实验七

存储器的设计

实验报告　　日期：2020 年 10 月 30 日

姓名： 朱嘉琦

学号： 191220185

班级： 数电实验一班

邮箱： 1477194584@qq.com

实验七报告——储存器

191220185 朱嘉琦

一、实验目的

本实验的目的是了解FPGA的片上存储器的特性，分析存储器的工作时序和结构，并学习如何设计存储器。

二、实验原理

储存器结构

存储器是一组存储单元，用于在计算机中存储二进制的数据，如图 7 1所 示。存储器的端口包括 + 输入端、输出端和控制端口。输入端口包括：读/写地址端口、数据输入端口等；输出端口一般指的是数据输出端口；控制端口包括时钟端和读/写控制端口。

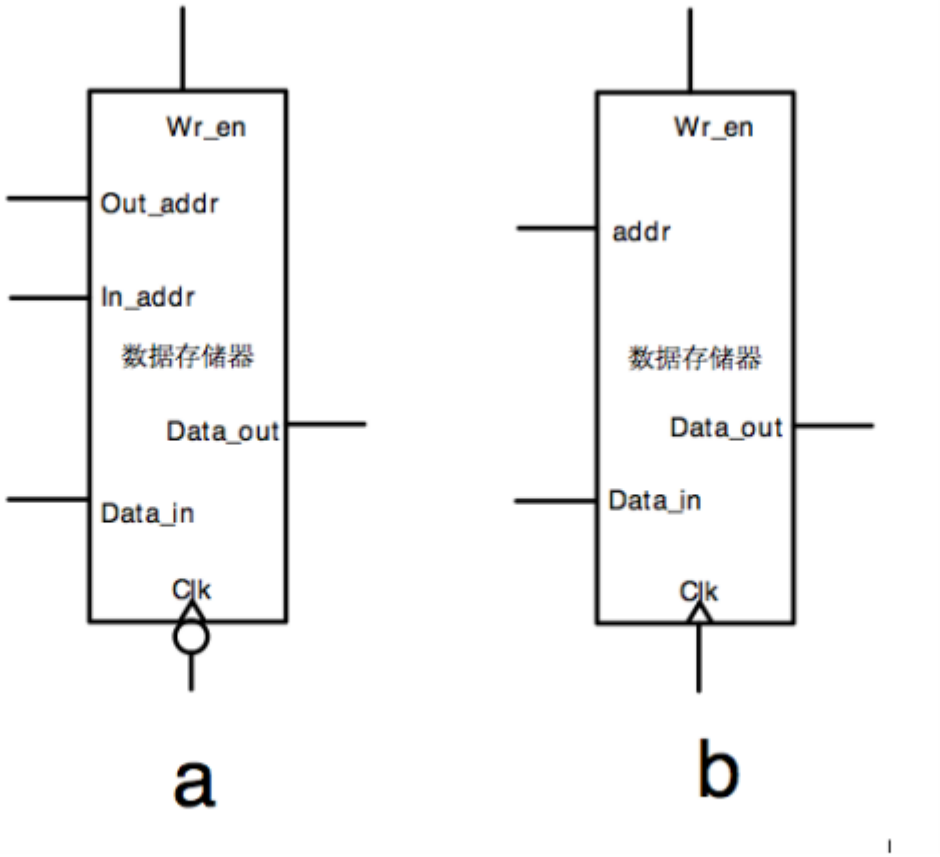

图 7-1: 存储器结构

写数据：

在时钟（clk）有效沿（上升或下降沿），如果写使能（Wr_en， 也可以没有使能端）有效，则读取输入总线（Data_in）上的数据，将其存储到 输入地址线（In_addr）所指的存储单元中。

读数据：

存储器的输出可以受时钟和使能端的控制，也可以不受时钟和使 能端的控制。如果输出受时钟的控制，则在时钟有效沿，将输出地址所指示的 单元中的数据，输出到输出总线上（Data_out）；如果不受时钟的控制，则只 要输出地址有效，就立即将此地址所指的单元中的数据送到输出总线上。

