```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: IIT Guwahati
// Engineer: ABBANNAGARI VARUN KUMAR REDDY
//
// Create Date: 29.02.2024 14:22:14
// Design Name:
// Module Name: sigmoid
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments: we have to implement sigmoid output for positive values of x only.
//                      I have taken 4 bits for integer part and 16 bits for fractional part.
//                      i.e '4.16' fixed point representation.(0000.(16zeroes) to (1111.(16ones)) is my x
range.
//                      while observing simulation waveforms,
//                      make sure output waveforms are in 4.16 fixed point representation
//                      that means binary point is after 16 bits from right to left.
//////////////////////////////////////////////////////////////////////////////////


module sigmoid(x,sigmoidx );
input [19:0]x;
output[19:0]sigmoidx;

wire [19:0]pwl;
wire [19:0]exp;

pwl p(x,pwl);
exp e(x,exp);
newtonrapson nr(pwl,exp,sigmoidx);

endmodule


////////////////////////////////PWL Module///////////////
module pwl(x,y );
input [19:0]x;
```

```
output reg [19:0]y;


always @ (*)


if(x>=20'd327680)
    y=20'd65536;
else if (x<20'd327680 && x>=20'd155648)
    y=(x>>5)+20'd55296;
else if (x<20'd155648 && x>=20'd65536)
    y=(x>>3)+ 20'd40960;
else if (x<20'd65536 && x>=0)
    y=(x>>2)+20'd32768;
else
    y=20'd0;

endmodule
//////////////////////////
///////////////exp function//////////////////

module exp (x,y);
 input [19:0] x;
 output [19:0] y;

wire [39:0] xbyln2;
wire [3:0]m;
wire [3:0]j;
wire [11:0]r_temp;

wire [19:0]shift;
wire [39:0]rout;
wire [19:0]tempj;
wire [19:0]r;
wire [39:0]out_temp;
assign xbyln2=x*20'd94548;

assign m=xbyln2[35:32];
assign j=xbyln2[31:28];

rom r1(j,tempj);

assign r_temp=xbyln2[27:16];
assign rout={8'd0,r_temp}*20'd45408;
```

```
assign r=rout[35:16];

assign shift=tempj>>m;

assign out_temp=(20'd65536-r)*shift;

assign y=out_temp[35:16];




endmodule
```

-

```
module rom(j,out);
input [3:0]j;
output reg [19:0]out;

reg [19:0] ram[0:15];

always @(*)
begin
ram[0]=20'd65536;
ram[1]=20'd62757;
ram[2]=20'd60096;
ram[3]=20'd57548;
ram[4]=20'd55103;
ram[5]=20'd52772;
ram[6]=20'd50535;
ram[7]=20'd48392;
ram[8]=20'd46340;
ram[9]=20'd44376;
ram[10]=20'd42494;
ram[11]=20'd40693;
ram[12]=20'd38967;
ram[13]=20'd37315;
ram[14]=20'd35733;
ram[15]=20'd34218;


case(j)
4'd0:out=ram[0];
```

```
4'd1:out=ram[1];
4'd2:out=ram[2];
4'd3:out=ram[3];
4'd4:out=ram[4];
4'd5:out=ram[5];
4'd6:out=ram[6];
4'd7:out=ram[7];
4'd8:out=ram[8];
4'd9:out=ram[9];
4'd10:out=ram[10];
4'd11:out=ram[11];
4'd12:out=ram[12];
4'd13:out=ram[13];
4'd14:out=ram[14];
4'd15:out=ram[15];


default:out=20'd0;

endcase
end



endmodule

////////////////////////////////// NR module//////////////////
module newtonrapson(initsol,exp,finalsol  );

input [19:0]initsol;
input [19:0]exp;
output [19:0]finalsol;

wire [19:0]twosigmax;
wire [39:0]sigmasquarex;
wire [59:0]finalsoltemp;
wire[59:0]temp;

 assign twosigmax=initsol<<1;

assign sigmasquarex=initsol * initsol;
assign temp=sigmasquarex*(20'd65536+exp);

assign finalsoltemp={8'd0,twosigmax,32'd0}-temp;
```

```
assign finalsol=finalsoltemp[51:32];


endmodule
```