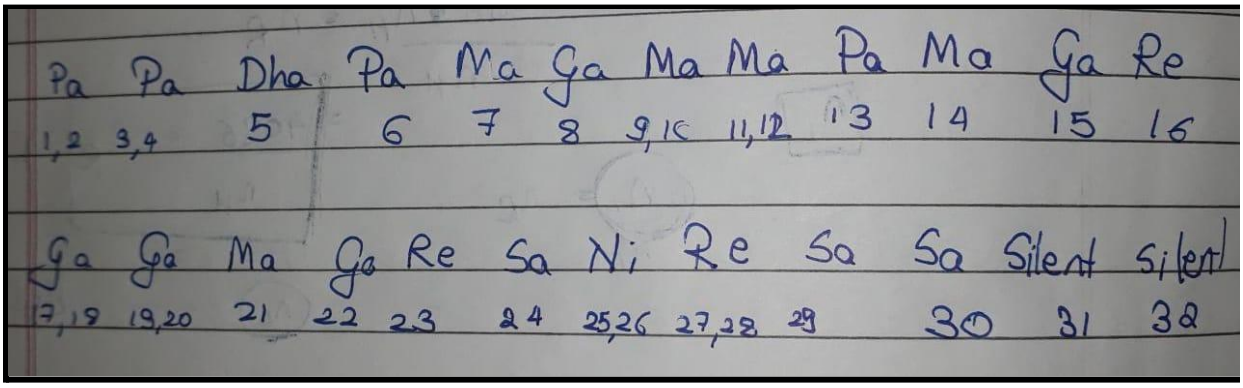
Experiment 6: Music Synthesizer

Nirmal Shah, Roll Number 190100085
EE-214, WEL, IIT Bombay
April 1, 2021

Overview of the experiment:

In this experiment, we played music using the notes which were introduced in the class.
The notes are given in a particular sequence like in music. We were needed to use FSM to automate such that notes will be played one after another in the given sequence and for a particular duration.

This is the table that was needed to follow



In this, each count corresponds to 0.25 seconds.

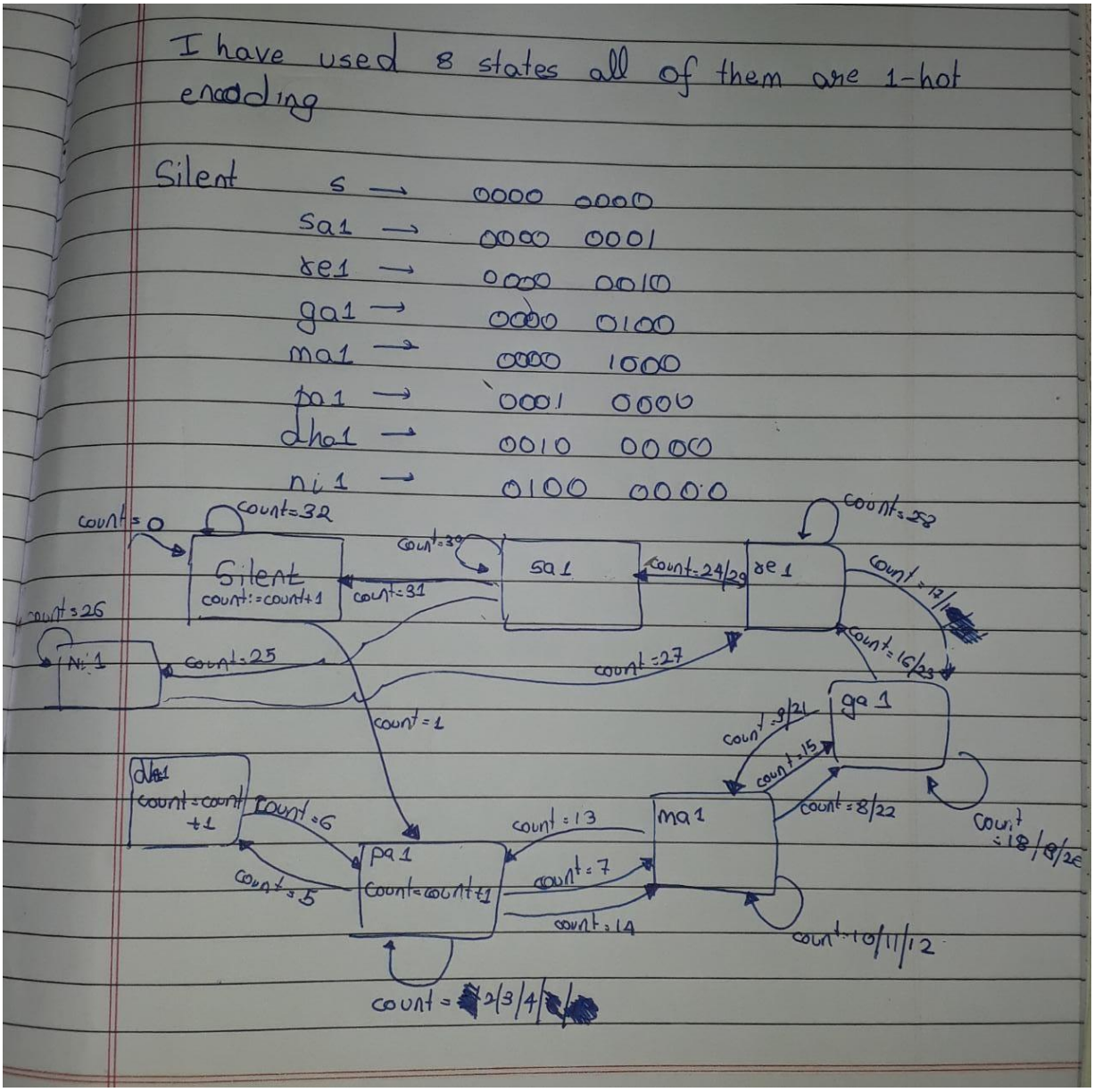
We also needed to make a clock(clock_music) with a frequency of 4 Hz(1/0.25s) from the master clock of 50MHz.

