哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验作业

开课学期：2021 春季

课程名称：计算机组成原理（实验）

实验名称：TileLink 总线协议设计

实验性质：综合设计型

实验学时：4 地点：T2

学生班级：19 级 7 班

学生学号：190110716

学生姓名：朱海峰

作业成绩：

实验与创新实践教育中心制

2021 年 4 月

一、 Master 模块添加注释

```
module master (
    input  wire        clk        , // 时钟信号
    input  wire        rst_n      , // 重置
    input  wire        cpu_wr     , // cpu 发出写指令
    input  wire        cpu_rd     , // cpu 发出读指令
    input  wire [3:0]  cpu_byte   , // 写指令指定的字节
    input  wire [3:0]  cpu_addr   , // 读写的字地址
    input  wire [31:0] cpu_wdata  , // 写数据
    output wire        cpu_rdata_v, // cpu 读到数据是否有效
    output wire [31:0] cpu_rdata  , // cpu 读到数据
    input  wire        a_ready    , // a 通道准备好
    output reg         a_valid    , // a 通道数据有效
    output reg  [3:0]  a_opcode   , // a 通道操作码
    output reg  [3:0]  a_mask     , // a 通道字节掩码
    output reg  [3:0]  a_address  , // a 通道读写的字地址
    output reg  [31:0] a_data     , // a 通道数据
    output reg         d_ready    , // d 通道准备好
    input  wire        d_valid    , // d 通道数据有效
    input  wire [3:0]  d_opcode   , // d 通道操作码
    input  wire [31:0] d_data     , // d 通道数据
    output reg         trans_over   // 是否传送完
);

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n)               d_ready <= 1'b0;
    else if (cpu_wr | cpu_rd) d_ready <= 1'b1;    // 需要读写时，d 通道数据有效
end

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n)               a_valid <= 1'b0;
    else if (cpu_wr | cpu_rd) a_valid <= 1'b1;    // 需要读写时，a 通道数据有效
    else                      a_valid <= 1'b0;
end

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n)                    a_opcode <= 4'h0;
    else if (cpu_wr & (&cpu_byte)) a_opcode <= 4'h0;    //PutFullData,完整写入 32 位数
据,cpu_byte4 位为 1
    else if (cpu_wr)               a_opcode <= 4'h1;    //PutPartialData,按字节写入数
据
    else if (cpu_rd)               a_opcode <= 4'h4;    //Get,读出某个字节数据
    else                           a_opcode <= 4'h0;
end

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n)               a_mask <= 4'h0;
    else if (cpu_wr | cpu_rd) a_mask <= cpu_byte;   // 需要读写时，传递字节掩码
    else                      a_mask <= 4'h0;
end

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n)               a_address <= 4'h0;
    else if (cpu_wr | cpu_rd) a_address <= cpu_addr;// 需要读写时，传递字地址
    else                      a_address <= 4'h0;
end

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n)      a_data <= 32'h0;
    else if (cpu_wr) a_data <= cpu_wdata;           // 需要写时，传递写数据
```

```
    else             a_data <= 32'h0;
end

reg rd_period;
reg trans_over_ff;  // 上一时刻的 transover

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n) trans_over_ff <= 1'b0;
    else        trans_over_ff <= trans_over;    // 记录上一时刻的 transover
end
// 检测上升沿
wire trans_over_pos = trans_over & ~trans_over_ff;

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n)              rd_period <= 1'b0;
    else if (trans_over_pos) rd_period <= 1'b0; // transover 上升，则读数据时段结束
    else if (cpu_rd)         rd_period <= 1'b1; // 读数据时段开始
end

assign cpu_rdata_v = rd_period & d_valid;   // d 通道数据有效，并且已经到了能读到数据的时间段，则 cpu 读到的数据有效

assign cpu_rdata = d_data;                  // cpu 读到的数据

always @ (posedge clk or negedge rst_n) begin
    if (~rst_n)                trans_over <= 1'b1; // 重置，传送结束，即没有在传送
    else if (a_ready & a_valid) trans_over <= 1'b0; // a 通道开始传送数据，传送未结束
    else if (d_ready & d_valid) trans_over <= 1'b1; // d 通道开始传送数据，传送完成
end

endmodule
```

