



System Verilog Lab1

TA: 孙佳磊

Mail: sjl_519021910940@sjtu.edu.cn

TA: 谢琳

Mail: lynnxie@sjtu.edu.cn

2024年10月31日



上海交通大学

SHANGHAI JIAO TONG UNIVERSITY



一、实验准备



▪ Questasim仿真器

QuestaSim其实就是Modelsim的扩展版，增加了System Verilog仿真的功能，除此之外，几乎没区别。我们为同学们提供了下载通道：

Pan: <https://pan.sjtu.edu.cn/web/share/36ae858274b47682c927c4620ccd3e40>

▪ 入门教程

https://www.bilibili.com/video/BV1t4411q76K/?spm_id_from=333.337.search-card.all.click&vd_source=0050f7f1fcda9fca97972668f947b778

一、实验准备



■ 实验材料

登录canvas, 在  ▼ SystemVerilog_2024_lab1 单元下下载以下文件

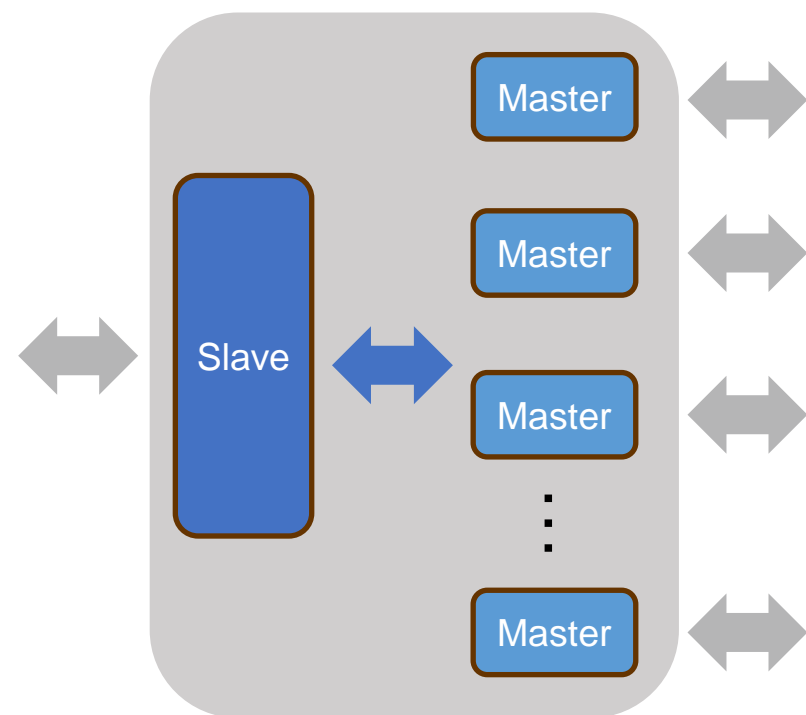
(1) 实验PPT (本文档)  SystemVerilog_2024_lab1.pptx

