EE2016 : Microprocessor Theory and Lab

Lab Report # 1

Implementation and Performance Comparison of 4 bit Serial-Parallel Multiplier with Booth's Algorithm in FPGA (Xilinx's Spartan 3E Board)

Mandla Siva Manoj, EE21B083
Ajoe George, EE21B009
Group 3

October 1, 2022

Contents

List of Figures

1 Aim of the Experiment

In this experiment we have to multiply two numbers using two different methods.

1. Using 4-bit Serial-Parallel Multiplier
2. Using Booth's Algorithm for 4-bit numbers

The aim of this experiment is to

- Familiarise us with Verilog, which allows us to model an electrical system.
- To demonstrate the working by writing a testbench code
- Compare between the two methods in terms of clock cycles.

2 Code

```
module mult1(
    input [3:0] A,
    input [3:0] B,
    output reg [7:0] O
    );
always @(A or B)
begin
        O = 0;

        if (B[0]==1'b1)
        O = O+(A<<0);

        if (B[1]==1'b1)
        O = O+(A<<1);

        if (B[2]==1'b1)
        O = O+(A<<2);

        if (B[3]==1'b1)
        O = O+(A<<3);

end
endmodule
```

Listing 1: Verilog Code for Serial Parallel Multiplier

```
// Inputs
        reg [3:0] A;
        reg [3:0] B;

        // Outputs
        wire [7:0] O;

        // Instantiate the Unit Under Test (UUT)
        mult1 uut (
                .A(A),
                .B(B),
                .O(O)
        );

        initial begin
                // Initialize Inputs
                A = 0;
                B = 0;

                // Wait 100 ns for global reset to finish
                #100;

                // Add stimulus here

        end

endmodule
```

Listing 2: Testbench of Serial Parallel Multiplier

```
module BOOTHS_MULTIPLIER(
        output [7:0] prod,
        output busy,
        input [3:0] mc,
        input [3:0] mp,
        input clk,
        input start
);

reg [3:0] A, Q, M; // all are 4 bit registers
reg Q_1;

reg [2:0] count;
wire [3:0] sum, difference;

always @(posedge clk)
begin
if (start)
        begin
        A <= 1'b0;
        M <= mc;
        Q <= mp;
        Q_1 <= 1'b0;
        count <= 3'b0;
end
else
        begin
        case ({Q[0], Q_1})
        2'b0_1 : {A, Q, Q_1} <= {sum[3], sum, Q};
        2'b1_0 : {A, Q, Q_1} <= {difference[3], difference, Q};
        default: {A, Q, Q_1} <= {A[3], A, Q};
        endcase
        count <= count + 1'b1;
end
end

alu adder(sum, A, M, 0); // adder
alu subtracter(difference, A,~M, 1); //subtracter using 2's compliment
assign prod = {A, Q};
assign busy = (count < 5);
endmodule


output [3:0] out;
input [3:0] a;
input [3:0] b;
input cin;
assign out = a + b + cin;
endmodule
```

Listing 3: Verilog Code for Booth's Algorithm

```
module BOOTHS_MULTIPLIER_TB;
// Inputs
reg [3:0] mc;
reg [3:0] mp;
reg clk;
reg start;
// Outputs
wire [7:0] prod;
wire busy;
// Instantiate the Unit Under Test (UUT)
BOOTHS_MULTIPLIER uut (
        .prod(prod),
        .busy(busy),
        .mc(mc),
        .mp(mp),
        .clk(clk),
        .start(start)
);

initial begin
// Initialize Inputs
mc = 4'b0011;
mp = 4'b0010;
clk = 1;
start = 1;
#10 clk = ~clk;
```

```
#10 clk = ~clk;
start = 0;
#10 clk = ~clk;
#10 clk = ~clk;
#10 clk = ~clk;
#10 clk = ~clk;
#10 clk = ~clk;
#10 clk = ~clk;
#10 clk = ~clk;
#10 clk = ~clk;


$finish;
end
initial begin
        $dumpfile("BOOTHS.vcd");
        $dumpvars(0,BOOTHS_MULTIPLIER_TB);
end

endmodule
```

