计算机组成原理-实验二报告

于新雨 计25 2022010841

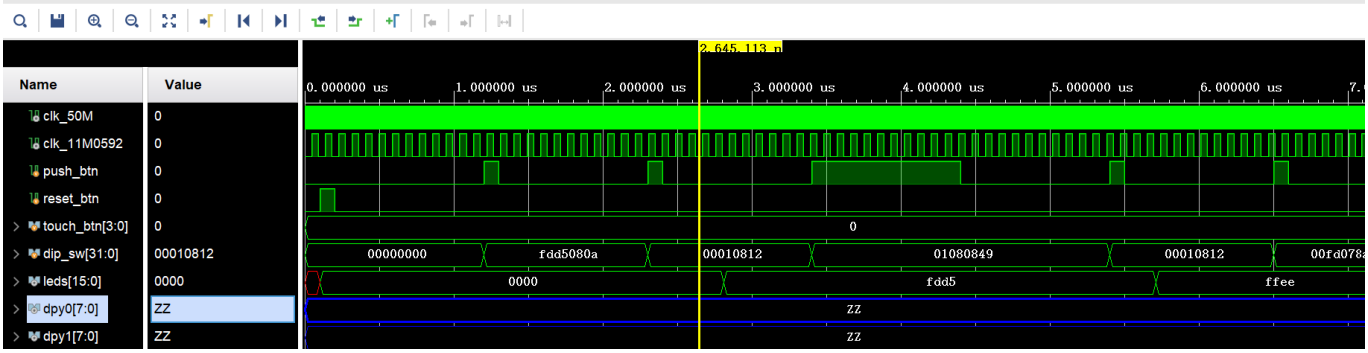
仿真

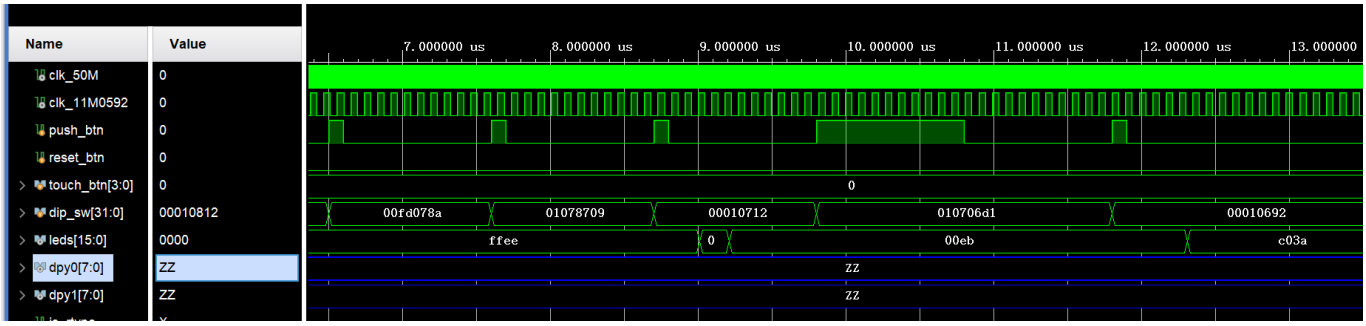
仿真主题部分波形的产生代码如下：

```
        dip_sw=`inst_poke(5'd16, 16'hfdd5);
        push_btn = 1;
        #100;
        push_btn = 0;
        #1000;

        dip_sw = `inst_peek(5'd16,one);
        push_btn=1;
        #100;
        push_btn=0;
        #1000;
        // 算数右移
        dip_sw=`inst_rtype(5'd16, 5'd16, 5'd16, 4'd9);
        push_btn = 1;
        #1000;
        push_btn = 0;
        #1000;

        dip_sw=`inst_peek(5'd16, one);
        push_btn=1;
        #100;
        push_btn=0;
        #1000;
        // 求和
        dip_sw= `inst_poke(5'd15, 16'hfd);
        push_btn = 1;
        #100;
        push_btn = 0;
        #1000;

        dip_sw = `inst_rtype(5'd14, 5'd15, 5'd16, 4'd1);

        push_btn = 1;
        #100;
        push_btn = 0;
        #1000;

        dip_sw = `inst_peek(5'd14,one);
        push_btn=1;
        #100;
        push_btn=0;
        #1000;
```

```
    dip_sw=`inst_rtype(5'd13, 5'd14, 5'd16, 4'd10);  // ROL 检查是否实现正确
    push_btn = 1;
    #1000;
    push_btn = 0;
    #1000;

    dip_sw=`inst_peek(5'd13,one);
    push_btn=1;
    #100;
    push_btn=0;
    #1000;
    #100000 $finish;
  end
```

