# 模块实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 姓名 | 叶福伦 | 学号 | 21051331 |
| 专业 | 计算机科学与技术 | 班级 | 21052313 |
| 指导教师 | 髙志刚 | 课程名称 | 计算机组成原理课程设计（甲） |
| 实验序号 | 06 | 实验名称 | 存储器设计实验 |
| 实验时间 | 05.10 | 实验地点 | 1教223 |

# 成绩表

|  |  |  |
| --- | --- | --- |
| **考核项目** | **考查分项** | **得分** |
| **一、实验方案设计（30分）** | **实验目的与要求（5分）** |  |
| **模块构成与连接（15分）** |  |
| **板级验证方案（10分）** |  |
| **二、FPGA程序设计（20分）** | **程序源代码（15分）** |  |
| **管脚约束程序源代码（5分）** |  |
| **三、仿真实验（15分）** | **仿真代码（5分）** |  |
| **仿真波形及结果分析（10分）** |  |
| **四、板级实验（15分）** | **实验操作（5分）** |  |
| **结果记录（5分）** |  |
| **结果分析（5分）** |  |
| **五、思考与探索（10分）** | **拓展提升（10分）** |  |
| **格式（10分）** | **书写规范、图表清晰、表达清楚** |  |
| **合计** | |  |

# 一、实验方案设计

1. 实验目的与要求（5分）

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；

学习存储器的结构及读写原理，掌握存储器的设计方法

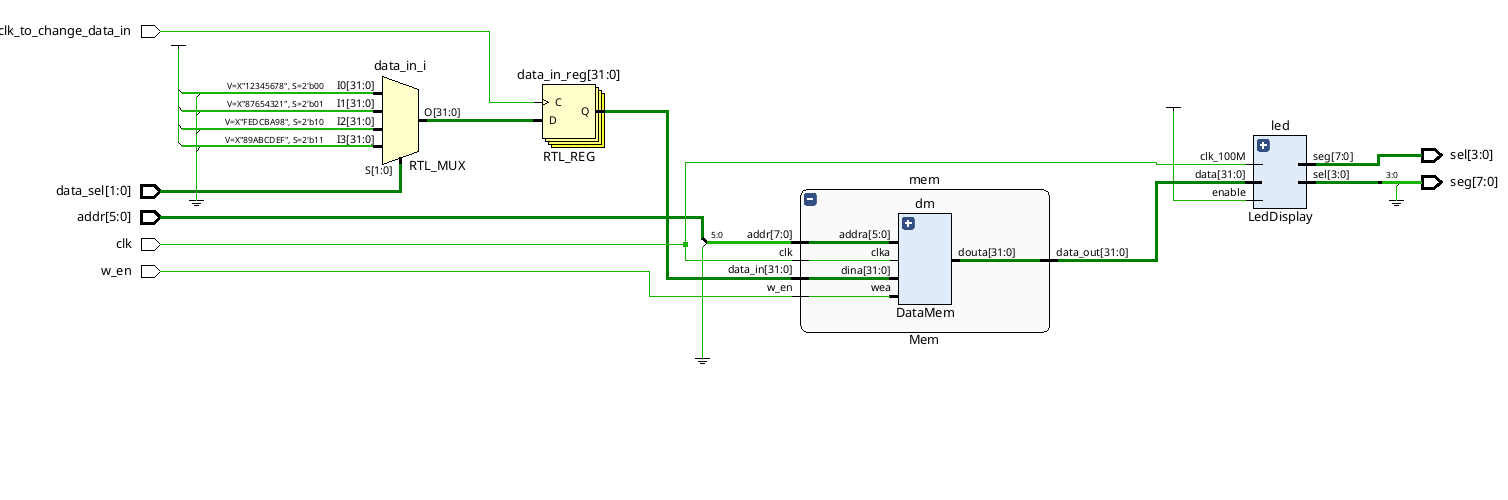
使用Vivado或者ISE工具，按照上述步骤新建一个64×32位的Single Port RAM，并使用COE关联文件初始化其内容。

