Experiment 3 - Function Generator

Ali Bahari,
810196688

Abstract— This document gives the final results of the third experiment of digital logic design laboratory.

I. Introduction

The goal of this experiment is to create arbitrary generator that generates different desired waveforms having distinct shapes as brought up in experiment description. Also we will use the frequency regulator from the previous experiment to create these mentioned waveform at three different frequencies. Besides frequency regulator we will create a new component that is amplitude selector to show the waveform in different ranges meaning generated waveforms can be divided by multiple values to generate new desired amplitude for the waveform. At the end all of the mentioned subjects above are tested and verified.

II. Waveform Generator

In this part we will make and code the waveform generator.

1. The code is provided in the sent files under the folder named 1. In the code based on the shape of the 8 wanted waveforms an action is taken like using a counter or the mentioned formulas for sine function. Also for rhomboid or modulated square wave we will give output based on the counter being even or odd. For checking whether the waves are shown correctly and the code is fine a test bench is conducted by myself and the shapes are shown below. The test bench is also available in sent files under the folder named 1. Arbitrary waveform is done in next part because ROM is not implemented here and it'll be put into place in Quartus. So the results for this waveform are shown in next parts.

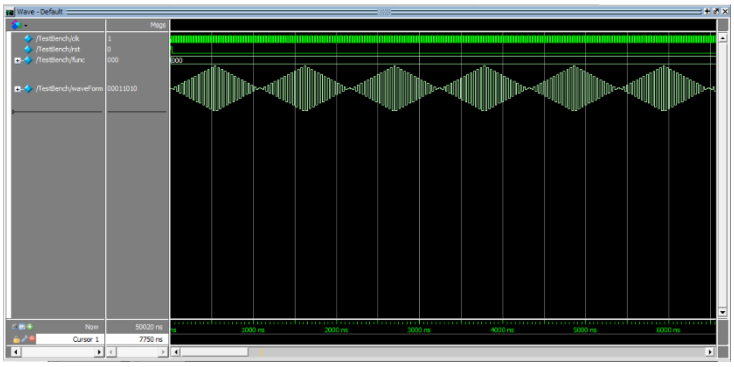

Fig. 1 Rhomboid waveform (code test)

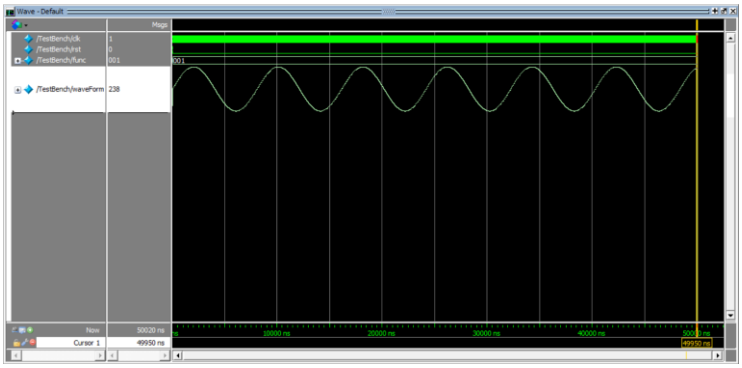

Fig. 2 Sine waveform (code test)

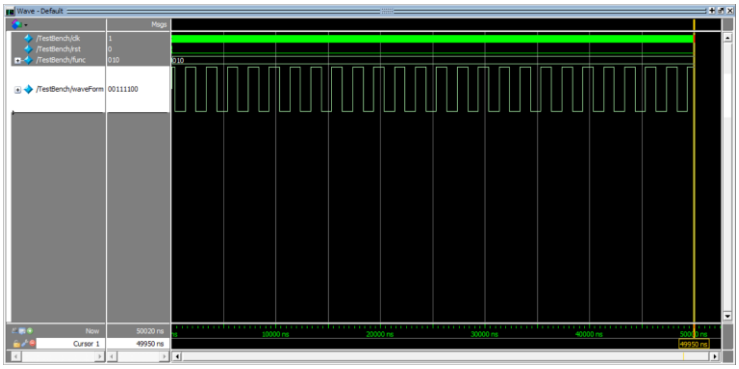

Fig. 3 Square waveform (code test)

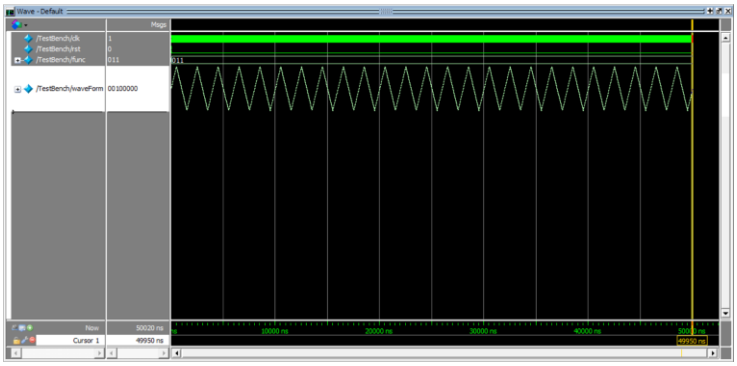

Fig. 4 Triangle waveform (code test)

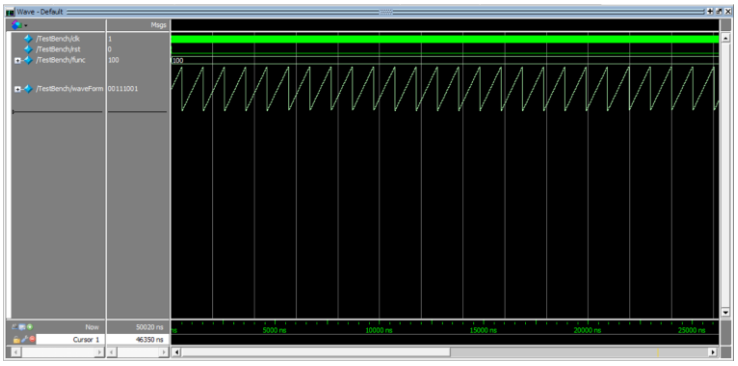

Fig. 5 Saw-tooth waveform (code test)

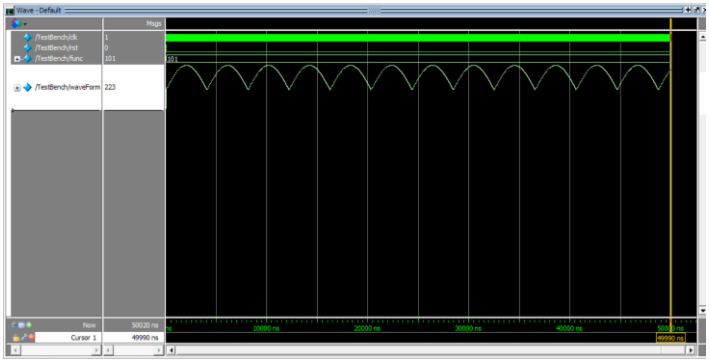

Fig. 6 Full-wave rectified waveform (code test)

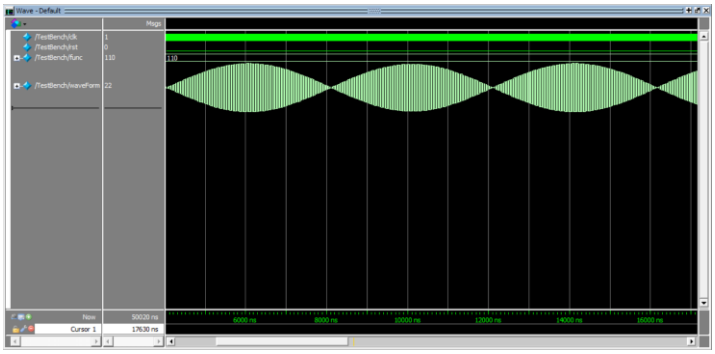

Fig. 7 Modulated sine waveform (code test)

2. This part is done in Quartus. The final circuit with the next two parts' components are implemented in Quartus and all the files are provided in the sent files under the folder named 1 Quartus. Also the final synthesis summary for the whole circuit is shown in figure 8 below.

Flow Summary	
Flow Status	Successful - Mon Aug 24 23:02:33 2020
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	waveFromGenerator
Top-level Entity Name	waveFromGenerator
Family	Cyclone IV E
Device	EP4CE6E22A7
Timing Models	Final
Total logic elements	410 / 6,272 (7 %)
Total combinational functions	409 / 6,272 (7 %)
Dedicated logic registers	126 / 6,272 (2 %)
Total registers	126
Total pins	51 / 92 (55 %)
Total virtual pins	0
Total memory bits	8,192 / 276,480 (3 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)

Fig. 8 synthesis summary (final circuit with all components)

3. This part is done in ModelSim. The circuit is provided in the sent files under the folder named 1 Quartus and the test bench is provided in sent files under the folder named 1 ModelSim.50 MHz clock was used for testing as said in the experiment description and the pictures shown below are using this clock. The results are shown in figures 9 to 16 below. There are some jumps in the waveforms and the reason is the transition between numbers like for example 15 and 16 that a lot of ones will change to zero to get to the result. Figures 1 to 7 are just the results of a test bench that I myself conducted just to make sure my code works but figures 9 to 16 are the waveform created after synthesizing in Quartus and testing in ModelSim using .vo file. So the wanted waveforms mentioned in the experiment description are figures 9 to 16. Also since ROM is implemented in Quartus, arbitrary waveform can be shown here.

