《数字逻辑》实验报告

姓名	汤英琦	年级	2024 级
学号	20241280	专业、班级	计算机科学与技术卓越 01

实验名称	存储器实验		
实验时间	2025/10/29	实验地点	DS1401
实验成绩		实验性质	□验证性 □设计性 □综合性

教师评价：

□算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；

□实验结果正确； □语法、语义正确； □报告规范；

评语：

评价教师签名（电子签名）：

一、实验目的

（1） 掌握随机存储器原理，学会 FPGA 内部存储器控制器的设计方法。

（2） 掌握单端口与双端口 RAM（随机存储器）设计与实现。

（3） 掌握 FIFO（先入先出存储队列）的设计与实现。

二、实验项目内容

1） 利用开发板片内存储器单元实现单端口 RAM 设计（带异步读和同步读两种模式）。

2） 实现双端口（同步与异步）RAM 设计。

3） 实现 FIFO 设计，FIFO 由存储单元队列或阵列构成，FIFO 没有地址，第一个被写入队列的数据也是第一个从队列中读出的数据。

三、实验设计

实验原理：

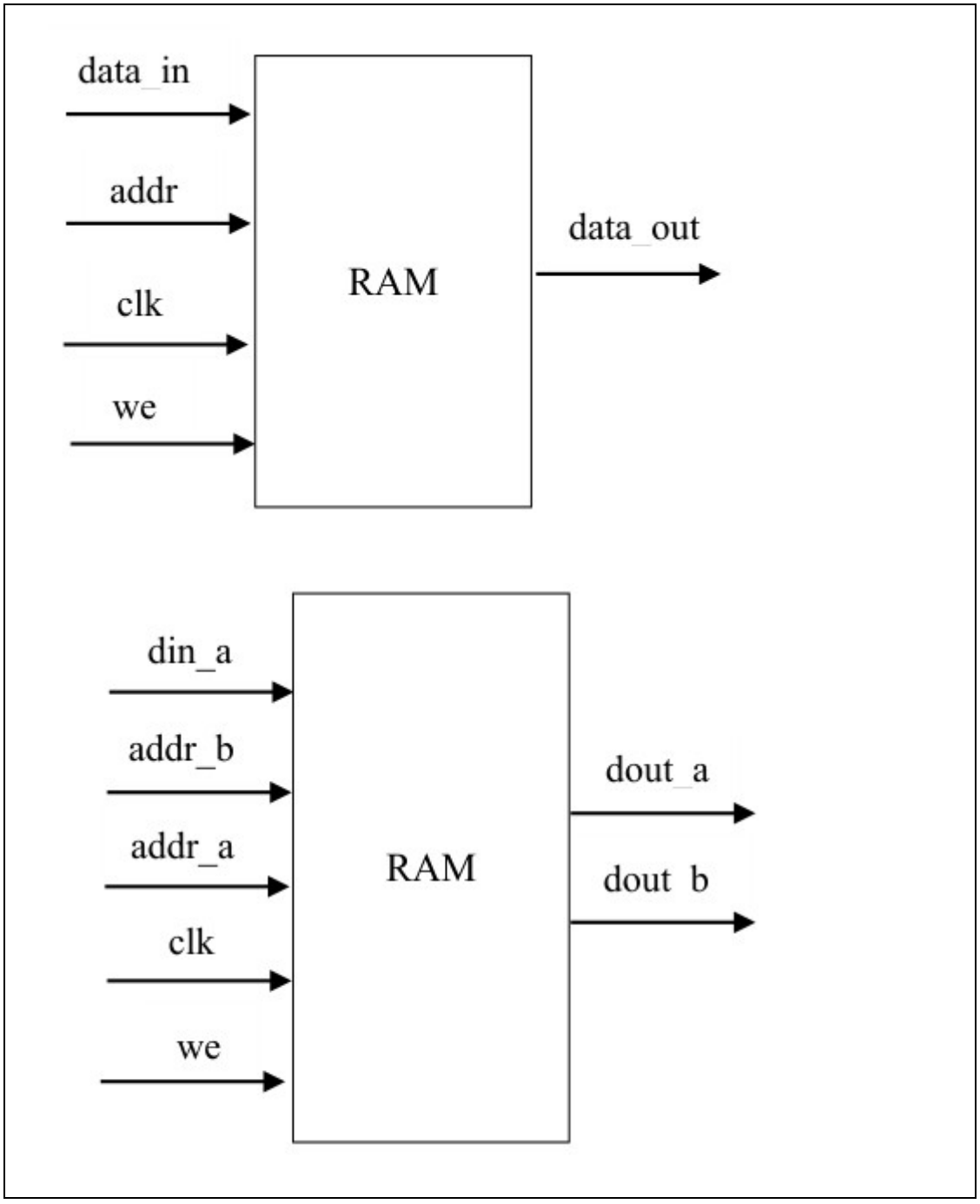
RAM（random access memory）又称“随机存储器”，存储单元的内容按需要随意取出或者存入，速度很快，但断电时将丢失数据，所以一般被作为临时数据的存储媒介。Basys3 开发板上拥有 1,800 Kbits 快速 RAM 块，可以根据需求定制 ROM、RAM 或者 FIFO。

