计算机组成原理
实验2 TileLink总线协议设计

马世禹

HITSZ 实验与创新实践教育中心
Education Center of Experiments and Innovations, HITSZ

实验回顾

- **组合逻辑**

```
reg busy;
always @ (start,end) begin
        if (start)       busy = 1’ b1;
        else if (end) busy = 1’ b0;
        else               busy = busy_ff;
end

wire busy;
assign busy = start ? 1’ b1 : end ? 1’ b0 : busy_ff;
```

组合逻辑一定要有else

实验回顾

- **计数器**

```
reg [3:0] cnt_reg;
reg       busy_ff;
wire end,busy;
assign end = busy_ff & (cnt == 4’h9);
assign busy = start ? 1’b1 : end ? 1’b0 : busy_ff;
always @ (posedge clk or negedge rst_n) begin
      if (~rst_n) busy_ff <= 1’b0;
      else        busy_ff <= busy;
 end
```

```
always @ (posedge clk or negedge rst_n) begin
      if (~rst_n)     cnt_reg <= 4’h0;
      else if (end)   cnt_reg <= 4’h0;
      else if (busy) cnt_reg <= cnt_reg + 4’h1;
end
```

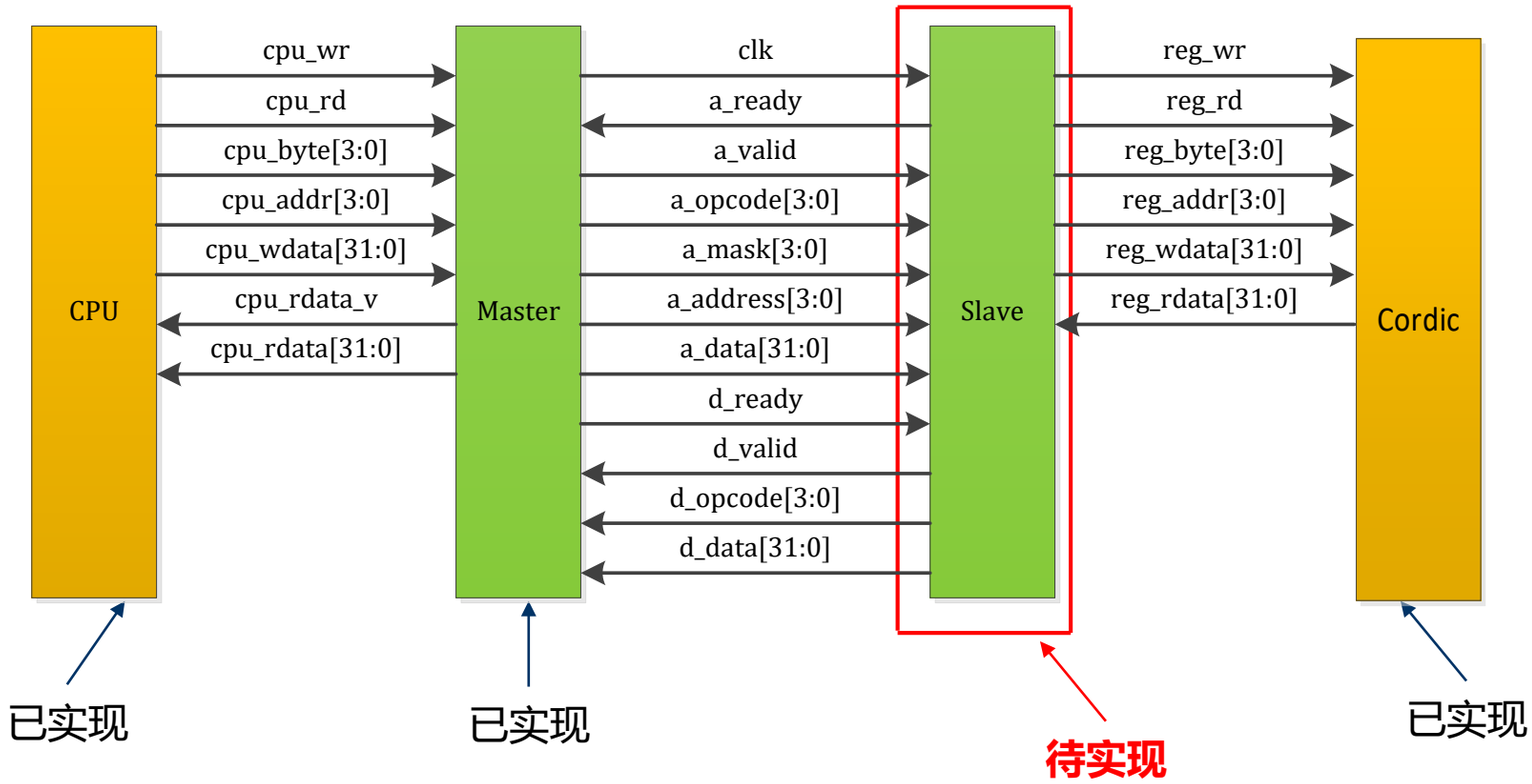