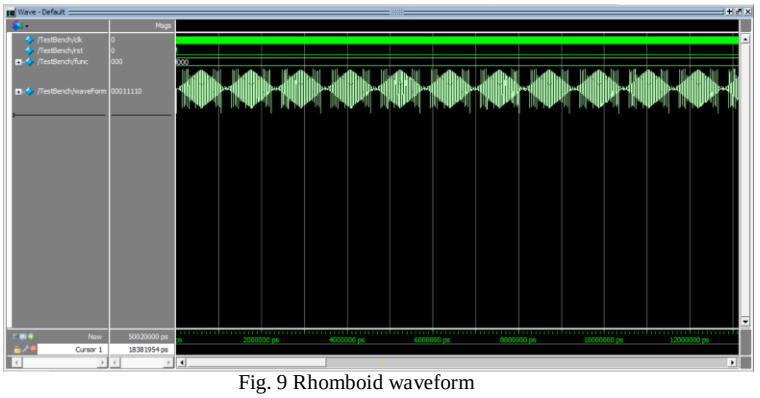

Fig. 9 Rhomboid waveform

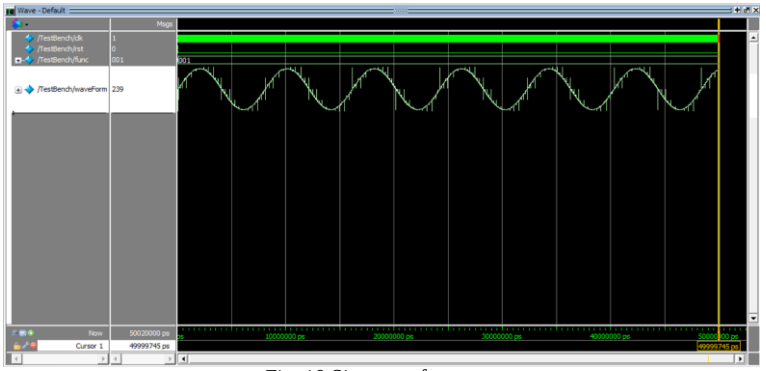

Fig. 10 Sine waveform

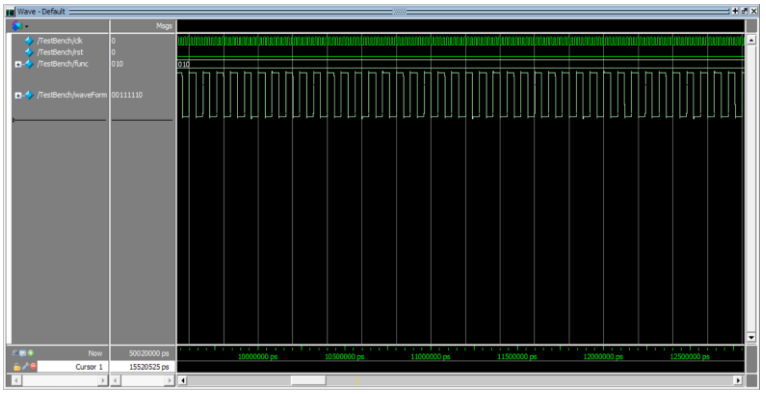

Fig. 11 Square waveform

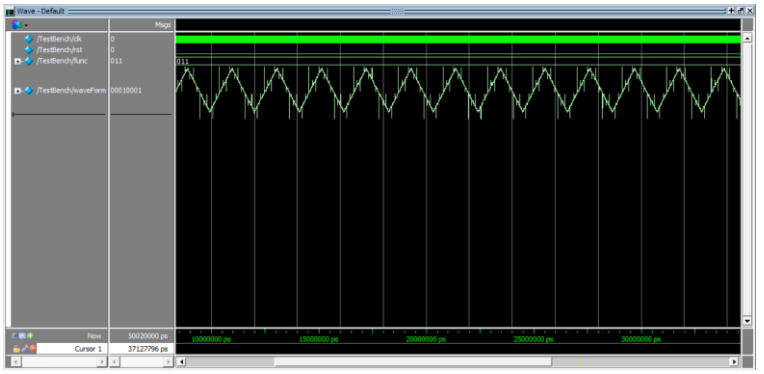

Fig. 12 Triangle waveform

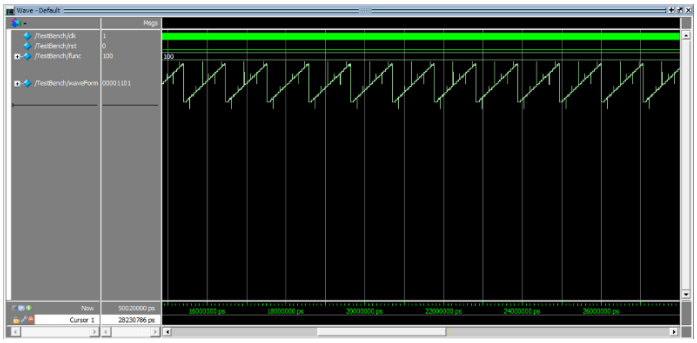

Fig. 13 Saw-tooth waveform

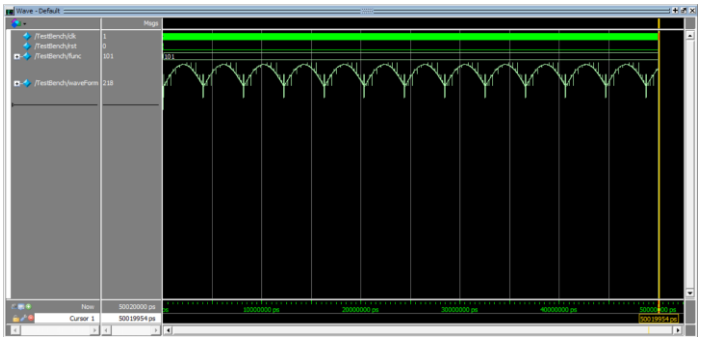

Fig. 14 Full-wave rectified waveform

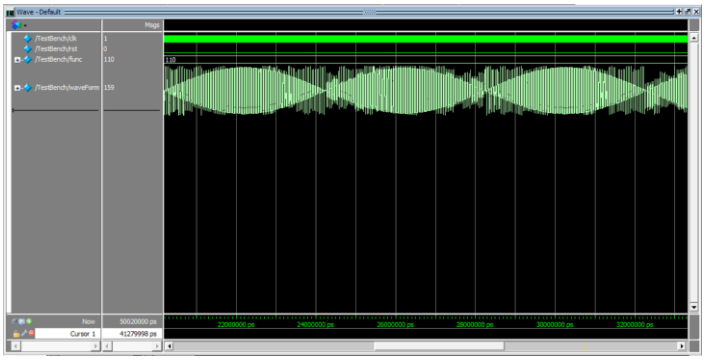

Fig. 15 Modulated sine waveform

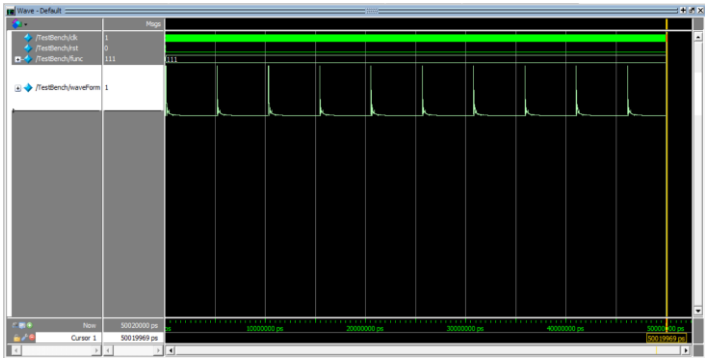

Fig. 16 Arbitrary waveform

III. Frequency Selector

In this part we will use the component created in the previous experiment to provided clock with desired frequency for waveform generator.

1. Done in Quartus. The circuit and all files are provided in sent files under the folder named 1 Quartus.
2. Done in Quartus. Bits 7 to 0 of SW is assigned as setPeriod of the frequency selector.
3. Done in ModelSim. Test bench file is provided in sent files under the folder named 1 ModelSim. Three frequencies are used for testing all the waveforms. FPGA clock has 50 MHz frequency for all tests. The desired frequencies that are used for testing are 400 KHz, 2 MHz, and 200 KHz. According to FPGA clock and based on calculations done setPeriod for these frequencies are 125, 25, and 250 respectively. Also since the clock cycle length for ring oscillator is 50 ns, the final parallel load values for these frequencies are 205, 245, and 155 respectively.
4. All waveforms for different frequencies are shown in figures 17 to 43 below.

