实验报告

2021 年 5 月 16 日　　成绩：_____

姓名	刘合	学号		班级	
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		指导老师		机位号	
实验序号	4	实验名称	存储器设计		
实验时间	2021.5.16	实验地点		实验设备号	

一、实验目的与要求

1、 **实验目的：**

1. 学习使用 Vivado 或者 ISE 开发工具的 Memory IP 核，设计生成存储器模块的方法；

2. 学习存储器的结构及读写原理，掌握存储器的设计方法。

2、 实验要求：

1. 使用 Vivado 或者 ISE 工具,按照上述步骤新建一个 64××32 位的 Single Port RAM 并使用 COE 关联文件初始化其内容。

2. 新建 一个 存储器 模块， 端口如图 11.18 所示； 引用 上述的存储器 IP 核，连接实例与存储器模块的端口 。

3. 编写仿真激励代码，进行仿真测试，确保存储器模块读写逻辑功能正确。

4. 针对使用的实验板卡，设计 存储器 模块的板级验证实验方案，编写顶层测试模块。观察图 11.18，输入的电平信号有 32 位写入数据、6 位地址信号和一个写信号，无论哪种实验板卡，输入设备资源都不够。图 11.36 给出一个简单的板级验证方案，由两位开关选择指定的四个常数，作为写入存储器的数据。请参考实验 2 中的板级验证设计方法，自行修改设计。

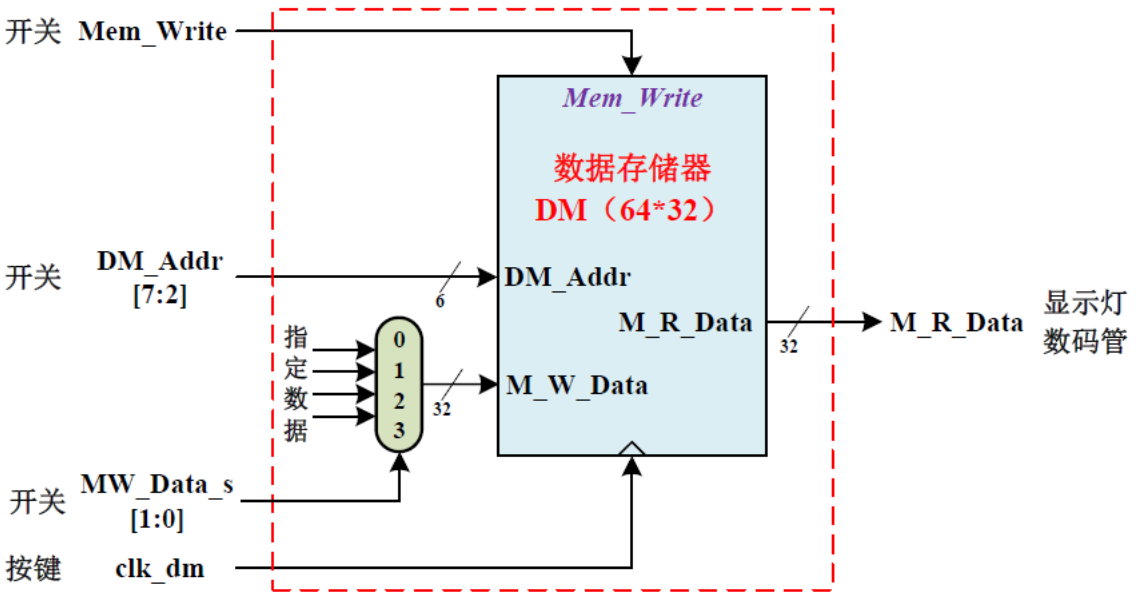


图11.36 实验 4 顶层测试模块范例

5. 选择存储器单元地址，执行读写操作，验证你的存储器是否能正常存取，将实验结果记录到表中，要求存储器读和写功能都被有效测试。

6. 撰写实验报告，格式见附录，重点内容包括：对仿真结果进行分析；描述你设计的板级验证实验方案、模块结构与连接；说明你的板级操作过程；分析记录下来的板级实验结果、得到有效结论。请力所能及回答或实践本实验的“思考与探索”部分。

