中国科学技术大学计算机学院

《数字电路实验报告》

实验题目：综合实验

学生姓名：林宸昊

学生学号：PB20000034

完成日期：2021.12.25

计算机实验教学中心制

2020年09月

【综合设计简述】

使用verilog语言编写，利用fpga平台进行交互（包括串口，开关，led等）的小型LC-3.

【设计复杂性】

时间

- 共花费两周，实际有效时长约20h。

代码

- 设计文件约700行，其中串口的接收模块来自老师编写的文件，占约100行代码。
- 仿真文件改编自老师所给文件。

总体架构

- 设计了一个含3个状态的有现状态机以实现目标功能。
- 使用RAM模拟LC-3的内存空间。
- 实现除TRAP指令之外的18条指令。
- 通过串口进行机器码命令的即时执行，或者在通过在RAM中初始化命令来执行所需的程序。
- 通过开关控制任意寄存器的显示，通过前四个数码管显示当前PC指向内存空间储存的值，有四个数码管显示寄存器的值。

【设计新颖性】

值得一提的部分是实现了利用串口进行交互，能够由操作者决定选择执行LC-3内原先预存的命令还是跳过它执行当前输入在串口的命令。

【设计完整性】

状态机简述

- 共具有五个状态

> S0：接收串口命令并输入到缓存区；
>
> S1：将缓存区命令输入指临时寄存器，即内置缓冲；
>
> S2：将内置缓冲中的命令输入IR，即最终用于处理的指令寄存器；
>
> S3：命令已全部输入IR，LC-3对IR进行处理；
>
> S4：一个特殊状态，用于处理从内存中读出数据存入寄存器中。

状态机详述

- S0，接受串口输入
  - 利用老师所给的对输入进行接收的rx模块以及一个IP核fifo接受输入，以输入回车且fifo非空作为跳转条件，跳转至S1。
- S1，二次缓冲
  - 同样利用老师所给代码，以缓冲区已空作为跳转条件，跳转至S2。
- S2，获取命令
  - 由于机器码只有0，1构成，而0，1的ASCII码以二进制表示时恰好最后一位同样为0，1，故将每一个缓冲寄存器内的值得末尾一位以逆序输出倒IR中，构成指令，并跳转至S3。
- S3，处理指令
  - 通过一个较复杂的时序逻辑电路对各信号进行处理，即对应有限状态机得第三状态：输出，在指令处理完毕后直接跳转至S0；
  - 若所接受到得指令涉及对内存的读取，则将RAM的读地址修改后跳转至S4，在S4状态中将内存中的值读入寄存器，以满足RAM改变所带来的延迟性。

指令处理准备部分

- 对于寄存器，在文件内部定义

```
reg signed [15:0] R [7:0]
```

作为寄存器。使其作为有符号数是为了方便对条件码nzp进行更改。

- 对于PC，IR，同样在文件内部定义

```
reg signed [15:0] PC;
reg [15:0] IR;
```

- 对于RAM，做如下定义

```
reg we;//写使能
wire [15:0] dpo, dpra;//读地址与对应输出
reg [15:0] a, d;//写地址与对应输入
```

指令处理实现部分——通过对IR前四位进行分类

- 加法指令——ADD（0001）

```
若为立即数模式：
R[IR[11:9]] <= R[IR[8:6]] + {{12{IR[4]}},IR[3:0]};
若为寄存器模式：
R[IR[11:9]] <= R[IR[8:6]] + R[IR[2:0]];
更新条件码（条件码的更新大致相同，此后不再多加赘述）：
if(R[IR[11:9]] > 16'h0000)
    {n,z,p} <= {1'b0, 1'b0, 1'b1};
else if (R[IR[11:9]] == 16'h0000)
    {n,z,p} <= {1'b0, 1'b1, 1'b0};
else
    {n,z,p} <= {1'b1, 1'b0, 1'b0};
PC自增：
PC <= PC + 1;
```

