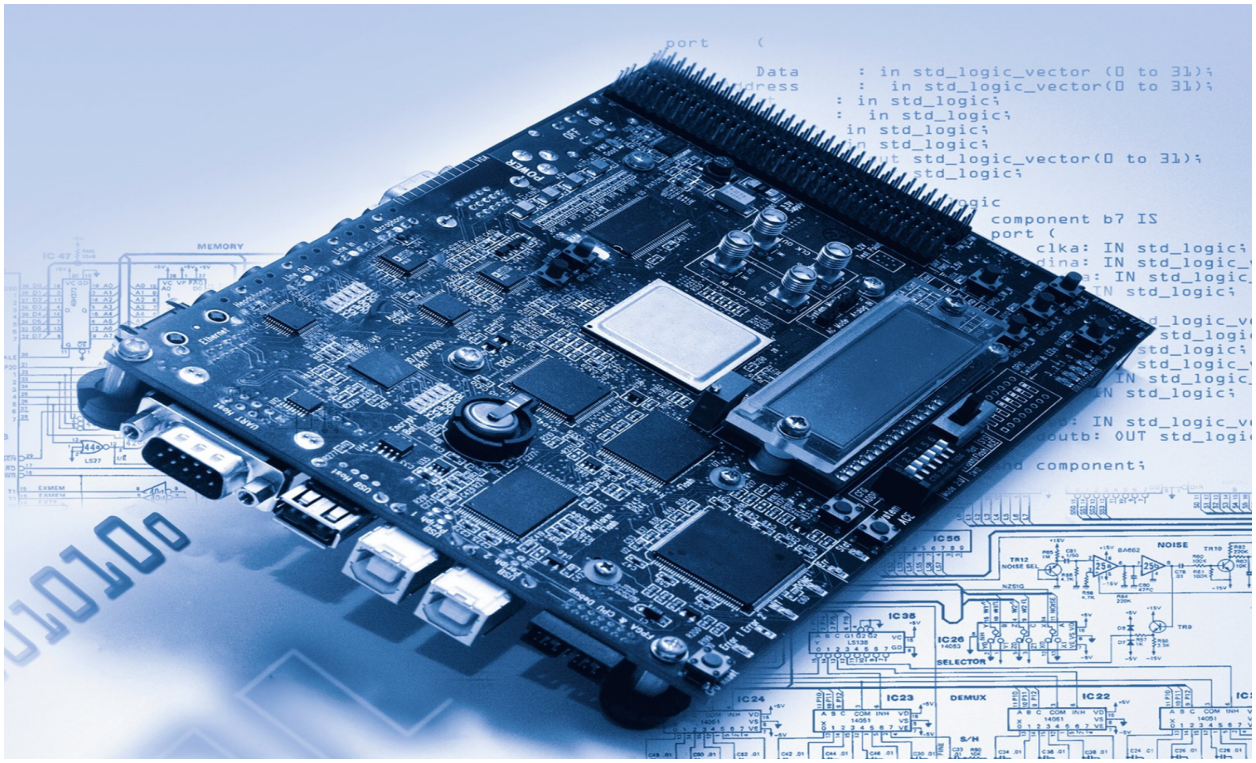


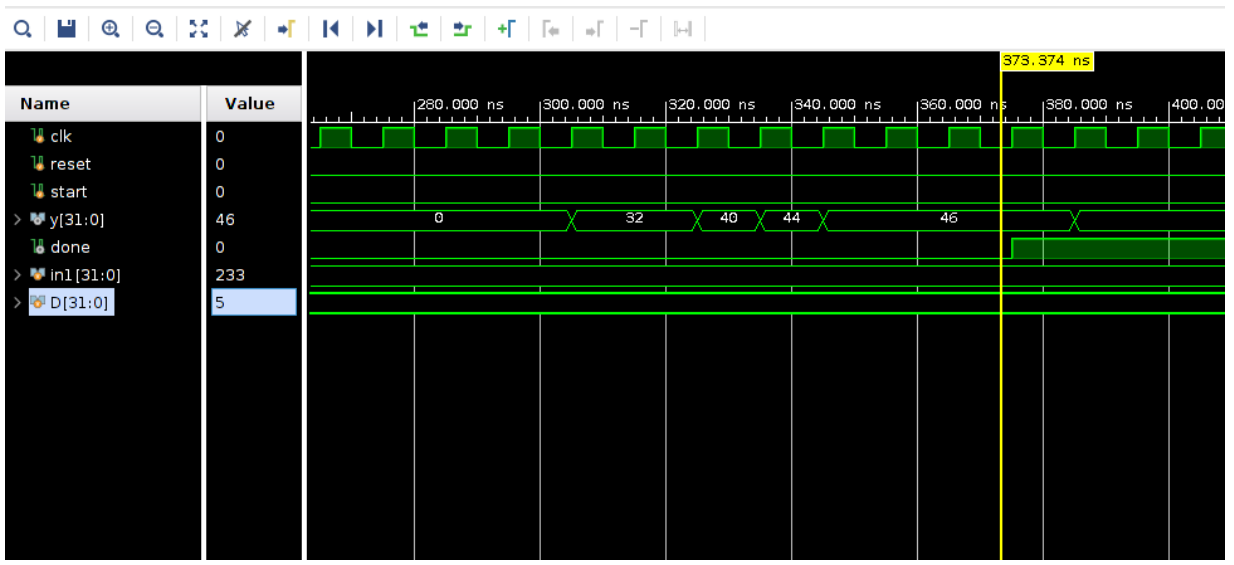
CMPE 691: Reconfigurable System Design

Project_Report # : Code

Name : Mehedi Galib
Student ID : UG34492

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12/13/2021 08:08:57 PM
// Design Name: Galib
// Module Name: Iterative_division
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module iterative_division(clk,reset,start,N,D,Q,done);
input wire [31:0] N,D;
output reg [31:0] Q;

 input wire clk;
 input wire reset;
 input wire start;
 output reg done;

 reg [31:0] R;
 reg [31:0] R_new;
 reg [63:0]sub;
 reg Q_new;


 reg [5:0] step; // 5 bits for counting upto 20;

 parameter [1:0] IDLE = 2'b00,
                 STEP  = 2'b01,
                 FINISH = 2'b10;
reg [1:0] state, next;
 // State transition logic
 always @(posedge clk) begin
 if (reset) state <= IDLE;
 else state <= next;
 end

 // Next state logic
 always@(*) begin
```

```
 next=2'bx;
 case(state)
 IDLE : begin
 if (start)
 begin
 next = STEP;
 end
 else next = IDLE;
 end
 STEP :
 begin
 if (step==0) next = FINISH;
 else begin next = STEP;
 sub=D<<step-1;
 if (R>=sub) begin R_new=R-sub[31:0]; Q_new=1'b1;end
 else begin R_new=R; Q_new=1'b0;end
 end
 end
 FINISH : next = IDLE;
 endcase
 end

// Output logic
 always @(posedge clk) begin
 if(reset) begin
 Q <= 0;
 R<=0;
 done <= 0;
 step<=6'd32;
 end
 else begin
 case(state)
 IDLE : begin
 step <= 6'd32;
 R <= N;
 Q<=0;
 end
 STEP : begin
 R<=R_new;
 Q[step-1]<=Q_new;
 step <= step - 1;
 end
 FINISH : begin
 done <= 1'b1;
 end
 endcase
 end
 end

endmodule
```




```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12/13/2021 04:47:00 AM
// Design Name:
// Module Name: log_calculation
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module log_calculation(clk,reset,start,in1,y,done);
```