二、 Slave 模块的实现

slave 模块的输入输出如下，左边为输入，右边为输出：

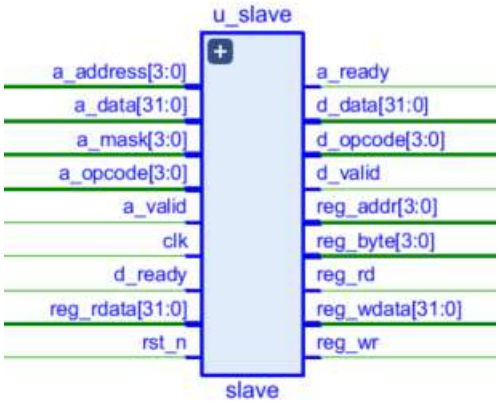

Figure 1

slave 模块的算法流程图：

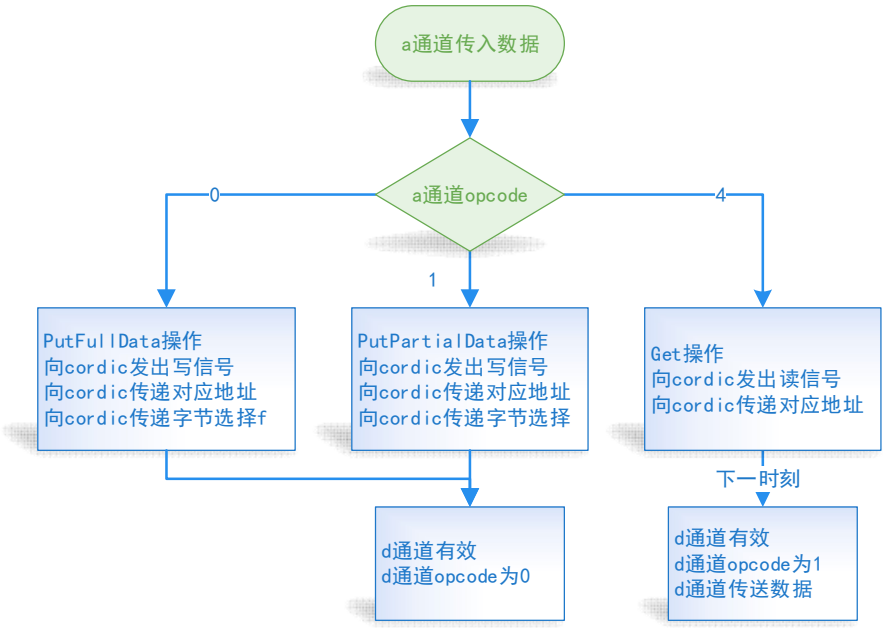

Figure 2

三、 调试报告

（自己写的 testbench 核心代码，仿真截图及时序分析。根据测试的完备性评分）

仿真核心代码：

```
clk <= 1;
addr <= 0;
byte <= 'hf;
wdata <= 'h1_0001;
rst_n <= 1;
rd <= 0;
wr <= 0;

#2  // 完成重置
rst_n <= 0;
#2
rst_n <= 1;
#6  // 对地址 1 进行 PutFullData
wr <= 1;
addr <= 1;
wdata <= 'ha;
#2
wr <= 0;
addr <= 0;

#10 // 对地址 0 进行 PutPartialData
wr <= 1;
byte <= 'b0011;
wdata <= 'h001;
#2
wr <= 0;

#100// 计算完成后，对地址 2Get，读出数据
rd <= 1;
addr <= 0;
#2
rd <= 0;
```

