[과제4] 4bit x 4bit = 8bit Multiplier(Signed Magnitude) 설계

|  |
| --- |
| //과제4  //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 1이면 음수 0이면 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 |
| //Simulation waveform |
| //-3:1101 |
| //7: 0111 |
| //-21:11101011 |

노트북에 VIVADO가 설치가 안돼서 <https://edaplayground.com/> 를 이용하였습니다.

이 웹에는 radix를 signed decimal로 변경 하는 기능이 없어서.. 제가 직접 바꿔서 첨부하겠습니다. 죄송합니다.

Testbench에서 -3과 7을 곱하였고 -21이 성공적으로 나왔습니다.

Start단계에서 multiplicand와 multiplier를 할당할 때 부호를 보고 음수라면 2의 보수를 취하게 하였습니다. 또한result의 부호를 따지는 레지스터, reg sign;을 추가하였고 start 단계에서 a, b의 부호를 xor하여 1이면 음수, 0이면 양수가 되도록 설정하였습니다. 마지막 result에 할당할 때 음수라면 2의 보수를 취해서 할당하도록 하였습니다.