## TASK 1 : 8-bit Verilog Code for Booth's Multiplier

```
module multiplier(prod, busy, mc, mp, clk, start);
output [15:0] prod;
output busy;
input [7:0] mc, mp;
input clk, start;
reg [7:0] A, Q, M;
reg Q_1;
reg [3:0] count;

wire [7:0] sum, difference;

always @(posedge clk)
begin
   if (start) begin
      A <= 8'b0;
      M <= mc;
      Q <= mp;
      Q_1 <= 1'b0;
      count <= 4'b0;
   end
   else begin
      case ({Q[0], Q_1})
         2'b0_1 : {A, Q, Q_1} <= {sum[7], sum, Q};
         2'b1_0 : {A, Q, Q_1} <= {difference[7], difference, Q};
         default: {A, Q, Q_1} <= {A[7], A, Q};
      endcase
      count <= count + 1'b1;
   end
end

alu adder (sum, A, M, 1'b0);
alu subtracter (difference, A, ~M, 1'b1);

assign prod = {A, Q};
assign busy = (count < 8);

endmodule

//The following is an alu.
//It is an adder, but capable of subtraction:
//Recall that subtraction means adding the two's complement--
//a - b = a + (-b) = a + (inverted b + 1)
//The 1 will be coming in as cin (carry-in)
module alu(out, a, b, cin);
output [7:0] out;
input [7:0] a;
input [7:0] b;
input cin;

assign out = a + b + cin;

endmodule
```

# **Testbench for Booth's Multiplier**

```
module testbench;

reg clk, start;
reg [7:0] a, b;

wire [15:0] ab;
wire busy;

multiplier multiplier1(ab, busy, a, b, clk, start);

initial begin
clk = 0;
$display("first example: a = 3 b = 17");
a = 3; b = 17; start = 1; #50 start = 0;
#80 $display("first example done");
$display("second example: a = 7 b = 7");
a = 7; b = 7; start = 1; #50 start = 0;
#80 $display("second example done");
$finish;
end

always #5 clk = !clk;

always @(posedge clk) $strobe("ab: %d busy: %d at time=%t", ab, busy,
$stime);

endmodule
```