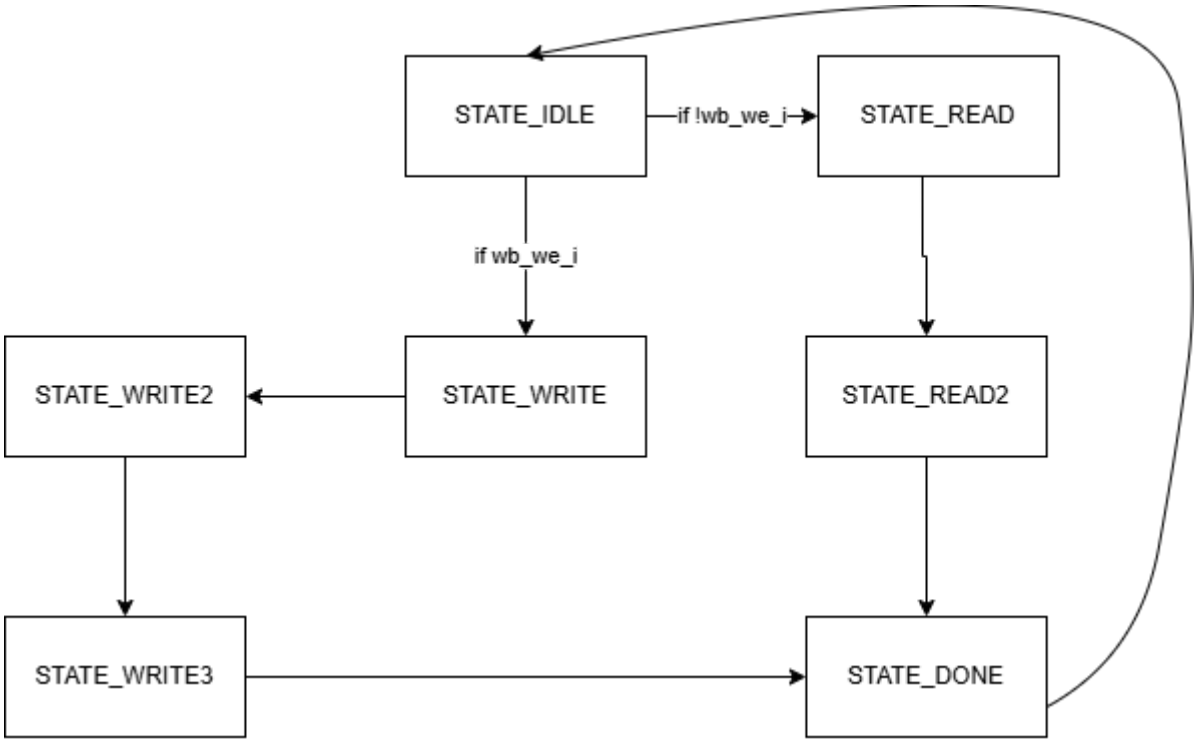
计算机组成原理-实验三、四报告

于新雨 计25 2022010841

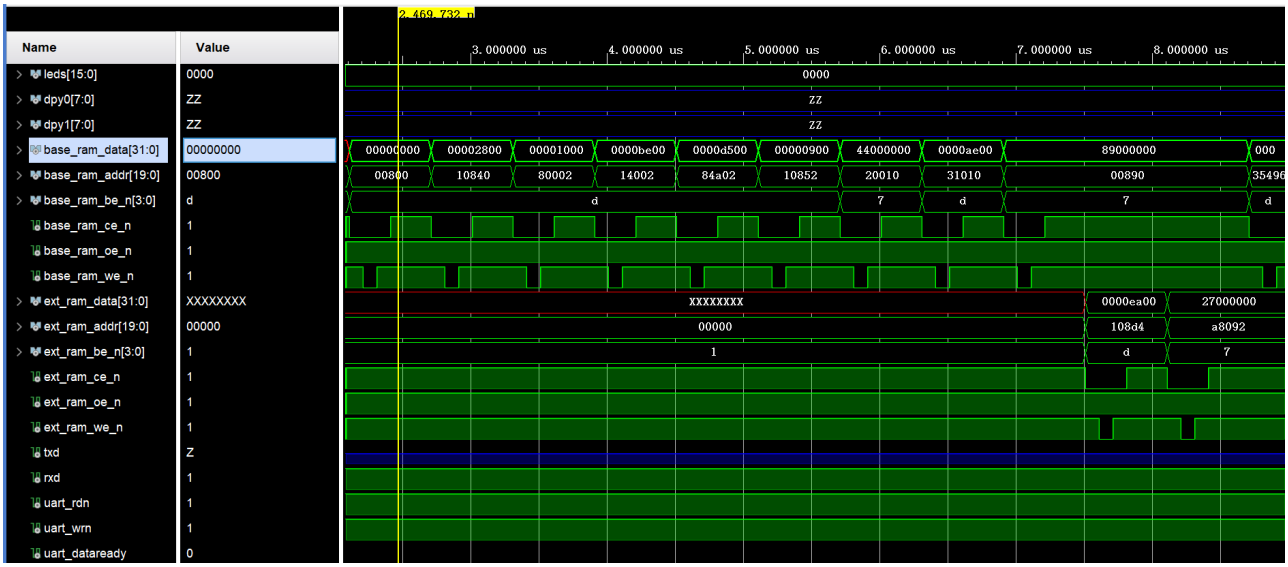
实验3

1. SRAM 状态机设计



- **STATE_IDLE**: 此时为空闲状态，如果写使能为高，则赋值 sram_addr,sram_data_o_comb,sram_oe_n 等信号，进入 STATE_WRITE 状态。否则赋值 sram_addr,sram_oe_n,sram_ce_n 等信号，进入 STATE_READ 状态。
- **STATE_WRITE**: 此时为写状态的第一个时钟周期，我们拉低 sram_we_n 信号，进入 STATE_WRITE2 状态。
- **STATE_WRITE2**: 此时为写状态的第二个时钟周期，我们拉高 sram_we_n 信号，进入 STATE_WRITE3 状态。
- **STATE_WRITE3**: 此时为写状态的第三个时钟周期，我们拉高 sram_ce_n 信号，赋值 wb_ack_o（发出 ack 信号），进入 STATE_DONE 状态。
- **STATE_READ**: 此时为读状态，我们要等待一个周期才能取得有效数据，所以该状态直接进入 STATE_READ2 状态。
- **STATE_READ2**: 此时为读状态的第二个时钟周期，我们拉高 sram_ce_n 信号，赋值 wb_dat_o，发出 ack 信号，进入 STATE_DONE 状态。
- **STATE_DONE**: 此时为完成状态，我们拉低 ack 信号，等待下一个周期进入 STATE_IDLE 状态。

2. 仿真波形



可以看出有正确的 ram_addr, ram_data, ce, oe 等波形