(2) sv文件模板  lab1_svfile

(3) quiz文件  lab1_quizpart

(4) 参考文档  doc_helps

二、实验内容



总线桥 (bus bridge) 是计算机系统中用于连接两种不同总线的硬件组件。

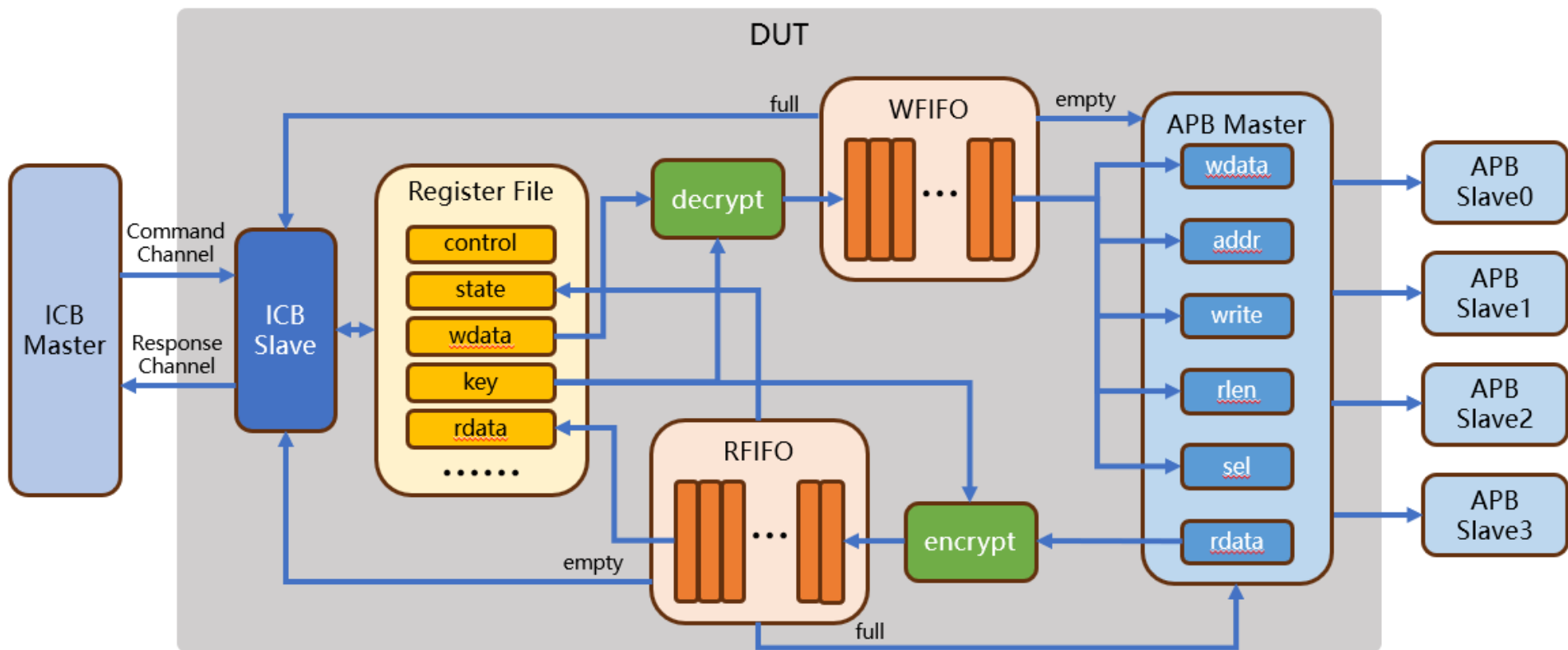
总线桥的主要作用是实现不同总线之间的**数据传输**和**协议转换**，以确保多个设备之间的兼容性和通信。

在SoC系统中，处理器常会与多个设备连接，因而总线桥常为**一主多从**或多主多从的实现形式。

在某些涉及信息安全的场景下，主设备传输的数据可能是具有特定格式的加密数据，当从设备不具备解密功能时，则需额外的硬件电路实现加解密。

本次实验需要实现一个具备**加解密功能的一主四从总线桥**

二、实验内容

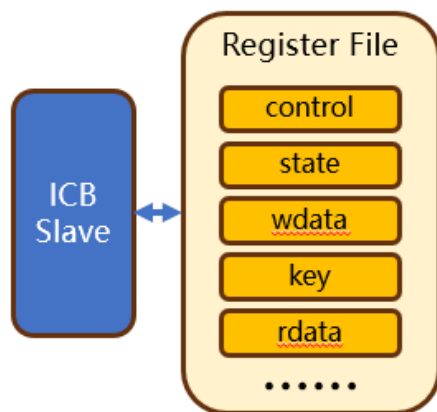


要求:

- (1) ICB协议正确实现
- (3) 数据加解密正确实现

- (2) APB协议正确实现
- (4) 仲裁正确实现

二、实验内容



■ ICB从机模块

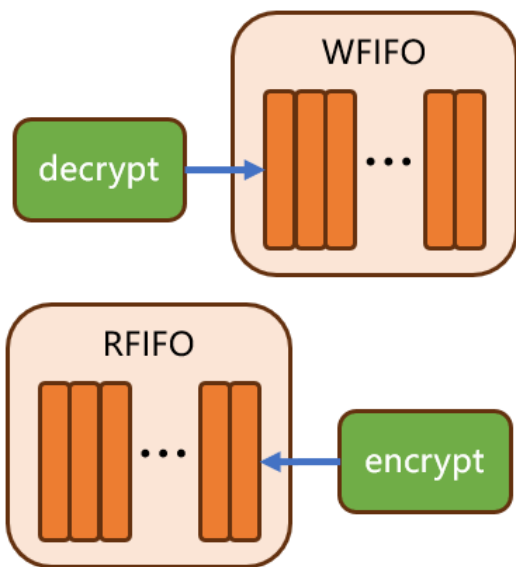
1. 实现满足标准ICB时序的从机端口，传输数据位宽为64bit，地址位宽为32bit。标准ICB时序说明见 doc_helps/ICB总线时序说明。 **(必做)**

2. 至少维护图中列出的寄存器，寄存器说明如下表 **(必做)**

名称	基地址	偏移量	权限	说明
CONTROL	0x20000000	0x00	可读可写	APB主机模块使能信号
STATE	0x20000000	0x04	只读	指示WFIFO、RFIFO的空满状态和APB Master当前状态（读、写、空闲……）
WDATA	0x20000000	0x08	可读可写 or 只写	需要传输给APB端从机的数据或传输给APB主机模块的控制信息
RDATA	0x20000000	0x0C	只读	从APB端从机读取的数据
KEY	0x20000000	0x10	可读可写 or 只写	密钥，用于加密及解密 (该寄存器可选择是否实现)

Note: 权限为“可读可写 or 只写”的寄存器意为该寄存器至少要可写

二、实验内容



■ 加密(encrypt)解密(decrypt)模块模块设计

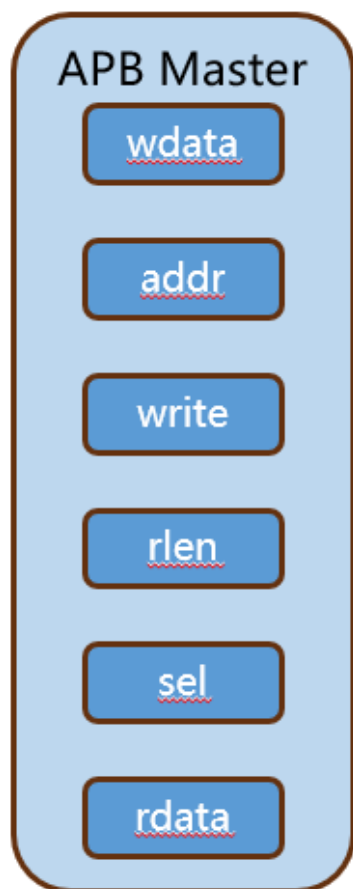
1. 使用寄存器key配置的密钥，在数据写入FIFO前完成加密及解密。 **(必做)**
2. 本次实验使用数据与密钥**异或**的方式进行加密、解密。 **(必做)**

Note: 常见的对称加解密算法包括DES、RC4、AES等。后续Project会要求同学们将加解密算法更换为DES算法，同学们可提前了解并实现 **(选做)**

■ FIFO模块设计

1. FIFO基本功能设计，能够由上游信号（ICB从机写入WDATA or APB主机接收到返回的读数据）控制数据写入及由下游信号控制信号决定数据的读出。FIFO的深度和数据位宽由同学们自行定义。 **(必做)**
2. 同学们可在不影响FIFO功能下自由探索扩展FIFO的实现方式，如读写指针使用格雷码变换、读写时钟异步等。 **(选做)**

