



SPRING 17


ECE 585
ADVANCED COMPUTER
ARCHITECTURE

PROJECT REPORT

HDL IMPLEMENTATION OF
DYNAMIC BRANCH PREDICTORS

**NAVANEETH GOWDA
THANDAVAMURTHY**

A20378763



ABSTRACT

The control hazard which is one of the key reason for the processor stall is overcome by the implementation of the branch predictors which will eliminate the limit on fetch cycle. Branch prediction is a form of the data prediction that attempts to predict branch results. Once the prediction is made the processor speculatively execute the instructions based on the outcome of the prediction. In the project the dynamic 1-bit, 2-bit and (4,2) dynamic predictors are realized using VHDL, a hardware description language and the number of miss-predictions are analysed to know the efficiency of the predictor.

INTRODUCTION

In recent processors, the performance is of the crucial importance which depends on the pipeline stage, stalls in the pipeline due to the data hazard, control hazard or due the memory miss. The stalls in the pipeline deteriorates the performance of the processor and it must be overcome. The data hazards can be overcome by using the forwarding buffers and the stalls due to memory can be overcome by the efficient design. The control hazards which are mainly caused due to the branch miss predictions and to overcome this, branch predictors are used. There are some schemes proposed for the implementation of the branch predictors among which static and dynamic predictors are prominent.

Static branch predictors predict the current branch based on the information gathered before the program execution and dynamic predictors stores the branch outcomes in the prediction buffer and based in that the current branch is predicted. Dynamic predictors include correlator predictors and tournament predictors of which one bit, two bit and (4,2) bit predictors are implemented in this project.

BACKGROUND

The pipeline implementation in the process faced many challenges when it was started to implement in the processor. The major issue was pertaining to the pipeline stall which can halt the processor. The stalls were mainly resulted due to the data hazards, control hazards and the hardware hazards. The main reason for the control hazard is the branch instruction due to which the processor has to be stalled till the branch outcome was known.

To overcome this delayed branches and other techniques were implemented. But the efficient implementations were the branch predictors which predicts the next instructions to be fetched which eliminates the restrictions on the fetch cycle.

Many predictors are proposed among which the dynamic predictors are implemented in the project. The dynamic predictors predict the branch outcome dynamically during the program execution. For the purpose of understanding the predictor implementation, one bit predictor, two-bit predictor and the (4,2) correlating predictors are considered.

ARCHITECTURAL EXPLORATION BRANCH PREDICTORS

1. One bit predictor:

The one-bit predictor uses the one bit prediction values stored in the branch history table. The lower bits of the PC address determine the address of the branch history table and based on the value in the table branch decision is made based on which the next instruction is fetched. The implementation of the one bit predictor is as shown in the figure below:

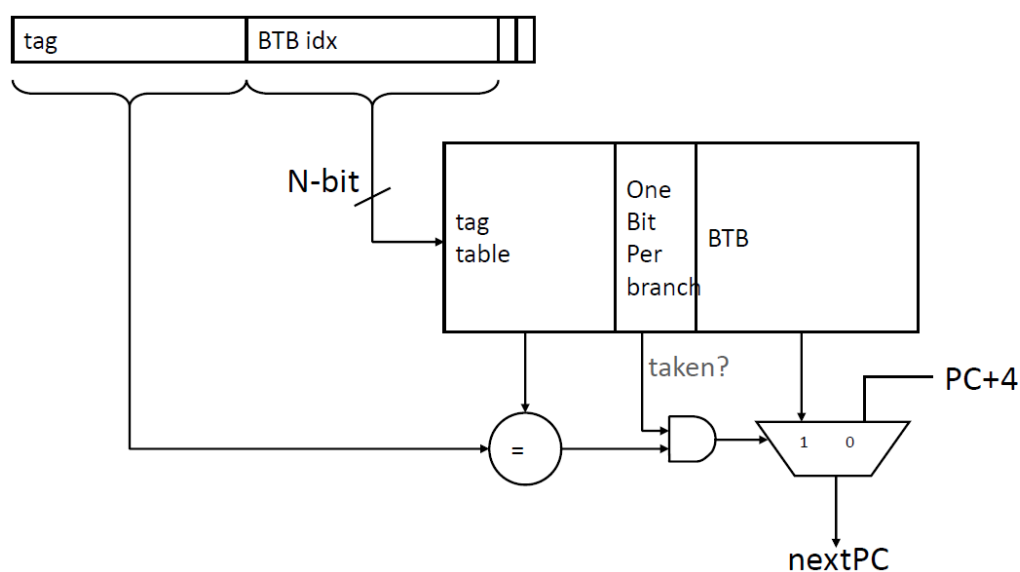


Fig1: Architectural implementation of 1-bit predictor

2. Two-bit predictor:

In two-bit predictor, each branch is associated with the saturating two bit counters. A single outcome cannot change the strong prediction but requires more hardware. The lower bits of the PC is used to search through the branch history table which is associated with the two bit saturating counter. The two-bit saturating counter is as shown in the figure below;

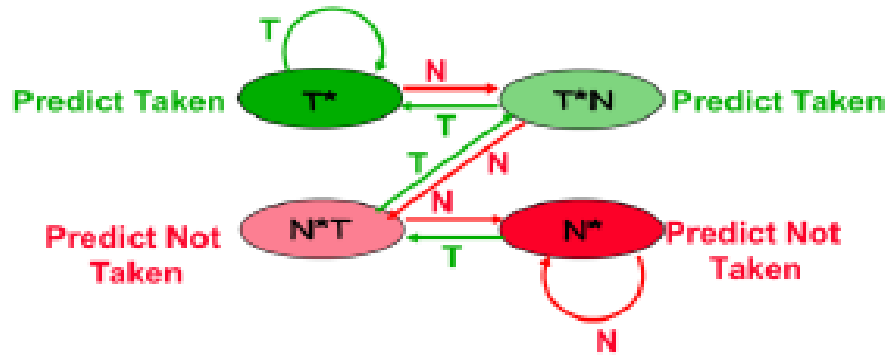


Fig2: Two-bit saturating counter

In this case, each entry in the branch history table is a saturating counter.

