Experiment #**4**

Accelerator and Wrappers

Kamyar Rahmani

810199422

Mohammad Mashreghi

810199492

***Abstract*— In this document, we are going to design an Exponential Accelerator Wrapper. For this design first we are going to design its controller and then implement our full design in Quartus environment and in the end we will implement this Accelerator on FPGA.**

***Keywords*— SoC, CP, Accelerator, Exponential Engine, Exponetial Accelerator Wrapper, 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.

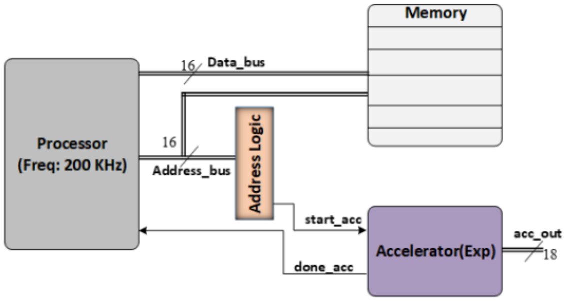


Fig.1 Block diagram of a typical integrated circuit

II. Exponential Engine

this module receives a 16-bit input ”x” and generates a 16-bit output ”Fractionalpart” 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.

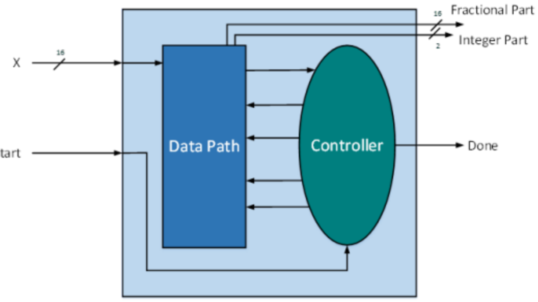


Fig.2 Block diagram of exponential accelerator

code examination by running Modelsim simulation :

We simulated the code with 3 values of x = 0.25 , 0.75, 0.1875

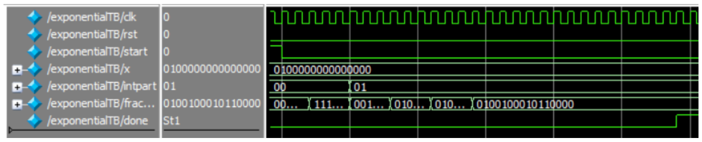
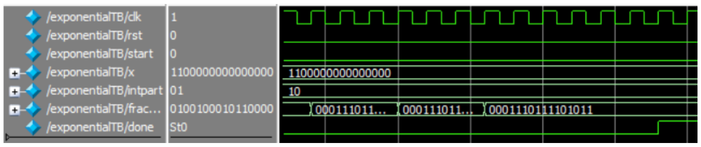


Fig.3 Value of Exponetial for x = 0.25

$$x = 0.25$$

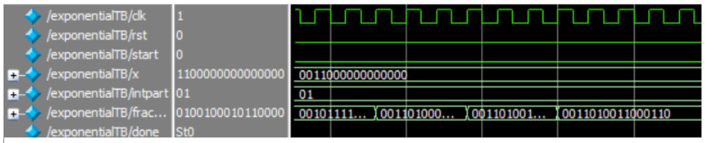
$$e^{0.25} \approx 1.284025$$
$$01.0100100010110000 \approx 1.283935546875$$



```
/exponentialTB/clk       1
/exponentialTB/rst       0
/exponentialTB/start     0
/exponentialTB/x         1100000000000000   1100000000000000
/exponentialTB/intpart   01                 10
/exponentialTB/frac...   0100100010110000   000111011... 000111011... 0001110111101011
/exponentialTB/done      St0
```

Fig.4 Value of Exponetial for x = 0.75

$$x = 0.75$$
$$e^{0.75} \approx 2.117000016$$
$$10.0001110111101011 \approx 2.1168670654296875$$



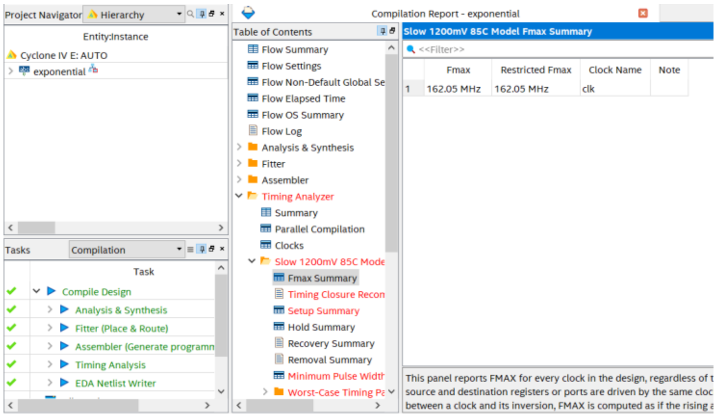
```
/exponentialTB/clk       1
/exponentialTB/rst       0
/exponentialTB/start     0
/exponentialTB/x         1100000000000000   0011000000000000
/exponentialTB/intpart   01                 01
/exponentialTB/frac...   0100100010110000   00101111... 001101000... 001101001... 0011010011000110
/exponentialTB/done      St0
```