新建一个存储器模块，引用上述的存储器IP核，连接实例与存储器模块的端口。

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。

1. 模块构成与连接（15分）

（请附图，并说明各模块的功能，以及模块间的连接关系。也可附开发工具中显示的电路模块连接图，加以说明。）

mem模块包装Memory IP核，使得内存访问更加易用。led模块负责把从存储器中读出的数据显示在数码管上。RTL\_REG负责从一组特定的数据中选出一个数据写入内存

1. 板级验证方案（10分）

（说明顶层测试模块的输入/输出信号，及对应使用的I/O设备；可附图说明）

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 信号名 | 使用的IO设备 | 说明 |
| input | clk | 全局时钟 |  |
| w\_en | 逻辑开关 | 写允许 |
| addr[4:0] | 一组逻辑开关 | 读写地址 |
| clk\_to\_change\_data\_in | 按键 | 产生打入数据脉冲 |
| data\_sel[1:0] | 一组逻辑开关 | 数据 |
| output | seg[7:0] | 数码管 | 驱动数码管显示数据 |
| sel[3:0] |

# 二、FPGA程序设计

1. 实验程序源代码（25分）

（实验各个模块的代码，包含功能注释）

//DataMem.coe

memory\_initialization\_radix=16;

memory\_initialization\_vector=000502b3 00058333 000603b3 0002ae03 00428293 00430313;

//Mem.v

//只实现了全字访问, 半字访问和字节访问并没有实现

//这让我深刻地理解为什么c语言结构体内存布局要对齐了

module Mem (

input clk,

input w\_en,

//input [1:0] mod,

input [7:0] addr, //存储器共2^6 = 64个字 ,对字节进行编码后2位会被丢弃

input [31:0] data\_in,

output [31:0] data\_out

);

DataMem dm (

.clka(clk), // input wire clka

.wea(w\_en), // input wire [0 : 0] wea

.addra(addr[7:2]), // input wire [5 : 0] addra

.dina(data\_in), // input wire [31 : 0] dina

.douta(data\_out) // output wire [31 : 0] douta

);

endmodule

//Mem.top.v 顶级模块

`include "../include/LedDisplay.v"

module MemTopper (

input clk,

input w\_en,

input [5:0] addr,

input clk\_to\_change\_data\_in,

input [1:0] data\_sel,

output [7:0] seg,

output [3:0] sel

);

reg [31:0] data\_in;

wire [31:0] data\_out;

always @(posedge clk\_to\_change\_data\_in) begin

case (data\_sel)

2'b00: data\_in <= 32'h12345678;

2'b01: data\_in <= 32'h87654321;

2'b10: data\_in <= 32'hfedcba98;

2'b11: data\_in <= 32'h89abcdef;

endcase

end

Mem mem (

.clk(clk),

.w\_en(w\_en),

.addr(addr),

.data\_in(data\_in),

.data\_out(data\_out)

);

LedDisplay led(

clk,

data\_out,

1'b1,

seg,

sel

);

endmodule

1. FPGA管脚约束代码（5分）

（引脚约束文件的内容，备注使用的具体设备）

clk使用全局时钟，w\_en使用逻辑开关，addr[5:0]使用逻辑开关，data\_sel[1:0]使用逻辑开关，clk\_to\_change\_data\_in使用按键，seg[7:0]和sel[3:0]一起驱动数码管

set\_property BITSTREAM.GENERAL.COMPRESS TRUE [current\_design]

set\_property -dict {IOSTANDARD LVCMOS18 PACKAGE\_PIN H4} [get\_ports clk]

set\_property PACKAGE\_PIN R4 [get\_ports clk\_to\_change\_data\_in]

set\_property IOSTANDARD LVCMOS18 [get\_ports clk\_to\_change\_data\_in]

set\_property IOSTANDARD LVCMOS18 [get\_ports w\_en]

