Experiment 4 – Accelerator and Wrappers

Amirhosein Yavarikhoo 810199514

***Abstract*— Exponential calculation is an important part of many high level designs. This calculation may take a lot of time if it's integrated into the processor. To improve speed and area, we use an accelerator which is specified to only do exponential function.**

***Keywords:* Accelerator, Wrapper, SOC, Handshaking, FPGA**

I. Introduction

Accelerators are used to improve speed of a system by doing a specified job. Because of this specific nature of the function, accelerator's datapath is rather simple. To connect this accelerator to other parts of a system, we use a wrapper.

II. Exponential Engine

An exponential engine is already designed. A testbench for this design is shown below:

```
Ln#
1   `timescale 1ns/1ns
2   module expoTB();
3   wire done;
4   wire [1:0] int;
5   wire [15:0] frac;
6   reg clk,rst,start;
7   reg [15:0] x;
8   exponential UUT (clk,rst,start,x,done,int,frac);
9   initial begin
10  clk=1'b0;
11  repeat (600) #5 clk=~clk;
12  end
13  initial begin
14  rst=1'b1;
15  start=1'b0;
16  #10 rst=1'b0;
17  start=1'b1;
18  x=16'd15000;
19  #5 start=1'b0;
20  #800;
21  x=16'd35000;
22  start=1'b0;
23  #5 start=1'b1;
24  #5 start=1'b0;
25  #800;
26  x=16'd50000;
27  start=1'b0;
28  #5 start=1'b1;
29  #5 start=1'b0;
30  #800;
31  end
32  endmodule
33
```

Waveform result of this testbench is shown in Fig. …. .

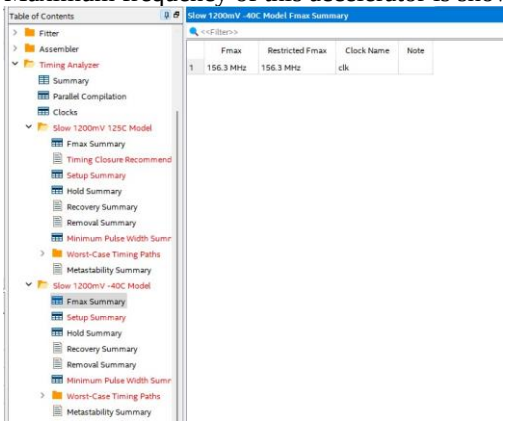
To verify the output, we calculate these exponentials by hand:

- Scenario 1: the first x is 15000. Considering that x is between 0 and 1, we have to divide it by 65536. 15000/65536=0.2288 . Exponential of this number is nearly 1.245. Integer part of this circuit is 01 which equals to 1 and fractional part is 0100000111010010 which equals to 16850. Since this is fractional part we have to divide it by 65536 which equals to 0.257. Exponential of this number was 1.245 and the output generated by circuit is 1.257 which is acceptable.
- X equals to 0.534. Exponential calculation done by hand is 1.702. Fractional part of the result generated by circuit is 1011010010101011 which equals to 0.705. Integer part of the result is 1. Calculation done by hand is 1.702 and the calculation done by circuit is 1.705 which is acceptable.
- X equals to 0.763. Exponential calculation done by hand is 2.1397. Fractional part of the result generated by circuit is 0010010011111000 which equals to 0.144. Integer part of the result is 2. Calculation done by hand is 2.1397 and the calculation done by circuit is 2.144 which is acceptable.

Synthesis report of this engine is shown below:

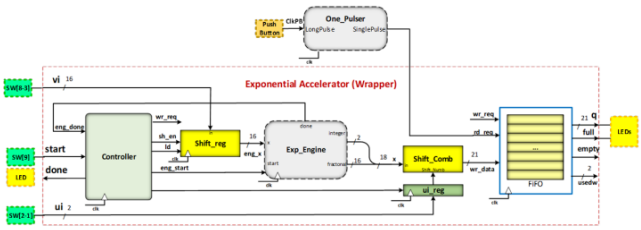
Flow Status	Successful - Wed May 24 08:52:00 2023
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	exponential
Top-level Entity Name	exponential
Family	Cyclone IV E
Device	EP4CE6E22A7
Timing Models	Final
Total logic elements	101 / 6,272 (2 %)
Total registers	60
Total pins	38 / 92 (41 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	2 / 30 (7 %)
Total PLLs	0 / 2 (0 %)

Maximum frequency of this accelerator is shown in Fig … .



