A PROJECT ON

Creat an IP for 4-bit precision multiplier (Shift AND ADD Method)

by

Ranjan kumar (ROLLNO:-2102102016)

Dr. santosh Kumar visvakarma

(Associate professor, department of EE , IIT INDORE)

DEPARTMENT OF ELECTRICAL ENGINEERING , IIT INDORE

CONTENTS

1. Introduction :-

SHIFT- AND-ADD MULTIPLICATION

- Let us consider two no X=1000 and Y=1001.

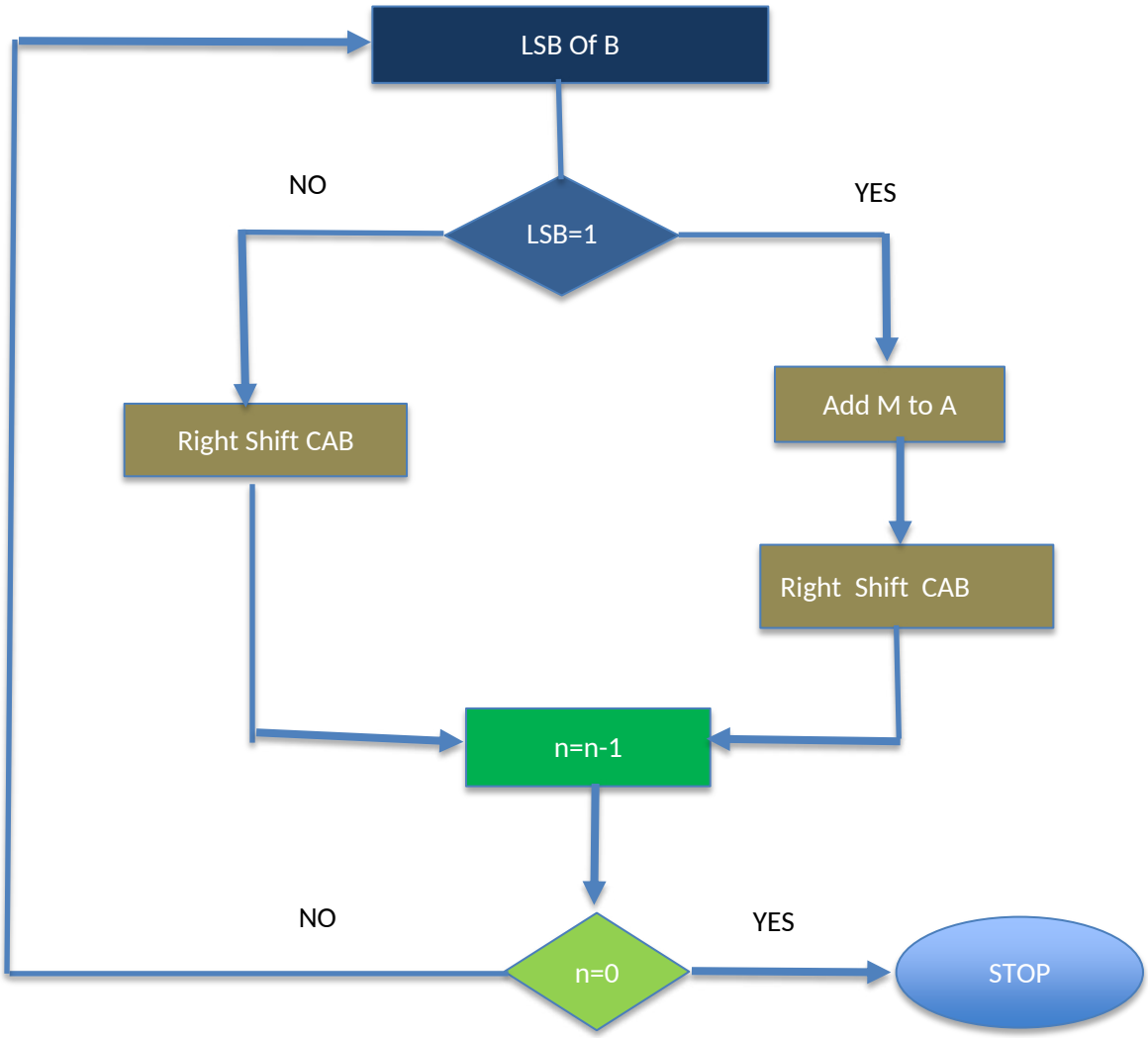
- This method adds the multiplicand X to itself Y times, where Y denotes the multiplier. To multiply two numbers the algorithm is to take the digits of the multiplier one at a time from right to left, multiplying the multiplicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of earlier results.

