# Experiment 5: Tone Generation

Rathour Param Jitendrakumar Roll Number: 190070049
EE-214, WEL, IIT Bombay
March 27, 2021

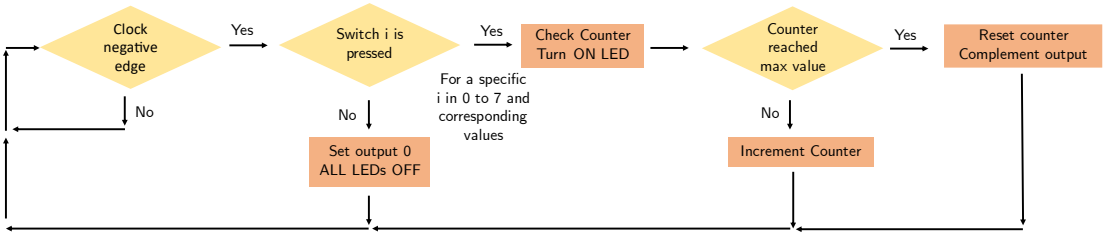
## Overview of the experiment:

The purpose of this experiment is generate the seven major notes in the Indian classical music named sa, re, ga, ma, pa, dha, ni and Sa (upper octave) using clock divider.
Describe it in VHDL using Behavioral Modelling and simulate with the given test-bench using Quartus.

I used the provided skeleton code, designed the circuit accordingly, verified my implementation using Register-transfer level simulation & observed required tones by mapping it to the Krypton board interfaced with speaker.

This report contains my approach to the experiment, the circuit's design with the relevant code followed by the RTL netlist view, PIN planning and design confirmation by output waveforms of RTL.

## Approach to the experiment:




Tone Generation

Appoach as shown in flow chart. One thing to note is that for each switch there are corresponding variables that store max count and current count for that note.
To generate these frequencies, I used clock divider and onboard clock, calculated count for each note using

Count = 50 MHz / (2 * f)

| Note | Frequency (Hz) | Calculations (Truncated till 2 decimal places) | | Count Value (Accurate to nearest integer) |
|---|---|---|---|---|
| Sa | 240 | 50 * 10^6 / (2 * 240) | 104166.66 | 104167 |
| Re | 270 | 50 * 10^6 / (2 * 270) | 92592.59 | 92593 |
| Ga | 300 | 50 * 10^6 / (2 * 300) | 83333.33 | 83333 |
| Ma | 320 | 50 * 10^6 / (2 * 320) | 78125 | 78125 |
| Pa | 360 | 50 * 10^6 / (2 * 360) | 69444.44 | 69444 |
| Dha | 400 | 50 * 10^6 / (2 * 400) | 62500 | 62500 |
| Ni | 450 | 50 * 10^6 / (2 * 450) | 55555.55 | 55556 |
| Sa (upper octave) | 480 | 50 * 10^6 / (2 * 480) | 52083.33 | 52083 |

## Design document and VHDL code if relevant:

This whole design is part of toneGenerator with 9 input bits & 9 output bits to confirm our design.
See toneGenerator Input/Output Format
I am using Behavioral Modelling for this experiment.
Code can be broken down as shown in approach flow chart

Entity declaration

```
library ieee;
use ieee.std_logic_1164.all;

entity 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 entity toneGenerator;
```

Architechture of toneGenerator

```
architecture behaviour of toneGenerator is
begin
    process(clk, switch)
                        -- variables for count and output
    variable count_sa, count_re, count_ga, count_ma, count_pa, count_dha, count_ni, count_saa  : integer range 0 to 1E8 := 1;
    variable sa, re, ga, ma, pa, dha, ni, saa : std_logic := '1';
    begin
        if (clk = '1' and clk' event) then
```

