Experiment 6: Music Synthesizer

Vineet Deepak Gala
Roll Number 190070072
EE-214, WEL, IIT Bombay
March 31, 2021

Overview of the experiment:

Aim – To design a music synthesizer that plays a particular set of music notes to make a tune. We have used an FSM for this.

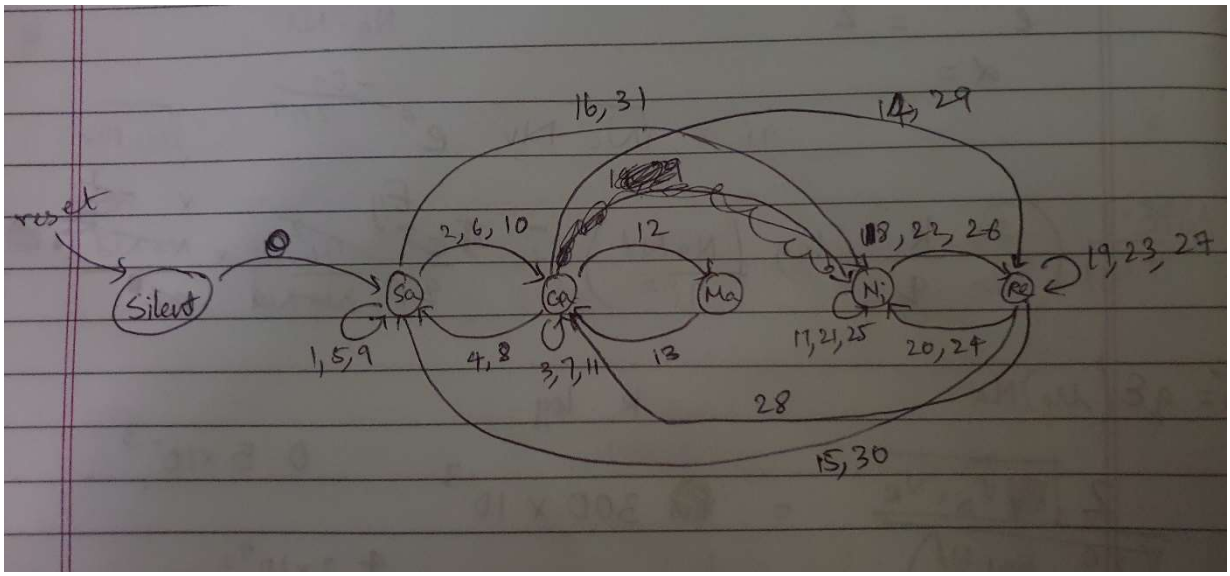
First, we have total 33 counts of notes in one cycle of the tune. Depending on current count, we have to assign next note, increment count and loopback from 32 to 0. Diff states in FSM are diff notes we have to play. We also made a clock divider to generate 4Hz clock to track 0.25 second delays between note changed. Transitions occur on +ve edge of this clock.

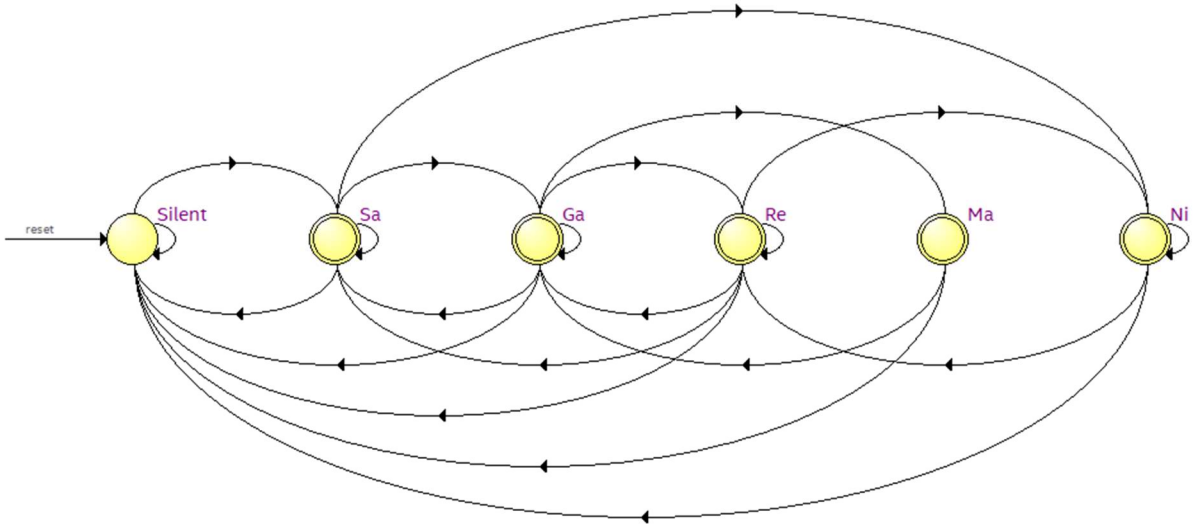
Approach to the experiment:

