VLSI PROJECT

BOOTH MULTIPLIER

ABSTRACT-

In the field of Very Large-Scale Integration (VLSI), multiplication is an important and computationally intensive operation. The efficiency of the multiplier has always been a critical issue and, therefore, the subject of many research projects and papers. Booth algorithm is a crucial improvement in the design of signed binary multiplication and unsigned binary multiplication. There has been progress in partial products reductions, adder structures and complementation methods but still there is scope in modifying the booth algorithm so as to further optimize. The proposed work aims at this. The modified booth multiplier is synthesized and implemented on XILINX software. In this paper, we have implemented a multiplication of higher bits such as 16-bit multiplier and 16-bit multiplicand to produce a 32-bit multiplied output/number.

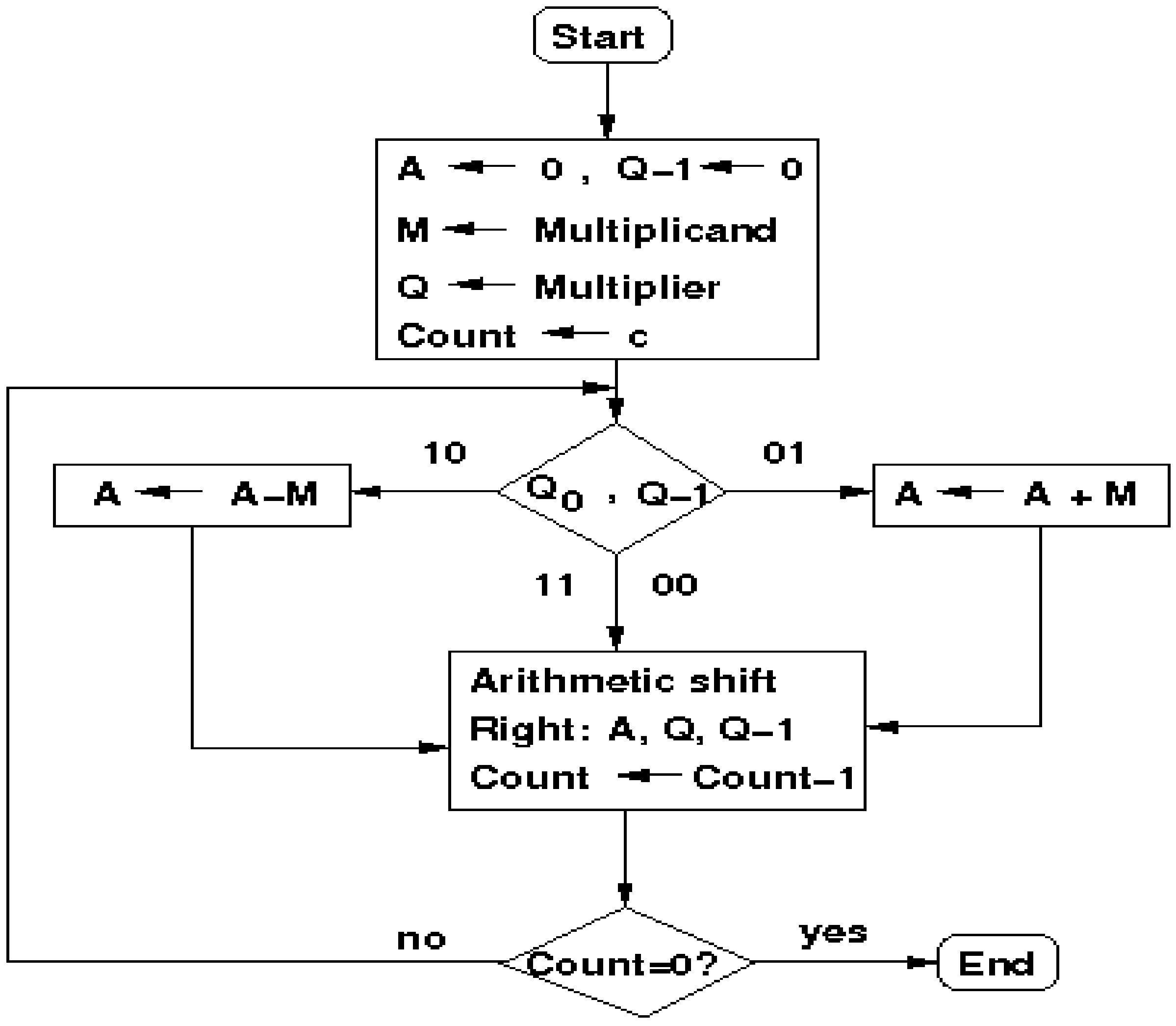
INTRODUCTION-

Very-large-scale integration (**VLSI**) is the process of creating an integrated circuit (IC) by combining hundreds of thousands of transistors or devices into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. Before the introduction of VLSI technology most ICs had a limited set of functions they could perform. An electronic circuit might consist of a CPU, ROM, RAM and other glue logic. VLSI lets IC designers add all of these into one chip. **Xilinx**, Inc. is an American technology company, primarily a supplier of programmable logic devices. It is known for inventing the field-programmable gate array (FPGA) and as the first semiconductor company with a fabless_manufacturing model. **ModelSim** is a multi-language HDL simulation environment by Mentor Graphics, for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger. ModelSim can be used independently, or in conjunction with Intel Quartus Prime, Xilinx ISE or Xilinx Vivado. Simulation is performed using the graphical user interface (GUI), or automatically using scripts.

