

EXPERMINT 4

Group's members:

Mohammad Taghizadeh 810198373

Arian Niakan 810198536

Contents

Exponential Engine.....	2
Wrapper Controller.....	6
Implementing Accelerator on FPGA.....	10

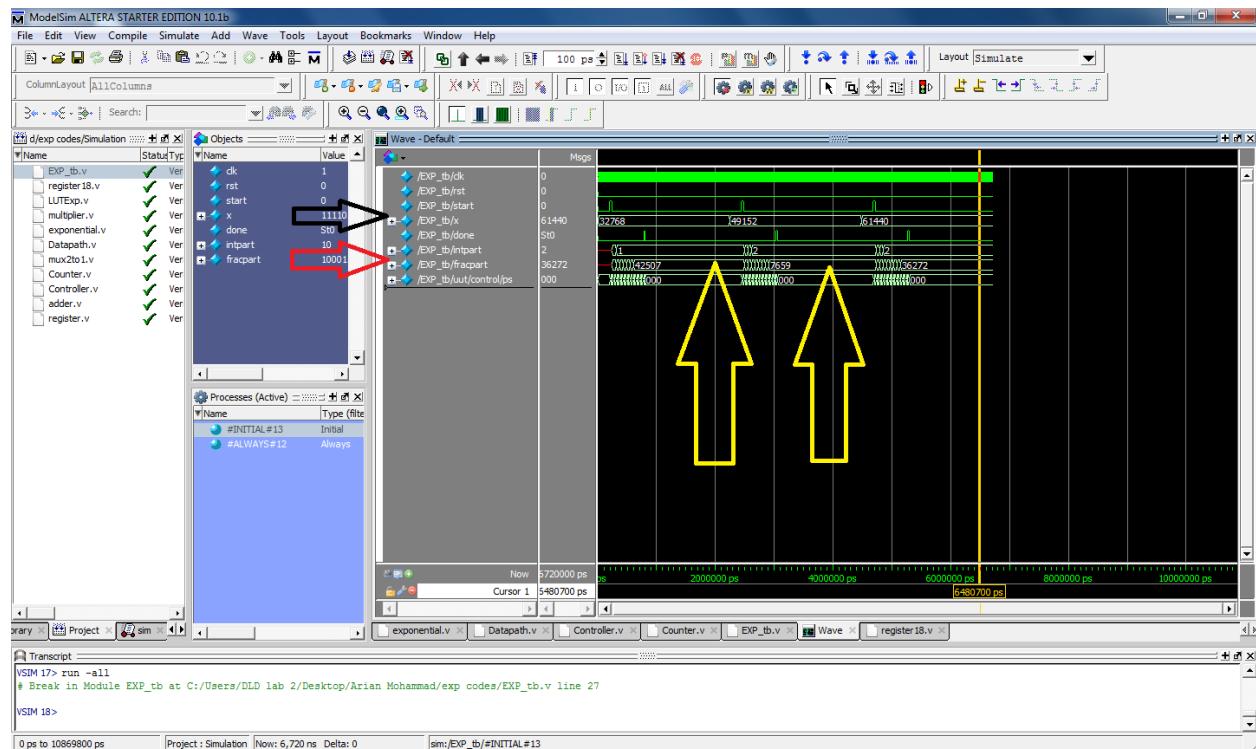
1. Exponential Engine

Exponential Engine Test Bench

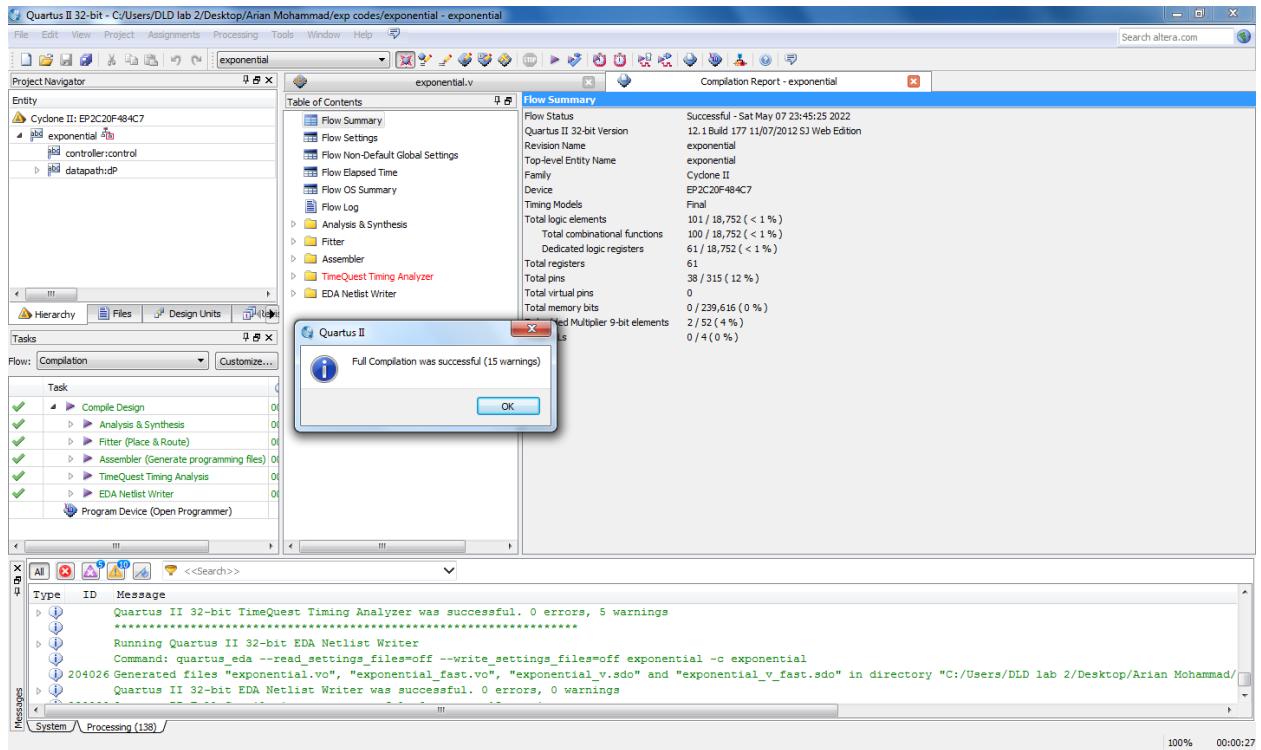
The screenshot shows the ModelSim ALTERA Starter Edition 10.1b interface. The main window displays the simulation script for `EXP_tb.v`. The script defines a module `EXP_tb()` with various ports and internal logic. It includes initial conditions for `clk`, `rst`, and `start`, and a process block containing multiple assignments and a final `$stop` statement. The left pane shows a tree view of the project files, including `register18.v`, `LUTExp.v`, `multiplexer.v`, `exponential.v`, `Datapath.v`, `mu2to1.v`, `Counter.v`, `Controller.v`, `adder.v`, and `register.v`, all marked as verified. The bottom pane shows the transcript window with the command `VSIM 17> run -all` and the message "# Break in Module EXP_tb at C:/Users/DLD lab 2/Desktop/Arian Mohammad/exp codes/EXP_tb.v line 27".

```
2 module EXP_tb();
3   reg clk, rst, start;
4   reg [15:0] x;
5   reg [15:0] done;
6   reg [10:0] intpart;
7   reg [15:0] fracpart;
8
9   exponential uut (clk, rst, start, x, done, intpart, fracpart);
10  always #10 clk=~clk;
11  initial begin
12    clk=0;
13    rst=0;
14    start=0;
15    x=16'b1000000000000000;
16
17    #200 start = 1;
18    #40 start = 0;
19    #2000 x=16'b1000000000000000;
20    #200 start = 1;
21    #40 start = 0;
22    #2000 x=16'b1111000000000000;
23    #200 start = 1;
24    #40 start = 0;
25
26    #2000 $stop;
27  end
28
29
30
31
32
endmodule
```