3. 实验步骤

1. 新建一个工程，通过 IP 核生成向导创建一个 64×32 位的单端口 RAM IP 核，并创建与编辑一个 COE 文件，初始化该 RAM 的内容。

2. 新建一个存储器模块,输入输出端口如图 11.18 所示,其中引用上述生成的 RAM IP 核，并正确连接存储器模块端口与 IP 核实例的端口。

3. 编写激励代码，仿真验证存储器模块功能；分析仿真结果，确保读写操作正确。

4. 设计存储器模块板级验证的实验方案，然后据此编写一个顶层测试模块。

5. 可以依据实际需要，对顶层测试模块进行仿真测试，或者直接进入管脚约束环节。

6. 新建管脚约束文件，依据 图 11.36 的提示 及实际板卡情况， 设计板级验证实验方案，进行相应的引脚配置。

7. 生成 *.bit 文件，下载到实验设备的 FPGA 芯片中。

8. 板级实验：按照你所设计的实验方案，操作输入设备、观察输出设备， 存储器读操作过程为：

（1） 拨动开关输入存储器单元地址

（2） 拨 Mem_Write 开关 =0 按动 时钟键 clk_dm

（3） 观察输出设备上读出的存储器数据是否正确（与初始化数据一致，或者与最后一次写入的数据一致）

存储器写操作的过程为：

（1）拨动开关输入存储器地址；

（2）拨动开关，选择或者输入准备写入存储器的数据

（3）拨 Mem_Write 开关 =1 按动时钟键 clk_dm。

注意：在板级实验中,要保证能观察或者经分析后确认：写入存储器的数据已经更新了指定存储器单元的值。分析实验结果是否正确

二、实验设计与程序代码

1、 模块设计说明

程序包含了 4 个模块一个存储器 IP 核：

1.顶层模块：有序调用其他模块的程序；

2.分频器模块：是为了给数码管扫描显示模块分配时钟频率；

3.数码管扫描显示模块：是为了把 BCD 码数据以十进制数字显示出来

4.数据选择模块：选择不同的值存储到存储器。

5. 存储器 IP 核：存储数据

2.实验程序源代码及注释等

```
module chucun(CLR,clk_1M,clka,wea,addra,choice,AN,seg);//顶层模块
  input CLR;
  input clk_1M;
  input wire clka;
  input wire [0 : 0] wea;
  input wire [7 : 0] addra;
  input [2:0]choice;
  wire [31 : 0] dina;
  output [7:0]AN;
  output [7:0]seg;
```

```
  wire CLK;
  wire [31:0]douta;
  xuanze ss(choice,dina);
    RAM_B aa(
    .clka(clka),        // input wire clka
    .wea(wea),            // input wire [0 : 0] wea
    .addra(addra[7:2]),    // input wire [5 : 0] addra
    .dina(dina),        // input wire [31 : 0] dina
    .douta(douta)    // output wire [31 : 0] douta
);
fpq gg(CLR,clk_1M,CLK);
smgsm ff(CLK,CLR,douta,AN,seg);
endmodule

module xuanze(choice,dina);
      input [2:0]choice;
      output    reg [31:0]dina;
      always@(*)
            begin
                case(choice)
                      3'b000:dina=32'hffff_0f0f;
                      3'b001:dina=32'h1111_1111;
                      3'b010:dina=32'h2222_3333;
                      3'b011:dina=32'habcd_3456;
                      3'b100:dina=32'h2173_9803;
                      3'b101:dina=32'hdadb_caca;
                      3'b110:dina=32'h0000_0000;
                      3'b111:dina=32'haaaa_bbbb;
                endcase
            end
```

```
endmodule
module fpq(CLR,clk_1M,CLK);              //分频器模块
   input CLR;
   input clk_1M;
   output reg CLK;
   reg [8:0]counter;
   always@(negedge CLR or posedge clk_1M)
      begin
         if(!CLR)
            begin
                counter<=9'd0;
                CLK<=1'b0;
            end
         else if(counter==9'd2000)
                 begin
                     CLK<=~CLK;
                     counter<=9'd0;
                 end
         else
                counter<=counter+1'b1;
      end
endmodule

module smgsm(CLK,CLR,Choice,AN,seg);//数码管扫描显示模块
      input CLK; //扫描的间隔始终，控制数码管轮流点亮的频率
      input CLR; //复位信号，用于初始化状态
      input[31:0]Choice;
      output reg [7:0]AN;//位选
      output reg [7:0]seg; //段选
      reg [3:0] data;
```

```
reg [2:0] bit_sel;  //数码管计数器指示，000~111  最左到最右