```
input wire [31:0] in1;
input wire clk,reset,start;
output reg signed [31:0] y;
output reg done;
reg [31:0] _x,t,x;
reg signed [31:0] _y;
parameter [3:0] IDLE = 4'b0000,
                state_1 = 4'b0001,
                state_2 = 4'b0010,
                state_3 = 4'b0011,
                state_4 = 4'b0100,
                state_5 = 4'b0101,
                state_6 = 4'b0110,
                state_7 = 4'b0111,
                state_8 = 4'b1000,
                state_9 = 4'b1001,
                state_10 = 4'b1010,
                state_11 = 4'b1011,
                state_12 = 4'b1100,
                state_13 = 4'b1101,
                state_14 = 4'b1110,
                state_15 = 4'b1111;
 reg [3:0] state, next;
 always @(posedge clk)
 if (reset) state <= IDLE;
 else state <= next;

 always @(*) begin
 next = 4'bx;
 case (state)
 IDLE : if (start) next = state_1;
 else begin next = IDLE; done=1'b0;end

 state_1: begin
 if (x<32'h00008000)
 begin
 _x = x<<16;
 _y = y-32'hb1721;
 end
```

```
else begin
_x=x;
_y=y;
end
next = state_2;
done=1'b0;
end

state_2: begin
if (x<32'h00800000)
begin
_x = x<<8;
_y = y-32'h58b91;
end
else begin
_x=x;
_y=y;
end
next = state_3;
done=1'b0;
end

state_3: begin
if (x<32'h08000000)
begin
_x = x<<4;
_y = y-32'h2c5c8;
end
 else begin
_x=x;
_y=y;
end
next = state_4;
done=1'b0;
end

state_4: begin
if (x<32'h20000000)
begin
```

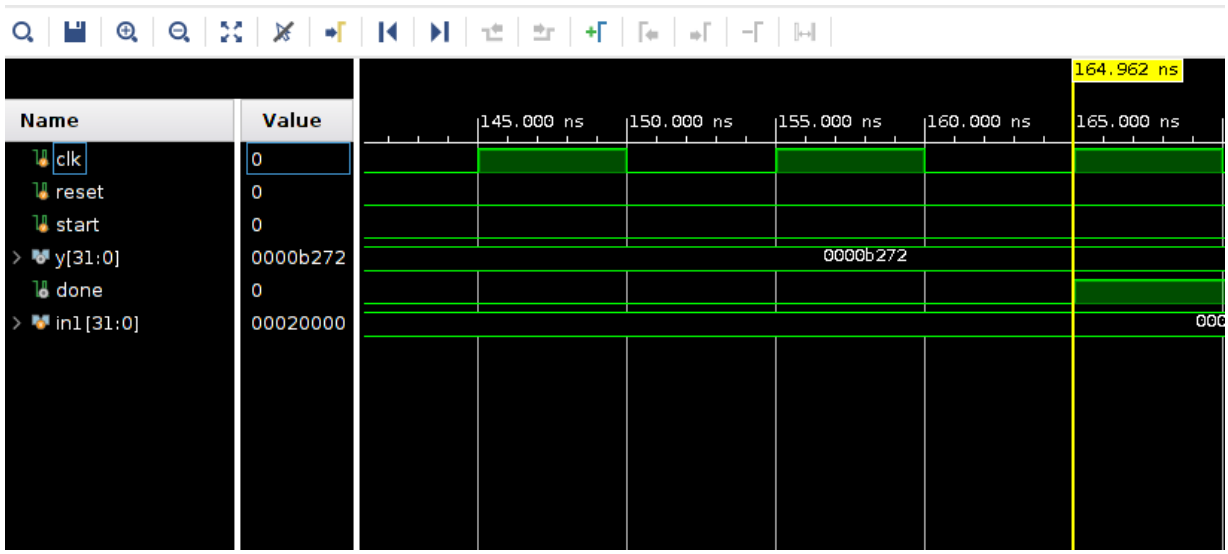
```
_x = x<<2;
_y = y-32'h162e4;
end
 else begin
_x=x;
_y=y;
end
next = state_5;
done=1'b0;
end

state_5: begin
if (x<32'h40000000)
begin
_x = x<<1;
_y = y-32'h0b172;
end
 else begin
_x=x;
_y=y;
end
next = state_6;
done=1'b0;
end

state_6: begin
t=x+(x>>>1);
if ((t&32'h80000000)==0)
begin
_x = t;
_y = y-32'h067cd;
end
 else begin
_x=x;
_y=y;
end
next = state_7;
done=1'b0;
end
```

```
state_7: begin
t=x+(x>>>2);
if ((t&32'h80000000)==0)
begin
_x = t;
_y = y-32'h03920;
end
 else begin
_x=x;
_y=y;
end
next = state_8;
done=1'b0;
end

state_8: begin
t=x+(x>>>3);
if ((t&32'h80000000)==0)
begin
_x = t;
_y = y-32'h01e27;
end
 else begin
_x=x;
_y=y;
end
next = state_9;
done=1'b0;
end

state_9: begin
t=x+(x>>>4);
if ((t&32'h80000000)==0)
begin
_x = t;
_y = y-32'h00f85;
end
 else begin
```

```
_x=x;
_y=y;
end
next = state_10;
done=1'b0;
end

state_10: begin
t=x+(x>>>5);
if ((t&32'h80000000)==0)
begin
_x = t;
_y = y-32'h007e1;
end

 else begin
_x=x;
_y=y;
end
next = state_11;
done=1'b0;
end

state_11: begin
t=x+(x>>>6);
if ((t&32'h80000000)==0)
begin
_x = t;
_y = y-32'h003f8;
end
 else begin
_x=x;
_y=y;
end
next = state_12;
done=1'b0;
end

state_12: begin
```

```
 t=x+(x>>>7);
 if ((t&32'h80000000)==0)
 begin
 _x = t;
 _y = y-32'h001fe;
 end
  else begin
 _x=x;
 _y=y;
 end
 next = state_13;
 done=1'b0;
 end

 state_13: begin
 _x = 32'h80000000-x;
 _y = y;
 next = state_14;
 done=1'b0;
 end

