# 数电实验EXP7

**刘永鹏 191220070 693901492@qq.com 9.30**

## 1. 实验内容

存储器（Memory）是电子设备中的**记忆器件**，用来存放程序和数据。电子 设备中全部信息，包括输入的原始数据、程序、中间运行结果和最终运行结果 都保存在存储器中。

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。

请在一个工程中完成如下两个存储器。两个存储器的大小均为 16×8，即每 个存储器共有 16 个存储单元，每个存储单元都是 8 位的，均可以进行读写。

1. RAM1：采用下面的方式进行初始化，输出端有输出缓存，输出地址有 效后，等时钟信号的上升沿到来时才输出数据。
2. RAM2：利用 IP 核设计一个双口存储器，利用.mif 文件进行初始化， 十六个单元的初始化值分别为：0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6,0xf7,0xf8, 0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff。

## 2. 实验原理

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。

**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。

1. 在verilog中，使用存储单元数组来实现存储器。

## 3. 实验环境

1. Quartus环境，手动建立项目。
2. 参考实验pdf上的说明，利用**IP Catalog**生成存储器。
3. 使用.mif文件对其初始化。
4. 对于自己写的存储器，使用.txt文件对其初始化。

## 4. 实验代码/实验截图

对于手写的RAM:

模块端口定义（模仿IP Catalog）：

```
module RAM1(
    input clock,
    input [7:0]data,
    input [3:0]rdaddress,
    input [3:0]wraddress,
    input wren,
    output reg [7:0]q
);
//registers
reg [7:0]mem [15:0];
```

初始化：

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

存储器逻辑：

```
always @ (posedge clock)
begin
    if(wren)//写有效
        mem[wraddress] <= data;
    else q <= mem[rdaddress];
end
```

该存储器在时钟上升沿进行读写操作，根据wren判断读还是写。

而对于IP Catalog自动生成的代码，不再贴在这里。

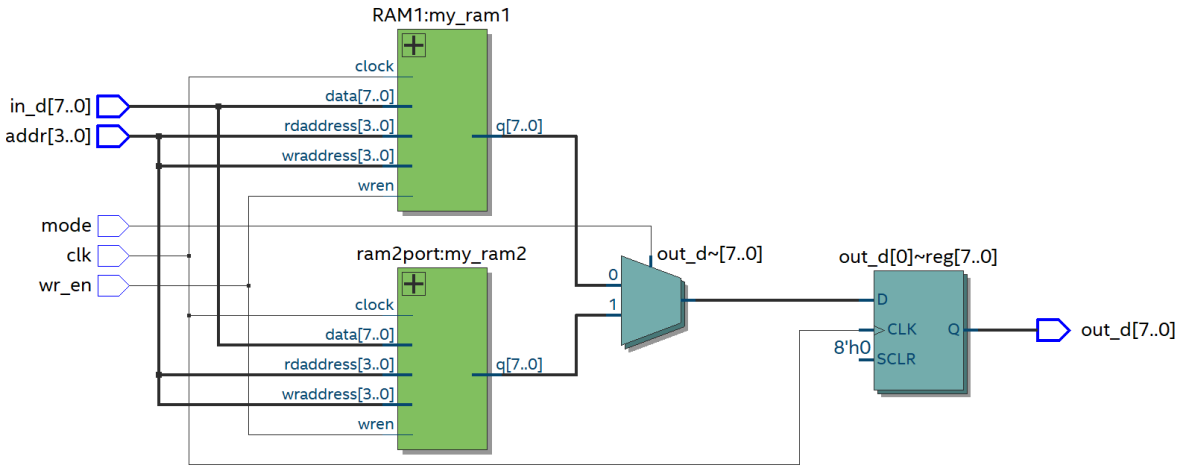
下面是顶层实体的调用：

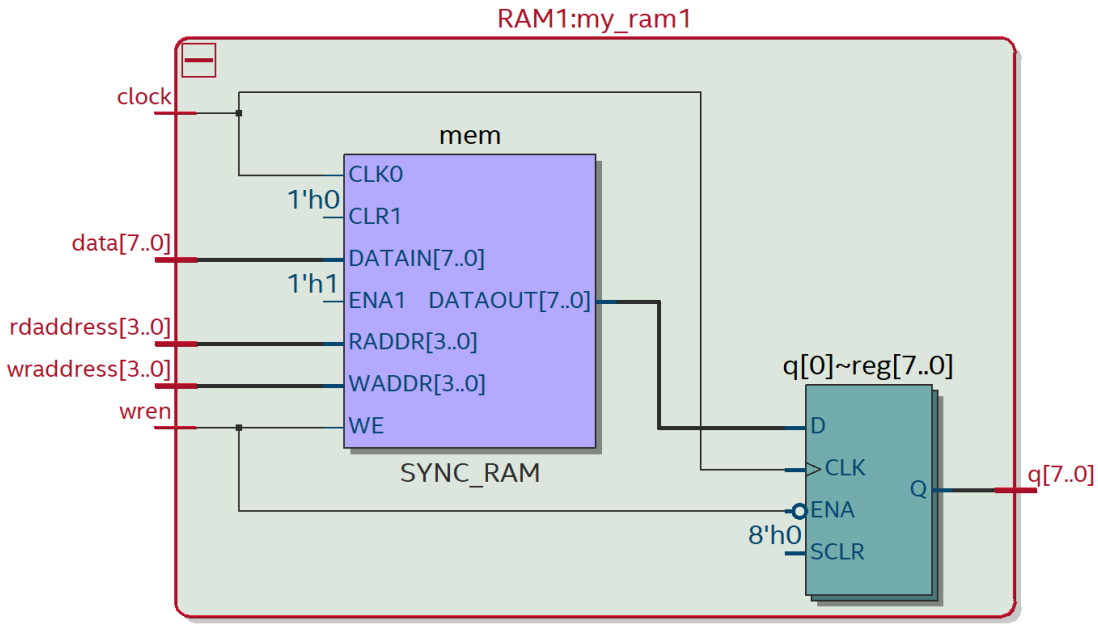
```
module RAM(
    input clk,//clock
    input wr_en,//SW[0]
    input [3:0]addr,//SW[4:1]
    input [7:0]in_d,//[8:5]
    input mode,//0 = RAM1, 1 = RAM2, SW[9]
    output reg[7:0]out_d//LEDR[7:0]

);
//inner signals
wire [7:0]out_d1;
wire [7:0]out_d2;

ram2port my_ram2(
    .clock(clk),
    .data(in_d),
    .rdaddress(addr),
    .wraddress(addr),
    .wren(wr_en),
    .q(out_d2)
);//IP

RAM1 my_ram1(
```

```
    .clock(clk),
    .data(in_d),
    .rdaddress(addr),
    .wraddress(addr),
    .wren(wr_en),
    .q(out_d1)
);//自己写的

always @ (posedge clk)
begin
    if(mode == 1)
        out_d <= out_d2;
    else out_d <= out_d1;
end
endmodule
```

其中，由于开发板输入不够，这里使用SW[8:5]只输入四位数据，**高四位由按钮控制，因此默认为1.**

RTL视图：




## 5. 实验过程

1. 做出初步设计。
2. Quartus编写代码。
3. 没有采用testbench仿真测试（不太需要）。
4. 烧板测试。

## 6. 实验结果

已完成验收。

## 7. 问题与解决方案

1. 没有遇到困难，比较顺利。 最多是在IPcatalog上因为不太熟悉多花了一些时间。