3. 云平台仿真

记录数量 20 ▶开始

#	读/写	地址(byte)	数据	BEn[3:0]	波形
0	W	2000	00000000	1101	
1	W	42100	00002800	1101	
2	W	200008	00001000	1101	
3	W	50008	0000be00	1101	
4	W	212808	0000d500	1101	
5	W	42148	00000900	1101	
6	W	80040	44000000	0111	
7	W	c4040	0000ae00	1101	
8	W	2240	89000000	0111	
9	W	d5258	0000a400	1101	
10	W	2138c8	83000000	0111	
11	W	14310c	0000dc00	1101	

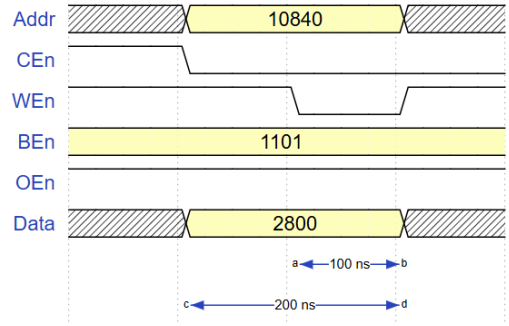
访存序列

本功能可以对BaseRAM访问进行采样。地址数据均以16进制显示。

记录数量 20 ▶开始

#	读/写	地址(byte)	数据	BEn[3:0]	波形
0	W	2000	00000000	1101	
1	W	42100	00002800	1101	
2	W	200008	00001000	1101	
3	W	50008	0000be00	1101	
4	W	212808	0000d500	1101	
5	W	42148	00000900	1101	
6	W	80040	44000000	0111	
7	W	c4040	0000ae00	1101	
8	W	2240	89000000	0111	
9	W	d5258	0000a400	1101	
10	W	2138c8	83000000	0111	
11	W	14310c	0000dc00	1101	

