**实验报告**

2021年05月25日 成绩：---------

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 刘爱兵 | 学号 | 19114801 | 班级 | 19计科2班 |
| 专业 | 计算机科学与技术 | 课程名称 | 计算机组成原理实验 | 任课教师 | 吴云志 |
| 实验序号 | 04 | 实验名称 | 存储器实验 | | |
| 实验时间 | 2021.05.25 | 机位号 | 18 | 实验设备号 | 21 |

1. **实验目的与要求**
2. 实验目的

掌握灵活运用Verilog HDL语言进行各种描述与建模的技巧和方法；

学习在Vivado中设计生成Memory IP核的方法；

学习存储器的结构及读写原理，掌握存储器的设计方法；

学习MIPS指令系统，熟悉MIPS指令格式及其汇编指令助记符，掌握机器指令编码方法；

学习MIPS汇编程序设计，学会使用MIPS汇编器将汇编语言程序翻译成二进制文件；

了解使用MIPS教学系统模拟器运行程序的方法；

1. 实验要求

按照方法一，编程实现基本的存储器模块，并通过仿真验证

按照方法二，生成一个RAM\_B存储器模块，关联文件中输入64个32位数据，16进制表示

编写一个实验验证的顶层模块，调用方法二生成的存储器模块；

课前任务：编程、仿真、验证，确保逻辑正确性；

方法一：采用Verilog语言中存储器类型（即reg的数组类型）定义，并自行管理；

方法二：使用FPGA内置的存储器IP核来实现。

两者的区别：

前者（采用Verilog语言中存储器类型）：

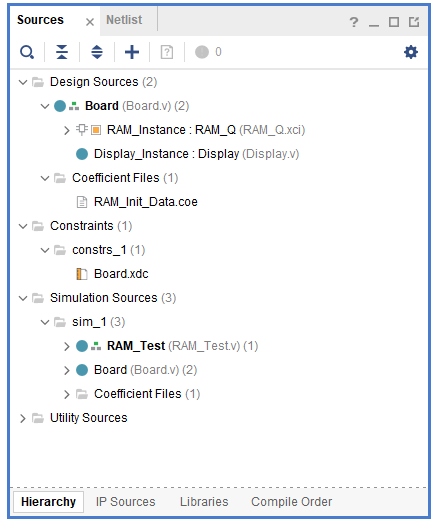
存储器模块需要简单编程来完成读写操作，且其中的数据（或指令）的初始化，也需要在模块内编程赋值完成；

后者（使用FPGA内置的存储器IP核）

无需编程，通过向导自动生成了一个存储模块，只需引用实例即可，且其中的数据（或指令）的初始化操作可以和外部的一个格式化文件关联，ISE /Vavido会自动从该文件装载程序或数据。另外，由Memory IP核实现的存储器模块性能更好、更可靠。