二、实验内容

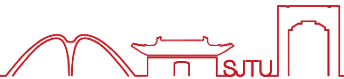


■ APB主机模块设计

1. 实现满足标准APB3时序的主机端口，传输数据位宽为32bit，地址位宽为32bit。标准APB3时序说明见 doc_helps/APB总线时序说明。 **(必做)**
2. 根据下表数据包格式对从WFIFO中获取的数据包进行解码，并对驱动APB Master执行相应操作。 **(必做)**
3. APB从机返回的读数据，在[63:32]位补零后，送到加密模块加密，再送入RFIFO。 **(必做)**

[31:8]	[7:2]	[1]	[0]
addr	select	cmd	flag
写到APB的地址，地址为0x20000000	000001: APB0 000010: APB1 000100: APB2 001000: APB3	0: 读 1: 写	0: 控制 1: 数据

二、实验内容



- 实验提供top.sv及interface.sv
- 可进一步丰富interface内容
- 不允许修改 top 及对应interface的端口名称
- Interface端口不可增减

```
6  /* don't change the signal */
7  module dut_top(
8      //input bus
9      icb_bus.slave icb_bus,
10
11      //output bus
12      apb_bus.master apb_bus_0,
13      apb_bus.master apb_bus_1,
14      apb_bus.master apb_bus_2,
15      apb_bus.master apb_bus_3
16
17  );
18
```

```
/*
This is only the basic interface, you may change it by your own.
But don't change this signal discription.
*/
interface icb_bus(input logic clk,input logic rst_n);
    //command channel
    logic icb_cmd_valid;
    logic icb_cmd_ready;
    logic [31:0] icb_cmd_addr;
    logic icb_cmd_read;
    logic [63:0] icb_cmd_wdata;
    logic [7:0] icb_cmd_wmask;

    //response channel
    logic icb_rsp_valid;
    logic icb_rsp_ready;
    logic [63:0] icb_rsp_rdata;
    logic icb_rsp_err;
endinterface
```

```
6
7  /*
8  This is only the basic interface, you may change it by your own.
9  But don't change this signal discription.
10 */
11 interface apb_bus(input logic clk,input logic rst_n);
12     logic pwrite;
13     logic psel;
14     logic [31:0] paddr;
15     logic [31:0] pwdata;
16     logic penable;
17
18     logic [31:0] prdata;
19     logic pready;
20 endinterface
```



三、报告与提交



■ 报告要求

实验报告需要包括详细的设计思路描述和时序描述，为了方便阅读和评估，我们强烈建议包括如下部分：

- (1) 模块设计说明
- (2) 基本验证设计说明，附时序图
- (3) 总结，概述完成的内容以及**相关指标**
- (4) quiz问答【详见下一个part】

请注意，实验报告将是判定实验成绩的主要依据，请尽量做到书写规范，思路清晰易懂，切勿互相抄袭，发现抄袭一律不给分。

三、报告与提交



▪ 文件提交

文件提交方式：Canvas

文件提交格式：压缩包

截止日期：11月12日23:59

```
文件名：学号_姓名_SV202XLAB_0X  
示例   ：124039910123_小明_SV2024LAB_01  
  
|----->design_files  
|           设计文件夹：存放设计相关的SystemVerilog、verilog文件  
|----->testbench  
|           测试平台文件夹：存放测试平台搭建相关文件  
|----->report  
|           报告文件夹：存放编写的报告  
|----->README（可选）  
|           若文件结构迫不得已需要更改或出现其他需要说明的情况，可在该文件中说明
```

