SRAM & UART 实验报告

刘泓尊 2018011446 计 84

一、实验目的

1. 熟悉 ThinPad-Cloud 教学计算机内存储器和串口配置及与总线的连接方式
2. 掌握 SRAM 的访问时序和方法
3. 掌握教学机串口 UART 的访问时序和方法
4. 理解总线数据传输的基本原理

二、实验过程

本次实验我将 SRAM 模块与控制模块耦合在一起作为顶层模块，同时采用了网络学堂提供的 UART 模块。

为了实现读取 10 次的功能，我使用计数器 `counter` 记录读取或写入了几次串口, 用 `address` 维护当前正在访问的内存地址。

为了正确实现功能，应先关闭 ext ram 和字节使能。

SRAM 和控制状态机设计：

`reset`: 读入地址，关闭 SRAM 和 UART 的使能，同时数据总线设为高阻。

`STATE_IDLE`：如果 counter < 10, 进入 `STATE_START_UART_READ`，否则进入 `STATE_START_MEM_READ`。

`STATE_START_UART_READ`：等待 uart_dataready 信号准备好，然后启动读串口状态机（即将 oe_uart_n 拉低），进入 `STATE_FINISH_UART_READ` 等待。

`STATE_FINISH_UART_READ`：等待读串口状态机完成读取（即等待 done 拉高），随后关闭串口读写功能，并将串口数据放到寄存器 data 中，打开总线通路(即关闭总线高阻)，进入状态 `STATE_START_MEM_WRITE`

`STATE_START_MEM_WRITE`：打开 SRAM 片选使能和写使能，SRAM 会在一个周期内把总线数据写入，进入 `STATE_FINISH_MEM_WRITE`。

`STATE_FINISH_MEM_WRITE`：关闭 SRAM 片选使能和写使能，总线设为高阻（为读 SRAM 做准备），同时地址+1，计数器+1。如果 counter >= 10，进入 `STATE_START_MEM_READ`，否则回到 `STATE_IDLE`。

`STATE_START_MEM_READ`: 打开 SRAM 片选使能和读使能，将地址设为初始地址。转 `STATE_FINISH_MEM_READ`。

`STATE_FINISH_MEM_READ`：经过上面一个周期，数据已经放到数据总线上，将总线数据存到寄存器 data。转 `STATE_START_UART_WRITE`。

`STATE_START_UART_WRITE`：打开 UART 写使能，输入线置为 data 寄存

器的值。转 STATE_FINISH_UART_WRITE。

STATE_FINISH_UART_WRITE:等待串口发送完数据(即等待 done 拉高)，计数器+1，当前地址+1，转 STATE_IDLE

UART 状态机设计：

reset：数据总线设为高阻，关闭 uart 读使能和写使能。进入 STATE_IDLE。

STATE_IDLE：如果读使能打开，数据总线设高阻，转 STATE_READ_0；如果写使能打开，将输入数据写入输出寄存器，并打开数据通路，转 STATE_WRITE_0。

STATE_READ_0：等待数据准备好(uart_dataready 拉高)，打开读使能，转 STATE_READ_1

STATE_READ_1：此时数据已经放在总线上，将其写入寄存器 data_out。转 STATE_DONE。

STATE_WRITE_0：uart_wrn 拉低，此时 cpld 开始读取总线数据，转 STATE_WRITE_1。

STATE_WRITE_1：uart_wrn 拉高，cpld 开始发送数据给串口。转 STATE_WRITE_2。

STATE_WRITE_2：等待发送数据进入移位寄存器，之后转 STATE_WRITE_3.

STATE_WRITE_3：等待数据向串口发送完毕，之后转 STATE_DONE

STATE_DONE:等待 UART 模块读使能和写使能关闭，之后数据总线设为高阻，关闭 uart 的读使能和写使能，转 STATE_IDLE.

在实际的实现中，我在拉低 uart_wrn 和拉高 uart_wrn 两个状态之间插入了 2 个节拍，以保证数据稳定写入串口。

三、遇到的困难

1. 从串口读的第一个数据出错，但后续数据正确。

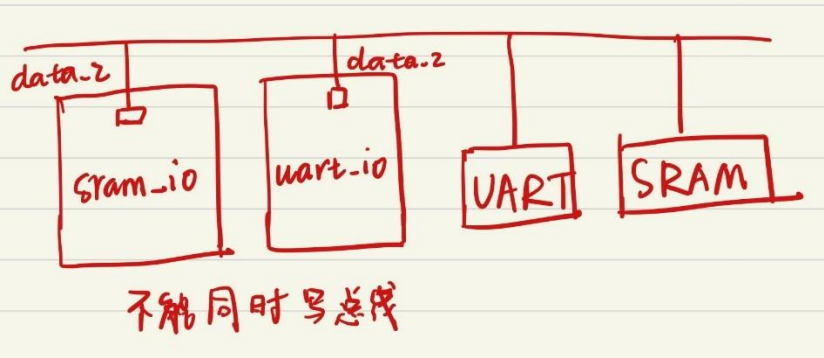
解决方法：在 reset 的时候就将数据总线置为高阻。

如果在读入串口第一个数据的时候 SRAM 对总线的输出不是高阻，那么 UART_io 模块读进来可能就是 data 寄存器的数据，而不是串口的数据，造成数据出错。而因为后续写完内存后总是将 SRAM 总线设为高阻，所以只有第一个数据出错了。