- In the case of binary multiplication, since the digits are 0 and 1.If the multiplier digit is 1, a copy of the multiplicand (1*multiplicand) is placed in the proper positions; if the multiplier digit is 0, a number of 0 digits(0*multiplicand) are placed in the proper positions.

	X		**1**	**0**	**0**	**0**
	Y		1	0	0	1
	*		1	0	0	0
		0	0	0	0	0
	0	0	0	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0

2. Algorithm for Shift AND Method

In the begin we have taken some registers and initial as belows- M=multiplicand , A=0000, B=Multiplier , n=4 (for 4-bit multiplication), C=0.



3.Verilog code for 4-bit multiplier:-

```
1  `timescale 1ns / 1ps
2  module multiplier4bit(X, Y, Z);
3     input[3:0] X,Y;   // two 4-bit inputs
4     output Z;   //  single 8-bit output
5     // defining some more variablesas per use
6     reg [7:0] Z;
7     reg [3:0] A,B,M;
8     reg[8:0] I;
9     reg C;
10    integer i;  // i as counter for loop
11    always @(X,Y)
12      begin
13        A=4'b0; // initializing A=0000
14        M=X;  // M=multiplicad
15        B=Y;  // B=Multiplier
16        C=1'b0;
17        I={C,A,B}; // combining bits of C,A,B REGISTERS
18        for(i=0; i<4; i=i+1)
19         begin
20           B=I[3:0];
21           A=I[7:4];
22           // If LSB of B IS ZERO THEN if PART EXERCUTES OTHER WISE else PART EXCECUTES
23           if(B[0]==0)
24            I=I>>1;
25           else
26            begin
27           {C,A}=A+M;
28            I={C,A,B};
29            I=I>>1;
30            end
31         end
32           Z=I[7:0]; // final results stores in register Z
33      end
34 endmodule
35
36
```

