

（深圳）

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

|  |
| --- |
| 1. **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 |
| 1. **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 |
| 1. **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  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  3. 对地址2进行Get操作，读出计算结果。    Figure |