- 第一步：先 poke 再 peek 检查是否正确写入并且输出
- 第二步：测试算数右移指令，此时有两个注意点：
  - 将 push_btn 设置为多个周期的1，且设置 rd 和 rs 相同 验证文档中所说：按照按下 push_btn 的次数来进行状态控制。同时检测 trigger 的无误性
  - 算数右移指令是带符号的运算
- 第三步：测试加法指令，是否能够正常运算，特别设计了加法溢出的 case，判断可以得到预期结果
- 第四步：测试 ROL 指令，其中 循环左移的位数是按照运算数 rs2 模 32 来做的
- 在每一步之后都用 peek 检验是否得到了正确的结果





实验

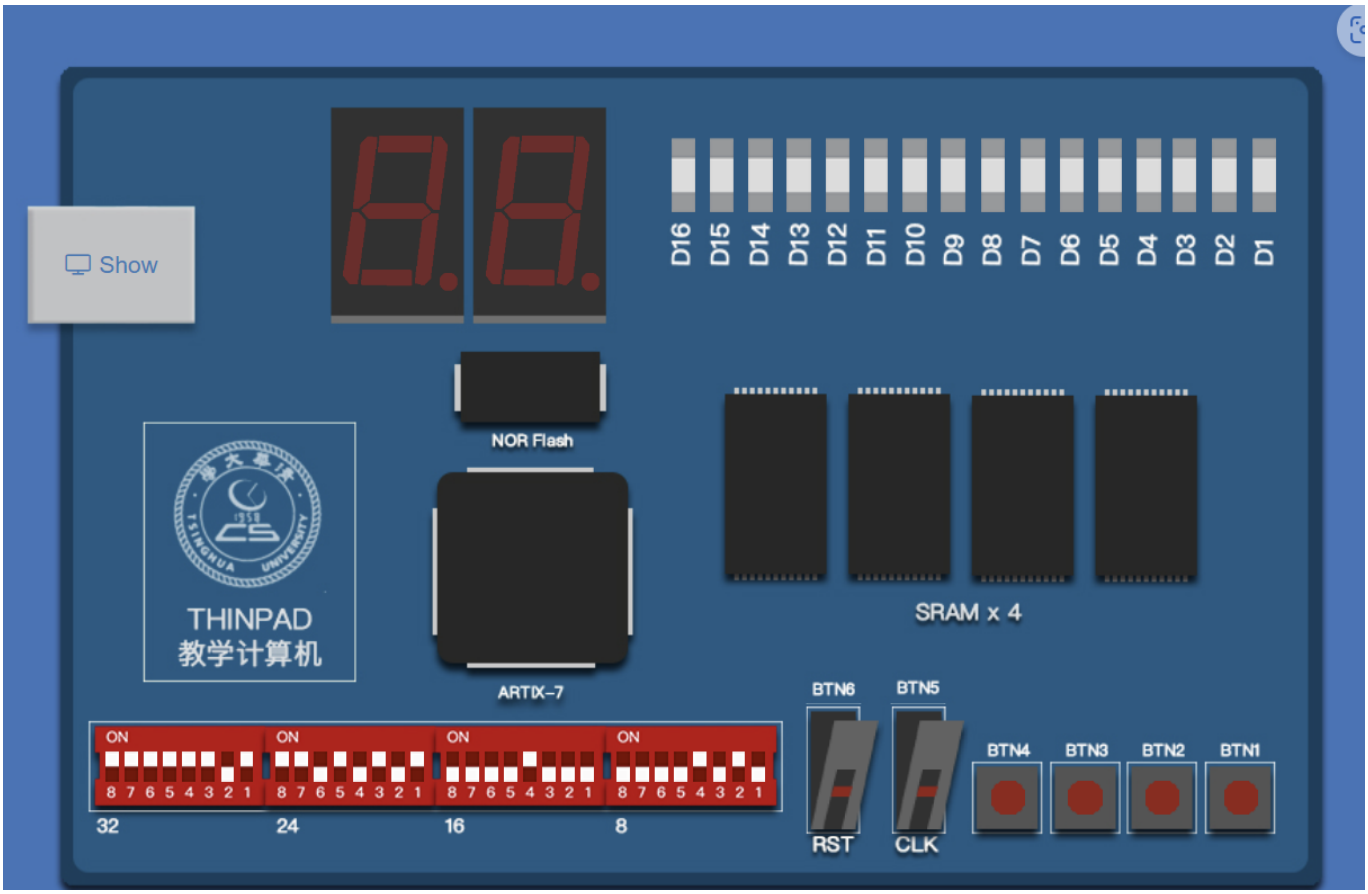
验证如下命令

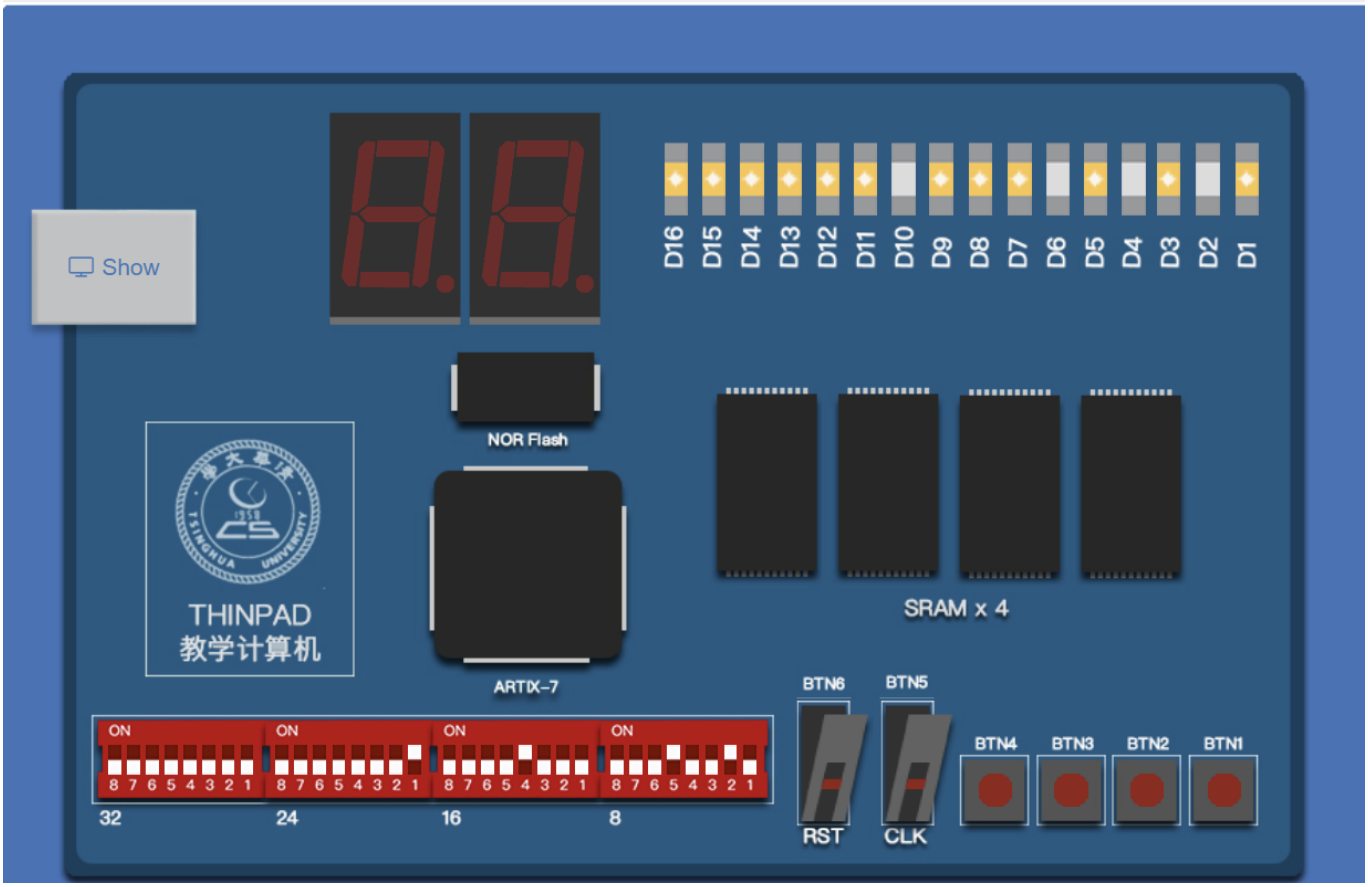
x16 = 0xfdd5 x16 = x16 SRA x16, inst 0x01080849 expect x16 = 0xffee