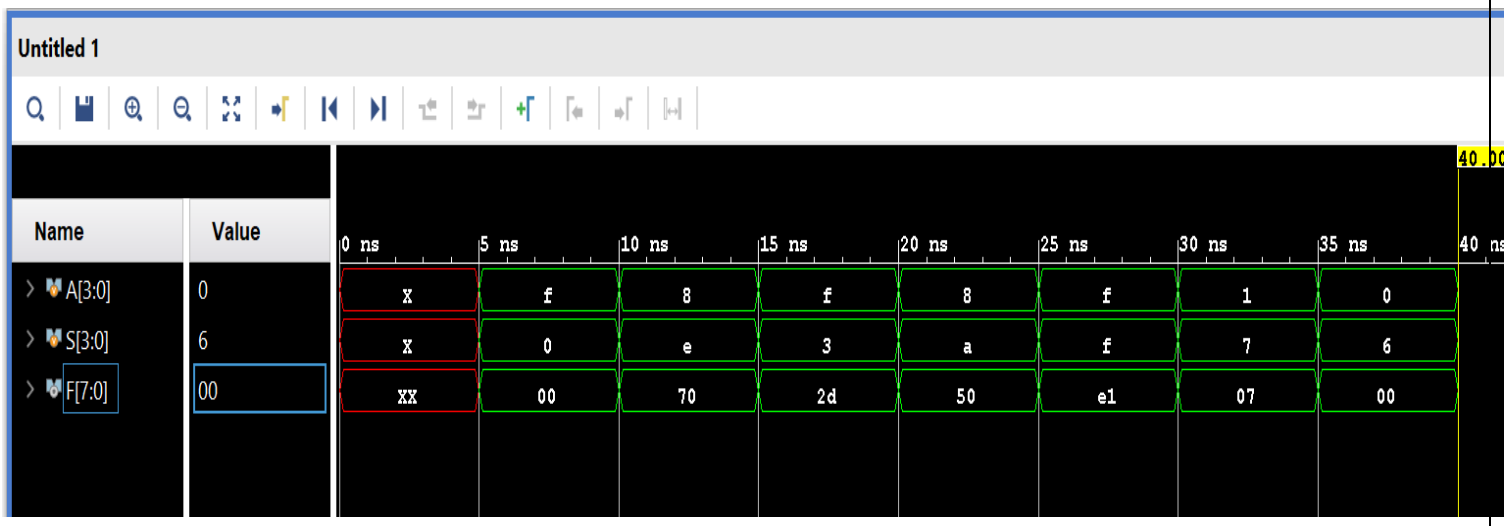
4. Test-bench:-

multiplier_test.v *

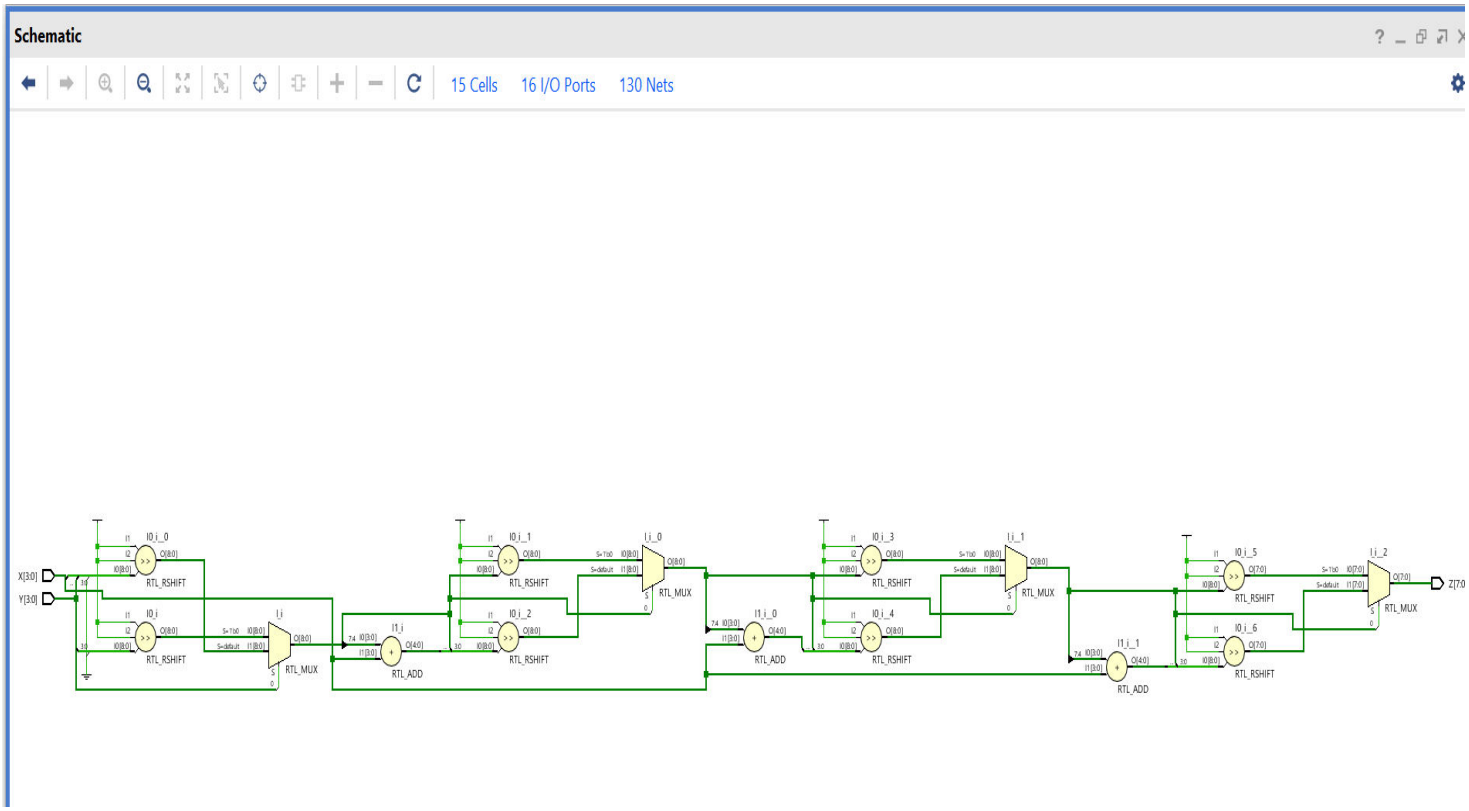
D:/vivado_files/project_12/project_12.srcs/sim_1/new/multiplier_test.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  //////////////////////////////////////////////////////////////////////////////////