1. **实验设计与程序代码**
2. 模块设计说明

（描述这个实验的设计方案，分几个模块，各模块的功能，各模块之间的连接关系，可附图）



1. 实验程序源代码及注释等

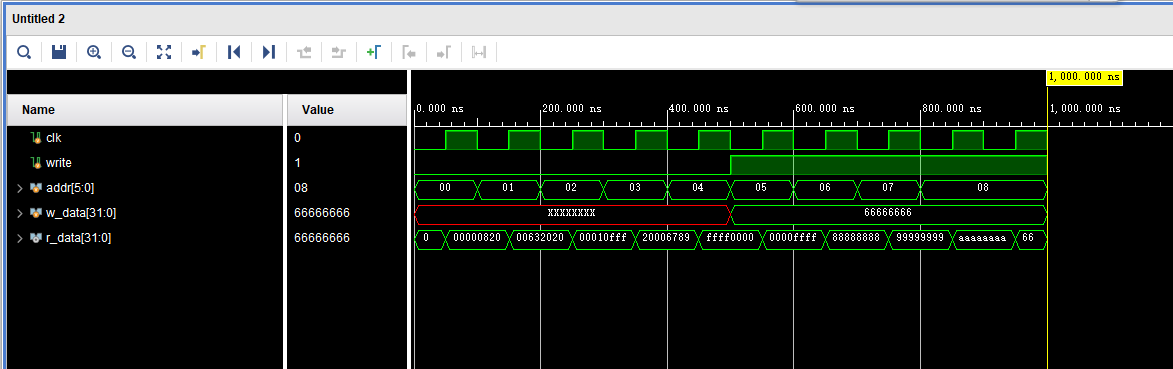
（实验各个模块的代码，包含功能注释）

|  |
| --- |
| 1. **Board.v**   `timescale 1ns / 1ps  // 单端口读优先 RAM IP核板极调试  module Board(sw, swb, led, clk, which, seg, enable);  input [1:32] sw;  input [1:6] swb;  output [1:32] led;  reg [5:0] addr;  reg [31:0] data;  RAM RAM\_Instance(  .clka(swb[1]), // SWB1：时钟  .wea(swb[2]), // SWB2：写入控制  .addra(addr),  .dina(sw), // 按住 SWB2 时按下 SWB1，将 SW 数据写入 RAM  .douta(led)); // LED：显示读出数据  always @(posedge swb[6]) addr <= sw[27:32]; // SWB6：将 SW 末六位提交至 读写地址  input clk; // 数码管相关  output [2:0] which;  output [7:0] seg;  output reg enable = 1; // 默认开启数码管使能，显示 RAM 读写地址  Display Display\_Instance(.clk(clk), .which(which), .seg(seg),  .data({3'b0, swb[1], // 最高位显示 时钟 电平状态  3'b0, swb[2], // 次高位显示 写入控制 电平状态  18'b0, addr})); // 最低两位显示 读写地址  endmodule // Board  **（2）RAM\_Init\_Data.coe**  memory\_initialization\_radix=16;  memory\_initialization\_vector=00000820 00632020 00010fff 20006789 FFFF0000 0000FFFF 88888888 99999999 aaaaaaaa bbbbbbbb; |

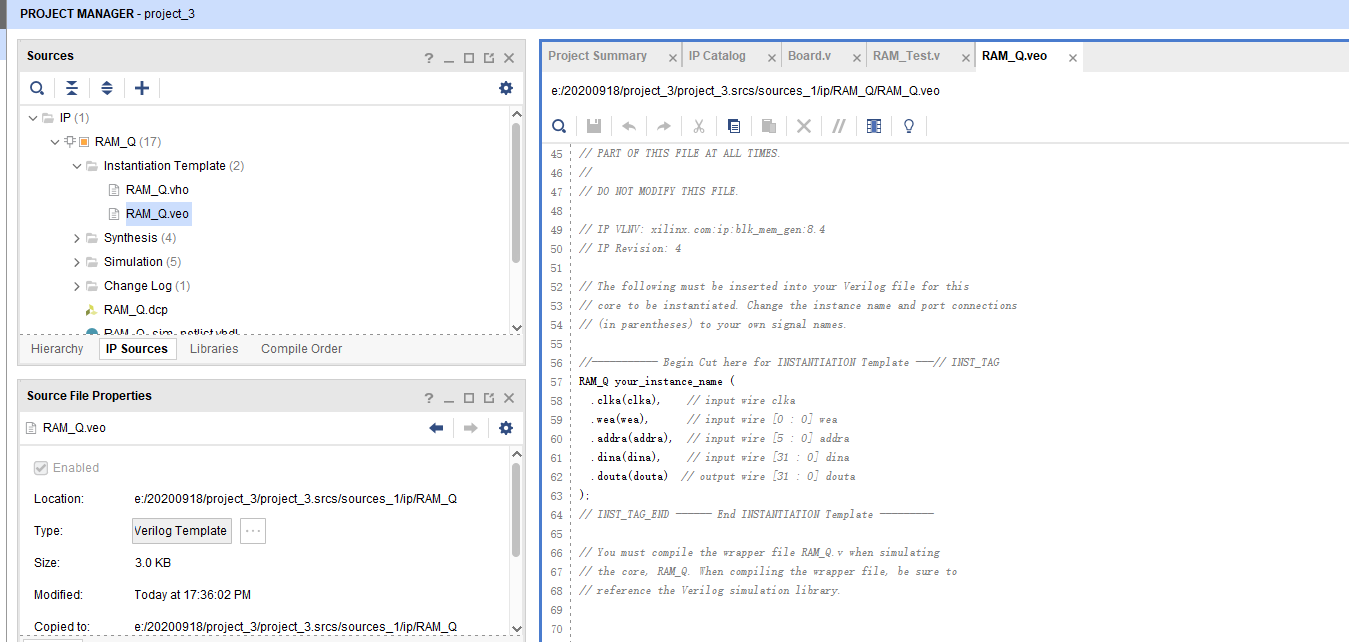
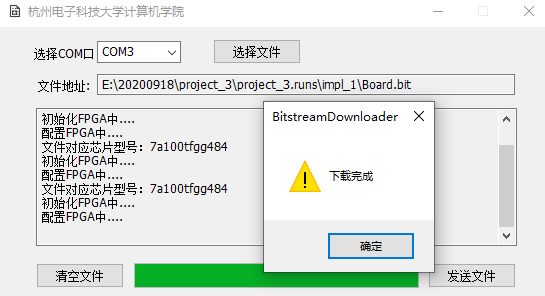
1. **实验仿真**
2. 仿真代码（仿真源代码）

