****

计算机组成原理（甲）

实 验 报 告

|  |  |
| --- | --- |
| **学 院** | 网络空间安全学院 |
| **专 业** | 网络工程 |
| **班 级** | 19272401 |
| **学 号** | 19061440 |
| **学生姓名** | F001 |
| **教师姓名** | 袁理峰 |
| **完成日期** | 2020.12.11 |
| **成 绩** |  |
| 实验五 存储器设计实验 | | |
| 1. **实验目的** 2. 掌握灵活运用Verilog HDL进行各种描述与建模的技巧和方法。 3. 学习在ISE中设计生成Memory IP核的方法。 4. 学习存储器的结构及读写原理，掌握存储器的设计方法。 | | |
| 1. **实验原理**   存储器用于存储数据和指令。MIPS32处理器的地址总线为32位，数据总线为32位，但是存储器按字节编址，因此存储器容量位4G，地址范围0x00000000~0xFFFFFFFF。但MIPS处理器绝大部分指令对存储器的访问尺寸位字，即访问地址必须按照4字节的边界严格对齐。至于MIPS的一个字的字节大小端问题，选择小端格式。高字节在高地址，低字节在低地址。  本实验要求为MIPS处理器设计一个256\*8位的物理存储器，具有读/写功能，按字节编址，按字访问，即64\*32位。输入端口为Mem\_Read、Mem\_Addr、M\_W\_Data、Mem\_Write，输出端口为M\_R\_Data。分别为读控制信号，存储器的访问地址，存储器的读出数据，写控制信号和存储器的写入数据。 | | |
| 1. **实验环境**   所用电脑的软硬件配置：自己的笔记本电脑、Windows10操作系统  实验所用的软件：ISE design suite | | |
| 1. **主要操作步骤及实验结果记录（不能光截图，要有相应的文字说明）**   （对实验过程中的主要操作步骤进行描述，并随时记录实验过程中观察到的结果，必要时可辅助截图）  任务一：生成FPGA内置的存储器IP核  (1)建立并编辑一个关联文件。在工程目录下生成一个扩展名为\*.coe的纯文本文件。该文件是用于Memor IP核的初始化初始化文件，文件包括两行，具体内容如下：    第一行用来表示数据是几进制的存在，第二行定义初始化向量。  (2)新建一个Memory IP核。在工程管理区的任意位置邮寄，New Source->New Source Wizard->IP(CORE Generator & Architecture Wizard)路径即为默认路径。之后需要选择IP核，找到Memories & Storage Elements->RAMs & ROMs->Block Memory Generator.带有时钟同步，但是相对麻烦。之后进入IP核的设置  (3)Memory IP核参数设置。共有6页，依次按照如下图配置进行。    1 2    3 4    5 6  (4)调用存储器模块。双击过程管理区的View HDL Instruction Template，在右侧的代码区会给出调用模板，将其复制到顶层模块中，修改模块实例名称和连接端口参数，就可以像一般模块引用了。  任务二：在Xilinx ISE中创建工程，编辑程序源代码，然后编译，综合；若编译出错，则需要修改程序代码，直至正确。    生成完毕的RAM模块。    上图为RAM所显示的模板情况。此处为了简便起见，直接忽略了顶层代码，在test文件中直接进行相关调试，并且成功，因此认为我们的RAM模块是正常的。  任务三：编写激励代码，观察仿真波形，若逻辑验证有误，则修改代码，重新编译、仿真，直至正确。  左图为所描写的激励代码。  下图为展示的波形。  对比可得知，波形正确，相关代码正确，RAM板块正常。 | | |
| 1. **实验分析总结及心得**   （结合所学知识对实验过程中观察到的实验结果进行分析总结，以便加深对知识的理解，并总结通过实验学到的知识或技术）  在进行思考题的实践中，发现ISE又出现了一些奇奇怪怪的错误。例如修改完test文件之后只能进行一次的仿真波形的呈现。第二次进行的时候总是会强行退出，目前原因尚不明朗。    以上为此次针对思考题进行的仿真。表格如下：   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 存储器地址 | 初始化数据 | 读出数据 | 写入新数据 | 读出数据 | | 6’b000001 | 8’h00000820 | 8’h00000820 | 4’b0011 | 4’b0011 | | 6’b000010 | 8’h00632020 | 8’h00632020 | 4’b0100 | 4’b0100 | | 6’b000011 | 8’h00010fff | 8’h00010fff | 4’b0101 | 4’b0101 | | 6’b000100 | 8’h20006789 | 8’h20006789 | 4’b0110 | 4’b0110 | | 6’b000101 | 8’Hffff0000 | 8’hffff0000 | 4’b0111 | 4’b0111 | | 6’b000110 | 8’h0000FFFF | 8’h0000FFFF | 4’b1000 | 4’b1000 | | 6’b000111 | 8’h88888888 | 8’h88888888 | 4’b1001 | 4’b1001 | | 6’b001000 | 8’h99999999 | 8’h99999999 | 4’b1010 | 4’b1010 |   根据上述表格我们发现，读出数据与初始化关联文件中的数据完全一致，说明相关操作正常。在进行写操作，覆盖初始化数据之后，再执行读操作，这些单元的数据全部被改写。但是相关数据会根据时钟信号进行覆写，若不谨慎编程，将会产生想刷新A存储器中的数据却刷新到A+1的数据区中。  若想基于上述情况更改为ROM，在接口上仅需要将写信号至于0时，且不再改变即可。并且需要去除M\_W\_Data这一接口。 | | |