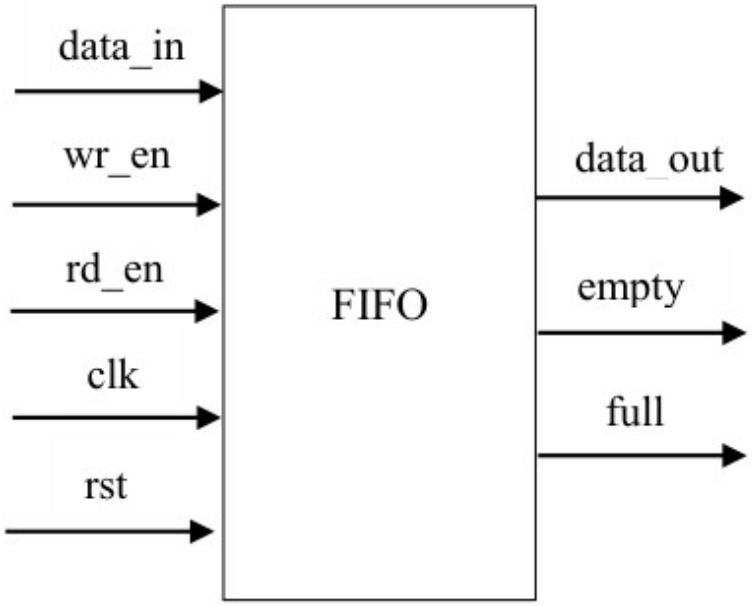
单端口 RAM 设计（带异步读和同步读两种模式），在时钟（clk）上升沿，采集地址（addr）、输入数据（data_in）、执行相关控制信息。当写使能（we）有效，则执行写操作，否则执行读取操作。同步与异步设计仅针对读操作：对于异步 RAM 而言，读操作为异步，即地址信号有效时，控制器直接读取 RAM 阵列；对于同步 RAM 而言，地址信号在时钟上升沿被采集，并保存在寄存器中，然后使用该地址信号读取 RAM 阵列。

双端口（同步与异步）RAM，相对于单端口 RAM 而言，双端口 RAM 存在两个存取端口，并且可独立进行读写操作，具有自己的地址（addr_a、addr_b）、数据输入（din_a、din_b）/输出端口（dout_a、dout_b）以及控制信号。双端口 RAM 常用于视频/图像处理设计中

FIFO 是一个先入先出的存储队列，和 RAM 不同的是 FIFO 没有地址，第一个被写入队列的数据也是第一个从队列中读出的数据。FIFO 可以在输入输出速率不匹配时，作为临时存储单元；可用于不同时钟域中间的同步；输入数据路径和输出数据路径之间数据宽度不匹配时，可用于数据宽度调整电路。

原理图：


data_in
addr
clk
we
RAM
data_out
din_a
addr_b
addr_a
clk
we
RAM
dout_a
dout_b



信号	功能
data_in	数据输入
data_out	数据输出

wr_en	写使能
rd_en	读使能
clk	FIFO 时钟
rst	FIFO reset
empty	表示 FIFO 空
full	表示 FIFO 满

四、实验过程或算法

实验步骤：

1） 在 Vivado 中新建项目，设计实现带异步读模式的单端口 RAM，在时钟（clk）上升沿，采集地址（addr）、输入数据（data_in）、 执行相关控制信息。当写使能（we）有效，则执行写操作，否则执行读取操作。读操作为异步，即地址信号有效时，控制器直接读取 RAM 阵列。

2） 编写仿真文件，观察波形图，验证其正确性，并下载到开发板进行验证。

3） 在 Vivado 中新建项目，设计实现带同步读模式的单端口 RAM，与异步读模式不同的是，地址信号在时钟上升沿被采集，并保存在寄存器中，然后使用该地址信号读取 RAM 阵列。

