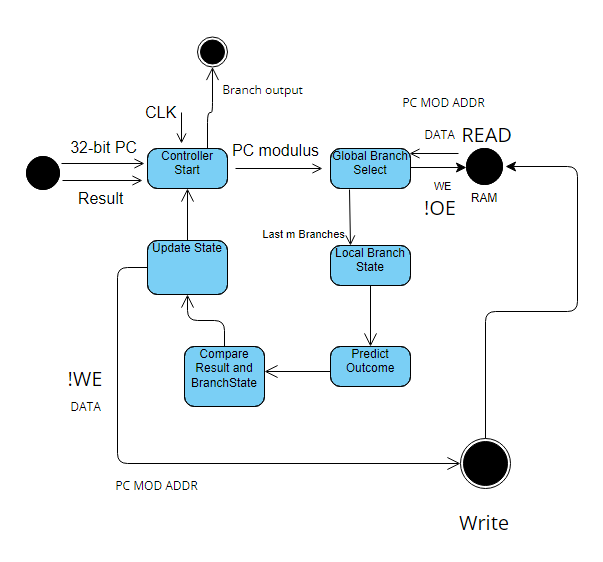
Vince Bengco

CPE 404, SEC 1001

4/20/24

**Project 2 - Correlating Branch Predictor**

**State Diagram**

****

***PARAMETERIZED RAM SYSTEMVERILOG CODE***

module RAM #(parameter m = 2, parameter n = 2 , parameter r = 8)(CS, WE, OE, ADDR, DATA);

input logic CS;

input logic WE;

input logic OE;

input logic [r+m-1:0] ADDR;

inout logic [n-1:0] DATA;

//n wide, 2^ entries, 2^m combinations

reg[n-1:0] RAM1[0:(2\*\*r+m)-1]; //Width RAM1 (2^(r+m)) - 1

assign DATA = (CS == 1'b1 | WE == 1'b0 | OE == 1'b1)? 8'bZZZZZZZZ : RAM1[ADDR]; //read from RAM

always @(negedge WE)

begin

if(CS == 1'b0) begin

RAM1[ADDR] <= DATA;

end

end

endmodule

***PARAMETERIZED RAM TESTBENCH CODE***

module testbench;

logic clk;

logic we;

logic oe;

logic cs;

logic [0:7] addr;

reg [7:0] data\_write;

wire [7:0] data;

wire [7:0] receive;

integer count, rcount;

//reg[7:0] RAM1[0:255];

// instantiate device to be tested

RAM #(.n(8), .m(0), .r(8)) dut(.CS(cs), .WE(we), .OE(oe), .ADDR(addr), .DATA(data));

//RAM dut(.CS(cs), .WE(we), .OE(oe), .ADDR(addr), .DATA(data));

// initialize test

initial

begin

cs <= 1'b0;

we <= 1'b1;

oe <= 1'b0;

rcount <= 0;

count <= 0;

addr <= 8'h00;

end

// generate clock to sequence tests

always

begin

clk <= 1; # 5; clk <= 0; # 5;

end

assign data = (!we) ? data\_write : 8'hZZ;

assign receive = data;

// check results

always @(negedge clk)

begin

we <= 0;

if(count % 2) begin

data\_write <= 8'b01010101;

//RAM1[addr] <= data;

$display("Value in Even Address ", data);

$display("Data Address ", addr);

count = count + 1;

end else

begin

data\_write <= 8'b10101010;

//RAM1[addr] <= data;

$display("Value in Even Address ", data);

$display("Data Address ", addr);

count = count + 1;

end

if(count == 256)begin

count <= 0;

rcount <= rcount + 1;

addr <= 0;

we <= 1'b1;

cs <= 1'b1;

oe <= 1'b1;

end

if((count == 256) & (rcount == 2))

$stop;

end

always @(posedge clk)

begin

if(count == 0)begin

end

else

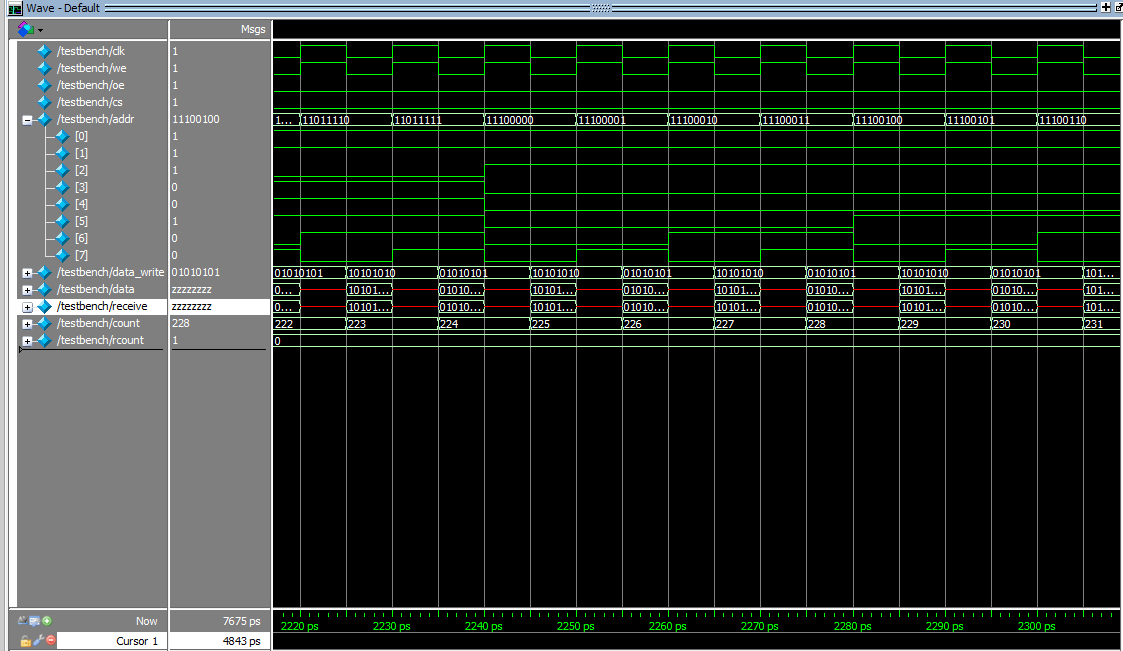
addr <= addr + 1'b1;

we <= 1'b1;