Fig.5 Value of Exponential for x=0.1875

$$x = 0.1875$$
$$e^{0.1875} \approx 1.20623024942$$
$$01.0011010011000110 \approx 1.206146240234375$$

Design Synthesis

As shown in the figure below, The maximum frequency that this exponential calculator can operate is 162.05 MHz on EP4CE6E22C6 Slow 1200mV 85C and 181.39 MHz on the same device slow 1200mV 0C.



Slow 1200mV 85C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	162.05 MHz	162.05 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of t source and destination registers or ports are driven by the same cloc between a clock and its inversion, FMAX is computed as if the rising a

Fig.6 Fmax of Exponential Engine

III. Exponential Accelerator Wrapper

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.

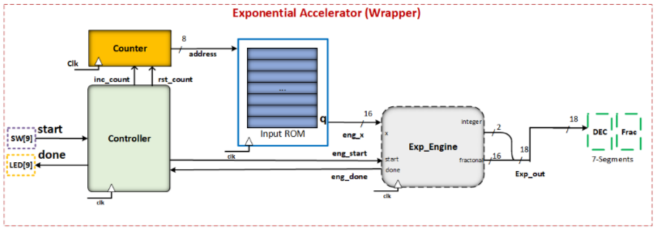


Fig.7 Exponential accelerator wrapper

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.

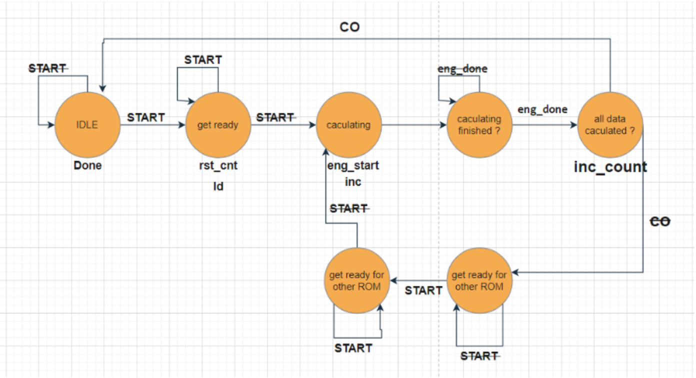


Fig.8 State machine of Exponential Accelerator

IV. Implementing Accelerator on FPGA

We connected Modules in Quartus as shown in the figure below.

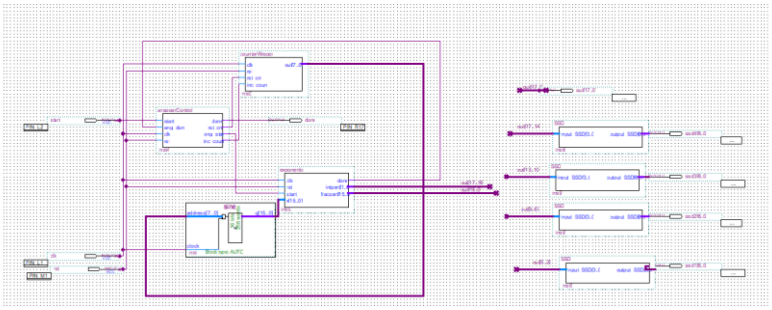


Fig.9 Final Design

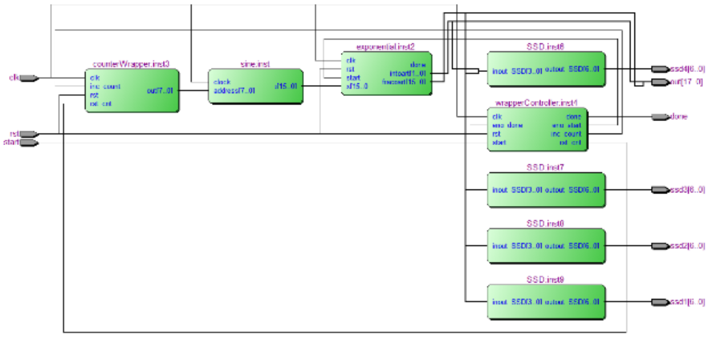
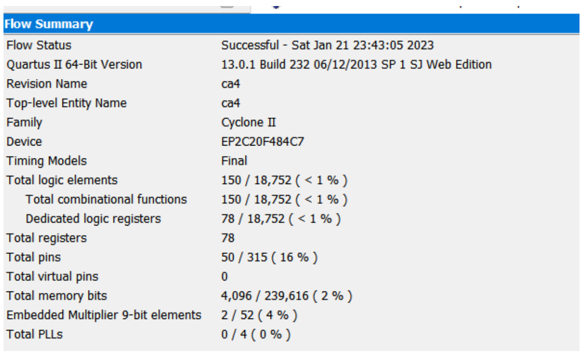