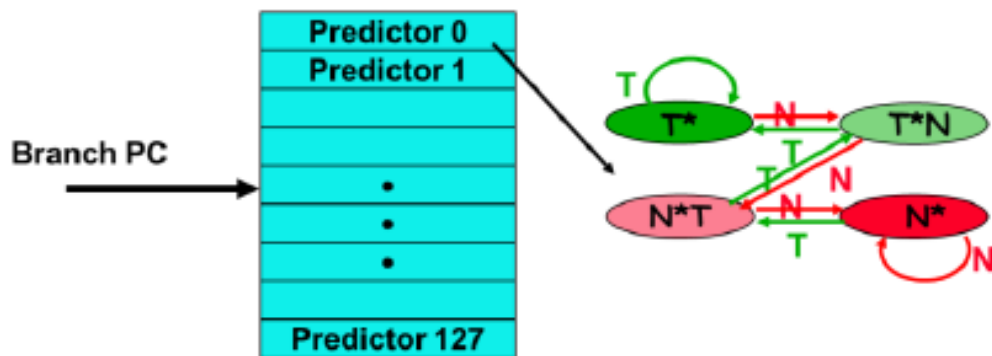


Fig3: Branch history table for 2-bit predictors

In this case, the branch prediction stored in the table is changed to T or not taken based on the two recent branch outcomes thus, two miss prediction can only change the state of the counter.

3. (4,2) Correlator predictor:

A (m,n) correlator predictor keeps track of last m branches based on which the current prediction is made. (4,2) correlator predictor stores the last four branches and the outcome of the current branch is based on the 2 bit data stored in the selected address. The bit history is implemented using the 4-bit shift register and there will be

16 selections where the branch predictions will be stored. The current prediction is selected among the 16 entries and branch result is predicted. Thus, each address has a 16-two-bit counter that stores the branch prediction value. A (2,n) predictor is as shown in the figure below:

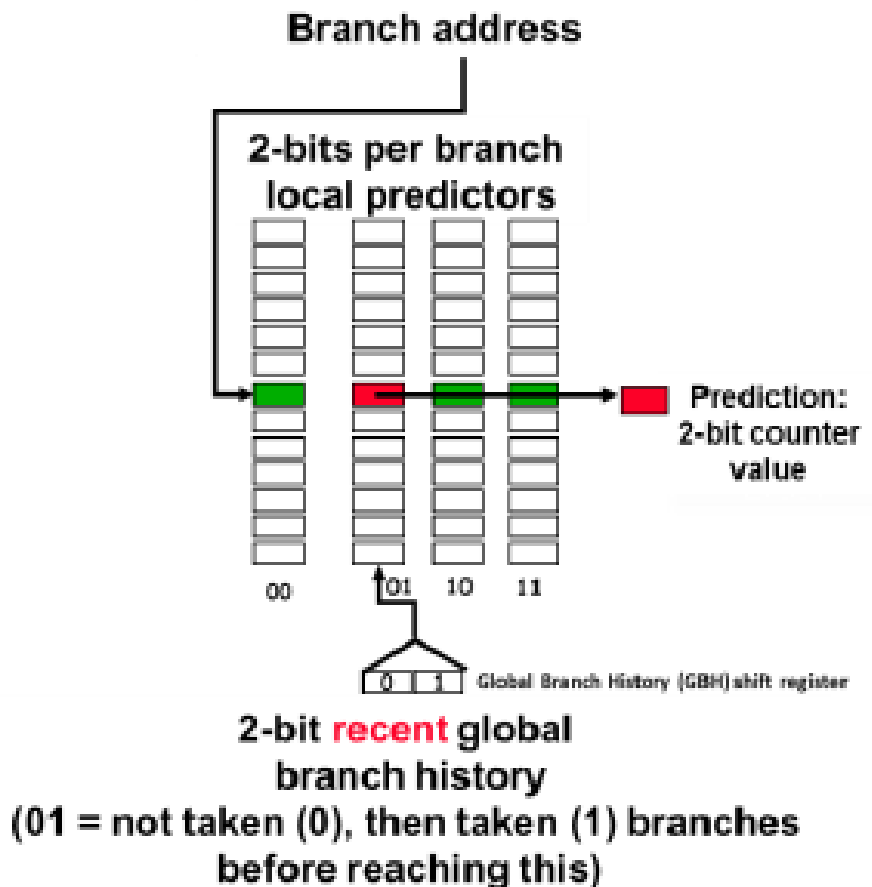


Fig4(2,n) bit predictor

In case of a (4,2) predictor there are 16 branch predictions based on which the current branch depends.

FUNCTIONAL VALIDATION AND VERIFICATION

The functioning of the branch predictors is determined by testing it against a sequence of the branch outcomes along with which the miss predictions are calculated to determine the performance of the branch predictions.

A loop is considered to validate the functions of predictors designed. The loop considered is shown in the below figure:

```
int c;  
int main () {  
    int i, j;  
    for (i=0; i<1000; i++) {  
        for (j=0; j<4; j++) {  
            c++;  
        }  
    }  
    return c;  
}
```

Fig5: Test program used for the validation of the predictors

The above program implementation in the MIPS is as shown below;

main:

move \$t0, 0 ; c = 0;beginning of outer loop

outer_loop_block: move \$t1, 0 ; for(int i = 0, _, _) i outer_loop_condition

inner_loop_block: move \$t2, 0 ; for(int j = 0, _, _);inner loop start:j
inner_loop_condition

add \$t0, \$t0, 1; c++

add \$t2, \$t2, 1; for(_, _ j++)

bne \$t2, 4, inner_loop_block ; for(_, j < 40 _)inner_loop_condition:

;inner loop end

add \$t1, \$t1, 1; for(_, _ j++)outer_loop_condition:

bne \$t1, 1000, outer_loop_block ; for(_, i < 1000, _); outer loop end

move \$v0, \$t0; return c

jr \$ra

In the above MIPS code, the branch to the outer loop takes place only when j>4.

The branch outcomes of the loop is generated using a script in the C program and it is given as input to the predictors. Based on which the branch predictors predict the outcome and the miss predictions.

The script considered is given in the appendix.