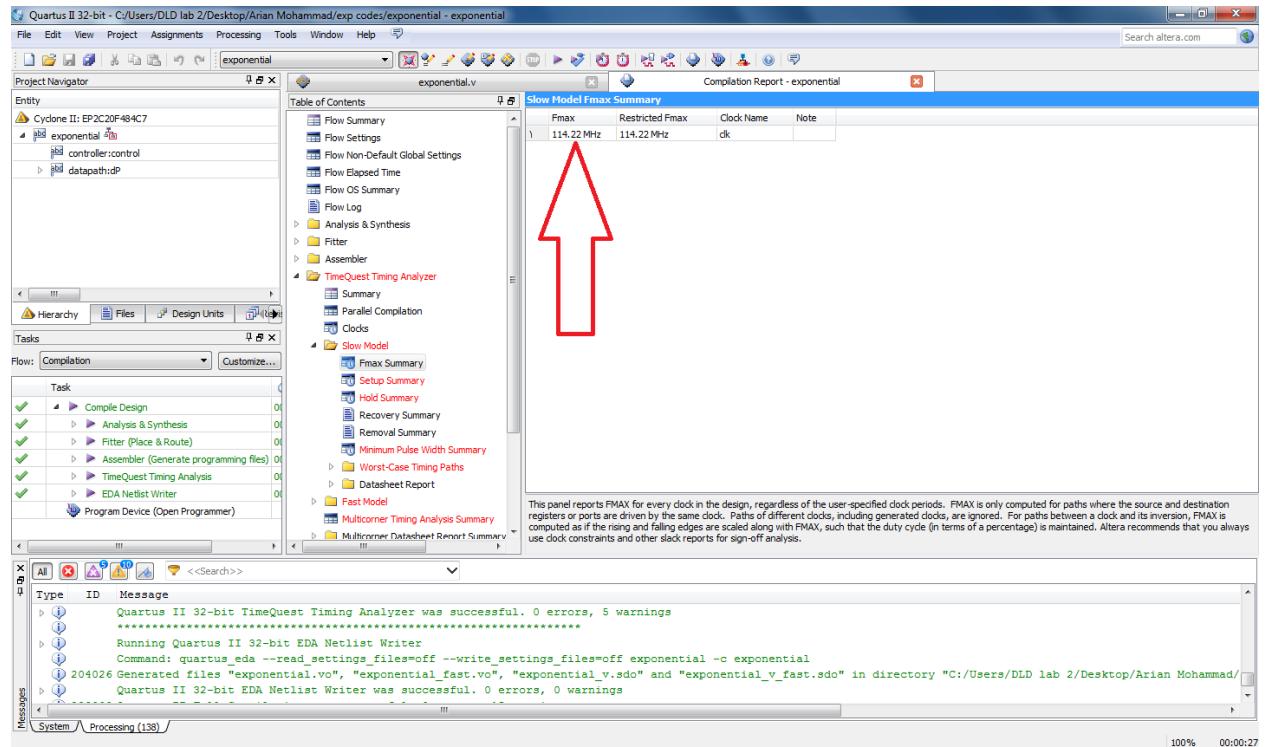
Simulation Output of above Exponential Engine Test Bench



Quartus Synthesis Report of Exponential Engine

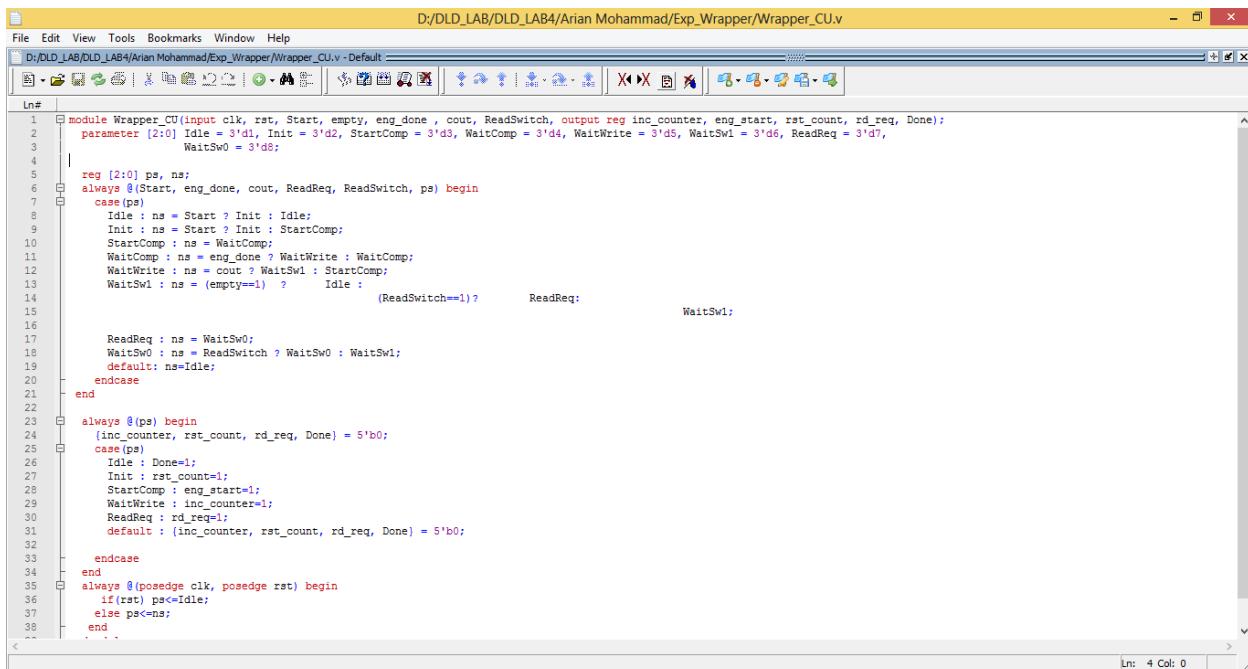


Maximum Frequency of Exponential Engine



2.2 Wrapper Controller

Wrapper Controller Verilog Code

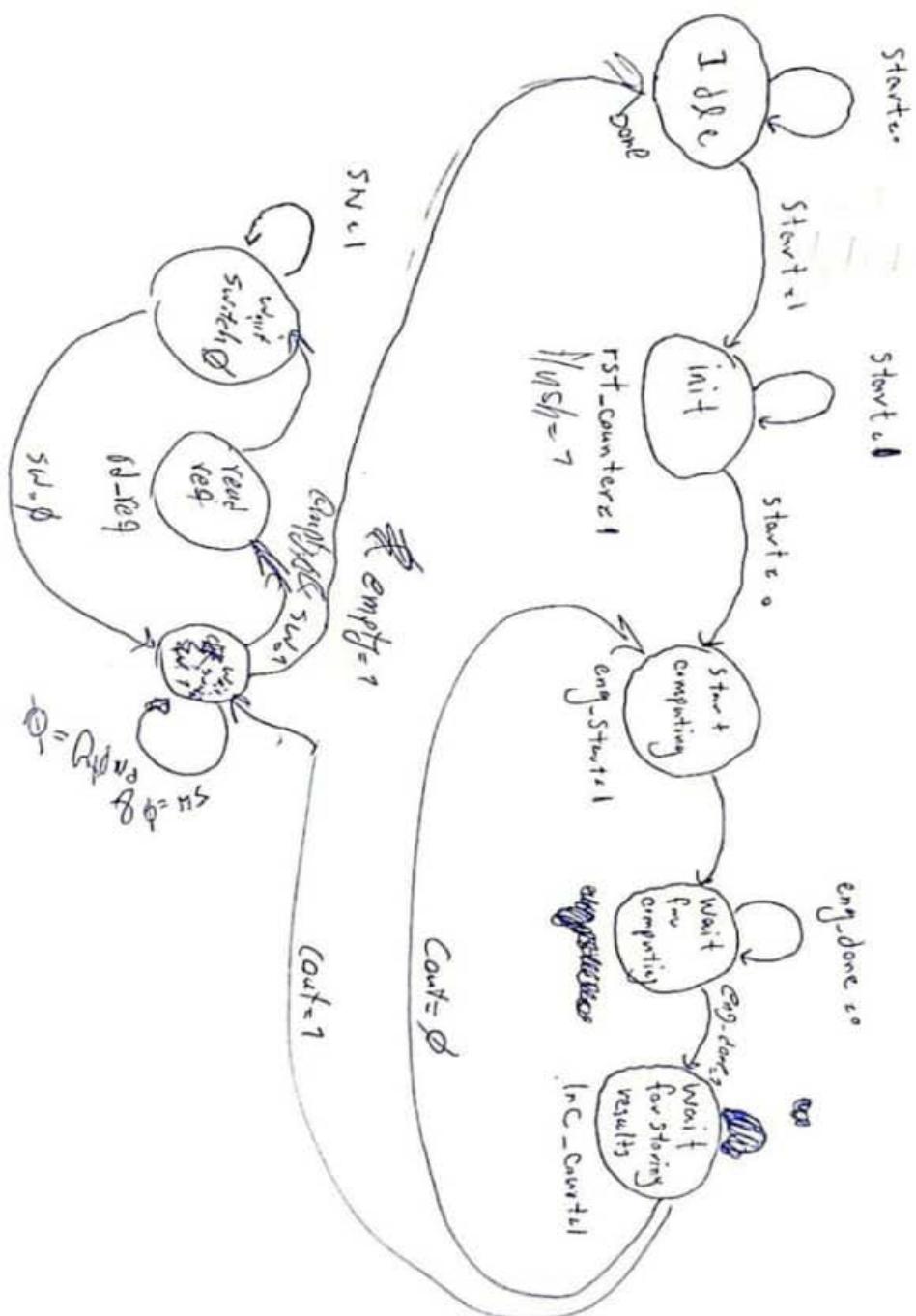


The screenshot shows a Verilog code editor window with the following details:

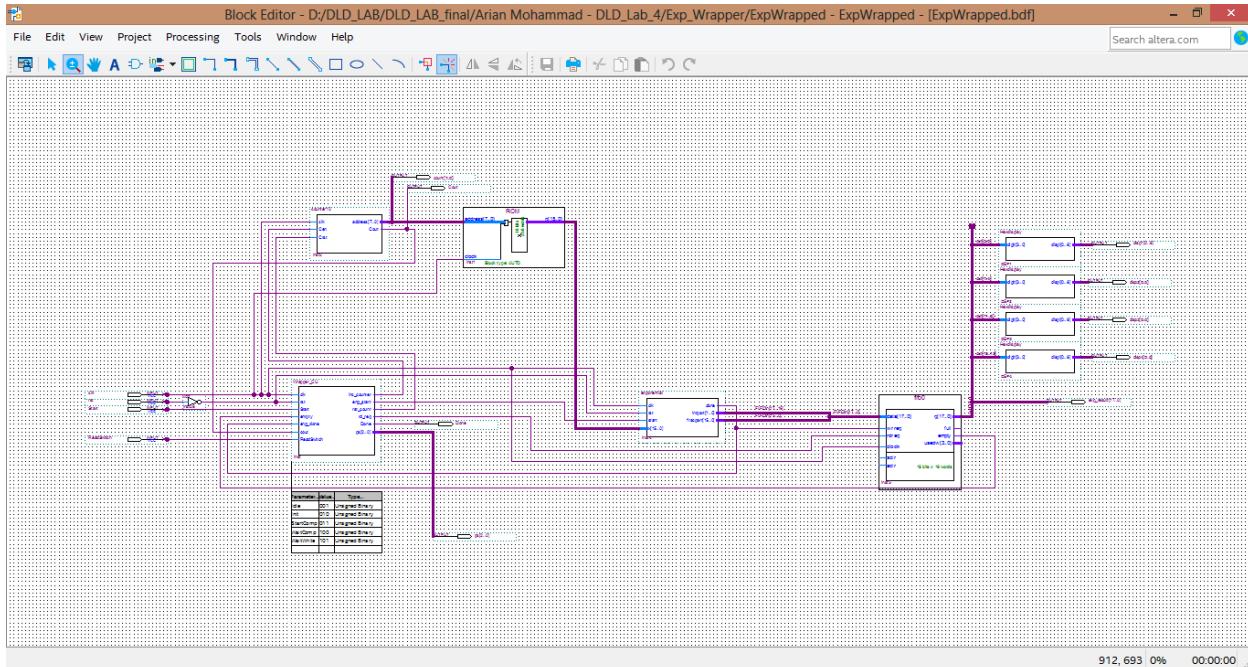
- Title Bar:** D:/DLD_LAB/DLD_LAB4/Arian Mohammad/Exp_Wrapper/Wrapper_CU.v - Default
- Toolbar:** Standard Verilog editor toolbar with icons for file operations, search, and zoom.
- Code Area:** The main area contains the Verilog code for the `Wrapper CU` module. The code is well-structured with indentation and syntax highlighting.
- Status Bar:** Shows "Ln: 4 Col: 0" at the bottom right.

```
1 module Wrapper_CU(input clk, rst, Start, empty, eng_done, cout, ReadSwitch, output reg inc_counter, eng_start, rst_count, rd_req, Done);
2   parameter [2:0] Idle = 3'd1, StartComp = 3'd3, WaitComp = 3'd4, WaitWrite = 3'd5, WaitSw1 = 3'd6, ReadReq = 3'd7,
3   WaitSw0 = 3'd8;
4
5   reg [2:0] ps, ns;
6   always @{Start, eng_done, cout, ReadReq, ReadSwitch, ps} begin
7     case(ps)
8       Idle : ns = Start ? Idle;
9       Init : ns = Start ? Init : StartComp;
10      StartComp : ns = WaitComp;
11      WaitComp : ns = eng_done ? WaitWrite : WaitComp;
12      WaitWrite : ns = cout ? WaitSw1 : StartComp;
13      WaitSw1 : ns = (empty==1) ? Idle :
14                                (ReadSwitch==1) ? ReadReq:
15                                WaitSw1;
16
17      ReadReq : ns = WaitSw0;
18      WaitSw0 : ns = ReadSwitch ? WaitSw0 : WaitSw1;
19      default: ns=Idle;
20    endcase
21  end
22
23  always @(ps) begin
24    {inc_counter, rst_count, rd_req, Done} = 5'b0;
25    case(ps)
26      Idle : Done=1;
27      Init : rst_count=1;
28      StartComp : eng_start=1;
29      WaitWrite : inc_counter=1;
30      ReadReq : rd_req=1;
31      default : {inc_counter, rst_count, rd_req, Done} = 5'b0;
32    endcase
33  end
34
35  always @(posedge clk, posedge rst) begin
36    if(rst) ps<=Idle;
37    else ps<=ns;
38  end
39
```

Wrapper Controller State Diagram`



Exponential Accelerator Schematic Diagram



Exponential Accelerator Test Bench

The screenshot shows a text editor window displaying Verilog test bench code for the Exponential Accelerator. The code initializes the system, sets up a continuous loop for reading data from a switch, and configures various control signals like Done, clk, reset, Start, and ReadSwitch. The code uses a mix of procedural statements and logic assignments. The top menu bar includes File, Edit, View, Tools, Bookmarks, Window, and Help. The bottom status bar shows line (Ln: 4) and column (Col: 0) information.

```
8 wire [2:0] ps;
9 wire [7:0] count;
10 wire [6 : 0] disp1, disp2, disp3, disp4;
11 wire Cout;
12
13 ExWrapped Exponential_Accelerator(Done, clk, reset, Start, ReadSwitch,Cout,count, disp1, disp2, disp3, disp4, exp_result, ps);
14
15 always #50 clk = ~clk;
16 initial begin
17   ReadSwitch = 0;
18   clk = 0; reset = 0; Start = 0;
19   #100 reset = 1;
20   #100 Start = 1;
21   #100 Start = 0;
22   #200000 ReadSwitch = 1;
23   #100 ReadSwitch = 0;
24   #100 ReadSwitch = 1;
25   #100 ReadSwitch = 0;
26   #100 ReadSwitch = 1;
27   #100 ReadSwitch = 0;
28   #100 ReadSwitch = 1;
29   #100 ReadSwitch = 0;
30   #100 ReadSwitch = 1;
31   #100 ReadSwitch = 0;
32   #100 ReadSwitch = 1;
33   #100 ReadSwitch = 0;
34   #100 ReadSwitch = 1;
35   #100 ReadSwitch = 0;
36   #100 ReadSwitch = 1;
37   #100 ReadSwitch = 0;
38   #100 ReadSwitch = 1;
39   #100 ReadSwitch = 0;
40   #100 ReadSwitch = 1;
41   #100 ReadSwitch = 0;
42   #100 ReadSwitch = 1;
43   #100 ReadSwitch = 0;
44   #100 ReadSwitch = 1;
45   #100 ReadSwitch = 0;
```

D:/DLD_LAB/DLD_LAB_final/Arian Mohammad - DLD_Lab_4/Exp_Wrapper/Exponential_Accelerator_TestBench.v

