实验名称：石头剪刀布 姓名： 学号：

装

订

线

浙江大学

本科实验报告

姓名：

学院：生物医学工程与仪器科学学院

系：生物医学工程系

专业：生物医学工程

学号：

指导教师：余锋

实验名称：石头剪刀布 姓名：__ 学号：

专业：生物医学工程
姓名：
学号：
日期：2025.4.1
地点：教 6－204

浙江大学 实验报告

课程名称：硬件描述语言 指导老师：余锋 实验名称：HDL 实验二

装 订 线

一、实验要求

模拟实现一个宽度为 32、深度为 256 的内存空间，先向内存空间写入一批数据，再读出这批数据，并比较数据是否正确。

```
module ram(
input           clk_i,
input           rst_i,
input           wr_en_i,
input [7:0]     addr_i,
input [31:0]        wr_data_i,
output reg [31:0]   rd_data_o
);

reg [31:0]      bram [255:0];

//add code here
endmodule
```

提供输入：clk_i 时钟信号，100MHz

rst_i 复位信号，高电平复位

wr_en_i 写使能，表示要求向内存写入数据

addr_i 地址总线，表示要访问的内存地址

wr_data_i 写数据总线，表示要写入内存的数据

实验名称：___石头剪刀布___ 姓名：__ 学号：

要求输出：rd_data_o　　读数据总线，表示从内存读出的数据

实验要求：

1、RAM 空间不需要赋初值。

2、rst_i 信号拉高后将 RAM 输出端口置 0，不对 RAM 进行写入操作。

3、rd_data_o 会延迟 addr_i 一个时钟周期。

4、编写 no change 模式、write first 模式、read first 模式的 RAM

no change 模式：wr_en_i 有效时，读操作无效，rd_data_o 保持不变

write first 模式：wr_en_i 有效时，rd_data_o 输出当前地址新写入的数据

read first 模式：wr_en_i 有效时，rd_data_o 输出当前地址原有的数据

5、使用 task 来实现 testbench 的编写，testbench 中地址总线与写数据总线的变化需要与时钟上升沿对齐.

6、Testbench 流程如下

第一步：rst 为 1’b0，wr_en 为 1’b1，在地址区间 8’d0-8’d15 写入 32’d0-32’d15

第二步：rst 为 1’b0，wr_en 为 1’b1，在地址区间 8’d0-8’d15 写入 32’d16-32’d31

第三步：rst 为 1’b1，wr_en 为 1’b0，addr 从 8’d0 走到 8’d15，同时 data_i 对应地址变化从 32’d32 走到 32’d47

第四步：rst 为 1’b0，wr_en 为 1’b0，addr 从 8’d0 走到 8’d15，同时 data_i 对应地址变化从 32’d0 走到 32’d15

7、将输入输出端口合并为一个 inout 类型端口，使用三态门加以实现。（扩展，不做要求，设计合理即可）

二、实验代码与注释

1、ram.v

ram_no_change 模块：实现了 no change 模式的 RAM。在复位信号 rst_i 有效时，将

实验名称：___石头剪刀布___ 姓名：__ 学号：

输出 rd_data_o 置为 0；当写使能信号 wr_en_i 有效时，将数据写入内存；当写使能无效时，从内存中读取数据输出。

ram_write_first 模块：实现了 write first 模式的 RAM。在复位时输出置 0，写使能有效时先写入数据再输出新写入的数据，写使能无效时从内存读取数据输出。