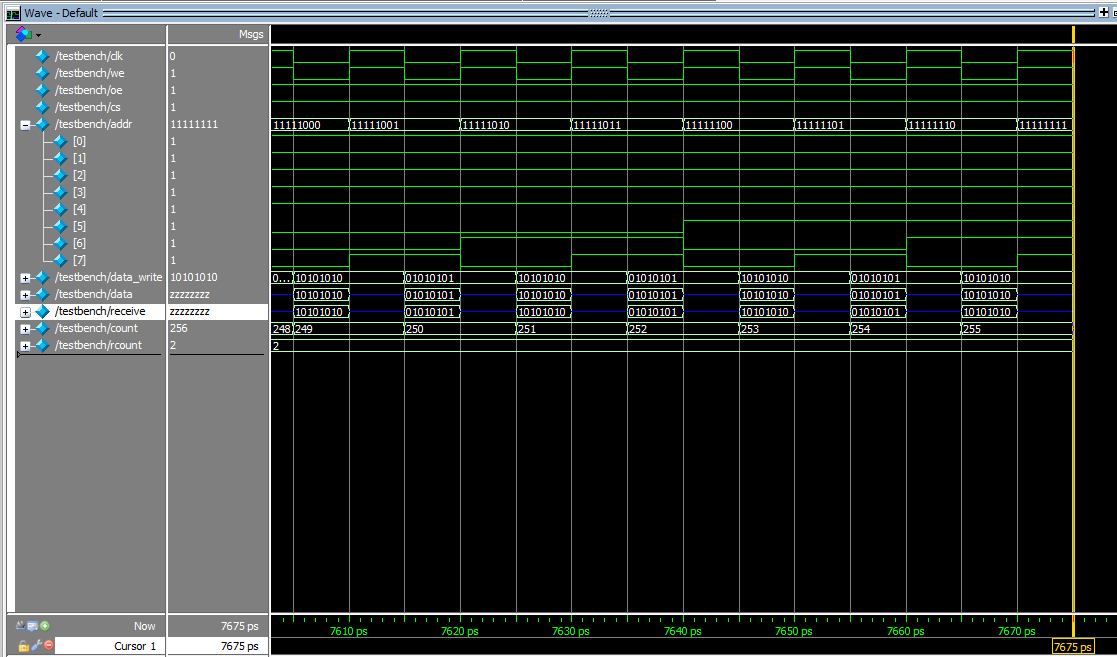
end

endmodule

**Simulation Results (28 x 8) RAM (Paramater n = 8, m =0, r = 8)**

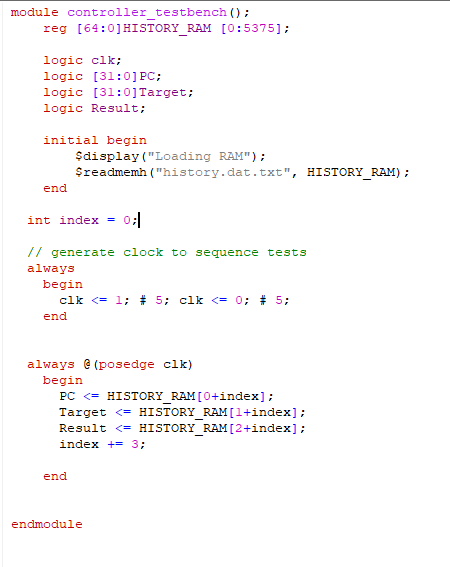


The above simulation shows the testbench simulation for writing alternating 0101010s and 101010s into RAM.

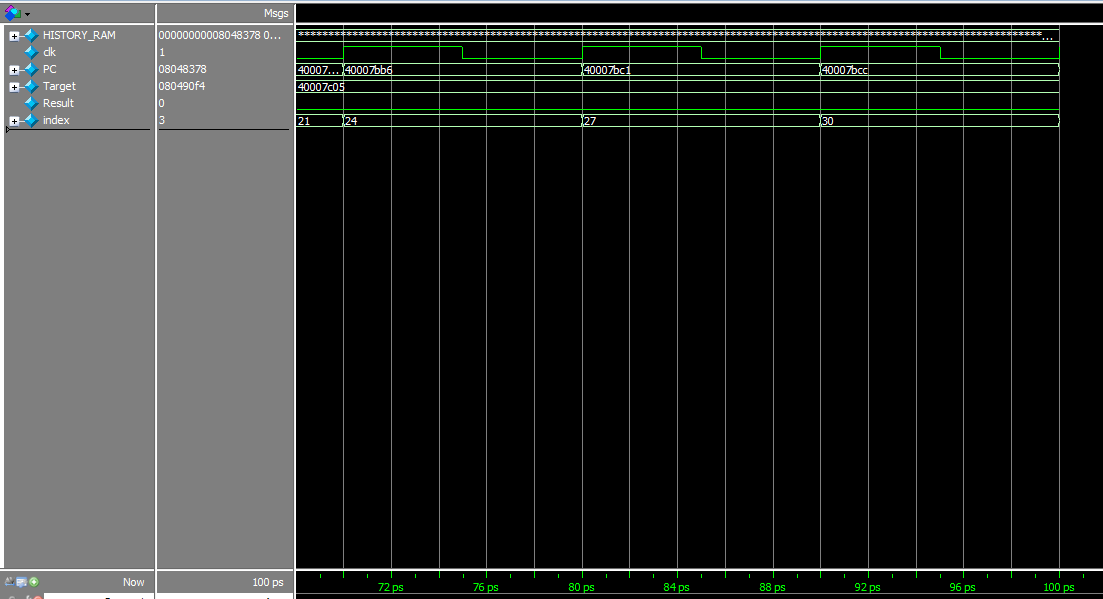


The simulation above shows reading the testbench as shown with the high impedance state of data and receive.

**Controller Testbench Simulation Results**

****

The above controller testbench shows the code for reading the history.dat file into memory.

****

The above simulation shows the testbench is reading the history.dat file correctly.