always@(negedge CLR or posedge CLK)
begin
    if(!CLR)
            bit_sel<=3'b000;
    else
         bit_sel<=bit_sel+1'b1;
end
always@(*)
    begin
        case(bit_sel)
            3'b000:data<=Choice[3:0];
            3'b001:data<=Choice[7:4];
            3'b010:data<=Choice[11:8];
            3'b011:data<=Choice[15:12];
            3'b100:data<=Choice[19:16];
            3'b101:data<=Choice[23:20];
            3'b110:data<=Choice[27:24];
            3'b111:data<=Choice[31:28];
            default:data<=data;
        endcase
    end
always@(*)
    begin
        case(bit_sel)
            3'b000:AN=8'b11111110;
            3'b001:AN=8'b11111101;
            3'b010:AN=8'b11111011;
            3'b011:AN=8'b11110111;
            3'b100:AN=8'b11101111;
```

```
                3'b101:AN=8'b11011111;
                3'b110:AN=8'b10111111;
                3'b111:AN=8'b01111111;
                default:AN=8'b11111111;
            endcase
        end
    always@(*)
        begin
            case(data)
                    0:seg<=8'b00000011;
                      1:seg<=8'b10011111;
                      2:seg<=8'b00100101;
                      3:seg<=8'b00001101;
                      4:seg<=8'b10011001;
                      5:seg<=8'b01001001;
                      6:seg<=8'b01000001;
                      7:seg<=8'b00011111;
                      8:seg<=8'b00000001;
                      9:seg<=8'b00001001;
                      10:seg<=8'b00010001;
                      11:seg<=8'b11000001;
                      12:seg<=8'b01100011;
                      13:seg<=8'b10000101;
                      14:seg<=8'b01100001;
                      15:seg<=8'b01110001;
                default:seg<=8'b11111111;
            endcase
        end
endmodule
```

三、实验仿真

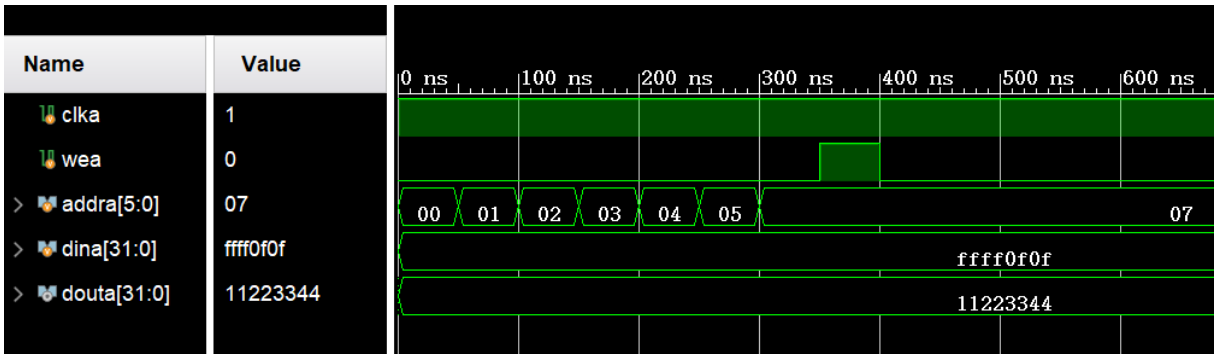
1、 仿真代码

```
module ccqfz();
//INPUT
reg clka;
reg wea;
reg [5:0]addra;
reg [31:0]dina;
//OUTPUT
wire [31:0]douta;
initial
        begin
            clka=0;
       addra=6'b000000;
       forever #50 addra = addra + 1'b1;
    end

 always
        begin
            #25 clka=~clka;
        end
/*initial
       begin
            #50 addra=5'b000000;wea=0;dina=32'hFFFF_0F0F;
            #50 addra=5'b000001;wea=0;dina=32'hFFFF_0F0F;
            #50 addra=5'b000010;wea=0;dina=32'hFFFF_0F0F;
            #50 addra=5'b000011;wea=0;dina=32'hFFFF_0F0F;
            #50 addra=5'b000100;wea=0;dina=32'hFFFF_0F0F;
            #50 addra=5'b000101;wea=0;dina=32'hFFFF_0F0F;
```

```
        #50 addra=5'b000111;wea=0;dina=32'hFFFF_0F0F;
        #50 addra=5'b000111;wea=1;dina=32'hFFFF_0F0F;
        #50 addra=5'b000111;wea=0;dina=32'hFFFF_0F0F;
    end*/


RAM_B vv(
    .clka(clka),        // input wire clka
    .wea(wea),          // input wire [0 : 0] wea
    .addra(addra),    // input wire [5 : 0] addra
    .dina(dina),        // input wire [31 : 0] dina
    .douta(douta)    // output wire [31 : 0] douta
);
endmodule
```

