6.8 RAM 与寄存器堆实验

1.实验介绍

在本次实验中,我们将使用 Verilog HDL 语言实现 RAM 以及寄存器堆的设计和仿真。

2.实验目标

- 深入了解 RAM 与寄存器堆的原理。
- 用 Logisim 画出一个包含 16 个寄存器的寄存器堆原理图。
- 学习使用 Verilog HDL 语言设计实现 RAM 以及寄存器堆。

3.实验原理

1) RAM

半导体随机读写存储器，简称 RAM，它是数字计算机和其他数字系统的重要存储部件，可存放大量的数据。图 6.25 所示为 RAM 的逻辑结构图，其主体是存储矩阵，另有地址译码器和读写控制电路两大部分。读写控制电路中加有片选控制和输入输出缓冲器等，以便组成双向 I/O 数据线。

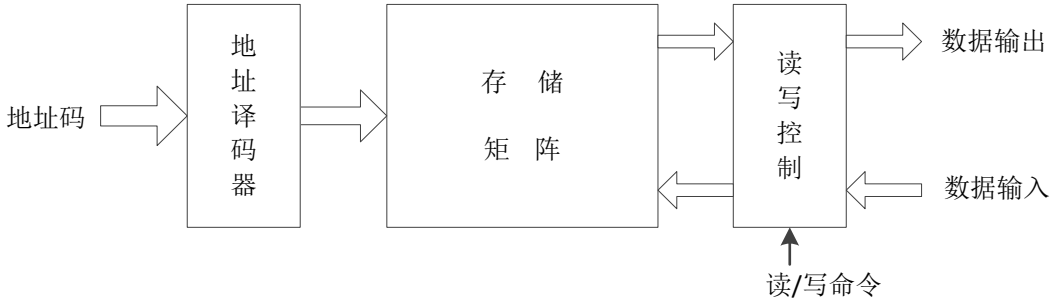


图 6.25 RAM 的逻辑结构图

RAM 有三组信号线:

- 地址线：单向，传送地址码（二进制数），以便按地址码访问存储单元;
- 数据线：双向，将数据码（二进制数）送入存储矩阵或从存储矩阵读出;
- 读/写命令线：单向控制线，分时发送这两个命令，要保证读时不写，写时不读。

如图 6.26 所示，为本实验所需要实现的 RAM 的示意图。

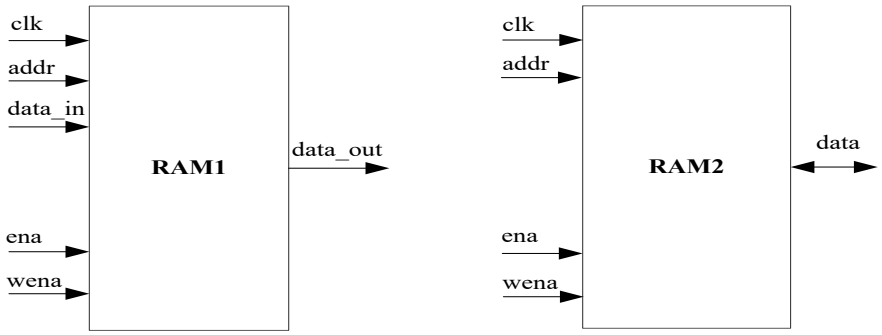


图 6.26 本实验的 RAM 示意图

- 接口定义:

```
module ram (
input clk,  //存储器时钟信号，上升沿时向 ram 内部写入数据
inputena, //存储器有效信号，高电平时存储器才运行，否则输出 z
input  wena,  //存储器读写有效信号，高电平为写有效，低电平为读有效，与 ena
                    同时有效时才可对存储器进行读写
input [4:0] addr, //输入地址，指定数据读写的地址
input [31:0] data_in, //存储器写入的数据，在 clk 上升沿时被写入
output [31:0] data_out //存储器读出的数据
)
```