RESULTS

1. One-bit predictor:

The one bit predictor predicts the current branch based on the last prediction result that is stored in the branch history table. For the given C code, the simulation and the miss predictions are as shown below:

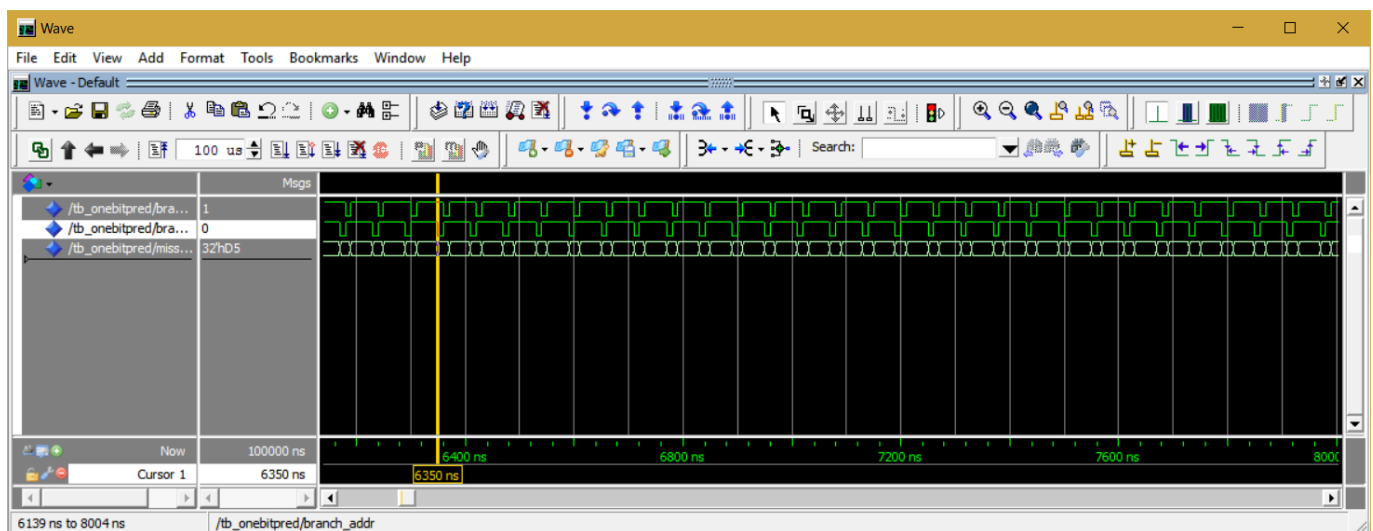


Fig6: simulation of one_bit predictor.

number of miss predictions = **X7D2 = 2002**.

2. Two-bit predictor:

The two-bit predictor is implemented using the two-bit saturating counter and the simulations for this implementation is as shown in the figure below:

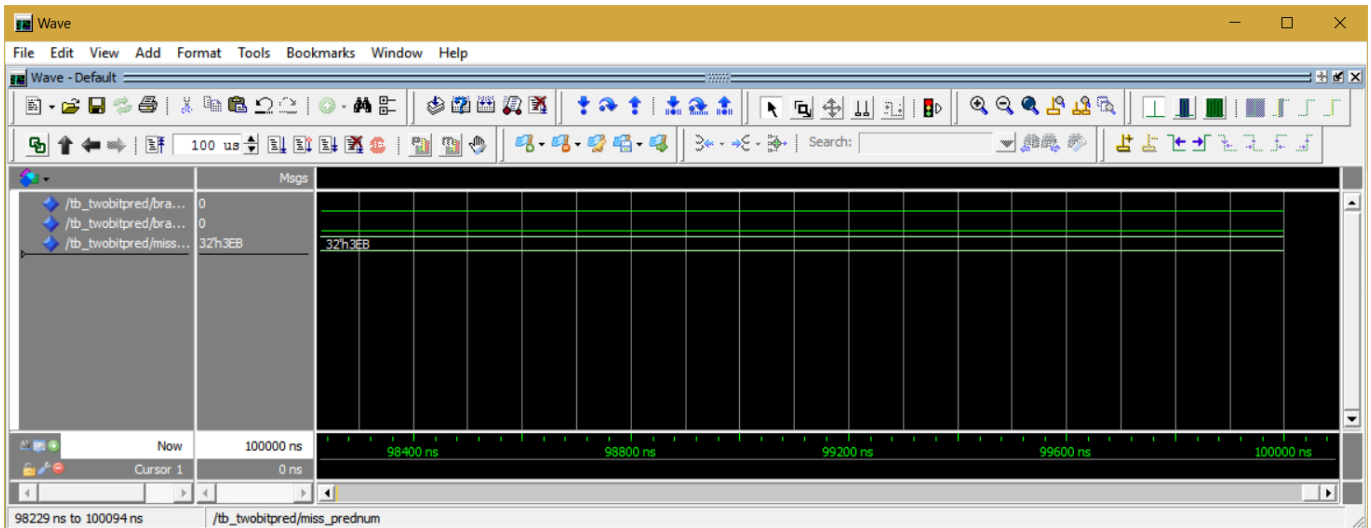
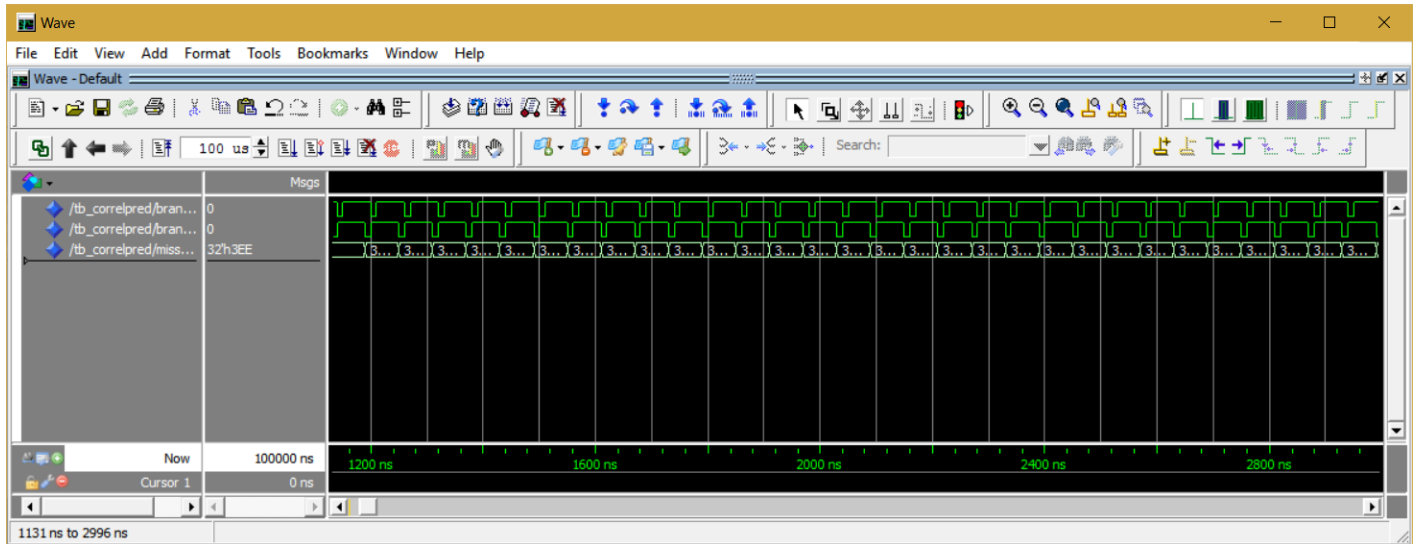


Fig7: simulation of two-bit predictor.