 state_14: begin
 _x = 32'h0;
 _y = y-x>>>15;
 next = IDLE;
 done=1'b1;
 end

 //state_15: begin
 //_x = x;
 //_y = y;
 //next = IDLE;
 //done=1'b1;
// end


 endcase
 end
```

```
 always @(posedge clk)
 if (reset) begin
 y <= 32'ha65af;
t<=32'h0;
x<=in1;
 end
 else begin
 y <= 32'ha65af;
 case (state)
 IDLE: begin
 y <= 32'ha65af; // default outputs
 x<=in1;
 end
 state_1, state_2, state_3, state_4,
state_5,state_6,state_7,state_8,state_9,state_10,state_11,state_12,st
ate_13,state_14,state_15: begin
  y <= _y;
  x<=_x;
  end
 endcase
 end
endmodule
```



log(2) calculation. Here in 32 bit datawidth, 16 bits are integral portions and 16 bits are fractional portions.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12/11/2021 12:25:38 PM
// Design Name:
// Module Name: predict
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module predict(clk,reset,start_calculation,mean_class1_input,mean_class0_input,
std_class1_input, std_class0_input,
test_feature_input,jointProbability_class0,jointProbability_class1,classification);
input wire [255:0] mean_class1_input;
input wire [255:0] mean_class0_input;
input wire [255:0] std_class1_input;
input wire [255:0] std_class0_input;
input wire [255:0] test_feature_input;
reg signed [31:0] mean_class1[7:0];
reg signed [31:0] mean_class0[7:0];
reg signed [31:0] std_class1[7:0];
reg signed [31:0] std_class0[7:0];
reg signed [31:0] feature_input[7:0];
input wire [31:0] start_calculation,jointProbability_class0,jointProbability_class1;

always@(*)
begin
     mean_class1[0]=mean_class1_input[31:0];
     mean_class1[1]=mean_class1_input[63:32];
        mean_class1[2]=mean_class1_input[95:64];
     mean_class1[3]=mean_class1_input[127:96];
        mean_class1[4]=mean_class1_input[159:128];
     mean_class1[5]=mean_class1_input[191:160];
        mean_class1[6]=mean_class1_input[223:192];
     mean_class1[7]=mean_class1_input[255:224];
     mean_class0[0]=mean_class0_input[31:0];
     mean_class0[1]=mean_class1_input[63:32];
        mean_class0[2]=mean_class1_input[95:64];
     mean_class0[3]=mean_class1_input[127:96];
        mean_class0[4]=mean_class1_input[159:128];
```

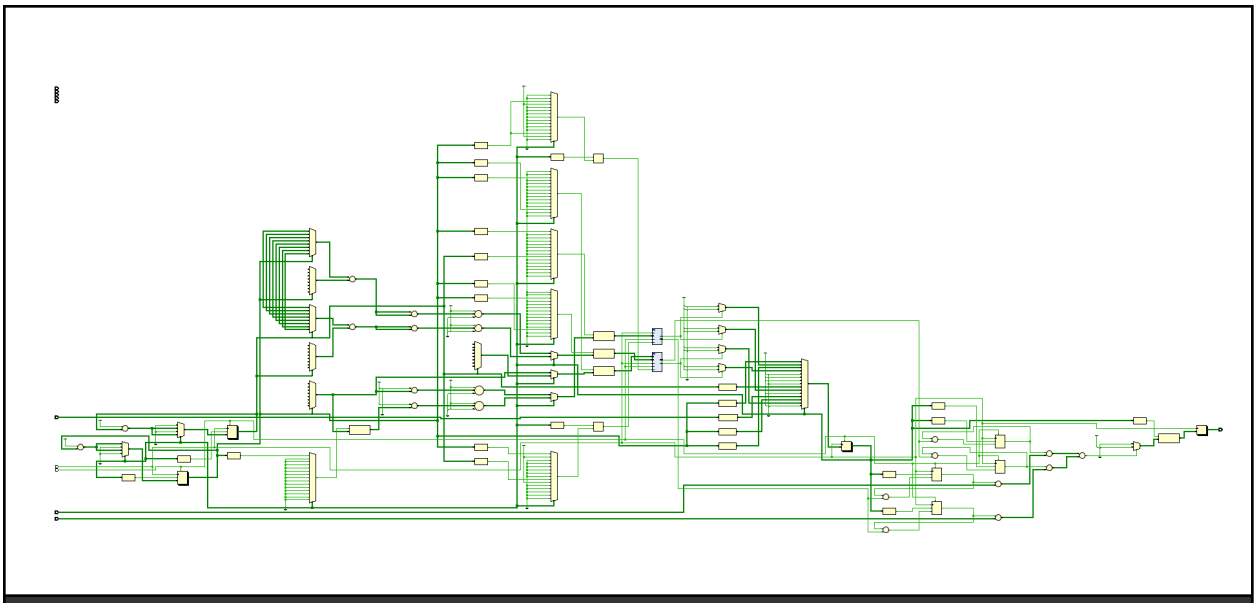
```
    mean_class0[5]=mean_class1_input[191:160];
       mean_class0[6]=mean_class1_input[223:192];
    mean_class0[7]=mean_class1_input[255:224];
    std_class1[0]=std_class1_input[31:0];
    std_class1[1]=std_class1_input[63:32];
       std_class1[2]=std_class1_input[95:64];
    std_class1[3]=std_class1_input[127:96];
       std_class1[4]=std_class1_input[159:128];
    std_class1[5]=std_class1_input[191:160];
       std_class1[6]=std_class1_input[223:192];
    std_class1[7]=std_class1_input[255:224];
    std_class0[0]=std_class0_input[31:0];
    std_class0[1]=std_class0_input[63:32];
       std_class0[2]=std_class0_input[95:64];
    std_class0[3]=std_class0_input[127:96];
       std_class0[4]=std_class0_input[159:128];
    std_class0[5]=std_class0_input[191:160];
       std_class0[6]=std_class0_input[223:192];
    std_class0[7]=std_class0_input[255:224];
    feature_input[0]=test_feature_input[31:0];
    feature_input[1]=test_feature_input[63:32];
       feature_input[2]=test_feature_input[95:64];
    feature_input[3]=test_feature_input[127:96];
       feature_input[4]=test_feature_input[159:128];
    feature_input[5]=test_feature_input[191:160];
       feature_input[6]=test_feature_input[223:192];
    feature_input[7]=test_feature_input[255:224];
end



//module predict(clk,reset,classification);
output reg signed [31:0] classification;
input wire clk, reset;

reg signed [31:0] out,out3,out4;
reg signed [63:0]
sigma1_intermediate1,sigma1_intermediate2,mean1_intermediate2,mean1_intermediate3,sigma0_int
ermediate1,sigma0_intermediate2,mean0_intermediate2,mean0_intermediate3;
parameter num_of_features=8;
reg [31:0]class0_probability,class1_probability;
reg signed class0_temp1,class0_temp2,class1_temp1,class1_temp2;
reg signed [31:0] pi;
reg signed [31:0] user_data[num_of_features-1:0];
reg signed [31:0] class1_mean[num_of_features-1:0];
reg signed [31:0] class0_mean[num_of_features-1:0];
reg signed [31:0] class1_std[num_of_features-1:0];
reg signed [31:0] class0_std[num_of_features-1:0];

reg signed [31:0]
sigma1,sigma0,sigma_log,mean1,mean0,_x1,_x0,mean_div,mean_divisor,mean1_intermediate1,mean0_
intermediate1,Class;
```

```
reg sigma_start,mean_start;
wire signed [31:0] sigma_log_result,mean_div_result;
wire done,mean_done;
initial
begin
pi=32'b00000011001001000001111110110;
class0_probability=32'b10110011101111011010;
end



reg [3:0] step1,step0; // 5 bits for counting upto 20;

 parameter [3:0] INITIAL_sigma1=4'b0000,
                 INITIAL_sigma0=4'b0001,
                 WAIT1 = 4'b0010,
                 UPDATE1=4'b0011,
                 WAIT0 = 4'b0100,
                 UPDATE0=4'b0101,
                 IDLE = 4'b0110,
                 MEAN1  = 4'b0111,
                 WAIT_mean1=4'b1000,
                 UPDATE_mean1=4'b1001,
                 MEAN0  = 4'b1010,
                 WAIT_mean0=4'b1011,
                 UPDATE_mean0=4'b1100,
                 Joint_probability=4'b1101;
reg [3:0] state, next;

log_calculation uut(clk,reset,sigma_start,sigma_log,sigma_log_result,done);
iterative_division uut1
(clk,reset,mean_start,mean_div,mean_divisor,mean_div_result,mean_done);
 // State transition logic
 always @(posedge clk) begin
 if (reset) state <= INITIAL_sigma1;
 else state <= next;
 end

 // Next state logic
 always@(*) begin
 next=3'bx;
 case(state)
 ////////////////class_1_sigma calculation, which requires only one during initialization
 INITIAL_sigma1: begin
 if (step1==num_of_features)
 begin
 next=INITIAL_sigma0;
 end
 else
 begin
 sigma1=class1_std[step1];
 sigma1_intermediate1=pi*sigma1;
```

```
sigma1_intermediate2=sigma1_intermediate1>>>16;
sigma_log=sigma1_intermediate2[31:0];
sigma_start=1'b1;
next=WAIT1;
end
end
WAIT1:
begin next=done?UPDATE1:WAIT1; sigma_start=1'b0; end
UPDATE1:next=INITIAL_sigma1;
 ////////////////class_0_sigma calculation, which requires only one during initialization
INITIAL_sigma0: begin
if (step0==num_of_features)
begin
next=IDLE;
end
else
begin
sigma0=class0_std[step0];
sigma0_intermediate1=pi*sigma1;
sigma0_intermediate2=sigma0_intermediate1>>>16;
sigma_log=sigma0_intermediate2[31:0];
sigma_start=1'b1;
next=WAIT0;
end
end
WAIT0:
begin next=done?UPDATE0:WAIT0; sigma_start=1'b0; end
UPDATE0:next=INITIAL_sigma0;
////////////////////////////IDLE state/////////////////////////////////////////////
IDLE: begin if (start_calculation==32'b1)
begin
next=MEAN1;
end
else next=IDLE;
end

////////////////class_1_mean calculation, which requires when we have each input
MEAN1: begin
if (step1==num_of_features)
begin
next=MEAN0;
end
else
begin
mean1=class1_mean[step1];
mean_divisor=class1_std[step1];
_x1=user_data[step1];
mean1_intermediate1=_x1-mean1;
mean1_intermediate2=mean1_intermediate1*mean1_intermediate1;
mean1_intermediate3=mean1_intermediate2>>16;
mean_div=mean1_intermediate3[31:0];
mean_start=1'b1;
```

```
 next=WAIT_mean1;
 end
 end
 WAIT_mean1:
 begin next=mean_done?UPDATE_mean1:WAIT_mean1; mean_start=1'b0; end
 UPDATE_mean1:next=MEAN1;

  ////////////////class_0_mean calculation, which requires when we have each input
 MEAN0: begin
 if (step1==num_of_features)
 begin
 next=Joint_probability;
 end
 else
 begin
 mean0=class0_mean[step0];
 mean_divisor=class0_std[step0];
 _x0=user_data[step0];
 mean0_intermediate1=_x0-mean0;
 mean0_intermediate2=mean0_intermediate1*mean0_intermediate1;
 mean0_intermediate3=mean0_intermediate2>>16;
 mean_div=mean0_intermediate3[31:0];
 mean_start=1'b1;
 next=WAIT_mean0;
 end
 end
 WAIT_mean0:
 begin next=mean_done?UPDATE_mean0:WAIT_mean0; mean_start=1'b0; end
 UPDATE_mean0:next=MEAN0;
 Joint_probability:
 begin
class0_probability=jointProbability_class0+class0_temp1+class0_temp2;
class1_probability=jointProbability_class1+class1_temp1+class1_temp2;
if (class0_probability>class1_probability) Class=32'b0;
else Class=32'b10000000000000000;
next=IDLE;
 end

 default:next=IDLE;
 endcase
 end




// Output logic
 always @(posedge clk) begin
```

```
 if(reset) begin
 class1_temp1 <= 0;
 class1_temp2<=0;
 class0_temp1 <= 0;
 class0_temp2<=0;
// done <= 0;
 //mean_done<=0;
 step1<=0;
 step0<=0;
 classification<=32'bx;
 end
 else begin
 classification<=32'bx;
 case(state)
 INITIAL_sigma1: step0<=1'b0;
 UPDATE1:
 begin
 class1_temp1<=class1_temp1-sigma_log_result;
 step1 <= step1 + 1;
 end
 INITIAL_sigma0: step1<=1'b0;
 UPDATE0:
 begin
 class0_temp1<=class0_temp1-sigma_log_result;
 step0 <= step0 + 1;
 end
 MEAN1: step0<=1'b0;
 UPDATE_mean1:
 begin
 class1_temp2<=class1_temp2-mean_div_result;
 step1 <= step1 + 1;
 end
 MEAN0: step1<=1'b0;
 UPDATE_mean0:
 begin
 class0_temp2<=class0_temp2-mean_div_result;
 step0 <= step0 + 1;
 end
 Joint_probability:
 classification<=Class;
 endcase
 end
 end


endmodule
```

