DDCO Mini Project ISA Submission

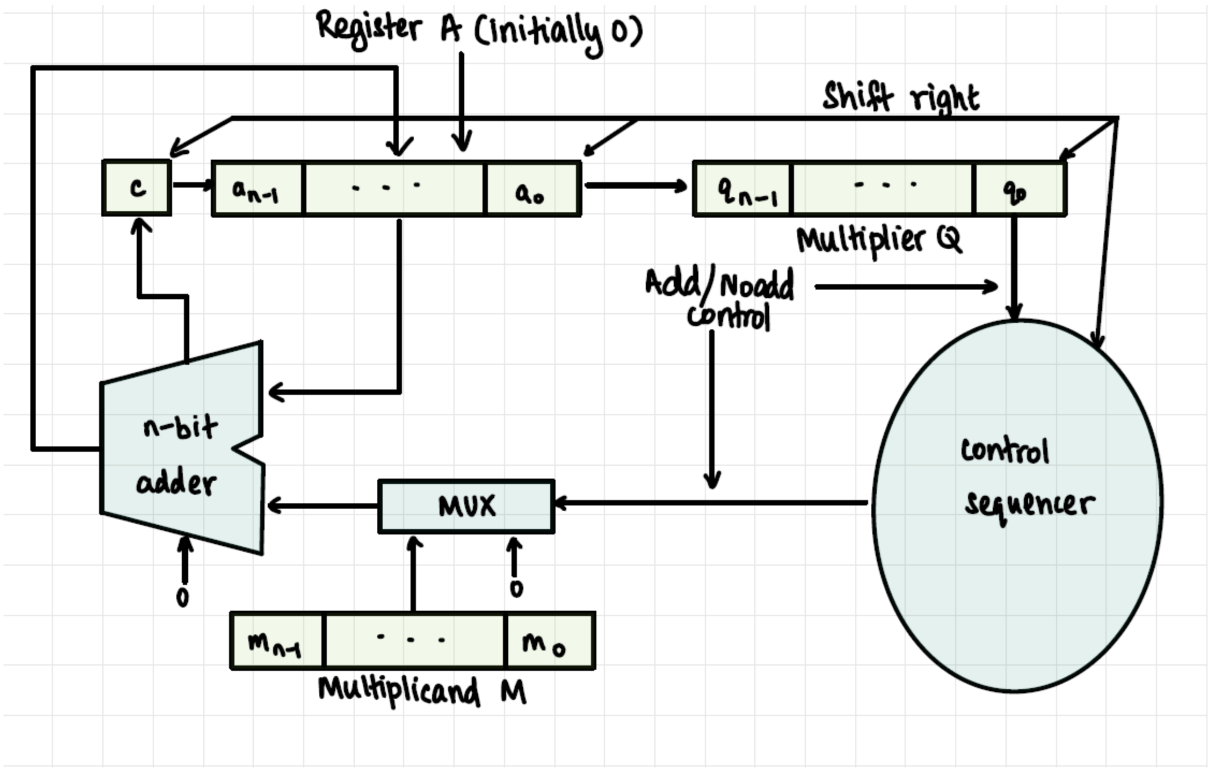
Project Title: Sequential Multiplier

Section : I

Batch Details

Student Name	SRN
Yashas L S	PES1UG19CS590
Yashaswini Nagarajan	PES1UG19CS591
Yashi Chawla	PES1UG19CS592

Circuit Diagram:



Algorithm (if any):

If any bit in the multiplier (b) is 0 then the multiplicand (a) is added with zero. An adder is used which is of the same length as of the operands. Output of the adder and the multiplier is augmented in a register bank. The operand “b” is shifted to the right by 1 bit for each iteration. The 0th bit of “B” is then multiplied by all the bits of “A” to find the partial product. This partial product is shifted to left based on the iteration count and added to the previous calculated sum. This process is repeated 7 more times and the final sum is passed on to the output registers.