III. Exponential Accelerator Wrapper

Datapath and Controller diagram of this wrapper is shown below:



This wrapper has multiple parts. Verilog description of the controller is shown below:

```
module wrapper_cu (input clk,rst,start,eng_done,co,output reg done,wr_req,sh_en,ld,eng_start,ui_reg_ld,cnt_en,cnt_rst);
  reg [2:0] ps, ns;
  parameter [2:0]
  S0 = 0, S1 = 1, S2 = 2, S3 = 3, S4 = 4, S5=5;
  always@(ps, start, eng_done, co)begin
    ns = S0;
    case(ps)
      S0:
        ns = (start)? S1 : S0;
      S1:
        ns = (start)? S1 : S2;
      S2:
        ns = S3;
      S3:
        ns = S4;
      S4: ns = (eng_done) ? ((~co)?S5:S0):S4;
      S5:
        ns = S3;
    endcase
  end

  always@(ps, start, eng_done, co)begin
  done = 1'b0; wr_req = 1'b0; sh_en = 1'b0; ld = 1'b0; eng_start = 1'b0; ui_reg_ld = 1'b0; cnt_rst=1'b0;
      cnt_en=1'b0;
  case(ps)
      S0:begin
        cnt_rst=1'b1;
        done=1'b1;
      end
      S1: begin

      end
      S2:begin
        ld = 1'b1;
        ui_reg_ld = 1'b1;
      end
      S3:begin
        eng_start = 1'b1;
      end
      S4:begin
      end
      S5:begin
        wr_req =1'b1;
        sh_en = 1'b1;
        cnt_en = 1'b1;
      end
    endcase
  end

  always@(posedge clk,posedge rst)begin
    if(rst == 1'b1)
      ps <= S0;
    else
      ps <= ns;
  end
endmodule
```

We also need a shift register. Verilog description of this component is shown below:

```
module shiftReg(input clk, rst, sh_en, ld, input[15:0] vi, output [15:0] out);
reg [15:0] shift_reg;
always@(posedge clk) begin
        if (rst) shift_reg = 16'd0;
        else if (ld) shift_reg=vi;
        else if (sh_en) shift_reg= shift_reg<<1;

end
assign out=shift_reg;
endmodule
```

A simple combinational shif and one pulser generator is needed too. Verilog descriptions of these components are shown in Fig … and Fig …. .

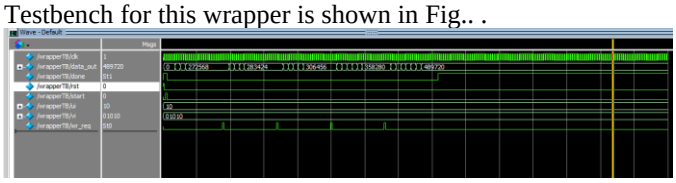
```
module shiftComb(input[1:0] shiftNum, input[17:0] in, output [20:0] out);
assign out = {3'b000,in} << shiftNum;
endmodule
```

```
module one_pulser( input clk, rst, clkPB, output reg clkEn );
    reg [1:0] ps, ns;

    always @(posedge clk, posedge rst) begin

  ns = 2'b0;
  case (ps)
   2'b0 :       ns = clkPB ? 2'b01 : 2'b0;
   2'b01 : begin ns = 2'b10; clkEn = 1'b1; end
   2'b10 : begin ns =  clkPB ? 2'b10 : 2'b0; clkEn = 1'b0; end
   default: ns = 2'b0;
   endcase
     end
  always @(posedge clk, posedge rst) begin
  if(rst)
   ps <= 2'b0 ;
  else
   ps <= ns;
     end
endmodule
```