2、 仿真波形

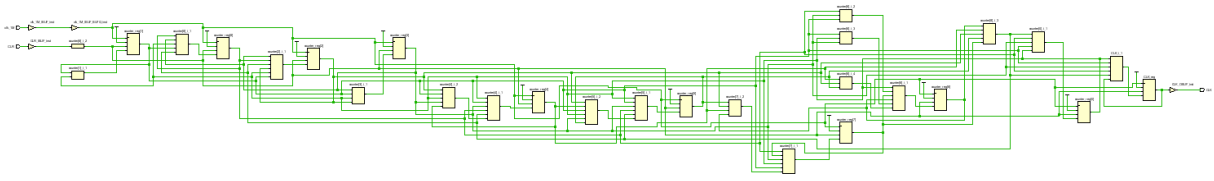
（运行仿真时，波形截图）



3、 仿真结果分析

存储器读写功能都符合预期的效果。逻辑成功。

四、电路图



五、引脚配置

（引脚约束文件的内容，描述主要配置情况）

```
set_property IOSTANDARD LVCMOS18 [get_ports clka]
set_property IOSTANDARD LVCMOS18 [get_ports {addra[7]}]
set_property IOSTANDARD LVCMOS18 [get_ports {addra[6]}]
set_property IOSTANDARD LVCMOS18 [get_ports {addra[5]}]
set_property IOSTANDARD LVCMOS18 [get_ports {addra[4]}]
set_property IOSTANDARD LVCMOS18 [get_ports {addra[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {addra[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {choice[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {choice[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {choice[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {wea[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[7]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[6]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[5]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[4]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[7]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports clk_1M]
set_property IOSTANDARD LVCMOS18 [get_ports CLR]
```

```
set_property PULLDOWN true [get_ports {addra[7]}]
set_property PULLDOWN true [get_ports {addra[6]}]
set_property PULLDOWN true [get_ports {addra[5]}]
set_property PULLDOWN true [get_ports {addra[4]}]
set_property PULLDOWN true [get_ports {addra[3]}]
set_property PULLDOWN true [get_ports {addra[2]}]
set_property PULLDOWN true [get_ports {choice[2]}]
set_property PULLDOWN true [get_ports {choice[1]}]
set_property PULLDOWN true [get_ports {choice[0]}]
set_property PULLDOWN true [get_ports {wea[0]}]
set_property PULLDOWN true [get_ports clk_1M]
set_property PULLDOWN true [get_ports CLR]

set_property PACKAGE_PIN E3 [get_ports clk_1M]
set_property PACKAGE_PIN U17 [get_ports CLR]

set_property PACKAGE_PIN F14 [get_ports {seg[7]}]
set_property PACKAGE_PIN N14 [get_ports {seg[6]}]
set_property PACKAGE_PIN J13 [get_ports {seg[5]}]
set_property PACKAGE_PIN G13 [get_ports {seg[4]}]
set_property PACKAGE_PIN F13 [get_ports {seg[3]}]
set_property PACKAGE_PIN G14 [get_ports {seg[2]}]
set_property PACKAGE_PIN M13 [get_ports {seg[1]}]
set_property PACKAGE_PIN H14 [get_ports {seg[0]}]

set_property PACKAGE_PIN C9 [get_ports {AN[7]}]
set_property PACKAGE_PIN C10 [get_ports {AN[6]}]
set_property PACKAGE_PIN D10 [get_ports {AN[5]}]
set_property PACKAGE_PIN C11 [get_ports {AN[4]}]
set_property PACKAGE_PIN M17 [get_ports {AN[3]}]
```

```
set_property PACKAGE_PIN J14 [get_ports {AN[2]}]
set_property PACKAGE_PIN K13 [get_ports {AN[1]}]
set_property PACKAGE_PIN P14 [get_ports {AN[0]}]

set_property PACKAGE_PIN V5 [get_ports {addra[7]}]
set_property PACKAGE_PIN T4 [get_ports {addra[6]}]
set_property PACKAGE_PIN V6 [get_ports {addra[5]}]
set_property PACKAGE_PIN T5 [get_ports {addra[4]}]
set_property PACKAGE_PIN T6 [get_ports {addra[3]}]
set_property PACKAGE_PIN V7 [get_ports {addra[2]}]

set_property PACKAGE_PIN V15 [get_ports {choice[2]}]
set_property PACKAGE_PIN R15 [get_ports {choice[1]}]
set_property PACKAGE_PIN U16 [get_ports {choice[0]}]

set_property PACKAGE_PIN V14 [get_ports {wea[0]}]
set_property PACKAGE_PIN N17 [get_ports clka]

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk_1M]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clka]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets CLR]
```