**Correlating Branch Predictor Code**

module Correlating\_Branch #(parameter m = 2, parameter n = 2 , parameter r = 8)(PC, Target, Result, Clk, Branch);

input logic [31:0] PC;

input logic [31:0] Target;

input logic Result;

input logic Clk;

output logic Branch;

//Taken or Not Taken Branch variable

const int NDIV = n/2;

//Global history

logic [m-1:0] m\_bit; //Sets the branches to zero

int m\_count = 0;

int minus\_one = 0;

int branch\_count=0;

//# of Entries

logic comb\_branch; //Combinational signal for branch logic

logic [r-1:0]PCindex; //The modulus of the PC

//Signals for RAM

logic OE, WE, CS;

logic [r+m-1:0] RAM\_ADDR;

reg [n-1:0]temp\_bit;

//State logic(tristate logic)

reg [n-1:0]n\_bit;

wire [n-1:0]data;

wire [n-1:0]receive;

//instantiate RAM

RAM #(m,n,r)R(CS, WE, OE, RAM\_ADDR, data);

//RAM inout assignments

assign data = (!WE) ? n\_bit : 8'hZZ;

assign receive = data;

always @(posedge Clk) begin

PCindex = PC[r-1:0]; //The masked address of how many entries there are

m\_count = 0;

//Enable Read State

WE = 1'b1;

OE = 1'b0;

CS = 1'b0;

//Check global correlating branches

if(m == 0)begin //Used to test the 0 case of m = 0

RAM\_ADDR = PCindex;

end

else begin

RAM\_ADDR = {PCindex,m\_bit}; //Concatenate the address

end

end

assign Branch = comb\_branch; //Continuous Assignment

always @(negedge Clk) begin

if(m == 0)begin

//Initialize local state

if(receive === 'x) //Need === for case-equality

n\_bit = 2'b00;

else

n\_bit = receive; //Otherwise n\_bit gets the value from RAM

if(n\_bit<NDIV) begin //Less than half of the state logic

comb\_branch = 0;

end

else begin

comb\_branch = 1;

end

if(Branch == Result)

if(n\_bit != 2'b11)

n\_bit++;//Decrement n\_bit state

else begin

if(n\_bit != 2'b00)

n\_bit--;//Decrement n\_bit state

end

end

//2^m combinations

else begin

m\_count=0;

if(receive === 'x) begin

n\_bit = '0;

m\_bit = '1; //Default to logic to all 1's

minus\_one = 1;

end

//Last m branches

else begin

while(m\_count!=m)begin

m\_count++;

m\_bit--;

if(minus\_one == 1)begin

n\_bit = '0;

minus\_one = 0;

break;

end

else begin

n\_bit = receive;

//n\_bit = receive;

/\*if(n\_bit < temp\_bit)

n\_bit = temp\_bit;

temp\_bit = n\_bit;\*/

end

end //End while

end //End else begin

if(n\_bit<NDIV) begin //Less than half of the state logic

comb\_branch = 0;

end

else begin

comb\_branch = 1;

end

if(Branch == Result)

if(n\_bit != 2'b11)

n\_bit++;//Decrement n\_bit state

else begin

if(n\_bit != 2'b00)

n\_bit--;//Decrement n\_bit state

end

end

//Enable Write State

CS = 1'b0;

OE = 1'b0;

WE = 1'b0;

end

always@(m\_bit)begin

if(temp\_bit < NDIV)begin

branch\_count++;

n\_bit = temp\_bit;

end

if(branch\_count == m)begin

m\_bit += branch\_count;

branch\_count = 0;

end

end

endmodule

**64-entry (0,2)**

module controller\_testbench();

reg [64:0]HISTORY\_RAM [0:5375];

logic clk;

logic [31:0]PC;

logic [31:0]Target;

logic Result;

logic branch;

initial begin

$display("Loading RAM");

$readmemh("history.dat.txt", HISTORY\_RAM);

end

int index = 0;

int accuracy = 0;

parameter int m = 0;

parameter int n = 2;

parameter int r = 6;

Correlating\_Branch #(m,n,r)CB (PC,Target,Result,clk,branch);

// generate clock to sequence tests

always

begin

clk <= 1; # 5; clk <= 0; # 5;

end

always @(posedge clk)

begin

PC <= HISTORY\_RAM[0+index];

Target <= HISTORY\_RAM[1+index];

Result <= HISTORY\_RAM[2+index];

index += 3;

if(Result == branch)

accuracy++;