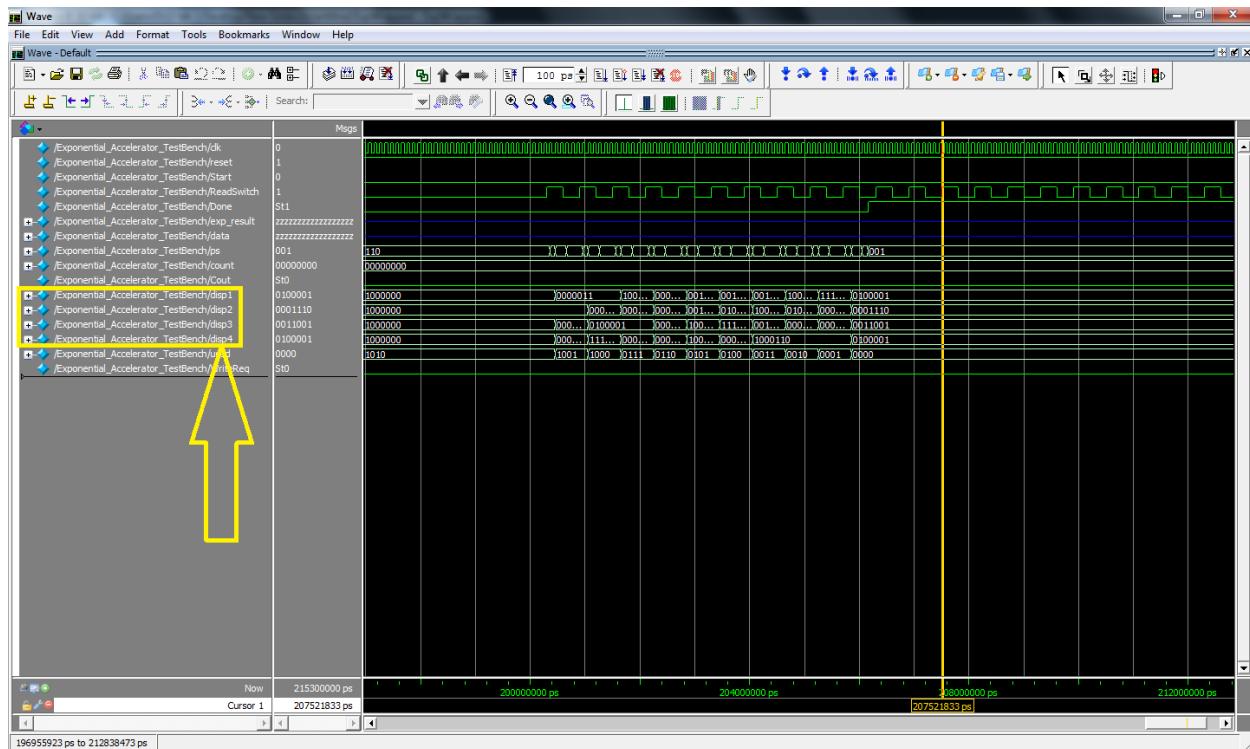
```

File Edit View Tools Bookmarks Window Help
D:/DLD_LAB/DLD_LAB_final/Arian Mohammad - DLD_Lab_4/Exp_Wrapper/Exponential_Accelerator_TestBench.v - Default
Ln# 20 #100 Start = 1;
21 #100 Start = 0;
22 #200000 ReadSwitch = 1;
23 #100 ReadSwitch = 0;
24 #100 ReadSwitch = 1;
25 #100 ReadSwitch = 0;
26 #100 ReadSwitch = 1;
27 #100 ReadSwitch = 0;
28 #100 ReadSwitch = 1;
29 #100 ReadSwitch = 0;
30 #100 ReadSwitch = 1;
31 #100 ReadSwitch = 0;
32 #100 ReadSwitch = 1;
33 #100 ReadSwitch = 0;
34 #100 ReadSwitch = 1;
35 #100 ReadSwitch = 0;
36 #100 ReadSwitch = 1;
37 #100 ReadSwitch = 0;
38 #100 ReadSwitch = 1;
39 #100 ReadSwitch = 0;
40 #100 ReadSwitch = 1;
41 #100 ReadSwitch = 0;
42 #100 ReadSwitch = 1;
43 #100 ReadSwitch = 0;
44 #100 ReadSwitch = 1;
45 #100 ReadSwitch = 0;
46 #100 ReadSwitch = 1;
47 #100 ReadSwitch = 0;
48 #100 ReadSwitch = 1;
49 #100 ReadSwitch = 0;
50 #100 ReadSwitch = 1;
51 #100 ReadSwitch = 0;
52 #100 ReadSwitch = 1;
53 #100 ReadSwitch = 0;
54 #100 $stop;
55 end
56 endmodule

