Experiment 4 – Accelerator and Wrappers

Shirin Jamshidi - 810199570, Mahya Shahshahani - 810199598

***Abstract*— In this experiment, we are going to design an Exponential Accelerator Wrapper. For this design, we are going to plan the exponential engine, Accelerator wrapper and its controller, then perform our full design on Quartus environment that helps us implement this Accelerator on FPGA, and finally watch the results on FPGA board.**

***Keywords*— SoC, CP, Accelerator, Exponential Engine, handshaking in an SoC, Exponential Accelerator Wrapper, CPU, ROM, FPGA**

I. Introduction

System on Chip is an integrated circuit that integrates multiple components including digital, analog, hardware ,and software programs all in a single chip. The main core of an SoC is a processor that handles different computational tasks within the system. In addition to the processor, the system includes memory, Input/Output ports ,and accelerators. accelerators are dedicated computation units that usually execute one specific task. This single task needs a smaller and less complicated datapath which leads to a high frequency of operation for the accelerators. This is contrary to CPUs in which millions of operations must be executed within a fixed time interval. This imposes a low frequency of operation for CPUs.

II. Experiment

A. Exponential Engine

This module receives a 16-bit input ”x” and generates a 16-bit output ”Fractional part” and 2-bit ”Integerpart”. The accelerator starts working with a complete pulse on the signal ”start” and when the computation is completed signal ”done” will be sent to the processor to acknowledge it.

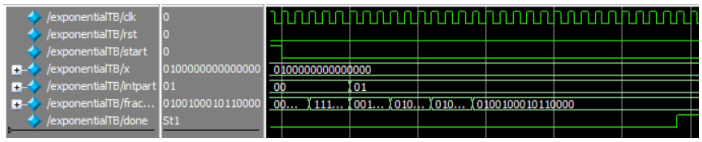

Figure 1: Simulation for 0.25 input as X.

e^(0.25)≈ 1.284025

01.0100100010110000≈1.283935546875

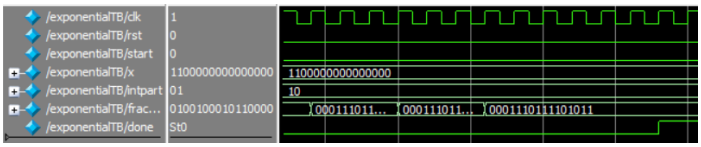

Figure 2: Simulation for 0.75 input as X.

e^(0.75)≈ 2.117000016

10.0001110111101011≈2.1168670654296875

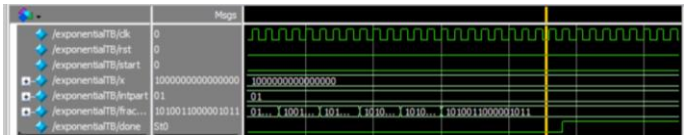

Figure 3: Simulation for 0.5 input as X.

e^(0.5)≈ 1.648721

01.1010011000001011≈1.6486053466796875

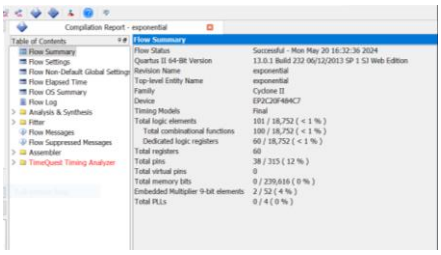

Figure 4: FPGA resources for exponential engine.

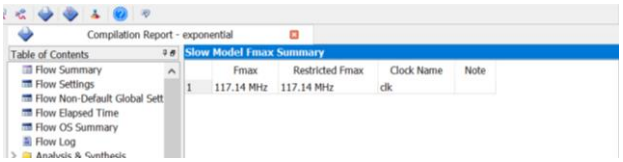

Figure 5: Max frequency for exponential engine.

B. Exponential Accelerator Wrapper Simulation

Since the accelerator data will be accessed before and after completing CPU task, the data has to be stored in memory elements in the accelerator wrapper when CPU is busy with other works. The memory element required in this experiment is an input ROM for storing the input data.

The controller in this wrapper is responsible for generating the ”start” signal for exponential engine and the address of each input data reading from the input ROM. The exponential engine should start each calculation when the previous one is completely done. For this purpose ”engdone” is fed to the controller and when done is asserted the controller generates a complete pulse on ”engstart”. At the same time, the correct value of x should appear on the corresponding input of exponential engine. To do this the controller issues the ”inccount” signal for reading data from the ROM. When all calculations are finished the controller sends a done signal on the wrapper output and issues the ”rstcount” signal to reset the counter for the next round of estimations.