Implementation (Code):

mult.v

```
`timescale 1ns/1ns

module mult_8(clk,start,reset,a,b,op);

input [7:0] a,b;

input clk,start,reset;

output reg [15:0] op;


reg [7:0]a1,b1;

reg [16:0]sum;

wire temp;

reg [3:0]i;

assign temp = b1[0];

wire [15:0]temp2;

assign temp2 =
{a1[7]&temp,a1[6]&temp,a1[5]&temp,a1[4]&temp,a1[3]&temp,a1[2]&temp,a1[1]&temp,a1[0]&temp}<<i;

always @(posedge clk)begin

        if (reset) begin
        a1 <= a;
        b1 <= b;
        sum <= 0;
        i <= 0;
        end else begin
                sum <= sum + temp2;
                b1 <= b1 >>1;
                i <= i+1;
                if(i>=8)begin
                        op <=sum;
                end
        end
end
endmodule
```

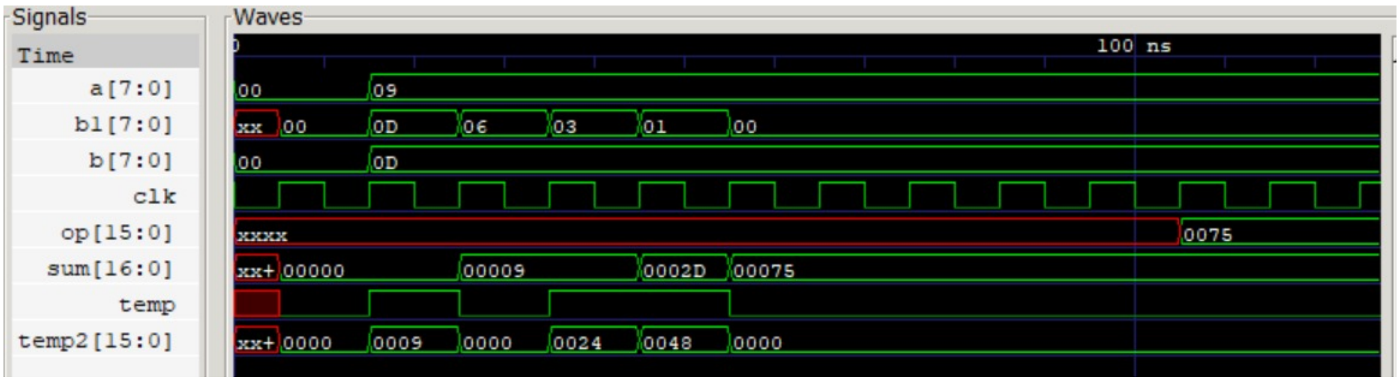
mult_tb.v

```
`timescale 1ns/1ns


module seq_mul_tb;
   reg reset;
        // Inputs
        reg clk;
        reg start;
        reg [7:0] a;
        reg [7:0] b;
```

```
wire [15:0] op;
     mult_8 uut (
             .clk(clk),
             .start(start),
             .reset(reset),
             .a(a),
             .b(b),
             .op(op)
     );
initial begin $dumpfile("seq_mul_tb.vcd"); $dumpvars(0,seq_mul_tb); end
initial begin reset = 1'b1; #25 reset = 1'b0; end
     initial clk = 1'b0;
     always #5 clk = ~clk;
     initial begin
             // Initialize Inputs
             start = 0;
             a = 0;
             b = 0;

             // Wait 100 ns for global reset to finish
             #15;
   start = 1;
             a = 8'b1001;
```

```
        b = 8'b1101;

        #15;

        start = 0;


        #1000 $finish;

        // Add stimulus here


    end


endmodule
```

Output:


Signals
Waves
Time
100 ns
a[7:0]
b1[7:0]
b[7:0]
clk
op[15:0]
sum[16:0]
temp
temp2[15:0]
00
09
xx
00
0D
06
03
01
00
00
0D
xxxx
0075
xx+
00000
00009
0002D
00075
xx+
0000
0009
0000
0024
0048
0000