Experiment 6: Music

Raghav Singhal, 19D070049
EE-214, WEL, IIT Bombay
April 1, 2021

Overview of the experiment:

The purpose of the experiment was to design a circuit with the following specification:
The circuit played notes in a particular sequence, as is done in music. The different notes were generated using the entity designed in the last experiment, while an FSM was used to automate the sequential order and duration of the notes. A switch is used as reset.

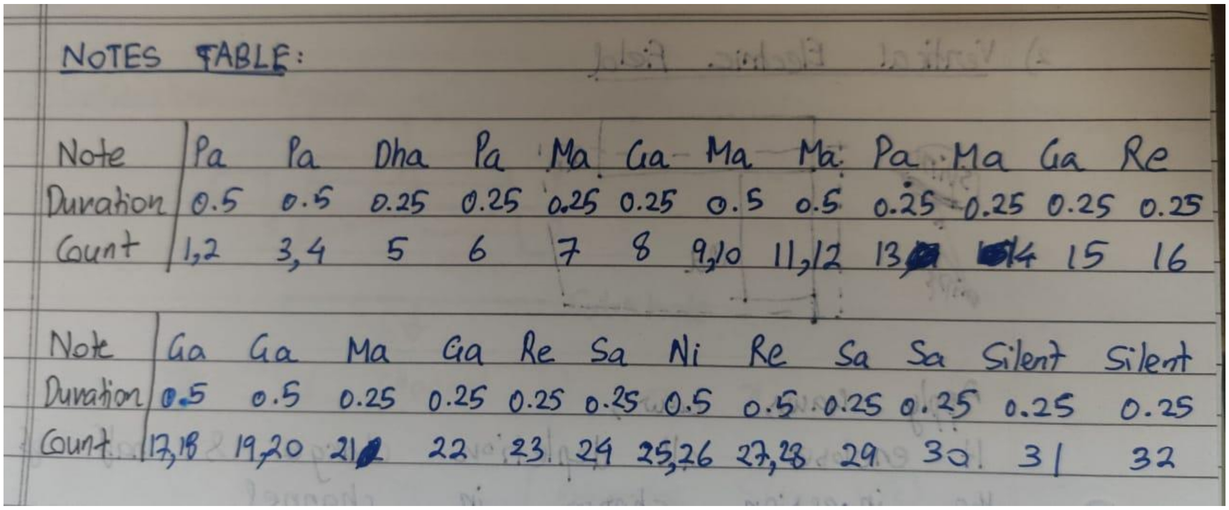
I first designed a logic circuit and drew a schematic to meet the specifications. This was followed by describing the logic circuit in behavioural VHDL, and simulating the circuit using the test-bench to confirm its correct functioning. The circuit was then mapped to the Krypton Board, interfaced with the speaker, and the music was played.

This report will outline the design ideology and results in the following order:

- Approach used to meet the specifications
- Design implementation in VHDL (including snaps of code of the main logic components)
- RTL simulation
- Mapping on the Krypton Board
- Observations

Approach to the experiment:

Notes Table:

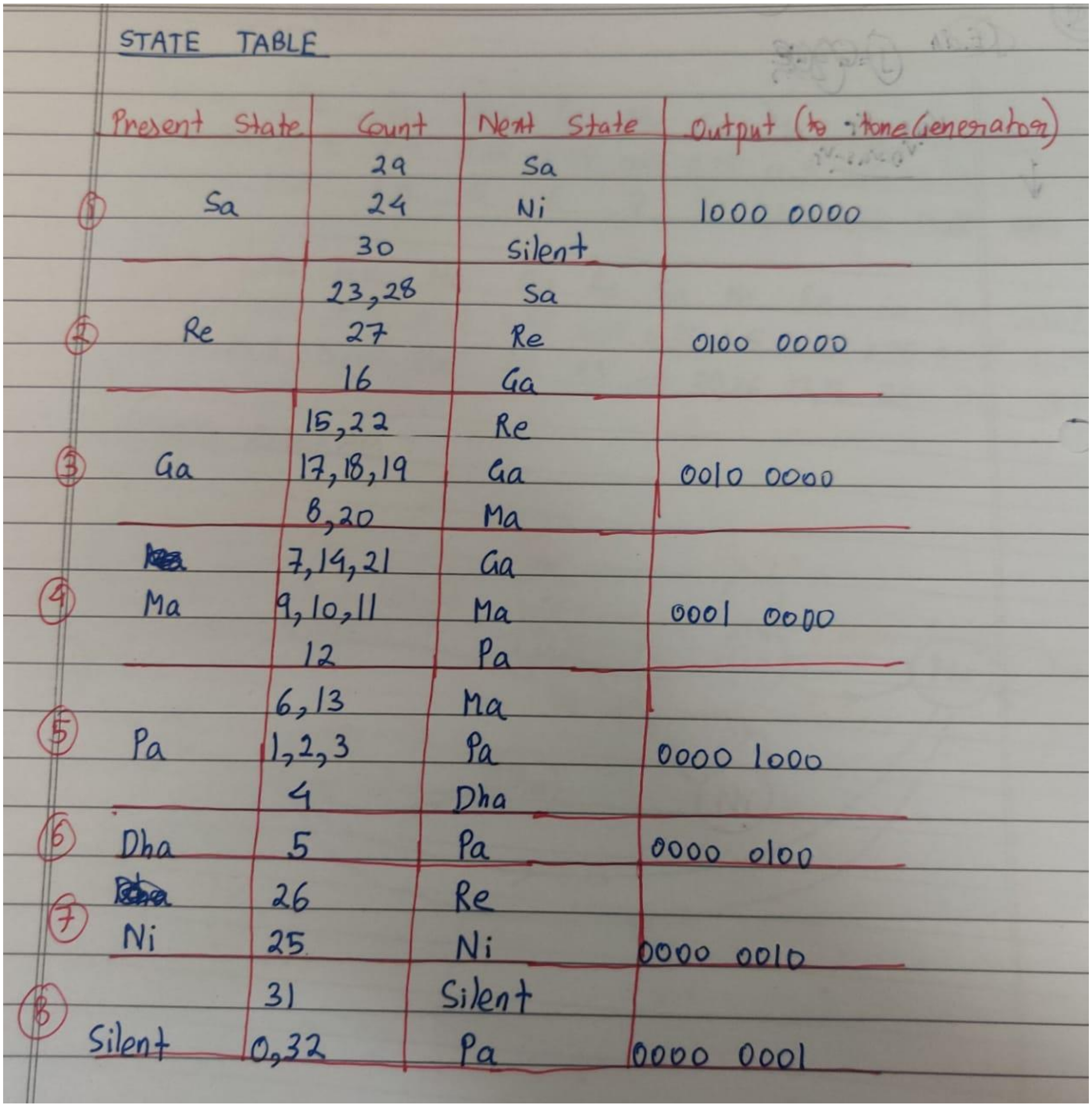


NOTES TABLE:

Note	Pa	Pa	Dha	Pa	Ma	Ga	Ma	Ma	Pa	Ma	Ga	Re
Duration	0.5	0.5	0.25	0.25	0.25	0.25	0.5	0.5	0.25	0.25	0.25	0.25
Count	1,2	3,4	5	6	7	8	9,10	11,12	13,	14	15	16

Note	Ga	Ga	Ma	Ga	Re	Sa	Ni	Re	Sa	Sa	Silent	Silent
Duration	0.5	0.5	0.25	0.25	0.25	0.25	0.5	0.5	0.25	0.25	0.25	0.25
Count	17,18	19,20	21	22	23	24	25,26	27,28	29	30	31	32