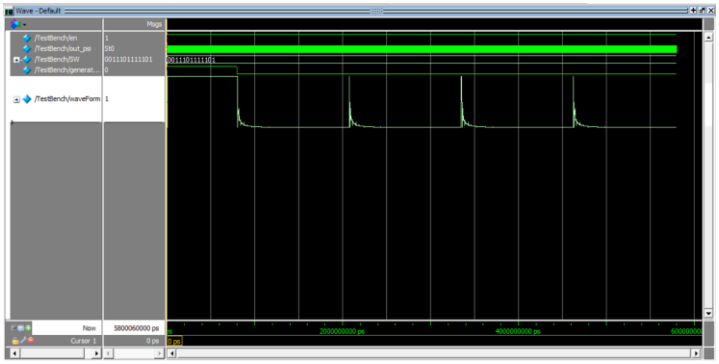

Fig. 17 Arbitrary waveform 400 KHz setPeriod = 125

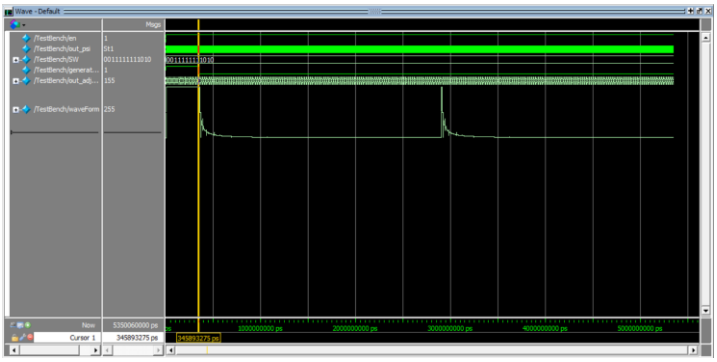

Fig. 18 Arbitrary waveform 200 KHz setPeriod = 250

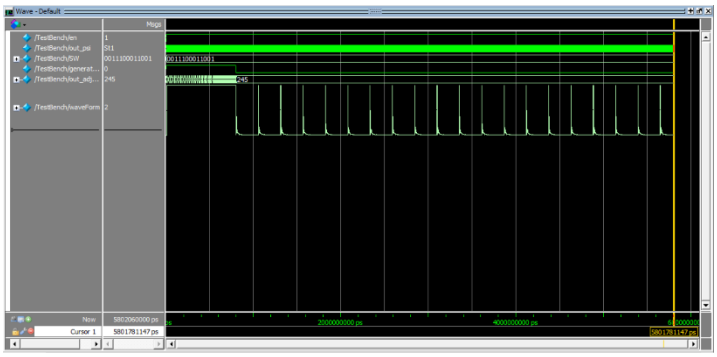

Fig. 19 Arbitrary waveform 2 MHz setPeriod = 25

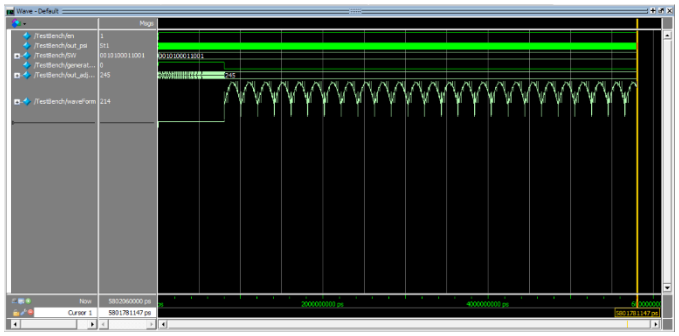

Fig. 20 Full-wave rectified waveform 2 MHz setPeriod = 25

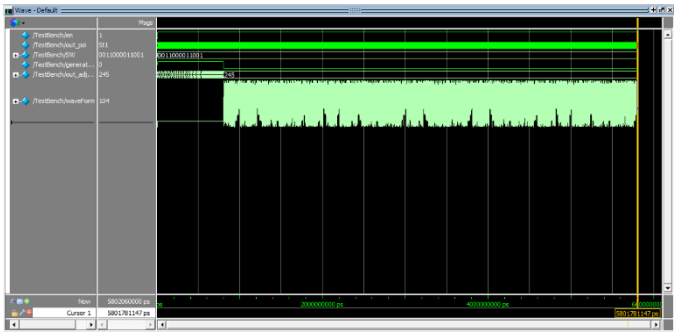

Fig. 24 Modulated sine waveform 2 MHz setPeriod = 25

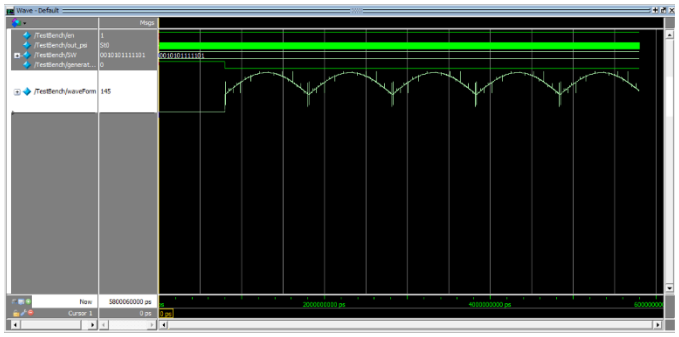

Fig. 21 Full-wave rectified waveform 400 KHz setPeriod = 125

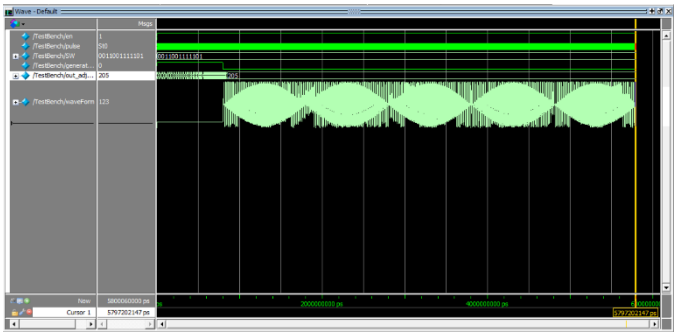

Fig. 25 Modulated sine waveform 400 KHz setPeriod = 125

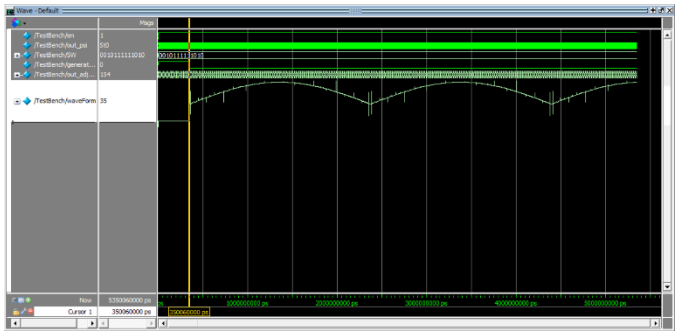

Fig. 22 Full-wave rectified waveform 200 KHz setPeriod = 250

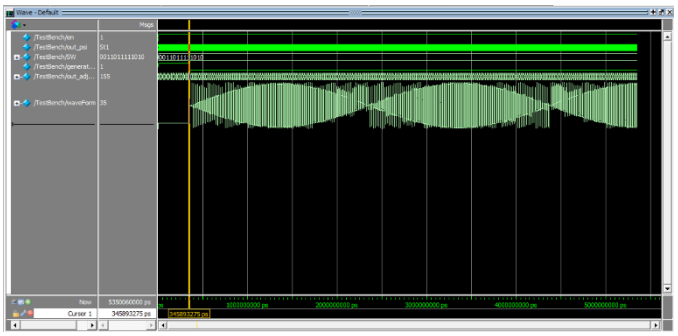

Fig. 26 Modulated sine waveform 200 KHz setPeriod = 250

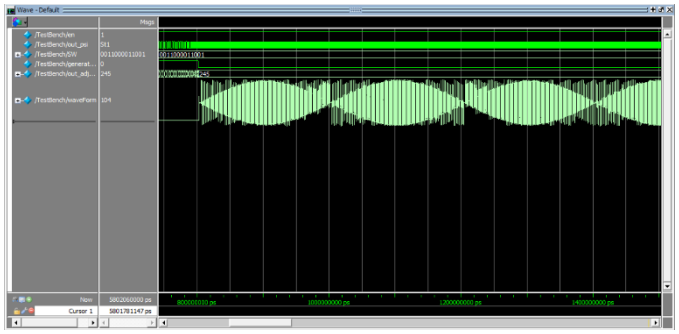

Fig. 23 Modulated sine waveform 2 MHz setPeriod = 25 zoomed in for clarification

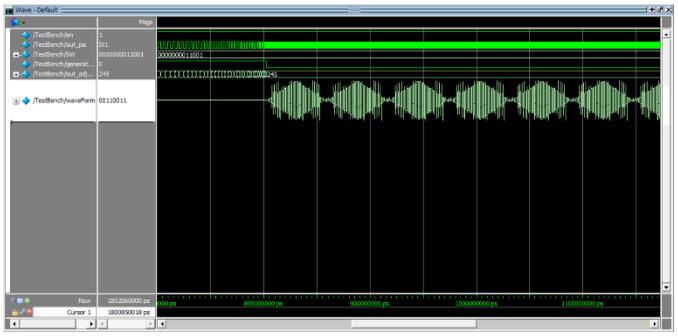

Fig. 27 Rhomboid waveform 2 MHz setPeriod = 25 zoomed in for clarification

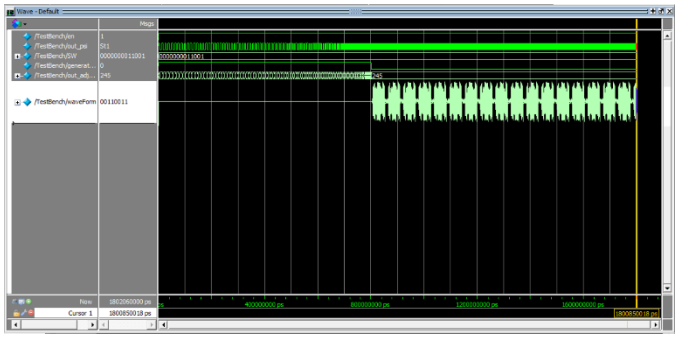

