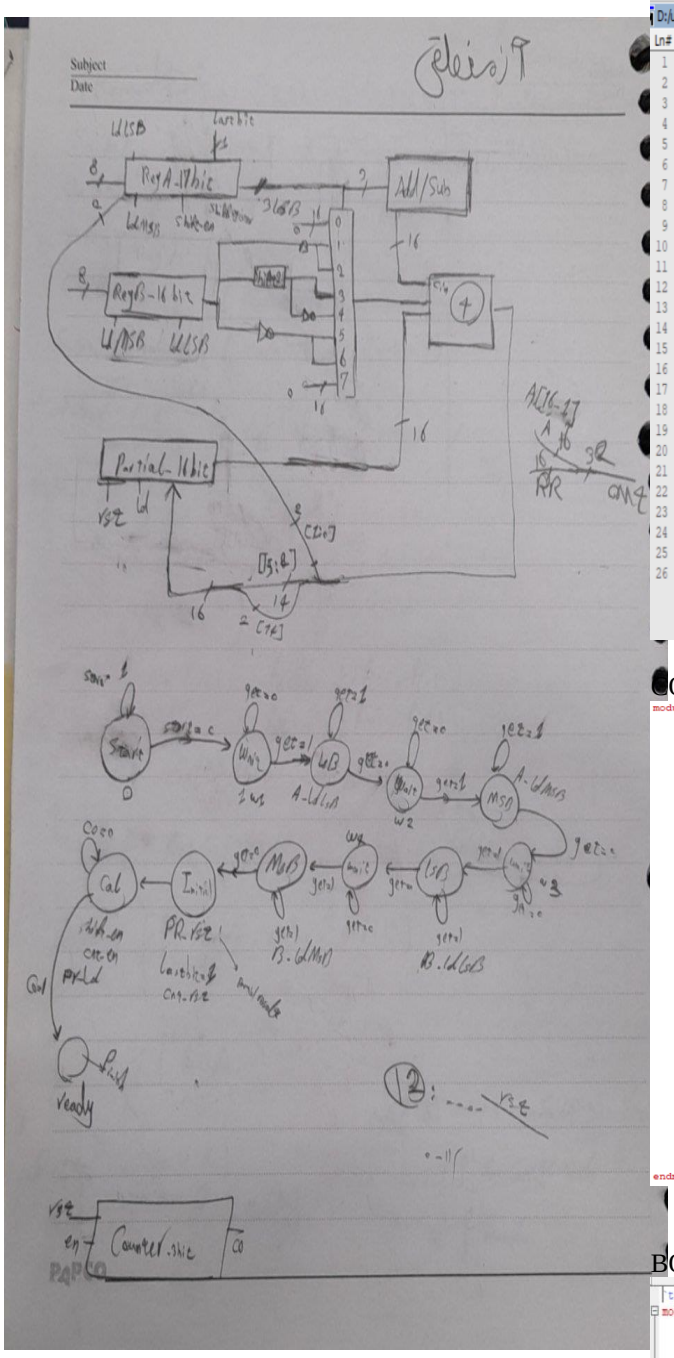
Experiment #2 - FPGA Realization of Radix-4 Multiplier

Sepehr Azardar
810199357

Amirhossein
Kahrobaeian
810199478

1- RTL Design and Simulation

DATA PATH AND CONTROLLER



DATA PATH VERILOG

```
timescale 1ns/1ns
module DP(input clk,pr_rst, rst, shen, ldAm, ldAl, ldBm, ldBl, ldP, lastbit, input [7:0] A, B, output [31:0] o

    wire wm2 ;
    wire [2:0] sel;
    wire [16:0] regA;
    wire [15:0] regb, wml, wp, wadd ;
    assign sel = regA[2:0] ;

    Reg17 RegA( .data(A), .clk(clk), .rst(rst), .ldm(ldAm), .ldl(ldAl), .lastbit(lastbit),
         .shen(shen), .shdata(wadd[1:0]), .w(regA) ) ;

    Reg16 RegB( .data(B), .clk(clk), .rst(rst), .ldm(ldBm), .ldl(ldBl), .w(regb) );

    RegP P( .data({wadd[15], wadd[15], wadd[15:2]}), .clk(clk), .rst(rst|pr_rst), .ld(ldP), .w(wp) );

    MUX8tol ml(.A(16'b 0), .B(regb), .C(regb), .D(regb << 1), .E( ~(regb << 1) ),
        .F(~regb), .G(~regb), .H(16'b 0), .s(sel), .w(wml));

    MUX8tol_1bit m2( .A(1'b0), .B(1'b0), .C(1'b0), .D(1'b0), .E(1'b1),
        .F(1'b1), .G(1'b1), .H(1'b0), .s(sel), .w(wm2));

    AddSub Adder(.A(wml), .B(wp), .ci(wm2), .co(), .w(wadd));
    assign out = {wp, regA[16:1]};

endmodule
```