The table with count values:

Note	Sa	Ga	Sa	Ga	Sa	Ga	Ma	Ga	Re	Sa
Duration (s)	0.5	0.5	0.5	0.5	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	Ni	Re	Ni	Re	Ni	Re	Ga	Re	Sa	Ni
Duration (s)	0.5	0.5	0.5	0.5	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





Design document and VHDL code if relevant:

toneGenerator: This produces the corresponding tone according to the given input.
song_tb: This acts as the testbench code for the whole design
music: This is the main design which implements a FSM and finally outputs the needed music and also the LED signal which glows up the correct LED corresponding to the correct note.

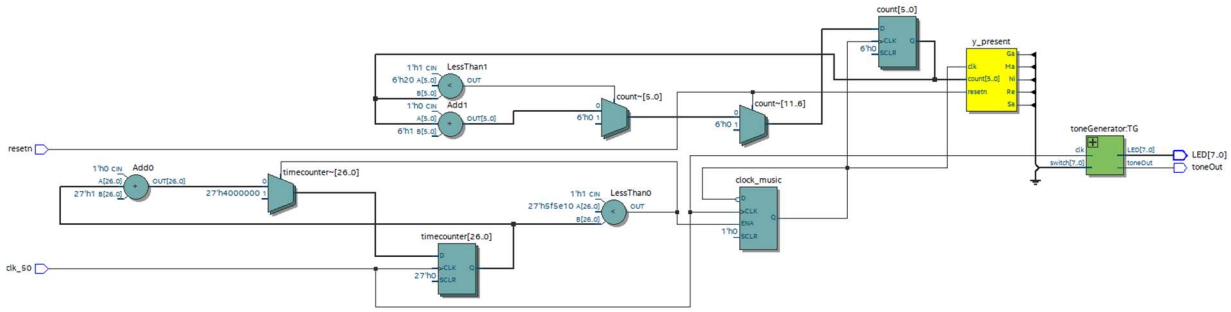
```
ntity music is
port (toneOut : out std_logic;
        clk_50, resetn : in std_logic;
        LED : out std_logic_vector(7 downto 0));
end entity music;
```

```
architecture fsm of music is
-- Fill all the states
------------------Code here--------------------------
-- Declare state types here
type state_type is (Silent, Ni, Sa, Re, Ga, Ma);
------------------Code here--------------------------
-- Declare all necessary signals here
signal y_present : state_type := Silent;
signal clock_music : std_logic := '1';
signal switch : std_logic_vector(7 downto 0);

------------------Code here--------------------------
-- Take the toneGenerator component
component toneGenerator is
port (toneOut : out std_logic;
                clk : in std_logic;
                LED : out std_logic_vector(7 downto 0);
                switch : in std_logic_vector(7 downto 0));
end component toneGenerator;
------------------Code here--------------------------
begin
        process(clk_50,resetn,clock_music)       -- Fill sensitivity list
                variable count : integer range 0 to 32 := 0;
                variable y_next_var : state_type :=Silent;
                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=>
                                switch <= (0=>'0',others=>'0');
                                y_next_var := Sa;
                        ------------------Code here--------------------------
                        ------------------Code here--------------------------
                                --assign the signal for switch which will be the input of toneGenerator
component
                        ------------------code here--------------------------

                        WHEN Sa =>    --if the machine in Sa state
                                if((count = 1) or (count = 5) or (count = 9)) then
                                        y_next_var:=Sa;
                                elsif((count = 2) or (count = 6) or (count = 10)) then
                                        y_next_var:=Ga;
                                elsif((count = 16) or (count=31)) then
                                        y_next_var:=Ni;
                                end if;
                                switch <= (0 => '1', others => '0');
                                --assign the signal for switch which will be the input of toneGenerator
                        WHEN Ga =>
                                if((count = 3) or (count = 7) or (count = 11)) then
                                        y_next_var:=Ga;
```

```
                                elsif((count = 4) or (count = 8)) then
                                        y_next_var:=Sa;
                                elsif((count = 14) or (count = 29)) then
                                        y_next_var:=Re;
                                elsif(count = 12) then
                                        y_next_var:=Ma;
                                end if;
                                switch <= (2 => '1', others => '0');
                        ------------------Code here--------------------------
                        ------------------Code here--------------------------
                        WHEN Re =>
                                if((count = 23) or (count = 27) or (count = 19)) then
                                        y_next_var:=Re;
                                elsif((count = 20) or (count = 24)) then
                                        y_next_var:=Ni;
                                elsif((count = 15) or (count = 30)) then
                                        y_next_var:=Sa;
                                elsif(count = 28) then
                                        y_next_var:=Ga;
                                end if;
                                switch <= (1 => '1', others => '0');
                        WHEN Ni =>
                                if((count = 21) or (count = 17) or (count = 25)) then
                                        y_next_var:=Ni;
                                elsif((count = 18) or (count = 22) or (count = 26)) then
                                        y_next_var:=Re;
                                elsif(count = 32) then
                                        y_next_var:=Silent;
                                end if;
                                switch <= (6 => '1', others => '0');

                        WHEN Ma=>
                                y_next_var:=Ga;
                                switch <= (3 => '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
                                timecounter:=1;
                                clock_music <= not clock_music;
                        else
                                timecounter:=timecounter+1;
                        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;
                                if(count>=32) then
```

```
                                        count :=0;
                                    else
                                        count := count + 1;
                                    end if;
                                end if;
                            end if;
        end process;
        TG: toneGenerator
        port map(toneOut,clk_50,LED,switch);

        -- instantiate the component of toneGenerator
end fsm;
```

