Experiment 3 - Function Generator

Amir Naddaf Fahmideh,
810101540,
Mostafa Kermani nia,
810101575

***Abstract*— This is the report of experiment 3 of logic design lab. The goal of this experiment is to create a function generator witch can generate different wave forms (such as Rhomboid, Square, sine) with different frequency and wave amplitudes.**

***Keywords*— Function Generator, Wave Generator, Frequency Selector, Oscilloscope, Amplitude Selector, PWM**

I. Introduction

An Arbitrary Function Generator (AFG) is a device used in electronic testing to create different types of waveforms with varying amplitudes and frequencies. These waveforms include sine, square, rhomboid, and saw-tooth shapes, as well as custom shapes. AFGs are useful for generating simple test signals, replicating real-world signals, and creating unique signals that are not otherwise available. These signals help in understanding how a circuit functions, examining an electronic component, and testing electronic theories. Figure 1 shows an actual AFG from Instek Company.

Fig. 1 A single channel AFG from Instek company

In this experiment, our goal is to design an AFG capable of producing the mentioned waveforms across a broad frequency range. Figure 2 illustrates a basic diagram of such a generator.

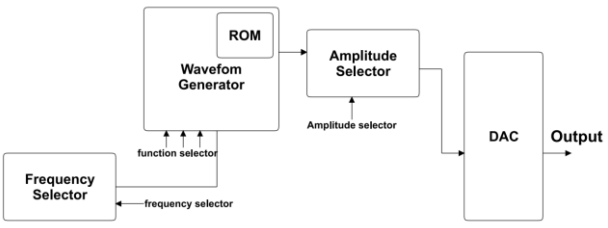

Fig. 2 Block diagram of the Arbitrary Function Generator (AFG)

The design includes:

- A main unit that creates the selected waveform based on function selection.
- A frequency control that adjusts the signal's frequency.
- An amplitude control that sets the signal's strength.
- A DAC (Digital to Analog Converter) that transforms the digital signal into an analog one, which can be viewed and analyzed using an oscilloscope.
- A ROM (Read-Only Memory) that stores custom waveforms for the generator.

The experiment will cover several key areas:

- How the waveform creation unit works.
- How to use ROM for storing waveforms.
- How to design the system using Quartus II software.

II. Waveform Generator

The wave form generator is responsible to make 6 different waves as you can see in figs 3 and 4. It contains two parts, first part makes first three waves by simple mathematical operations. The other part is responsible to make sine waveform. It was hard to make sine waveform with mathematical operations so in this part we used a ROM which has the value of sine wave in different times. After that we used the sine waveform to generate full-wave rectified and half-wave rectified waves. At the end we used a multiplexer to select one of these waveforms as the output of waveform generator.

1. Simulation outcomes

The simulation outcomes for all waveforms are shown in fig 3 and fig4.

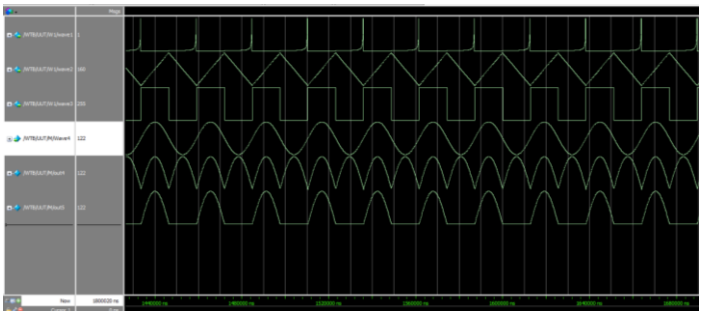

Fig. 3 Simulation outcomes for all waveforms with simulation details

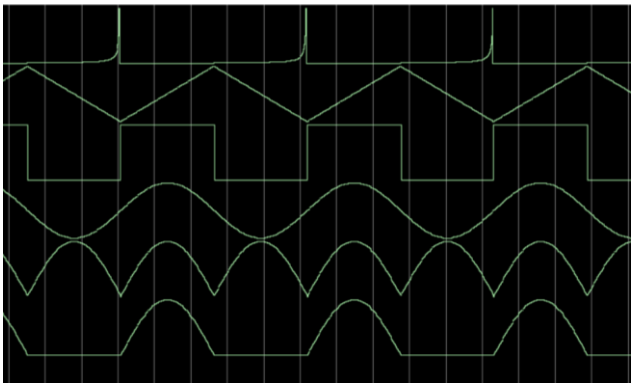

Fig. 4 Simulation outcomes for all waveforms without simulation details

2. *FPGA resource utilization*

As you can see the result with romstyle keyword leads us to have memory elements (fig5), but when we don't use romstyle keyword, quartus doesn't use memory elements, instead it uses more registers to synthesis ROM (fig6).