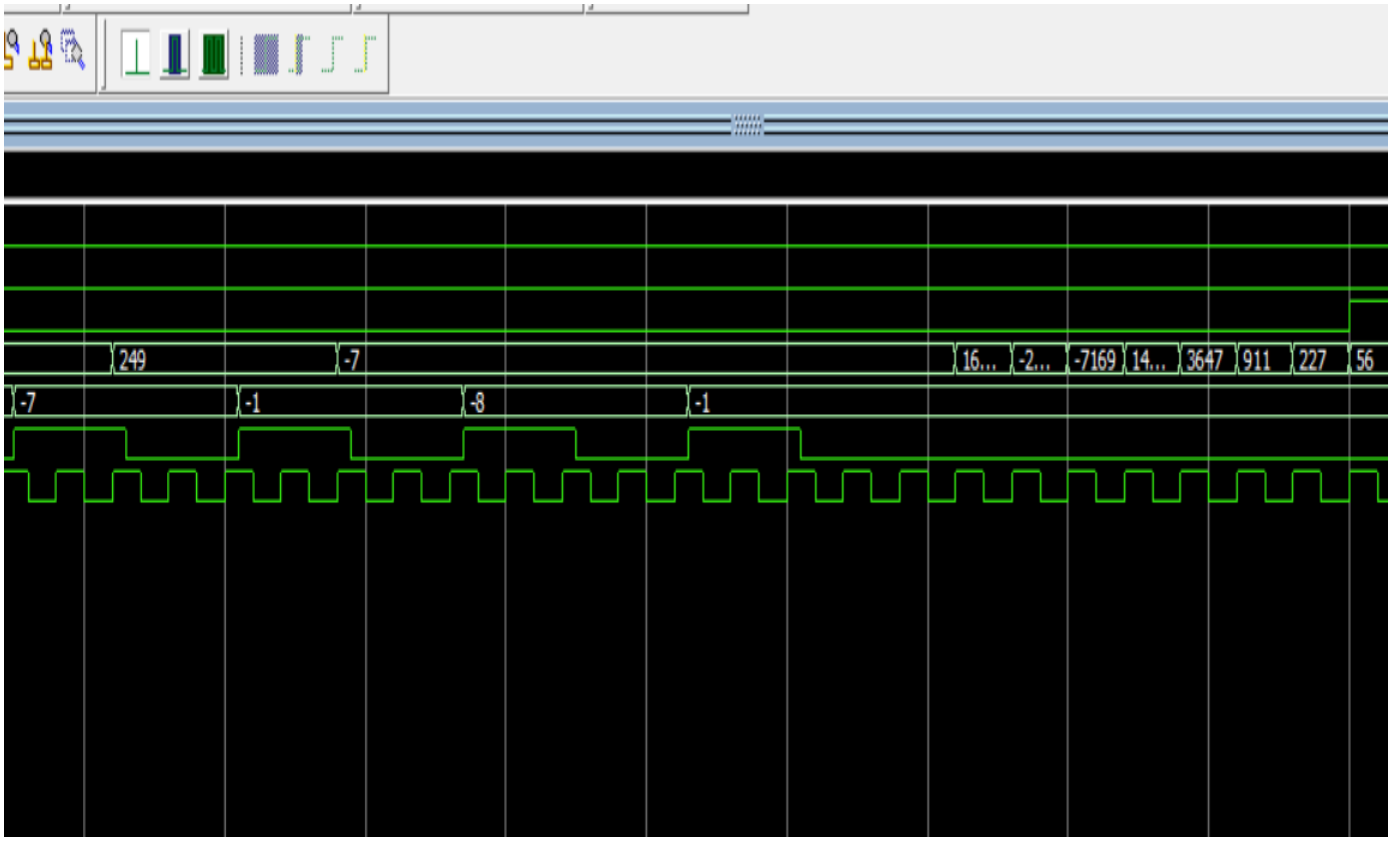
CONTROLLER VEILOG

```
module controller(clk,rst,start,get,ready,A_ldLsb,A_ldMsb,B_ldLsb,B_ldMsb,pr_rst,last_bit,pr_ld,shift_en)
    input clk,rst,start,get;
    output reg ready,A_ldLsb,A_ldMsb,B_ldLsb,B_ldMsb,pr_rst,last_bit,pr_ld,shift_en;
    reg cnt_rst,cnt_en;
    wire cnt_co;
    counter count1(.rst(cnt_rst),.en(cnt_en),.co(cnt_co),.clk(clk));
    reg [3:0] ps=0,ns;
    always @(ps,start,get,rst,cnt_co) begin
        ns = 's;
        {ready,A_ldLsb,A_ldMsb,B_ldLsb,B_ldMsb,pr_rst,last_bit,pr_ld,shift_en,cnt_rst} = 10'b0;
        if(rst) begin
            ps = 's;cnt_rst = 1'b1;
        end
        case (ps)
            `s: ns = (start == 1) ? `s : `wl;
            `wl: ns = (get == 0) ? `wl : `a_lsb;
            `a_lsb : begin ns = (get == 1) ? `a_lsb : `w2 ; A_ldLsb = 1;end
            `w2 : ns = (get == 0) ? `w2 : `a_msb;
            `a_msb : begin ns = (get == 1) ? `a_msb : `w3 ; A_ldMsb = 1;end
            `w3 : ns = (get == 0) ? `w3 : `b_lsb;
            `b_lsb : begin ns = (get == 1) ? `b_lsb : `w4 ; B_ldLsb = 1;end
            `w4 : ns = (get == 0) ? `w4 : `b_msb;
            `b_msb : begin ns = (get == 1) ? `b_msb : `init ; B_ldMsb = 1;end
            `init : begin ns = `cal; pr_rst = 1;last_bit = 1; cnt_rst =1; end
            `cal : begin ns = (cnt_co == 1) ?`finish:`cal; {cnt_en, pr_ld, shift_en} = 3'b111; end
            `finish : begin ns = `s; ready = 1'b1; end
        endcase
    end

    always @(posedge clk,posedge rst) begin
        if(rst)
            ps <= `s;
        else
            ps <= ns;
    end


endmodule
```