ram_read_first 模块：实现了 read first 模式的 RAM。复位时输出置 0，写使能有效时先读取原有数据输出再写入新数据，写使能无效时从内存读取数据输出。

```
`timescale 1ns / 1ps
module ram_no_change(
    input               clk_i,
    input               rst_i,
    input               wr_en_i,
    input [7:0]         addr_i,
    input [31:0]        wr_data_i,
    output reg [31:0]    rd_data_o
);
    reg [31:0]    bram[255:0];
    // 定义名为 bram 的 32 比特宽、深度为 256 的寄存器数组，用于模拟内存空间
    always @(posedge clk_i) begin
        if (rst_i) begin
            rd_data_o <= 32'b0;
            // 将输出数据 rd_data_o 置为 32 位全 0，表示复位操作
        end else begin
            if (wr_en_i) begin
                bram[addr_i] <= wr_data_i;
                // 将写数据 wr_data_i 写入到指定地址 addr_i 的内存中
            end else begin
                rd_data_o <= bram[addr_i];
                // 从指定地址 addr_i 的内存中读取数据，并将其赋值给输出数据
rd_data_o。
            end
        end
    end
endmodule
module ram_write_first(
    // 实现写优先模式的 RAM
    input               clk_i,
    input               rst_i,
    input               wr_en_i,
    input [7:0]         addr_i,
    input [31:0]        wr_data_i,
    output reg [31:0]    rd_data_o
```

实验名称：___石头剪刀布___ 姓名：__ 学号：

装

订

线

```
);
    reg [31:0]    bram[255:0];
    always @(posedge clk_i) begin
        // 在时钟上升沿触发执行
        if (rst_i) begin
            // 复位操作
            rd_data_o <= 32'b0;
        end else begin
            if (wr_en_i) begin
                // 写使能有效时
                bram[addr_i] <= wr_data_i;
                // 先将数据写入指定地址的内存
                rd_data_o <= wr_data_i;
                // 然后将写入的数据直接输出，实现写优先
            end else begin
                // 写使能无效时
                rd_data_o <= bram[addr_i];
                // 从内存中读取数据并输出
            end
        end
    end
endmodule
module ram_read_first(
    // 实现读优先模式的 RAM。
    input               clk_i,
    input               rst_i,
    input               wr_en_i,
    input [7:0]         addr_i,
    input [31:0]        wr_data_i,
    output reg [31:0]    rd_data_o
);
    reg [31:0]    bram[255:0];
    // 定义内存数组
    always @(posedge clk_i) begin
        // 在时钟上升沿触发
        if (rst_i) begin
            // 复位操作
            rd_data_o <= 32'b0;
        end else begin
            if (wr_en_i) begin
                // 写使能有效时
                rd_data_o <= bram[addr_i];
                // 先从内存中读取原有数据并输出
                bram[addr_i] <= wr_data_i;
```

实验名称：___石头剪刀布___ 姓名：__ 学号：

```
                // 然后将新数据写入内存
            end else begin
                // 写使能无效时
                rd_data_o <= bram[addr_i];
                // 从内存中读取数据并输出
            end
        end
    end
endmodule
```

2、tb_ram.v

测试平台模块 tb_ram：包含时钟信号 clk、复位信号 rst、写使能信号 wr_en、地址信号 addr 和写数据信号 wr_data，以及三个不同模式 RAM 的输出信号。

时钟生成：通过 initial 块和 forever 循环生成 100MHz 的时钟信号。

测试任务 test_sequence：按照实验要求的四个步骤进行测试，每个步骤通过 for 循环和 @(posedge clk) 确保地址和数据的变化与时钟上升沿对齐。

仿真控制：使用 $dumpfile 和 $dumpvars 生成仿真波形文件，调用 test_sequence 任务进行测试，最后使用 $finish 结束仿真。

```
`timescale 1ns / 1ps

module tb_ram;
    reg           clk;
    reg           rst;
    reg           wr_en;
    reg  [7:0]    addr;
    reg  [31:0]   wr_data;
    wire [31:0]   rd_data_no_change;
    wire [31:0]   rd_data_write_first;
    wire [31:0]   rd_data_read_first;
    // 实例化模块
    ram_no_change uut_no_change (
        .clk_i(clk),
        .rst_i(rst),
        .wr_en_i(wr_en),
        .addr_i(addr),
        .wr_data_i(wr_data),
        .rd_data_o(rd_data_no_change)
    );
    ram_write_first uut_write_first (
```