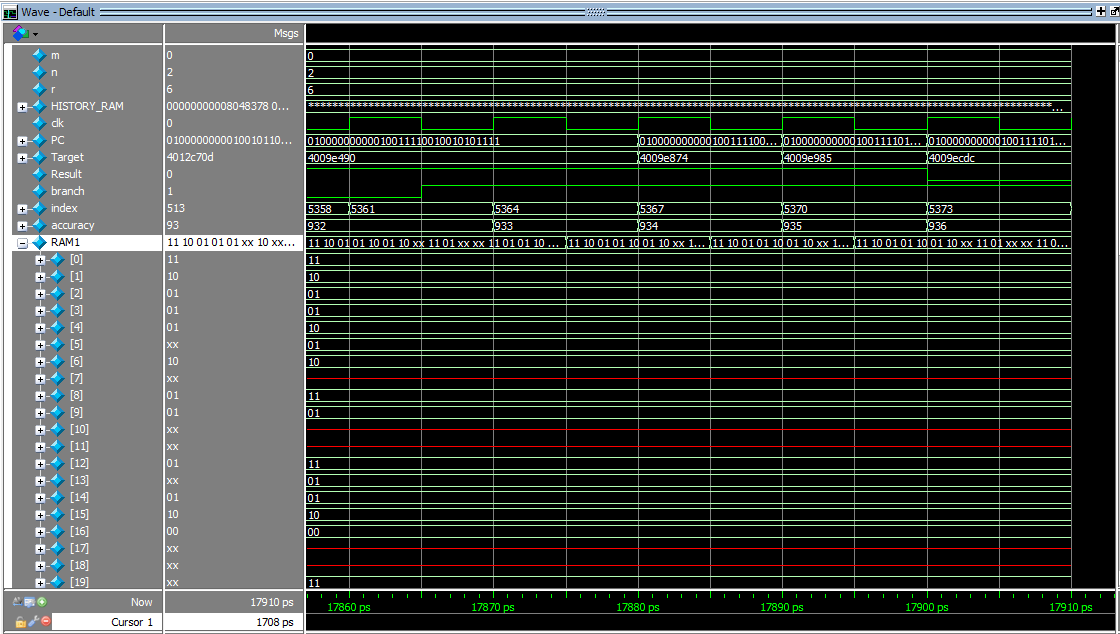
if(index > 5375)begin

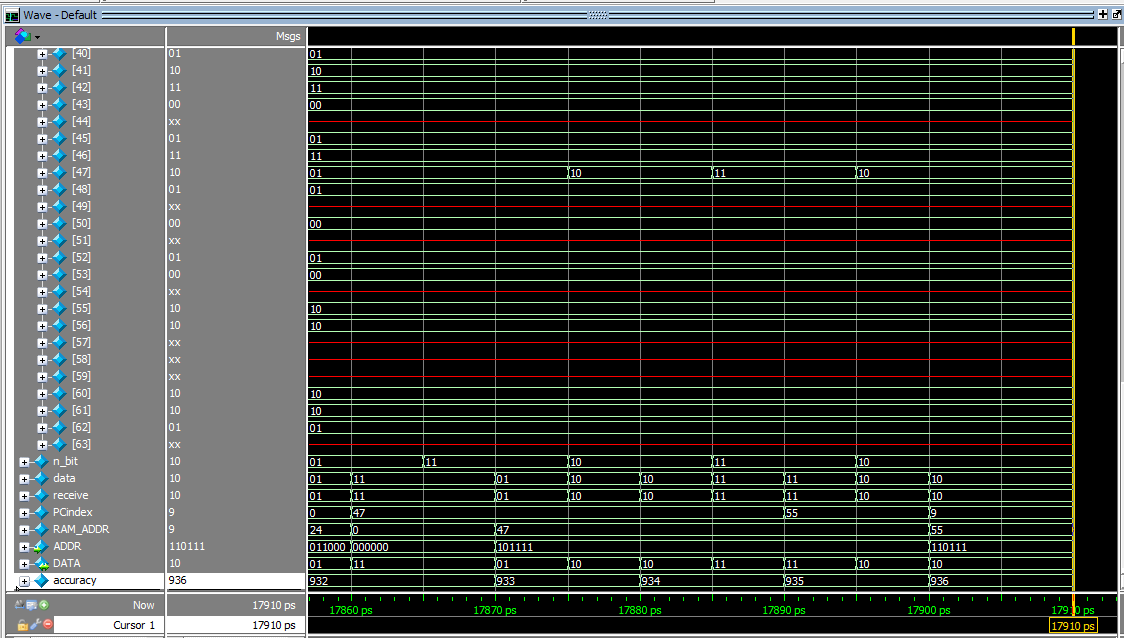
$stop;

end

end

endmodule

****

****

Accuracy is 936/1792 =0.522 = 52.2% Predicted Taken

**128-entry (0,2)**

module controller\_testbench();

reg [64:0]HISTORY\_RAM [0:5375];

logic clk;

logic [31:0]PC;

logic [31:0]Target;

logic Result;

logic branch;

initial begin

$display("Loading RAM");

$readmemh("history.dat.txt", HISTORY\_RAM);

end

int index = 0;

int accuracy = 0;

parameter int m = 0;

parameter int n = 2;

parameter int r = 7;

Correlating\_Branch #(m,n,r)CB (PC,Target,Result,clk,branch);

// generate clock to sequence tests

always

begin

clk <= 1; # 5; clk <= 0; # 5;

end

always @(posedge clk)

begin

PC <= HISTORY\_RAM[0+index];

Target <= HISTORY\_RAM[1+index];

Result <= HISTORY\_RAM[2+index];

index += 3;

if(Result == branch)

accuracy++;