访存波形



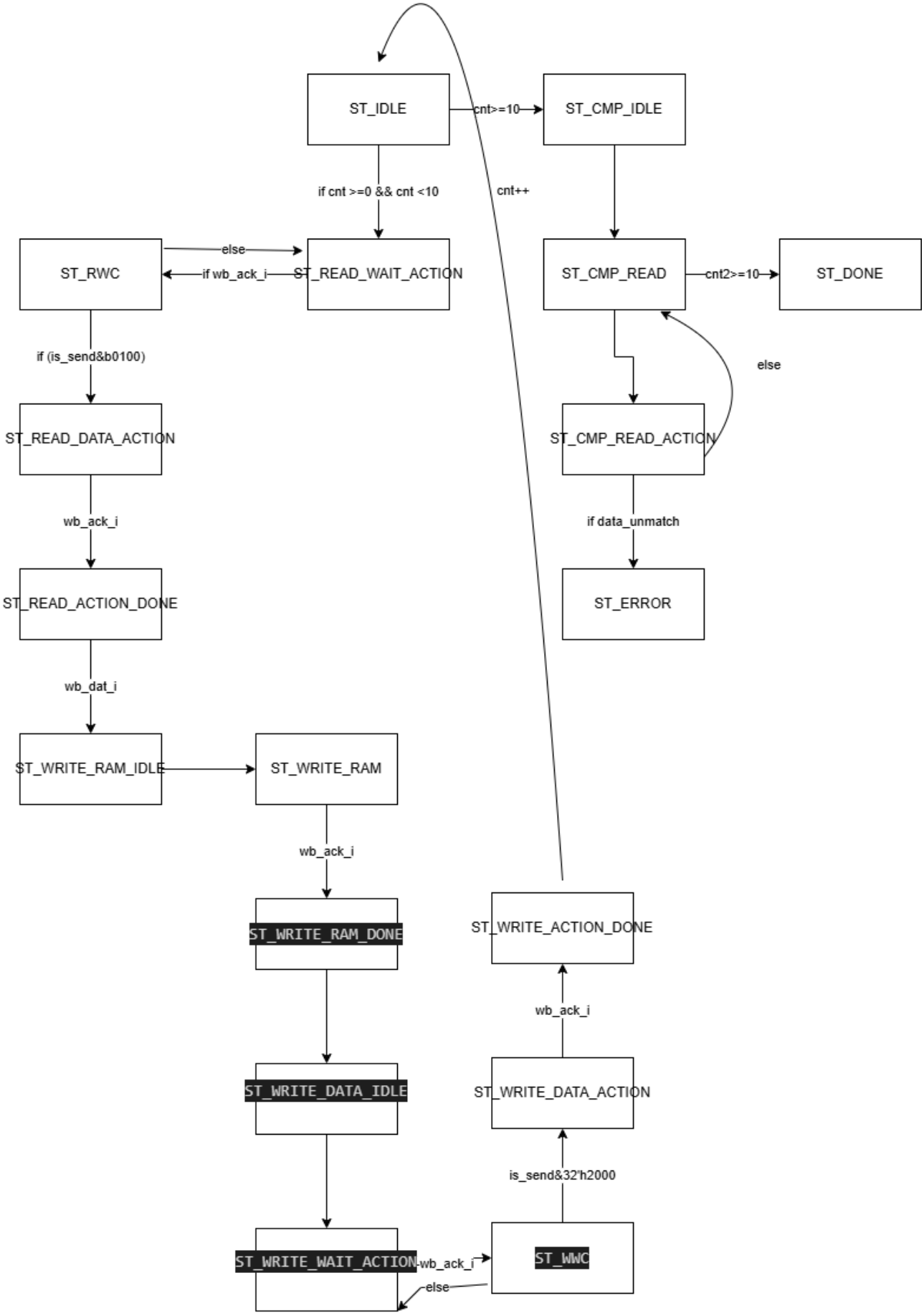
可以看到地址和数据和本地仿真信号相符，且图片2中 ce, we, be 等波形和仿真的波形相符