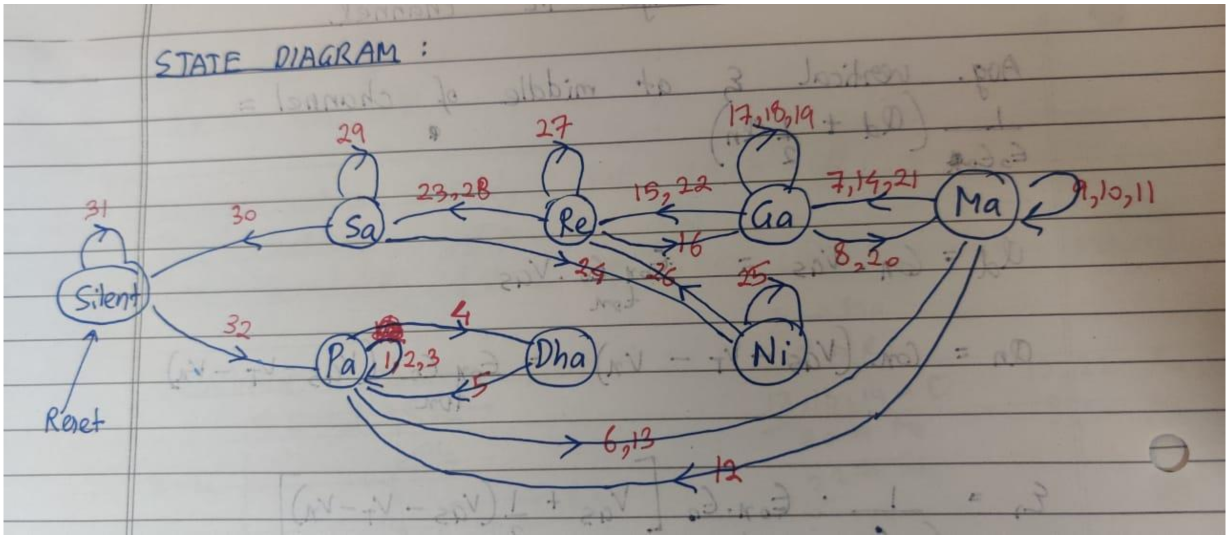
State Table:



STATE TABLE

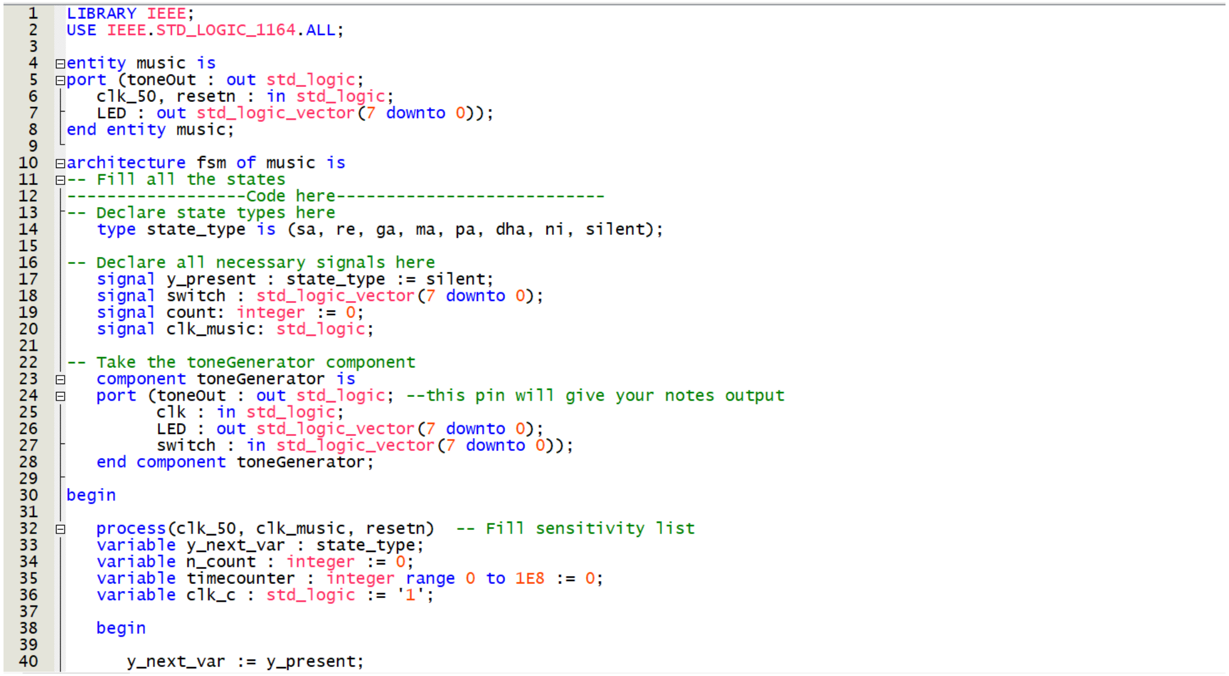
Present State	Count	Next State	Output (to tone Generator)
	29	Sa	
1 Sa	24	Ni	1000 0000
	30	Silent	
	23,28	Sa	
2 Re	27	Re	0100 0000
	16	Ga	
	15,22	Re	
3 Ga	17,18,19	Ga	0010 0000
	8,20	Ma	
	7,14,21	Ga	
4 Ma	9,10,11	Ma	0001 0000
	12	Pa	
	6,13	Ma	
5 Pa	1,2,3	Pa	0000 1000
	4	Dha	
6 Dha	5	Pa	0000 0100
	26	Re	
7 Ni	25	Ni	0000 0010
	31	Silent	
8 Silent	0,32	Pa	0000 0001