|  |
| --- |
| `timescale 1ns / 1ps  // 单端口读优先 RAM IP核仿真测试  module RAM\_Test();  reg clk, write;  reg [5:0] addr;  reg [31:0] w\_data; // input  wire [31:0] r\_data; // output  RAM RAM\_Instance( // Instantiate a RAM IP core  .clka(clk), // input wire clka  .wea(write), // input wire [0 : 0] wea  .addra(addr), // input wire [5 : 0] addra  .dina(w\_data), // input wire [31 : 0] w\_data  .douta(r\_data)); // output wire [31 : 0] douta  always #50 clk = ~clk;  initial begin // #1000 r\_data changes when posedge of clk  clk = 0; write = 0; // read  addr = 0; #100; // 0000\_0000 -> 0000\_0820  addr = 1; #100; // 0000\_0820 -> 0063\_2020  addr = 2; #100; // 0063\_2020 -> 0001\_0fff  addr = 3; #100; // 0001\_0fff -> 2000\_6789  addr = 4; #100; // 2000\_6789 -> ffff\_0000  w\_data = 32'h6666\_6666; write = 1; // write  addr = 5; #100; // ffff\_0000 -> 0000\_ffff  addr = 6; #100; // 0000\_ffff -> 8888\_8888  addr = 7; #100; // 8888\_8888 -> 9999\_9999  addr = 8; #100; // 9999\_9999 -> aaaa\_aaaa  addr = 8; #100; // aaaa\_aaaa -> 6666\_6666，可判断出读取优先  end  endmodule // RAM\_Test |