All the state transition and everything will happen at the rising edge of clock music
These distinct notes will be the states of the FSM. There will be an initial state(reset) which is the silent state.
At reset, the count will be 0, thereafter, at the first rising edge of the clock music, the count is incremented to 1 and the note Pa, will be played, for the second rising edge the count value gets incremented to 2, however, the corresponding note is again 'Pa'. At the fifth rising edge of the clock music, when the count is incremented to 5, the note 'Dha' should be played..and this will be continued.

State Table:-

Count	Present State	Next state
1	Pa	Pa
2	Pa	Pa
3	Pa	Pa
4	Pa	Dha
5	Dha	Pa
6	Pa	Ma
7	Ma	Ga
8	Ga	Ma
9	Ma	Ma
10	Ma	Ma
11	Ma	Ma
12	Ma	Pa
13	Pa	Ma
14	Ma	Ga
15	Ga	Re
16	Re	Ga
17	Ga	Ga
18	Ga	Ga
19	Ga	Ga
20	Ga	Ma
21	Ma	Ga
22	Ga	Re
23	Re	Sa
24	Sa	Ni
25	Ni	Ni
26	Ni	Re
27	Re	Re
28	Re	Sa
29	Sa	Sa
30	Sa	Silent
31	Silent	Silent
32	Silent	Reset

Approach to the experiment:



Design document and VHDL code if relevant:

```
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;
```