1. 对地址 1 进行 PutFullData 操作，写入数据 ’ha，如图 3。

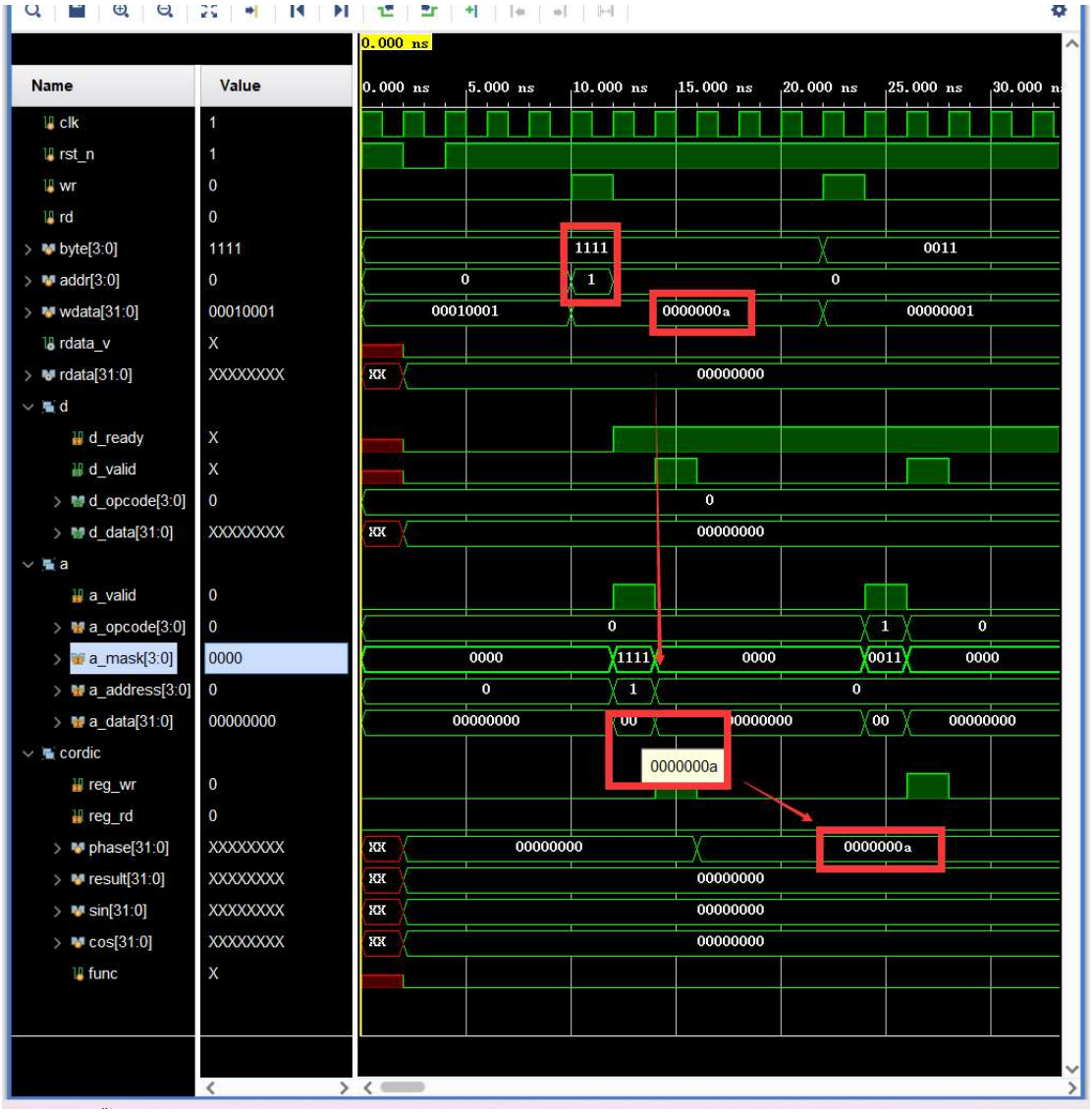

Figure 3