FPGA 存储器的工作模式

表 7-1: 存储器的工作模式

存储器模式	说明
单口存储器	某一时刻，只读或者只写
简单双口存储器模式	简单双口模式支持同时读写（一读一写）
混合宽度的简单双口存储器模式	读写使用不同的数据宽度的简单双口模式
真双口储存器模式	真双口模式支持任何组合的双口操作：两个读口、两个写口和两个不同时钟频率下的一读口一写口
混合宽度的真双口存储器模式	读写使用不同的数据宽度的真双口模式
ROM	工作于 ROM 模式，ROM 中的内容已经初始化
FIFO 缓冲器	可以实现单时钟或双时钟的 FIFO

三、实验环境/器材

实验环境是Quarters 17.1 Lite，实验器材是DE10 开发板

四、程序代码或流程图

1. RAM1

```
module ram1(
    input clk,
    input we,
    input [3:0] inaddr,
    input [3:0] outaddr,
    input [7:0] din,
    output reg [7:0] dout=0
);
    reg[7:0]ram[15:0];
    //reg[7:0]cache;
    initial
    begin
        $readmemh("mem.txt",ram,0,15);

    end

    always@(negedge clk)begin
        if(!we) begin
            dout=ram[outaddr];
        end
        else
            ram[inaddr]<=din;
    end

endmodule
```

2. RAM2

```
module ram2port (
    clock,
    data,
    rdaddress,
    wraddress,
    wren,
    q);

    input    clock;
    input [7:0]  data;
    input [3:0]  rdaddress;
    input [3:0]  wraddress;
    input    wren;
    output   [7:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
    tri1     clock;
    tri0     wren;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

    wire [7:0] sub_wire0;
    wire [7:0] q = sub_wire0[7:0];

    altsyncram  altsyncram_component (
              .address_a (wraddress),
              .address_b (rdaddress),
              .clock0 (clock),
              .data_a (data),
              .wren_a (wren),
              .q_b (sub_wire0),
              .aclr0 (1'b0),
              .aclr1 (1'b0),
              .addressstall_a (1'b0),
              .addressstall_b (1'b0),
              .byteena_a (1'b1),
```

```
            .byteena_b (1'b1),
            .clock1 (1'b1),
            .clocken0 (1'b1),
            .clocken1 (1'b1),
            .clocken2 (1'b1),
            .clocken3 (1'b1),
            .data_b ({8{1'b1}}),
            .eccstatus (),
            .q_a (),
            .rden_a (1'b1),
            .rden_b (1'b1),
            .wren_b (1'b0));
    defparam
        altsyncram_component.address_aclr_b = "NONE",
        altsyncram_component.address_reg_b = "CLOCK0",
        altsyncram_component.clock_enable_input_a = "BYPASS",
        altsyncram_component.clock_enable_input_b = "BYPASS",
        altsyncram_component.clock_enable_output_b = "BYPASS",
        altsyncram_component.init_file = "RAM.mif",
        altsyncram_component.intended_device_family = "Cyclone V",
        altsyncram_component.lpm_type = "altsyncram",
        altsyncram_component.numwords_a = 16,
        altsyncram_component.numwords_b = 16,
        altsyncram_component.operation_mode = "DUAL_PORT",
        altsyncram_component.outdata_aclr_b = "NONE",
        altsyncram_component.outdata_reg_b = "CLOCK0",
        altsyncram_component.power_up_uninitialized = "FALSE",
        altsyncram_component.read_during_write_mode_mixed_ports = "DONT_CARE",
        altsyncram_component.widthad_a = 4,
        altsyncram_component.widthad_b = 4,
        altsyncram_component.width_a = 8,
        altsyncram_component.width_b = 8,
        altsyncram_component.width_byteena_a = 1;

endmodule
```