set\_property IOSTANDARD LVCMOS18 [get\_ports {addr[5]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {addr[4]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {addr[3]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {addr[2]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {addr[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {addr[0]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {data\_sel[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {data\_sel[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 {sel[3]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {sel[2]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {sel[1]}]

set\_property IOSTANDARD LVCMOS18 [get\_ports {sel[0]}]

set\_property PACKAGE\_PIN T3 [get\_ports {addr[5]}]

set\_property PACKAGE\_PIN U3 [get\_ports {addr[4]}]

set\_property PACKAGE\_PIN T4 [get\_ports {addr[3]}]

set\_property PACKAGE\_PIN V3 [get\_ports {addr[2]}]

set\_property PACKAGE\_PIN V4 [get\_ports {addr[1]}]

set\_property PACKAGE\_PIN Y4 [get\_ports w\_en]

set\_property PACKAGE\_PIN W4 [get\_ports {addr[0]}]

set\_property PACKAGE\_PIN Y6 [get\_ports {data\_sel[1]}]

set\_property PACKAGE\_PIN W7 [get\_ports {data\_sel[0]}]

set\_property PACKAGE\_PIN H19 [get\_ports {seg[7]}]

set\_property PACKAGE\_PIN G20 [get\_ports {seg[6]}]

set\_property PACKAGE\_PIN J22 [get\_ports {seg[5]}]

set\_property PACKAGE\_PIN K22 [get\_ports {seg[4]}]

set\_property PACKAGE\_PIN K21 [get\_ports {seg[3]}]

set\_property PACKAGE\_PIN H20 [get\_ports {seg[2]}]

set\_property PACKAGE\_PIN H22 [get\_ports {seg[1]}]

set\_property PACKAGE\_PIN J21 [get\_ports {seg[0]}]

set\_property PACKAGE\_PIN L21 [get\_ports {sel[3]}]

set\_property PACKAGE\_PIN M22 [get\_ports {sel[2]}]

set\_property PACKAGE\_PIN M21 [get\_ports {sel[1]}]

set\_property PACKAGE\_PIN N22 [get\_ports {sel[0]}]

set\_property PULLDOWN true [get\_ports w\_en]

set\_property PULLDOWN true [get\_ports {addr[5]}]

set\_property PULLDOWN true [get\_ports {addr[4]}]

set\_property PULLDOWN true [get\_ports {addr[3]}]

set\_property PULLDOWN true [get\_ports {addr[2]}]

set\_property PULLDOWN true [get\_ports {addr[1]}]

set\_property PULLDOWN true [get\_ports {addr[0]}]

set\_property PULLDOWN true [get\_ports {data\_sel[1]}]

set\_property PULLDOWN true [get\_ports {data\_sel[0]}]

# 三、仿真实验

（只对CPU模块仿真，无需对板级验证的顶层测试模块仿真）

1. 仿真代码（5分）

（含仿真源代码、仿真验证方案）

`timescale 1ns / 1ps

module MemTester();

// Mem Parameters

parameter PERIOD = 10;

reg clk = 0;

reg w\_en = 0 ;

reg [7:0] addr = 0 ;

reg [31:0] data\_in = 0 ;

wire [31:0] data\_out;

initial begin

forever #(PERIOD/2) clk=~clk;

end

Mem mem(

.clk(clk),

.w\_en(w\_en),

.addr(addr),

.data\_in(data\_in),

.data\_out(data\_out)

);

initial begin

addr = 8'b0000\_0000;

#20

addr = 8'b0000\_0100;

#20

addr = 8'b0000\_1000;

#20

addr = 8'b0000\_1100;

#20

addr = 8'b0001\_0000;

#20

addr = 8'b0001\_0100;

#20

data\_in = 32'hffff\_ffff;

w\_en = 1;