2.“读内存写串口”阶段，从内存中读取的数据与内存实际数据不同。

解决办法：在 uart_io.v 进入 done 状态的时候将总线设为高阻。

这个 bug 困扰了我很长时间，因为我一直认为是我的 SRAM 写错了，没有想到去验证 uart_io.v 的功能。这个问题的原理和上面一个是类似的。在进行了一次读内存写串口之后，如果 uart_io.v 不将对总线的输出设为高阻，那么在读内存阶段总线上就有 SRAM 和 uart_io 两个输出，那么再进行读内存操作可能会读进来 uart_io 的 ram_data 数据，最终会导致读内存得到的所有数据都是相同的。



四、实验数据

SRAM 表格

写内存		读内存		读写一致性
地址	数据	地址	数据	
0x55	33	0x55	33	一致
0x56	65	0x56	65	一致
0x57	123	0x57	123	一致
0x58	165	0x58	165	一致
0x59	232	0x59	232	一致
0x5a	117	0x5a	117	一致
0x5b	21	0x5b	21	一致
0x5c	169	0x5c	169	一致
0x5d	21	0x5d	21	一致
0x5e	208	0x5e	208	一致

串口表格

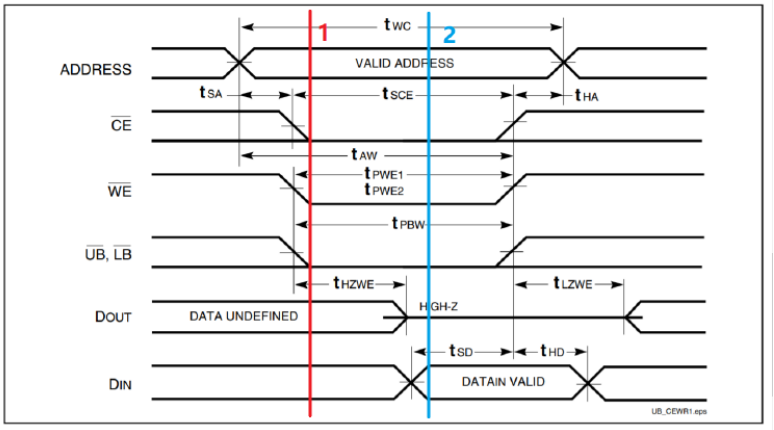
写串口		读串口	
拨码开关数据	收到数据	发送数据	数码管显示数据
10	10	12	12
11	11	13	13
12	12	14	14
13	13	15	15
14	14	16	16

五、思考题

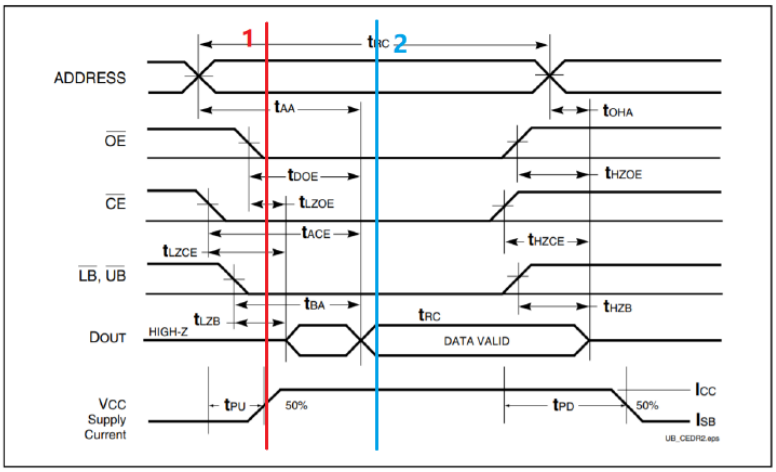
1. 静态存储器的读和写各有什么特点？

静态存储器是不需要进行周期性刷新就可读可写的存储器，在读时需要准备好地址和信号，在写时需要准备好地址、信号和数据，异步读写，没有时钟。它的优势是简化了外部电路，不需要动态刷新。但是静态存储器的存储电路中管子较多，功耗也较大。