2.对地址 0 进行 PutPartialData，数据为 ’h1，选择后面两字节有效，功能选择 ’b0(sin)，并触发 start，如图 4。

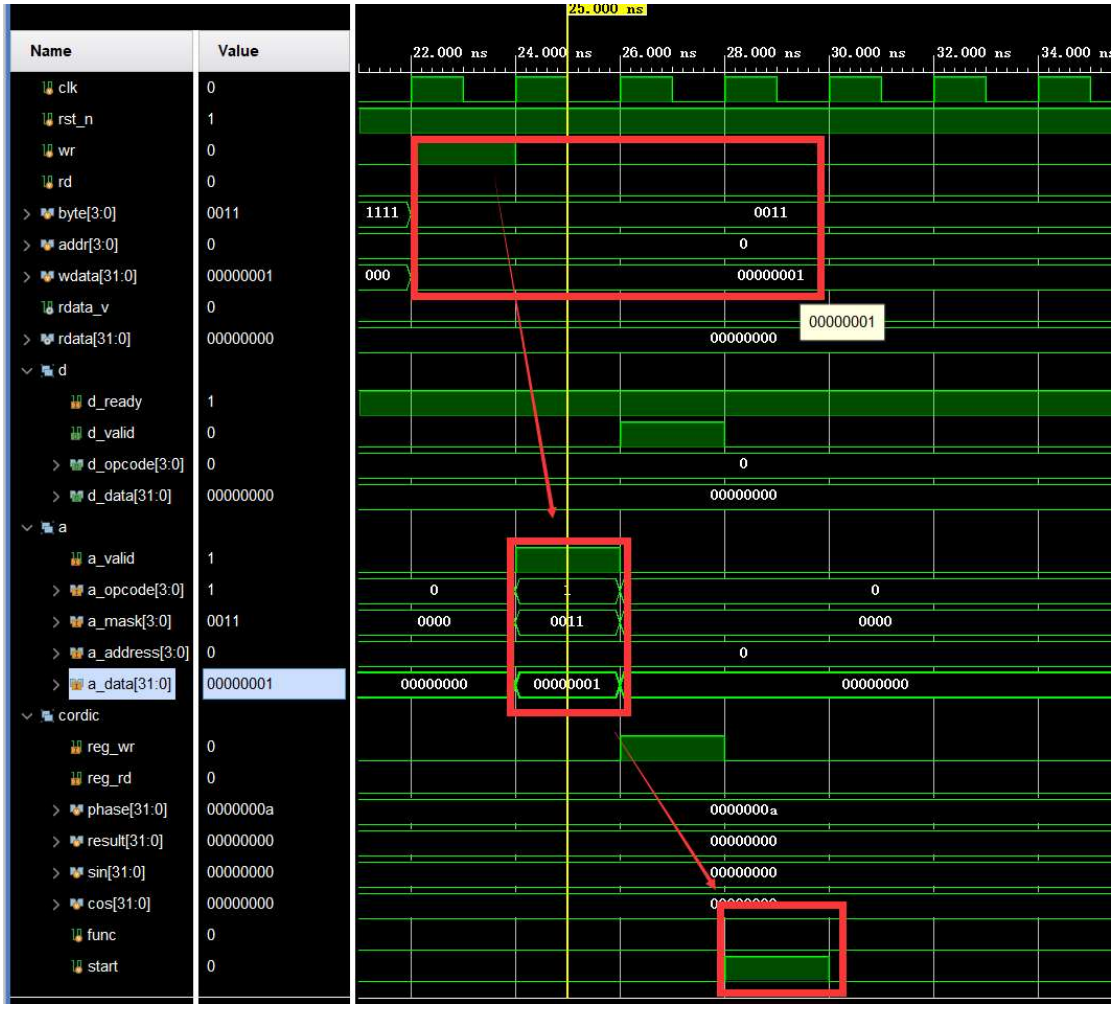

