**实验报告**

2019 年 4 月 22 日 成绩：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 於文卓 | 学号 | 17061833 | 班级 | 17052317 |
| 专业 | 计算机科学与技术 | | 课程名称 | 计算机组成原理课程设计 | |
| 任课老师 | 冯建文 | 指导老师 | 冯建文 | 机位号 |  |
| 实验序号 | 4 | 实验名称 | 存储器设计实验 | | |
| 实验时间 | 4月22日 | 实验地点 | 1-225 | 实验设备号 | #9 |
| **一、实验目的与要求** | | | | | |
| 1. 实验目的： 2. 掌握运用Verilog HDL进行行为描述与建模的技巧和方法 3. 学习使用Memory IP核生成存储器的方法 4. 学习存储器的结构及读写原理，掌握存储器的设计方法 5. 实验要求： 6. 实现一个MIPS32的容量为256字节的、按字节编制、按字访问的、单端口RAM存储器模块，完成仿真验证和板级验证 7. 使用IP核实现该存储器模块 8. 编写顶层验证模块，调用该存储器模块，以实现板级验证 | | | | | |
| **二、实验设计与程序代码** | | | | | |
| 1. 模块设计说明 2. 设置IP核，具体操作如下          1. 实验程序源代码及注释等 2. 顶层模块代码  |  | | --- | | module top  (  input Clk,  input scan\_clk,  input Mem\_Write,  input rst\_,  input [7:0]Mem\_Addr,  input [2:0]choiceWriteData,  output [7:0]wei,  output [7:0]duan    );  reg [31:0]M\_W\_Data;  wire [31:0]M\_R\_Data;  always@(\*)  begin  case(choiceWriteData)  3'b000:M\_W\_Data = 32'hABCD\_EFFF;  3'b001:M\_W\_Data = 32'h1111\_1111;  3'b010:M\_W\_Data = 32'h1234\_5678;  3'b011:M\_W\_Data = 32'h0000\_0001;  3'b100:M\_W\_Data = 32'h0000\_1234;  3'b101:M\_W\_Data = 32'h0000\_FFFF;  3'b110:M\_W\_Data = 32'hFFFF\_1234;  3'b111:M\_W\_Data = 32'hFFFF\_FFFF;  endcase  end  RAM\_B DATA\_RAM  (  .clka(Clk),  .wea(Mem\_Write),  .addra(Mem\_Addr[7:2]),  .dina(M\_W\_Data),  .douta(M\_R\_Data)  );  scan m1(scan\_clk,rst\_,M\_R\_Data,wei,duan);  endmodule |  1. 数码管模块代码  |  | | --- | | module scan(    input clk,//100MHZ  input rst\_,//reset  input [31:0]key,//通过拨动按钮控制段选  output reg [7:0]wei,//在always里赋值 需要用到reg  output reg [7:0]duan  );  wire clk\_4ms;//分频过后2ms一个方波(改过了 是2ms了)  reg [2:0]wei\_count;//位码计数器，控制4个数码管  reg [7:0]segcode[0:15] = {//用数组的方式定义 左表示位数 右表示个数  8'b0000\_0011,//0  8'b1001\_1111,//1  8'b0010\_0101,//2  8'b0000\_1101,//3  8'b1001\_1001,//4  8'b0100\_1001,//5  8'b0100\_0001,//6  8'b0001\_1111,//7  8'b0000\_0001,//8  8'b0000\_1001,//9  ~8'b11101110,//A  ~8'b00111110,//b  ~8'b10011100,//C  ~8'b01111010,//d  ~8'b10011110,//E  8'b01110001//F  };  div f1(rst\_,clk,clk\_4ms);//分频器模块  wire clk\_slow;//一秒钟两次  div\_slow f2(rst\_,clk,clk\_slow);  always @(posedge clk\_4ms or negedge rst\_)//每一个上升沿都使得计数器+1 做到连续扫描数码管  begin  if(!rst\_)  begin  wei\_count<=3'b00;  end  else  begin  wei\_count<=wei\_count+1'b1;  if(wei\_count ==3'b111)  begin  wei\_count<=3'b000;  end  end    end  reg [7:0] pos\_1,pos\_2,pos\_3,pos\_4,pos\_5,pos\_6,pos\_7,pos\_8;  //用开关控制数码管显示  always @(posedge clk\_4ms or negedge rst\_)  begin  if(!rst\_)  begin  pos\_1 <= 4'b0000;  pos\_2 <= 4'b0000;  pos\_3 <= 4'b0000;  pos\_4 <= 4'b0000;  pos\_5 <= 4'b0000;  pos\_6 <= 4'b0000;  pos\_7 <= 4'b0000;  pos\_8 <= 4'b0000;  end  else  begin  pos\_1 <= key[3:0];//if(pos\_1 > 4'd9) pos\_1<=4'b0000;  pos\_2 <= key[7:4];//if(pos\_2 > 4'd9) pos\_2<=4'b0000;  pos\_3 <= key[11:8];//if(pos\_3 > 4'd9) pos\_3<=4'b0000;  pos\_4 <= key[15:12];//if(pos\_4 > 4'd9) pos\_4<=4'b0000;  pos\_5 <= key[19:16];  pos\_6 <= key[23:20];  pos\_7 <= key[27:24];  pos\_8 <= key[31:28];  end  end  always @(\*)//数码管扫描进程 想想为什么是\* ？？？  