Number of Miss predictions = **X3EB=1003**

3. (4,2) correlating predictor:

This correlating predictor has the 16 choices for each branch address based on the four recent branch outcomes stored in the shift register. The number of the miss predictions and the simulation results are shown below:



From the simulation, it can be found that the number of miss predictions is **X3EE equal to 1006**, obtained based on the initial prediction set as 11 in the counter.

According to the above miss prediction values, the 2-bit predictor is a good predictor for the given loop. The correlator predictor may function well when large number of branches are considered.

Table: Miss predictions of different predictors

Predictor	One bit predictor	Two bit predictor	(4,2) correlator predictor
Miss_predictions	2002	1003	1006

CONCLUSION AND FUTURE WORK

The branch predictors are studied and implemented using VHDL to understand operation on the hardware implementation. The tournament predictors can be considered for the future work.

REFERENCES

- Computer Architecture Text book: John L hennessy and david patterson
- <https://www.cs.umd.edu/class/spring2012/cmsc411/lectures/lec09.pdf>
- <https://people.eecs.berkeley.edu/~kubitron/courses/cs252-S09/lectures/lec09-prediction2.pdf>
- https://blackboard.iit.edu/bbcswebdav/pid-497539-dt-content-rid-3139340_1/courses/XLSEX201720/ECE%20585%20Final%20Project%20Spring%202017.pdf (project description)

APPENDIX

Source code

```
#include <stdio.h>

#include <stdlib.h>

int main()
{
    int i,j;
    FILE * fp;

    fp = fopen ("file.txt", "w+");
    for (i=0;i<1000;i++){
        fprintf(fp, "branch_addr <= '0';" "\n");
        fprintf(fp, "branch_pred <= '1';" "\n" );
        fprintf(fp, "wait for 10 ns;" "\n" );

        for (j=0;j<4;j++){
            fprintf(fp, "branch_addr <= '1';" "\n");
            fprintf(fp, "branch_pred <= '1';" "\n");
            fprintf(fp, "wait for 10 ns;" "\n" );
        }
        fprintf(fp, "branch_addr <= '1';" "\n");
    }
```

```

    fprintf(fp, "branch_pred <= '0';" "\n");
    fprintf(fp, "wait for 10 ns;" "\n" );
}

fprintf(fp, "branch_addr <= '0';" "\n");
fprintf(fp, "branch_pred <= '0';" "\n");
fprintf(fp, "wait for 10 ns;" "\n" );

fclose(fp);

return(0);
}

```

Correlating pred src code

```

library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_bit.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use ieee.std_logic_unsigned.all;

```

```

entity correlatorpredictor4_2 is
port(branch_address: in bit;
branch_prediction: in std_logic;

```

```
miss_prediction:out integer);  
end correlatorpredictor4_2;
```

architecture behavioral of correlatorpredictor4_2 is

```
signal previous00:std_logic_vector(1 downto 0):="11";  
signal previous01:std_logic_vector(1 downto 0):="11";  
signal previous02:std_logic_vector(1 downto 0):="11";  
signal previous03:std_logic_vector(1 downto 0):="11";  
signal previous04:std_logic_vector(1 downto 0):="11";  
signal previous05:std_logic_vector(1 downto 0):="11";  
signal previous06:std_logic_vector(1 downto 0):="11";  
signal previous07:std_logic_vector(1 downto 0):="11";  
signal previous08:std_logic_vector(1 downto 0):="11";  
signal previous09:std_logic_vector(1 downto 0):="11";  
signal previous010:std_logic_vector(1 downto 0):="11";  
signal previous011:std_logic_vector(1 downto 0):="11";  
signal previous012:std_logic_vector(1 downto 0):="11";  
signal previous013:std_logic_vector(1 downto 0):="11";  
signal previous014:std_logic_vector(1 downto 0):="11";  
signal previous015:std_logic_vector(1 downto 0):="11";
```

```

signal previous10:std_logic_vector(1 downto 0):="11";
signal previous11:std_logic_vector(1 downto 0):="11";
signal previous12:std_logic_vector(1 downto 0):="11";
signal previous13:std_logic_vector(1 downto 0):="11";
signal previous14:std_logic_vector(1 downto 0):="11";
signal previous15:std_logic_vector(1 downto 0):="11";
signal previous16:std_logic_vector(1 downto 0):="11";
signal previous17:std_logic_vector(1 downto 0):="11";
signal previous18:std_logic_vector(1 downto 0):="11";
signal previous19:std_logic_vector(1 downto 0):="11";
signal previous110:std_logic_vector(1 downto 0):="11";
signal previous111:std_logic_vector(1 downto 0):="11";
signal previous112:std_logic_vector(1 downto 0):="11";
signal previous113:std_logic_vector(1 downto 0):="11";
signal previous114:std_logic_vector(1 downto 0):="11";
signal previous115:std_logic_vector(1 downto 0):="11";

signal global_history:std_logic_vector(3 downto 0):="1111";
signal miss_pred:integer:=0;
signal temp_global: std_logic_vector(3 downto 0);

```



```

begin

process(branch_address,branch_prediction,global_history)
begin
if(branch_address='0') then
case global_history is

when "0000" =>
case previous00 is

when "00" =>
if branch_prediction = '1' then
previous00 <= previous00 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous00 <= previous00 ;
end if;
when "01" =>
if branch_prediction = '1' then
previous00 <= previous00 + "01";
miss_pred <=miss_pred + 1;
else

```

```

previous00 <= previous00 - "01" ;
end if;

when "10" =>

if branch_prediction = '1' then
previous00 <= previous00 + "01";
else
previous00 <= previous00 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>

if branch_prediction = '1' then
previous00 <=previous00;
else
previous00 <=previous00 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when others => previous00 <= previous00;
end case;

when "0001" =>

case previous11 is