1. 仿真波形（运行仿真时波形截图）

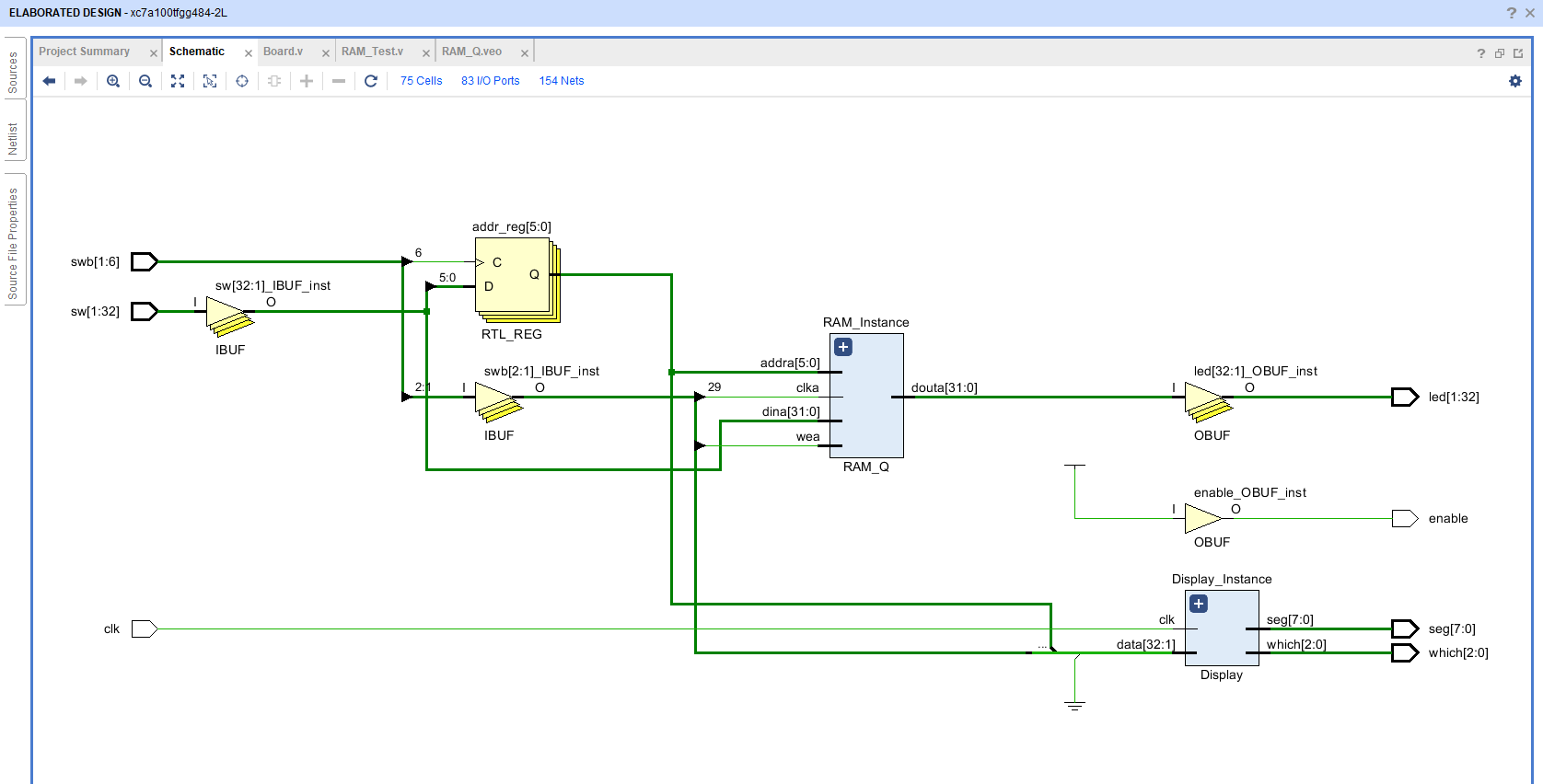


1. 仿真结果分析（对仿真波形进行分析）



1. **电路图**

（开发工具中显示的电路模块图）



1. **引脚配置**

（引脚约束文件的内容，描述主要配置情况）

|  |
| --- |
| # 开启比特流压缩，优化 .bit 文件大小  set\_property BITSTREAM.GENERAL.COMPRESS TRUE [current\_design]  # Switch，开关  set\_property PULLDOWN true [get\_ports sw]  set\_property IOSTANDARD LVCMOS18 [get\_ports sw]  set\_property PACKAGE\_PIN T3 [get\_ports {sw[1]}]  set\_property PACKAGE\_PIN U3 [get\_ports {sw[2]}]  set\_property PACKAGE\_PIN T4 [get\_ports {sw[3]}]  set\_property PACKAGE\_PIN V3 [get\_ports {sw[4]}]  set\_property PACKAGE\_PIN V4 [get\_ports {sw[5]}]  set\_property PACKAGE\_PIN W4 [get\_ports {sw[6]}]  set\_property PACKAGE\_PIN Y4 [get\_ports {sw[7]}]  set\_property PACKAGE\_PIN Y6 [get\_ports {sw[8]}]  set\_property PACKAGE\_PIN W7 [get\_ports {sw[9]}]  set\_property PACKAGE\_PIN Y8 [get\_ports {sw[10]}]  set\_property PACKAGE\_PIN Y7 [get\_ports {sw[11]}]  set\_property PACKAGE\_PIN T1 [get\_ports {sw[12]}]  set\_property PACKAGE\_PIN U1 [get\_ports {sw[13]}]  set\_property PACKAGE\_PIN U2 [get\_ports {sw[14]}]  set\_property PACKAGE\_PIN W1 [get\_ports {sw[15]}]  set\_property PACKAGE\_PIN W2 [get\_ports {sw[16]}]  set\_property PACKAGE\_PIN Y1 [get\_ports {sw[17]}]  set\_property PACKAGE\_PIN AA1 [get\_ports {sw[18]}]  set\_property PACKAGE\_PIN V2 [get\_ports {sw[19]}]  set\_property PACKAGE\_PIN Y2 [get\_ports {sw[20]}]  set\_property PACKAGE\_PIN AB1 [get\_ports {sw[21]}]  set\_property PACKAGE\_PIN AB2 [get\_ports {sw[22]}]  set\_property PACKAGE\_PIN AB3 [get\_ports {sw[23]}]  set\_property PACKAGE\_PIN AB5 [get\_ports {sw[24]}]  set\_property PACKAGE\_PIN AA6 [get\_ports {sw[25]}]  set\_property PACKAGE\_PIN R2 [get\_ports {sw[26]}]  set\_property PACKAGE\_PIN R3 [get\_ports {sw[27]}]  set\_property PACKAGE\_PIN T6 [get\_ports {sw[28]}]  set\_property PACKAGE\_PIN R6 [get\_ports {sw[29]}]  set\_property PACKAGE\_PIN U7 [get\_ports {sw[30]}]  set\_property PACKAGE\_PIN AB7 [get\_ports {sw[31]}]  set\_property PACKAGE\_PIN AB8 [get\_ports {sw[32]}]  # Switch Button，按钮  set\_property