Fig. 28 Rhomboid waveform 2 MHz setPeriod = 25

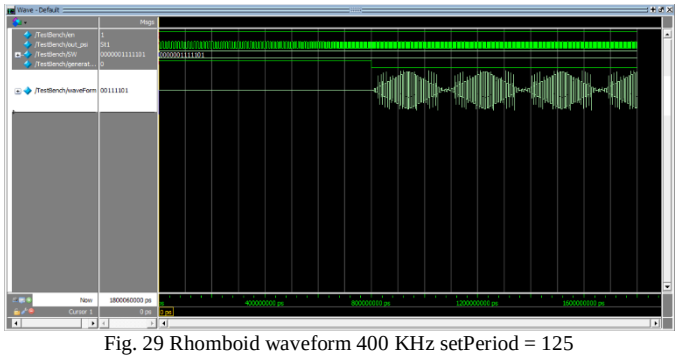

Fig. 29 Rhomboid waveform 400 KHz setPeriod = 125

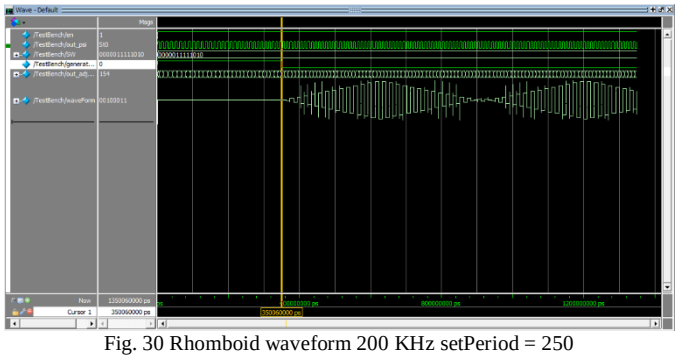

Fig. 30 Rhomboid waveform 200 KHz setPeriod = 250

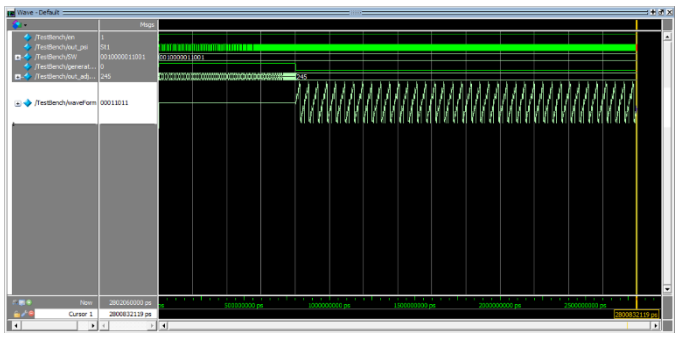

Fig. 31 Saw-tooth waveform 2 MHz setPeriod = 25

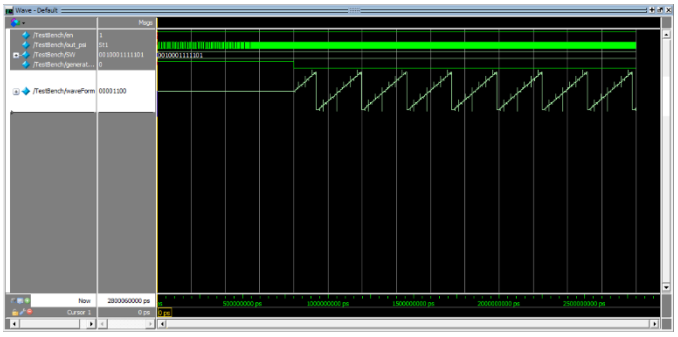

Fig. 32 Saw-tooth waveform 400 KHz setPeriod = 125

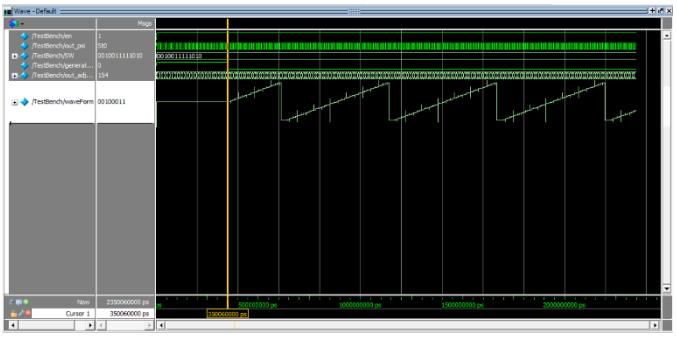

Fig. 33 Saw-tooth waveform 200 KHz setPeriod = 250

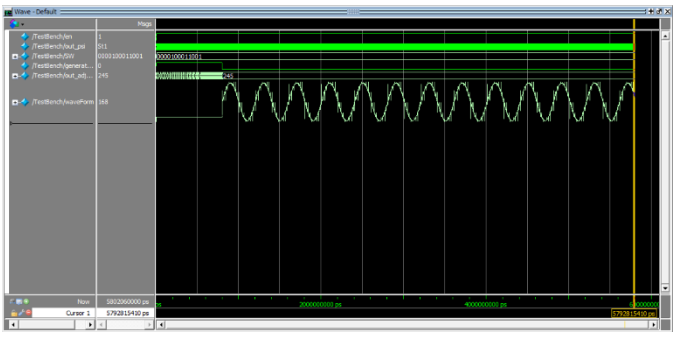

Fig. 34 Sine waveform 2 MHz setPeriod = 25

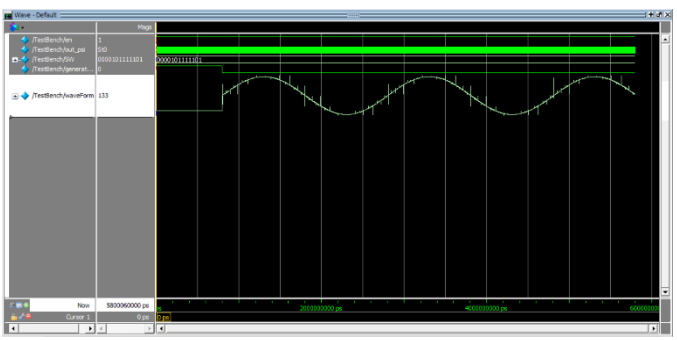

Fig. 35 Sine waveform 400 KHz setPeriod = 125

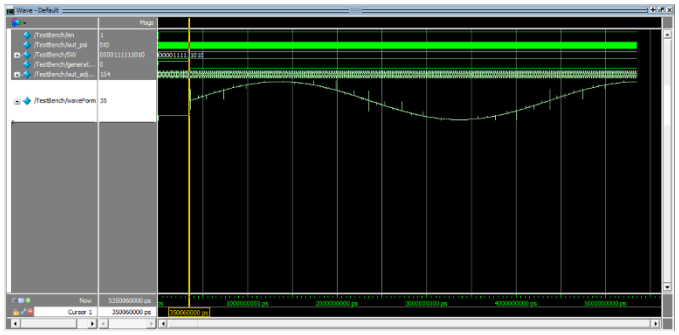

Fig. 36 Sine waveform 200 KHz setPeriod = 250

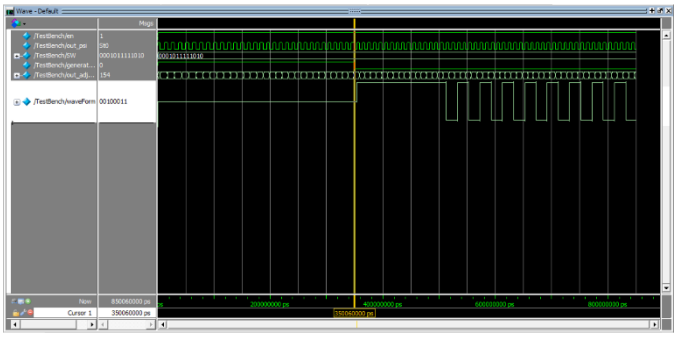

Fig. 40 Square waveform 200 KHz setPeriod = 250

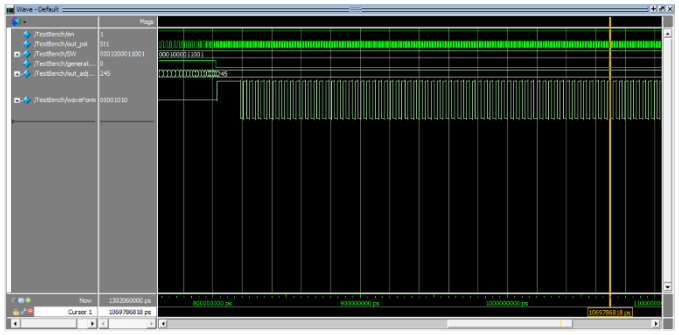

Fig. 37 Square waveform 2 MHz setPeriod = 25 zoomed in for clarification

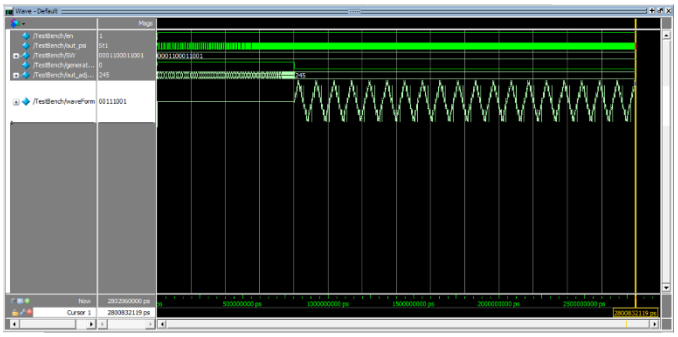

Fig. 41 Triangle waveform 2 MHz setPeriod = 25

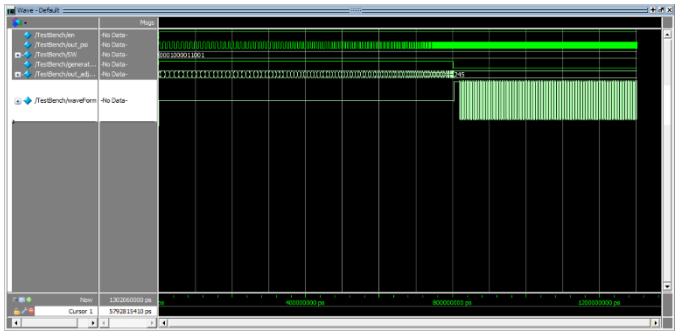

Fig. 38 Square waveform 2 MHz setPeriod = 25