```

```

when "00" =>
  if branch_prediction = '1' then
    previous01 <= previous01 + "01" ;
    miss_pred <= miss_pred + 1 ;
  else
    previous01 <= previous01 ;
  end if;
when "01" =>
  if branch_prediction = '1' then
    previous01 <= previous01 + "01";
    miss_pred <=miss_pred + 1;
  else
    previous01 <= previous01 - "01" ;
  end if;
when "10" =>
  if branch_prediction = '1' then
    previous01 <= previous01 + "01";
  else
    previous01 <= previous01 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;

```

```
when "11" =>
  if branch_prediction = '1' then
    previous01 <=previous01;
  else
    previous01 <=previous01 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;
when others => previous01 <= previous01;
end case;
```

```
when "0010" =>
  case previous02 is
```

```
when "00" =>
  if branch_prediction = '1' then
    previous02 <= previous02 + "01" ;
    miss_pred <= miss_pred + 1 ;
  else
    previous02 <= previous02 ;
  end if;
when "01" =>
```

```

if branch_prediction = '1' then
previous02 <= previous02 + "01";
miss_pred <=miss_pred + 1;
else
previous02 <= previous02 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous02 <= previous02+ "01";
else
previous02 <= previous02 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous02 <=previous02;
else
previous02 <=previous02 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous02 <= previous02;

```

end case;

when "0011" =>

case previous03 is

when "00" =>

if branch_prediction = '1' then

previous03 <= previous03 + "01" ;

miss_pred <= miss_pred + 1 ;

else

previous03 <= previous03 ;

end if;

when "01" =>

if branch_prediction = '1' then

previous03 <= previous03 + "01";

miss_pred <=miss_pred + 1;

else

previous03 <= previous03 - "01" ;

end if;

when "10" =>

if branch_prediction = '1' then

previous03 <= previous03+ "01";

```

else
previous03 <= previous03 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous03 <=previous03;
else
previous03<=previous03 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous03 <= previous03;
end case;

when "0100" =>
case previous04 is

when "00" =>
if branch_prediction = '1' then
previous04 <= previous04 + "01" ;
miss_pred <= miss_pred + 1 ;

```

```

else
previous04 <= previous04 ;
end if;
when "01" =>
if branch_prediction = '1' then
previous04 <= previous04 + "01";
miss_pred <=miss_pred + 1;
else
previous04 <= previous04 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous04 <= previous04+ "01";
else
previous04<= previous04 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous04 <=previous04;
else

```



```
previous04<=previous04 - "01" ;  
miss_pred <= miss_pred + 1;  
end if;  
when others => previous04 <= previous04;  
end case;
```

```
when "0101" =>  
case previous05 is
```

```
when "00" =>  
if branch_prediction = '1' then  
previous05 <= previous05 + "01" ;  
miss_pred <= miss_pred + 1 ;  
else  
previous05 <= previous05 ;  
end if;
```

```
when "01" =>  
if branch_prediction = '1' then  
previous05 <= previous05 + "01";  
miss_pred <=miss_pred + 1;  
else  
previous05 <= previous05 - "01" ;
```

```

end if;
when "10" =>
if branch_prediction = '1' then
previous05 <= previous05+ "01";
else
previous05 <= previous05 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous05<=previous05;
else
previous05<=previous05 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous05 <= previous05;
end case;

when "0110" =>
case previous06 is

```

```

when "00" =>
  if branch_prediction = '1' then
    previous06 <= previous06 + "01" ;
    miss_pred <= miss_pred + 1 ;
  else
    previous06 <= previous06 ;
  end if;
when "01" =>
  if branch_prediction = '1' then
    previous06 <= previous06 + "01";
    miss_pred <=miss_pred + 1;
  else
    previous06 <= previous06 - "01" ;
  end if;
when "10" =>
  if branch_prediction = '1' then
    previous06<= previous06+ "01";
  else
    previous06 <= previous06- "01" ;
    miss_pred <= miss_pred + 1;
  end if;

```

```
when "11" =>
  if branch_prediction = '1' then
    previous06<=previous06;
  else
    previous06<=previous06 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;
when others => previous06 <= previous06;
end case;
```

```
when "0111" =>
  case previous07 is
```

```
    when "00" =>
      if branch_prediction = '1' then
        previous07 <= previous07 + "01" ;
        miss_pred <= miss_pred + 1 ;
      else
        previous07 <= previous07 ;
      end if;
    when "01" =>
      if branch_prediction = '1' then
```

```

previous07 <= previous07 + "01";
miss_pred <=miss_pred + 1;
else
previous07 <= previous07 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous07 <= previous07+ "01";
else
previous07 <= previous07 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous07<=previous07;
else
previous07<=previous07 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous07 <= previous07;
end case;

```

```

when "1000" =>
case previous08 is

when "00" =>
if branch_prediction = '1' then
previous08 <= previous08 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous08 <= previous08 ;
end if;
when "01" =>
if branch_prediction = '1' then
previous08 <= previous08 + "01";
miss_pred <=miss_pred + 1;
else
previous08 <= previous08 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous08 <= previous08+ "01";
else

```

```
previous08 <= previous08- "01" ;  
miss_pred <= miss_pred + 1;  
end if;
```

```
when "11" =>  
  if branch_prediction = '1' then  
    previous08<=previous08;  
  else  
    previous08<=previous08 - "01" ;  
    miss_pred <= miss_pred + 1;  
  end if;  
when others => previous08 <= previous08;  
end case;
```

```
when "1001" =>  
  case previous09 is  
  
    when "00" =>  
      if branch_prediction = '1' then  
        previous09 <= previous09 + "01" ;  
        miss_pred <= miss_pred + 1 ;  
      else
```

```

previous09 <= previous09 ;
end if;
when "01" =>
  if branch_prediction = '1' then
    previous09 <= previous09 + "01";
    miss_pred <= miss_pred + 1;
  else
    previous09 <= previous09 - "01" ;
  end if;
when "10" =>
  if branch_prediction = '1' then
    previous09 <= previous09 + "01";
  else
    previous09 <= previous09 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;

when "11" =>
  if branch_prediction = '1' then
    previous09 <= previous09;
  else
    previous09 <= previous09 - "01" ;