```

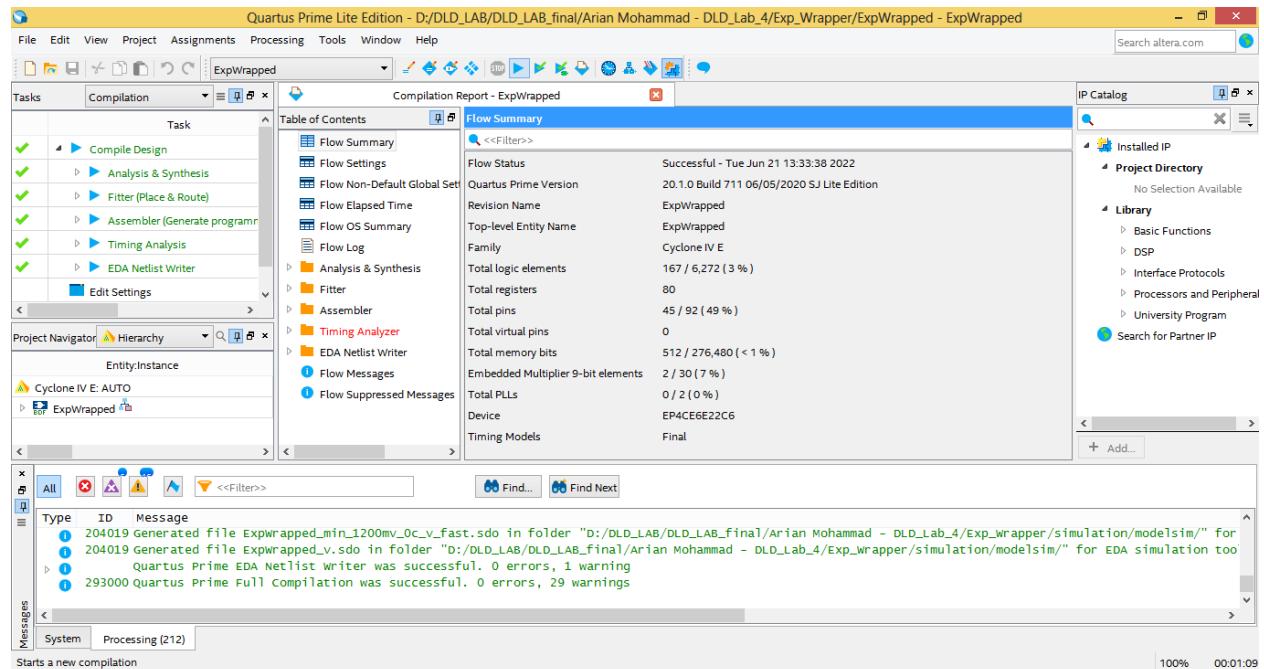
Ln: 4 Col: 0

Simulation Output of above Exponential Accelerator Test Bench

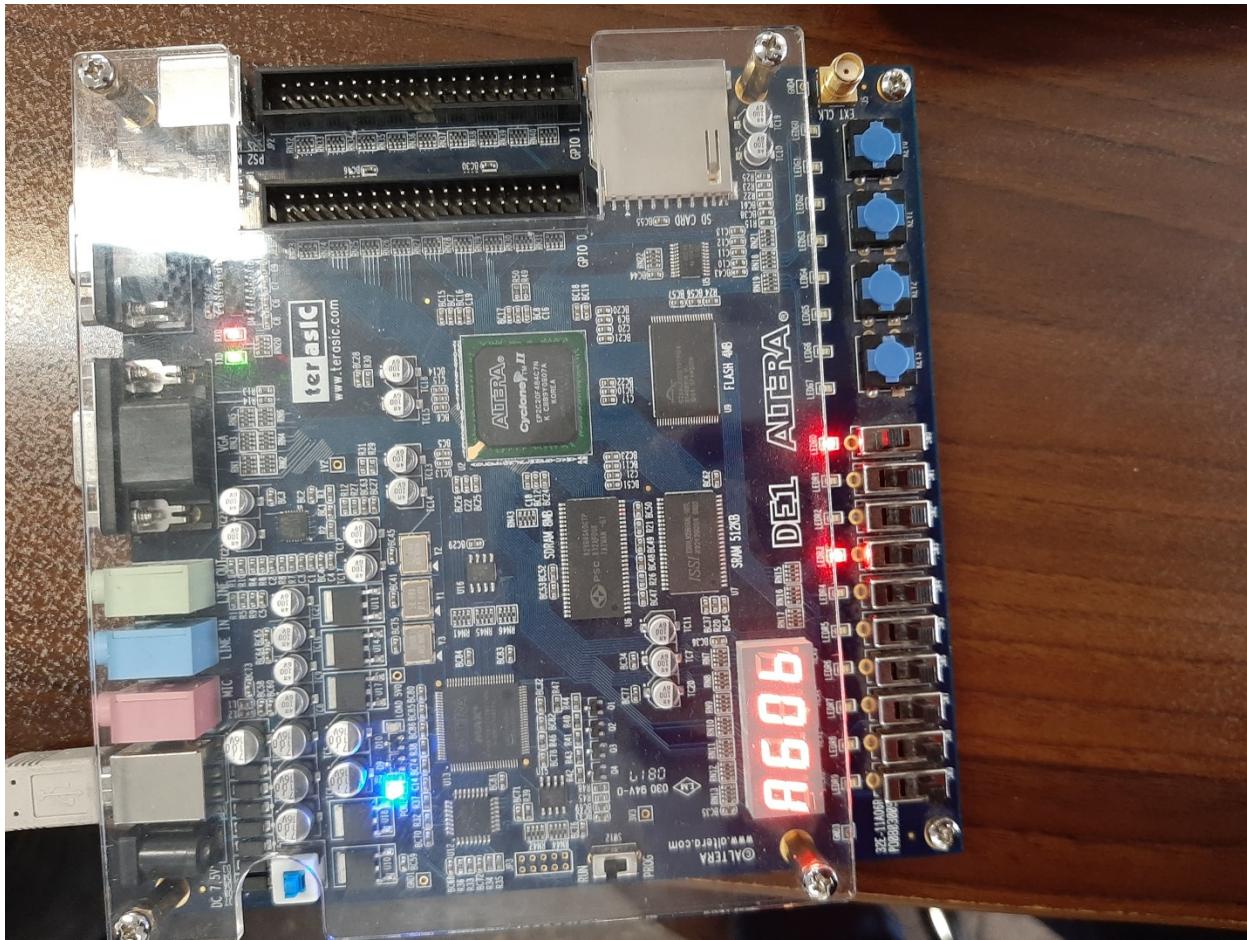


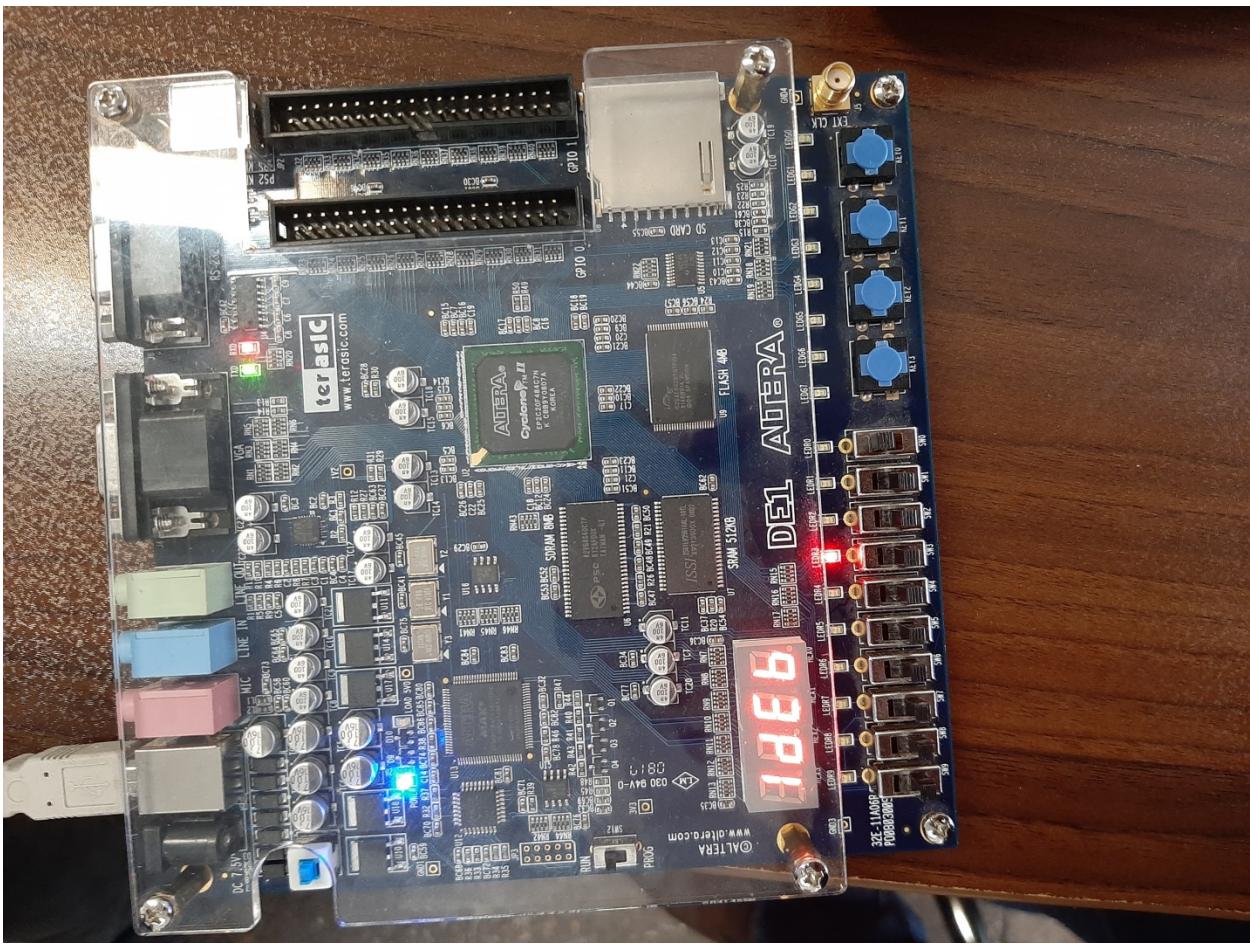
3 Implementing Accelerator on FPGA

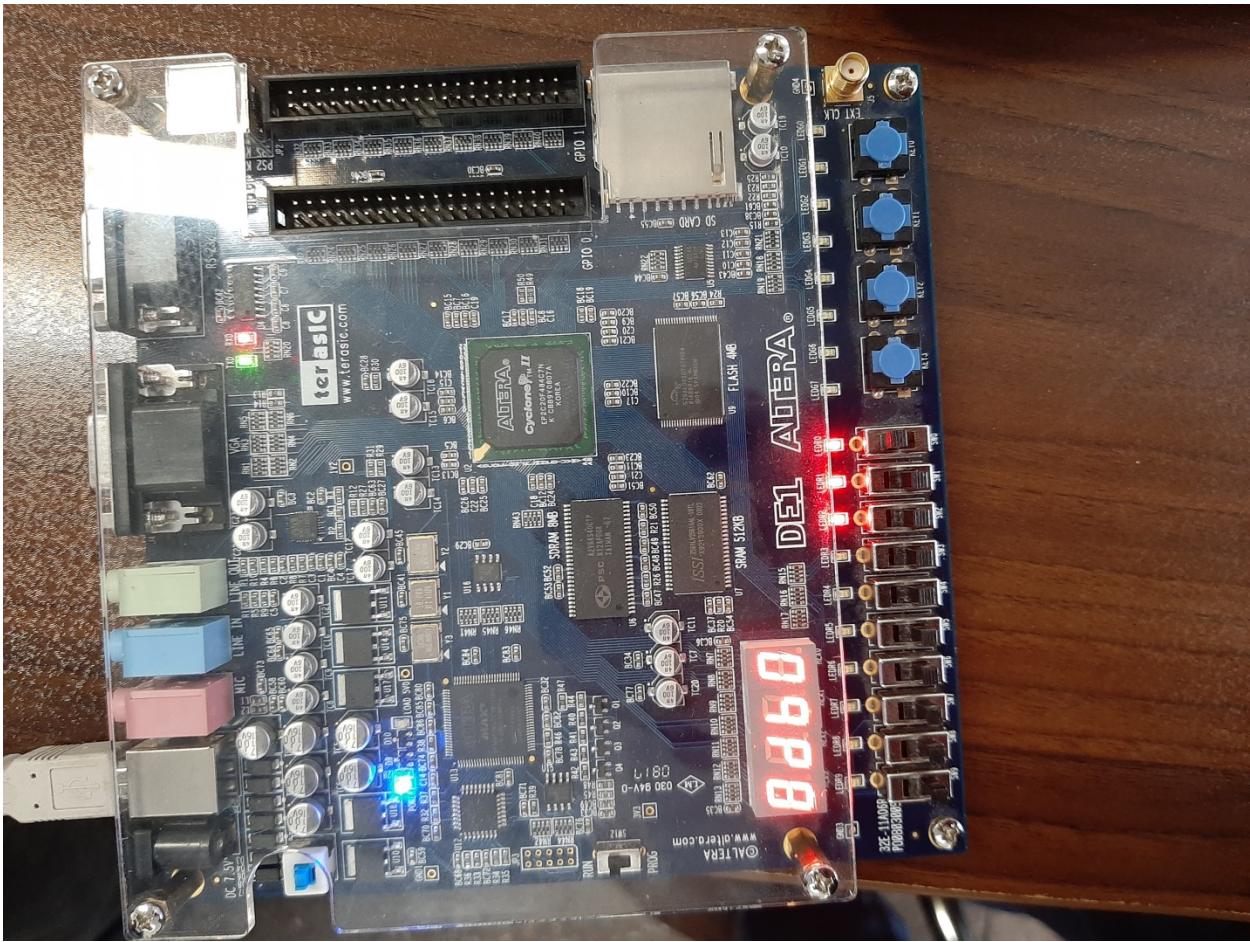
Quartus Synthesis Report of Exponential Accelerator