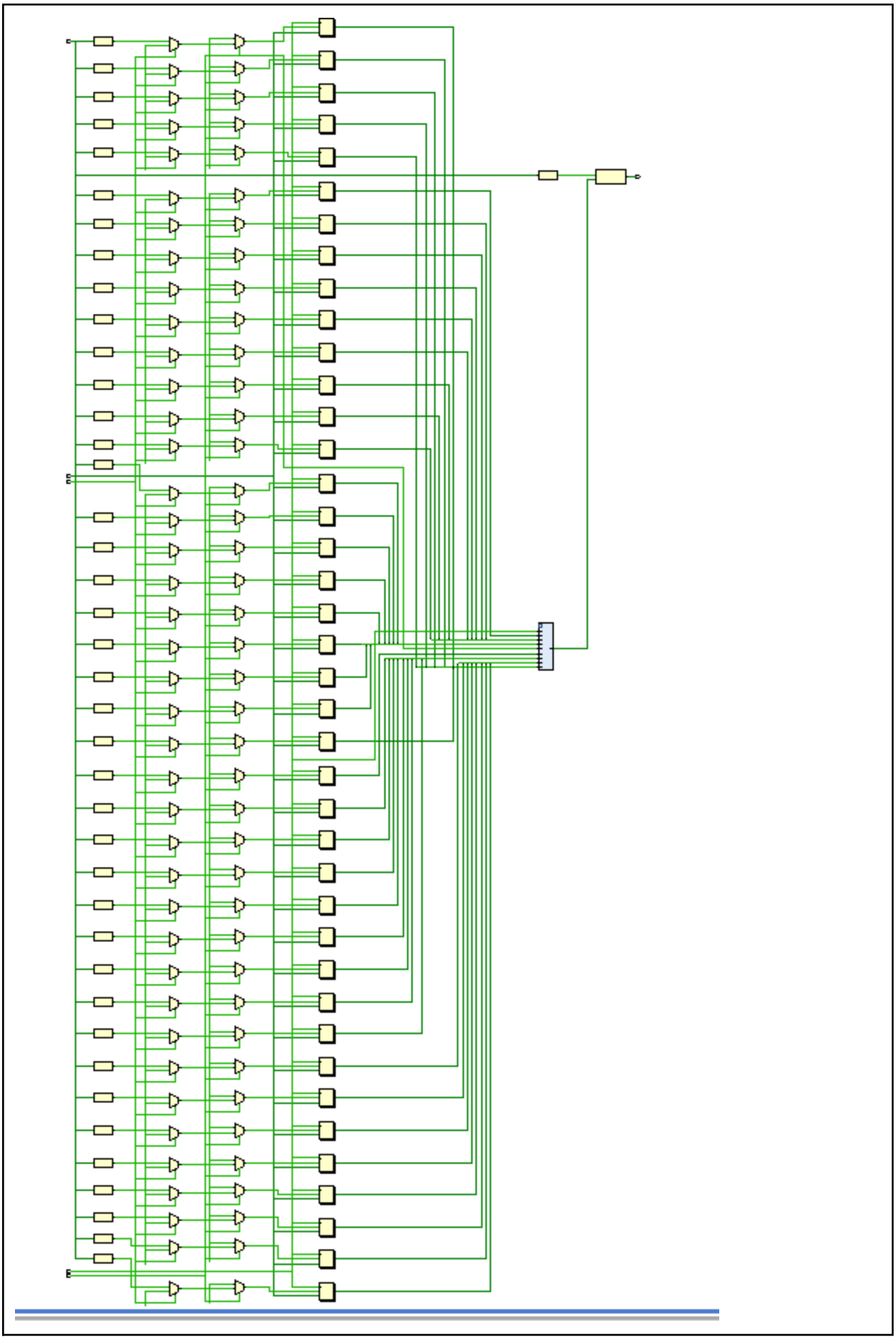


```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 12/13/2021 05:26:24 PM
// Design Name:
// Module Name: prediction_top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module prediction_top(// Signals for Avalon-MM slave port "s1" with
```

```
irq
csi_clockreset_clk, //clockreset clock interface
csi_clockreset_reset_n,//clockreset clock interface
avs_s1_address,//s1 slave interface
avs_s1_read, //s1 slave interface
avs_s1_write,  //s1 slave interface
avs_s1_writedata, //s1 slave interface
avs_s1_readdata //s1 slave interface
//ins_irq0_irq; //irq0 port interrupt sender interface
 );

 input csi_clockreset_clk;
 input csi_clockreset_reset_n;
 input [7:0] avs_s1_address;
 input avs_s1_read;input avs_s1_write;
 input [31:0] avs_s1_writedata;
 output reg [31:0] avs_s1_readdata;
 //output ins_irq0_irq;


reg signed [31:0] mean_class1[7:0];
reg signed [31:0] mean_class0[7:0];
reg signed [31:0] std_class1[7:0];
reg signed [31:0] std_class0[7:0];
reg signed [31:0] feature_input[7:0];
reg [31:0]
start_calculation,jointProbability_class0,jointProbability_class1;
wire [31:0] classification;
//a0,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,control,status,r0
,r1;
always @(posedge csi_clockreset_clk) begin
  if (csi_clockreset_reset_n==0) begin

  end else if (avs_s1_write) begin
    case (avs_s1_address)
      0: mean_class1[0]<=avs_s1_writedata;
      1: mean_class1[1]<=avs_s1_writedata;
       2: mean_class1[2]<=avs_s1_writedata;
      3: mean_class1[3]<=avs_s1_writedata;
```

```
    4: mean_class1[4]<=avs_s1_writedata;
   5: mean_class1[5]<=avs_s1_writedata;
    6: mean_class1[6]<=avs_s1_writedata;
   7: mean_class1[7]<=avs_s1_writedata;
   8: mean_class0[0]<=avs_s1_writedata;
   9: mean_class0[1]<=avs_s1_writedata;
   10: mean_class0[2]<=avs_s1_writedata;
  11: mean_class0[3]<=avs_s1_writedata;
   12: mean_class0[4]<=avs_s1_writedata;
  13: mean_class0[5]<=avs_s1_writedata;
   14: mean_class0[6]<=avs_s1_writedata;
  15: mean_class0[7]<=avs_s1_writedata;
  16: std_class1[0]<=avs_s1_writedata;
  17: std_class1[1]<=avs_s1_writedata;
   18: std_class1[2]<=avs_s1_writedata;
  19: std_class1[3]<=avs_s1_writedata;
   20: std_class1[4]<=avs_s1_writedata;
  21: std_class1[5]<=avs_s1_writedata;
   22: std_class1[6]<=avs_s1_writedata;
  23: std_class1[7]<=avs_s1_writedata;
  24: std_class0[0]<=avs_s1_writedata;
  25: std_class0[1]<=avs_s1_writedata;
   26: std_class0[2]<=avs_s1_writedata;
  27: std_class0[3]<=avs_s1_writedata;
   28: std_class0[4]<=avs_s1_writedata;
  29: std_class0[5]<=avs_s1_writedata;
   30: std_class0[6]<=avs_s1_writedata;
  31: std_class0[7]<=avs_s1_writedata;
  32:jointProbability_class0<=avs_s1_writedata;
  33:jointProbability_class0<=avs_s1_writedata;
  32: feature_input[0]<=avs_s1_writedata;
  33: feature_input[1]<=avs_s1_writedata;
   34: feature_input[2]<=avs_s1_writedata;
  35: feature_input[3]<=avs_s1_writedata;
   36: feature_input[4]<=avs_s1_writedata;
  37: feature_input[5]<=avs_s1_writedata;
   38: feature_input[6]<=avs_s1_writedata;
  39: feature_input[7]<=avs_s1_writedata;
  40: start_calculation<=avs_s1_writedata;
```

```
    endcase
  end
end

predict
Block1(csi_clockreset_clk,csi_clockreset_reset_n,start_calculation,{m
ean_class1[7],mean_class1[6],mean_class1[5],mean_class1[4],mean_class
1[3],mean_class1[2],mean_class1[1],mean_class1[0]},{mean_class0[7],me
an_class0[6],mean_class0[5],mean_class0[4],mean_class0[3],mean_class0
[2],mean_class0[1],mean_class0[0]},
{std_class1[7],std_class1[6],std_class1[5],std_class1[4],std_class1[3
],std_class1[2],std_class1[1],std_class1[0]},
{std_class0[7],std_class0[6],std_class0[5],std_class0[4],std_class0[3
],std_class0[2],std_class0[1],std_class0[0]},
{feature_input[7],feature_input[6],feature_input[5],feature_input[4],
feature_input[3],feature_input[2],feature_input[1],feature_input[0]},
jointProbability_class0,jointProbability_class1,classification);

// Output logic returns values based on the offset from the base
address,
//  offsets 0,4, and 12  are valid here
// using the provided slave address

always @(*) begin//avs_s1_readdata = 32'hf0f0f0f0;
  case (avs_s1_address)
    0: avs_s1_readdata = classification;
  endcase
end
endmodule
```



