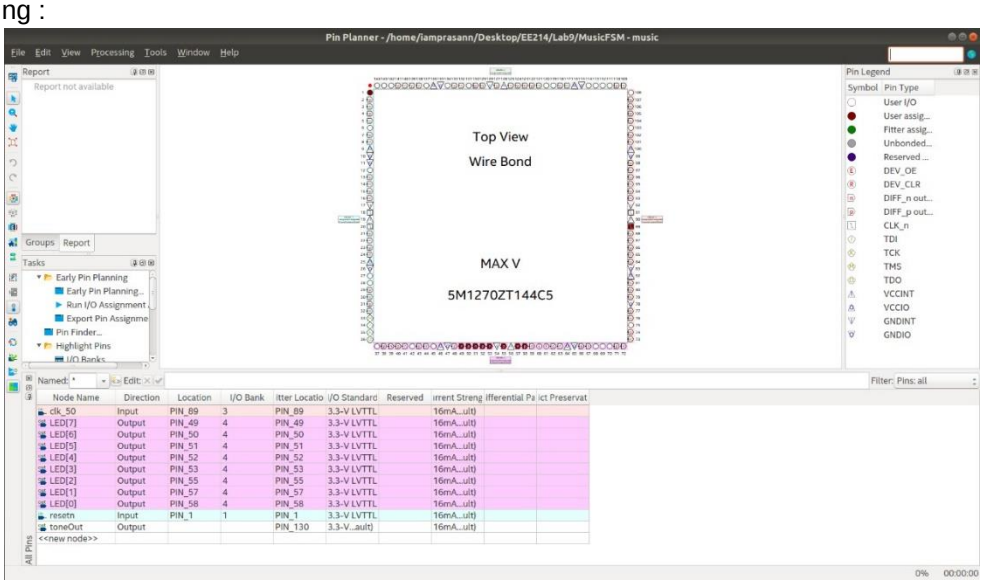


Gate-level Simulation:

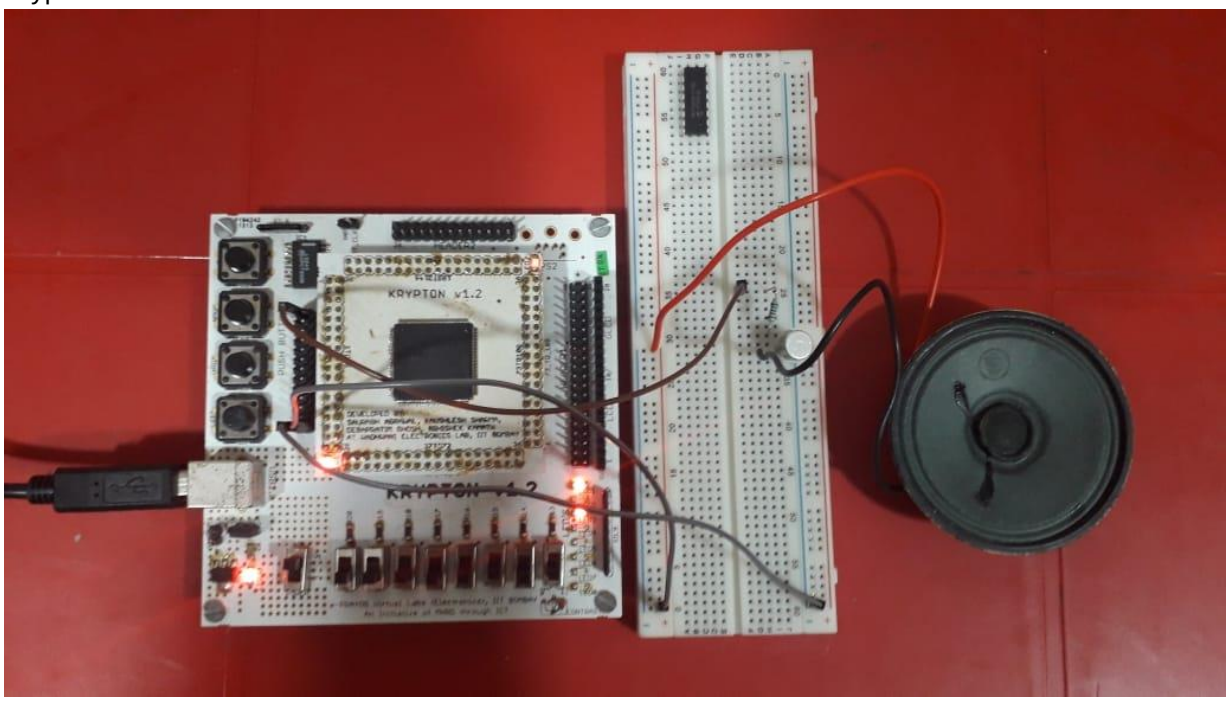
NA

Krypton board:

Pin Planning :



Krypton Board Connections :



Observations:

The speakers play the musical tune as desired. It keeps looping over itself unless a switch is pressed after which it stops.

Video link :

https://drive.google.com/file/d/1jNUXD1_pkg8osc0ClsxDcnC26Gyk7ta3/view?usp=sharing

References:

NA