- 与指令——AND（0101），取非指令——NOT（1001）均与加法指令类似
- 条件跳转指令——BR（0000）

```
if({n & IR[11], z & IR[10], p & IR[9]})
    begin
        PC <= PC + 1 + {{7{IR[8]}},IR[8:0]};
    end
else begin
    PC <= PC + 1;
    end
```

- 寄存器跳转指令——JMP（1100）

```
PC <= R[IR[8:6]];
```

- 直接跳转指令——JSR（0100）

```
首先储存增量PC值：
R[7] <= PC + 1;
若为立即数模式：
PC <= PC + {{6{IR[10]}},IR[9:0]} + 1;
若为寄存器模式：
PC <= R[IR[8:6]];
```

- 读指令——LD，LDI，LDR（0010，1010，0110）

  三者思路大致相同，此处仅摘选LD指令做说明

```
dpra <= PC + {{7{IR[8]}},IR[8:0]} + 1;//修改读地址
PC <= PC + 1;//PC自增
curr_state <= S4;//直接跳转至下一状态
```

  而在S4状态下再进行寄存器的修改，否则将出现寄存器储存值为读地址更改前RAM的输出的情况。

```
if(curr_state == S4)
    R[IR[11:9]] <= dpo;
else
    R[IR[11:9]] <= R[IR[11:9]];
```

- 地址读取指令——LEA（1110）

```
R[IR[11:9]] <= PC+{{7{IR[8]}},IR[8:0]};
PC <= PC + 1;
```

- 写指令——ST，STI，STR（0111，1011，0111）

```
we <= 1;
a <= R[IR[8:6]] + {{10{IR[5]}},IR[5:0]};
d <= R[IR[11:9]];
PC <= PC + 1;
```

- 最后我们剩下一个未使用指令（1101）。在通过串口的交互中，使用者如果输入1101，则相当于命令LC-3按序执行之前预存的指令，否则可以跳过这条指令，命令LC-3执行即时输入的指令。

```
IR <= dpo;
curr_state <= S3;
```

  即将当前RAM预存的指令赋值给IR，然后强制将S3的持续时间延长一个时钟周期进行下一次处理。

显示部分

- 后四个数码管用于显示寄存器的值，通过开关控制——哪个开关打开则显示对应寄存器的内容，若均未打开则显示当前PC的值。
- 前四个数码管用于显示当前PC所指向位置的值

```
hex_seg_buff[31:16] <= dpo;
```

  通过这种显示方式，可以大致了解LC-3当前的内部情况。

【总结与思考】

- 首先对于串口的输入转化为内部IR中的指令即为一大难点，我另一个做LC-3的同学最终还是没能够实现串口的交互，只能通过预存指令执行。
- 在初版中对于所有的信号输出我都使用的是组合逻辑，在仿真时能输出正确的波形，但是烧到板子上就会出现现实问题，在老师的建议下改为时序逻辑即能够正常显示，原因如下：

  > 由于所有的组合逻辑都集中在一个状态下，而处理IR的状态只有一个时钟周期的时间，大量电路元件所造成的延时很容易导致在实际运行中部分指令未能成功执行或者被反复执行，最终导致无法预料的结果。解决方法可以将组合逻辑电路改为时序逻辑电路，并且最好改为时序逻辑后将处理指令的状态延长数个时钟周期，以保证每一个指令均能被正确的执行。

- 对于内存的读取需要一个额外状态也是一种无奈的尝试。RAM的读地址端口由于显示的缘故需要在除了S3的状态下与PC同步，这就导致每一次的更改只会持续一个时钟周期，寄存器的值还未来得及更改，读地址就恢复为原PC，所以增加一个状态用于特殊处理这种情况。
- 最后仍存在的缺陷是，所有逻辑电路仍处于同一状态下。其实根据IR的类型可以额外得到很多状态但由于实在没有精力再进行优化重写所以只能作罢。如上所述，这种处理会使得延迟变得很大，而且会给电路的运行带来很大的不确定性。
- 汇编代码其实会更简洁易懂，并且若能实现串口的回显将使可操作性大大增强，这些都算是可优化的点吧。

最后——

终于解放了aaaaaaaaaaaa