数字电路与数字系统

实验报告

实　　验: 寄存器组及存储器

姓　　名: 周心同

学　　号: 201220069

目录

1.实验目的

了解 FPGA 的触发器及片上存储器的特性

分析存储器的工作时序和结构

学习如何设计寄存器组和主存

2.实验原理

2.1 寄存器堆

将多个寄存器组合在一起。

2.2 RAM

存储器是一组 asd 存储单元，用于在计算机中存储二进制的数据。

2.3 寄存器和存储器的异同

- 寄存器一般要求存取速度快、并行访问要求高，所以通常寄存器的容量较小。在 CPU 中，PC 及通用寄存器会经常被访问，因此存取的时延要求在一个时钟周期内。对于单周期 CPU，每个时钟周期往往要求同时读取 2 个通用寄存器并完成 1 个寄存器的写回。在要求较高的时候，有可能寄存器组输出的结果需要异步输出，即不在时钟沿上读取，输出随着输入地址实时改变。在这样高的要求下，寄存器组的大小不可能太大，否则会消耗非常多的资源。
- 主存一般容量较大，但是读写时间较长，并且读写过程有严格的时序要求。
- 在 Verilog 中，虽然寄存器组和存储器的描述都是二维数组的方式。但是，编译和综合过程中会根据代码访问的要求来选择具体的实现方式。例如，当代码中没有严格在时钟信号沿上进行读写时，系统会认为该存储单元的读写要求较高，直接采用 FPGA 逻辑单元实现。这种实现方式消耗的资源巨大，一般只能支持数 K 量级的存储单元。如果要求大量的此类存储功能，系统可能会花很长时间进行编译综合，甚至无法实现。如果一个存储单元的访问严格按照时序要求，仅在时钟沿上进行每次单个单元的读写时，系统可以用大容量的 M10K 实现存储，一般可以支持到数百 K 字节的容量。因此，在实验中对存储器的读写应特别关注，避免用高级语言的二维数组的思路来看待存储器，否则会造成很多意想不到的后果。

3.实验环境

- 软件环境

Quartus 17.1 Lite

- 硬件环境

开发板：DE10 Standard

FPGA：Intel Cyclone V SE 5CSXFC6D6F31C6N

4.实验步骤和结果

4.1 寄存器

4.1.1 代码

```
module regfile(
   input  [3:0]  rw, //地址端
   input  [3:0] wrdata, //数据端
   input  regwr,  //使能端
   input  wrclk,  //时钟信号
   output [3:0] outa  //输出端
   );

   //The regfile
   reg [7:0] regs[15:0];

   //add your code here
initial
begin
$readmemh("D:/NJU_ALL/Sophomore/Digital Logic and Principles of Computer Organization/2021/exp5_board/me
end

always @(posedge wrclk)
begin
    if(regwr)//写入
   begin
      regs[rw]=wrdata;
      regs[0]=0;
   end
end

assign outa=regs[rw];

endmodule
```