These components are connected and instanciated in the code shown below:

```
module wrapper (input clk,start,rst,input [1:0]ui,input[4:0] vi,output wr_req,output [20:0] wr_data,output done);
wire [15:0] shift_reg_input;
assign shift_reg_input={3'b000,vi,8'b00000000};
wire [15:0] exp_input,exp_frac;
wire [1:0] exp_int,ui_out;
wire exp_done,ui_load,sh_en,shift_reg_load,exp_start,shift_reg_en,count_en,rst_count,count_co;
exponential expo (clk,rst,exp_start,exp_input,exp_done,exp_int,exp_frac);
shiftComb shiftcomb (ui_out,{exp_int,exp_frac},wr_data);
Register ui_reg (clk,rst,ui_load,ui,ui_out);
shiftReg shiftreg (clk,rst,shift_reg_en,shift_reg_load,shift_reg_input,exp_input);
wrapper_cu cu (clk,rst,start,exp_done,count_co,done,wr_req,shift_reg_en,shift_reg_load,exp_start,ui_load,count_en,rst_count);
twocounter count (clk,rst_count,count_en,count_co);
endmodule
```

Testbench for this wrapper is shown in Fig.. .



Vi is 0000101000000000 which equals to 0.0390625.
ui is 2.
We have:

$$z = \frac{u}{log_2^e}$$

So z=1.386
According to the formulas:

$$output = e^{z+v*2^{N-1}}$$

Where N is each iteration of the calculation.

Hand calculation	Result from accelerator	N
4.157	4.159	1
4.323	4.324	2
4.674	4.676	3
5.463	5.466	4

Secodn scenario:
Vi is 00011000000000 which equals to 0.0234375.
Ui is 3.
Z equals to 2.0794.

Hand calculation	Result from accelerator	N
8.189	8.1892	1
8.383	8.383	2
8.781	8.785	3
9.611	9.649	4

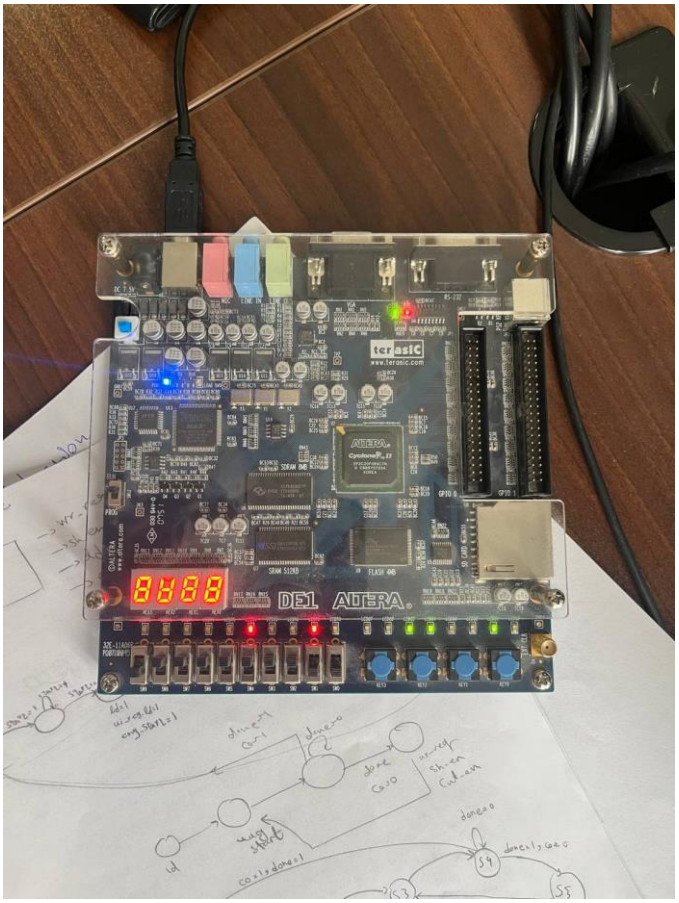
Outputs are almost equal to hand calculations.

