lab06_ip 核使用

实验目的：

掌握 IP 核的使用方法

熟悉 RAM 的接口及时序

能熟练设计时序逻辑电路

实验内容：

例化一个数据位宽为 32bit，深度为 32 的简单双端口 ram

- 读写端口采用同一时钟
- 将 ram 初始化为 1,1,0,0,0,0.....
- 利用有限状态机实现斐波那契数列的计算
  - 读出 0 号地址、读出 1 号地址，将两个数据相加，写入 2 号地址
  - 读出 1 号地址、 读出 2 号地址，将两个数据相加， 写入 3 号地址
  - 读出 2 号地址、 读出 3 号地址，将两个数据相加， 写入 4 号地址
  - ....
  - 读出 29 号地址、 读出 30 号地址，将两个数据相加， 写入 31 号地址
  - 停止

实验结果：

成功仿真。

实验分析：

根据题目要求，学习 IP 核的使用，理解 RAM 的接口及时序。

对于输出斐波那契数列的要求：

分为三个状态：

0： 更新 dina<=tmp+doutb //tmp 存放的是上一个循环的 doutb；
1： 更新 tmp <=doutb; //上文提到的一步
置写使能 wea<=1
置写地址 addra++
2： 置读地址 addrb++；
置写使能 wea<=0
state<=0
另外，为了在 addrb==31 时置 state 为 0

关于实验的过程：

这次首先要学习一样新东西 IP 核的使用，花了点时间。

在写算法的时候没有听老师的意见一开始先画状态图。一开始也没有完全理解 ram 的使用，以及整个算法。

到了六点多的时候理清了思路，重新开始画状态图，整体留长了很多，几分钟就搞定了。

所以说：

要听老师的话。

思路一定要清晰。

意见建议：

无

附录：

由于只需要仿真，故只用了一个文件。

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
//////////
// Company:
// Engineer:
//
// Create Date:     15:37:17 11/18/2016
```

```
// Design Name:     top
// Module Name:
C:/Users/Zevin/Documents/verilog/lab6/TF.v
// Project Name:    lab6
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: top
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////
//////////


module TF;


reg clk;
```

```
reg     [2:0]   state;
reg         [31 : 0] tmp;
reg     [0 : 0]      wea;
reg     [4 : 0]      addra;
wire    [31 : 0]    doutb;
reg     [31 : 0]    dina;
reg     [4 : 0]      addrb;


always@(posedge clk)begin
    if(addrb == 31)
        state <= 0;
    else if(state < 3)
        state <= state+1;
    else
        state <= 0;
end

always@(posedge clk)begin
    if(state == 0)
        dina <= tmp + doutb;
end
```

```
always@(posedge clk)begin
    if(state == 1)
        tmp <= doutb;
end
always@(posedge clk)begin
    if(state == 1)
        addra <= addra + 1;
end
always@(posedge clk)begin
    if(state == 1) begin

            wea <= 1;
    end
    else wea <= 0;
end
always@(posedge clk)begin
    if(state == 2)
        addrb <= addrb +1;
end
myram mr(clk, wea, addra, dina, clk, addrb, doutb);
```

```
    // Instantiate the Unit Under Test (UUT)   initial begin
        // Initialize Inputs
        initial begin
        tmp = 0;
        wea = 0;
        clk = 0;
        addra = 0;
        addrb = 0;
        dina = 0;
        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
            forever #5 clk = ~clk;
        end

endmodule
```