Test-Bench

My_adder

```
`timescale 1ns / 1ps

module tb_add();
    parameter BITWIDTH = 32;

    //for my IP
    reg [BITWIDTH-1:0] ain;
    reg [BITWIDTH-1:0] bin;
    wire [BITWIDTH-1:0] dout;
    wire   overflow;

    //for test
    integer i;
        //random test vector generation
        initial begin
          for(i=0; i<32; i=i+1) begin
            ain = $urandom%(2**31);
            bin = $urandom%(2**31);
            #10;
          end
        end

    //my IP
    my_add #(BITWIDTH) MY_ADDER(
        .ain(ain),
        .bin(bin),
        .dout(dout),
        .overflow(overflow)
    );
endmodule
```

My_mul

```
`timescale 1ns / 1ps

module tb_mul();
    parameter BITWIDTH = 32;

    //for my IP
    reg [BITWIDTH-1:0] ain;
    reg [BITWIDTH-1:0] bin;
    wire [2*BITWIDTH-1:0] dout;

    //for test
    integer i;
        //random test vector generation
```

```
initial begin
  for(i=0; i<32; i=i+1) begin
    ain = $urandom%(2**31);
    bin = $urandom%(2**31);
    #10;
  end
end

//my IP
my_mul #(BITWIDTH) MY_MUL(
    .ain(ain),
    .bin(bin),
    .dout(dout)
);


endmodule
```

My_fusedmult

```
`timescale 1ns / 1ps
module tb_fusedmult();
    parameter BITWIDTH = 32;

    //for my IP
    reg [BITWIDTH-1:0] ain;
    reg [BITWIDTH-1:0] bin;
    reg clk;
    reg en;
    wire [2*BITWIDTH-1:0] dout;


    //for test
    integer i;
        //random test vector generation
```

```
initial begin
    clk<=0;
    en<=0;
    #30;
    en<=1;
  for(i=0; i<32; i=i+1) begin
    ain = $urandom%(2**31);
    bin = $urandom%(2**31);
    #10;
  end
end

//my IP
my_fusedmult #(BITWIDTH) MY_MAC(
    .ain(ain),
    .bin(bin),
    .en(en),
    .clk(clk),
    .dout(dout)
);

always #5 clk = ~clk;

endmodule
```