提示：可以使用 reg 数组来实现，大小至少 1024bit。

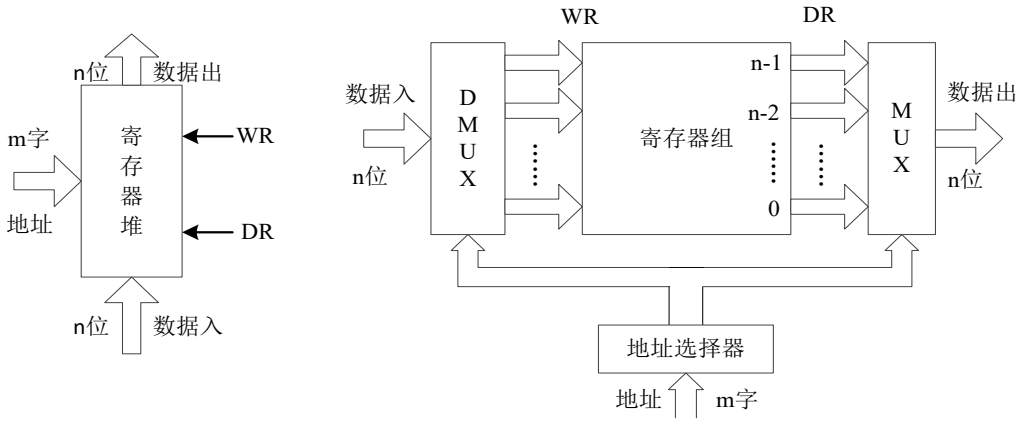
测试时可以利用$readmemh("文件名", 数组名）语句使用文件初始化 reg 数组。

```
module ram2 (
    input clk,   //存储器时钟信号，上升沿时向 ram 内部写入数据
    input ena,   //存储器有效信号，高电平时存储器才运行，否则输出 z
    input wena,  //存储器读写有效信号，高电平为写有效，低电平为读有效，与
                  ena 同时有效时才可对存储器进行读写
    input [4:0] addr,   //输入地址，指定数据读写的地址
    inout [31:0] data,  //存储器数据线，可传输存储器读出或写入的数据。
                          写入的数据在 clk 上升沿时被写入
)
```

2）寄存器堆（regfiles）

一个寄存器是由 n 个触发器或锁存器按并行方式输入且并行方式输出连接而成。它只能记忆 1 个字，1 个字的长度等于 n 个比特。当需要记忆多个字时，一个寄存器就不够用了，在这种情况下，需要使用由多个寄存器组的寄存器堆。

图 6.27 为寄存器堆的逻辑结构与原理示意图，它由寄存器组、地址译码器、多路选择器 MUX 及多路分配器 DMUX 等部分组成。向寄存器写数据或读数据，必须先给出寄存器的地址编号。写数据时，控制信号 WR 有效，待写入的数据经 DMUX 送到地址给定的某个寄存器。读数据时，控制信号 RD 有效，由地址给定的某个寄存器的数据内容经多路开关 MUX 送出。由于读/写工作是分时进行的，所以寄存器组在逻辑上能满足写数据或读数据的需要。



（a）逻辑结构图（b）原理示意图

图 6.27 寄存器堆的逻辑结构

图 6.28 给出了由四个 4 位寄存器组成的具有两个数据输出端口的寄存器堆原理图，它可以同时从寄存器堆中取出两个数据，和加法器一起构成一个简单的运算通路。其主要由 1 个 2-4 译码器（ENB 为使能端，S_1、S_2 为两位编码输入端，D_1~D_4 为译码输出端 ）、4 个 4 位寄存器（ENB 为使能端，A~D 为数据输入端，Q_1~Q_4 为数据输出端）和 2 个 4 位 4 选 1 数据选择器（ENB 为使能端，S_1~S_4 为 4 路数据输入端，C_1、C_2 为选择控制端，D 为数据输出端）组成。读数据时，读写控制信号 we 为低电平，由地址 raddr1 和 raddr2 指定的两个寄存器的数据分别送到 rdata1 和 rdata2。写数据时，待存入的数据放到输入端 wdata，并给出写的地址 waddr，当读写控制信号 we 为高电平时，waddr 指定的寄存器在时钟上升沿将数据写入到该寄存器。