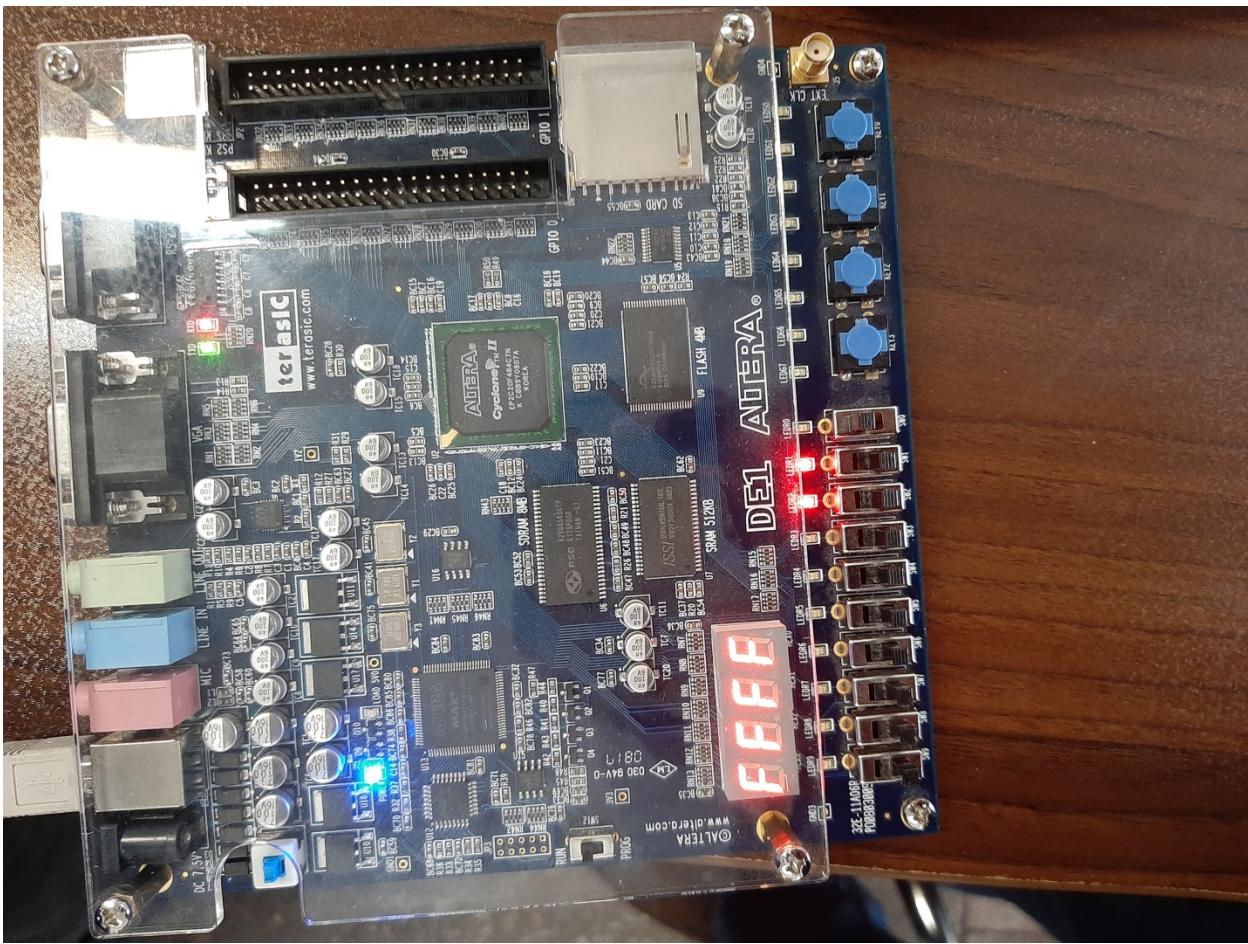


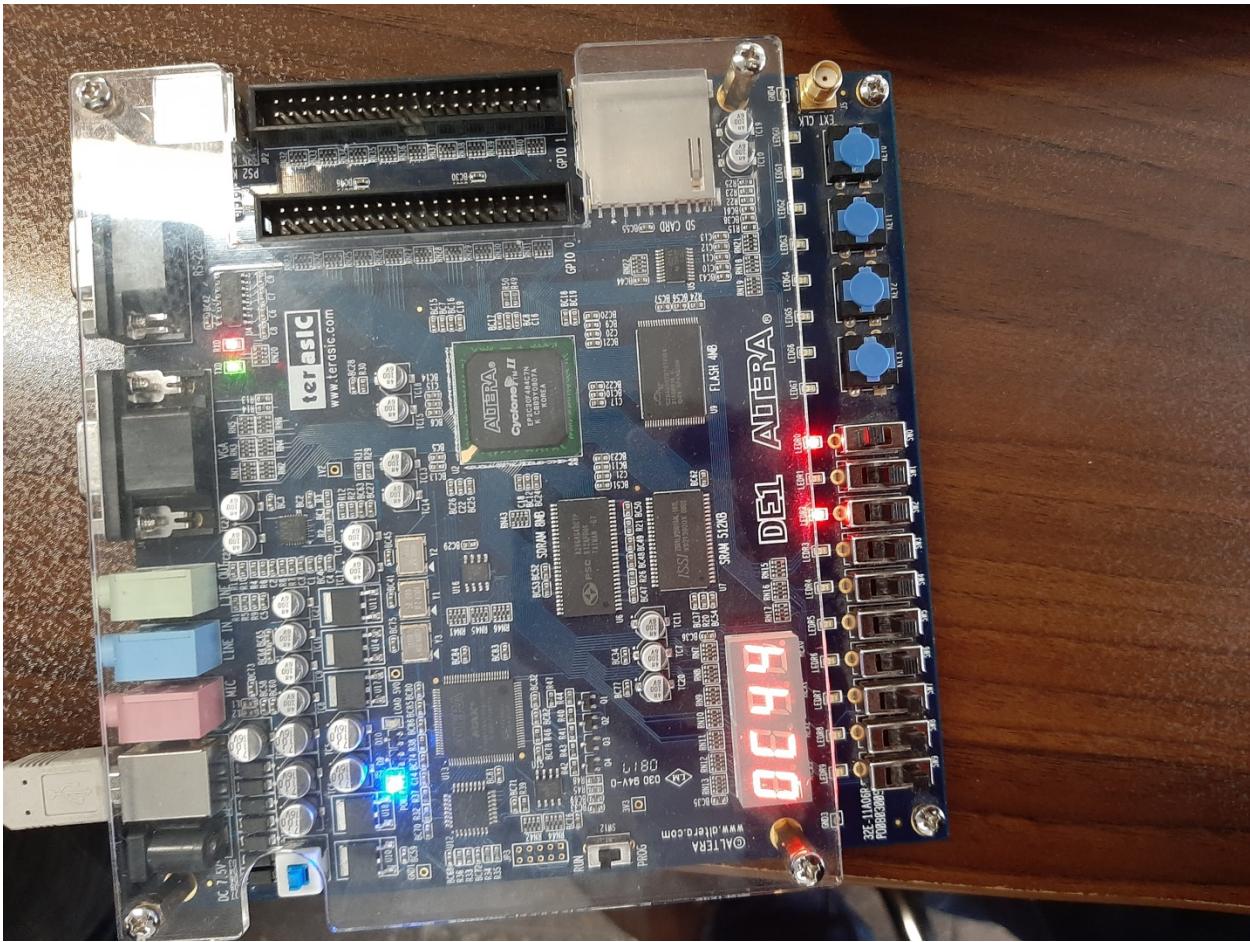
Output Results of Exponential Accelerator on FPGA