begin  case(wei\_count)  3'b000: begin wei<=8'b11111110; duan<=segcode[pos\_1];end  3'b001: begin wei<=8'b11111101; duan<=segcode[pos\_2];end  3'b010: begin wei<=8'b11111011; duan<=segcode[pos\_3];end  3'b011: begin wei<=8'b11110111; duan<=segcode[pos\_4];end  3'b100: begin wei<=8'b11101111; duan<=segcode[pos\_5];end  3'b101: begin wei<=8'b11011111; duan<=segcode[pos\_6];end  3'b110: begin wei<=8'b10111111; duan<=segcode[pos\_7];end  3'b111: begin wei<=8'b01111111; duan<=segcode[pos\_8];end  default:begin wei<=8'b11111111;end  endcase  end    endmodule | | | | | | |
| **三、实验仿真** | | | | | |
| 1. 仿真代码  |  | | --- | | `timescale 1ns / 1ps  ////////////////////////////////////////////////////////////////////////////////  // Company:  // Engineer:  //  // Create Date: 00:27:25 04/27/2019  // Design Name: top\_fortest  // Module Name: D:/fpga/shiyan5/test.v  // Project Name: shiyan5  // Target Device:  // Tool versions:  // Description:  //  // Verilog Test Fixture created by ISE for module: top\_fortest  //  // Dependencies:  //  // Revision:  // Revision 0.01 - File Created  // Additional Comments:  //  ////////////////////////////////////////////////////////////////////////////////  module test;  // Inputs  reg Clk;  reg Mem\_Write;  reg [7:0] Mem\_Addr;  reg [31:0] M\_W\_Data;  // Outputs  wire [31:0] M\_R\_Data;  // Instantiate the Unit Under Test (UUT)  top\_fortest uut (  .Clk(Clk),  .Mem\_Write(Mem\_Write),  .Mem\_Addr(Mem\_Addr),  .M\_W\_Data(M\_W\_Data),  .M\_R\_Data(M\_R\_Data)  );  initial begin  // Initialize Inputs  Clk = 0;  Mem\_Write = 0;  Mem\_Addr = 0;  M\_W\_Data = 0;  // Wait 100 ns for global reset to finish  #100;    // Add stimulus here  end  always #10 Clk=~Clk;  initial  begin  #20;  Mem\_Write = 0;  Mem\_Addr = 8'b0000\_0000;  M\_W\_Data = 32'h1111\_1111;  #20;  Mem\_Write = 1;  Mem\_Addr = 8'b0000\_0000;  M\_W\_Data = 32'h1111\_1111;  #20;  Mem\_Write = 0;  Mem\_Addr = 8'b0000\_0000;  M\_W\_Data = 32'h2222\_2222;    #20;  Mem\_Write = 0;  Mem\_Addr = 8'b0011\_1111;  M\_W\_Data = 32'h2222\_2222;  #20;  Mem\_Write = 1;  Mem\_Addr = 8'b0011\_1111;  M\_W\_Data = 32'h2222\_2222;  #20;  Mem\_Write = 0;  Mem\_Addr = 8'b0011\_1111;  M\_W\_Data = 32'h2222\_2222;      end    endmodule |  1. 仿真波形      1. 仿真结果分析 2. 读取地址为8’b0000\_0000处的数据，设置Mem\_Write 为 0，此时读到的数据为存储器中的初始化数据，为8fa40000 3. 设置Mem\_Write为1并且对地址为8’b0000\_0000写入数据32’h1111\_1111 4. 重新执行1操作，读到的数据为11111111 5. 读取地址为32’h0000\_0000处的数据，设置Mem\_Write 为 0，此时读到的数据为存储器中的初始化数据，为8fa40000 6. 设置Mem\_Write为1并且对地址为32’h0011\_1111写入数据32’h2222\_2222 7. 重新执行1操作，读到的数据为22222222   仿真结果符合要求，仿真结果正确,说明IP核配置正确 | | | | | |
| **四、电路图** | | | | | |
|  | | | | | |
| **五、引脚配置（约束文件）** | | | | | |
