

（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： T2612

学生班级： 1901105

学生学号： 190110509

学生姓名： 王铭

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 ,  input wire rst\_n ,  input wire cpu\_wr ,//cpu写使能  input wire cpu\_rd ,//cpu读使能  input wire [3:0] cpu\_byte ,//选择写入的数据的有效部分  input wire [3:0] cpu\_addr ,//写入的地址  input wire [31:0] cpu\_wdata ,//写入的数据  output wire cpu\_rdata\_v,//读数据有效的标志  output wire [31:0] cpu\_rdata ,//读出的数据  input wire a\_ready ,//a通道可以进行传输  output reg a\_valid , //a通道传输信号有效  output reg [3:0] a\_opcode ,//进行的操作  output reg [3:0] a\_mask ,//选择写入数据的有效部分  output reg [3:0] a\_address ,//写入数据的地址  output reg [31:0] a\_data ,//写入的数据  output reg d\_ready ,//d通道可以进行传输  input wire d\_valid ,//d通道传输的数据有效  input wire [3:0] d\_opcode ,//d信号给的反馈  input wire [31:0] d\_data ,//读出的数据  output reg trans\_over //读/写是否结束  );  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;//读或写使能时，d通道可以进行传输  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; //读或写使能有效时，a通道信号有效  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;//部分写数据  else if (cpu\_wr) a\_opcode <= 4'h1;//全部写数据  else if (cpu\_rd) a\_opcode <= 4'h4;//读数据  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;//选择数据有效字节位  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;//选择写或读数据的地址  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;//写使能有效时，a传输的数据为 cpu\_wdata  else a\_data <= 32'h0;  end  reg rd\_period; //表示是否正在读阶段，为1表示在读  reg trans\_over\_ff;//便于采集读/写过程结束信号的上升沿  always @ (posedge clk or negedge rst\_n) begin  if (~rst\_n) trans\_over\_ff <= 1'b0;  else trans\_over\_ff <= trans\_over;//较trans\_over滞后一个时钟，使得trans\_over\_pos能采trans\_over上升沿时也置1.  end  wire trans\_over\_pos = trans\_over & ~trans\_over\_ff;//当刚好结束读/写时的第一个时钟周期内有效。  always @ (posedge clk or negedge rst\_n) begin  if (~rst\_n) rd\_period <= 1'b0;  else if (trans\_over\_pos) rd\_period <= 1'b0;//读取过程结束，置0  else if (cpu\_rd) rd\_period <= 1'b1;//读取过程未结束，置1  end  assign cpu\_rdata\_v = rd\_period & d\_valid; //正在读且读出数据有效表示读完，给出读出信号有效的反馈  assign cpu\_rdata = d\_data;  always @ (posedge clk or negedge rst\_n) begin  if (~rst\_n) trans\_over <= 1'b1;  else if (a\_ready & a\_valid) trans\_over <= 1'b0;//开始写/读操作  else if (d\_ready & d\_valid) trans\_over <= 1'b1;//读或写操作结束  end  endmodule |
| 1. **Slave模块的实现** |
| **Slave模块的实现：**  在复位信号无效的条件下，reg\_addr,reg\_byte,reg\_wdata,d\_data赋值成传入slave的a\_address,a\_mask,a\_data,reg\_rdata即可。  在复位信号无效的条件下，根据a\_valid信号以及opcode的值确定reg\_wr,reg\_rd是否置1。  d\_valid信号在写操作时接受到a\_valid和a\_opcode的有效信号时置1.  读操作时，设置计数器rd\_cnt，接受到读信号时置为2’b01,此时钟周期向cordic模块传递地址，第二周期置为2’b10,此时在cordic模块中读取数据,根据cnt是否是2’b10确定d\_valid是否置1.  d\_opcode与d\_valid同步变化，若进行了读操作，d\_opcode应置1，否则置0.  **部分关键代码：**  always@(posedge clk or negedge rst\_n)  begin  if(~rst\_n) reg\_wr <= 1'b0;  else if(a\_valid && (a\_opcode == 4'b0000 || a\_opcode == 4'b0001)) reg\_wr <= 1'b1;//根据opcode为0/1置reg\_wr为1  else reg\_wr <= 1'b0;  end  always@(posedge clk or negedge rst\_n)  begin  if(~rst\_n) reg\_rd <= 1'b0;  else if(a\_valid && a\_opcode == 4'b0100) reg\_rd <= 1'b1;//根据a\_opcode为4置reg\_rd为1  else reg\_rd <= 1'b0;  end  reg [1:0]rd\_cnt;//控制读过程  always@(posedge clk or negedge rst\_n)  begin  if(~rst\_n) rd\_cnt <= 'b0;  else if(a\_valid && a\_opcode == 'b0100) rd\_cnt <= 'b01;  else if(rd\_cnt == 'b01) rd\_cnt <= 'b10;  else rd\_cnt <= 'b0;  end  always@(posedge clk or negedge rst\_n)  begin  if(~rst\_n) d\_valid <= 1'b0;  else if(a\_valid && (a\_opcode == 'b0001 || a\_opcode == 'b0000)) d\_valid <= 1'b1;  else if(rd\_cnt == 'b10) d\_valid <= 1'b1;//读需要三个周期才能输出信号有效  else d\_valid <= 1'b0;  end  always@(posedge clk or negedge rst\_n)  begin  if(~rst\_n) d\_opcode <= 4'b0000;  else if(a\_valid && (a\_opcode == 4'b0001 || a\_opcode == 'b0000)) d\_opcode <= 4'b0000;//写操作置0  else if(rd\_cnt == 'b10) d\_opcode <= 'b0001;//读操作置1  else d\_opcode <= 4'b0000;  end |
| 1. **调试报告** |
| **仿真核心代码：**  always #5 clk = ~clk; //时钟周期为10  initial  begin  #25 rst\_n = 1;  //全写0x00ab\_00ab并读  #5 begin byte = 4'b1111; wdata = 32'h00ab\_00ab; addr=4'b0001; wr = 1; end  #10 wr = 0;  #50 begin byte = 4'b1111; addr = 4'b0001;rd = 1;end  #10 begin rd = 0;end  //写0x0000\_0303后两个字节并读  #50 begin wdata = 32'h0000\_0303; addr = 4'b0001; byte = 4'b0011; wr = 1; end  #10 wr = 0;  #50 begin addr = 4'b0001;byte = 4'b1111; rd = 1; end  #10 rd = 0;  #50 rst\_n = 0;  #10 rst\_n = 1;  //sin 是0 cos是1 phase是0x1 result为0x2  //sin,0x1000a,0xfc1b  # 50 begin wdata = 32'h0001\_000a;addr = 4'b0001;byte = 4'b1111;wr = 1; end  # 10 wr = 0;  # 50 begin wdata = (1 | (0 << 8));addr = 4'b0000;byte = 4'b1111;wr = 1;end  # 10 wr = 0;  # 200 begin addr = 4'b0010;byte = 4'b1111;rd = 1;end  # 10 rd = 0;  //cos,0x10014,0xffffa872  # 50 begin wdata = 32'h0001\_0014;addr = 4'b0001;byte = 4'b1111;wr = 1; end  # 10 wr = 0;  # 50 begin wdata = (1 | (1 << 8));addr = 4'b0000;byte = 4'b1111;wr = 1;end  # 10 wr = 0;  # 200 begin addr = 4'b0010;byte = 4'b1111;rd = 1;end  # 10 rd = 0;  end    时序分析：30ns进行第一次写操作，向addr=0x1全写数据0x00ab\_00ab，第一个时钟周期a\_valid和d\_ready置1，第二个时钟周期a\_ready和d\_valid置1，写操作结束,d\_opcode为0.    时序分析：90ns时进行对0x1进行读操作，在第二个周期向cordic模块传输地址和读使能，第三个周期读出数据，0x00ab\_00ab，并置d\_valid和rdata\_v为1，表示读出的数据有效，d\_opcode置1。    时序分析：150ns时对0x1进行部分写操作，byte为4’b0011表示写入0x0000\_0303的后两个字节即0303。第二个周期d\_valid置1，d\_opcode为0    时序分析：210ns对0x1读取数据，第二个周期时向cordic传递地址，第三个周期读取数据为0x00ab\_0303，置d\_valid和rdata\_v为1表示读出数据有效，d\_opcode给出高电平反馈信号.    测试复位功能，d\_data恢复0    时序分析：330ns对phase寄存器进行写操作，写入0x0001\_000a，390ns时向config寄存器写入1表示计算sin，均在第二个周期置d\_valid信号为1,d\_opcode为0    时序分析：600ns时对result寄存器进行读操作，第二个周期向cordic模块传地址，第三个周期读数据，置d\_valid和rdata\_v为1，d\_opcode为1，读出计算的sin函数结果为0x0000\_fc1b。    时序分析：660ns对phase寄存器进行写操作，写入0x0001\_0014，720ns时向config寄存器写入0x101表示计算cos，均在第二个周期置d\_valid信号为1，d\_opcode为0    时序分析：930ns对result寄存器进行读操作，第二个周期向cordic模块传地址，第三个周期读数据，置d\_valid和rdata\_v为1，d\_opcode为1，读出计算的cos函数结果为0xffff\_a872。 |