Fig.10 RTL view



Flow Summary	
Flow Status	Successful - Sat Jan 21 23:43:05 2023
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	ca4
Top-level Entity Name	ca4
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	150 / 18,752 (< 1 %)
Total combinational functions	150 / 18,752 (< 1 %)
Dedicated logic registers	78 / 18,752 (< 1 %)
Total registers	78
Total pins	50 / 315 (16 %)
Total virtual pins	0
Total memory bits	4,096 / 239,616 (2 %)
Embedded Multiplier 9-bit elements	2 / 52 (4 %)
Total PLLs	0 / 4 (0 %)

Fig.11 Flow Summary

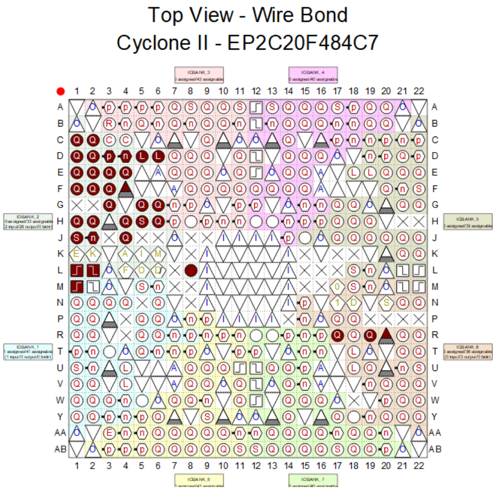
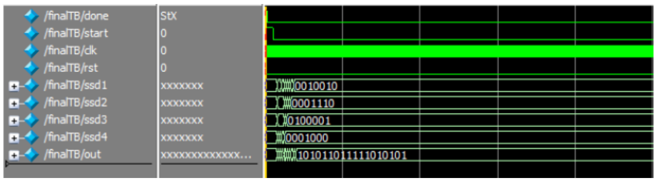


Fig.12 Pin Planner

V. Post Synthesis



```
/finalTB/done    StX
/finalTB/start   0
/finalTB/clk     0
/finalTB/rst     0
/finalTB/ssd1    xxxxxxx          0010010
/finalTB/ssd2    xxxxxxx          0001110
/finalTB/ssd3    xxxxxxx          0100001
/finalTB/ssd4    xxxxxxx          0001000
/finalTB/out     xxxxxxxxxxxxxx...  101011011111010101
```

Fig.13 Fig.4 Value of e^1

$$e^1 \approx 2.718281828459$$

$$10.1011011111010101 \approx 2.7180938720703125$$

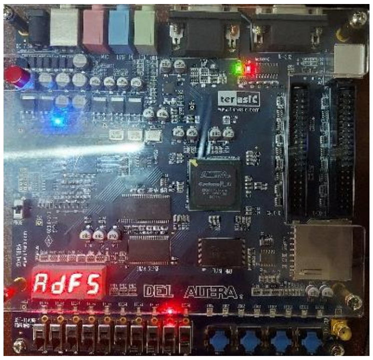


Fig.14 Value of e^1 on Board

On 7-segment we have 16 bits but our fractional part and integer part are 18 bits so we decided to give 2 bit integer and most 14 most significant bits of fractional part to 7-segment.