State Transition Diagram:

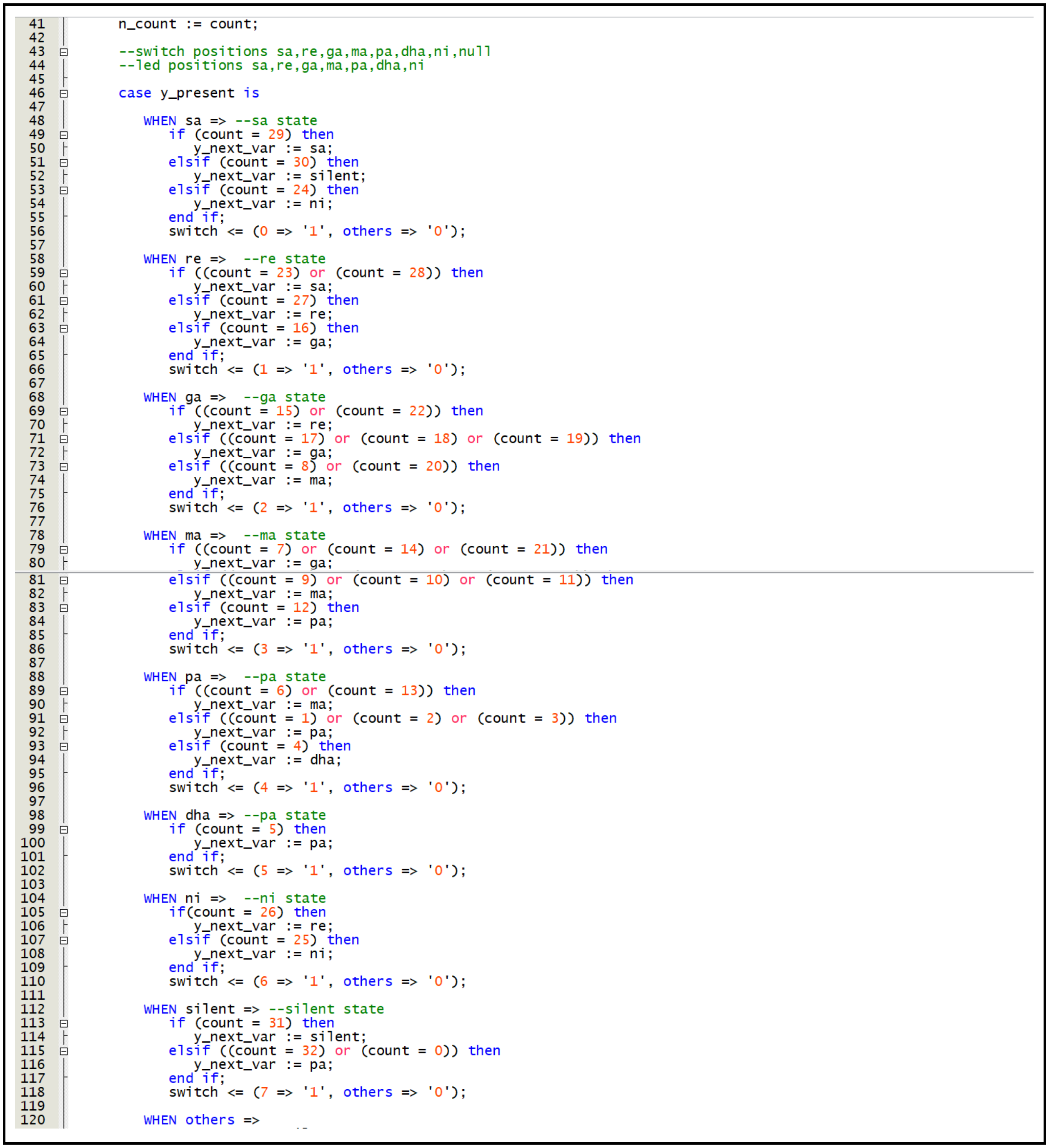


Design document and VHDL code if relevant:

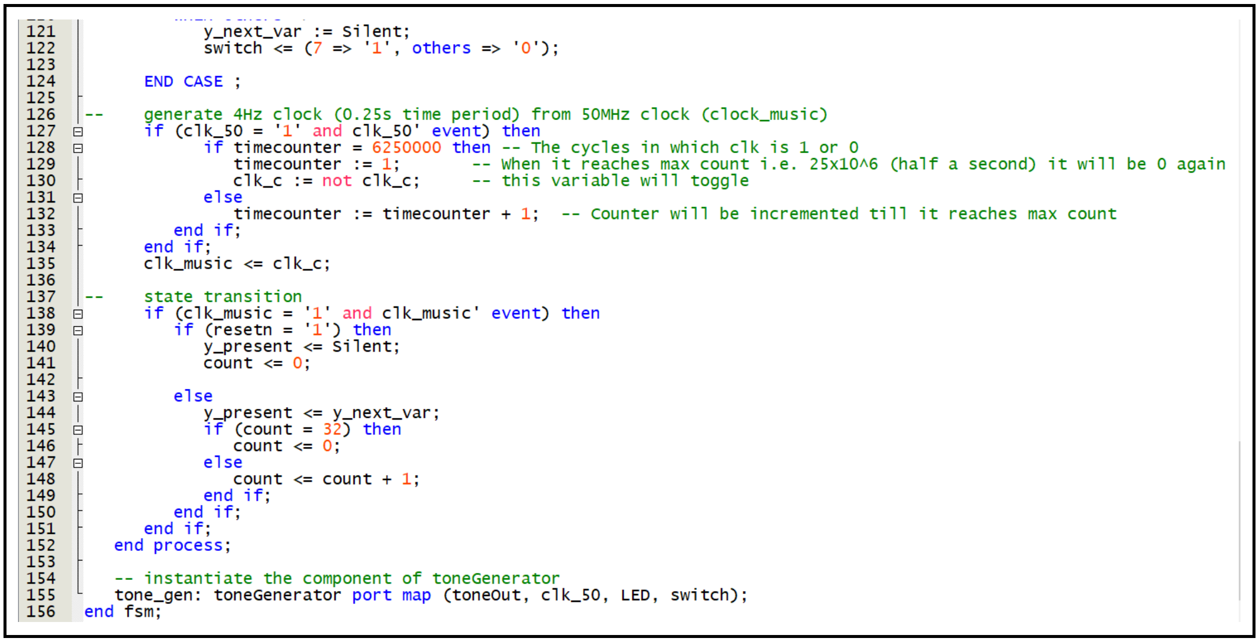
VHDL code for music.vhdl:



```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

entity music is
port (toneOut : out std_logic;
    clk_50, resetn : in std_logic;
    LED : out std_logic_vector(7 downto 0));
end entity music;

architecture fsm of music is
-- Fill all the states
------------------Code here---------------------------
-- Declare state types here
   type state_type is (sa, re, ga, ma, pa, dha, ni, silent);

-- Declare all necessary signals here
   signal y_present : state_type := silent;
   signal switch : std_logic_vector(7 downto 0);
   signal count: integer := 0;
   signal clk_music: std_logic;

-- Take the toneGenerator component
   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 toneGenerator;

begin

   process(clk_50, clk_music, resetn)  -- Fill sensitivity list
   variable y_next_var : state_type;
   variable n_count : integer := 0;
   variable timecounter : integer range 0 to 1E8 := 0;
   variable clk_c : std_logic := '1';

   begin

      y_next_var := y_present;
```