#20

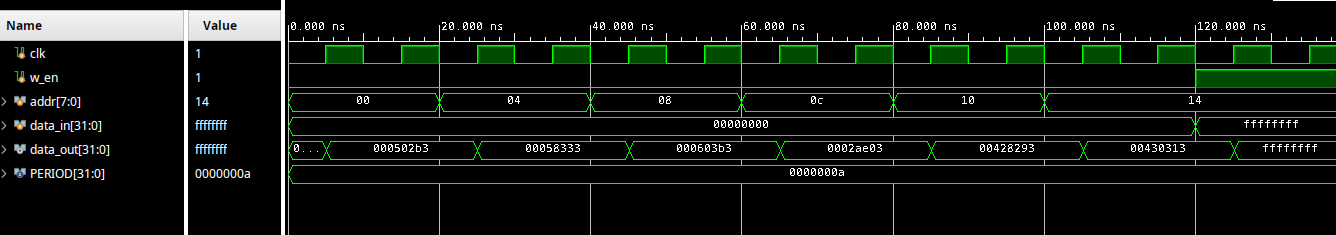
$finish;

end

endmodule

1. 仿真波形及结果分析（15分）

（仿真波形截图，并对仿真波形进行分析）

如图前6个阶段addr依次指向前六个内存字，跟.coe初始化文件比较可知，该存储器正确地按照了.coe初始化存储器且读出功能有效.最后一个阶段检查是否能够写入存储器。令写允许信号为1, 写入0xFFFF\_FFFF。对应读出数据变成了0xFFFF\_FFFF说明写入是有效的

# 四、板级实验

1. 板级实验操作说明（5分）

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格中，要求存储器读和写功能都被有效测试。

1. 板级实验结果记录（5分）

（以表格形式展现，具体见教材上实验结果记录表）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 存储器地址 | 初始化数据 | 读出数据 | 写入数据 | 读出新数据 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. 结果分析与结论（5分）

（分析实验结果，给出实验结论）

# 五、思考与探索

（10分）

1. 问题与解决方案

（整个实验过程中发生了什么问题？你是如何解决的。）

实例化时addr时定义成了8位地址线，而存储器只有6位地址线: 新定义一个模块。

包装IP memory核把把8位地址线的后两位丢弃，使用高6位连接IP memory核。如果输入的是非字对齐的内存地址，按道理应该有一个标志寄存器设为1表明抛出异常，但是这里是返回所在的内存字的数据。

1. 思考题

（力所能及，尝试实践或回答教材上的思考与探索题目，至少完成1道）

1. 写入成功是data\_out也会变成写入内存的新值，所以可以通过data\_out来判断。

5. 内存非字对齐读的话一种实现是使用位掩码（bit mask）和位偏移量（bit offset）来实现。用位掩码选中要访问的数据，再使用位偏移量来指定字单元中的具体字节。

非字对齐写需要进行字节掩码操作，这可以通过在内存模块中添加一个额外的字节掩码输入端口来实现。

总之，支持非字对齐访问的存储器使得存储器实现复杂性能下降甚至系统异常。另外，非字对齐访问还可能会导致一些未定义的行为。例如，对于跨越两个字（或更多）的非对齐访问，一些体系结构可能会将它们视为两个独立的操作。这可能会导致一个操作只修改了其中一个字，而另一个操作则修改了另一个字，从而导致不一致的状态。总之应该避免非对齐访问。高级语言的数据访问也考虑到了这一点。数据都是对齐的。

# 六、实验心得体会、意见建议

（如果是小组合作，请给出具体分工、分别给出心得体会）

作为一个学习计算机组成原理实验课程的学生，我认为存储器设计实验是一个非常有意义的实践项目。在这个实验中，我学习了如何设计和实现一个基本的存储器，并深入了解了存储器的工作原理和基本组成部分。

这个存储器设计实验是一个非常有意义的实践项目。通过参与这个实验，我深入了解了存储器的工作原理和基本组成部分，学习了如何使用VHDL语言进行存储器的设计和仿真，以及掌握了重要的团队合作和沟通技能。这些经验对我未来的学习和职业生涯都将非常有帮助。