OBJECTIVE-

To perform 16 bit multiplication of two sixteen bit numbers using booth algorithm in the Xilinx software and to obtain the simulated output (32 bit result) in ModelSim.

BLOCK DIAGRAM-



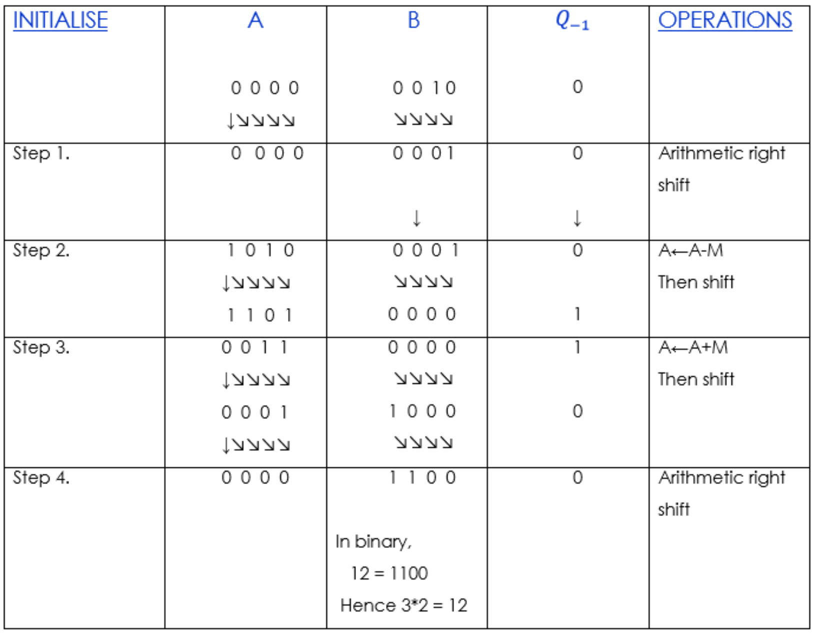
METHODOLOGY-

- Let us consider two numbers that are signed one of which is a multiplicand (M) is and other is a multiplier (Q).
- Let us consider and variable say A to be binary 0 (16 bit 0) and Q_{-1}=0;

- Now the Booth algorithms important feature comes into place.
- If Q_0,Q_{-1} have the same values say 00 or 11 then we have to perform Arithmetic right shift by 1 bit on A and Q(including Q_{-1}) .
- If Q_0,Q_{-1} have different values, say, 01,then we have to perform A🡐A+M and then perform the previous step, that is to perform Arithmetic right shift by 1 bit on A and Q(including Q_{-1}) .
- If Q_0,Q_{-1} have different values, say,10, then we have to perform A🡐A-M and then perform the second step, that is to perform Arithmetic right shift by 1 bit on A and Q(including Q_{-1}) .

Example: Let us consider a 8 bit multiplied output of two 4 bit numbers say 6(0110) and 2(0010).

Booth's algorithm calculates the product in n steps, where n is the number of bits used to represent the numbers.



INITIALISE	A	B	Q_{-1}	OPERATIONS
	0 0 0 0 ↓↘↘↘↘	0 0 1 0 ↘↘↘↘	0	
Step 1.	0 0 0 0	0 0 0 1 ↓	0 ↓	Arithmetic right shift
Step 2.	1 0 1 0 ↓↘↘↘↘ 1 1 0 1	0 0 0 1 ↘↘↘↘ 0 0 0 0	0 1	A←A-M Then shift
Step 3.	0 0 1 1 ↓↘↘↘↘ 0 0 0 1 ↓↘↘↘↘	0 0 0 0 ↘↘↘↘ 1 0 0 0 ↘↘↘↘	1 0	A←A+M Then shift
Step 4.	0 0 0 0	1 1 0 0 In binary, 12 = 1100 Hence 3*2 = 12	0	Arithmetic right shift

DRAWBACKS-

The important drawback here is the complexity in Program code of 16 bit multiplier and due to its complexity it may cause unnecessary lag in the execution of the program or during simulation the experiment in ModelSim.