Flow Summary

Flow Status	Successful - Tue Apr 09 07:25:15 2024
Quartus II 64-Bit Version	12.1 Build 177 11/07/2012 SJ Web Edition
Revision Name	FunctionGenerator
Top-level Entity Name	FunctionGenerator
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	213 / 18,752 (1 %)
Total combinational functions	213 / 18,752 (1 %)
Dedicated logic registers	29 / 18,752 (< 1 %)
Total registers	29
Total pins	14 / 315 (4 %)
Total virtual pins	0
Total memory bits	448 / 239,616 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Fig. 5 FPGA resource utilization with romstyle

Flow Status	Successful - Tue Apr 09 07:29:53 2024
Quartus II 64-Bit Version	12.1 Build 177 11/07/2012 SJ Web Edition
Revision Name	FunctionGenerator
Top-level Entity Name	FunctionGenerator
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	237 / 18,752 (1 %)
Total combinational functions	237 / 18,752 (1 %)
Dedicated logic registers	35 / 18,752 (< 1 %)
Total registers	35
Total pins	14 / 315 (4 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Fig. 6 FPGA resource utilization without romstyle

3. *Synthesis report*

The synthesis report for the Waveform Generator's design is shown below.

Analysis & Synthesis Status	Successful - Mon May 13 21:24:34 2024
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	waveform_generator
Top-level Entity Name	waveform_generator
Family	Cyclone V
Logic utilization (in ALMs)	N/A
Total registers	17
Total pins	13
Total virtual pins	0
Total block memory bits	448
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Fig. 7 Synthesis report for the Waveform Generator

III. PWM (PULSE WIDTH MODULATION)

1. *Explanation*

PWM actually is used for sampling from the wave form and it uses a faster clock to gets more samples to be more accurate. In this experiment it is used for digital to analog conversion. A PWM signal is a sequence of periods in which the duration of the logic-high (or logic-low) voltage varies according to external conditions, and these variations can be used to transmit information. The PWM carrier frequency is constant, so the active and inactive state duration increases or decreases and vice versa.

PWM converts a 8-bit digital number to a 1-bit digital number. From its input it receives a number. If this input

number was greater than “N” the output becomes 1 otherwise it becomes 0 (fig 8)

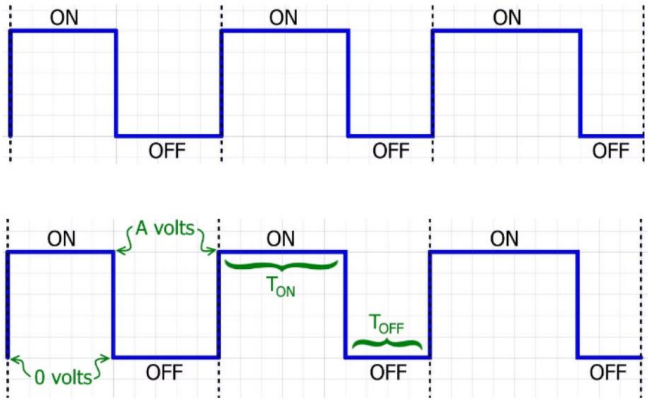

Fig. 8 Pulse Width Modulation (PWM)

2. Simulation results

The duty cycle of the PWM signal is equal to:
duty cycle = T_{on} / (T_{on} + T_{off}) (fig 8)
So the duty cycle for the first input is about 19 percent (fig 9), for the second one is about 75.39 (fig 10) and for last one is about 48 percent (fig 11).