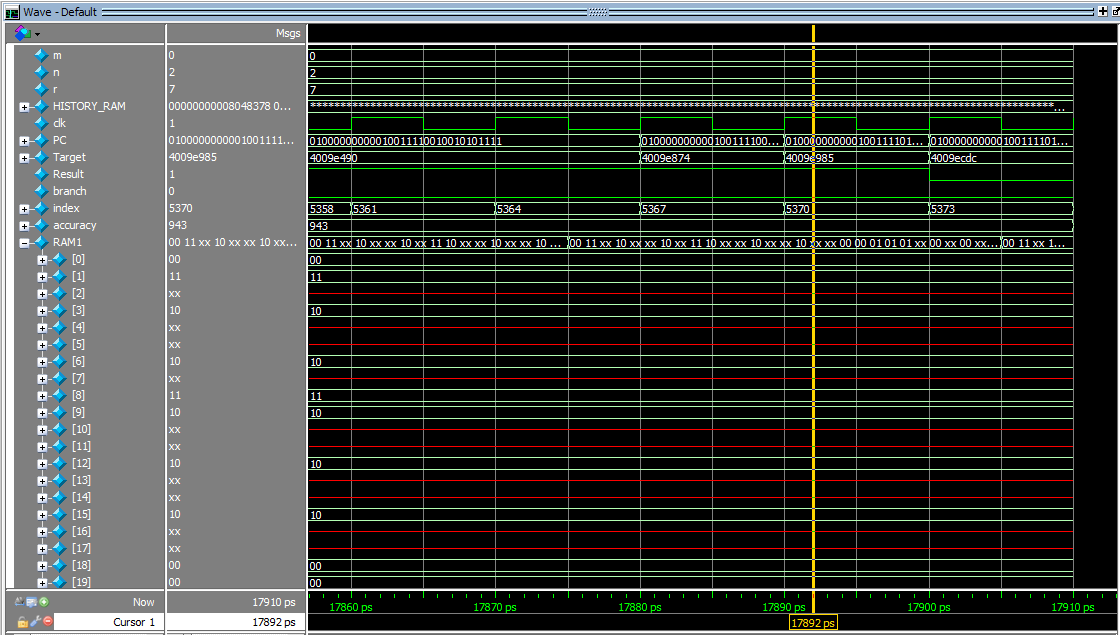
if(index > 5375)begin

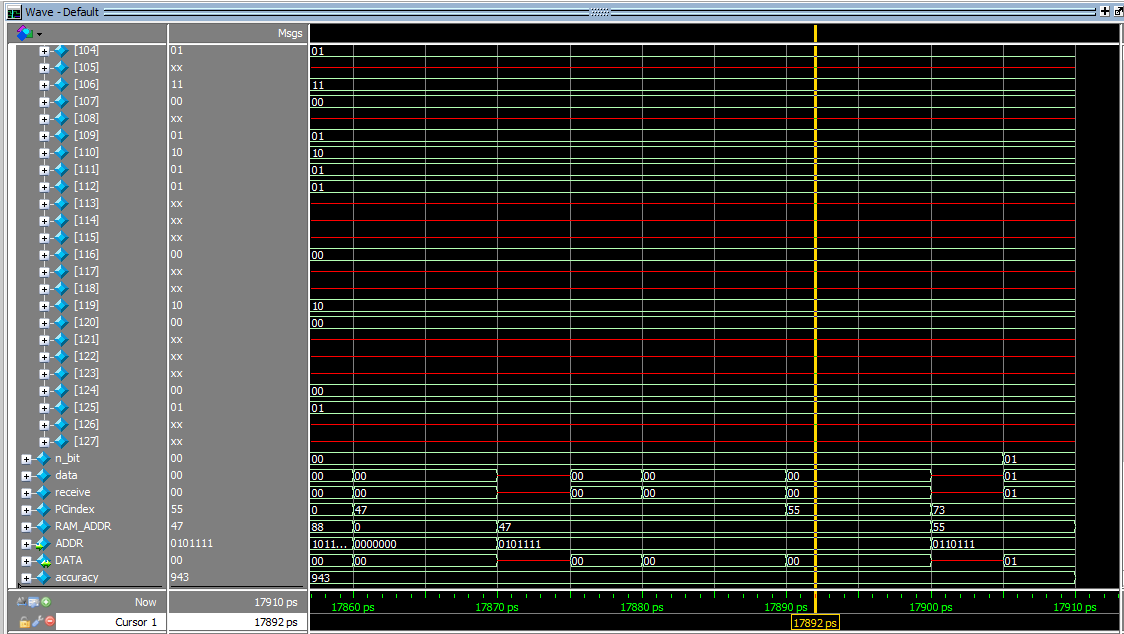
$stop;

end

end

endmodule

****

****

Accuracy is 944/1792 = 0.527 = 52.7% Predicted Taken

**256-entry (0,2)**

module controller\_testbench();

reg [64:0]HISTORY\_RAM [0:5375];

logic clk;

logic [31:0]PC;

logic [31:0]Target;

logic Result;

logic branch;

initial begin

$display("Loading RAM");

$readmemh("history.dat.txt", HISTORY\_RAM);

end

int index = 0;

int accuracy = 0;

parameter int m = 0;

parameter int n = 2;

parameter int r = 8;

Correlating\_Branch #(m,n,r)CB (PC,Target,Result,clk,branch);

// generate clock to sequence tests

always

begin

clk <= 1; # 5; clk <= 0; # 5;

end

always @(posedge clk)

begin

PC <= HISTORY\_RAM[0+index];

Target <= HISTORY\_RAM[1+index];

Result <= HISTORY\_RAM[2+index];

index += 3;

if(Result == branch)

accuracy++;