Code for each note

```
if (switch(0) = '1') then        -- Sa
    if (count_sa = 104167) then -- 240Hz
        count_sa := 1;
        sa := not sa;
    else
        count_sa := count_sa + 1;
    end if;
    toneOut <= sa;
    LED <= (0 => '1', others => '0');
```

```
elsif (switch(1) = '1') then    -- Re
    if (count_re = 92593) then -- 270Hz
        count_re := 1;
        re := not re;
    else
        count_re := count_re + 1;
    end if;
    toneOut <= re;
    LED <= (1 => '1', others => '0');
```

```
elsif (switch(2) = '1') then    -- Ga
    if (count_ga = 83333) then -- 300Hz
        count_ga := 1;
        ga := not ga;
    else
        count_ga := count_ga + 1;
    end if;
    toneOut <= ga;
    LED <= (2 => '1', others => '0');
```

```
elsif (switch(3) = '1') then    -- Ma
    if (count_ma = 78125) then -- 320Hz
        count_ma := 1;
        ma := not ma;
    else
        count_ma := count_ma + 1;
    end if;
    toneOut <= ma;
    LED <= (3 => '1', others => '0');
```

```
elsif (switch(4) = '1') then    -- Pa
    if (count_pa = 69444) then -- 360Hz
        count_pa := 1;
        pa := not pa;
    else
        count_pa := count_pa + 1;
    end if;
    toneOut <= pa;
    LED <= (4 => '1', others => '0');
```

```
elsif (switch(5) = '1') then     -- Dha
    if (count_dha = 62500) then -- 400Hz
        count_dha := 1;
        dha := not dha;
    else
        count_dha := count_dha + 1;
    end if;
    toneOut <= dha;
    LED <= (5 => '1', others => '0');
```

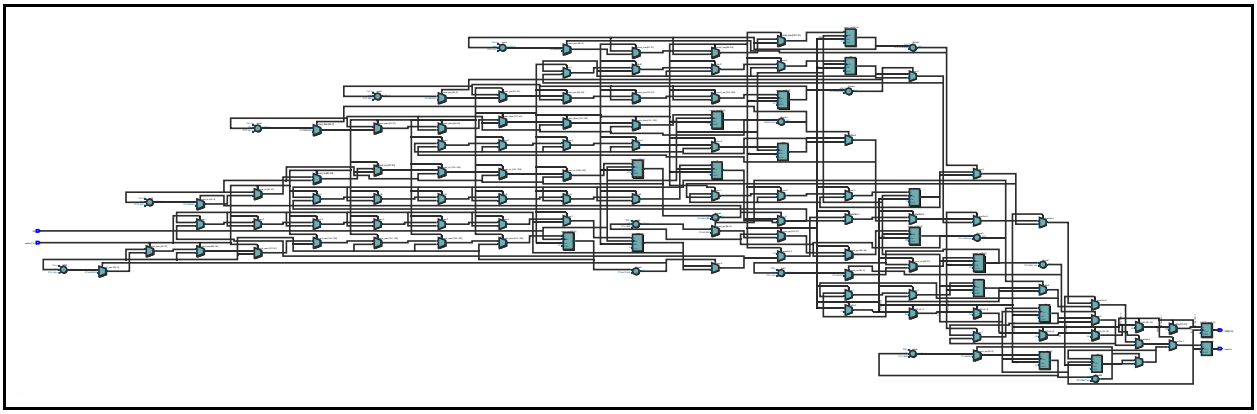
```
elsif (switch(6) = '1') then    -- Ni
    if (count_ni = 55556) then -- 450Hz
        count_ni := 1;
        ni := not ni;
    else
        count_ni := count_ni + 1;
    end if;
    toneOut <= ni;
    LED <= (6 => '1', others => '0');
```

```
elsif (switch(7) = '1') then     -- Saa
    if (count_saa = 52083) then -- 480Hz
        count_saa := 1;
        saa := not saa;
    else
        count_saa := count_saa + 1;
    end if;
    toneOut <= saa;
    LED <= (7 => '1', others => '0');
```

Else condtion

```
            else                            -- when no switch is pressed
                toneOut <= '0';             -- output 0
                LED <= (others => '0');     -- all LED's are off
            end if;
        end if;
    end process;
end behaviour;
```