装 订 线

实验名称：___石头剪刀布___ 姓名：__ 学号：

装订线

```
        .clk_i(clk),
        .rst_i(rst),
        .wr_en_i(wr_en),
        .addr_i(addr),
        .wr_data_i(wr_data),
        .rd_data_o(rd_data_write_first)
    );
    ram_read_first uut_read_first (
        .clk_i(clk),
        .rst_i(rst),
        .wr_en_i(wr_en),
        .addr_i(addr),
        .wr_data_i(wr_data),
        .rd_data_o(rd_data_read_first)
    );
    // 时钟生成
    initial begin
        clk = 0;
        // 初始化时钟信号为低电平
        forever #5 clk = ~clk; // 100MHz 时钟，周期为 10ns
        // forever 无限循环，#5 表示延迟 5 个时间单位（这里是 5ns）
        // ~clk 表示对时钟信号取反，从而实现时钟信号的周期性变化，周期为 10ns，
对应 100MHz 的时钟频率。
    end

    // 测试任务
    task test_sequence;
        integer i;   // 用于循环计数。
        // task1
        rst = 1'b0;
        // 将复位信号置为低电平，表示不进行复位操作。
        wr_en = 1'b1;
        // 将写使能信号置为高电平，表示允许写入数据。
        for (i = 0; i < 16; i = i + 1) begin
            // 循环 16 次，从地址 0 到 15。
            @(posedge clk);
            // 等待时钟信号的上升沿，确保地址和数据的变化与时钟上升沿对齐。
            addr = i;
            // 将地址信号设置为当前循环的计数值。
            wr_data = i;
            // 将写数据信号设置为当前循环的计数值。
            #1;

        end
```

实验名称：___石头剪刀布___ 姓名：__ 学号：

装

订

线

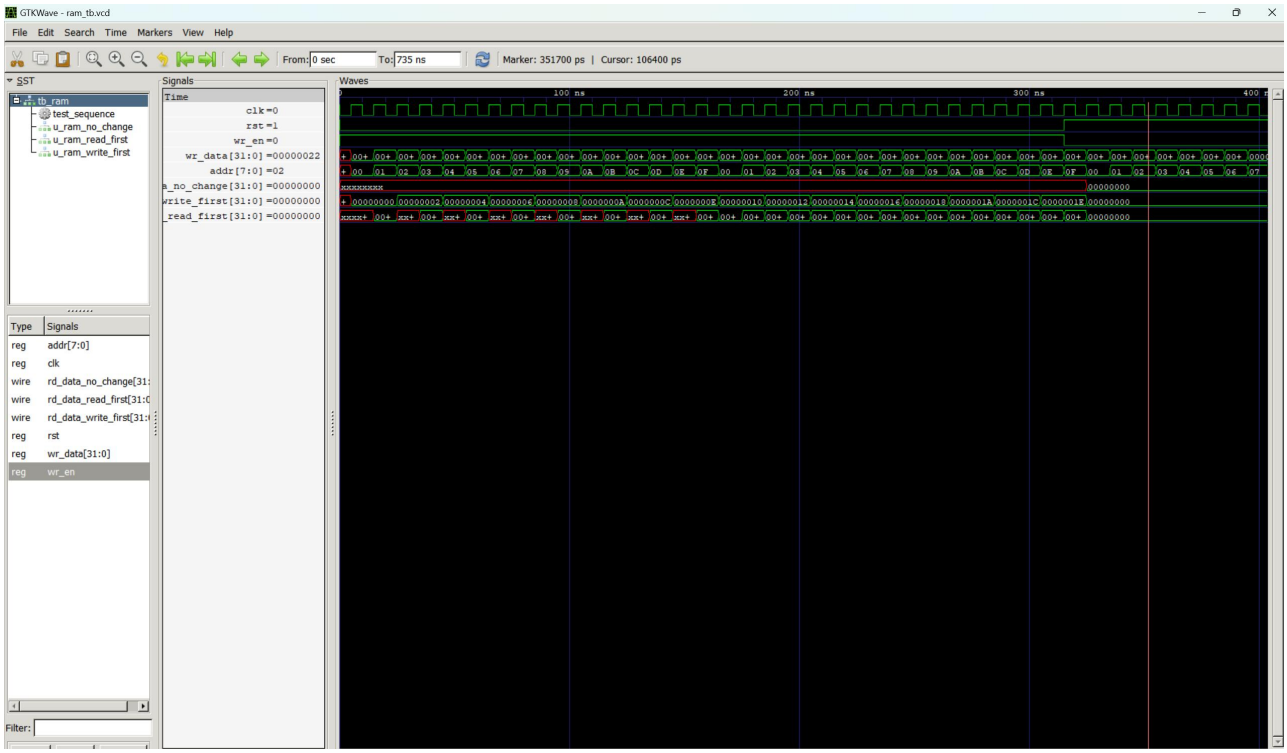
```
        // task2
        for (i = 0; i < 16; i = i + 1) begin
            // 再次循环 16 次
            @(posedge clk);
            addr = i;
            wr_data = i + 16;
            // 这次写入的数据是计数值加 16。
            #1;

        end
        // task3
        rst = 1'b1;
        // 将复位信号置为高电平，进行复位操作。
        wr_en = 1'b0;
        // 将写使能信号置为低电平，禁止写入数据。
        for (i = 0; i < 16; i = i + 1) begin
            // 循环 16 次
            @(posedge clk);
            addr = i;
            wr_data = i + 32;
            // 地址从 0 到 15，写数据从 32 到 47。
            #1;

        end
        // task4
        rst = 1'b0;
        // 复位信号置为低电平。
        wr_en = 1'b0;
        // 写使能信号置为低电平。
        for (i = 0; i < 16; i = i + 1) begin
            // 循环 16 次
            @(posedge clk);
            addr = i;
            wr_data = i;
            #1;
            // 地址从 0 到 15，写数据从 0 到 15。
        end
    endtask
    // 开始测试
    initial begin
        $dumpfile("ram_tb.vcd");
        $dumpvars(0, tb_ram);
        test_sequence;
        #100 $finish;
```