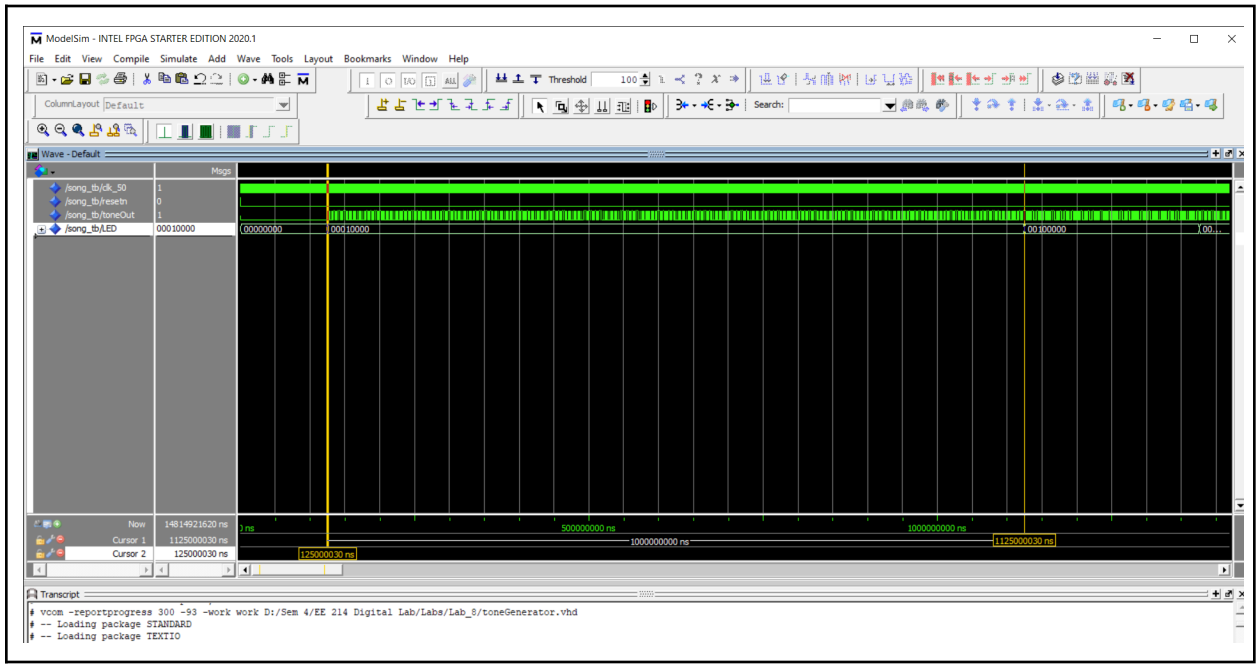
```
architecture fsm of music is
-- Fill all the states
type state_type is (Silent,sa,re,ga,ma,pa,dha,ni);

signal y_present : state_type:=Silent;
signal count        : integer          :=0;
signal clock_music        : std_logic:='0';

constant s:std_logic_vector(7 downto 0):=(others=>'0');
constant sa1 : std_logic_vector(7 downto 0):=(0=>'1', others=>'0');
constant re1 : std_logic_vector(7 downto 0):=(1=>'1', others=>'0');
constant ga1 : std_logic_vector(7 downto 0):=(2=>'1', others=>'0');
constant ma1 : std_logic_vector(7 downto 0):=(3=>'1', others=>'0');
constant pa1 : std_logic_vector(7 downto 0):=(4=>'1', others=>'0');
constant dha1 : std_logic_vector(7 downto 0):=(5=>'1', others=>'0');
constant ni1 : std_logic_vector(7 downto 0):=(6=>'1', others=>'0');

signal switch                : std_logic_vector(7 downto 0):=s;

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,count,clock_music)      -- Fill sensitivity list
        variable y_next_var : state_type := Silent;
        variable n_count : integer := 0;
        variable timecounter : integer range 0 to 1E8 := 0;

        variable clk_c                        : std_logic:='0';

        begin

                y_next_var := y_present;
                n_count := count;

                case y_present is
                        when Silent=>
                                switch<=s;
                                if(count = 0) then
                                        y_next_var:=pa;
                                        n_count := count + 1;
                                elsif(count = 31) then
                                        y_next_var:=Silent;
                                        n_count:=count+1;
                                elsif(count = 32) then
                                        y_next_var:=pa;
                                        n_count:=1;

                                else
                                        y_next_var:=pa;
```