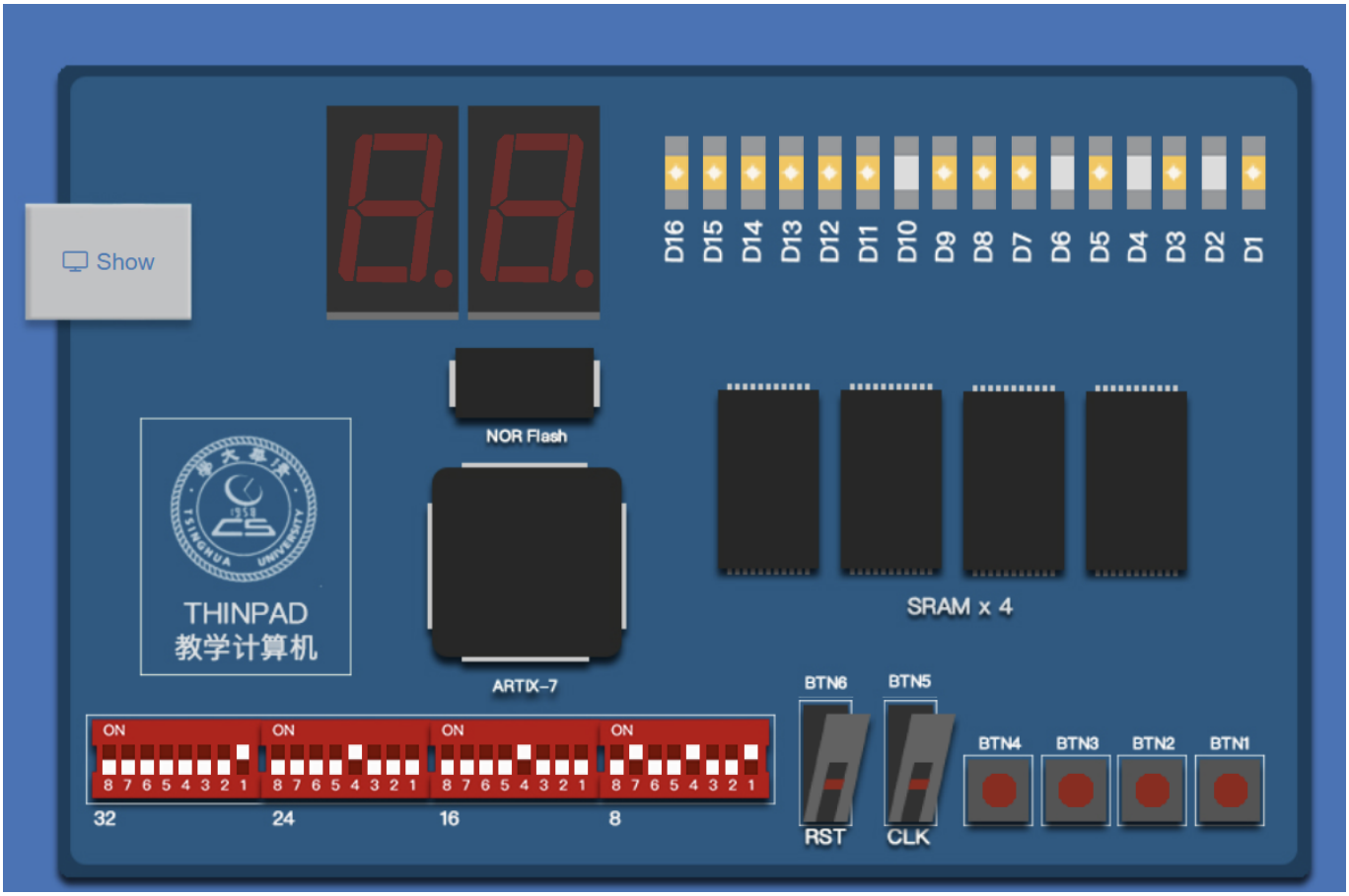
输入指令分别为：

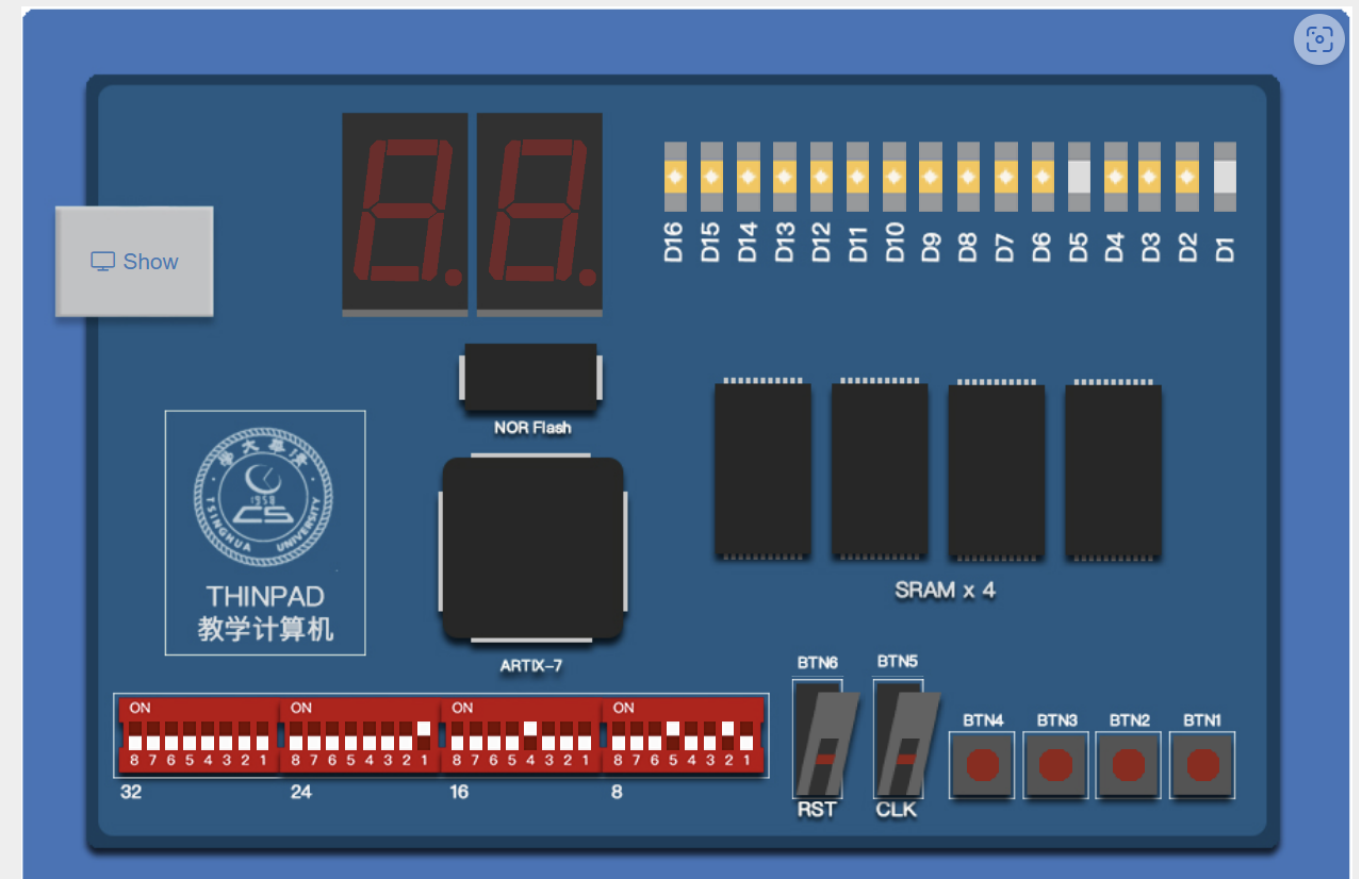
```
    `inst_poke(5'd16, 16'hfdd5);   // 0xfdd5080a
    `inst_peek(5'd16, 16'h1);      // 0x00010812
    `inst_rtype(5'd16, 5'd16, 5'd16, 4'd9);  // 0x01080849
    `inst_peek(5'd16, 16'h1);      // 0x00010812
```

上版实验结果




Show
D16 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1
NOR Flash
SRAM x 4
THINPAD
教学计算机
ARTIX-7
BTN6 BTN5
BTN4 BTN3 BTN2 BTN1
ON
8 7 6 5 4 3 2 1
32 24 16 8
RST CLK


Show
D16 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1
NOR Flash
SRAM x 4
THINPAD
教学计算机
ARTIX-7
BTN6 BTN5
BTN4 BTN3 BTN2 BTN1
ON
8 7 6 5 4 3 2 1
32 24 16 8
RST CLK



思考题

> 假设需要你的 ALU 支持乘法和除法运算，而这些运算均不能在一个周期内完成，应该如何改动你的状态机？给出新的状态转移图，并做简要说明

- ALU 的乘法的常见算法是布斯算法，将一个乘数转为2的幂次乘以相邻两项之差的形式，将乘法转为移位和加法减法的组合。而除法则是采用加减交替的形式来实现。
- 在实现 ALU 时，我们易于知道何时算法终止，并且较易实现在算法结束将要返回结果的同时返回一个信号 ack，从而我们把 ALU 改为 ALU_with_ack 的版本
- 当接收到 ack 的时候，controller 才进行下一步状态转移