```



```
miss_pred <= miss_pred + 1;
end if;
when others => previous09 <= previous09;
end case;
```

```
when "1010" =>
case previous010 is
```

```
when "00" =>
if branch_prediction = '1' then
previous010 <= previous010 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous010 <= previous010 ;
end if;
```

```
when "01" =>
if branch_prediction = '1' then
previous010 <= previous010 + "01";
miss_pred <=miss_pred + 1;
else
previous010 <= previous010 - "01" ;
end if;
```

```
when "10" =>
  if branch_prediction = '1' then
    previous010<= previous010+ "01";
  else
    previous010 <= previous010- "01" ;
    miss_pred <= miss_pred + 1;
  end if;
```

```
when "11" =>
  if branch_prediction = '1' then
    previous010<=previous010;
  else
    previous010<=previous010 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;
when others => previous010 <= previous010;
end case;
```

```
when "1011" =>
  case previous011 is
```

```
when "00" =>
```

```

if branch_prediction = '1' then
previous011 <= previous011 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous011 <= previous011 ;
end if;
when "01" =>
if branch_prediction = '1' then
previous011 <= previous011 + "01";
miss_pred <=miss_pred + 1;
else
previous011 <= previous011 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous011<= previous011+ "01";
else
previous011 <= previous011- "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>

```

```
if branch_prediction = '1' then
previous011<=previous011;
else
previous011<=previous011 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous011 <= previous011;
end case;
```

```
when "1100" =>
case previous012 is
```

```
when "00" =>
if branch_prediction = '1' then
previous012 <= previous012 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous012 <= previous012 ;
end if;
```

```
when "01" =>
if branch_prediction = '1' then
previous012 <= previous012 + "01";
```

```

miss_pred <= miss_pred + 1;
else
previous012 <= previous012 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous012<= previous012+ "01";
else
previous012 <= previous012- "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous012<=previous012;
else
previous012<=previous012 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous012 <= previous012;
end case;

```

when "1101" =>

case previous013 is

when "00" =>

if branch_prediction = '1' then

previous013 <= previous013 + "01" ;

miss_pred <= miss_pred + 1 ;

else

previous013 <= previous013 ;

end if;

when "01" =>

if branch_prediction = '1' then

previous013 <= previous013 + "01";

miss_pred <=miss_pred + 1;

else

previous013 <= previous013 - "01" ;

end if;

when "10" =>

if branch_prediction = '1' then

previous013<= previous013+ "01";

else

previous013 <= previous013- "01" ;

```
miss_pred <= miss_pred + 1;  
end if;
```

```
when "11" =>  
  if branch_prediction = '1' then  
    previous013<=previous013;  
  else  
    previous013<=previous013 - "01" ;  
    miss_pred <= miss_pred + 1;  
  end if;  
when others => previous013 <= previous013;  
end case;
```

```
when "1110" =>  
  case previous014 is
```

```
    when "00" =>  
      if branch_prediction = '1' then  
        previous014 <= previous014 + "01" ;  
        miss_pred <= miss_pred + 1 ;  
      else  
        previous014<= previous014 ;
```

```

end if;
when "01" =>
  if branch_prediction = '1' then
    previous014<= previous014 + "01";
    miss_pred <=miss_pred + 1;
  else
    previous014 <= previous014 - "01" ;
  end if;
when "10" =>
  if branch_prediction = '1' then
    previous014<= previous014+ "01";
  else
    previous014 <= previous014- "01" ;
    miss_pred <= miss_pred + 1;
  end if;

when "11" =>
  if branch_prediction = '1' then
    previous014<=previous014;
  else
    previous014<=previous014 - "01" ;
    miss_pred <= miss_pred + 1;

```



```
end if;
when others => previous014 <= previous014;
end case;
```

```
when "1111" =>
case previous015 is
```

```
when "00" =>
if branch_prediction = '1' then
previous015 <= previous015 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous115<= previous015 ;
end if;
```

```
when "01" =>
if branch_prediction = '1' then
previous015<= previous015 + "01";
miss_pred <=miss_pred + 1;
else
previous015 <= previous015 - "01" ;
end if;
```

```
when "10" =>
```

```

if branch_prediction = '1' then
previous015<= previous015+ "01";
else
previous015 <= previous015- "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous015<=previous015;
else
previous015<=previous015 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous015 <= previous015;
end case;

temp_global<=global_history;
global_history(3)<=branch_prediction;
when others => global_history <= global_history;
end case;

```

```

elsif(branch_address='1') then
case global_history is
when "0000" =>
case previous10 is
when "00" =>
if branch_prediction = '1' then
previous10 <= previous10 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous10 <= previous10 ;
end if;
when "01" =>
if branch_prediction = '1' then
previous10 <= previous10 + "01";
miss_pred <=miss_pred + 1;
else
previous10 <= previous10 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous10 <= previous10 + "01";

```

```

else
previous10 <= previous10 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous10 <=previous10;
else
previous10 <=previous10 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous10 <= previous10;
end case;

when "0001" =>
case previous11 is

when "00" =>
if branch_prediction = '1' then
previous11 <= previous11 + "01" ;
miss_pred <= miss_pred + 1 ;

```

```

else
previous11 <= previous11 ;
end if;
when "01" =>
    if branch_prediction = '1' then
previous11 <= previous11 + "01";
miss_pred <=miss_pred + 1;
    else
previous11 <= previous11 - "01" ;
end if;
when "10" =>
    if branch_prediction = '1' then
previous11 <= previous11 + "01";
    else
previous11 <= previous11 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
    if branch_prediction = '1' then
previous11 <=previous11;
    else

```

```
previous11 <=previous11 - "01" ;  
miss_pred <= miss_pred + 1;  
end if;  
when others => previous11 <= previous11;  
end case;
```

```
when "0010" =>  
case previous12 is
```

```
when "00" =>  
if branch_prediction = '1' then  
previous12 <= previous12 + "01" ;  
miss_pred <= miss_pred + 1 ;  
else  
previous12 <= previous12 ;  
end if;
```

```
when "01" =>  
if branch_prediction = '1' then  
previous12 <= previous12 + "01";  
miss_pred <=miss_pred + 1;  
else  
previous12 <= previous12 - "01" ;
```

```

end if;

when "10" =>

if branch_prediction = '1' then
previous12 <= previous12+ "01";
else
previous12 <= previous12 - "01" ;
miss_pred <= miss_pred + 1;
end if;


when "11" =>

if branch_prediction = '1' then
previous12 <=previous12;
else
previous12 <=previous12 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when others => previous12 <= previous12;

end case;


when "0011" =>

case previous13 is