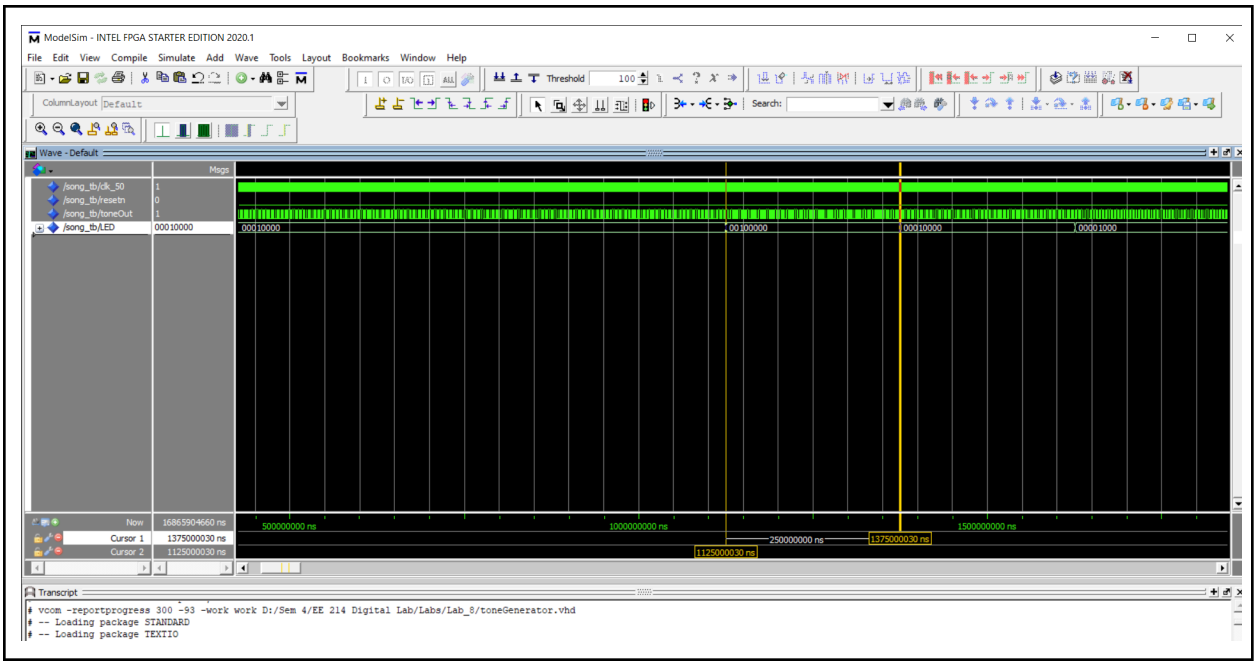
```
                n_count:=1;
        end if;


WHEN sa =>      --if the machine in Sa state
        switch<=sa1;


        if((count = 24)) then
                y_next_var:=ni;
                n_count := count + 1;
        elsif(count=29) then
                y_next_var:= sa;
                n_count := count + 1;
        elsif(count = 30) then
                y_next_var:=Silent;
                n_count := count + 1;
        else
                y_next_var:=sa;
                n_count:=count+1;
        end if;


WHEN re =>
        switch<=re1;

        if(count = 16) then
                y_next_var:=ga;
                n_count := count + 1;
        elsif((count = 23) or (count = 28)) then
                y_next_Var:=sa;
                n_count := count + 1;
        elsif(count = 27) then
                y_next_var:=re;
                n_count := count + 1;
        end if;

WHEN ga =>
        switch<=ga1;

        if(count = 8) then
                y_next_var:=ma;
                n_count := count + 1;
        elsif((count = 15) or (count = 22))then
                y_next_Var:=re;
                n_count := count + 1;
        elsif((count = 17) or (count = 18) or (count = 19)) then
                y_next_var:=ga;
                n_count := count + 1;
        elsif(count = 20) then
                y_next_var:=ma;
                n_count:=count+1;
        end if;

        WHEN ma =>
```

```
                        switch<=ma1;

                        if((count = 7) or (count = 14) or (count = 21)) then
                                y_next_var:=ga;
                                n_count := count + 1;
                        elsif((count = 9) or (count = 10) or (count = 11))then
                                y_next_Var:=ma;
                                n_count := count + 1;
                        elsif((count = 12)) then
                                y_next_var:=pa;
                                n_count := count + 1;
                        end if;

                        WHEN pa =>
                        switch<=pa1;

                        if((count = 1) or (count = 2) or (count = 3)) then
                                y_next_var:=pa;
                                n_count := count + 1;
                        elsif((count = 4))then
                                y_next_Var:=dha;
                                n_count := count + 1;
                        elsif((count = 6) or (count = 13)) then
                                y_next_var:=ma;
                                n_count := count + 1;
                        end if;

                        WHEN dha =>
                        switch<=dha1;

                        if((count = 5)) then
                                y_next_var:=pa;
                                n_count := count + 1;
                        end if;

                        WHEN ni =>
                        switch<=ni1;

                        if((count = 25)) then
                                y_next_var:=ni;
                                n_count := count + 1;
                        elsif((count = 26))then
                                y_next_Var:=re;
                                n_count := count + 1;
                        end if;

                WHEN others =>
                                        y_next_var:=Silent;
                                        n_count:=0;

        END CASE ;

        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
```

