```
// its just a sample, we can add as many modes as we want to this sfp, which can be treated as
alphas later.


module sfp (clk,ofifo_data,sram_data,mode,out);

parameter psum_bw = 16;
//parameter col = 8;
input clk;
input signed [psum_bw-1:0] ofifo_data;
input signed [psum_bw-1:0] sram_data;
input [2:0] mode;  // can extend to include other functions as well
output reg signed [psum_bw-1:0] out;


wire signed [psum_bw-1 : 0] result_from_relu;
sfp_relu #(.BW(psum_bw)) relu (.a(ofifo_data),.out(result_from_relu));

wire signed [psum_bw-1:0] result_from_accumulation;

sfp_accumulation #(.BW(psum_bw)) accu
(.ofifo_in(ofifo_data),.sram_data(sram_data),.out(result_from_accumulation));

always @(*) begin
   case (mode)
         3'b000 : out = result_from_relu;
         3'b001 : out = result_from_accumulation;
         3'b010 : out = ofifo_data;
         default : out = ofifo_data;
   endcase
end
endmodule




module sfp_accumulation #(parameter BW =16) (ofifo_in,sram_data,out);
//parameter bw;


input signed [BW-1:0]ofifo_in;
input signed [BW-1:0]sram_data;
output signed [BW-1:0]out;
```

```
assign out = ofifo_in+sram_data;

endmodule

module sfp_relu #(parameter BW = 16)( a, out);

input signed [BW-1:0]a;
output reg signed [BW-1:0] out;

always @(*) begin
   if (a <= 0)
   out = {BW{1'b0}};
         else
   out = a;
end
endmodule
```