```

```

when "00" =>
  if branch_prediction = '1' then
    previous13 <= previous13 + "01" ;
    miss_pred <= miss_pred + 1 ;
  else
    previous13 <= previous13 ;
  end if;
when "01" =>
  if branch_prediction = '1' then
    previous13 <= previous13 + "01";
    miss_pred <=miss_pred + 1;
  else
    previous13 <= previous13 - "01" ;
  end if;
when "10" =>
  if branch_prediction = '1' then
    previous13 <= previous13+ "01";
  else
    previous13 <= previous13 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;

```



```
when "11" =>
  if branch_prediction = '1' then
    previous13 <=previous13;
  else
    previous13<=previous13 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;
when others => previous13 <= previous13;
end case;
```

```
when "0100" =>
  case previous14 is
```

```
when "00" =>
  if branch_prediction = '1' then
    previous14 <= previous14 + "01" ;
    miss_pred <= miss_pred + 1 ;
  else
    previous14 <= previous14 ;
  end if;
when "01" =>
  if branch_prediction = '1' then
```

```

previous14 <= previous14 + "01";
miss_pred <=miss_pred + 1;
else
previous14 <= previous14 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous14 <= previous14+ "01";
else
previous14<= previous14 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous14 <=previous14;
else
previous14<=previous14 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous14 <= previous14;
end case;

```

when "0101" =>

case previous15 is

when "00" =>

if branch_prediction = '1' then

previous15 <= previous15 + "01" ;

miss_pred <= miss_pred + 1 ;

else

previous15 <= previous15 ;

end if;

when "01" =>

if branch_prediction = '1' then

previous15 <= previous15 + "01";

miss_pred <=miss_pred + 1;

else

previous15 <= previous15 - "01" ;

end if;

when "10" =>

if branch_prediction = '1' then

previous15 <= previous15+ "01";

else

```
previous15 <= previous15 - "01" ;  
miss_pred <= miss_pred + 1;  
end if;
```

```
when "11" =>  
  if branch_prediction = '1' then  
    previous15<=previous15;  
  else  
    previous15<=previous15 - "01" ;  
    miss_pred <= miss_pred + 1;  
  end if;  
when others => previous15 <= previous15;  
end case;
```

```
when "0110" =>  
  case previous16 is
```

```
    when "00" =>  
      if branch_prediction = '1' then  
        previous16 <= previous16 + "01" ;  
        miss_pred <= miss_pred + 1 ;  
      else
```

```

previous16 <= previous16 ;
end if;
when "01" =>
  if branch_prediction = '1' then
    previous16 <= previous16 + "01";
    miss_pred <= miss_pred + 1;
  else
    previous16 <= previous16 - "01" ;
  end if;
when "10" =>
  if branch_prediction = '1' then
    previous16 <= previous16 + "01";
  else
    previous16 <= previous16 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;

when "11" =>
  if branch_prediction = '1' then
    previous16 <= previous16;
  else
    previous16 <= previous16 - "01" ;

```

```
miss_pred <= miss_pred + 1;
end if;
when others => previous16 <= previous16;
end case;
```

```
when "0111" =>
case previous17 is
```

```
when "00" =>
if branch_prediction = '1' then
previous17 <= previous17 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous17 <= previous17 ;
end if;
```

```
when "01" =>
if branch_prediction = '1' then
previous17 <= previous17 + "01";
miss_pred <=miss_pred + 1;
else
previous17 <= previous17 - "01" ;
end if;
```

```
when "10" =>
  if branch_prediction = '1' then
    previous17 <= previous17 + "01";
  else
    previous17 <= previous17 - "01";
    miss_pred <= miss_pred + 1;
  end if;
```

```
when "11" =>
  if branch_prediction = '1' then
    previous17 <= previous17;
  else
    previous17 <= previous17 - "01";
    miss_pred <= miss_pred + 1;
  end if;
when others => previous17 <= previous17;
end case;
```

```
when "1000" =>
  case previous18 is
```

```
when "00" =>
```

```

if branch_prediction = '1' then
previous18 <= previous18 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous18 <= previous18 ;
end if;
when "01" =>
if branch_prediction = '1' then
previous18 <= previous18 + "01";
miss_pred <=miss_pred + 1;
else
previous18 <= previous18 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous18 <= previous18+ "01";
else
previous18 <= previous18- "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>

```



```

if branch_prediction = '1' then
previous18<=previous18;
else
previous18<=previous18 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous18 <= previous18;
end case;

```

```

when "1001" =>
case previous19 is

```

```

when "00" =>
if branch_prediction = '1' then
previous19 <= previous19 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous19 <= previous19 ;
end if;

```

```

when "01" =>
if branch_prediction = '1' then
previous19 <= previous19 + "01";

```

```

miss_pred <= miss_pred + 1;
else
previous19 <= previous19 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous19 <= previous19 + "01";
else
previous19 <= previous19 - "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous19 <= previous19;
else
previous19 <= previous19 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous19 <= previous19;
end case;

```

when "1010" =>

case previous110 is

when "00" =>

if branch_prediction = '1' then

previous110 <= previous110 + "01" ;

miss_pred <= miss_pred + 1 ;

else

previous110 <= previous110 ;

end if;

when "01" =>

if branch_prediction = '1' then

previous110 <= previous110 + "01";

miss_pred <=miss_pred + 1;

else

previous110 <= previous110 - "01" ;

end if;

when "10" =>

if branch_prediction = '1' then

previous110<= previous110+ "01";

else

previous110 <= previous110- "01" ;

```
miss_pred <= miss_pred + 1;  
end if;
```

```
when "11" =>  
  if branch_prediction = '1' then  
    previous110<=previous110;  
  else  
    previous110<=previous110 - "01" ;  
    miss_pred <= miss_pred + 1;  
  end if;  
when others => previous110 <= previous110;  
end case;
```

```
when "1011" =>  
  case previous111 is
```

```
    when "00" =>  
      if branch_prediction = '1' then  
        previous111 <= previous111 + "01" ;  
        miss_pred <= miss_pred + 1 ;  
      else  
        previous111 <= previous111 ;
```

```

end if;
when "01" =>
  if branch_prediction = '1' then
    previous111 <= previous111 + "01";
    miss_pred <= miss_pred + 1;
  else
    previous111 <= previous111 - "01" ;
  end if;
when "10" =>
  if branch_prediction = '1' then
    previous111 <= previous111 + "01";
  else
    previous111 <= previous111 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;

when "11" =>
  if branch_prediction = '1' then
    previous111 <= previous111;
  else
    previous111 <= previous111 - "01" ;
    miss_pred <= miss_pred + 1;