BOOTH MULTIPLIER

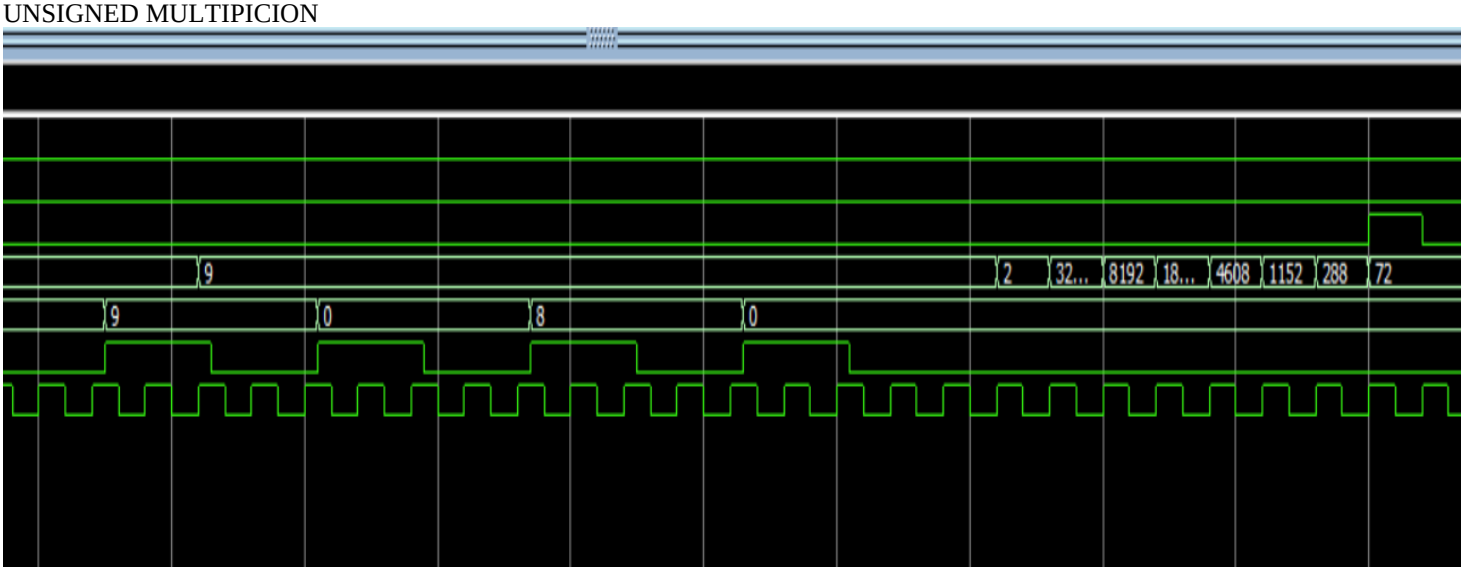
```
timescale 1ns/1ns
module booth(in,start,clk,rst,ready,out28,get);
    input [7:0] in;
    input start,rst,clk,get;
    output ready;
    output [0:27] out28;
    wire [15:0] out;
    wire shen,ldAm,ldAl,ldBm,ldBl,ldp,lastbit,pr_rst;
    wire [7:0] A,B;
    wire [31:0]out32;
    assign out = out32[15:0];

    DP DP1(.clk(clk),.pr_rst(pr_rst),.rst(rst),.shen(shen), .ldAm(ldAm), .ldAl(ldAl),
     .ldBm(ldBm), .ldBl(ldBl), .ldP(ldp), .lastbit(lastbit), .A(in), .B(in),.out(out32));

    controller CU(.clk(clk),.rst(rst),.start(start),.get(get),.ready(ready),.A_ldLsb(ldAl),.A_ldMsb(ldAm),.B_ldLsb(ldBl),.B_ldMs
        .last_bit(lastbit),.pr_ld(ldp),.shift_en(shen));
    wire [0:6] d1,d2,d3,d4;
    Hexdisplay h1( out[15:12], d1);
    Hexdisplay h2( out[11:8] , d2);
    Hexdisplay h3( out[7:4] , d3);
    Hexdisplay h4( out[3:0] , d4);
    assign out28 = {d1,d2,d3,d4};
endmodule
```