实验名称：___石头剪刀布___ 姓名：__ 学号：

```
    end
endmodule
```

三、仿真结果与分析

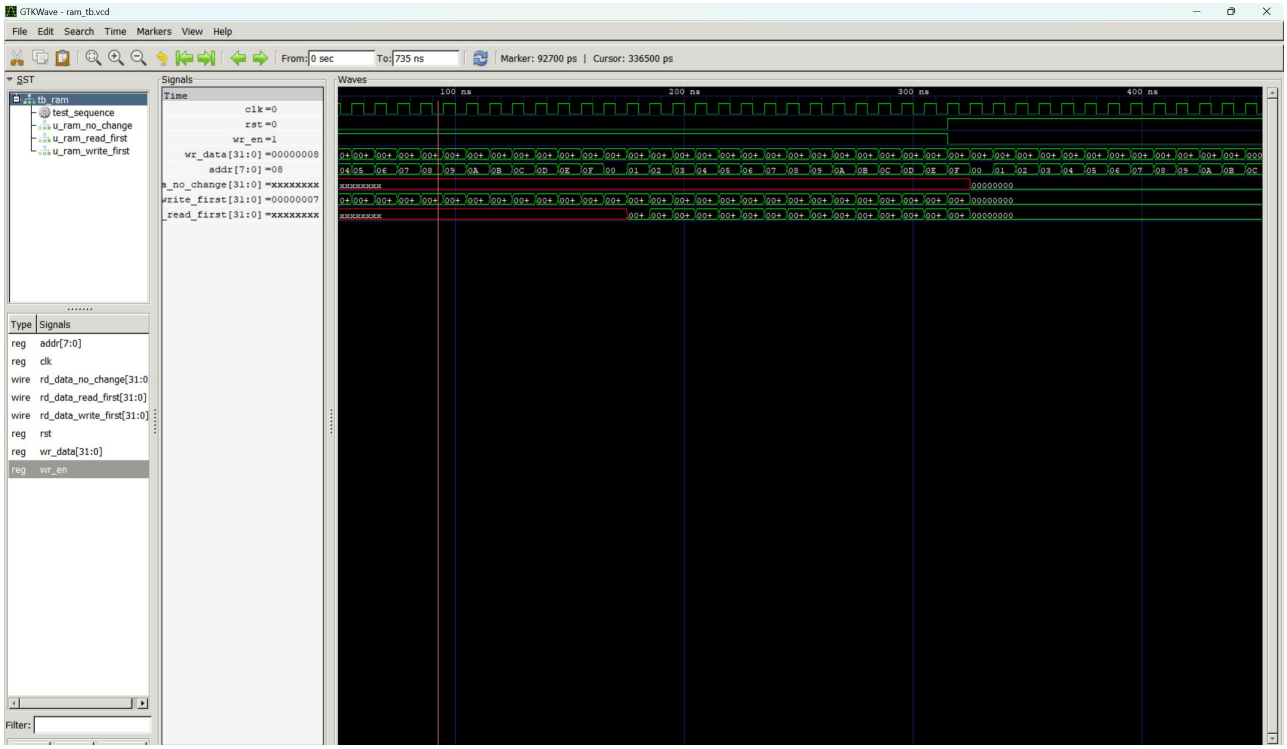
本次实验同样选择采用 iverilog + gtkwave 进行仿真波形分析。在初版代码完成之后出现了一些小问题：