请注意，由于提交的文件需要被运行检查，这是一个繁重的工作，所以我们同样强烈建议严格按照如上文件格式提交，以方便我们运行你的设计。

X、Additional Support



▪ 异或加解密算法介绍

异或算法是极为简单的对称加密算法。将数据与密钥按位异或，即可实现对数据的加密，将加密后数据与相同的密钥再次按位异或，即可实现对数据的解密。

$$X \oplus KEY \oplus KEY = X \oplus 0 = X$$

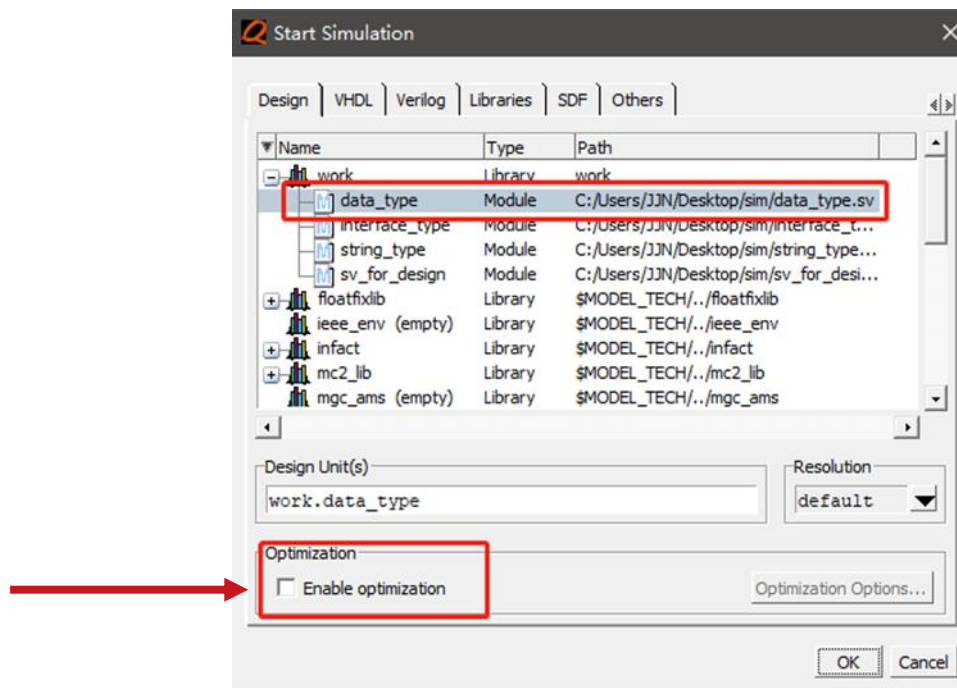
对称加密算法是一种加密方法，在这种方法中，加密和解密使用相同的密钥。它的核心思想是，发送方使用密钥对数据进行加密，而接收方使用相同的密钥对数据进行解密。这种加密方式的最大特点是**密钥对称**，即加密和解密的密钥是相同的。

X、Additional Support



▪ Error QA

取消Enable optimization.