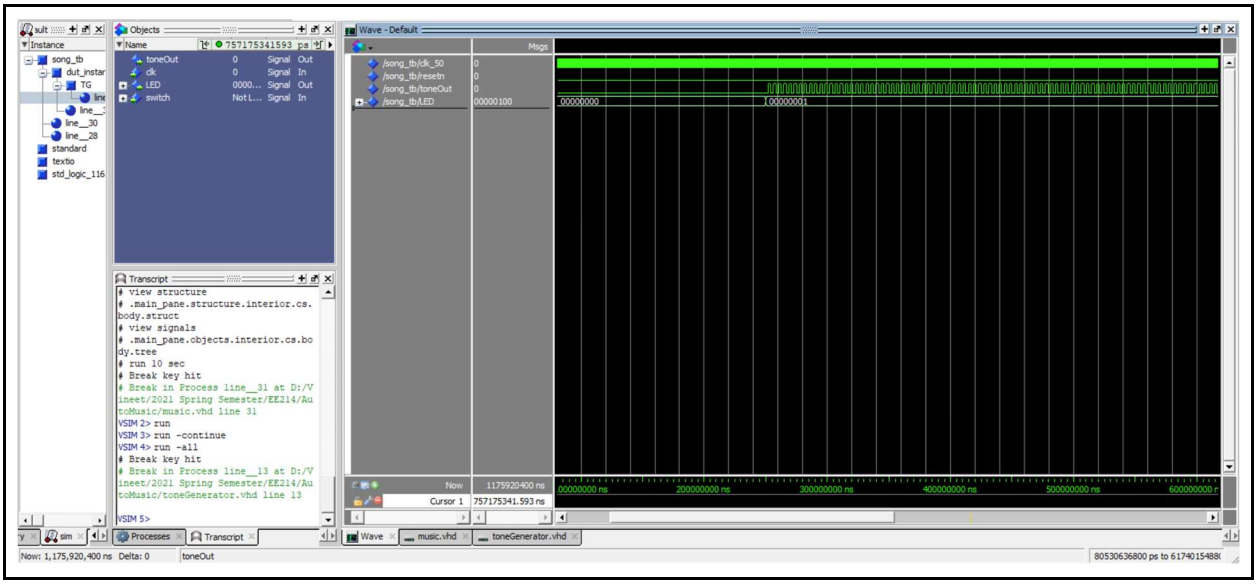
RTL View:



DUT Input/Output Format:

Switch 1 is used for reset input, PIN 89 is used to get 50MHz clock. For the toneGenerator, we use a variable to pass input to it.
Output to the speaker is on PIN 1 and 8 LEDs corresponding to each note is also a part of output.