实验目的

- 通过实验TileLink总线协议的设计，理解总线的控制。
- 通过实现TileLink总线协议的通信，理解总线的通信过程。

实验内容

Cordic运算单元，三角函数的加速运算模块，如sin，cos等运算。

通过TileLink总线协议，实现CPU对Cordic运算单元寄存器的访问，继而完成三角函数的加速运算。

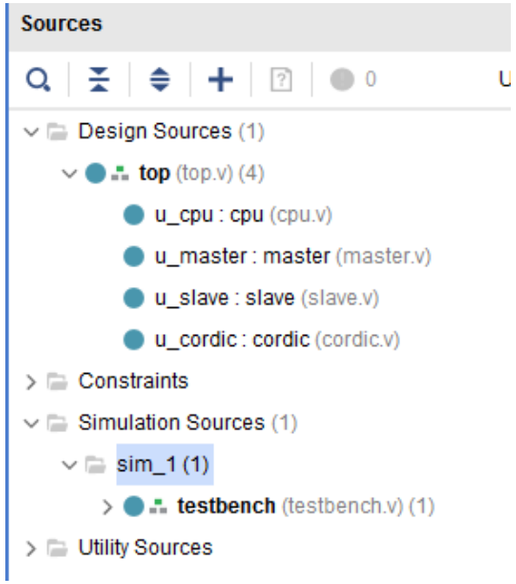


Cordic模块寄存器说明

Register	Address	Bits	Description
config	0x0	31~17	保留位
		16	计算完成标志
		15~9	保留位
		8	选择使能的功能： 0：sin；1：cos
		7~1	保留位
		0	启动cordic模块使能
phase	0x1	31~0	输入的角度
result	0x2	31~0	输出的计算结果

项目框架

top (top.v) —— 顶层模块，模拟输入输出并进行正确性检查
cpu (cpu.v) —— 模拟CPU读写模块
master (master.v) —— tilelink主接口模块
slave (**slave.v**) —— tilelink从接口模块（需要完成）
cordic (cordic.v) —— 三角函数硬件加速模块

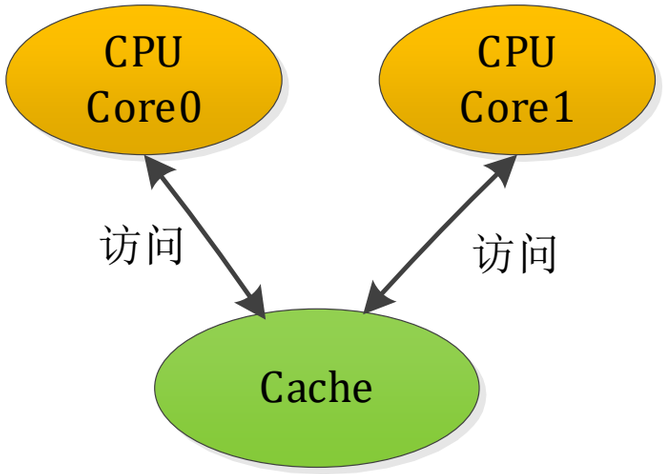


实验原理

- **1、TileLink协议简介**
- TileLink总线协议是伯克利大学定义的一种高速片上总线协议，它诞生的初衷主要是为了定义一种标准的支持**缓存一致性**(Cache Coherence)的协议。

分类

- TileLink无缓存轻量级(Uncached Lightweight, **TL-UL**)
- TileLink无缓存重量级(Uncached Heavyweight, TL-UH)
- TileLink缓存支持级(Cached , TL-C)