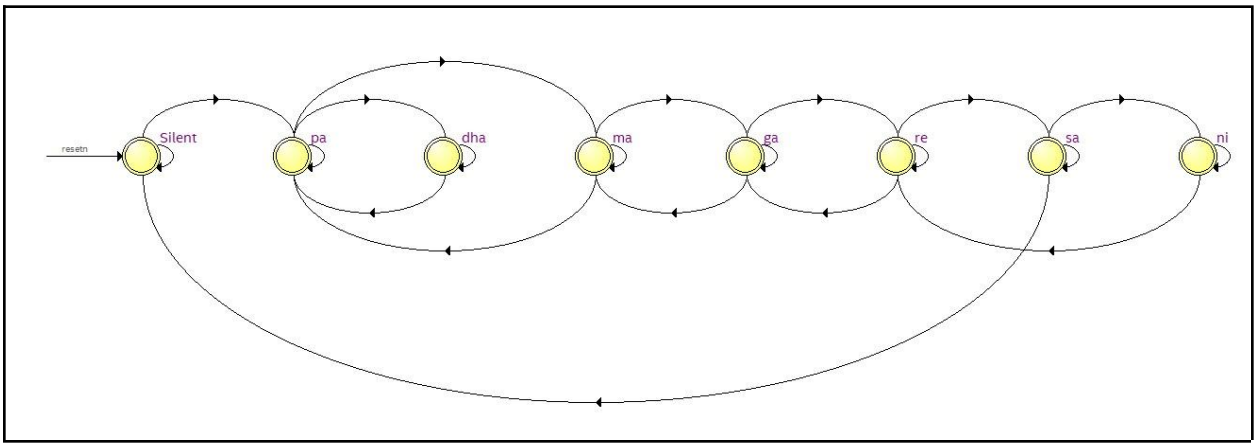
```
                                        timecounter := 1;                       -- When it reaches
max count i.e. 25x10^6 (half a second) it will be 0 again
                                        clk_c := not clk_c;             -- this variable will toggle
                                else
                                        timecounter := timecounter + 1;  -- Counter will be incremented
till it reaches max count

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


                if(resetn = '1') then
                        y_present<= Silent;
                        count<=0;
                        elsif(clock_music = '1' and clock_music' event) then
                                y_present<=y_next_var;
                                count <=n_count;


                end if;
        end process;

i0      : toneGenerator port map(toneOut=>toneOut,clk=>clk_50,LED=>LED,switch=>switch);
end fsm;
```

RTL View:

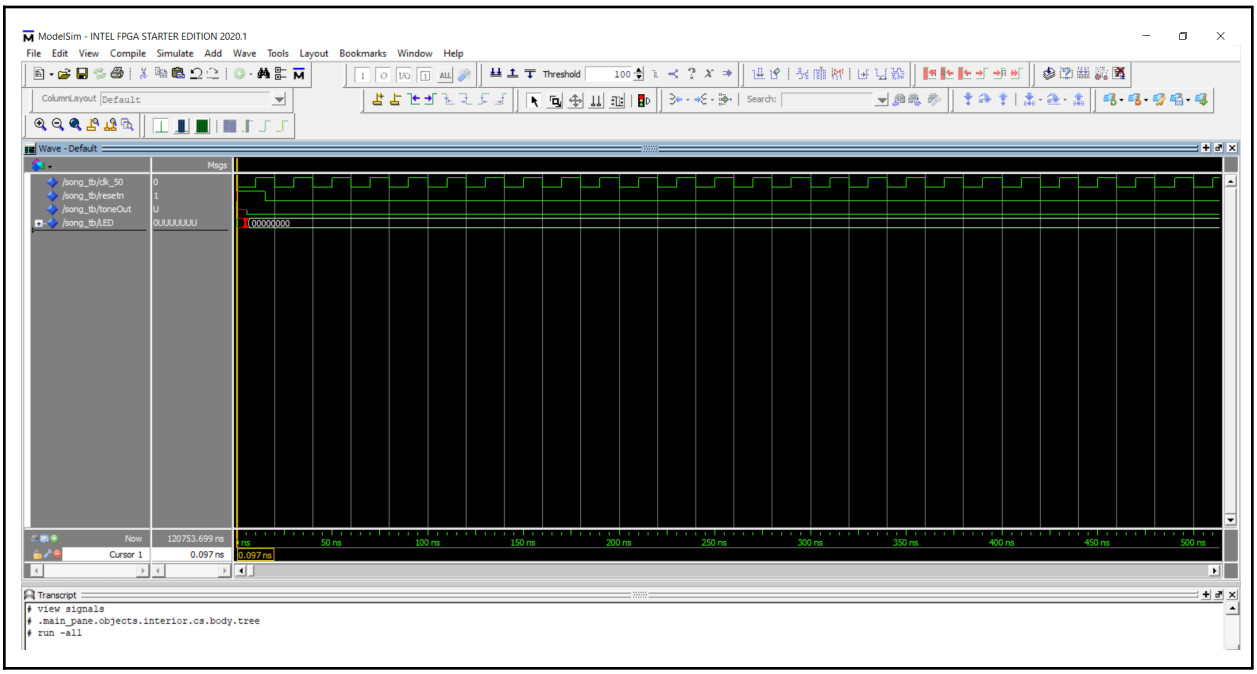




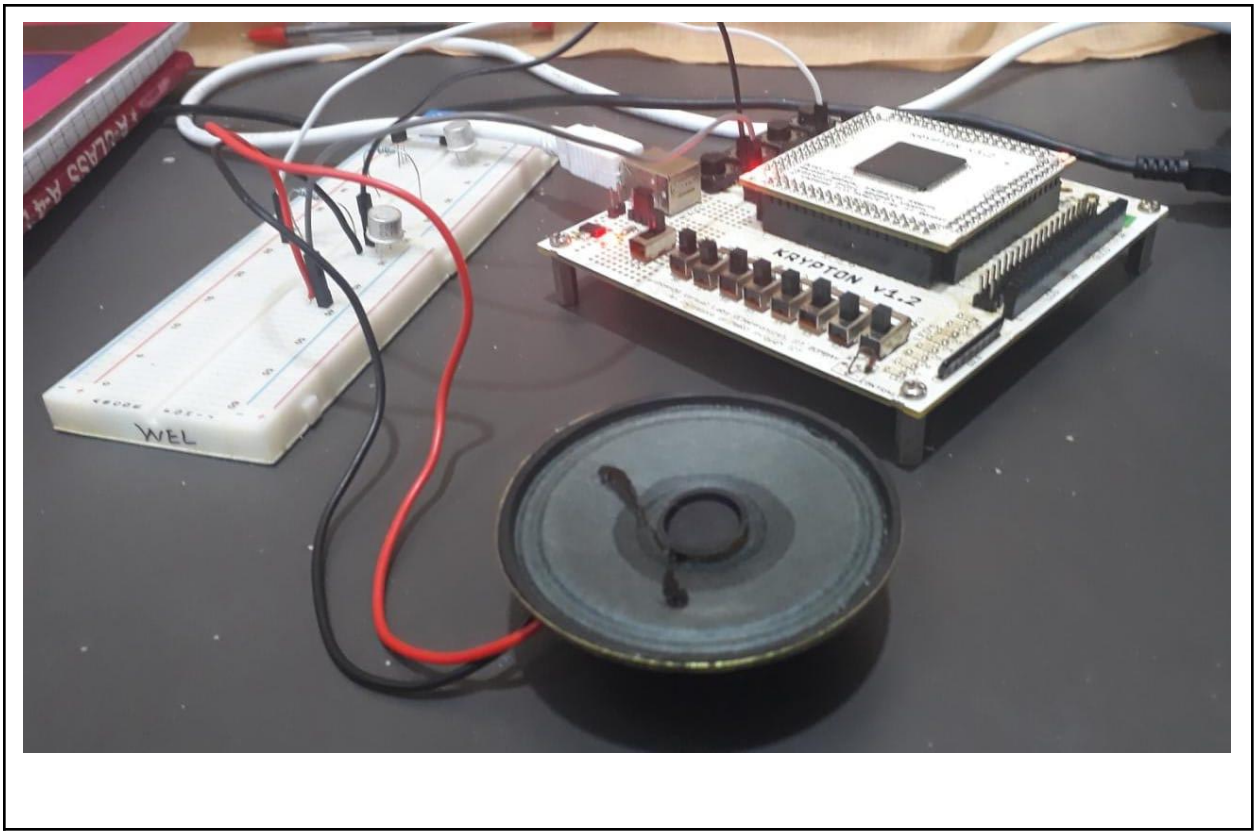
State Machine Viewer:-



Gate-level Simulation:



Krypton board*:



Observations*:

Video can be seen here