[과제 4] 4bit x 4bit = 8bit Multiplier(Signed Magnitude) 설계

```
//과제 4
//이름: 이창민
//학번:2019043890
//융합전자공학부
```

```
// Code your design here
module mul8_sign (

  input signed [3:0] a, b,
  input clk, rstn, start,
  output reg [7:0] result,
  output reg done
);

  localparam IDLE = 3'b000, //0
             START = 3'b001, //1
             LSB = 3'b010,   //2
             ADD = 3'b011,   //3
             SHIFT = 3'b100, //4
             DONE = 3'b101;  //5

  reg [7:0] r_multiplicand, r_product;
  reg [3:0] r_multiplier;
  reg [2:0] r_state,next_state;
  reg [1:0] r_count;
  reg sign;

always @(posedge clk, negedge rstn) begin
    if (!rstn) r_state <= IDLE;
    else r_state<=next_state;
end
always @(*) begin
  case(r_state)
    IDLE:
      begin
        if (start) begin
            next_state = START;
        end
        else begin
            next_state = IDLE;
        end
      end
    START:
      begin
        next_state=LSB;
```

```
      end
    LSB:
      begin
        if(r_multiplier[0]) next_state = ADD;
        else next_state = SHIFT;
      end
    ADD:
      begin
        next_state = SHIFT;
      end
    SHIFT:
      begin
        if(r_count != 0)next_state=LSB;
        else next_state = DONE;
      end
    DONE:
      begin
        next_state= IDLE;
      end
    default:
      begin
        next_state= IDLE;
      end
    endcase
end
always @(posedge clk, negedge rstn) begin
    if (!rstn) begin
      r_multiplicand <= 0;
      r_multiplier <= 0;
      r_product <= 0;
      r_count <= 4;
      result <= 0;
      done <= 0;
    end
    else begin
      case (next_state)
        IDLE:
          begin
            r_multiplicand <= 0;
            r_multiplier <= 0;
            r_product <= 0;
            r_count <= 4;
            result <= 0;
            done <= 0;
          end
        START:
        begin
```

```
            if (a[3] == 1) begin //음수인지 체크
                r_multiplicand <= ~a + 1;
            end
            else begin
                r_multiplicand <= {4'b0000, a};
            end

            if (b[3] == 1) begin //음수인지 체크
                r_multiplier <= ~b + 1;
            end
            else begin
                r_multiplier = b;
            end
            sign <= a[3] ^ b[3]; //결과값 부호 1 이면 음수 0 이면 양수
            r_product <= 0;
            r_count <= 4;
            result <= 0;
            done <= 0;
        end
        LSB:begin
            r_multiplicand <= r_multiplicand; //순차회로에서 자기자신을
기억해라
            r_multiplier <= r_multiplier;
            r_count<=r_count -1;

            result <= 0;
            done <= 0;
        end

        ADD:
          begin
            r_product = r_multiplicand + r_product;
          end
        SHIFT:
          begin
            r_multiplicand = r_multiplicand << 1;
            r_multiplier = r_multiplier >> 1;

          end
        DONE:
          begin
            if (sign) begin //check if the result is negative
                result <= ~r_product+1;
            end
            else begin
                result <= r_product;
            end
```

```
            done = 1;
          end
        endcase
      end
      end

endmodule
```

```
module tb;
    reg clk, rstn, start;
    reg [3:0] a,b;

    wire [7:0]result;
    wire done;

    initial begin
        clk=0;
        forever begin
            #5 clk=!clk;
        end
    end

    mul8_sign mul8_sign(a,b,clk, rstn, start, result, done);


    initial begin
        a=-3; b=7;

    #200 $finish;
    end

    initial begin
        rstn = 0;
        #5 rstn=1;
    end

    initial begin
        start = 0;
        #20 start = 1;
    end

    initial begin
        $dumpfile("wave.vcd");
        $dumpvars(0,tb);
    end
endmodule
```

노트북에 VIVADO 가 설치가 안돼서 https://edaplayground.com/ 를 이용하였습니다.

이 웹에는 radix 를 signed decimal 로 변경 하는 기능이 없어서.. 제가 직접 바꿔서 첨부하겠습니다. 죄송합니다.

Testbench 에서 -3 과 7 을 곱하였고 -21 이 성공적으로 나왔습니다.

Start 단계에서 multiplicand 와 multiplier 를 할당할 때 부호를 보고 음수라면 2 의 보수를 취하게 하였습니다. 또한 result 의 부호를 따지는 레지스터, `reg sign;`을 추가하였고 start 단계에서 a, b 의 부호를 xor 하여 1 이면 음수, 0 이면 양수가 되도록 설정하였습니다. 마지막 result 에 할당할 때 음수라면 2 의 보수를 취해서 할당하도록 하였습니다.