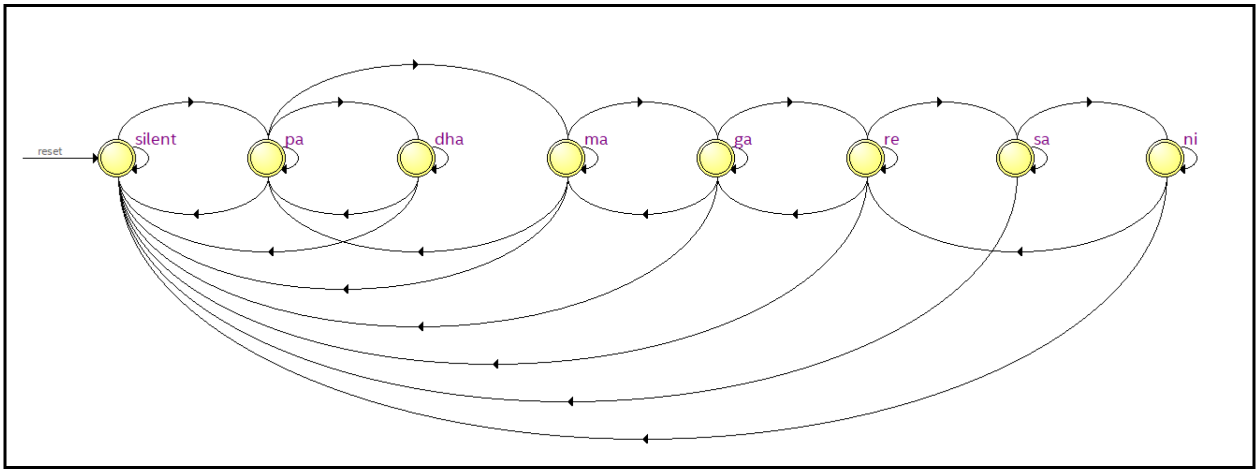


```
        n_count := count;

        --switch positions sa,re,ga,ma,pa,dha,ni,null
        --led positions sa,re,ga,ma,pa,dha,ni

        case y_present is

           WHEN sa => --sa state
              if (count = 29) then
                 y_next_var := sa;
              elsif (count = 30) then
                 y_next_var := silent;
              elsif (count = 24) then
                 y_next_var := ni;
              end if;
              switch <= (0 => '1', others => '0');

           WHEN re =>  --re state
              if ((count = 23) or (count = 28)) then
                 y_next_var := sa;
              elsif (count = 27) then
                 y_next_var := re;
              elsif (count = 16) then
                 y_next_var := ga;
              end if;
              switch <= (1 => '1', others => '0');

           WHEN ga =>  --ga state
              if ((count = 15) or (count = 22)) then
                 y_next_var := re;
              elsif ((count = 17) or (count = 18) or (count = 19)) then
                 y_next_var := ga;
              elsif ((count = 8) or (count = 20)) then
                 y_next_var := ma;
              end if;
              switch <= (2 => '1', others => '0');

           WHEN ma =>  --ma state
              if ((count = 7) or (count = 14) or (count = 21)) then
                 y_next_var := ga;
              elsif ((count = 9) or (count = 10) or (count = 11)) then
                 y_next_var := ma;
              elsif (count = 12) then
                 y_next_var := pa;
              end if;
              switch <= (3 => '1', others => '0');

           WHEN pa =>  --pa state
              if ((count = 6) or (count = 13)) then
                 y_next_var := ma;
              elsif ((count = 1) or (count = 2) or (count = 3)) then
                 y_next_var := pa;
              elsif (count = 4) then
                 y_next_var := dha;
              end if;
              switch <= (4 => '1', others => '0');

           WHEN dha => --pa state
              if (count = 5) then
                 y_next_var := pa;
              end if;
              switch <= (5 => '1', others => '0');

           WHEN ni =>  --ni state
              if(count = 26) then
                 y_next_var := re;
              elsif (count = 25) then
                 y_next_var := ni;
              end if;
              switch <= (6 => '1', others => '0');

           WHEN silent => --silent state
              if (count = 31) then
                 y_next_var := silent;
              elsif ((count = 32) or (count = 0)) then
                 y_next_var := pa;
              end if;
              switch <= (7 => '1', others => '0');

           WHEN others =>
```



```
            y_next_var := Silent;
            switch <= (7 => '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 timecounter = 6250000 then -- The cycles in which clk is 1 or 0
               timecounter := 1;       -- When it reaches max count i.e. 25x10^6 (half a second) it will be 0 again
               clk_c := not clk_c;     -- this variable will toggle
            else
               timecounter := timecounter + 1;  -- Counter will be incremented till it reaches max count
         end if;
      end if;
      clk_music <= clk_c;

--    state transition
      if (clk_music = '1' and clk_music' event) then
         if (resetn = '1') then
            y_present <= Silent;
            count <= 0;

         else
            y_present <= y_next_var;
            if (count = 32) then
               count <= 0;
            else
               count <= count + 1;
            end if;
         end if;
      end if;
   end process;

   -- instantiate the component of toneGenerator
   tone_gen: toneGenerator port map (toneOut, clk_50, LED, switch);
end fsm;
```