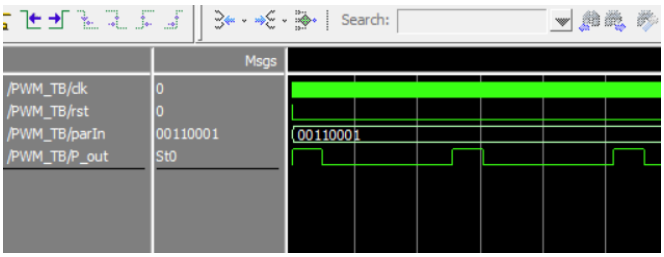

Fig. 9 PWM simulation with first input

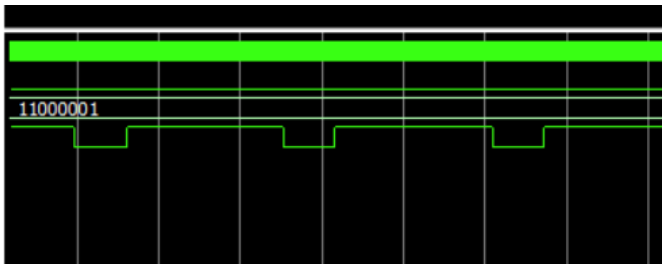

Fig. 10 PWM simulation with second input

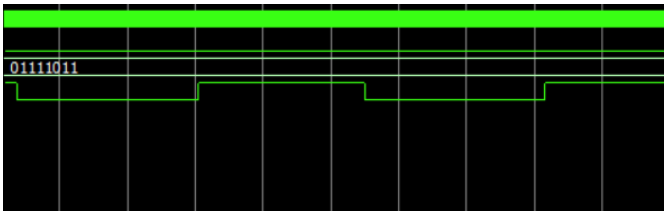

Fig. 11 PWM simulation with third input

IV. Frequency Selector

For frequency selector we used a 9-bit counter that its carry out is the clock of the rest of parts except PWM. So it controls the frequency of our generated waveform. This counter gets 5 MSB bits as its input and 4 LSB bits are fixed. This 9-bit are used as the load of the counter so the bigger the number is the faster the carry out becomes one.

1. Simulation results

There are the simulation results for three selected frequencies. In this simulation you can see the output of frequency selector. The impact of changing the frequency on waveform is shown in VI.2 part.

Fig. 12 Simulation results

V. Amplitude Selector

Amplitude Selector is an multiplexer which has a waveform as input and uses these waveforms and make waves with different Amplitudes and selects one of them with its multiplexer. The different amplitudes are generated by

multiply the input waveform amplitude by 4 different numbers which are 1,3/2, 3/4, 3/8.

1. *Accuracy*

For this part we used default amplitude and different opcodes for amplitude selector to see if it works as it should or not. We test it on all waveforms and the result is shown in following images:

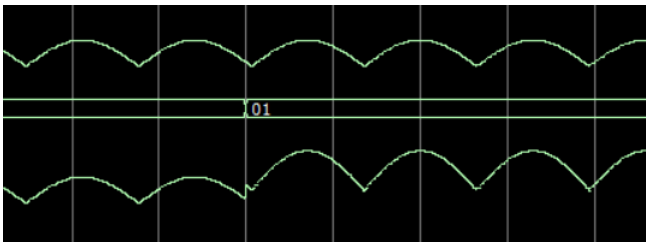

Fig. 13 Different amplitudes for full-wave rectified

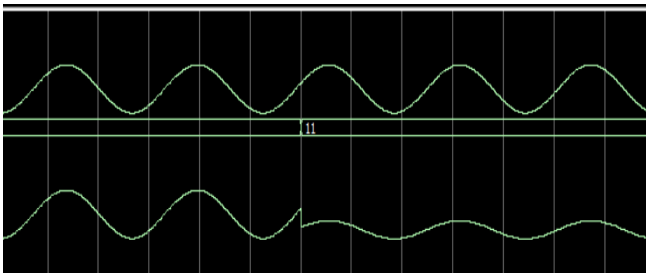

Fig. 14 Different amplitudes for sine wave

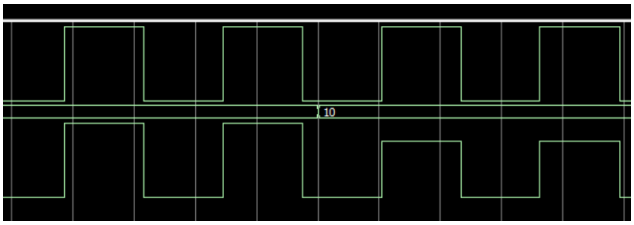

Fig. 15 Different amplitudes for square wave

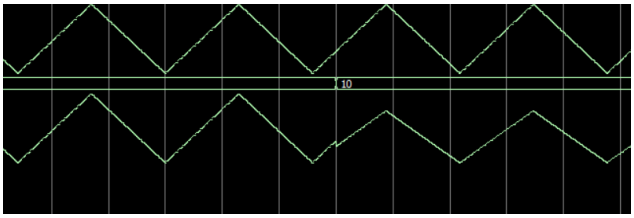

Fig. 16 Different amplitudes for triangle wave

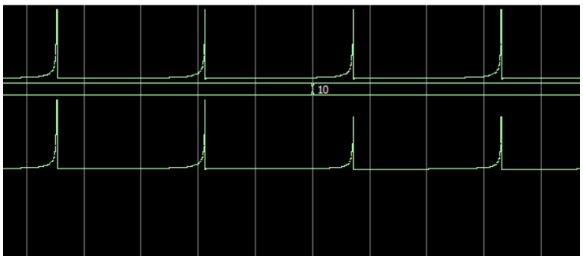

Fig. 17 Different amplitudes for reciprocal wave

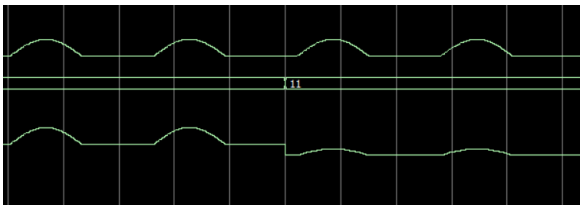

Fig. 18 Different amplitudes for half-wave rectified

VI. IMPLEMENTATION

1. *Schematic diagram*

The schematic diagram of the Function Generator is shown in fig 19

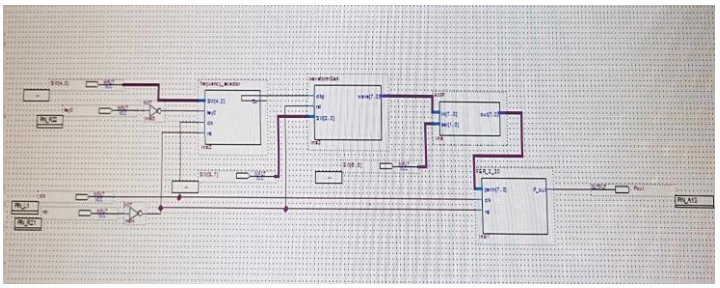

Fig. 19 schematic diagram of the Function Generator

2. *Oscilloscope visualization*

We used triangle wave to show the wave with different frequencies and amplitudes as you can see in Fig 20, 21 and 22.

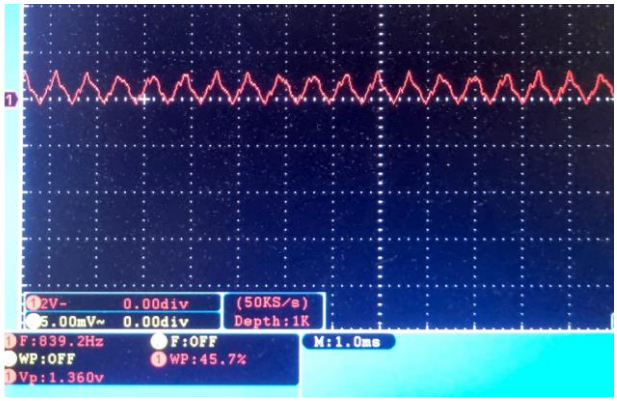

Fig. 20 triangle wave with low amplitude and low frequency

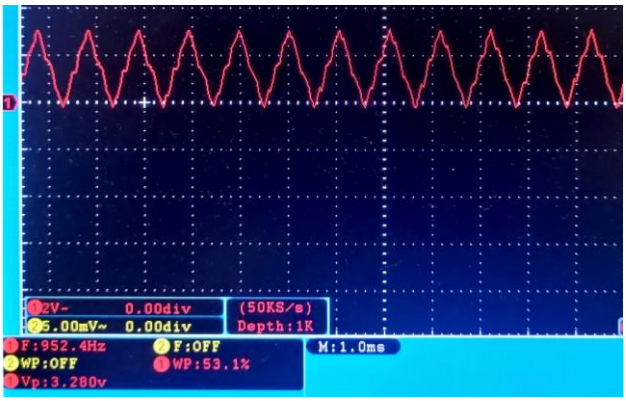

Fig. 21 triangle wave with high amplitude and low frequency

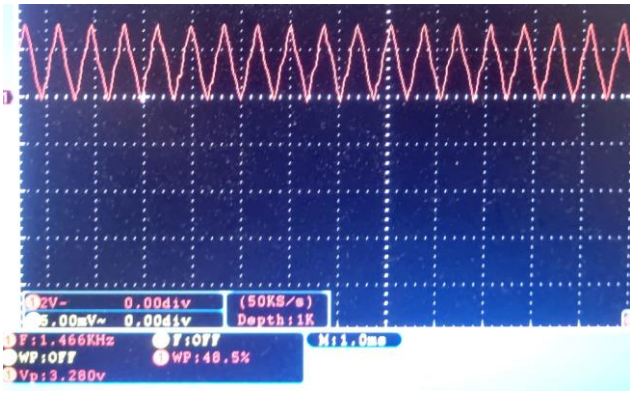

Fig. 22 triangle wave with higher amplitude and higher frequency

VII. TESTS

We tested other waveforms in oscilloscope. You can see the result in following figs.