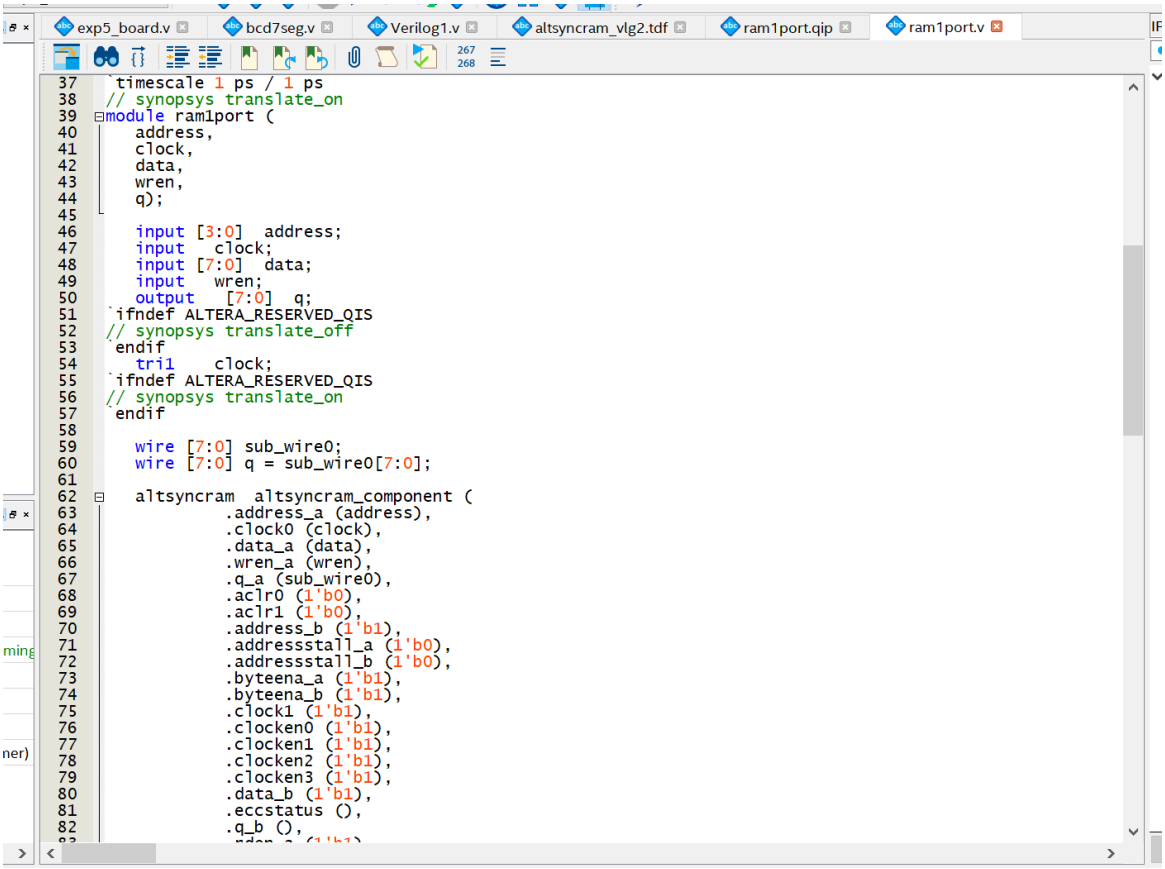
4.1.1 仿真

由于波形图不能直观看结果，所以直接在上板仿真。

4.2 存储器

4.2.1 代码

使用 IP 核自动生成存储器



4.2.2 仿真

同样波形图不便仿真，通过上板仿真。

4.3 接口

```
wire [7:0] a;

regfile re(
    SW [7:4], //地址端
    SW [3:0], //数据端
    SW [8],   //使能端
    KEY[0],   //时钟信号
    a //输出端
    );

bcd7seg bc(a[3:0],HEX0);
bcd7seg bc2(a[7:4],HEX1);

wire [7:0] b;
ram1port ram(
    SW [7:4],
    KEY[0],
    SW [3:0],
    SW [9],
    b);
bcd7seg bc3(b[3:0],HEX2);
bcd7seg bc4(b[7:4],HEX3);
```

5. 实验中遇到的问题及解决办法

问题：readmemh 读取错误

解决办法：把 reg[15:0] regs[7:0]（8 个 16 位） 改成 reg [7:0] regs[15:0]（16 个 8 位）；

6.实验启示和建议

存储器和寄存器的区别：寄存器可以随时读数据（或较短的时延）但是存储器有较大的时延。（具体可以在 RTL 中看到实现电路图不同）。

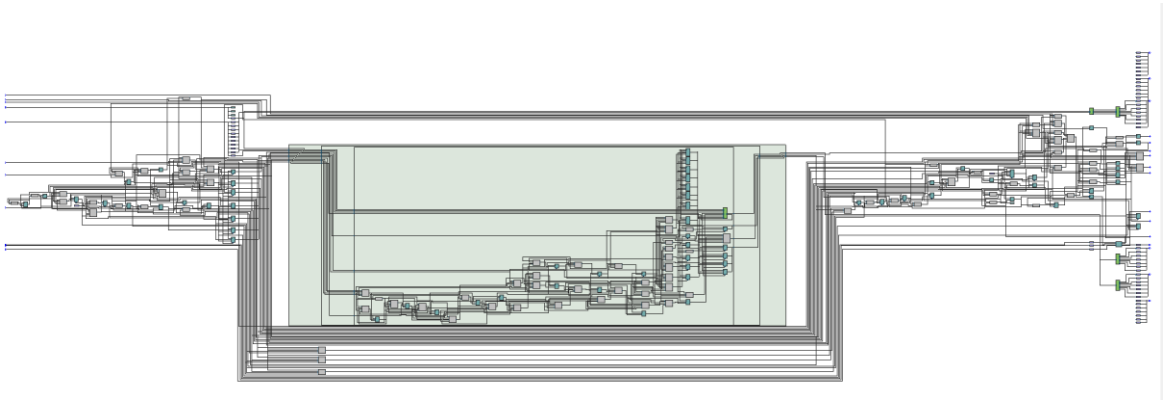
7.思考题

1.请使用 In-System Memory Content Editor 来修改 RAM 中的数据，验证的修改确实更新到开发板上了。已在上板验收中验证。

2.请使用开发板上的按钮来做为存储器的时钟信号。观察两个不同的实现方式下各需要几个时钟周期来完成读取或写入操作？对于读取操作，RAM 需要一个时钟周期，而寄存器不需要时钟周期。

对于写入操作，两者皆需要一个时钟周期。

3.打开 Tools→Netlist Viewers→ Techonlogy Map Viewer，点开实现的树形结构寻找到你生成的两个存储器，观察综合后这两个存储器分别使用了什么方式来实现，为什么？



寄存器用触发器实现，而 RAM 用特殊的实现单元。

4.注意观察综合后输出的资源消耗情况，图 5 19 中两个红框部分消耗的资 源可能是由哪个存储器产生的？如果用寄存器方式，我们用的开发板大约可以 支持多大容量的存储？用 Block Memory 呢？

Logic utilization (in ALMs)	138 / 41,910 (< 1 %)
Total registers	187
Total pins	70 / 499 (14 %)
Total virtual pins	0
Total block memory bits	128 / 5,662,720 (< 1 %)

上方资源消耗应该是寄存器产生的，下方是 RAM 产生的。

如果用寄存器方式 138 对应 16*8bit=16 字节，换算成 41910 资源能用 4859 字节。

而用 RAM 方式 128 对应 16 字节，能用 707840 字节。