SIGNED MULTIPICION

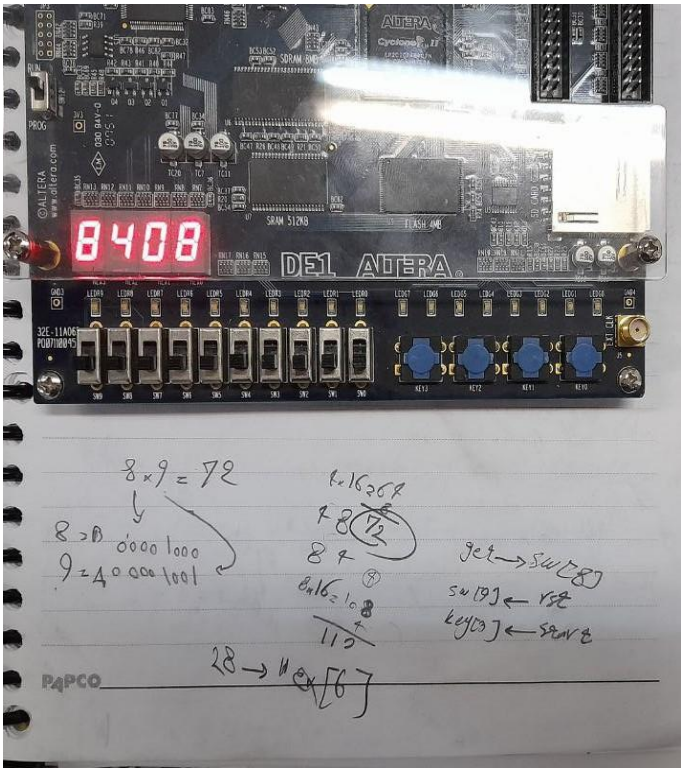


as you can see we first load lsb and after that msb for each oprand. the 8 lower bits is negative -7;
and the upper 8 bit is 8'b11111111;
and the same goes for the second oprand;
finally the result is -7*-8 = 56 which is correct;

UNSIGNED MULTIPICION



9 * 8 = 72 AS THE final result in test bench; both oprand their msb is 8'b0;



out28[16]	Output	PIN_H5	3.3-V L...efault)
out28[15]	Output	PIN_H6	3.3-V L...efault)
out28[14]	Output	PIN_E1	3.3-V L...efault)
out28[13]	Output	PIN_D3	3.3-V L...efault)
out28[12]	Output	PIN_E4	3.3-V L...efault)
out28[11]	Output	PIN_E3	3.3-V L...efault)
out28[10]	Output	PIN_C1	3.3-V L...efault)
out28[9]	Output	PIN_C2	3.3-V L...efault)
out28[8]	Output	PIN_G6	3.3-V L...efault)
out28[7]	Output	PIN_G5	3.3-V L...efault)
out28[6]	Output	PIN_D4	3.3-V L...efault)
out28[5]	Output	PIN_F3	3.3-V L...efault)
out28[4]	Output	PIN_L8	3.3-V L...efault)
out28[3]	Output	PIN_J4	3.3-V L...efault)
out28[2]	Output	PIN_D6	3.3-V L...efault)
out28[1]	Output	PIN_D5	3.3-V L...efault)
out28[0]	Output	PIN_F4	3.3-V L...efault)
ready	Output	PIN_R17	3.3-V L...efault)
rst	Input	PIN_L2	3.3-V L...efault)
start	Input	PIN_T21	3.3-V L...efault)