C_implementation of Gaussian naive bayes method

```c
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include "hwlib.h"
#include "socal/socal.h"
#include "socal/hps.h"
#include "socal/alt_gpio.h"
#include "hps_0.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/tcp.h>
#include <sys/ioctl.h>

#include <time.h>

#define HW_REGS_BASE ( ALT_STM_OFST )
#define HW_REGS_SPAN ( 0x04000000 )
#define HW_REGS_MASK ( HW_REGS_SPAN - 1 )

// convert 8-digit hex string to uint32_t
uint32_t h2i(char c) {
uint32_t i = 0;
switch(c) {
case '0': i=0; break;
case '1': i=1; break;
case '2': i=2; break;
case '3': i=3; break;
case '4': i=4; break;
```

```
case '5': i=5; break;
case '6': i=6; break;
case '7': i=7; break;
case '8': i=8; break;
case '9': i=9; break;
case 'a':
case 'A': i=10; break;
case 'b':
case 'B': i=11; break;
case 'c':
case 'C': i=12; break;
case 'd':
case 'D': i=13; break;
case 'e':
case 'E': i=14; break;
case 'f':
case 'F': i=15; break;
}
return i;
}

//convert hex assii string to integer
uint32_t sh2i(char *s) {
   //printf("Calling sh2i\n");
uint32_t integer = 0;
uint32_t index=0;
while (s[index] != '\0') {
 integer = (integer<<4) + h2i(s[index]);
index++;
//printf(":%d:", index);
}
   //printf("\nExiting sh2i\n");
return integer;
}
 //convert decimal ascii string to integer
uint32_t sd2i(char *s) {
   //printf("Calling sh2i\n");
uint32_t integer = 0;
uint32_t index=0;
```

```
while (s[index] != '\0') {
 integer = (integer*10) + s[index]-'0';
index++;
//printf(":%d:", index);
}
   //printf("\nExiting sh2i\n");
return integer;
}
 //convert binary ascii string to integer
uint32_t sb2i(char *s) {
   //printf("Calling sh2i\n");
uint32_t integer = 0;
uint32_t index=0;
while (s[index] != '\0') {
 integer = (integer*2) + s[index]-'0';
index++;
//printf(":%d:", index);
}
   //printf("\nExiting sh2i\n");
return integer;
}

//convert ascii string to integer
//if string has prefix 0x or 0b then the rest of the string is assumed
to be hex or binary respectively
//otherwise, a decimal string is assumed
// i = s2i("0xhf")
// i = s2i("0b010110")
// i = s2i("0b921")
uint32_t s2i(char *s) {
uint32_t integer;
if (s[0]=='0' && s[1] == 'x'){
integer = sh2i(s);
} else if (s[0]=='0' && s[1] == 'b'){
integer = sb2i(s);
}else{
integer = sd2i(s);
}
return integer;
```

```
}



#define IO_PTR(_BASE) (virtual_base+((unsigned
long)(ALT_LWFPGASLVS_OFST+(_BASE))&(unsigned long)(HW_REGS_MASK)))
#define GET_BIT(x,b) (((x)>>(b))&1)

int main(int argc, char *argv[])
{

float epsilon = 1e-9;
 int num_of_features = 8;
  float mean_samples[num_of_features],std_samples[num_of_features]; //
mean and standard deviation for whole sample
 float meanclass0[num_of_features],stdclass0[num_of_features]; // mean
and standard deviation for class zero
 float meanclass1[num_of_features],stdclass1[num_of_features]; // mean
and standard deviation for class one
 float jointProbablity_class0,jointProbablity_class1; // Joint
probability for diffrent class
 int class0=0, class1 = 0;
 float data[] ={70,  17, 1,  18, 1,  0.03227344074374504,
0.22464497623584065,    0.012858208372436414,   0,
 25, 38, 4, 1,   41, 0.04487346687872599,    0.06150886599011331,
0.013798560999914878,   1,
 24, 49, 5,  1,  41, 0.03866265769041786,    0.07725706315503511,
0.013672384821919827,   1,
 23,   54, 22, 1,  231,    0.020987556173195386,   0.39143764927370334,
0.020987556173195386,   0,
 25,   36, 3,  1,  31, 0.04292823375859413,    0.07091777122125874,
0.018163133657940478,   1,
 25,   43, 4,  1,  31, 0.043279842849755425,   0.07326860946567212,
0.01826227359329245,    1,
 23,   59, 23, 1,  231,    0.02987569479330077,    0.3908339705039096,
0.020965205796172552,   0};
 int len_data=sizeof(data)/sizeof(data[0]);
 printf("%d\n",len_data);
```

```
 int samples = (len_data/num_of_features);
 float user_data[num_of_features];


  float fraction = 20.1111111111,frac;
  int decimal,rem,len;
  int i = 0;
  int counter = 0;

  char bnum[32],t;




  for(int i =0;i<num_of_features;i++)
 {
   mean_samples[i] = 0;
   meanclass0[i]=0;
   meanclass1[i]=0;
   stdclass0[i] = 0;
   stdclass1[i] = 0;
   std_samples[i] = 0;
 }
 ////////////////////////////////////////////////////// For calculating
all mean//////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
   for(int j =num_of_features;j<len_data;j=j+num_of_features+1)
   {
       if (data[j]==0)
       {
           class0++;
           for(int i =0;i<num_of_features;i++)
           {
```

```
                meanclass0[i]+=data[i+j-num_of_features];
            }
        }
        else
        {
            class1++;
             for(int i =0;i<num_of_features;i++)
            {
                meanclass1[i]+=data[i+j-num_of_features];
            }
        }
          for(int i =0;i<num_of_features;i++)
            {
                mean_samples[i]+=data[i+j-num_of_features];
            }
    }
 for(int i =0;i<num_of_features;i++)
 {
   mean_samples[i] /= samples;
   meanclass0[i]/=class0;
   meanclass1[i]/=class1;
 }
 ////////////////////////////////////////////////////// End of mean
calculation ////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////// For calculating all
standard deviation//////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////
    for(int j =num_of_features;j<len_data;j=j+num_of_features+1)
```

```
   {
       if (data[j]==0)
       {

           for(int i =0;i<num_of_features;i++)
           {
                stdclass0[i]+=
(meanclass0[i]-data[i+j-num_of_features])*(meanclass0[i]-data[i+j-num_of
_features]);
           }
       }
       else
       {

            for(int i =0;i<num_of_features;i++)
           {
                stdclass1[i]+=
(meanclass1[i]-data[i+j-num_of_features])*(meanclass1[i]-data[i+j-num_of
_features]);
           }
       }
         for(int i =0;i<num_of_features;i++)
           {
                std_samples[i]+= (mean_samples[i] -
data[i+j-num_of_features])*(mean_samples[i] -
data[i+j-num_of_features]);
           }
   }
       float maximum = 0;
     for(int i =0;i<num_of_features;i++)
         {
             std_samples[i] /= samples;
             stdclass0[i]/=class0;
             stdclass1[i]/=class1;
              if(std_samples[i] > maximum)
              {
                  maximum = std_samples[i];
             }
         }
```

```
          epsilon *= maximum;
           for(int i =0;i<num_of_features;i++)
          {
              stdclass0[i]+= epsilon;
              stdclass1[i]+= epsilon;
          }
 ////////////////////////////////////////////////// End of standard
deviation calculation ////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////
 jointProbablity_class1 = 1*(class1*1.0/(samples));
 jointProbablity_class0 = 1*(class0*1.0/(samples));

 ////////////////////////////////////////////////// Prediction
//////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////

/************************************************ This module is
implemented in hardware*********************************
**************************************************************************
************************************************************
**************************************************************************
************************************************************
**************************************************************************
************************************************************
**************************************************************************
************************************************************
*/

/*
```