4） 编写仿真文件，观察波形图，验证其正确性，并下载到开发板进行验证。

5） 在 Vivado 中新建项目，实现双端口（同步或异步）RAM 设计，相对于单端口 RAM 而言，双端口 RAM 存在两个存取端口，并且可独立进行读写操作，具有自己的地址（addr_a、addr_b）、数据输入（din_a、din_b）/输出端口（dout_a、dout_b）。

6） 通过仿真、下板验证其正确性。

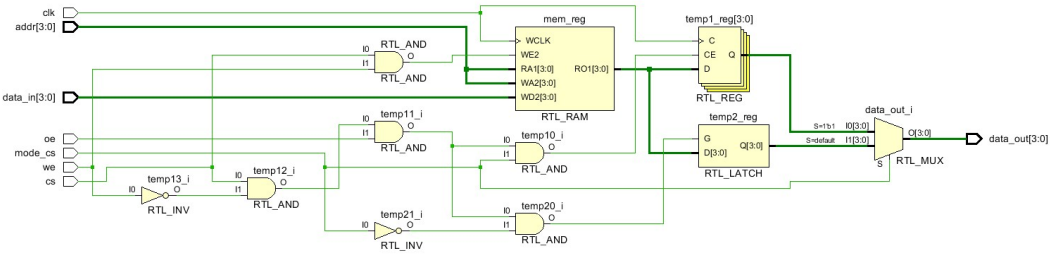
7） 在 Vivado 中新建项目，实现 FIFO 设计，FIFO 由存储单元队列或阵列构成，和 RAM 不同的是 FIFO 没有地址，第一个被写入队列的数据也是第一个从队列中读出的数据，FIFO 有表示“空”和“满”的状态输出信号。

8） 通过仿真、下板验证其正确性。

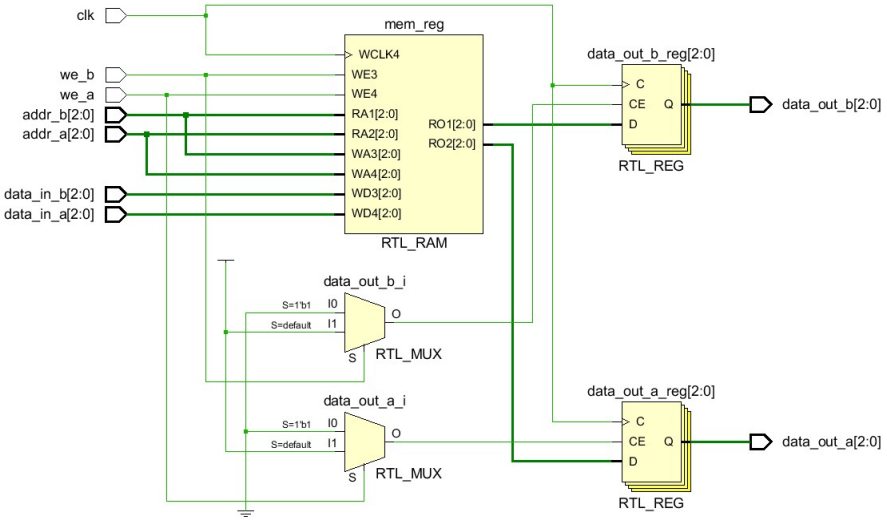
核心代码：

```
module SP_RAM
#(parameter DW = 4,AW = 4)   // data_width & address width
(
    input [AW-1:0]addr,
    input clk,
    input cs,                // chip select
    input oe,                // output enable
    input we,                // write enable
    input mode_cs,           // mode_cs = 1 sync  mode_cs = 0 asynch
    input [DW-1:0]data_in,      // data in
    output [DW-1:0]data_out     // data output
    );

parameter DP = 1 << AW;      // depth
reg [DW-1:0]mem[0:DP-1];
reg [DW-1:0]temp1;
reg [DW-1:0]temp2;


// initial
integer i;
initial begin
    for(i = 0; i < DP; i = i + 1) begin
        mem[i] = 4'h00;
    end
end

// write block
always@(posedge clk)
begin
    if (cs && we)
        begin
            mem[addr] <= data_in;
        end
    else
        begin
            mem[addr] <= mem[addr];
        end
end

//  sync-read block
always@(posedge clk)
begin
    if (cs && !we && oe && mode_cs)
        begin
            temp1 <= mem[addr];
        end
end

// asynch-read block
always @(*)
begin
    if (cs && !we && oe && !mode_cs)
        begin
            temp2 <= mem[addr];
        end
end


assign data_out = (mode_cs)? temp1 : temp2;

endmodule
```