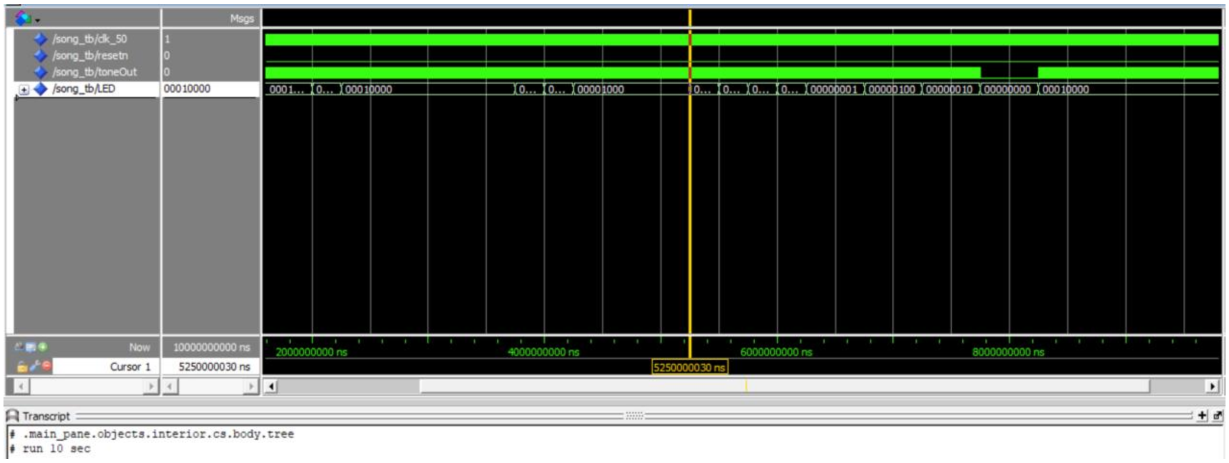
State Machine Viewer:



DUT Input/Output Format:

Not needed for this experiment.

RTL Simulation:



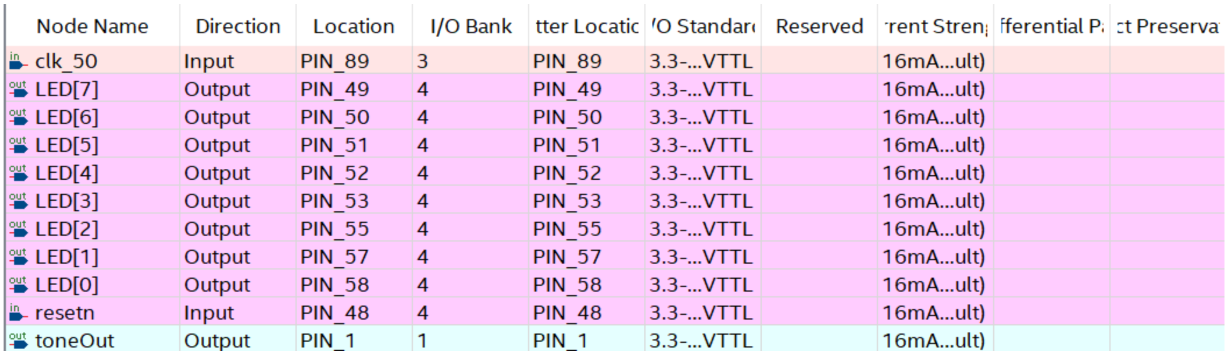
```
# .main_pane.objects.interior.cs.body.tree
# run 10 sec
```

Gate-level Simulation:

Not needed for this experiment.

Krypton board:

Screenshot of Pin Planning:



Node Name	Direction	Location	I/O Bank	tter Locatic	'O Standar	Reserved	·rent Stren	fferential P	ct Preserva
clk_50	Input	PIN_89	3	PIN_89	3.3-...VTTL		16mA...ult)		
LED[7]	Output	PIN_49	4	PIN_49	3.3-...VTTL		16mA...ult)		
LED[6]	Output	PIN_50	4	PIN_50	3.3-...VTTL		16mA...ult)		
LED[5]	Output	PIN_51	4	PIN_51	3.3-...VTTL		16mA...ult)		
LED[4]	Output	PIN_52	4	PIN_52	3.3-...VTTL		16mA...ult)		
LED[3]	Output	PIN_53	4	PIN_53	3.3-...VTTL		16mA...ult)		
LED[2]	Output	PIN_55	4	PIN_55	3.3-...VTTL		16mA...ult)		
LED[1]	Output	PIN_57	4	PIN_57	3.3-...VTTL		16mA...ult)		
LED[0]	Output	PIN_58	4	PIN_58	3.3-...VTTL		16mA...ult)		
resetn	Input	PIN_48	4	PIN_48	3.3-...VTTL		16mA...ult)		
toneOut	Output	PIN_1	1	PIN_1	3.3-...VTTL		16mA...ult)		

Observations:

The RTL simulation gave a correct result. After mapping the circuit to the Krypton Board, the music played was also as expected, demonstrating the correctness of the design.

References:

Lectures and Tutorials by the EE 214 team
Lectures by Prof. M.P. Desai