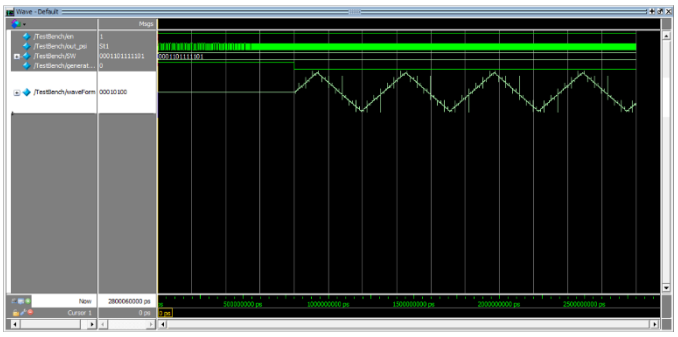

Fig. 42 Triangle waveform 400 KHz setPeriod = 125

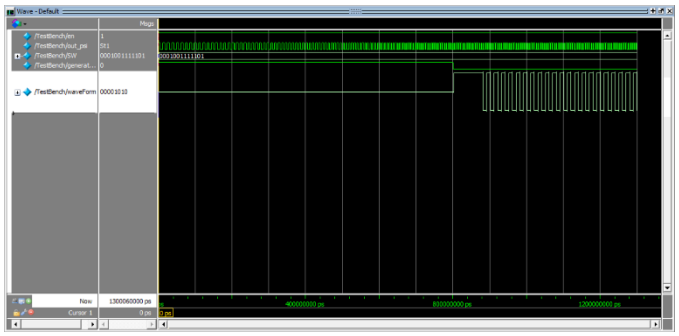

Fig. 39 Square waveform 400 KHz setPeriod = 125

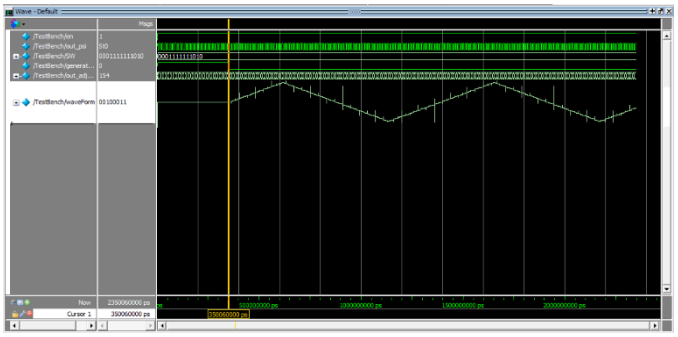

Fig. 43 Triangle waveform 200 KHz setPeriod = 250

IV. Amplitude Selector

In this part we will divide the output of the wave generator by different values such as 1, 2, 4 and 8 so to change the amplitude of the output waveform and achieve desirable range for generated waveform.

1. Done in Quartus. The final circuit and all files are provided in sent files under the folder named 1 Quartus. All of the codes that were written by myself are put in folder 1 such as wave generator for part one, 10 bit counter for ROM address and amplitude selector for this part.
2. Done in ModelSim. Test bench file is provided in sent files under the folder named 1 ModelSim. Divisor values are used from table 2. Dedicated bits 12 and 11 of SW for this process and testing.
3. All waveforms for different divisor values are shown in figures 44 to 75 below. Frequency used for testing is 400 KHz so setPeriod is 125.