4. 思考题

1. 静态存储器的读和写各有什么特点？
    1. 读操作：静态存储器的读操作是非破坏性的，即读取数据不会改变存储器中的数据。并且读操作很快，只用两个周期左右即可返回，且协议较为简单
    2. 写操作：静态存储器的写操作是破坏性的，即写入数据会改变存储器中的数据。写操作需要三个周期，且需要在第一个周期设置写使能，第二个周期拉低写使能，第三个周期拉高写使能，且写操作较为简单，和读操作的控制信号也有明显区别
2. 什么是 RAM 芯片输出的高阻态？它的作用是什么？
    1. 高阻态是输出端与电路断开，呈现出一种高阻抗的状态。
    2. 作用：
        1. 因为多个设备（如 SRAM, 外设）可能共用一条数据总线。当一个设备向总线发送数据时，其他设备必须将它们的输出端置于高阻态，以免多个设备同时向总线上发送信号，导致信号冲突（总线争用）。所以高阻态可以避免总线争用，防止干扰总线上其他设备的信号输出和信号读取。
        2. 如果两个设备同时尝试在总线上输出相反的电平信号（一个输出高电平，另一个输出低电平），可能会产生冲突，导致设备损坏。通过使用高阻态，可以保护电路
3. 本实验完成的是将 BaseRAM 和 ExtRAM 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？
    1. 有两个思路可以进行，首先可以对本实验中的 sram_controller, wb_mux_2 等模块的数据位数进行修改，使得可以支持64位的数据，并且对 ram_be_n 等信号也进行相应修改
    2. 另一种思路是将 BASERAM 作为高 32 位，ExtRAM 作为低 32 位，对于 sram_controller, 设置 DATA_WIDTH 的参数为64，然后在读时将两个存储器的数据合并，在写时将高32位，低32位分别写入 BASERAM ExtRAM，这样就可以实现 64 位数据的存储器访问，该思路相对上一种的改动较少