$$AdF5 \ to \ binary = 1010110111110101$$

$$10.10110111110101 \ to \ decimal \approx 2.718078613$$

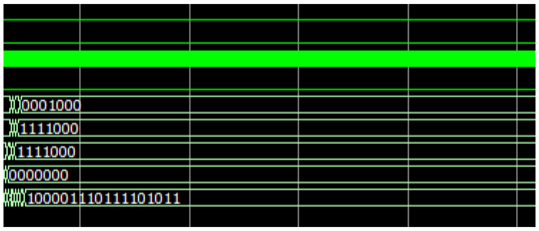


Fig.15 Value of e^0.75

$$e^{0.75} \approx 2.117000016$$

$$10.0001110111101011 \approx 2.1168670654296875$$

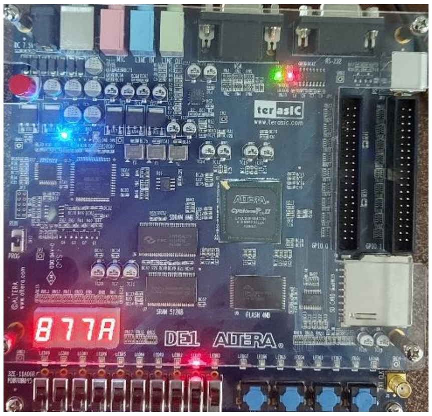


Fig.16 Value of e^0.75 on Board

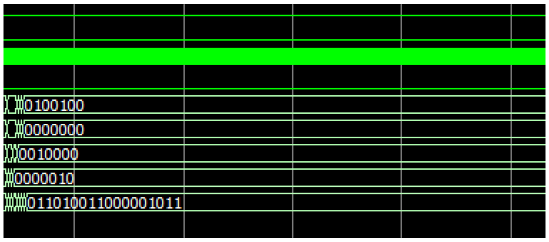


Fig.17 Value of e^0.5

$$e^{0.5} \approx 1.6487212707001$$

$$01.1010011000001011 \approx 1.6486053466796875$$

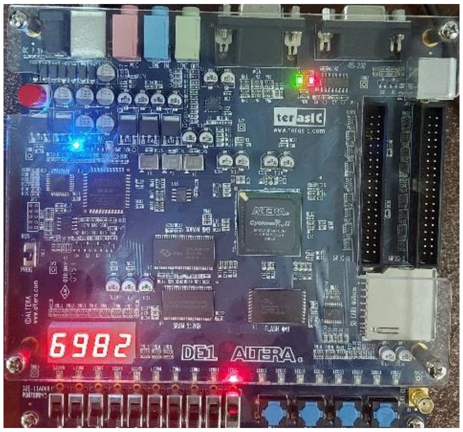


Fig.18 Value of e^0.5 on Board

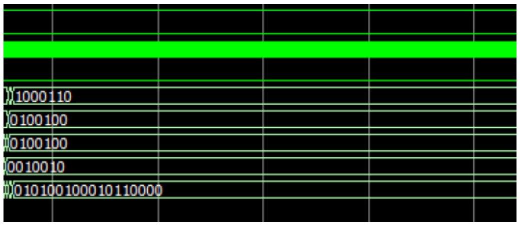


Fig.19 Value of e^0.25

$$e^{0.25} \approx 1.28402541668$$

$$01.0100100010110000 \approx 1.283935546875$$

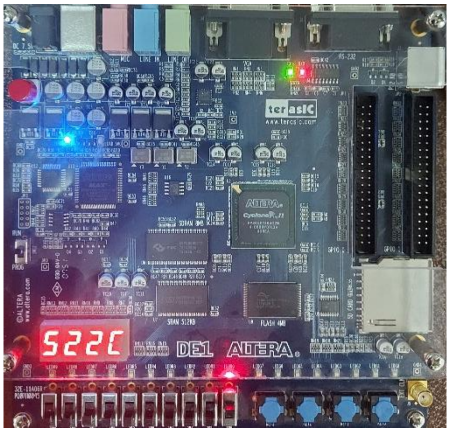


Fig.20 Value of e^0.25 on Board

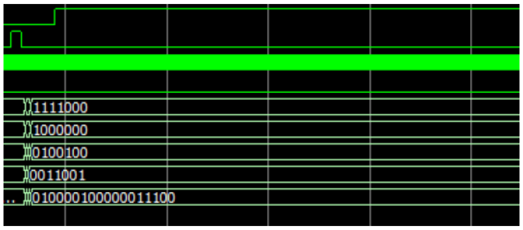


Fig.21 Value of e^0.03125

$$e^{0.03125} \approx 1.031743407499$$

$$01.0000100000011100 \approx 1.03167724609375$$

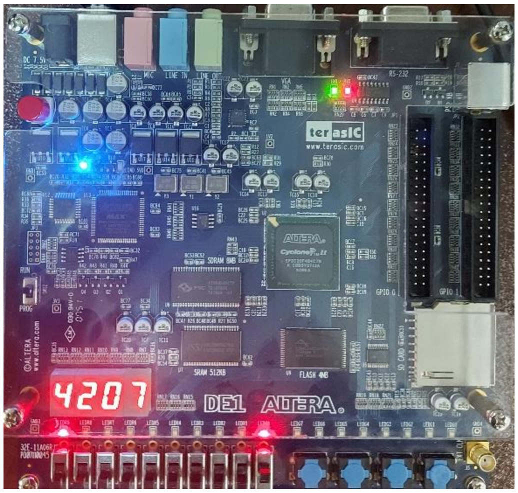


Fig.22 Value of e^0.03125 on Board

VI. CONCLUSION

In this experiment we designed an Exponential accelerator wrapper by designing its controller and implement our circuit in quartus.

and at last we tested our design by implementing it on FPGA board.

VII. REFRENCES

[1] Cyclone II Device Datasheet, Provided by intel.com