静态存储器写之前，在 1 时刻需要准备好数据和地址，还需要控制信号配合（打开片选和写使能，关闭读使能），打开总线输出。一段时间后（很快），在 2 时刻，SRAM 便将数据总线上的数据写到对应的地址，完成了写入，撤掉控制信号即可。



静态存储器读之前，在 1 时刻需要准备好地址，还需要控制信号配合（打开片选和读使能，关闭写使能），总线输出设为高阻。一段时间后（很快），在 2 时刻，SRAM 将数据放到总线上，此时便可以将总线数据写入寄存器。



2. 什么是 RAM 芯片输出的高阻态？它的作用是什么？

高阻态在电路上意味着更高的阻抗。RAM 的高阻态输出通常配合总线使用，如果总线上有多个设备，那么这些设备不能同时写总线，不然就会相互影响，所

以同一时刻只能有一个设备不是高阻。但是输出设为高阻的设备是可以读取总线状态的，所以可以同时读总线。

高阻态保证了多个设备可以使用同一个总线。

3. 本实验完成的是将 Base_RAM 和 Ext_RAM 作为独立的存储器单独进行访问的功能。如果希望将 Base_RAM 和 Ext_RAM 作为一个统一的 64 位数据的存储器进行访问，该如何进行？

BaseRAM 和 ExtRAM 各 32 位，拼在一起是 1M x 64bit，地址宽度 20bit，数据宽度 64bit.

地址：因为地址宽度不变，所以将地址分别对应于 BaseRAM 和 ExtRAM 的地址即可。

数据：将 64 位数据分为高 32 位和低 32 位，分别写入 BaseRAM 和 ExtRAM 对应的地址。

时序：

如果数据总线是 64 位：那么直接将总线高 32 位连到 BaseRAM，低 32 位连到 ExtRAM。写入数据时，将数据、地址准备好，打开两个 RAM 的片选使能和写使能，在下一个周期撤掉控制信号即可。读取数据时，将地址准备好，打开两个 RAM 的片选使能和读使能，在下一个周期读取总线的数据即可。

如果数据总线是 32 位：那么在写入时，需要先将数据高 32 位放到总线上，设置地址和控制信号，在下一个周期写入 BaseRAM；之后再将数据低 32 位放到总线上，设置地址和控制信号，在下一个周期写入 ExtRAM。

在读取时，先将 BaseRAM 的数据和控制信号准备好，在下一个周期将 32 位总线上的地址写入寄存器的高 32 位；之后把 ExtRAM 的数据和控制信号准备好，在下一个周期将 32 位总线上的数据写入寄存器的低 32 位。

4. 请总结教学机上的 UART 和普通的串口芯片 8251 的异同点？

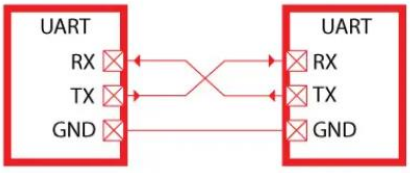
相同点：

支持异步通信，全双工，串行传输，都有缓冲接收器和发送器。都设有起始位、停止位和空闲位，可以进行奇偶校验。

不同点：

8251 属于 USART，支持同步传送，自动插入同步字符或使用硬件同步信号，是高度灵活的串行通信设备。还包含调制/解调控制电路。

UART 是异步收发器，只能进行异步通信。



5. 如果要求将 PC 端发送过来的数据存入到 Base_RAM 的某个单元，然后从该单元中读出，再加 1 送回到 PC 端，则代码需要进行怎样的修改？有兴趣可以试着做一下。

在读内存阶段，寄存器的值设为数据总线值+1 即可，简要的改动如下：

```
//before:
data <= base_ram_data;     // 读取数据到寄存器 data
//after:
data <= base_ram_data + 1; // 读取数据到寄存器 data
```

在实验平台上验证我的结果如下，可以看到发送给串口的数据都增加了 1：

```
## Test round 0 with Addr=0x55
Data Sent:   [33, 65, 123, 165, 232, 117, 21, 169, 21, 208]
Data in RAM: [33, 65, 123, 165, 232, 117, 21, 169, 21, 208]
Received:    [34, 66, 124, 166, 233, 118, 22, 170, 22, 209]
```

六、总结

本次实验我下了很大功夫才完成，主要是因为缺乏硬件思维，对总线数据传递的认识不足。两个看似很小的总线数据传输的问题就困扰了我很久。当然，在克服这些困难之后，我对总线数据传输的基本原理认识深刻了很多，对 SRAM 和串口的机制也有了基本了解，收获很大。

感谢老师和助教以及“不是助教”在微信群中的耐心答疑和悉心指导！