实验4

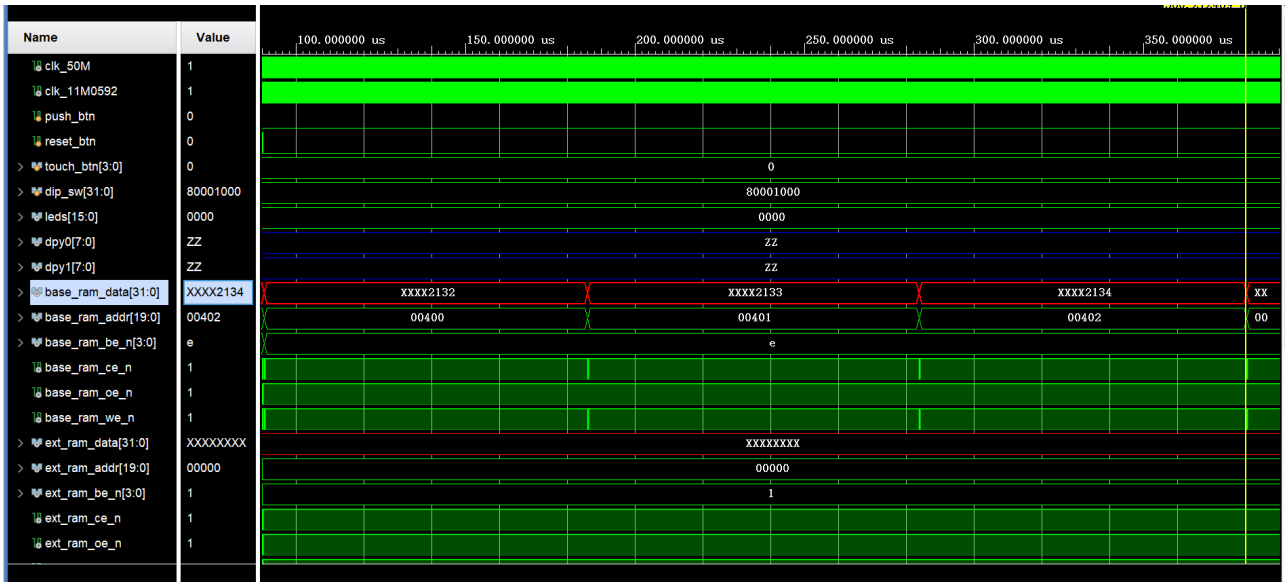
1. 给出你的状态机设计，并简要解释每个状态的功能



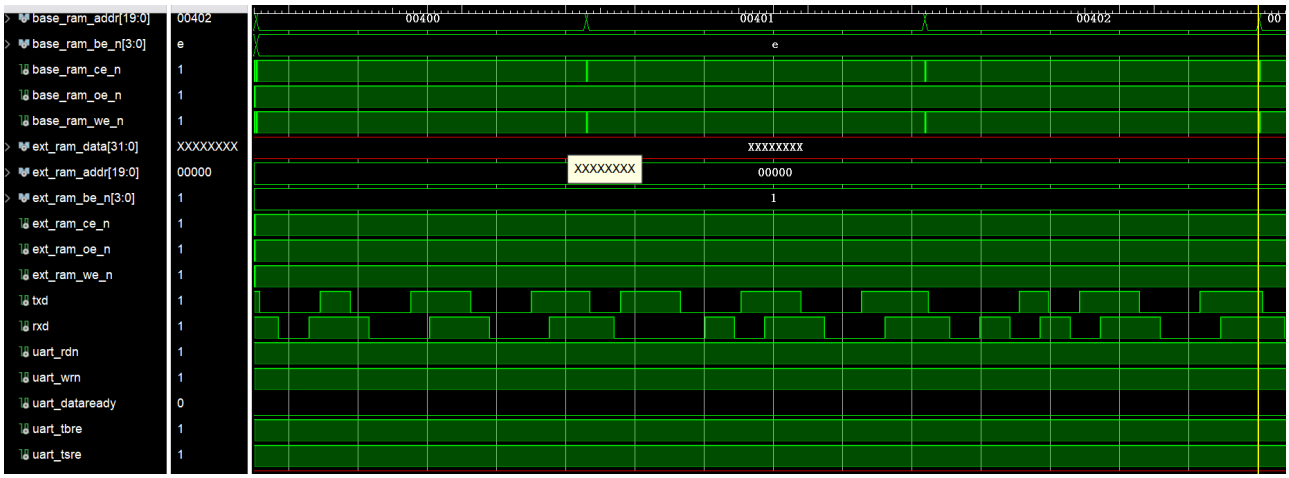
- 总体分为两个大的阶段：第一阶段：从串口读取数据并且写内存和写回串口；第二阶段：比较内存中的数和原值是否相同

- **ST_IDLE**: 判断是在第一阶段还是第二阶段，如果是第一阶段则准备读取串口数据，如拉高 wb_cyc_o wb_stb_o，设置 wb_adr_o，进入 ST_READ_WAIT_ACTION 阶段，如果是第二阶段则进入 ST_CMP_IDLE 状态
- **ST_READ_WAIT_ACTION**: 等待串口数据读取，如果 wb_ack_i 为高则读取数据，进入 ST_READ_ACTION 状态
- **ST_RWC**: read-write-check 判断是否可以读取数据，如果可以则读取数据，进入 ST_READ_DATA_ACTION 状态，否则回到 ST_READ_WAIT_ACTION 状态
- **ST_READ_DATA_ACTION**: 在收到 ack 时读取数据，进入 ST_READ_ACTION_DONE 状态
- **ST_READ_ACTION_DONE**: 进入写内存状态
- **ST_WRITE_RAM_IDLE**: 设置 wb_cyc_o,wb_stb_o,wb_adr_o 等数据，准备写入
- **ST_WRITE_RAM**: 接收到 ack 时拉低 wb_cyc_o wb_stb_o 等信号，进入 ST_WRITE_RAM_DONE 状态
- **ST_WRITE_RAM_DONE**: ram_addr 加4，进入写回串口状态
- **ST_WRITE_DATA_IDLE**: 拉高 wb_cyc_o wb_stb_o 等信号，准备读取 32'h10000005 数据
- **ST_WRITE_WAIT_ACTION**： 读取串口标志位
- **ST_WWC**： 判断是否可以发送，如果可以则设置串口地址发送数据，否则回到 ST_WRITE_WAIT_ACTION 状态
- **ST_WRITE_DATA_ACTION**: 收到 wb_ack_i 后拉低信号
- **ST_WRITE_ACTION_DONE**: cnt 加 1，回到 ST_IDLE 状态
- **ST_CMP_IDLE**： 准备比较数据
- **ST_CMP_READ**： 设置标志位，准备从内存中读取数据
- **ST_CMP_READ_ACTION**： 拿读到的数据和原本串口发来的数据比较，判断是否一致，如果不一致则到 ST_ERROR 状态
- **ST_ERROR**： 标志该状态有错误
- **ST_DONE**: 过程结束