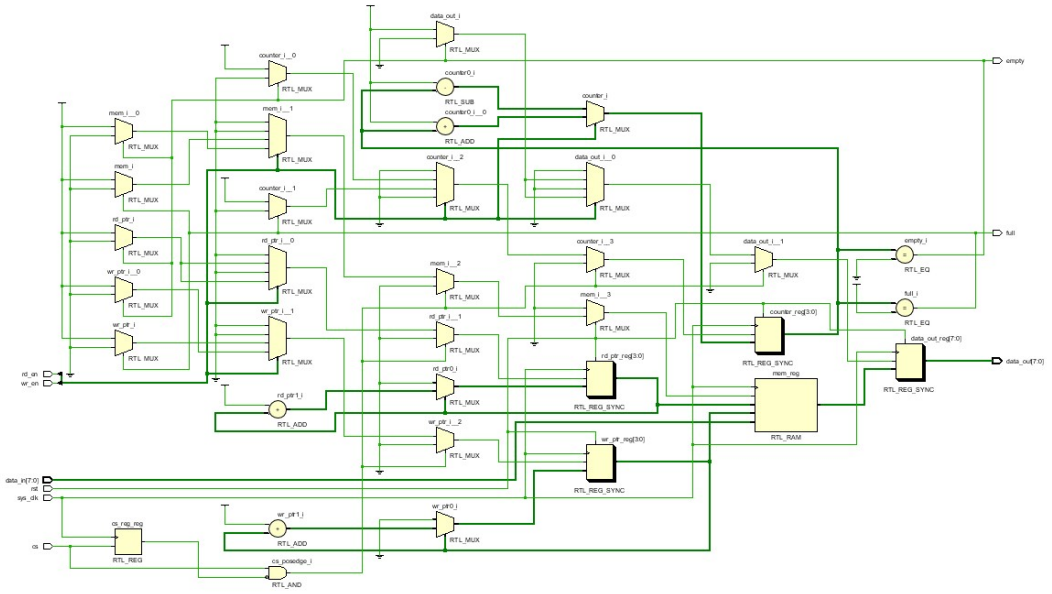
单端口 RAM：



双端口 RAM：



带时钟片选信号的 FIFO 队列：



五、实验过程中遇到的问题及解决情况

问题一：“满”（full）标志信号在 FIFO 还未完全满时就置位，或者“空”（empty）标志信号判断不及时

解决情况：仔细检查指针的递增、比较和复位逻辑，特别是用于区分“满”和“空”的条件（通常需要额外的位来区分指针相等时的两种状态）

问题二：数据在输入输出速率不匹配时，FIFO 溢出或下溢

解决情况：外部控制逻辑必须在 full 信号有效时停止写入，并在 empty 信号有效时停止读取，以防止数据丢失或读取无效数据

问题三：读操作时，数据输出出现毛刺或不确定性

解决情况：

在 RAM 模块的顶层，对数据输出添加一个寄存器，使其在时钟上升沿被同步捕获，从而消除异步读取带来的不稳定因素。

六、实验结果及分析和（或）源程序调试过程

在存储器实验的结果分析中，核心在于验证设计的逻辑功能和时序特性是否符合规范。对于 RAM 模块，成功实现了数据读写操作以及对地址的正确译码。对于异步读模式，数据输出是地址变化后立即响应的；而对于同步读模式，读取到的数据会滞后于地址信号一个时钟周期，这是由寄存器采集地址所导致的固有延迟。对于 FIFO（先入先出存储队列），数据的出队顺序是否严格遵循先进先出原则，这是与 RAM 最大的区别。此外， full 信号应在 FIFO 满时及时置位以阻止溢出，而 empty 信号则需在队列为空时有效，确保不会发生下溢读取到无效数据。最后，对于双端口 RAM，两个端口能独立地执行读写操作，实现数据的并发访问。

七、小组分工情况说明

组员汤英琦：

1. 仿真与验证（Testbench）： 所有模块的功能仿真

2. 硬件下载与报告撰写： 下载验证与文档整理

组员石航恺：

1. Verilog/VHDL 代码设计： 单端口 RAM 模块实现

2. Verilog/VHDL 代码设计： 双端口 RAM 与 FIFO 模块实现