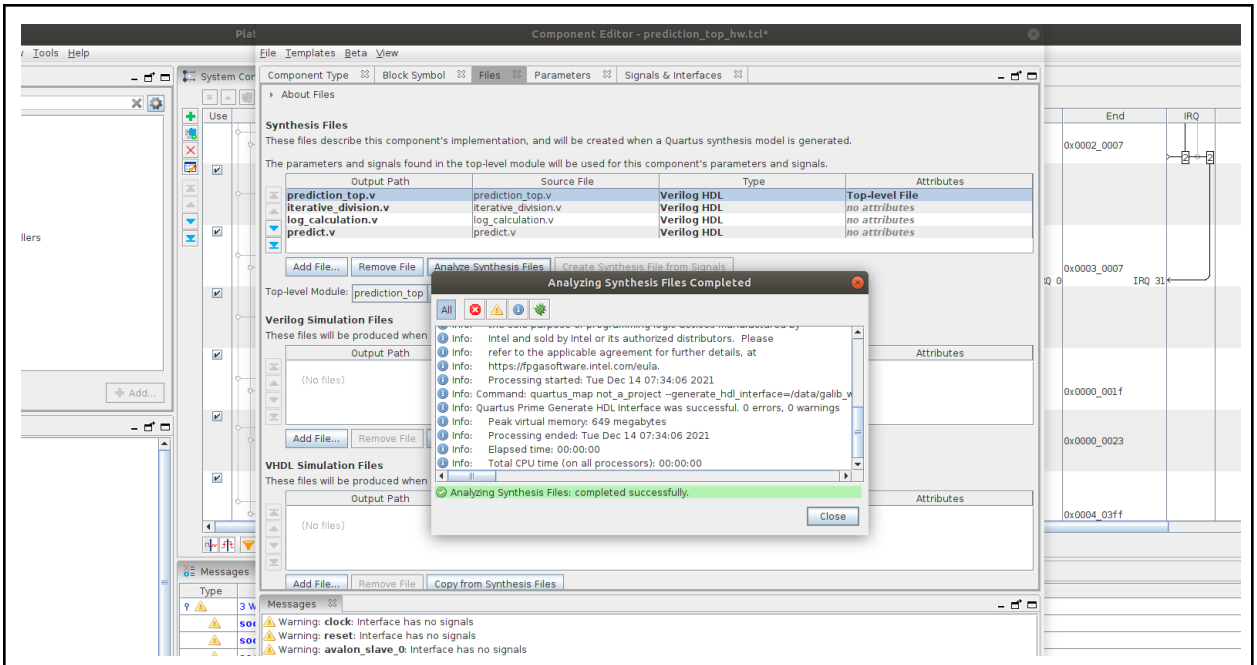
```
int predict(float user_data[],float class1_mean[],float
class1_std[],float class0_mean[],float class0_std[],int
num_of_features,float jointProbability_class0,float
jointProbability_class1){
  float pi=3.14159265358979323846;
 float class0_probability,class1_probability;
 float class0_temp1= 0,class0_temp2=0,class1_temp1=0,class1_temp2=0;
 float x,mean1,sigma1,mean0,sigma0;
 for (int i = 0; i < num_of_features; i++) {
     x = user_data[i];
     sigma1 = class1_std[i];
     class1_temp1 += -(2*pi*sigma1);
     mean1 = class1_mean[i];
     class1_temp2-= (((x - mean1)*(x-mean1))/sigma1);
     sigma0 = class0_std[i];
     class0_temp1 +=-(2*pi*sigma0);
     mean0 = class0_mean[i];
     class0_temp2-= (((x - mean0)*(x-mean0))/sigma0);
 }
 class0_probability=jointProbability_class0+class0_temp1+class0_temp2;
 class1_probability=jointProbability_class1+class1_temp1+class1_temp2;
 if ( class0_probability >class1_probability ) {
     return 0;
 }
 else {
     return 1;
 }
}*/



//////////////////////////////binary string to integer
conversion////////////////////////////////



   int b=frac;
   decimal = b;
   frac = fraction - decimal;
```