RTL Simulation:

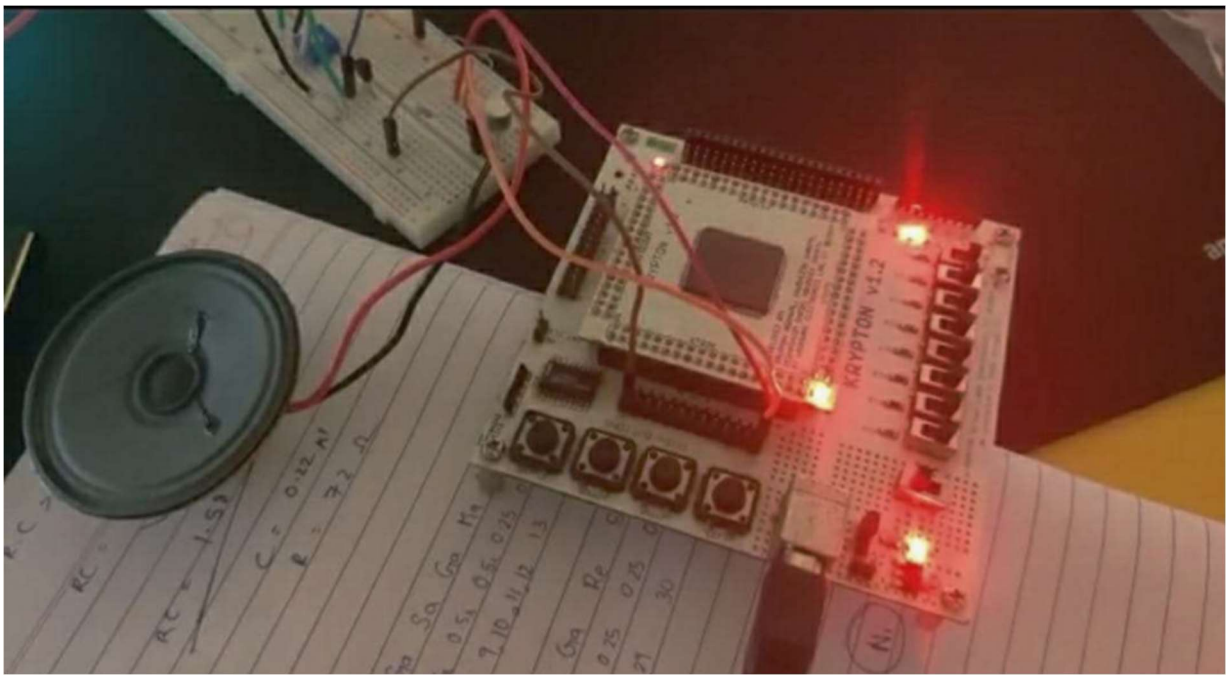


Gate-level Simulation:

Attach the clearly visible screen-shot of Gate-level Simulation.

Krypton board*:

Node Name	Direction	Location	I/O Bank	itter Locatior	I/O Standard	Reserved	ırrent Streng
clk_50	Input	PIN_89	3	PIN_89	3.3-V LVTTL		16mA ...ault)
LED[7]	Output	PIN_49	4	PIN_49	3.3-V LVTTL		16mA ...ault)
LED[6]	Output	PIN_50	4	PIN_50	3.3-V LVTTL		16mA ...ault)
LED[5]	Output	PIN_51	4	PIN_51	3.3-V LVTTL		16mA ...ault)
LED[4]	Output	PIN_52	4	PIN_52	3.3-V LVTTL		16mA ...ault)
LED[3]	Output	PIN_53	4	PIN_53	3.3-V LVTTL		16mA ...ault)
LED[2]	Output	PIN_55	4	PIN_55	3.3-V LVTTL		16mA ...ault)
LED[1]	Output	PIN_57	4	PIN_57	3.3-V LVTTL		16mA ...ault)
LED[0]	Output	PIN_58	4	PIN_58	3.3-V LVTTL		16mA ...ault)
resetn	Input	PIN_39	4	PIN_39	3.3-V LVTTL		16mA ...ault)
toneOut	Output	PIN_1	1	PIN_1	3.3-V LVTTL		16mA ...ault)



Observations*:

Correct output heard on speakers and correct LEDs glowed

References:

You may include the references if any.

* To be submitted after the tutorial on ”Using Krypton.