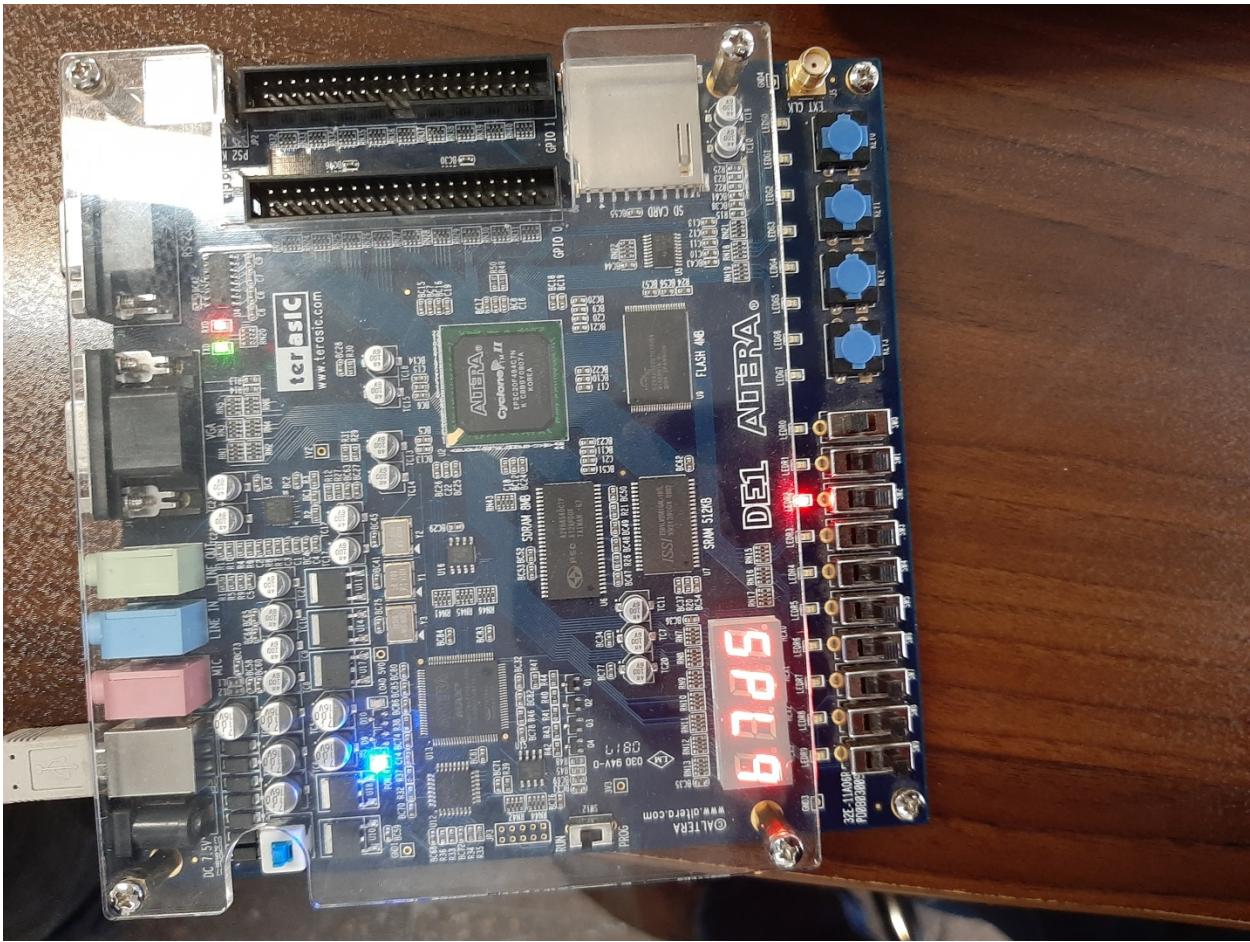


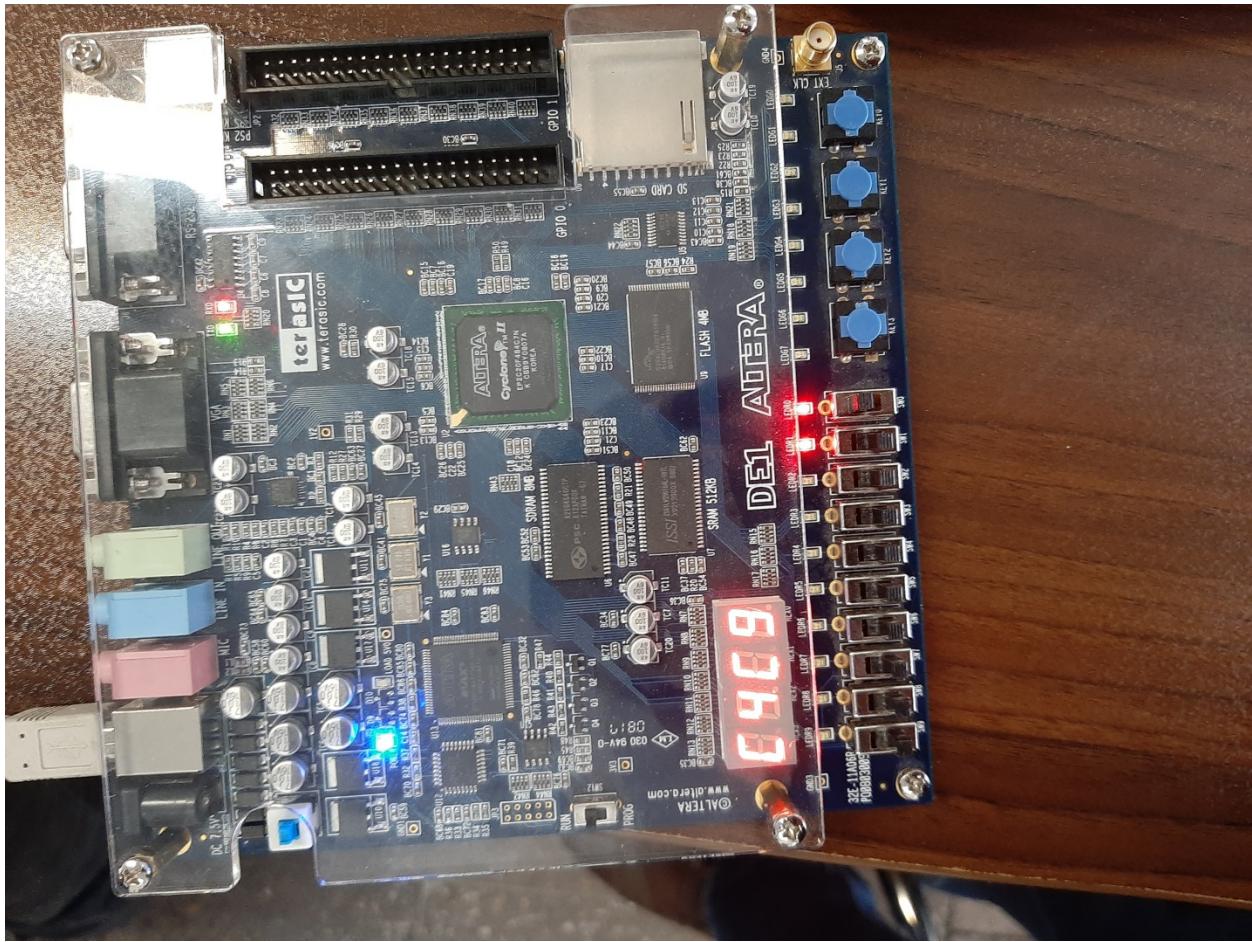


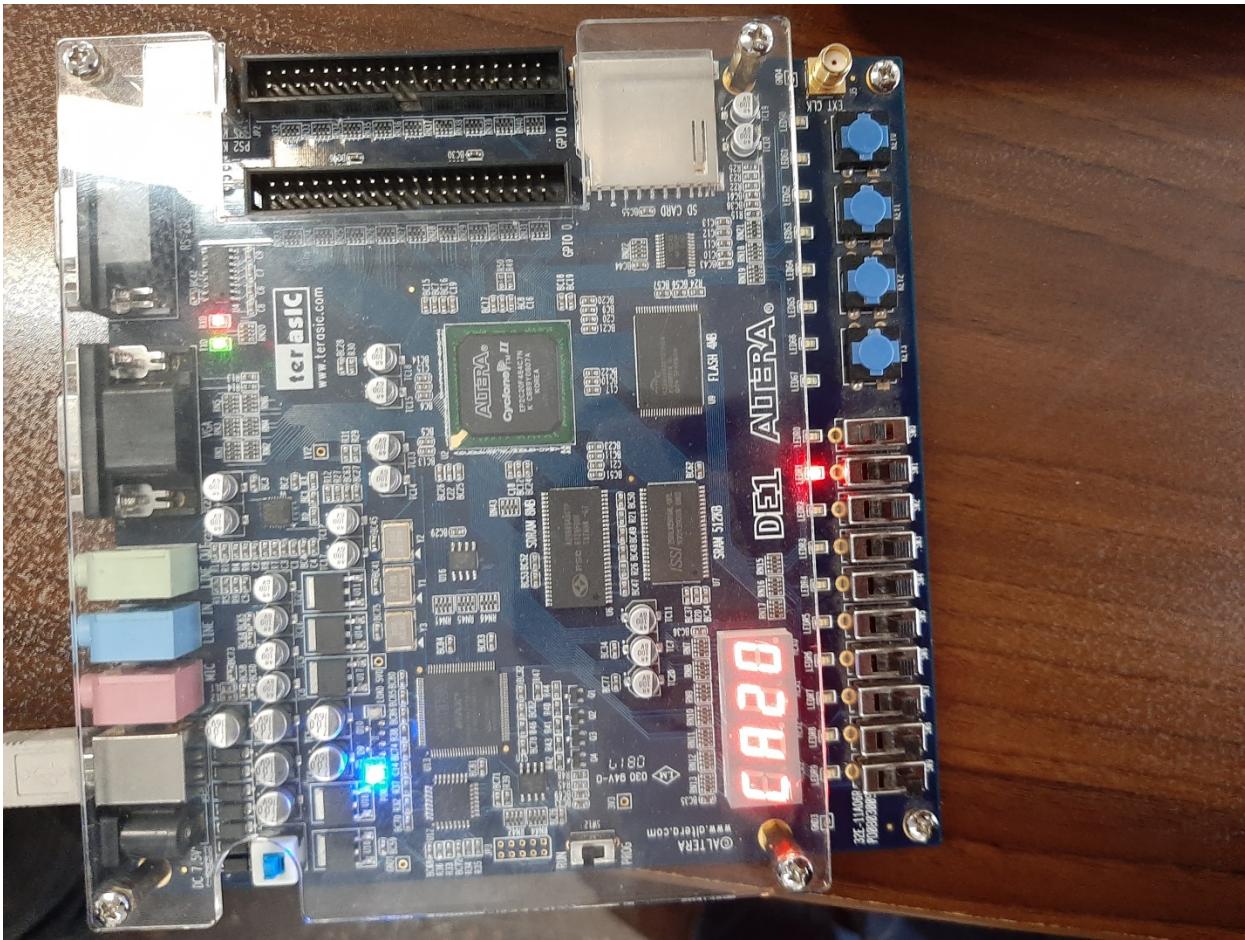


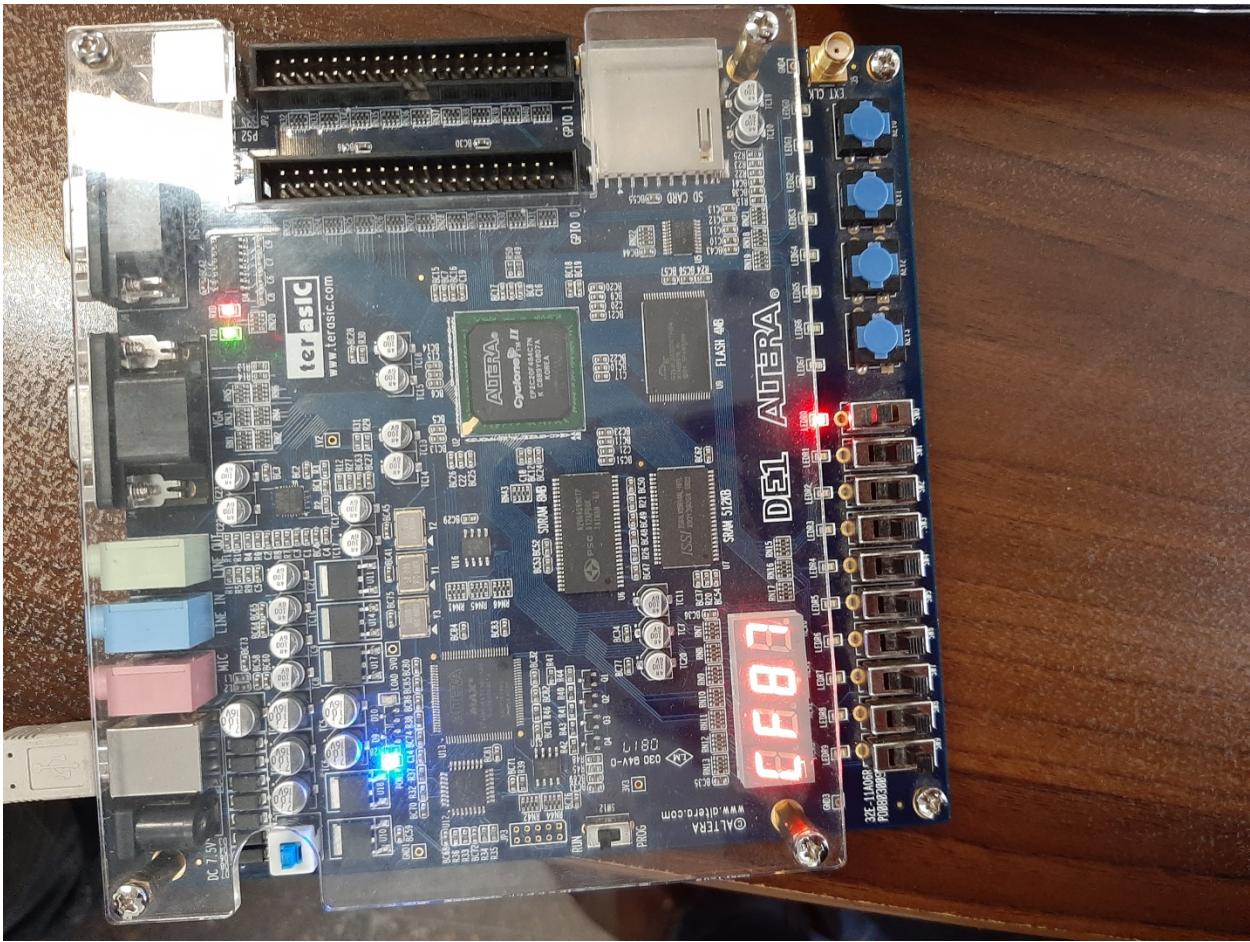


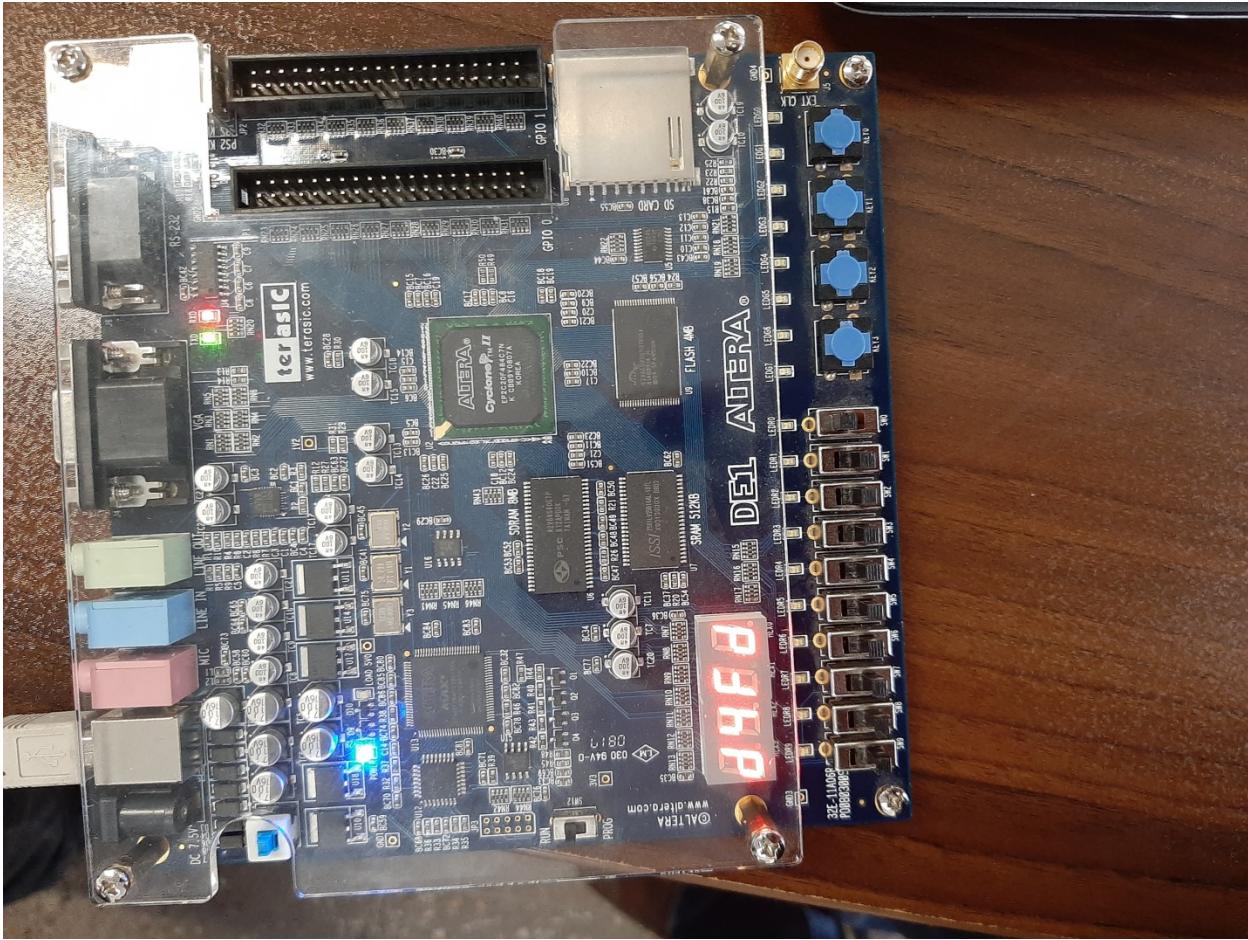












As you see at first there are 10 used places in FIFO, each time we put FPGA's switch up and down, exponential result is shown on seven segment and one used place is thrown away so used places in FIFO decreases. In the end, there is no used place in FIFO.