Fig. 23 Reciprocal wave

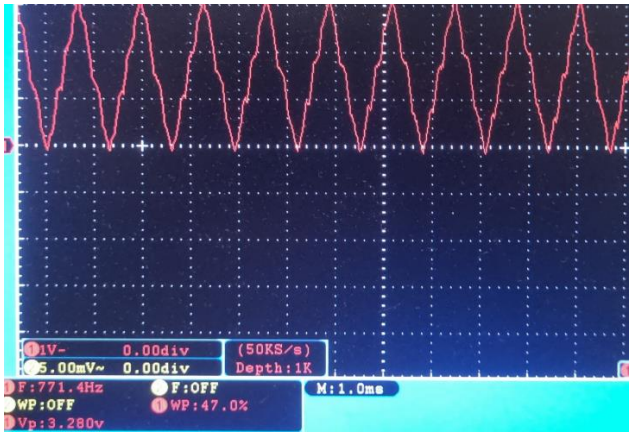

Fig. 24 Triangle wave

Fig. 25 Square wave

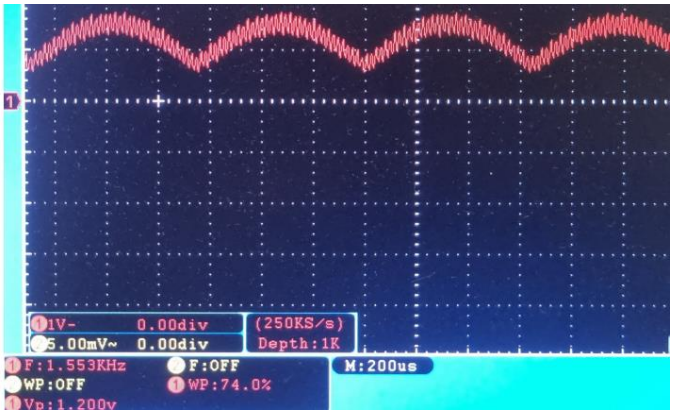

Fig. 26 Full-wave rectified

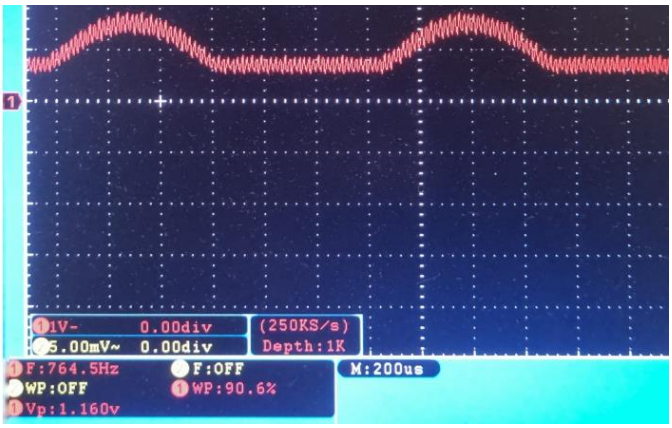

Fig. 27 Half-wave rectified

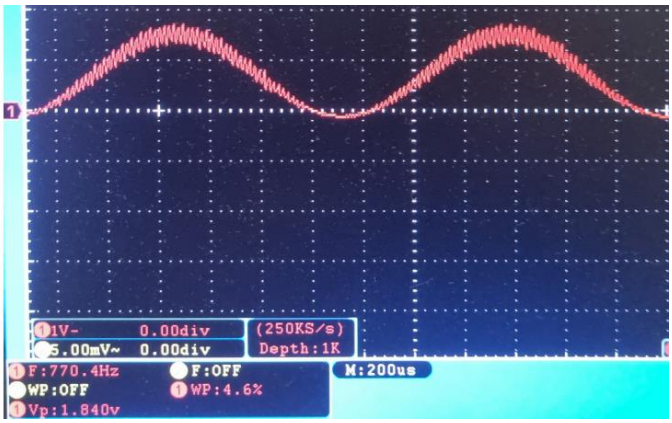

Fig. 28 Sine wave

VIII. CONCLUSIONS

In this experiment we learned how to make a function generator using simple arithmetic operations and for more complicated waves we used ROM. We also learned the functionality of PWM and used it to visualize our waves on oscilloscope. We used quartus to program the FPGA with our codes and connected the FPGA output to a circuit to make our waveform visible by oscilloscope.

REFERENCES

[1] K. Basharkhah “Experiment 3 Digital Logic Laboratory,” under supervision of Professor Z. Navabi, University of Tehran, Spring 1403