六、思考与探索

1、 实验结果记录：

（实验操作的过程及结果记录）

存储器地址	初始化数据	读出数据	是否写入	写入新数据	读出数据
000000	1122_3344	1122_3344	否	FFFF_0F0F	1122_3344
000001	8899_aabb	8899_aabb	否	FFFF_0F0F	8899_aabb
000010	5555_6666	5555_6666	否	FFFF_0F0F	5555_6666

000011	2021_0303	2021_0303	否	FFFF_0F0F	2021_0303
000100	fff8_0f0f	fff8_0f0f	否	FFFF_0F0F	fff8_0f0f
000101	cabd_1234	cabd_1234	否	FFFF_0F0F	cabd_1234
000111	0000_0000	0000_0000	是	FFFF_0F0F	FFFF0F0F
000111	FFFF0F0F	FFFF0F0F	是	1111_1111	1111_1111

2、 实验结论：

实验结果符合预期结果，逻辑正确，实验成功。

3、 问题与解决方案：

在板级验证中，我需要按两次 clka 才能读出数据；

因为在 IP 核设置中勾选了“Primitives Output Register”选项，在第二个 clk 上跳沿才能读出数据。去掉勾选就解决了。

4、 思考题：

1.） 在仿真测试和板级测试中，你是如何确认存储器单元被写入成功的？请具体说明。

答：先读取某个储存地址的内容，再写入数据到该存储地址，再次读取该存储地址的内容，如果已经变成了新写入的数据，则存储器单元被写入成功。

2.） 在板级实验中，你执行存储器写操作时，按下 clk_dm 时钟键后，观察存储器读出的数据，是新写入的数据还是原来的数据？请你分析为什么会是这样的现象。

答：读出的数据是新写入的数据。写入数据速度快于读出数据。

3. ）接上题，请通过仿真测试，观察写操作的 clk_dm 来临后，读出的数据是什么？和板级验证结果一致吗？分析原因。

答：不一致；在仿真中有延迟，先显示了写之前的数据，然后显示了写入后的数据。

6.） 谈谈你在实验中碰到了哪些问题？又是如何解决的？

答：在板级验证中，我需要按两次 clka 才能读出数据；因为在 IP 核设置中勾选了“Primitives Output Register”选项，在第二个 clk 上跳沿才能读出数据。去掉勾选就解决了。