Max frequency of this wrapper is shown below:

Slow 1200mV 125C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	136.11 MHz	136.11 MHz	clk	

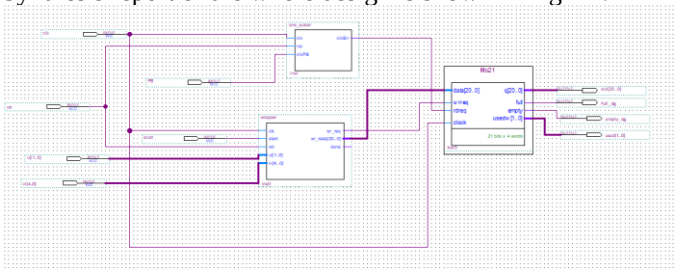


IV. Implementing Accelerator on FPGA

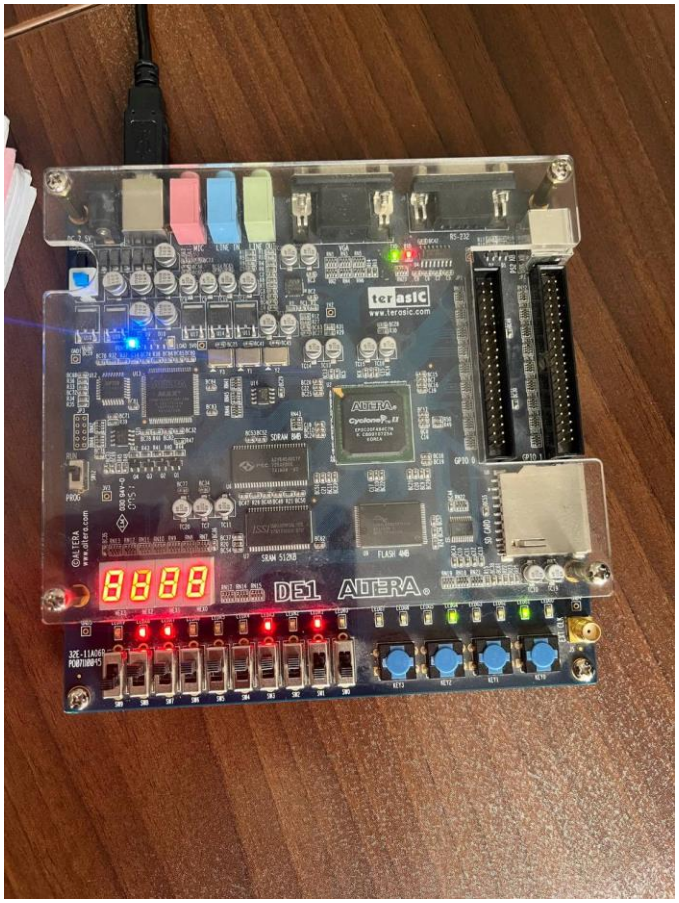
Accelerator with FIFO and Onepulser design is shown below:

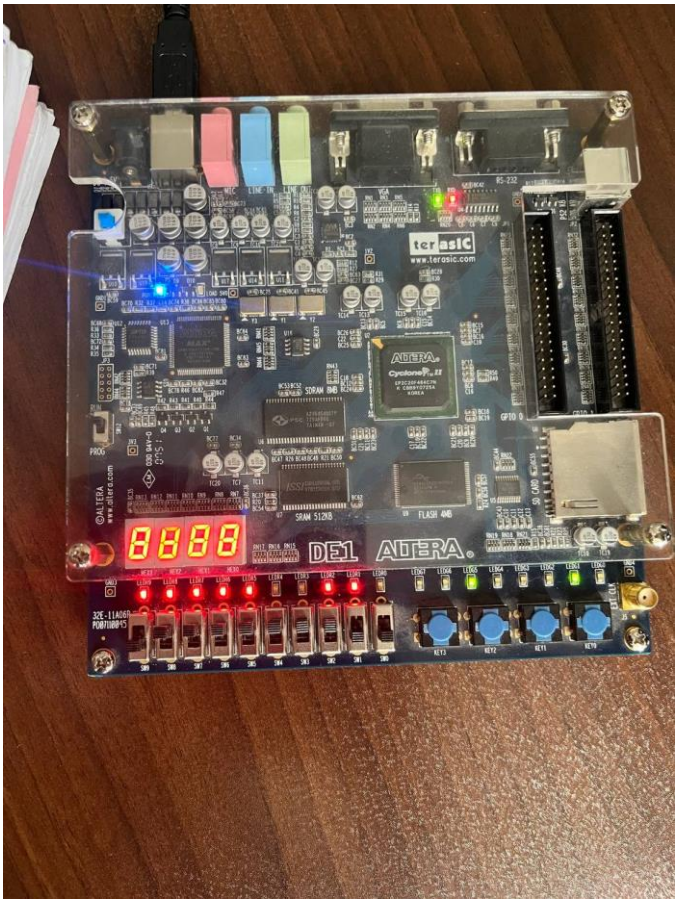
Flow Status	Successful - Mon Jun 12 08:19:17 2023
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	LAB4
Top-level Entity Name	LAB4
Family	Cyclone IV E
Device	EP4CE6E22A7
Timing Models	Final
Total logic elements	170 / 6,272 (3 %)
Total registers	82
Total pins	36 / 92 (39 %)
Total virtual pins	0
Total memory bits	84 / 276,480 (< 1 %)
Embedded Multiplier 9-bit elements	2 / 30 (7 %)
Total PLLs	0 / 2 (0 %)

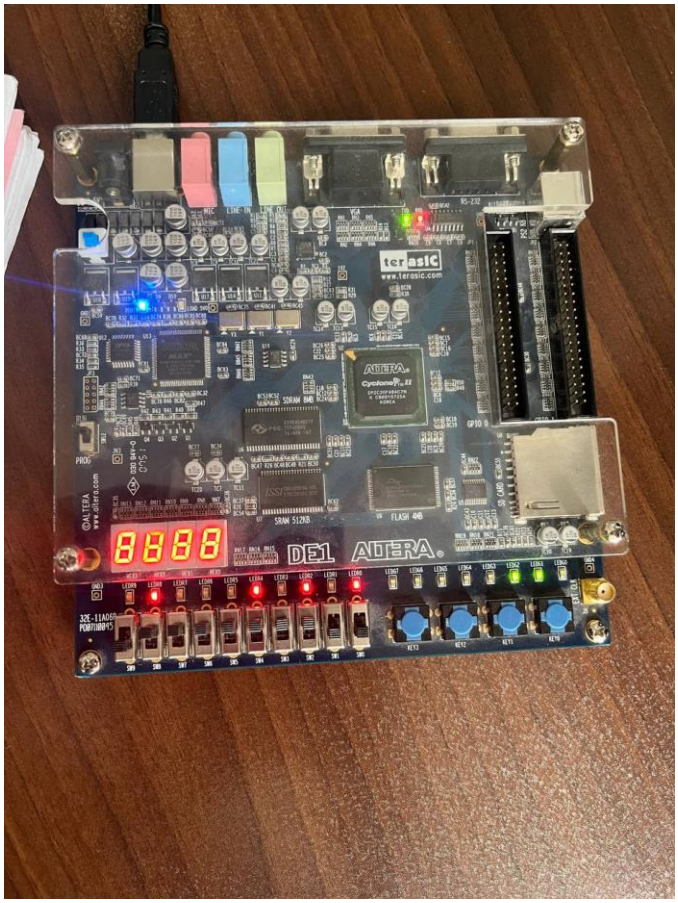
Synthesis report of the whole design is shown in Fig … .



Results of the testing on board is shown below:


terasIC
www.terosic.com
DE1
ALTERA


terasIC
www.terosic.com
DE1
ALTERA



LED green 5 and 4 reperesent used signal that fifo shows. As we push the button each time an ouput pops from the FIFO. Red LEDs show most significant bits of output.