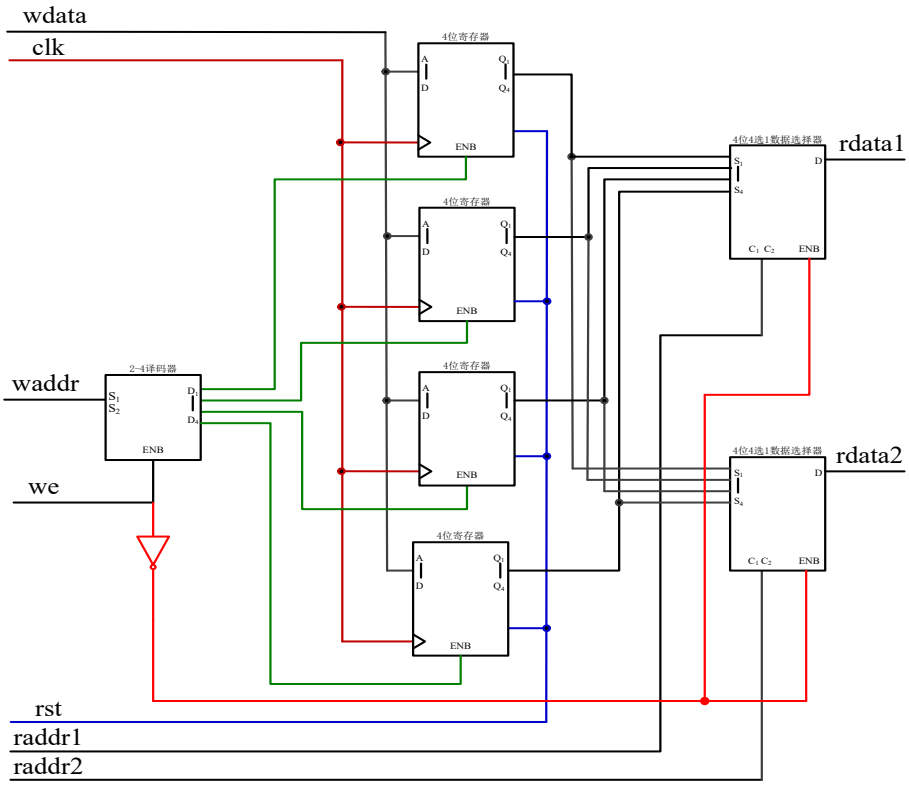


图 6.28 由 4 个寄存器组成的寄存器堆原理图

本实验所要建模的寄存器堆如图 6.29 所示，

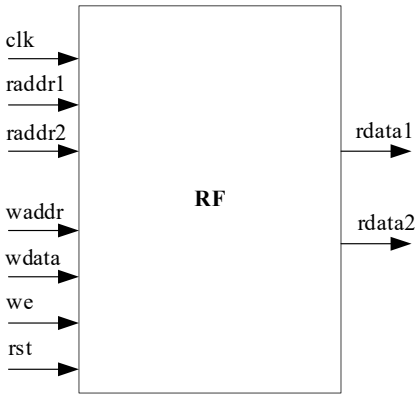


图 6.29 本实验所要建模的寄存器堆功能框图

- 接口定义：

```
module Regfiles(
input clk, //寄存器组时钟信号，下降沿写入数据
input rst, //异步复位信号，高电平时全部寄存器置零
input we, //寄存器读写有效信号，高电平时允许寄存器写入数据，
                    低电平时允许寄存器读出数据
input [4:0] raddr1, //所需读取的寄存器的地址
input [4:0] raddr2, //所需读取的寄存器的地址
input [4:0] waddr, //写寄存器的地址
input [31:0] wdata, //写寄存器数据，数据在 clk 下降沿时被写入
output [31:0] rdata1, //raddr1 所对应寄存器的输出数据
output [31:0] rdata2  //raddr2 所对应寄存器的输出数据
);
```

注：要求使用以前实验中的译码器、寄存器、以及选择器的模块实例化来实现。

4.实验步骤

1.参考图 6.8.4，用 logisim 画出由 16 个 4 位寄存器组成的寄存器堆电路原理图，并验证逻辑。

2.新建 Vivado 工程，编写各个模块。

3.用 ModelSim 仿真测试各模块。

4.按照要求书写实验报告。