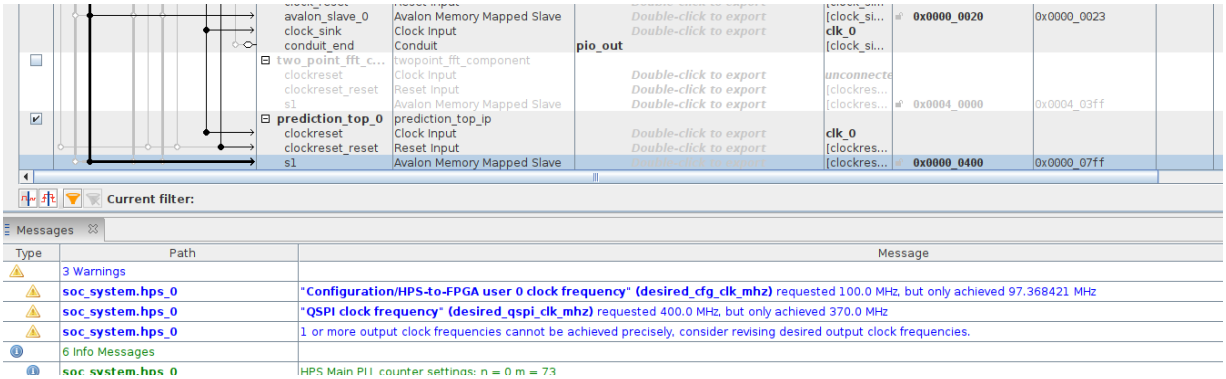
```
   while(decimal != 0)
   {
        rem = decimal%2;
        decimal = decimal/2;
        bnum[i++] = rem+'0';

   }
   bnum[i] = '\0';


   len = strlen(bnum);
   for(int i = 0;i<len/2;i++)
   {
        t = bnum[i];
        bnum[i] = bnum[len-i-1];
        bnum[len-i-1] = t;

   }

   while(counter != 16)
   {
        frac = frac * 2;
        b = frac;
        bnum[len++] = b+'0';
        frac = frac - b;
        counter++;
   }

   bnum[len] = '\0';

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////
```