4  module multiplier_test;
5  reg [3:0] A;  reg [3:0] S;  wire[7:0] F;
6  multiplier4bit DUT (.X(A), .Y(S), .Z(F));
7  initial
8    begin
9
10     #5 A=4'b1111; S=4'b0000;
11     #5 A=4'b1000; S=4'b1110;
12     #5 A=4'b1111; S=4'b0011;
13     #5 A=4'b1000; S=4'b1010;
14     #5 A=4'b1111; S=4'b1111;
15     #5 A=4'b0001; S=4'b0111;
16     #5 A=4'b0000; S=4'b0110;
17     #5 $finish;
18   end
19 endmodule
```

5.Simulation waveform:-

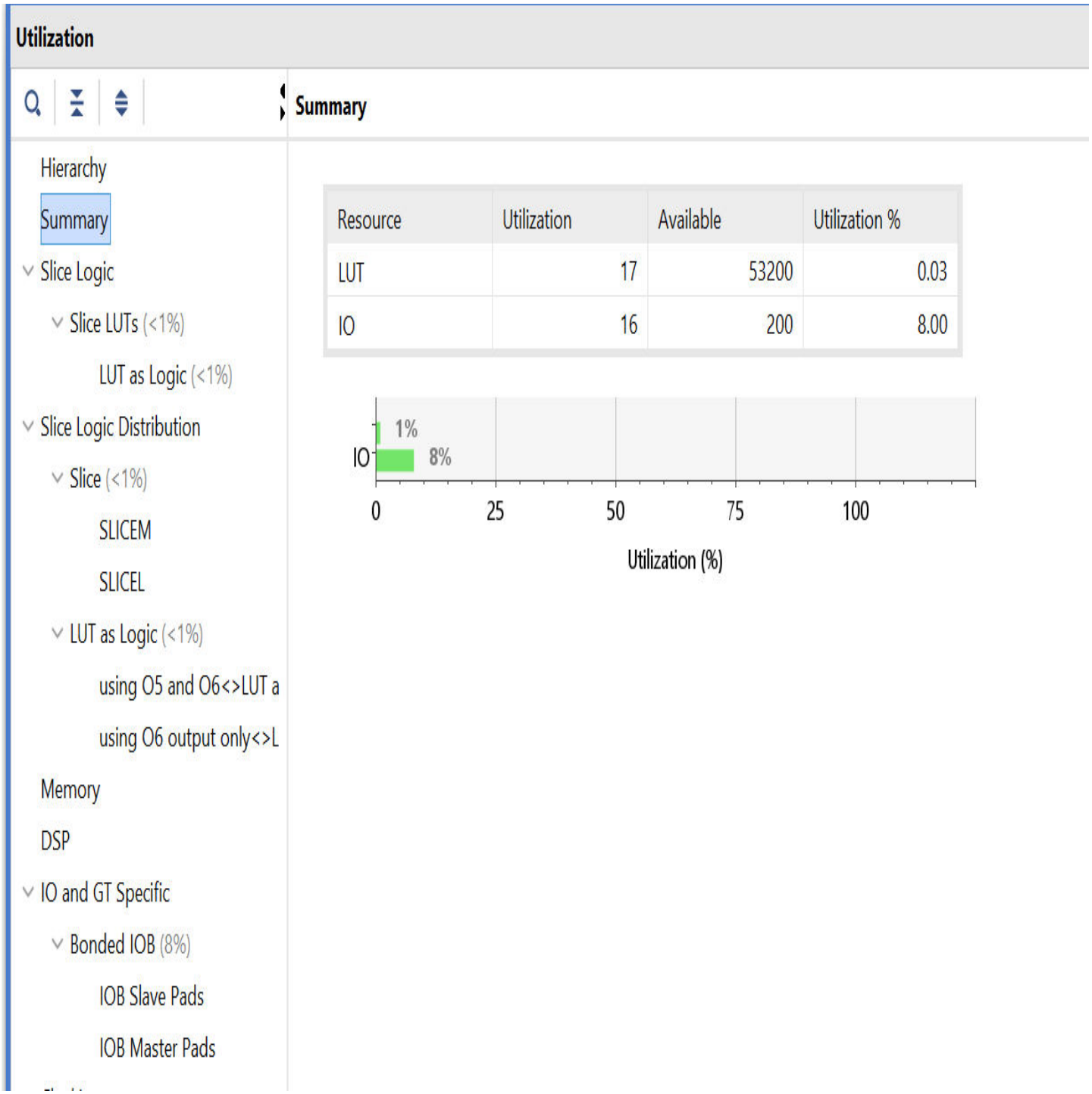


6.RTL Schematic:-

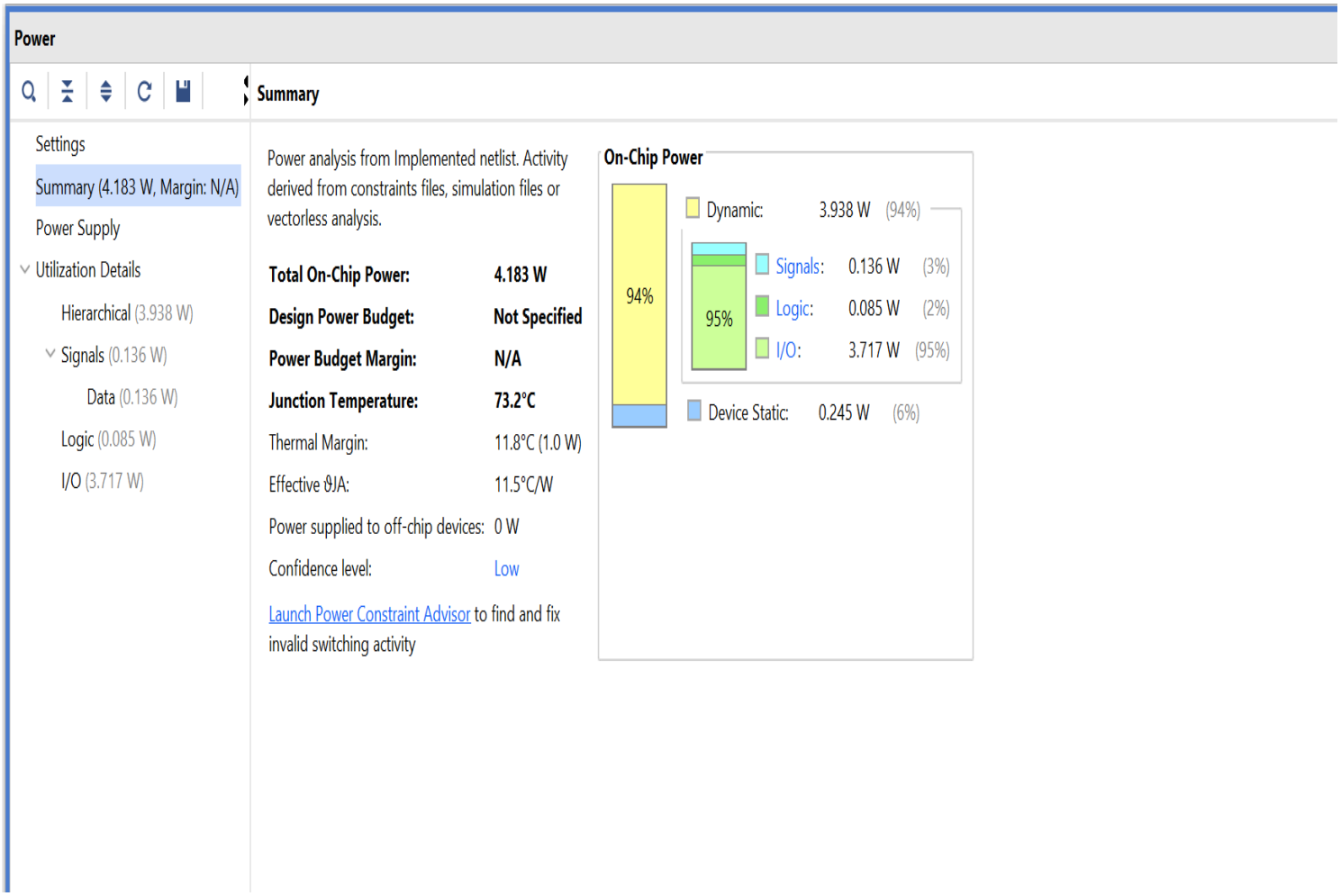


7. synthesis results

i) Resource utilization



ii). POWER ANALYSIS



8. IP Gengerated block Diagram.