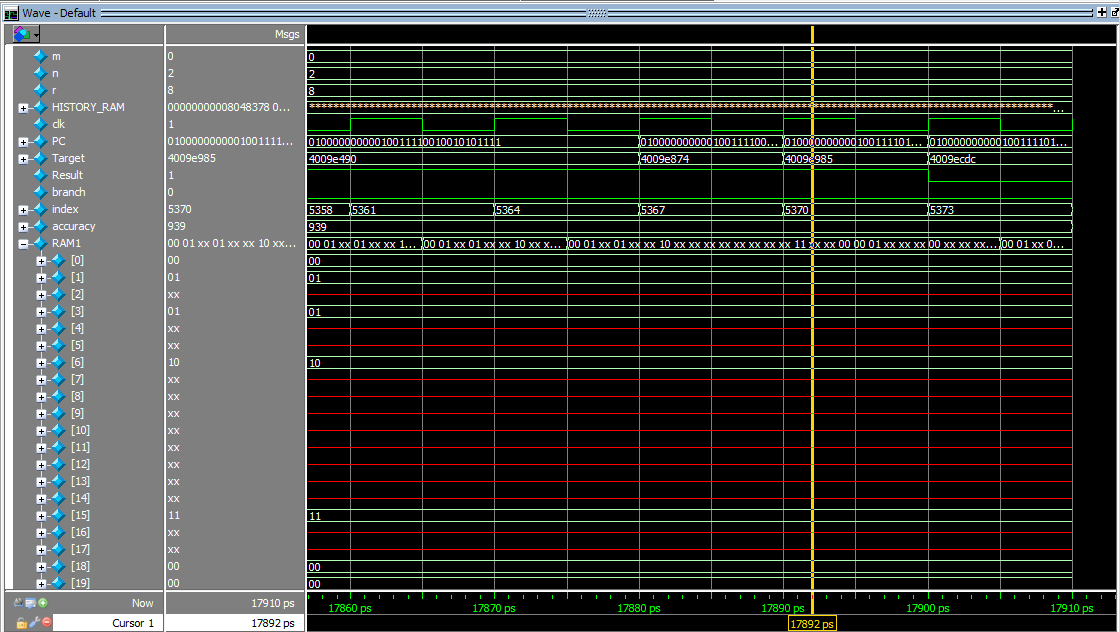
if(index > 5375)begin

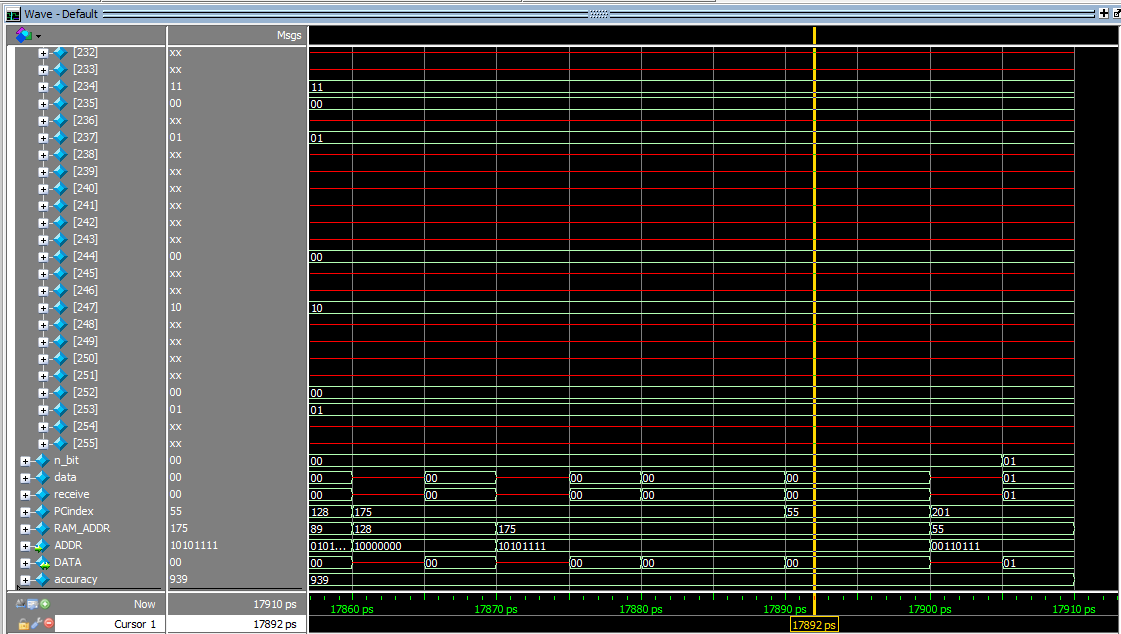
$stop;

end

end

endmodule





Accuracy is 940/1792 =0.525 = 52.5% Predicted Taken

**64-entry (2,2)**

module controller\_testbench();

reg [64:0]HISTORY\_RAM [0:5375];

logic clk;

logic [31:0]PC;

logic [31:0]Target;

logic Result;

logic branch;

initial begin

$display("Loading RAM");

$readmemh("history.dat.txt", HISTORY\_RAM);

end

int index = 0;

int accuracy = 0;

parameter int m = 2;

parameter int n = 2;

parameter int r = 6;

Correlating\_Branch #(m,n,r)CB (PC,Target,Result,clk,branch);

// generate clock to sequence tests

always

begin

clk <= 1; # 5; clk <= 0; # 5;

end

always @(posedge clk)

begin

PC <= HISTORY\_RAM[0+index];

Target <= HISTORY\_RAM[1+index];

Result <= HISTORY\_RAM[2+index];

index += 3;

if(Result == branch)

accuracy++;

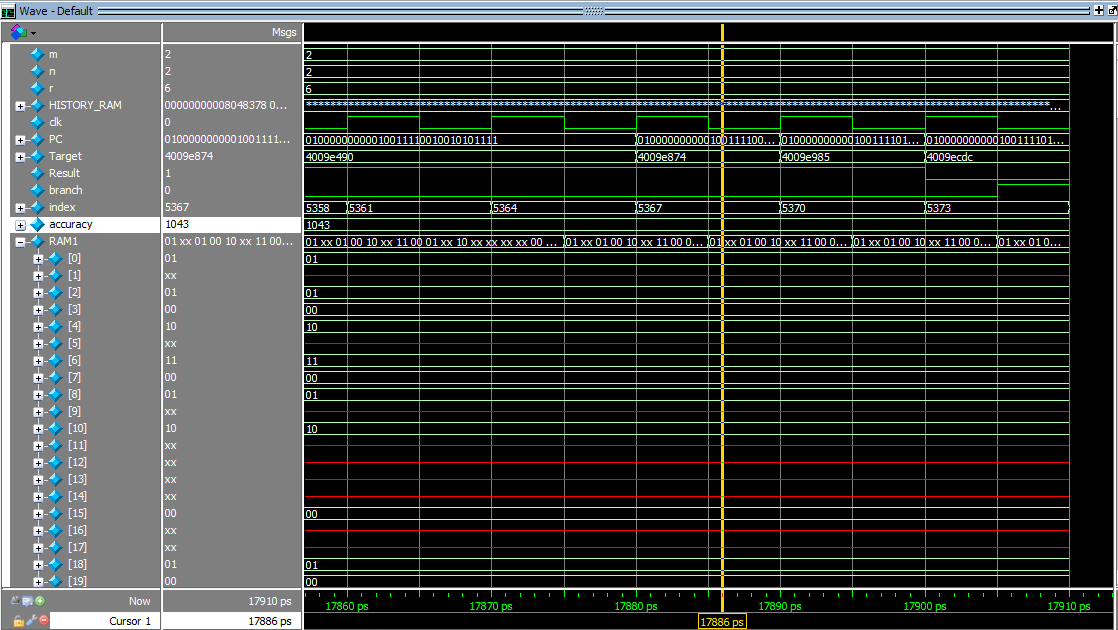
if(index > 5375)begin

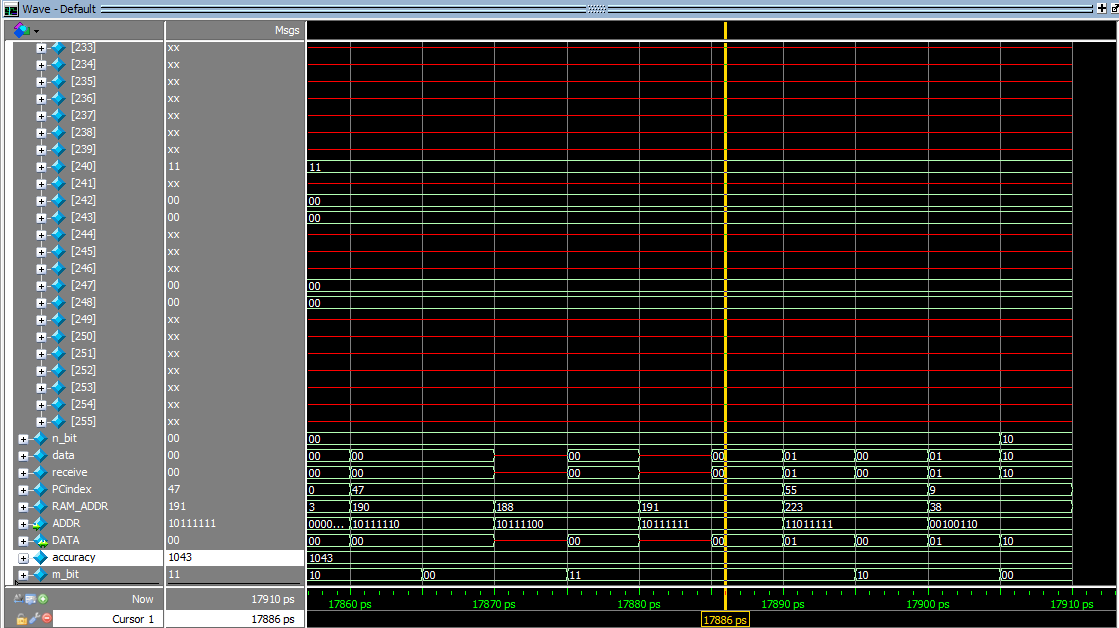
$stop;