| //如果边沿输入信号，还要在";"前加上：| CLOCK\_DEDICATED\_ROUTE = FALSE  //譬如: NET "clk" LOC = N17 | IOSTANDARD = LVCMOS18 | CLOCK\_DEDICATED\_ROUTE = FALSE; //BT\_S[7]  //HCS-A01板卡上的实际开关  NET "scan\_clk" LOC = E3 |IOSTANDARD = LVCMOS18;  NET "Clk" LOC = P18 |IOSTANDARD = LVCMOS18 |CLOCK\_DEDICATED\_ROUTE = FALSE;  NET "rst\_" LOC = N17 |IOSTANDARD = LVCMOS18 |CLOCK\_DEDICATED\_ROUTE = FALSE;  NET "Mem\_Addr[7]" LOC = V5 | IOSTANDARD = LVCMOS18; //Mem\_Addr[7]  NET "Mem\_Addr[6]" LOC = T4 | IOSTANDARD = LVCMOS18; //Mem\_Addr[6]  NET "Mem\_Addr[5]" LOC = V6 | IOSTANDARD = LVCMOS18; //Mem\_Addr[5]  NET "Mem\_Addr[4]" LOC = T5 | IOSTANDARD = LVCMOS18; //Mem\_Addr[4]  NET "Mem\_Addr[3]" LOC = T6 | IOSTANDARD = LVCMOS18; //Mem\_Addr[3]  NET "Mem\_Addr[2]" LOC = V7 | IOSTANDARD = LVCMOS18; //Mem\_Addr[2]  NET "Mem\_Addr[1]" LOC = R8 | IOSTANDARD = LVCMOS18; //Mem\_Addr[3]  NET "Mem\_Addr[0]" LOC = U9 | IOSTANDARD = LVCMOS18; //Mem\_Addr[2]  NET "choiceWriteData[2]" LOC = V14 | IOSTANDARD = LVCMOS18; //choiceWriteData[2]  NET "choiceWriteData[1]" LOC = T14 | IOSTANDARD = LVCMOS18; //choiceWriteData[1]  NET "choiceWriteData[0]" LOC = V15 | IOSTANDARD = LVCMOS18; //choiceWriteData[0]  NET "MEm\_Write" LOC = R15 | IOSTANDARD = LVCMOS18; //MEm\_Write  NET "wei[7]" LOC = C9 |IOSTANDARD = LVCMOS18;  NET "wei[6]" LOC = C10 |IOSTANDARD = LVCMOS18;  NET "wei[5]" LOC = D10 |IOSTANDARD = LVCMOS18;  NET "wei[4]" LOC = C11 |IOSTANDARD = LVCMOS18;  NET "wei[3]" LOC = M17 |IOSTANDARD = LVCMOS18;  NET "wei[2]" LOC = J14 |IOSTANDARD = LVCMOS18;  NET "wei[1]" LOC = K13 |IOSTANDARD = LVCMOS18;  NET "wei[0]" LOC = P14 |IOSTANDARD = LVCMOS18;  NET "duan[7]" LOC = F14 |IOSTANDARD = LVCMOS18;  NET "duan[6]" LOC = N14 |IOSTANDARD = LVCMOS18;  NET "duan[5]" LOC = J13 |IOSTANDARD = LVCMOS18;  NET "duan[4]" LOC = G13 |IOSTANDARD = LVCMOS18;  NET "duan[3]" LOC = F13 |IOSTANDARD = LVCMOS18;  NET "duan[2]" LOC = G14 |IOSTANDARD = LVCMOS18;  NET "duan[1]" LOC = M13 |IOSTANDARD = LVCMOS18;  NET "duan[0]" LOC = H14 |IOSTANDARD = LVCMOS18; | | | | | |
| **六、思考与探索** | | | | | |
| 1. 实验结果记录：  |  |  |  |  |  | | --- | --- | --- | --- | --- | | 存储器地址 | 初始化数据 | 读出数据 | 写入新数据 | 读出数据 | | Index：0 | 8fa40000 | 8fa40000 | 11112222 | 11112222 | | Index：1 | 27a50004 | 27a50004 | abcdaaaa | abcdaaaa | | Index：4 | 00c23021 | 00c23021 | ccccdddd | ccccdddd | | Index：5 | 0c000000 | 0c000000 | 00001111 | 00001111 | | Index：8 | 0000000c | 0000000c | 12345678 | 12345678 | | Index：11 | 0001102b | 0001102b | 23232323 | 23232323 | | Index：15 | 00a33020 | 00a33020 | 00022222 | 00022222 | | Index：26 | 00c77825 | 00c77825 | ffffffff | ffffffff |  1. 实验结论：   实验结果也符合预期  读出数据和初始化数据相同，写入后读出的数据和写入的数据也相同  说明实验结果正确   1. 问题与解决方案： 2. IP核配置时一开始配置出错，通过看书本和网上资料正确配置了IP核 3. 思考题： 4. 使用IP核设计一个ROM   your\_instance\_name : thinkTest  PORT MAP (  clka => clka,  addra => addra,  douta => douta  );    端口为时钟信号clk，地址addra和输出数据，不存在与写相关的地址。 | | | | | |