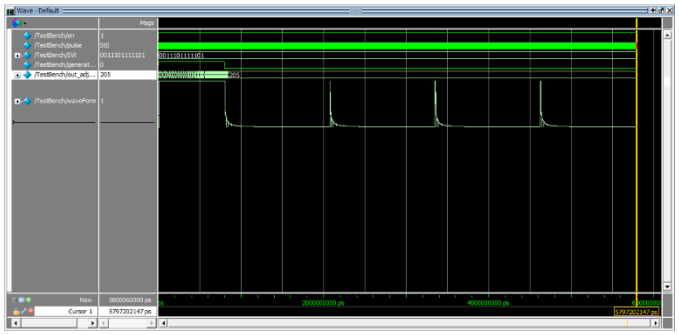

Fig. 44 Arbitrary waveform divisor 1

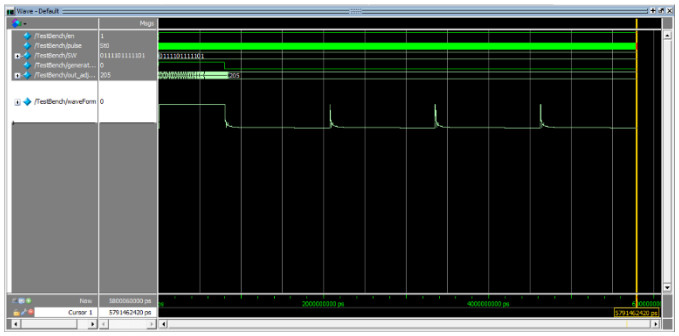

Fig. 45 Arbitrary waveform divisor 2

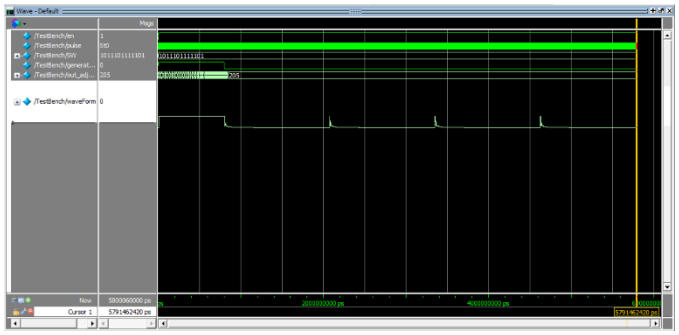

Fig. 46 Arbitrary waveform divisor 4

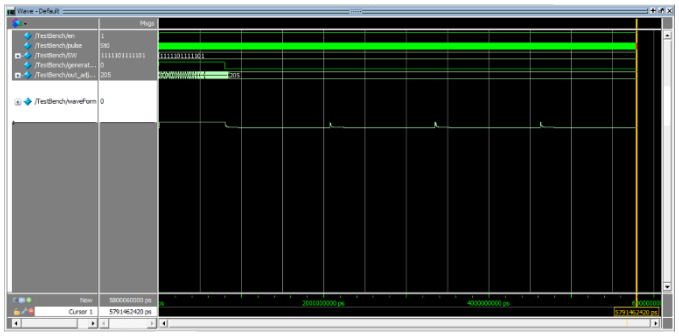

Fig. 47 Arbitrary waveform divisor 8

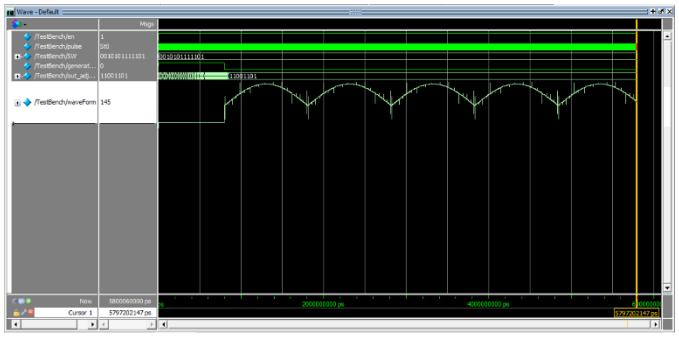

Fig. 48 Full-wave rectified waveform divisor 1

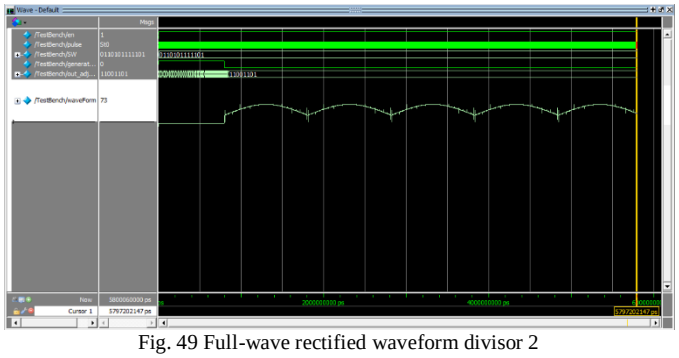

Fig. 49 Full-wave rectified waveform divisor 2

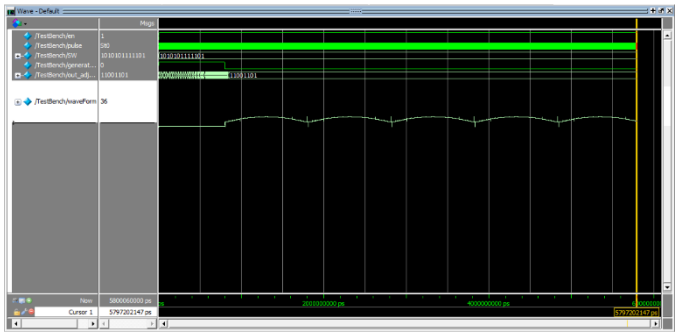

Fig. 50 Full-wave rectified waveform divisor 4

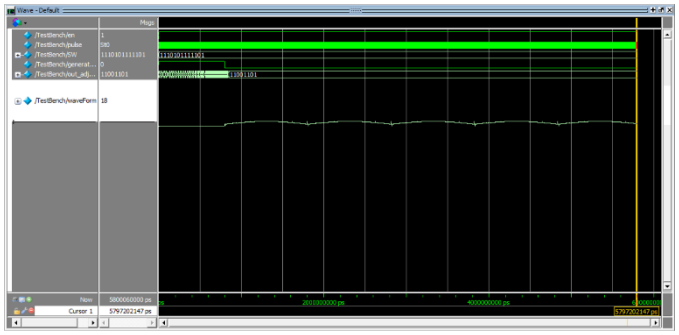

Fig. 51 Full-wave rectified waveform divisor 8

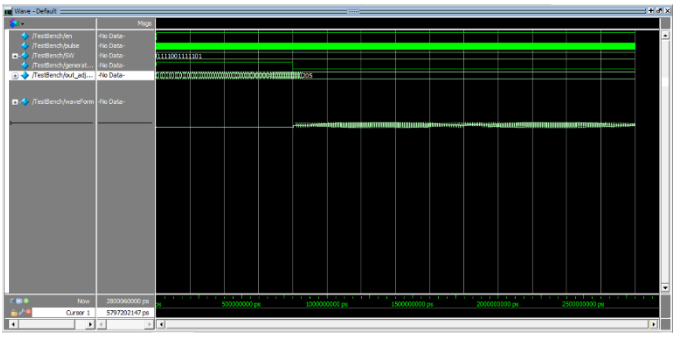

Fig. 55 Modulated sine waveform divisor 8

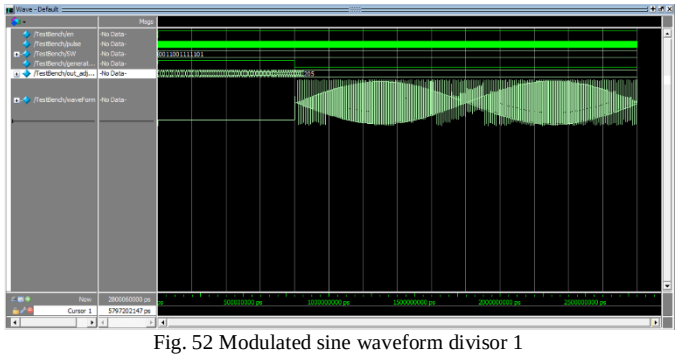

Fig. 52 Modulated sine waveform divisor 1

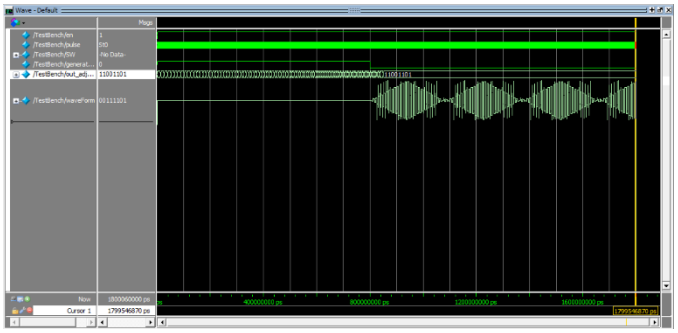

Fig. 56 Rhomboid waveform divisor 1

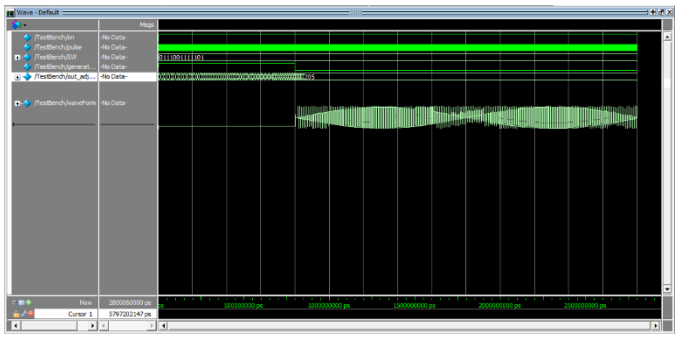

Fig. 53 Modulated sine waveform divisor 2

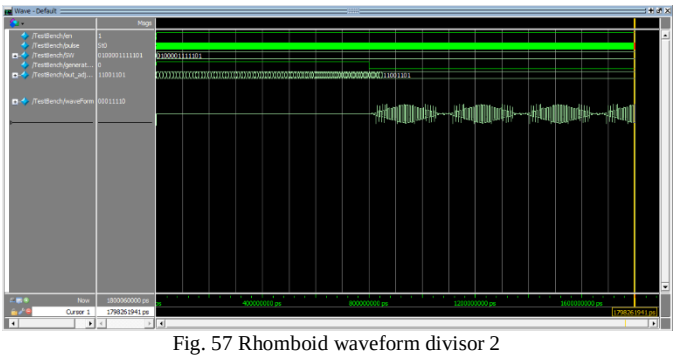