```

```
end if;
when others => previous111 <= previous111;
end case;
```

```
when "1100" =>
case previous112 is
```

```
when "00" =>
if branch_prediction = '1' then
previous112 <= previous112 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous112 <= previous112 ;
end if;
```

```
when "01" =>
if branch_prediction = '1' then
previous112 <= previous112 + "01";
miss_pred <=miss_pred + 1;
else
previous112 <= previous112 - "01" ;
end if;
```

```
when "10" =>
```

```
if branch_prediction = '1' then
previous112<= previous112+ "01";
else
previous112 <= previous112- "01" ;
miss_pred <= miss_pred + 1;
end if;
```

```
when "11" =>
if branch_prediction = '1' then
previous112<=previous112;
else
previous112<=previous112 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous112 <= previous112;
end case;
```

```
when "1101" =>
case previous113 is
```

```
when "00" =>
if branch_prediction = '1' then
```

```

previous113 <= previous113 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous113 <= previous113 ;
end if;
when "01" =>
if branch_prediction = '1' then
previous113 <= previous113 + "01";
miss_pred <=miss_pred + 1;
else
previous113 <= previous113 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous113<= previous113+ "01";
else
previous113 <= previous113- "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then

```



```
previous113<=previous113;
else
previous113<=previous113 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous113 <= previous113;
end case;
```

```
when "1110" =>
case previous114 is
```

```
when "00" =>
if branch_prediction = '1' then
previous114 <= previous114 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous114<= previous114 ;
end if;
```

```
when "01" =>
if branch_prediction = '1' then
previous114<= previous114 + "01";
miss_pred <=miss_pred + 1;
```

```

else
previous114 <= previous114 - "01" ;
end if;
when "10" =>
if branch_prediction = '1' then
previous114<= previous114+ "01";
else
previous114 <= previous114- "01" ;
miss_pred <= miss_pred + 1;
end if;

when "11" =>
if branch_prediction = '1' then
previous114<=previous114;
else
previous114<=previous114 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous114 <= previous114;
end case;

when "1111" =>

```

case previous115 is

when "00" =>

if branch_prediction = '1' then

previous115 <= previous115 + "01" ;

miss_pred <= miss_pred + 1 ;

else

previous115 <= previous115 ;

end if;

when "01" =>

if branch_prediction = '1' then

previous115 <= previous115 + "01";

miss_pred <= miss_pred + 1;

else

previous115 <= previous115 - "01" ;

end if;

when "10" =>

if branch_prediction = '1' then

previous115 <= previous115 + "01";

else

previous115 <= previous115 - "01" ;

miss_pred <= miss_pred + 1;

```

end if;

when "11" =>
if branch_prediction = '1' then
previous115<=previous115;
else
previous115<=previous115 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous115 <= previous115;
end case;
when others => global_history <= global_history;
end case;
end if;
temp_global<=global_history;
global_history(2 downto 0)<=temp_global(3 downto 1);
global_history(3)<=branch_prediction;
end process;
miss_prediction<=miss_pred;
end;

```

2-bit predictor

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.numeric_bit.all;
```

```
use IEEE.STD_LOGIC_ARITH.ALL;
```

```
use ieee.std_logic_unsigned.all;
```

```
entity twobit_predictor is
```

```
port( b_address:in bit;
```

```
      branch_prediction:in std_logic;
```

```
      miss_prediction:out integer);
```

```
end twobit_predictor;
```

```
architecture behavioral of twobit_predictor is
```

```
signal previous0:std_logic_vector(1 downto 0):="00";
```

```
signal previous1:std_logic_vector(1 downto 0):="00";
```

```
signal miss_pred:integer:=0;
```

```
begin
```

```
process (b_address,branch_prediction)
```

```
begin
```

```
if (b_address='0') then
```

```
case previous0 is
when "00" =>
if branch_prediction = '1' then
previous0 <= previous0 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous0 <= previous0 ;
end if;
```

```
when "01" =>
if branch_prediction = '1' then
previous0 <= previous0 + "01";
miss_pred <=miss_pred + 1;
else
previous0 <= previous0 - "01" ;
end if;
```

```
when "10" =>
if branch_prediction = '1' then
previous0 <= previous0 + "01";
else
previous0 <= previous0 - "01" ;
miss_pred <= miss_pred + 1;
```

```
end if;

when "11" =>
if branch_prediction = '1' then
previous0 <=previous0;
else
previous0 <=previous0 - "01" ;
miss_pred <= miss_pred + 1;
end if;
when others => previous0 <= previous0;
end case;
```

```
elsif b_address = '1' then
case previous1 is
when "00" =>
if branch_prediction= '1' then
previous1 <= previous1 + "01" ;
miss_pred <= miss_pred + 1 ;
else
previous1 <= previous1 ;
end if;
```

```
when "01" =>
  if branch_prediction = '1' then
    previous1 <= previous1 + "01";
    miss_pred <= miss_pred + 1;
  else
    previous1 <= previous1 - "01" ;
  end if;
```

```
when "10" => if branch_prediction = '1' then
  previous1 <= previous1 + "01";
  else
    previous1 <= previous1 - "01";
    miss_pred <= miss_pred + 1;
  end if;
```

```
when "11" =>
  if branch_prediction = '1' then
    previous1 <= previous1;
  else
    previous1 <= previous1 - "01" ;
    miss_pred <= miss_pred + 1;
  end if;
```



```
when others => previous1 <= previous1;
end case;
end if;
```

```
end process;
miss_prediction<= miss_pred;
end ;
```

1-bit pred

```
library IEEE ;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;
use IEEE.NUMERIC_STD.ALL;
```

```
entity one_bitpredictor is
port(branch_address:in bit;
      current_prediction:in std_logic;
      miss_prediction:out integer);
end one_bitpredictor;
```

```
architecture behavioral of one_bitpredictor is
signal miss_pred:integer:=0;
signal counter_out:integer;
```

```

signal previous_0:std_logic:='1';
signal previous_1:std_logic:='1';
begin
process(branch_address,current_prediction)is
begin

case branch_address is
when '0' =>
--if(branch_address='0') then
if(current_prediction /= previous_0) then
miss_pred<=miss_pred+1;
previous_0<=current_prediction;
end if;

when '1' =>
--elsif(branch_address='1') then
if(current_prediction /= previous_1) then
miss_pred<=miss_pred+1;
previous_1<=current_prediction;
end if;

```

```
--end if;  
end case;  
  
end process;  
miss_prediction<=miss_pred;  
end;
```