将 tb 改为非阻塞赋值后：

装 订 线

实验名称：___石头剪刀布___ 姓名：__ 学号：



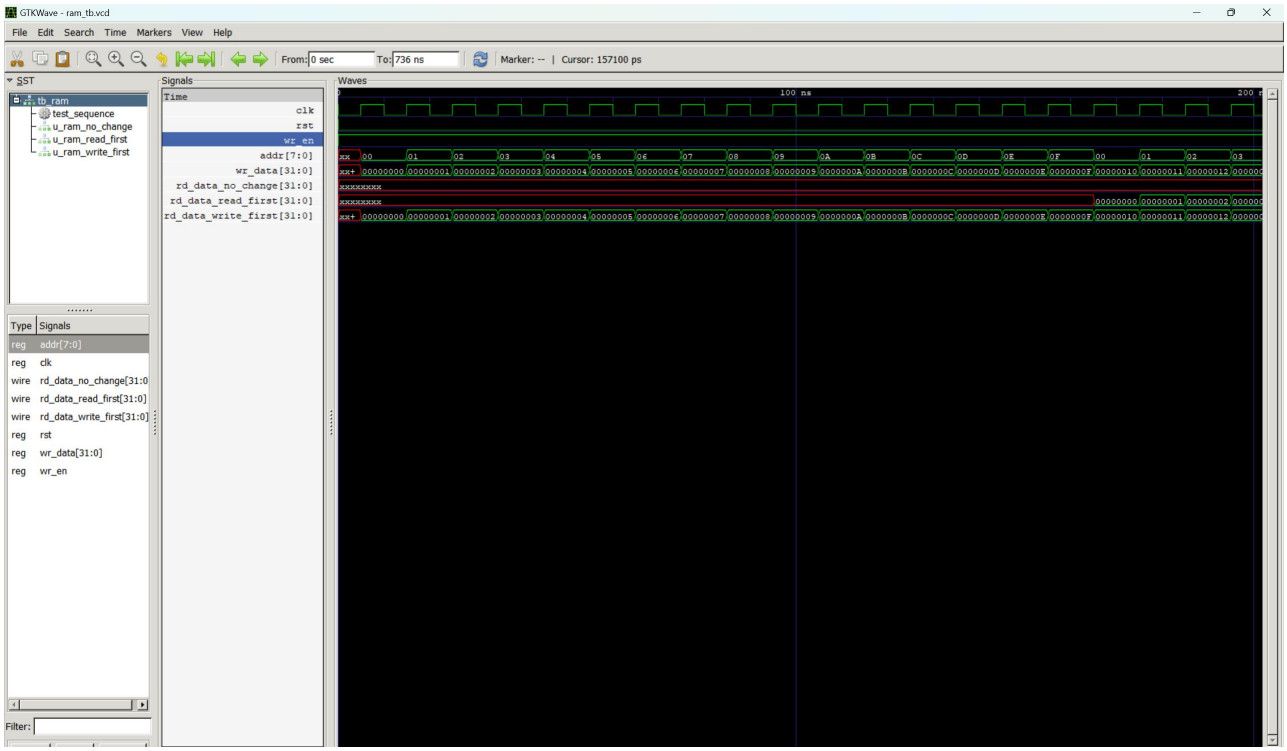
装

订

线

rd_data_write_first 与 rd_data_read_first 显示一个周期延迟。解决方法是在 test_sequence 任务中，在更新 addr 和 wr_data 后，可以增加一个短暂的延迟（比如 #1），从而成功确保 addr 信号已经稳定，并且在模块中被正确采样。

1、Task1



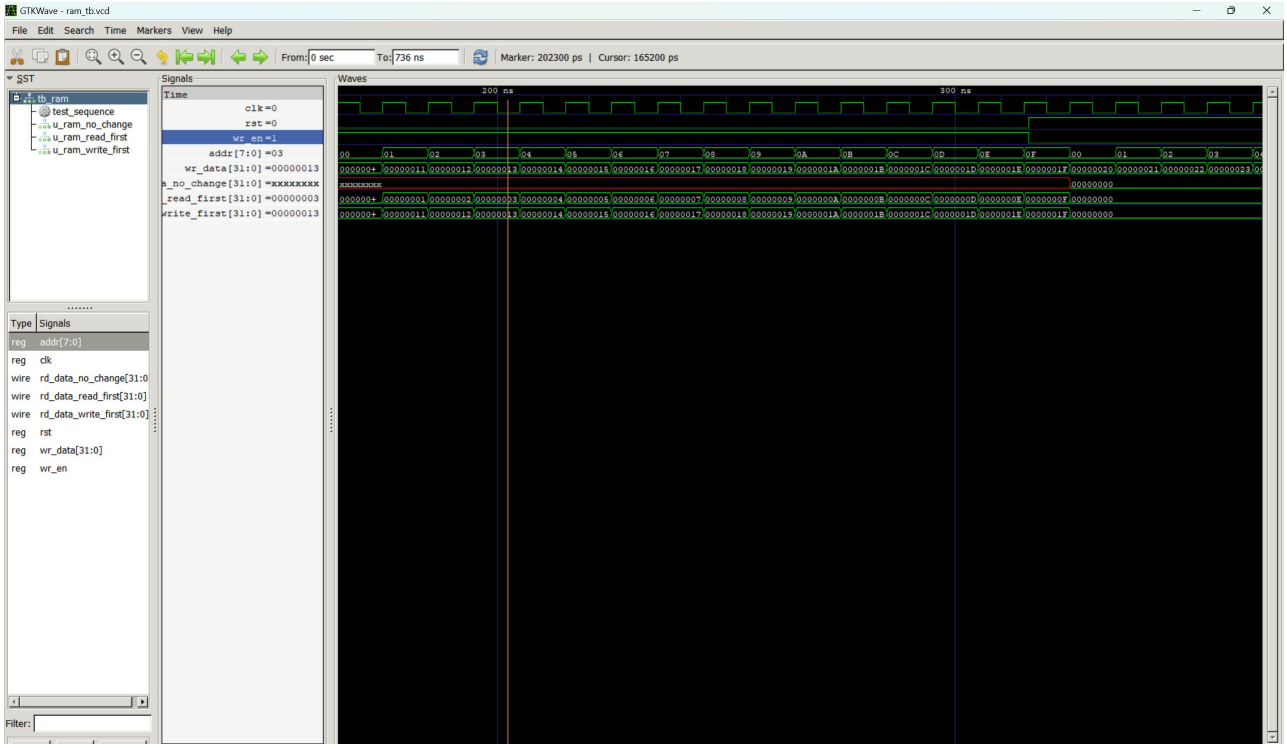
实验名称：___石头剪刀布___ 姓名：__ 学号：

rd_data_no_change：在写操作期间保持不变，直到写使能无效后才根据地址输出对应内存的值。这是因为 no change 模式下，写使能有效时读操作无效，只有写使能无效时才进行读操作。

rd_data_write_first：在写操作时，输出当前写入的数据。符合 write first 模式的特点，即写使能有效时先写入数据并立即输出新写入的数据。

rd_data_read_first：在写操作时，先输出当前地址原有的数据（初始为 xx），然后写入新数据。这是 read first 模式的特性，写使能有效时先读取原有数据输出再写入新数据。