PROGRAM CODE-

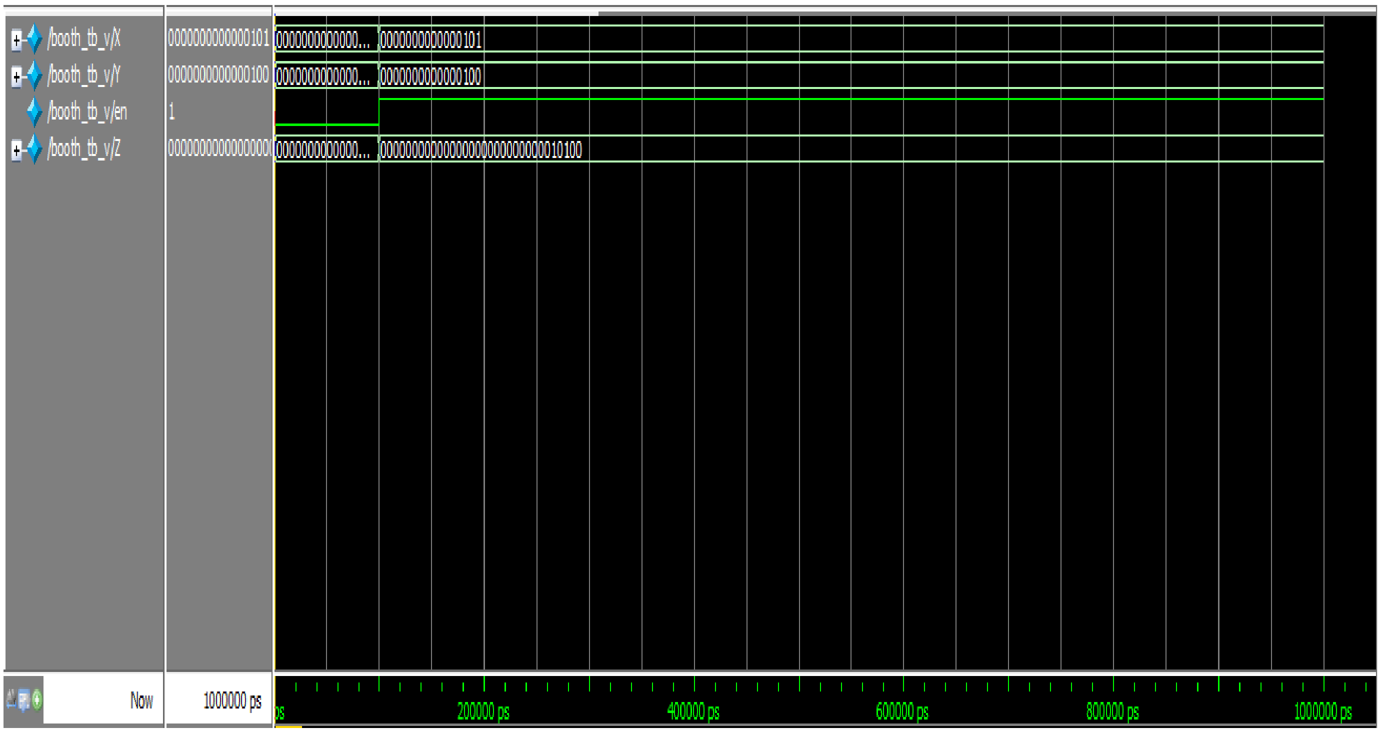
```
module booth(X, Y, Z,en);

input signed [15:0] X, Y;

input en;

output signed [31:0] Z;

reg signed [31:0] Z;

reg [1:0] temp;

integer i;

reg E1;

reg [15:0] Y1;

always @ (X, Y,en)

begin

Z = 32'd0;

E1 = 1'd0;

Y1 = - Y;

Z[15:0]=X;

for (i = 0; i < 16; i = i + 1)

begin

temp = {X[i], E1};

case (temp)

2'd2 : Z [31 : 16] = Z [31 : 16] + Y1;

2'd1 : Z [31 : 16] = Z [31 : 16] + Y;
```

```
default : begin end

endcase

Z = Z >> 1;

Z[31] = Z[30];

E1 = X[i];

end

end

endmodule
```

<u>TEST BENCH</u>-

```
module booth_tb_v;



        // Inputs

        reg [15:0] X;

        reg [15:0] Y;

        reg en;



        // Outputs

        wire [31:0] Z;



        // Instantiate the Unit Under Test (UUT)

        booth uut (

                .X(X),

                .Y(Y),
		.Z(Z),

		.en(en)

	);


	initial begin

		// Initialize Inputs

		X = 0;

		Y = 0;

		en = 0;


		// Wait 100 ns for global reset to finish

		#100;X = 16'b0000000000000101;

 Y = 16'b0000000000000100;

en = 1;

		// Add stimulus here

end

endmodule
```

SIMULATION RESULT-



CONCLUSION-

Thus we have performed the multiplication of two 16 bit numbers to obtain a 32 bit output using booth algorithm in Xilinx software and obtained the simulation results in ModelSim and verified the results.

REFERENCES-

https://en.wikipedia.org

“Verilog HDL Guide to Digital Design and synthesis “- SAMIR PALNITKAR.

“BASIC VLSI DESIGN”- DOUGHLAS A.PUCKNELL.