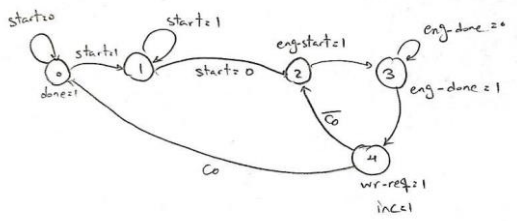

Figure 6: State Machine of Exponential Accelerator

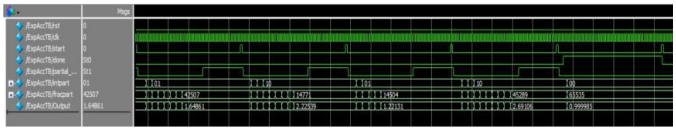

Figure 7: Output simulation for multiple numbers.

The simulation result is for the given inputs, X: 0.5 , 0.79 , 0.19 , 0.99 , 0.

The system works properly.

e^0.5 ≈ 1.648, e^0.79= ≈2.203, e^0.19 ≈ 1.209

e^0.99 ≈ 2.691, e^0 ≈1

C. Implemented Accelerator on FPGA

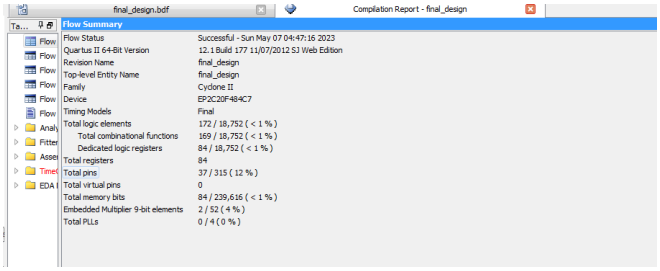

Figure 8: FPGA resources for Exponential Accelerator Wrapper.

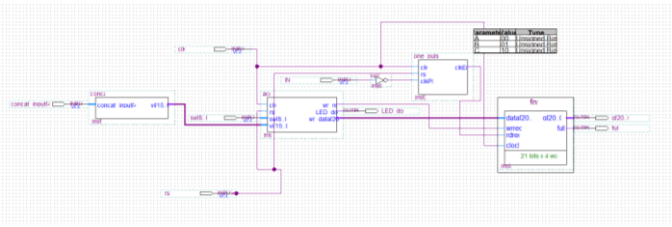

Figure 9: Schematic of the design in Quartus.