Fig. 57 Rhomboid waveform divisor 2

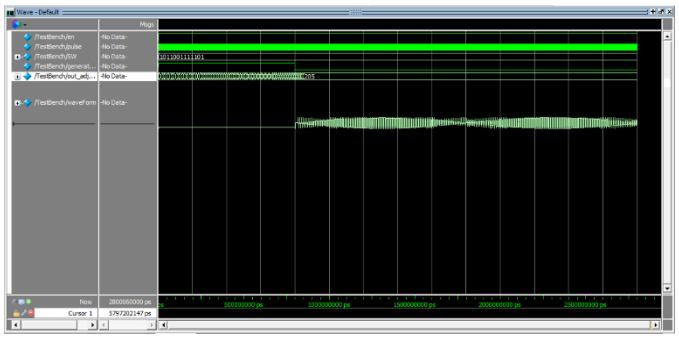

Fig. 54 Modulated sine waveform divisor 4

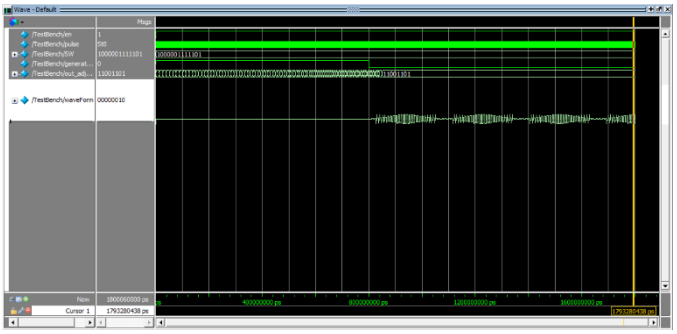

Fig. 58 Rhomboid waveform divisor 4

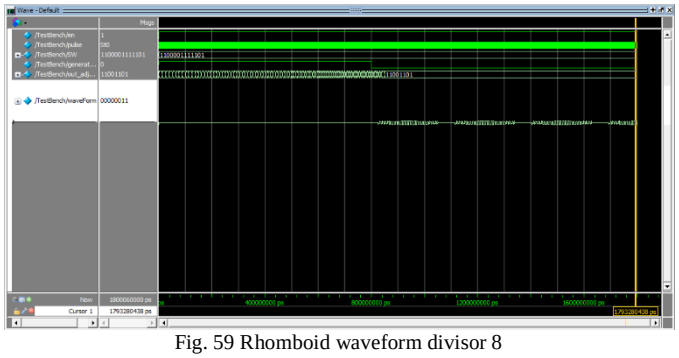

Fig. 59 Rhomboid waveform divisor 8

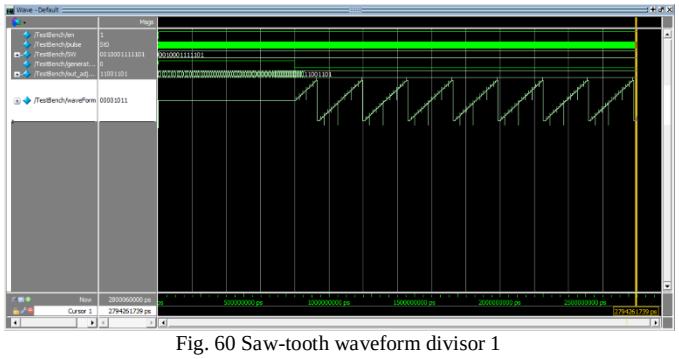

Fig. 60 Saw-tooth waveform divisor 1

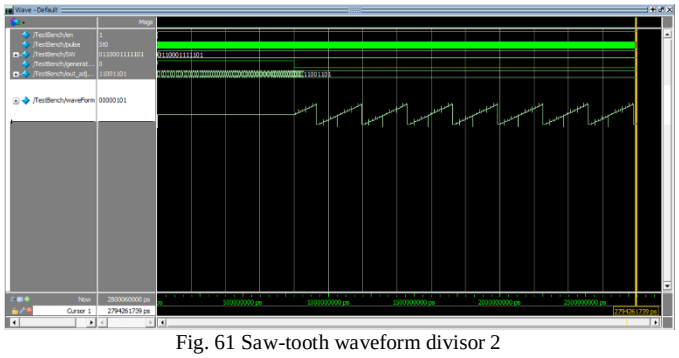

Fig. 61 Saw-tooth waveform divisor 2

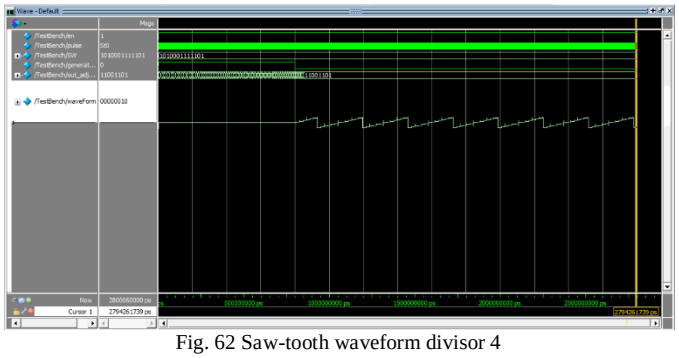

Fig. 62 Saw-tooth waveform divisor 4

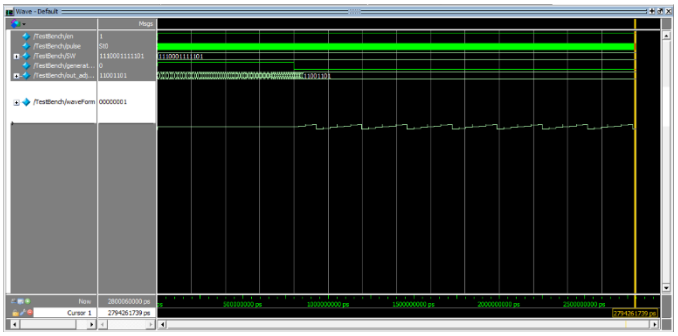

Fig. 63 Saw-tooth waveform divisor 8

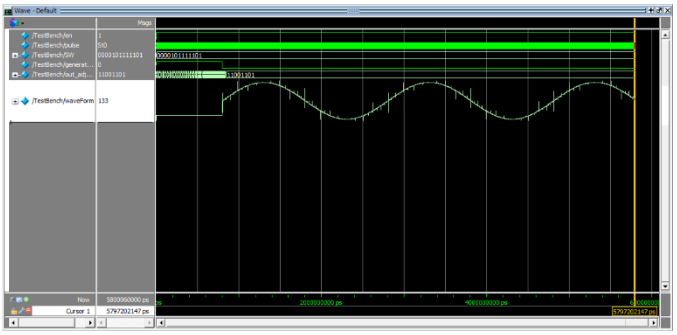

Fig. 64 Sine waveform divisor 1

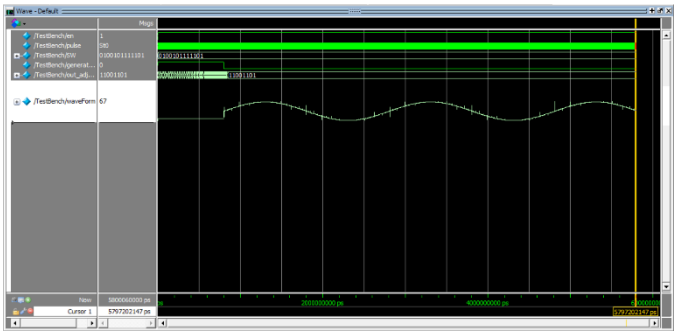

Fig. 65 Sine waveform divisor 2

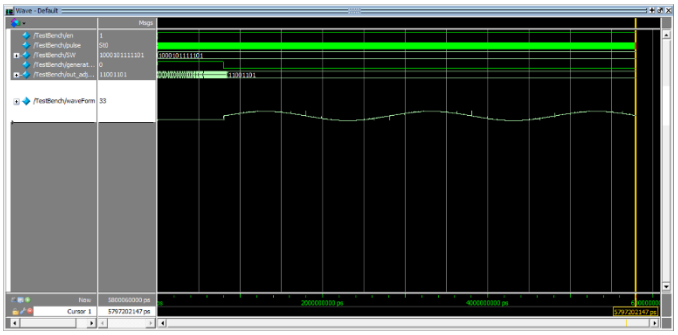

Fig. 66 Sine waveform divisor 4

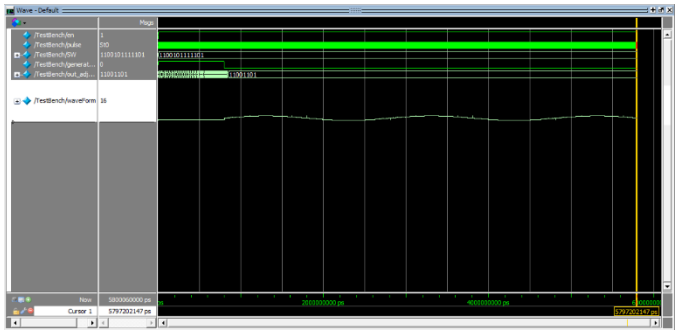

Fig. 67 Sine waveform divisor 8

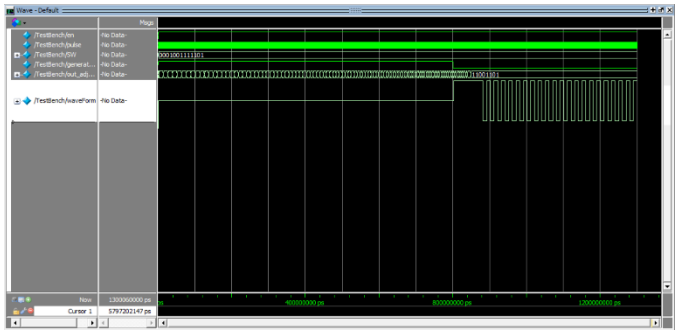

Fig. 68 Square waveform divisor 1

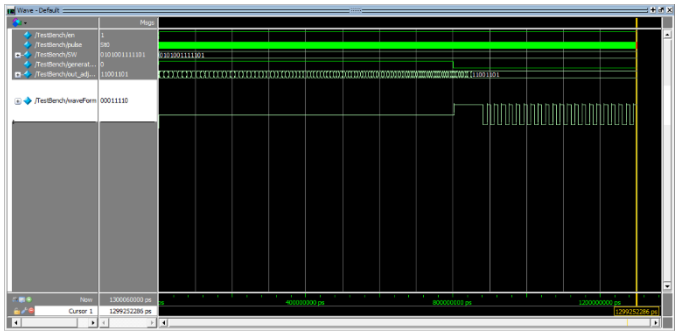

Fig. 69 Square waveform divisor 2