IOSTANDARD LVCMOS18 [get\_ports swb]  set\_property PACKAGE\_PIN R4 [get\_ports {swb[1]}]  set\_property PACKAGE\_PIN AA4 [get\_ports {swb[2]}]  set\_property PACKAGE\_PIN AB6 [get\_ports {swb[3]}]  set\_property PACKAGE\_PIN T5 [get\_ports {swb[4]}]  set\_property PACKAGE\_PIN V8 [get\_ports {swb[5]}]  set\_property PACKAGE\_PIN AA8 [get\_ports {swb[6]}]  # LED  set\_property IOSTANDARD LVCMOS18 [get\_ports led]  set\_property PACKAGE\_PIN R1 [get\_ports {led[1]}]  set\_property PACKAGE\_PIN P2 [get\_ports {led[2]}]  set\_property PACKAGE\_PIN P1 [get\_ports {led[3]}]  set\_property PACKAGE\_PIN N2 [get\_ports {led[4]}]  set\_property PACKAGE\_PIN M1 [get\_ports {led[5]}]  set\_property PACKAGE\_PIN M2 [get\_ports {led[6]}]  set\_property PACKAGE\_PIN L1 [get\_ports {led[7]}]  set\_property PACKAGE\_PIN J2 [get\_ports {led[8]}]  set\_property PACKAGE\_PIN G1 [get\_ports {led[9]}]  set\_property PACKAGE\_PIN E1 [get\_ports {led[10]}]  set\_property PACKAGE\_PIN D2 [get\_ports {led[11]}]  set\_property PACKAGE\_PIN A1 [get\_ports {led[12]}]  set\_property PACKAGE\_PIN L3 [get\_ports {led[13]}]  set\_property PACKAGE\_PIN G3 [get\_ports {led[14]}]  set\_property PACKAGE\_PIN K4 [get\_ports {led[15]}]  set\_property PACKAGE\_PIN G4 [get\_ports {led[16]}]  set\_property PACKAGE\_PIN K1 [get\_ports {led[17]}]  set\_property PACKAGE\_PIN J1 [get\_ports {led[18]}]  set\_property PACKAGE\_PIN H2 [get\_ports {led[19]}]  set\_property PACKAGE\_PIN G2 [get\_ports {led[20]}]  set\_property PACKAGE\_PIN F1 [get\_ports {led[21]}]  set\_property PACKAGE\_PIN E2 [get\_ports {led[22]}]  set\_property PACKAGE\_PIN D1 [get\_ports {led[23]}]  set\_property PACKAGE\_PIN B1 [get\_ports {led[24]}]  set\_property PACKAGE\_PIN B2 [get\_ports {led[25]}]  set\_property PACKAGE\_PIN N3 [get\_ports {led[26]}]  set\_property PACKAGE\_PIN M3 [get\_ports {led[27]}]  set\_property PACKAGE\_PIN K3 [get\_ports {led[28]}]  set\_property PACKAGE\_PIN H3 [get\_ports {led[29]}]  set\_property PACKAGE\_PIN N4 [get\_ports {led[30]}]  set\_property PACKAGE\_PIN L4 [get\_ports {led[31]}]  set\_property