2. 仿真结果



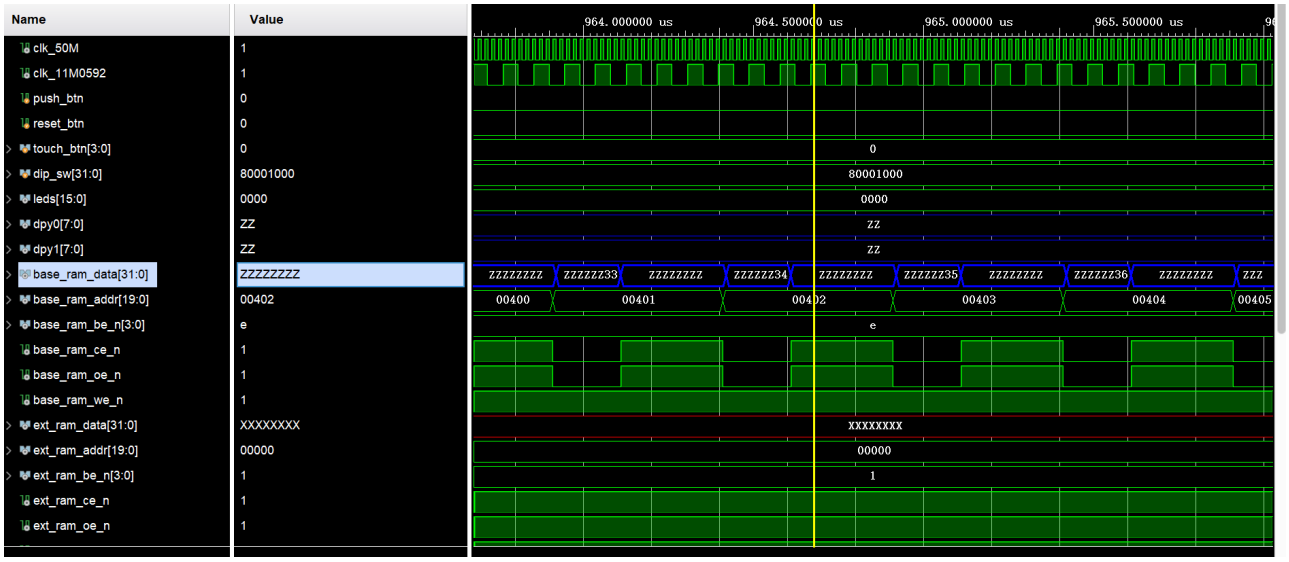
如图可见，base_ram_data, base_ram_addr, ce_n 等值符合预期