2、Task2



rd_data_no_change：保持不变，直到写使能无效。

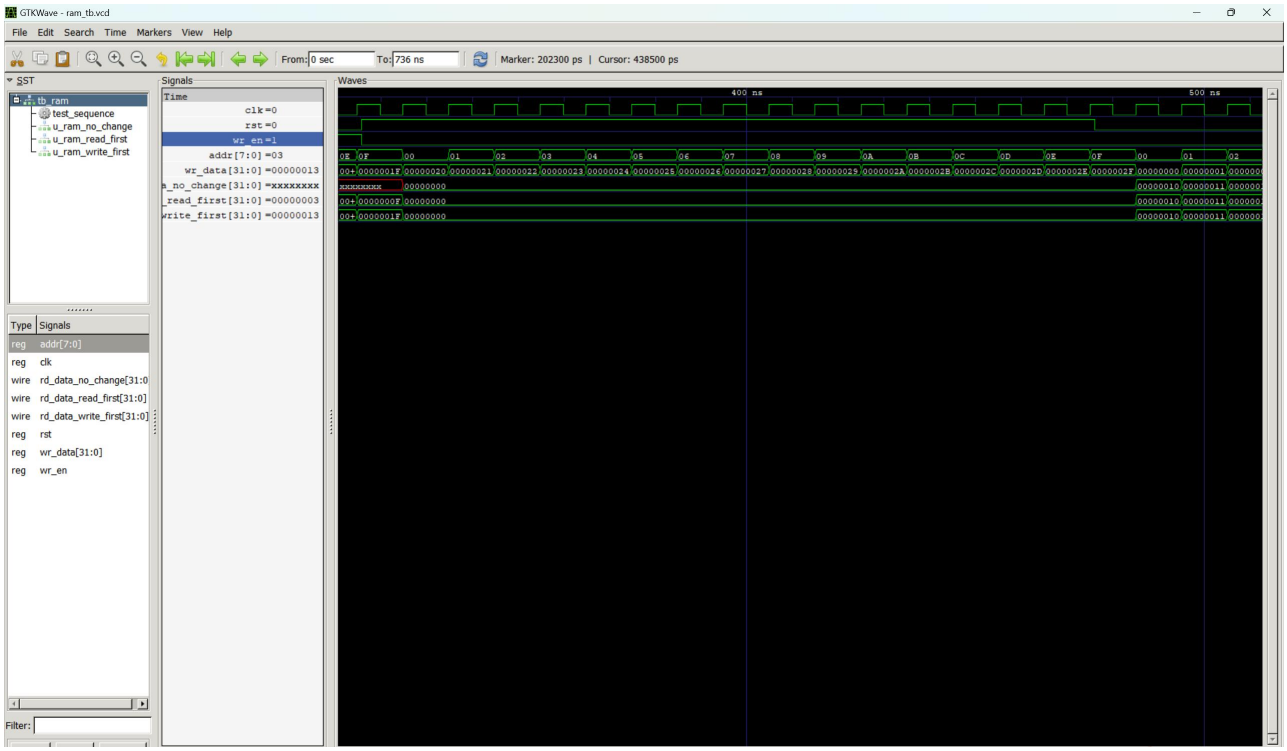
rd_data_write_first：输出当前写入的数据。

rd_data_read_first：先输出当前地址原有的数据（即任务 1 写入的数据），然后写入新数据。

3、Task3

装 订 线

实验名称：___石头剪刀布___ 姓名：__ 学号：



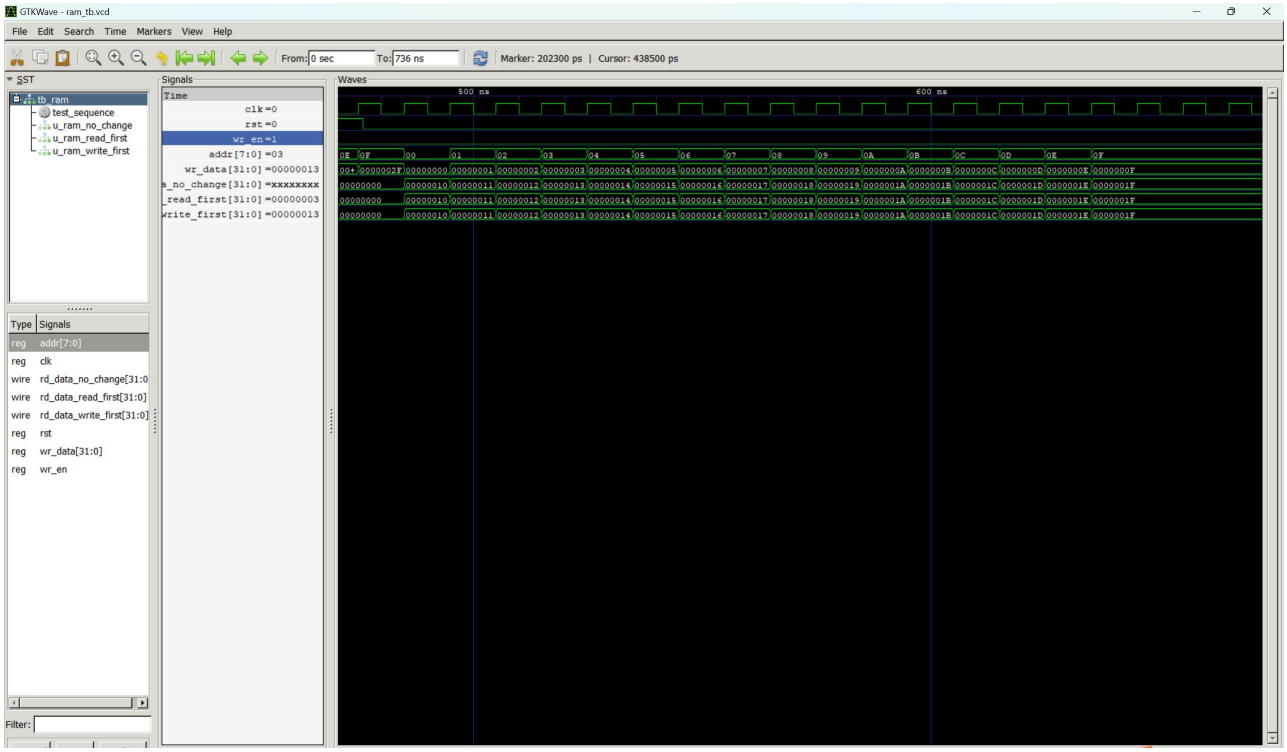
装

订

线

rd_data_no_change、rd_data_write_first、rd_data_read_first：被复位为 0。因为此时 rst 信号拉高，根据实验要求，复位时 RAM 输出端口应置为 0。

4、Task4



rd_data_no_change、rd_data_write_first、rd_data_read_first：根据地址输出 Task2 中对应内存赋值后的值。此时 rst 信号拉低，wr_en 信号也拉低，三个模式的 RAM 都从内

实验名称：___石头剪刀布___ 姓名：__ 学号：

存中读取数据并输出。

四、讨论与心得

通过本次实验，我对 Verilog 硬件描述语言有了更深入的理解和掌握，特别是对时序逻辑和组合逻辑的设计有了更清晰的认识。在实验过程中，遇到问题并通过查阅资料和调试代码来解决问题的过程，让我提高了自己的问题解决能力和动手实践能力。