Listing 4: Testbench for Booth's Algorithm

3 Outputs

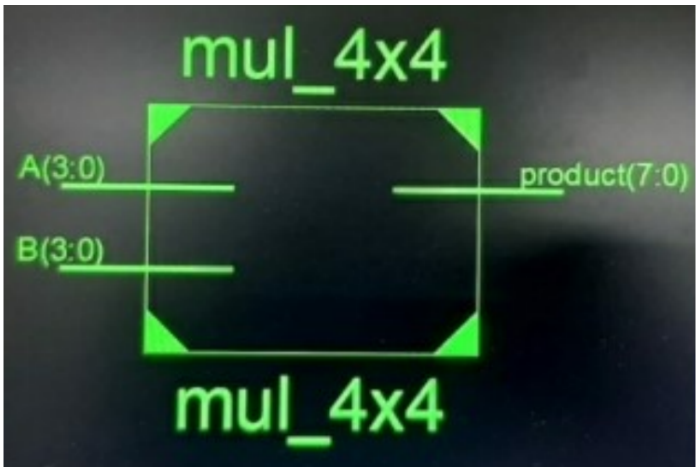

Figure 1: Synthesis Diagram of Serial Parallel Multiplier

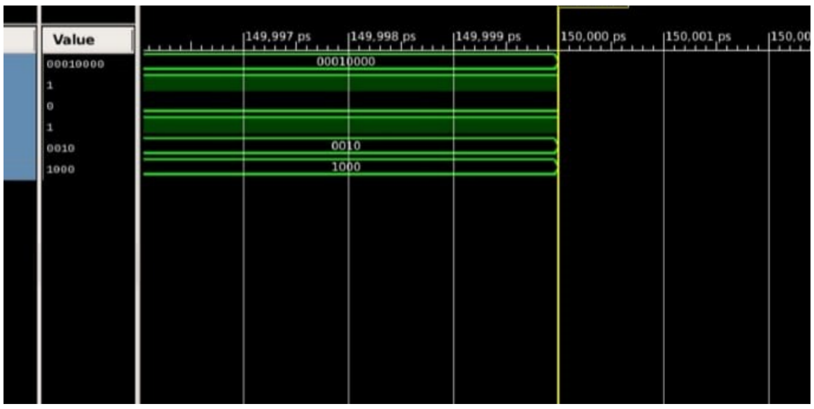

Figure 2: Output of Serial Parallel Multiplier

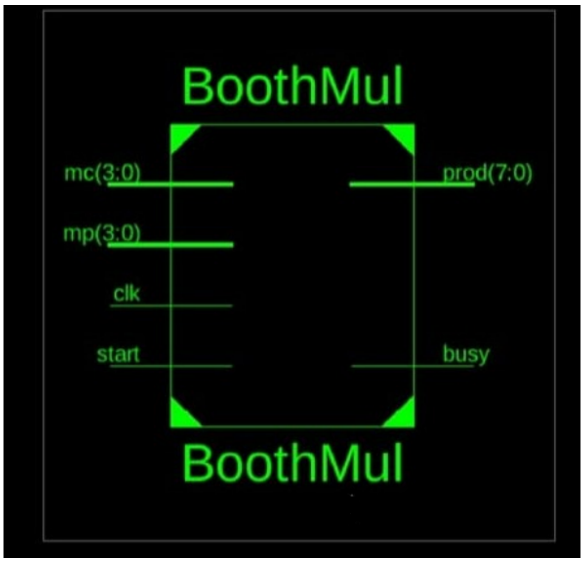

Figure 3: Synthesis Diagram of Booth's Multiplier

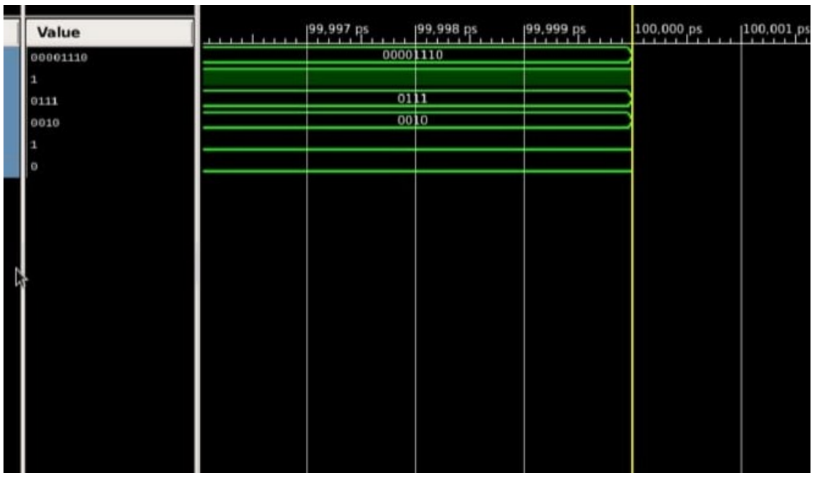

Figure 4: Output of Booth's Multiplier

4 Explanation

4.1 Serial Parallel Multiplier

In Serial Parallel Multiplier, if the multiplicand is 1, we sequentially go through each bit and shift the result by one bit.

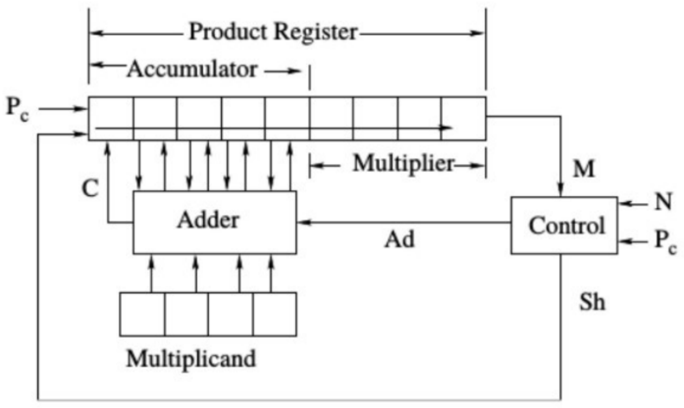

Figure 5: Diagram of Serial Parallel Multiplier

4.2 Booth's Multiplier