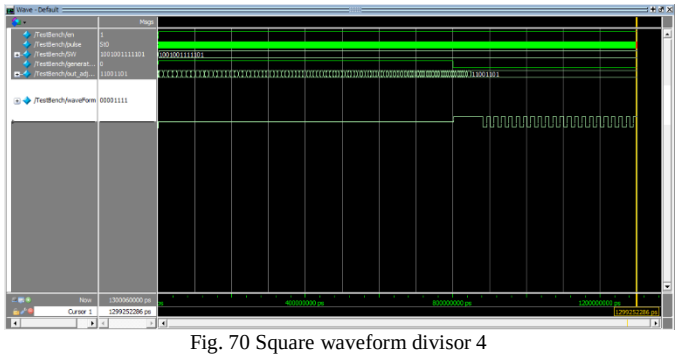

Fig. 70 Square waveform divisor 4

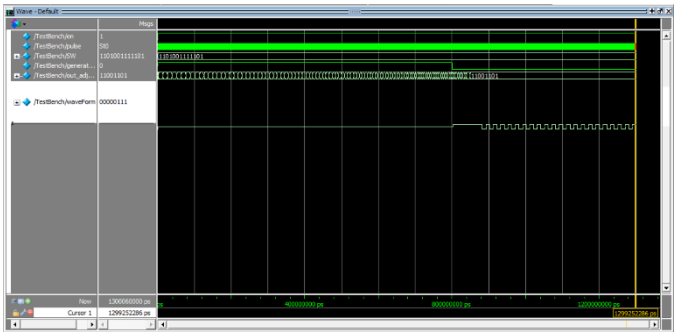

Fig. 71 Square waveform divisor 8

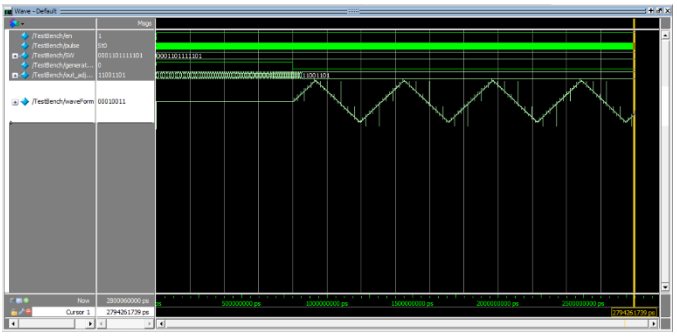

Fig. 72 Triangle waveform divisor 1

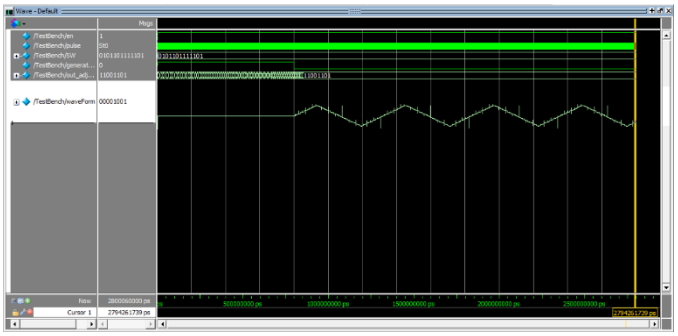

Fig. 73 Triangle waveform divisor 2

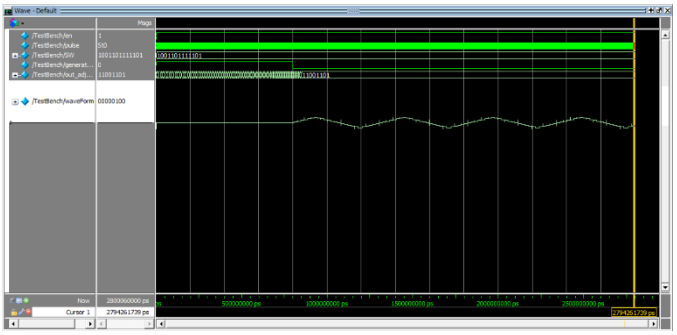

Fig. 74 Triangle waveform divisor 4

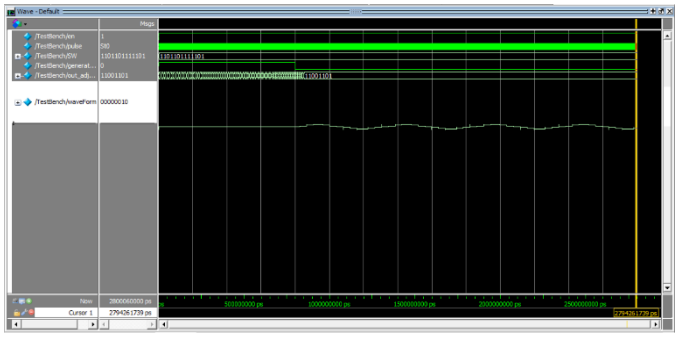

Fig. 75 Triangle waveform divisor 8

At the end in figure 76 a photo of the final circuit is shown. The left part is for frequency regulator brought here from previous experiment for part two, the middle part is wave generator created in part one with its dedicated ROM and finally the right part is for amplitude selector created in part three. In last two parts we will wait for frequency regulator to reach the desired frequency first then we will let waveform generator to start working. This is done by using reset value of waveform generator.

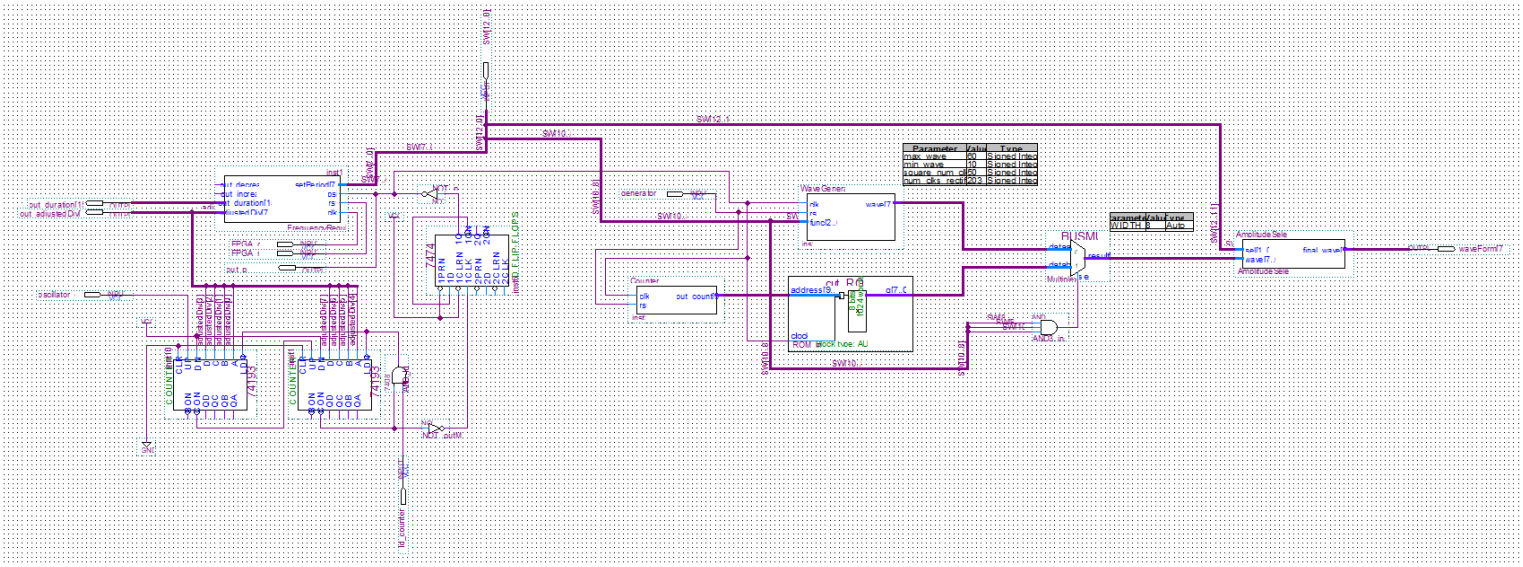

Fig. 77 Final circuit with all the components included