**DeBAM: Decoder Based Approximate Multiplier for Low Power Applications**

**Design Code:**

module debam(input [7:0] a,b, output reg [15:0] q);

reg [7:0]temp[4:0];

reg [15:0] add[4:0];

    always@(\*) begin

     temp[0] <= a[1] ? (a[0] ? ((b << 1'b1)|b) : (b << 1'b1)) : (a[0] ? b : 8'd0);

     temp[1] <= a[3] ? (a[2] ? ((b << 1'b1)|b) : (b << 1'b1)) : (a[2] ? b : 8'd0);

     temp[2] <= a[5] ? (a[4] ? ((b << 1'b1)|b) : (b << 1'b1)) : (a[4] ? b : 8'd0);

     temp[3] <= a[6] ? b : 0;

     temp[4] <= a[7] ? b : 0;

     add[0] <= temp[0];

     add[1] <= {temp[1],2'd0};

     add[2] <= {temp[2],4'd0};

     add[3] <= {temp[3],6'd0};

     add[4] <= {temp[4],7'd0};

     q <= add[0] + add[1] + add[2] + add[3] + add[4];

    end

endmodule

**Test Bench Code:**

module tb();

reg [7:0] a,b; wire [15:0] q;

integer av\_per,count,percent,eff;

reg [15:0]s,p;

debam uut(a,b,q);

initial begin

    count = 0; percent = 0; av\_per = 0; eff = 0;

    repeat(100) begin

        {a,b} = $random;

        #1;

        s = a\*b;

        p = q;

        if(p == s) av\_per = 100;

        else if (s == 0) av\_per = 0;

        else av\_per = p\*100/s;

        count = count + 1;

        percent = percent + av\_per;

        #9;

    end

    eff = percent/count;

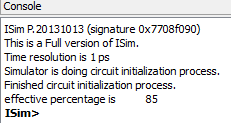
    $display("effective percentage is %d",eff);

end

endmodule

**Output:**

The output is attained with error factor of just 15 percent.

****