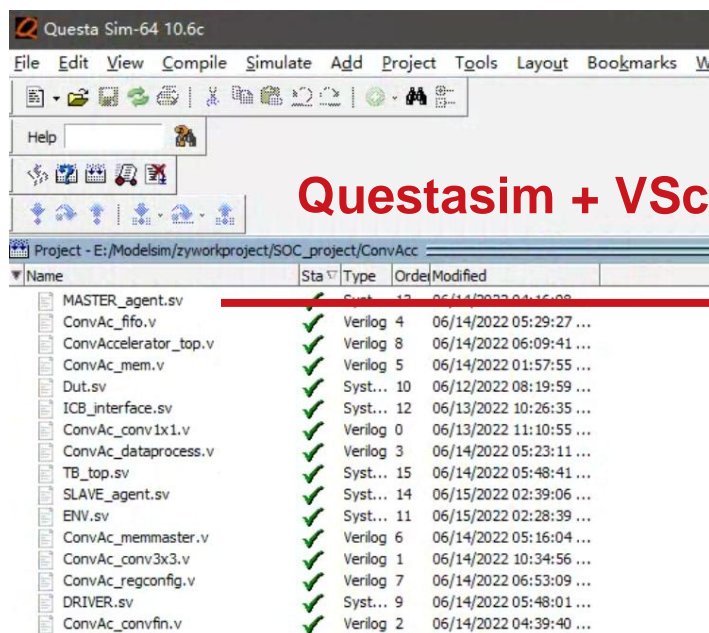


X、Additional Support

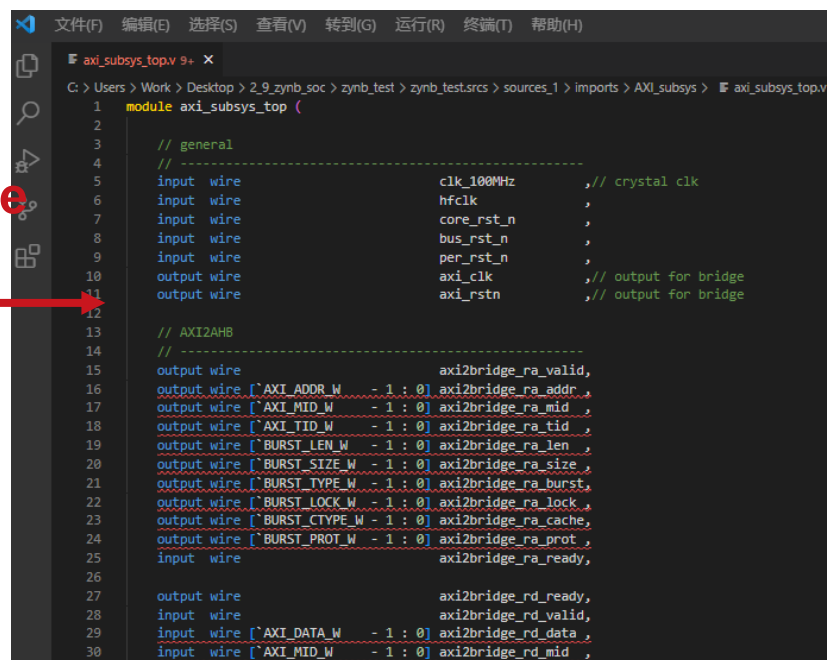


■ 更友好的实验体验——个性化文本编辑器

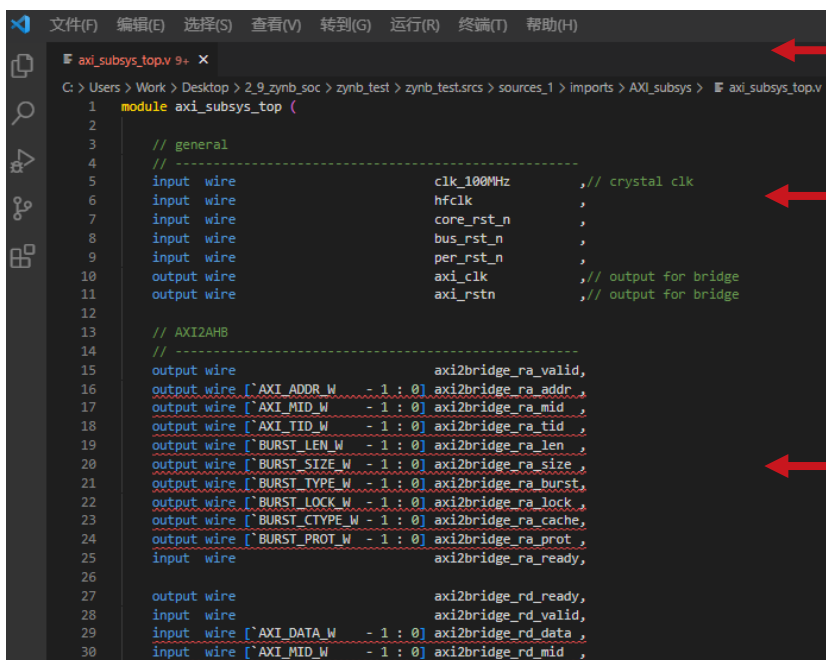
浏览器搜索 “Vscode + modelsim 环境搭建”，有相关教程。



Questasim + VScode



X、Additional Support



```
1 module axi_subsys_top (
2
3     // general
4     // -----
5     input wire          clk_100MHz, // crystal clk
6     input wire          hfclk,
7     input wire          core_rst_n,
8     input wire          bus_rst_n,
9     input wire          per_rst_n,
10    output wire          axi_clk, // output for bridge
11    output wire          axi_rstn, // output for bridge
12
13    // AXI2AHB
14    // -----
15    output wire          axi2bridge_ra_valid,
16    output wire [^AXI_ADDR_W - 1 : 0] axi2bridge_ra_addr,
17    output wire [^AXI_MID_W - 1 : 0] axi2bridge_ra_mid,
18    output wire [^AXI_TID_W - 1 : 0] axi2bridge_ra_tid,
19    output wire [^BURST_LEN_W - 1 : 0] axi2bridge_ra_len,
20    output wire [^BURST_SIZE_W - 1 : 0] axi2bridge_ra_size,
21    output wire [^BURST_TYPE_W - 1 : 0] axi2bridge_ra_burst,
22    output wire [^BURST_LOCK_W - 1 : 0] axi2bridge_ra_lock,
23    output wire [^BURST_CTYPE_W - 1 : 0] axi2bridge_ra_cache,
24    output wire [^BURST_PROT_W - 1 : 0] axi2bridge_ra_prot,
25    input wire          axi2bridge_ra_ready,
26
27    output wire          axi2bridge_rd_ready,
28    input wire          axi2bridge_rd_valid,
29    input wire [^AXI_DATA_W - 1 : 0] axi2bridge_rd_data,
30    input wire [^AXI_MID_W - 1 : 0] axi2bridge_rd_mid,
```

VScode代码书写界面

语法高亮

及时更新的基于Questasim
内核的语法检测



System Verilog Lab1