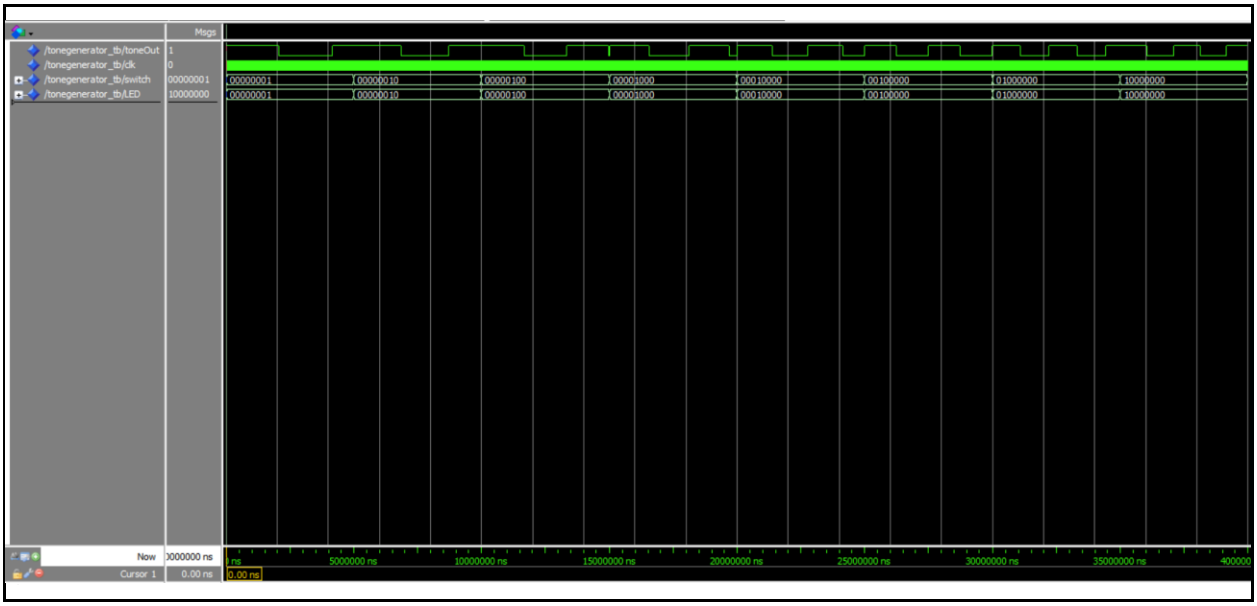
## RTL View:



## toneGenerator Input/Output Format:

| | |
|---|---|
| Input Format: $CS_8S_7S_6S_5S_4S_3S_2S_1$ | (C (Onboard Clock) & 8-bit number S ($S_8$ is MSB $S_1$ is LSB)) |
| Output Format: $OL_8L_7L_6L_5L_4L_3L_2L_1$ | (O is output & 8-bit number L ($L_8$ is MSB $L_1$ is LSB)) |
| Here, when $S_i$ is switched ON, $L_i$ will glow (other LEDs are OFF) and output to speaker oscillates between 0 and 1 according to frequencies of notes. | |

## RTL Simulation:




## Gate-level Simulation:

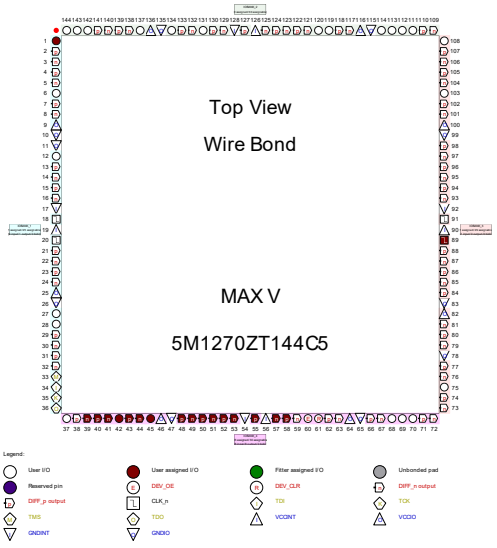
| |
|---|
| Gate-level Simulation takes a lot of time and my computer crashes. |