顶层文件

```
module RAM(

    ////////////// KEY //////////
    input             [3:0]     KEY,

    ////////////// SW //////////
    input             [9:0]     SW,

    ////////////// LED //////////
    output            [9:0]     LEDR,

    ////////////// Seg7 //////////
    output            [6:0]     HEX0,
    output            [6:0]     HEX1,
    output            [6:0]     HEX2,
    output            [6:0]     HEX3
);

wire [7:0] dout1;
wire [7:0] dout2;
ram1(KEY[0],SW[9],SW[3:0],SW[3:0],{6'b000000,SW[5:4]},dout1);
ram2(KEY[0],SW[9],SW[3:0],SW[3:0],{6'b000000,SW[7:6]},dout2);
decoder d0(dout1[3:0],HEX0);
decoder d1(dout1[7:4],HEX1);
decoder d2(dout2[3:0],HEX2);
decoder d3(dout2[7:4],HEX3);
```

五、实验步骤

- 根据实验讲义复习存储器的相关内容。
- 了解如何利用verilog语言设计寄存器

表 7-3: 存储器实例代码

```
module v_rams_8 (clk, we, inaddr,outaddr, din, dout0,dout1,dout2);
input clk;
input we;
input [2:0] inaddr;
input [2:0] outaddr;
input [7:0] din;
output [7:0] dout0,dout1,dout2;

reg [7:0] ram [7:0];
reg [7:0] dout0,dout1;

initial
begin
ram[7] = 8'hf0; ram[6] = 8'h23; ram[5] = 8'h20; ram[4] = 8'h50;
ram[3] = 8'h03; ram[2] = 8'h21; ram[1] = 8'h82; ram[0] = 8'h0D;
end

always @(posedge clk)
begin
    if (we)
        ram[inaddr] <= din;
    else
        dout0 <= ram[outaddr];
end
always @(negedge clk)
begin
    if (!we)
        dout1 <= ram[outaddr];
end
assign  dout2 = ram[outaddr];
endmodule
```

- 根据以上内容设计一个大小为 16*8 的存储器，并采用下面的方式初始化

```
initial
begin
   $readmemh("mem.txt",ram,0,15);

end
```

RAM.v | ram1.v | ram2.v | ram2port.v | mem.txt

```
@0 00
@1 01
@2 02
@3 03
@4 04
@5 05
@6 06
@7 07
@8 08
@9 09
@a 0a
@b 0b
@c 0c
@d 0d
@e 0e
@f 0f
```

- 进行分析与综合，检查是否有语法错误，最终编译通过。
- 利用ModelSim进行功能仿真，编写testbench文件，进行仿真测试 。
- 根据讲义内容利用 IP 核设计一个双口存储器，并利用.mif 文件进行初始化

Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	240	241	242	243	244	245	246	247	.....
8	248	249	250	251	252	253	254	255	.....

- 进行编译，待编译通过后将二进制文件写入开发板 。
- 在开发板上进行硬件验证。

六、测试方法

1. 运行仿真测试，给输入信号赋予不同的值，观察图中输出与理论输出是否一致 。
2. 将sof文件导入开发板中实际测试电路，观察输出是否符合预期。

七、实验结果

两种实现存储器都正常工作，ip核的实现比用txt文件初始化的存储器慢一个周期。（已验收）

八、思考

☞ **思考题**

如果将表7-2中存储器实现部分改为

```
always @(posedge clk)
        if (we)
            ram[inaddr] <= din;
        else
            dout <= ram[outaddr];
```

该存储器的行为是否会发生变化?

存储器的行为会发生改变，因为原代码中dout通过assign语句赋 值，即无论we信号是否有效，dout都会赋值为 ram[outaddr]；但改变后，只有当we信号无效时, dout才会被赋值为 ram[outaddr]，即当outaddr的值发生变化时，可能无法第一时间改变dout的值。

九、实验中遇到的问题及解决方法

在使用 IP 核生成的存储器时不知道端口的具体用法，最终通过实例分析后了解了如何使用和它的具体行为；因为数据输入只写入低2位， 一开始我高6位并未定义，因此进入了高阻态，在实际输出中出现了很多问题，在询问老师后，知道了应该用拼接的方法向存储器中传入数据输入信号

十、意见和建议

无