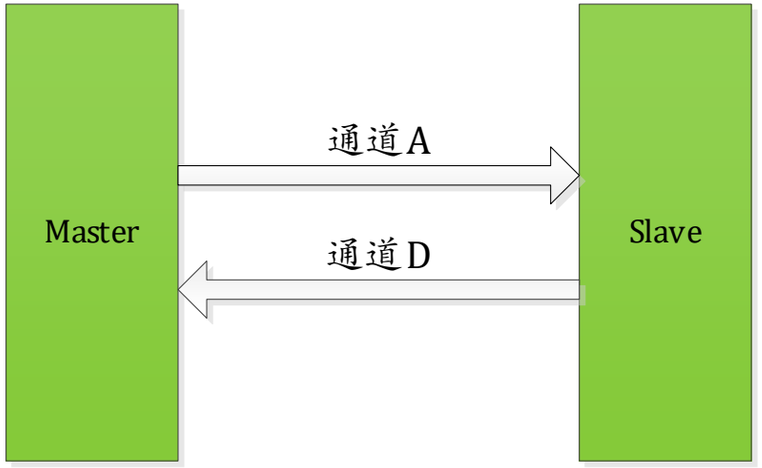


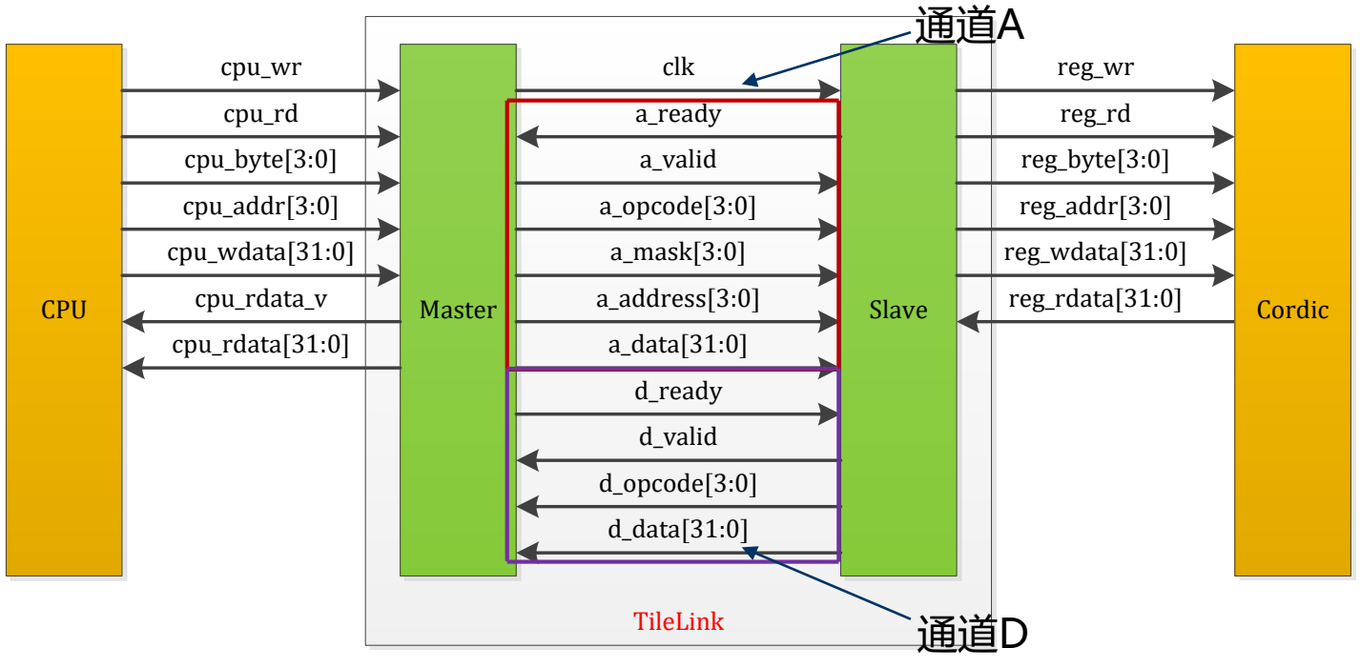
实验原理

- ## 2、TileLink无缓存轻量级(TL-UL)