PACKAGE\_PIN J4 [get\_ports {led[32]}]  # 数码管相关  set\_property IOSTANDARD LVCMOS18 [get\_ports seg]  set\_property PACKAGE\_PIN H19 [get\_ports {seg[7]}]  set\_property PACKAGE\_PIN G20 [get\_ports {seg[6]}]  set\_property PACKAGE\_PIN J22 [get\_ports {seg[5]}]  set\_property PACKAGE\_PIN K22 [get\_ports {seg[4]}]  set\_property PACKAGE\_PIN K21 [get\_ports {seg[3]}]  set\_property PACKAGE\_PIN H20 [get\_ports {seg[2]}]  set\_property PACKAGE\_PIN H22 [get\_ports {seg[1]}]  set\_property PACKAGE\_PIN J21 [get\_ports {seg[0]}]  set\_property IOSTANDARD LVCMOS18 [get\_ports which]  set\_property PACKAGE\_PIN N22 [get\_ports {which[0]}]  set\_property PACKAGE\_PIN M21 [get\_ports {which[1]}]  set\_property PACKAGE\_PIN M22 [get\_ports {which[2]}]  set\_property -dict {IOSTANDARD LVCMOS18 PACKAGE\_PIN L21} [get\_ports enable]  set\_property -dict {IOSTANDARD LVCMOS18 PACKAGE\_PIN H4} [get\_ports clk]  # [Place 30-574] Poor placement for routing between an IO pin and BUFG.If this  # sub optimal condition is acceptable for this design, you may use the  # CLOCK\_DEDICATED\_ROUTE constraint in the .xdc file to demote this message to a  # WARNING. However, the use of this override is highly discouraged.  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets clk\_IBUF]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[1]]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[2]]  # set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[3]]  # set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[4]]  # set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[5]]  set\_property CLOCK\_DEDICATED\_ROUTE FALSE [get\_nets swb\_IBUF[6]] |

1. **思考与探索**
2. 实验结果记录：

（实验操作的过程及结果记录）

SWB1 时钟信号

SWB2 写入控制，按住后再按下 SWB1，将 SW 当前输入数据写入 RAM IP 核

SWB6 将 SW 末六位提交至读写地址

LED 显示读出数据

1. 实验结论

（分析实验结果，给出实验结论）

实验结果与逻辑相符，操作板控制正确。

1. 问题与解决方案：

（整个实验过程中发生了什么问题？你是如何解决的。）

实验代码阅读有些许困难，与同学讨论后顺利解决。

4、思考与探索

（1）选择8个存储器单元执行读操作，将实验结果记录到表中的第二列和第三列，分析你的读出数据是否和初始化关联文件中的数据一致；若不一致，分析原因。

答：一致。

（2）对上面的8个存储器单元执行写操作，覆盖初始化数据，然后再执行读操作，将读出数据记录到表的第四列和第五列中。这些单元的数据有否改写？分析读出数据是否和写入数据一致；如果不一致，请分析原因。

答：一致。