end

end

endmodule

**’**

****

Accuracy is 1043/1792 =0.582 = 58.2%

**128-entry (2,2)**

module controller\_testbench();

reg [64:0]HISTORY\_RAM [0:5375];

logic clk;

logic [31:0]PC;

logic [31:0]Target;

logic Result;

logic branch;

initial begin

$display("Loading RAM");

$readmemh("history.dat.txt", HISTORY\_RAM);

end

int index = 0;

int accuracy = 0;

parameter int m = 2;

parameter int n = 2;

parameter int r = 7;

Correlating\_Branch #(m,n,r)CB (PC,Target,Result,clk,branch);

// generate clock to sequence tests

always

begin

clk <= 1; # 5; clk <= 0; # 5;

end

always @(posedge clk)

begin

PC <= HISTORY\_RAM[0+index];

Target <= HISTORY\_RAM[1+index];

Result <= HISTORY\_RAM[2+index];

index += 3;

if(Result == branch)

accuracy++;

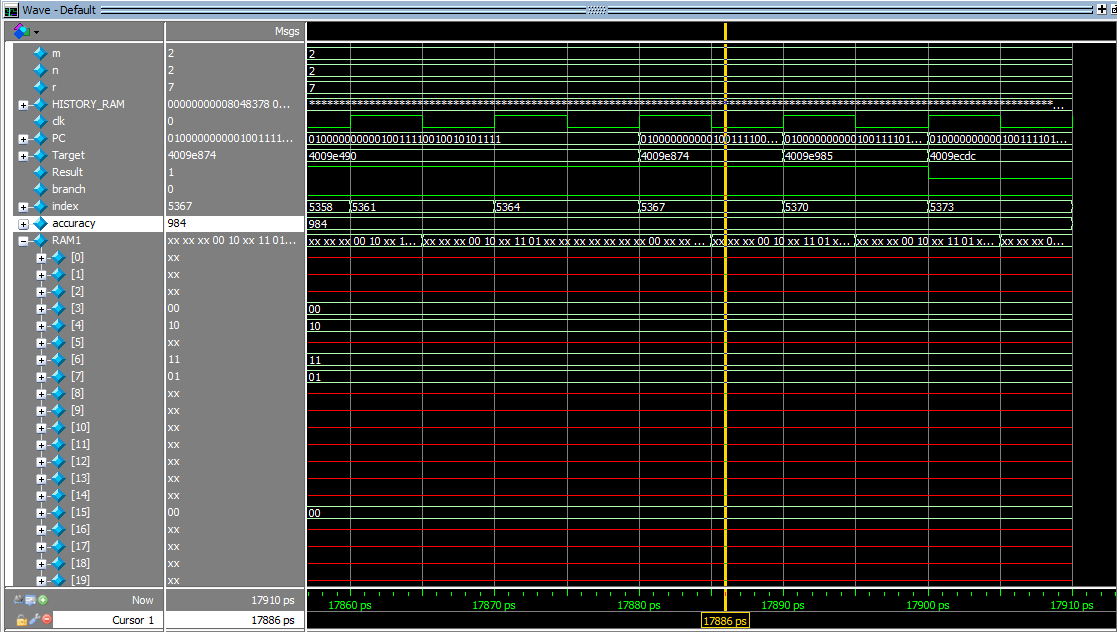
if(index > 5375)begin

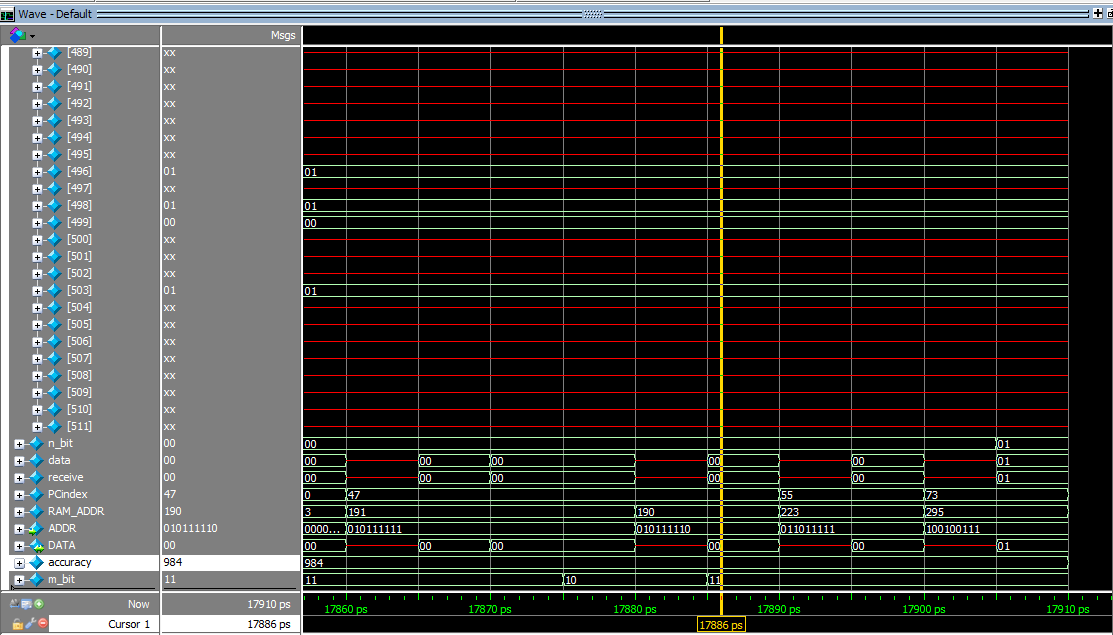
$stop;

end

end

endmodule

****

****

Accuracy is 984/1792 = 0.549 =54.9%

**256-entry (2,2)**

module controller\_testbench();

reg [64:0]HISTORY\_RAM [0:5375];

logic clk;

logic [31:0]PC;

logic [31:0]Target;

logic Result;

logic branch;

initial begin

$display("Loading RAM");