Figure 4

3. 对地址 2 进行 Get 操作，读出计算结果。

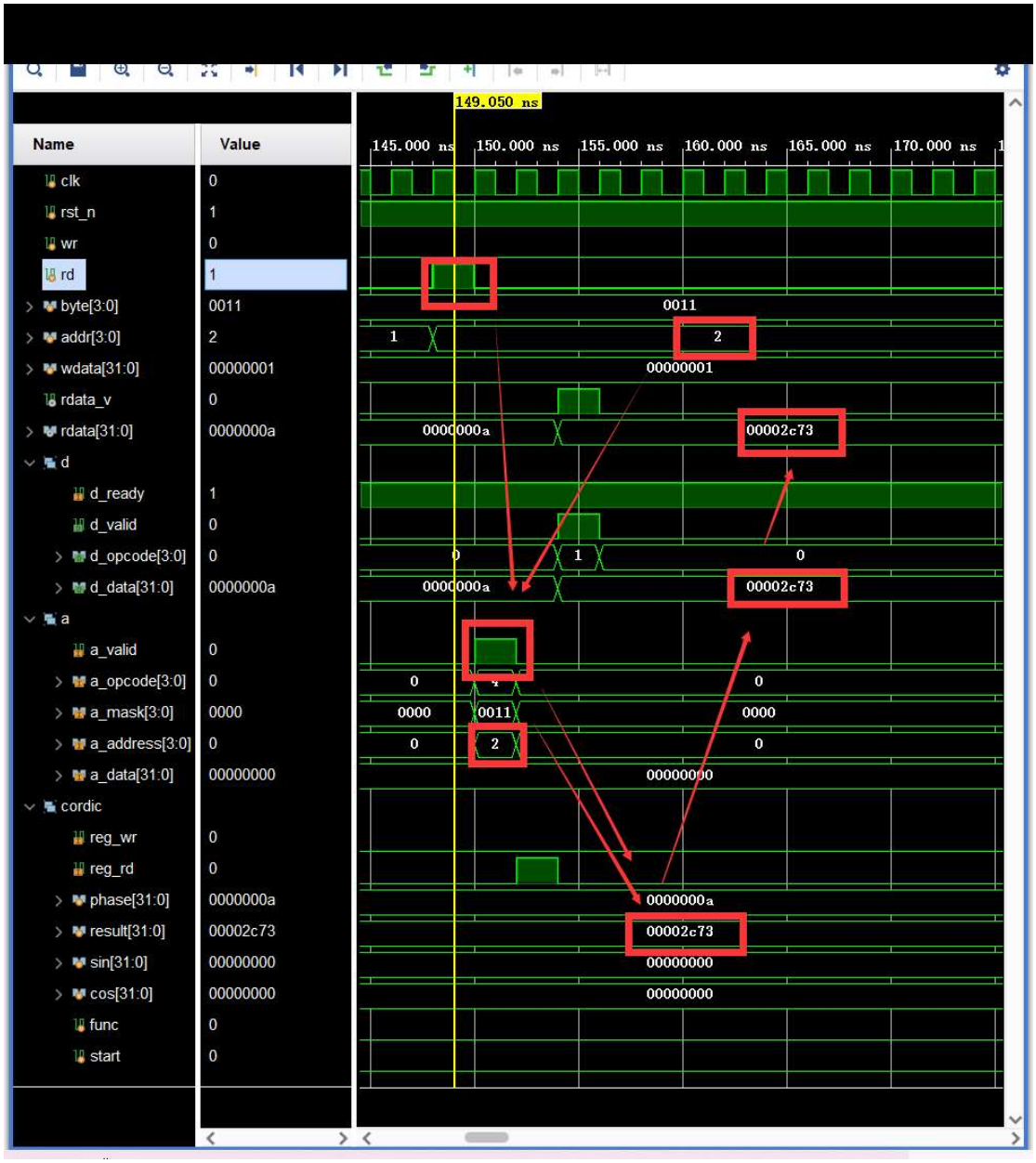

Figure 5