Quiz Part



上海交通大學
SHANGHAI JIAO TONG UNIVERSITY



To Do



- 在Canvas的下载资料中打开quizpart文件夹

 data_type.sv

 diff_arrays.sv

 interface_type.sv

 task_and_function.sv

- 请结合思考指导仿真运行这些文件，文件运行的**仿真波形或输出截图**和**对思考题的解答**请附在实验报告的末尾。

熟悉数据类型



- 运行diff_arrays.sv, 观察输出, 回忆sv数据类型知识
- 运行data_type.sv
 - 有符号数和无符号数
 - 四值逻辑和二值逻辑
 - 枚举类型
 - 结构体

熟悉数据类型



- 将b_signed_vs_unsigned置为1，其余为0；
- 直接编译运行，查看结果。

```
# signed_vs_unsigned process block started  
# byte variable b0 = -128  
# bit vector variable b1 = 128
```

- 思考：如何可以使两者呈现一样的结果？
- 将b_bit_vs_logic置为1，其余为0
- 观察结果，思考：四值逻辑与二值逻辑在SV中的区别和用途，SV如何进行四值逻辑与二值逻辑的转换。

熟悉数据类型



- 将b_enum_type置为1，其余为0
- \$cast()表示强制类型转换，如果失败，则返回0，成功则为1.

```
enum_type process block started
stl value = 0 (int)
stl value = IDLE (string)
stl value = IDLE (string)
stl value = 1 (int)
stl value = START (string)
** Error: int 4 to state_t conversion failure!
    Time: 0 ns  Scope: data_type.enum_type File:
stl value = 4 (int)
stl value = (string)
```

- 思考：如何修改使每次打印对应不同的状态并且不出现error?

熟悉数据类型



- 将b_struct_type置为1，其余为0
 - 仿真查看结果，了解对结构体的不同赋值方式。
 - 将t1赋值语句中的'删去，重新编译，发现error并改正

```
waitm_struct_type == 1); $display(" struct_type pr  
t1 = '48'h10, 32'h1122_3344, 1'b1, 32'h1000};  
$display("t1 data content is %n", t1);
```

- 思考：为什么会出现错误？SV对packed数据和Unpacked数据赋值时有怎样的不同。

Interface类型



- 编译并仿真interface_type.sv
 - 理解接口类型如何定义和例化
 - 了解如何在接口中定义Task并调用



Task与function



- 仿真task_and_function.sv并运行查看打印信息
 - 只将b_function_define置为1，体会function的定义与参数传递操作
 - 只将b_task_define置为1，体会task的定义与参数传递操作
 - 只将b_inout_vs_ref置为1，体会task参数定义时，inout与ref的区别
 - **思考：function与Task的区别**