$readmemh("history.dat.txt", HISTORY\_RAM);

end

int index = 0;

int accuracy = 0;

parameter int m = 2;

parameter int n = 2;

parameter int r = 8;

Correlating\_Branch #(m,n,r)CB (PC,Target,Result,clk,branch);

// generate clock to sequence tests

always

begin

clk <= 1; # 5; clk <= 0; # 5;

end

always @(posedge clk)

begin

PC <= HISTORY\_RAM[0+index];

Target <= HISTORY\_RAM[1+index];

Result <= HISTORY\_RAM[2+index];

index += 3;

if(Result == branch)

accuracy++;

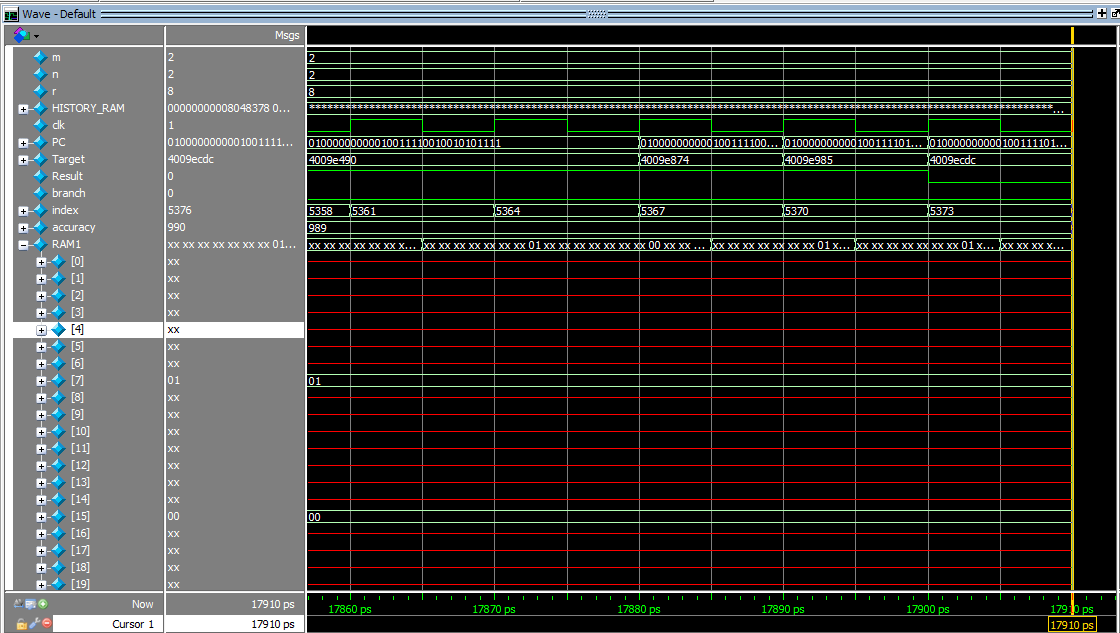
if(index > 5375)begin

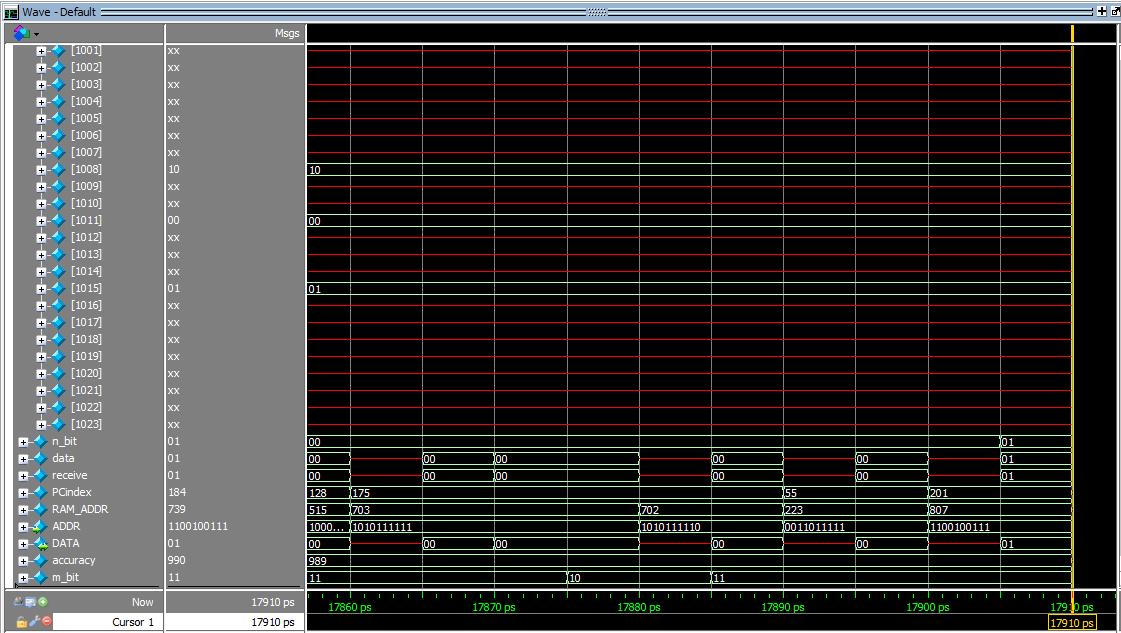
$stop;

end

end

endmodule

****

****

Accuracy is 990/1792 = 0.552 = 55.2%