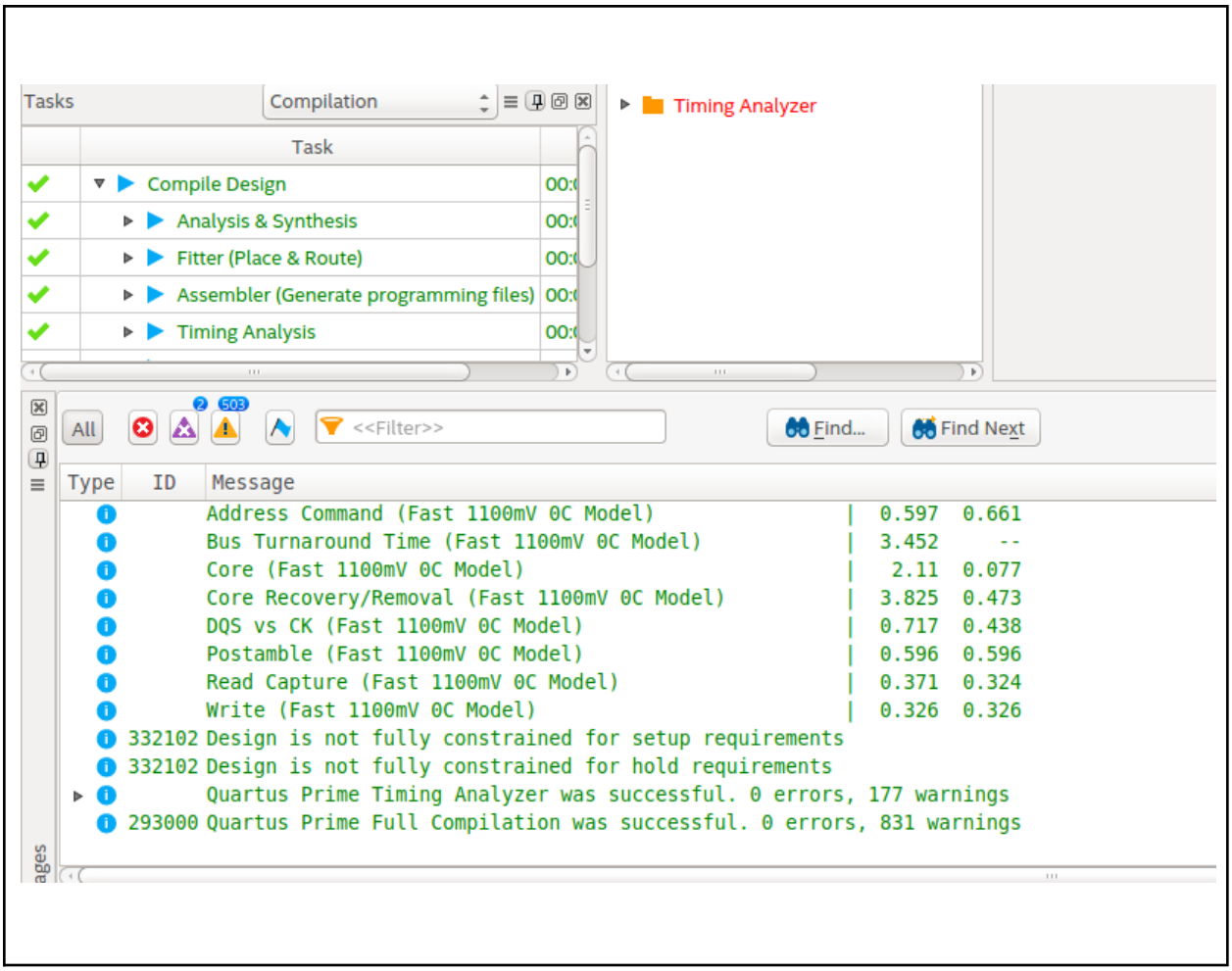
```
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
/////////////////

 void *virtual_base;
 int fd;



 if( ( fd = open( "/dev/mem", ( O_RDWR | O_SYNC ) ) ) == -1 ) {
   printf( "ERROR: could not open \"/dev/mem\"...\n" );
   return( 1 );
 }
 virtual_base = mmap( NULL, HW_REGS_SPAN, ( PROT_READ | PROT_WRITE ),
MAP_SHARED,   fd, HW_REGS_BASE );
 if( virtual_base == MAP_FAILED ) {
   printf( "ERROR: mmap() failed...\n" );
   close( fd );
   return( 1 );
 }

 //void * ptr_dac_db_14b  =  virtual_base + ( ( unsigned long  )(
ALT_LWFPGASLVS_OFST + PIO_DAC_DB_14B_BASE ) & ( unsigned long)(
HW_REGS_MASK ) );
 // Writting 0-9 in the custom component's base address
 for(int i = 0; i<10;i++){
 *(uint32_t *)IO_PTR(REG32_AVALON_INTERFACE_0_BASE) = i;
 printf("i:%8d, CUSTOM: 0x%8x PIO_FEEDBACK: 0x%8x\n", i,
    *(uint32_t *)IO_PTR(REG32_AVALON_INTERFACE_0_BASE),
    *(uint32_t *)IO_PTR(PIO_0_BASE));
 }

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
////////////////////////////////
```

```
//////////////////////////////////////Taking test data from the
user////////////////////////////////////////////////////////////////////////////
////////////////////////////////////
 for(int i = 0;i<num_of_features;i++)
 {
     scanf("%f",&user_data[i]);
 }



 /* End Main Loop */
 // clean up our memory mapping and exit
 if( munmap( virtual_base, HW_REGS_SPAN ) != 0 ) {
   printf( "ERROR: munmap() failed...\n" );
   close( fd );
   return( 1 );
 }
 close( fd );
 return 0;
}
```

Multi-Address, Multi-Cycle Custom Component


Component Editor - prediction_top_hw.tcl*
File Templates Beta View
Component Type
Block Symbol
Files
Parameters
Signals & Interfaces
About Files
Synthesis Files
These files describe this component's implementation, and will be created when a Quartus synthesis model is generated.
The parameters and signals found in the top-level module will be used for this component's parameters and signals.
Output Path
Source File
Type
Attributes
prediction_top.v
prediction_top.v
Verilog HDL
Top-level File
iterative_division.v
iterative_division.v
Verilog HDL
no attributes
log_calculation.v
log_calculation.v
Verilog HDL
no attributes
predict.v
predict.v
Verilog HDL
no attributes
Add File...
Remove File
Analyze Synthesis Files
Create Synthesis File from Signals
Top-level Module:
prediction_top
Verilog Simulation Files
These files will be produced when
(No files)
VHDL Simulation Files
Copy from Synthesis Files
Analyzing Synthesis Files Completed
All
Info: Intel and sold by Intel or its authorized distributors. Please
Info: refer to the applicable agreement for further details, at
Info: https://fpgasoftware.intel.com/eula.
Info: Processing started: Tue Dec 14 07:34:06 2021
Info: Command: quartus_map not_a_project --generate_hdl_interface=/data/galib_v
Info: Quartus Prime Generate HDL Interface was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 649 megabytes
Info: Processing ended: Tue Dec 14 07:34:06 2021
Info: Elapsed time: 00:00:00
Info: Total CPU time (on all processors): 00:00:00
Analyzing Synthesis Files: completed successfully.
Close
Messages
Warning: clock: Interface has no signals
Warning: reset: Interface has no signals
Warning: avalon_slave_0: Interface has no signals
End
IRQ
0x0002_0007
0x0003_0007
0x0000_001f
0x0000_0023
0x0004_03ff


avalon_slave_0
Avalon Memory Mapped Slave
clock_sink
Clock Input
conduit_end
Conduit
pio_out
two_point_fft_c...
twopoint_fft_component
clockreset
Reset Input
s1
Avalon Memory Mapped Slave
prediction_top_0
prediction_top_ip
Double-click to export
clk_0
0x0000_0020
0x0000_0023
0x0004_0000
0x0004_03ff
0x0000_0400
0x0000_07ff
Current filter:
Messages
Type
Path
Message
3 Warnings
soc_system.hps_0
"Configuration/HPS-to-FPGA user 0 clock frequency" (desired_cfg_clk_mhz) requested 100.0 MHz, but only achieved 97.368421 MHz
soc_system.hps_0
"QSPI clock frequency" (desired_qspi_clk_mhz) requested 400.0 MHz, but only achieved 370.0 MHz
soc_system.hps_0
1 or more output clock frequencies cannot be achieved precisely, consider revising desired output clock frequencies.
6 Info Messages
soc_system.hps_0
HPS Main PLL counter settings: n = 0 m = 73


Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
Timing Analysis
Timing Analyzer
All
<<Filter>>
Find...
Find Next
Type
ID
Message
Address Command (Fast 1100mV 0C Model) | 0.597 0.661
Bus Turnaround Time (Fast 1100mV 0C Model) | 3.452 --
Core (Fast 1100mV 0C Model) | 2.11 0.077
Core Recovery/Removal (Fast 1100mV 0C Model) | 3.825 0.473
DQS vs CK (Fast 1100mV 0C Model) | 0.717 0.438
Postamble (Fast 1100mV 0C Model) | 0.596 0.596
Read Capture (Fast 1100mV 0C Model) | 0.371 0.324
Write (Fast 1100mV 0C Model) | 0.326 0.326
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus Prime Timing Analyzer was successful. 0 errors, 177 warnings
293000 Quartus Prime Full Compilation was successful. 0 errors, 831 warnings