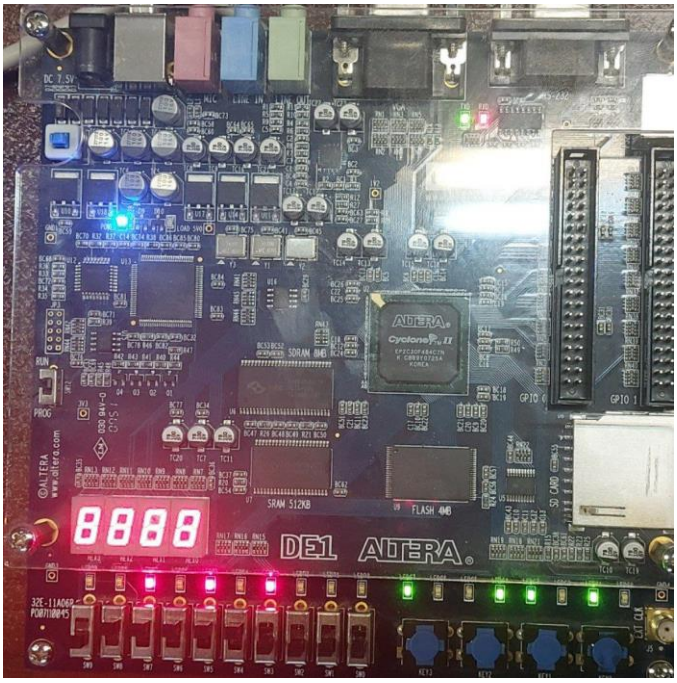

Figure 10: Value of e^1 on Board

0010.101000100110 = 2.63

e^0.967 = 2.63

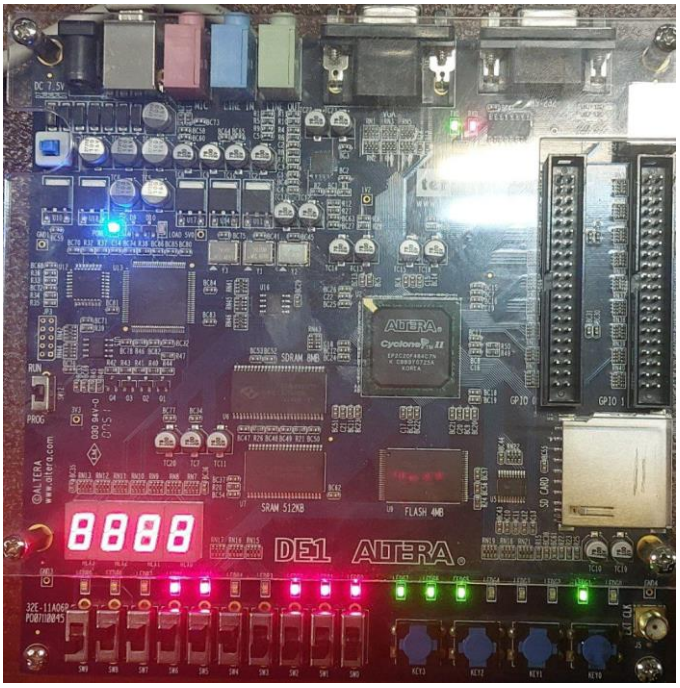

Figure 11: Value of e^0.5 on Board

0001.100111111000 = 1.623

e^0.48 = 1.623

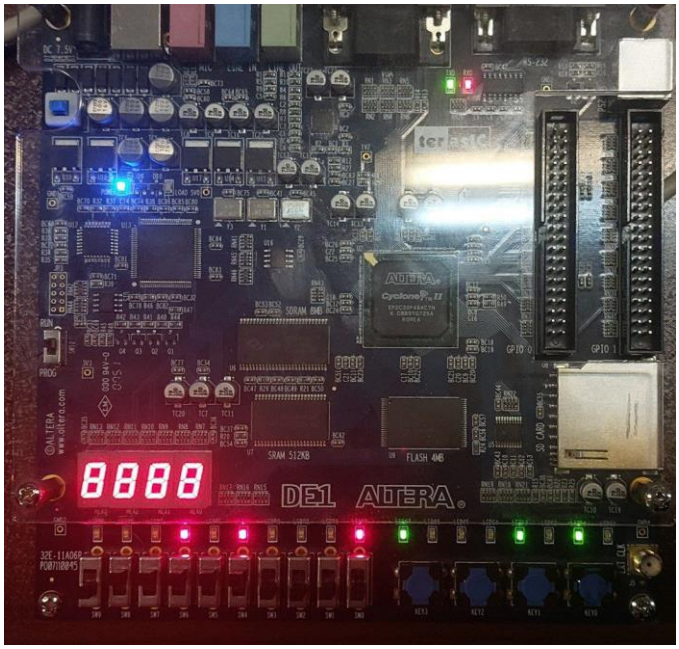

Figure 12: Value of e^0.25 on Board

0001.010001100010 = 1.274
e^0.24 = 1.274

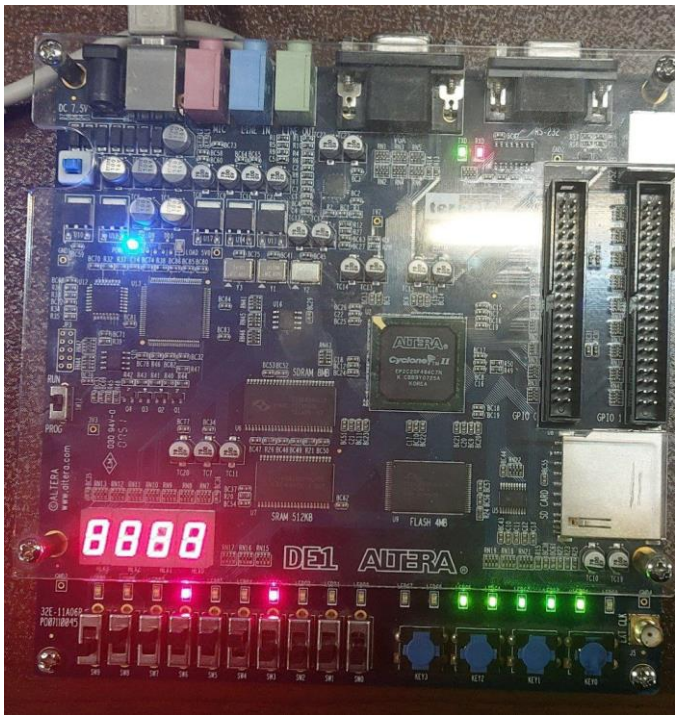

Figure 13: Value of e^0.125 on Board

0001.001000001111 = 1.1286
e^0.121 = 1.1286

The main difference is that part 4 applies an exponential acceleration to the input value by using a larger value for ui, while part 3 does not apply any acceleration.

D. Conclusion

In this experiment, we designed an Exponential accelerator wrapper by designing its controller and implementing our circuit in Quartus.

And at last, we tested our design by implementing it on the FPGA board.

E. References

[1] Katayoon Basharkhah and Zahra Jahanpeima and Prof. Zain Navabi, *Digital Logic Laboratory*, University of Tehran, Spring 1403.