内存串口实验报告

计 71 张程远 2017011429

1 实验过程

本次实验可以分为 4 个过程，即读串口、写内存、读内存以及写串口。每个部分都需要实现，最后再用整体的状态机将所有过程串联起来。

1.1 读串口

按照实验指导所述流程，首先把 ce 和 we 拉高禁用内存；之后拉高 rdn，并令 data 处于高阻态；之后检查 uart_dataready 信号量，如为 1 则读出 data 并拉高 rdn，进入下一个状态，否则回到上一状态。此处设置 readstatus = 0 的状态的意图是指 rst 被按下之前，让整个状态机卡在 0 处不继续进行。

```
case(status)
       4'b0000:begin
           case(readstatus)
           4'b0000:begin
           end
           4'b0001:begin
              ram_ce_n <= 1;
              ram_we_n <= 1;
              readstatus <= readstatus + 1;
           end
           4'b0010:begin
              rdn <= 1;
              highz <= 1;
              readstatus <= readstatus + 1;
           end
           4'b0011:begin
               if(uart_dataready)begin
                   rdn <= 0;
                   readstatus <= readstatus + 1;
               end
               else begin
                   readstatus <= readstatus - 1;
               end
           end
           4'b0100:begin
               data <= base_ram_data[7:0];
               status <= status + 1;
               rdn <= 1;
               readstatus <= 4'b0001;
           end
           endcase
```

1.2 写内存

首先准备之前读到的 data 和地址。addr 变量一直与地址总线绑定，在 rst 按下时已经被读入，因此地址一直是准备好的；准备 data 则先将数据总线解除高阻，然后读入 data。接下来拉低 ce 和 we，等待一周期即可写入。

```
4'b0001:begin
            case(startwrite)
            2'b00:begin
                highz <= 0;
                startwrite <= startwrite + 1;
            end
            2'b01:begin
                base_data <= data;
                addr <= addr + 1;
                startwrite <= startwrite + 1;
            end
            2'b10:begin
                ram_ce_n <= 0;
                ram_we_n <= 0;
                times <= times + 1;
                startwrite <= startwrite + 1;
            end
            2'b11:begin
                if (times == 10) begin
                    status <= status + 1;
                    startwrite <= 0;
                end
                else begin
                    status <= status - 1;
                    startwrite <= 0;
                end
            end
            endcase
        end
```

1.3 读内存

4'b0010 是从读写到写读的一个过渡态，在这个状态里将所有的使能端都变为 1，将地址变为原来的值，给接下来的操作做准备。读内存状态和写内存状态类似，不同的是只需要准备好地址就可以了。

```
4'b0011:begin
            case(startread)
            2'b00:begin
                highz <= 1;
                ram_ce_n <= 0;
                ram_oe_n <= 0;
                startread <= startread + 1;
            end
            2'b01:begin
                data <= base_ram_data[7:0];
                startread <= startread + 1;
            end
            2'b10:begin
                ram_ce_n <= 1;
                ram_oe_n <= 1;
                status <= status + 1;
                startread <= 0;
            end
            endcase
        end
```

1.4 写串口

首先准备好数据，拉低 wrn 后拉高 wrn，之后检验 uart_tsre 和 uart_tbre 的值就可以了。

```
4'b0100:begin
            case(writestatus)
            4'b0000:begin
                highz <= 0;
                writestatus <= writestatus + 1;
            end
            4'b0001:begin
                base_data <= data;
                wrn <= 0;
                writestatus <= writestatus + 1;
            end
            4'b0010:begin
                wrn <= 1;
                writestatus <= writestatus + 1;
            end
            4'b0011:begin
                 if (uart_tbre)begin
                    writestatus <= writestatus + 1;
                end
                else begin
                end
            end
            4'b0100:begin
                if(uart_tsre)begin
                    times <= times + 1;
                    highz <= 1;
                    writestatus <= writestatus + 1;
                end
                else begin
                end
            end
            4'b0101:begin
                if (times == 10) begin
                    status <= status + 1;
                     writestatus <= 0;
                end
                else begin
                    status <= status - 1;
                    addr <= addr + 1;
                     writestatus <= 0;
                end
            end
            endcase
        end
```

2 实验数据

写内存		读内存		读写一致性
地址	数据	地址	数据	
0x88	33	0x88	33	√
0x89	65	0x89	65	√
0x8a	123	0x8a	123	√
0x8b	165	0x8b	165	√
0x8c	232	0x8c	232	√
0x8d	117	0x8d	117	√
0x8e	21	0x8e	21	√
0x8f	169	0x8f	169	√
0x90	21	0x90	21	√

0x91	208	0x91	208	√

3 总结

这次试验用的是 clk_11M 的自动时钟，因此一个周期的时间是定的。之前我在每个周期里都使用了阻塞赋值，导致超时，我意识到阻塞赋值是很消耗时间的，如果一个周期内没有做完应做的赋值操作，那程序运行的结果就不可控制，正如我第一次提交的代码在 60-80 分之间徘徊。在之后含有类似敏感信号的时序逻辑我都会尽量采用非阻塞赋值。

4 思考题

第一部分

（1）读操作：通过数据线控制，首先准备好地址，再将数据线设为高阻态，即可在下个周期获取数据，一共需要 3 个周期：写操作：准备好数据和地址以后，拉低控制写操作的使能信号就可以写入，共需要两个周期。两个操作需要的时间都很少，因此 SRAM 的访问速度很快。

（2）RAM 芯片输出的高阻态是对应位数据线的字节输出，用于表示对应的字节数据不需要。它既不是高电平也不是低电平，对下一级输入毫无影响，可以算作开路。

（3）将两个芯片的控制信号连在一起，并让两个芯片具有相同地址的数据组合成为一个位的数据。

第二部分

（1）不同点：UART 是不可编程的，而普通的串口芯片 8251 是可以编程的；UART 是异步接收/发送器，而 8251 芯片有同步和异步两种工作模式；UART 是半全双工通信，8251 芯片是全双工通信。相同点：UART 与 8251 芯片都以串行方式收发数据，都配有奇偶校验电路，都通过一定的数据信号告知另一方自己的状态等。

（2）选择合适的波特率，读写内存串口的时候选择统一的时钟，还要选择适合的时钟频率（如 50M 就不适合本次实验，一个周期时间太短，而读写内存串口时间又不稳定，导致输出随机）