如图可见，串口的 txd rxd 也有符合预期的变化

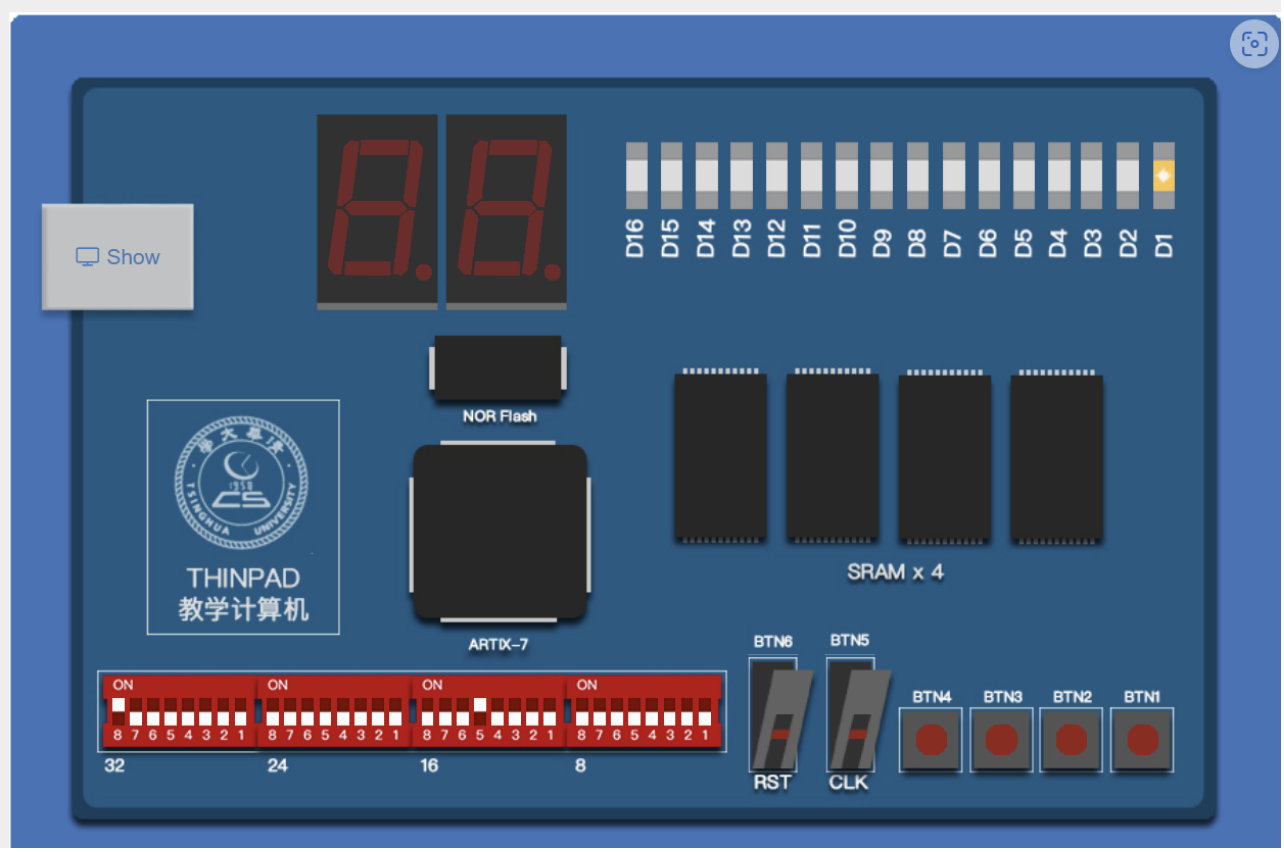
```
[180166000]: uart received 0x32, ASCII: 2
[275651000]: uart received 0x33, ASCII: 3
[373307000]: uart received 0x34, ASCII: 4
[469878000]: uart received 0x35, ASCII: 5
[566449000]: uart received 0x36, ASCII: 6
[664104000]: uart received 0x38, ASCII: 8
[760585000]: uart received 0x39, ASCII: 9
[857155000]: uart received 0x41, ASCII: A
[954811000]: uart received 0x42, ASCII: B
[1051382000]: uart received 0x43, ASCII: C
$finish called at time : 1069513053 ps : File "D:/coderyxy4/cod24-xy-yu22/thinpad_top.srcs/sim_1/new/tb/lab5_tb.sv" Line 102
run: Time (s): cpu = 00:00:00 ; elapsed = 00:00:05 . Memory (MB): peak = 1330.082 ; gain = 0.000
```

如图，有正确写回到串口



在比较阶段有正确的值读到

3. 云平台仿真



可见完成 cmp 时达到了正确的结果

思考题

阅读 Wishbone UART 控制器的代码，体会 MMIO 寄存器的概念。映射到地址空间上的“内存”的内容一定是只受 Master 端控制的吗？如何将数码管和拨码开关也映射到地址空间上？

- MMIO 体现了 “万物皆地址” 的思想，可以把对外设等的访问抽象为对内存地址的访问
- “内存”的内容不一定只受 Master 端控制，如像外设这种，可能对我们 Master 端而言就像黑盒一样
- 将数码管和拨码开关也映射到地址空间，需要以下几步，首先我们把他们当作 wishbone slave 设置好接口，再通过 wb_mux 来实现，如设置 `.wbs1_addr`, `.wbs1_addr_msk` 为数码管和拨码开关的地址，然后通过 “设置该地址的值为我们想要显示的数” 的方式来使得数码管和拨码开关得到输入输出