## Krypton board:

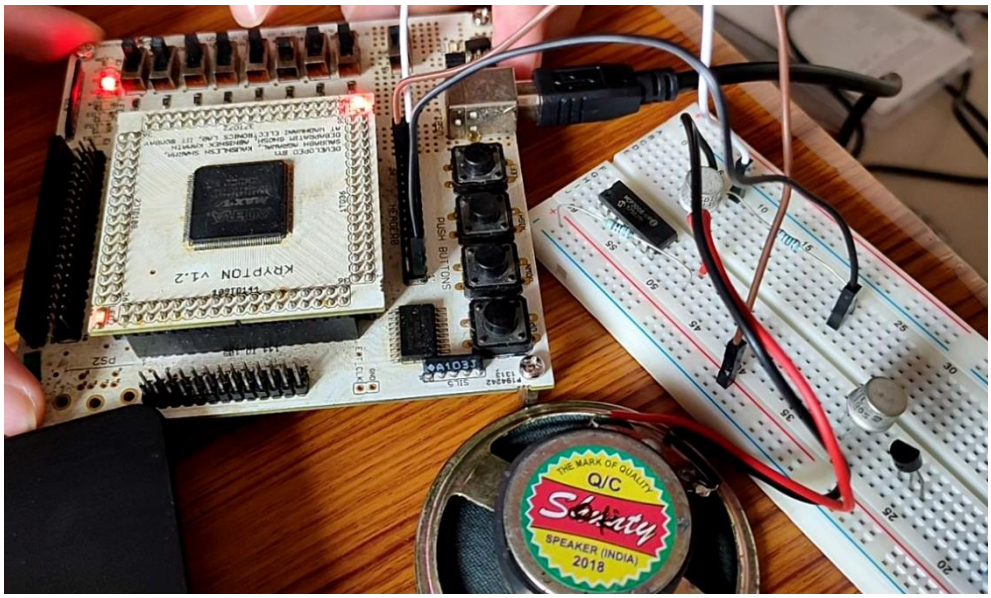
### Pin Planning

(Note: VCC at PIN 26 and GND at PIN 25)

| Node Name | Direction | Location | I/O Bank | Fitter Location |
|---|---|---|---|---|
| clk | Input | PIN_89 | 3 | PIN_89 |
| LED[7] | Output | PIN_58 | 4 | PIN_58 |
| LED[6] | Output | PIN_57 | 4 | PIN_57 |
| LED[5] | Output | PIN_55 | 4 | PIN_55 |
| LED[4] | Output | PIN_53 | 4 | PIN_53 |
| LED[3] | Output | PIN_52 | 4 | PIN_52 |
| LED[2] | Output | PIN_51 | 4 | PIN_51 |
| LED[1] | Output | PIN_50 | 4 | PIN_50 |
| LED[0] | Output | PIN_49 | 4 | PIN_49 |
| switch[7] | Input | PIN_48 | 4 | PIN_48 |
| switch[6] | Input | PIN_45 | 4 | PIN_45 |
| switch[5] | Input | PIN_44 | 4 | PIN_44 |
| switch[4] | Input | PIN_43 | 4 | PIN_43 |
| switch[3] | Input | PIN_42 | 4 | PIN_42 |
| switch[2] | Input | PIN_41 | 4 | PIN_41 |
| switch[1] | Input | PIN_40 | 4 | PIN_40 |
| switch[0] | Input | PIN_39 | 4 | PIN_39 |
| toneOut | Output | PIN_1 | 1 | PIN_1 |




### Board




## Observations:

Observed frequencies from RTL simulation are very close to required frequencies

When multiple switches are on, note of lower frequency is played (given more priority) than higher frequency.

## References:


Tertulien Ndjountche *Digital Electronics 2 Sequential and Arithmetic Logic Circuits* Wiley