The following flowchart illustrates the Booth's Multiplier algorithm:

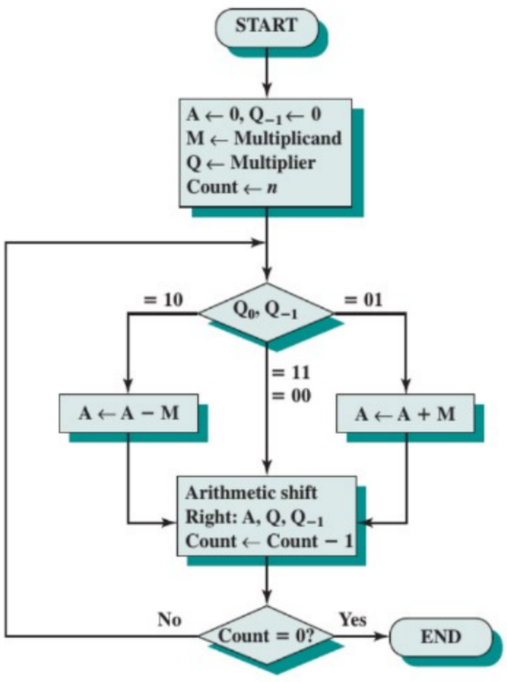

Figure 6: Booths Multiplier Flowchart

5 Learning Outcomes

By doing this experiment we learnt:

1. How to use Verilog to implement Serial Parallel Multiplier and the Booth's algorithm
2. How to set up a test bench and perform two-number multiplication
3. In terms of clock cycles, Booth's algorithm is significantly faster than serial parallel multiplier.
    - Booths algorithm performs signed bit multiplication correctly, whereas serial parallel multiplier will produce an incorrect result.
    - Theoretically, a serial parallel multiplier has a time complexity of 2n, whereas the Booth's algorithm computes a n bit multiplication in n clock cycles.