TL-UL包含Master和Slave两端，通过通道A和通道D实现消息的发送，支持以下两种操作：

- 读(Get)操作：从从设备中读取一定量的数据。
- 写(Put)操作：向从设备中写入一定数目的数据。





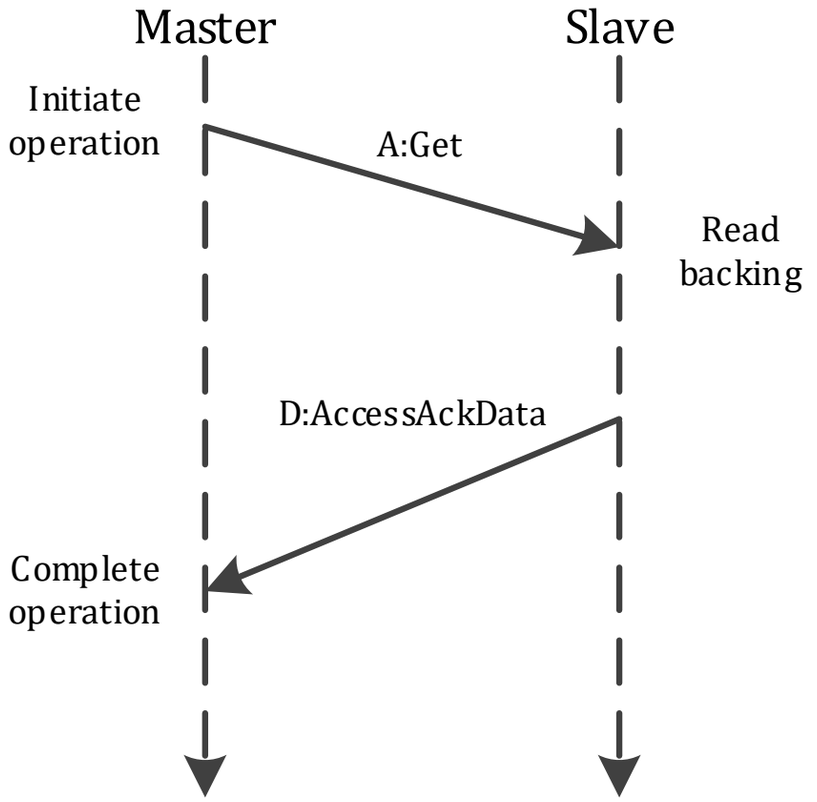
实验原理

- **2、TileLink无缓存轻量级(TL-UL)**

读(Get)操作：

- **读(Get)消息：**用于在读取数据时，访问一块特定的数据存储块。

- **带数据确认(AccessAckData)消息：**是一个向原请求返回数据的确认消息。

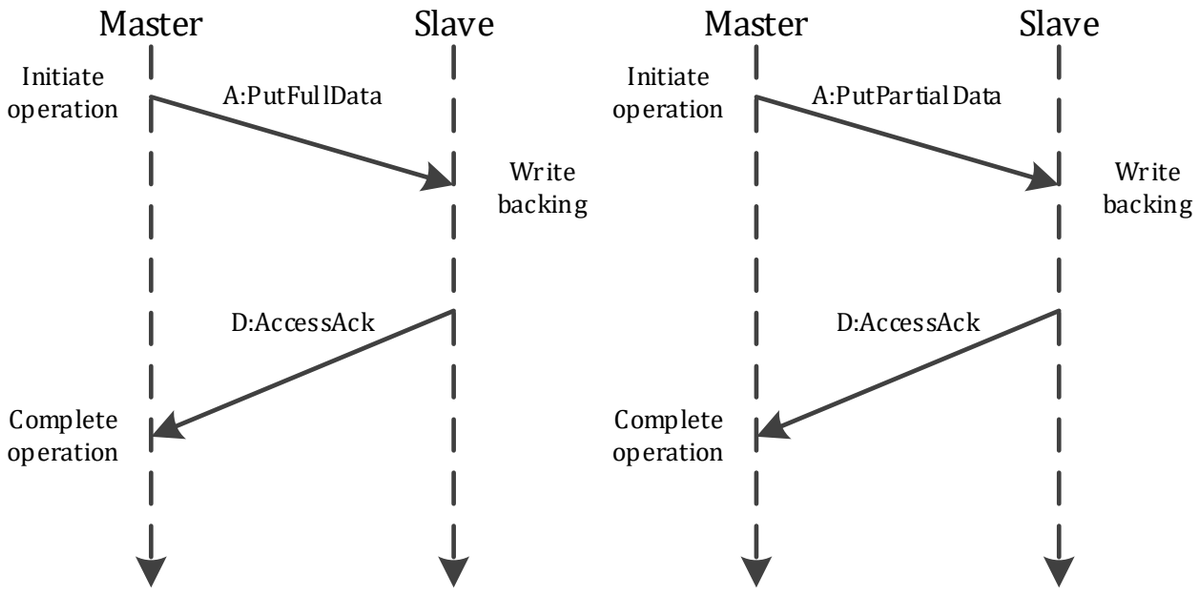


实验原理

• 2、TileLink无缓存轻量级(TL-UL)

写(Put)操作：

- **完整写(PutFullData)消息：**写入一整块数据时发出的消息。
- **部分写(PutPartialData)消息：**写入一整块内部分数据时发出的消息报文。
- **无数据确认(AccessAck)消息**：是一个送往原请求的无数据确认消息。



实验原理

• 2、TileLink无缓存轻量级(TL-UL)

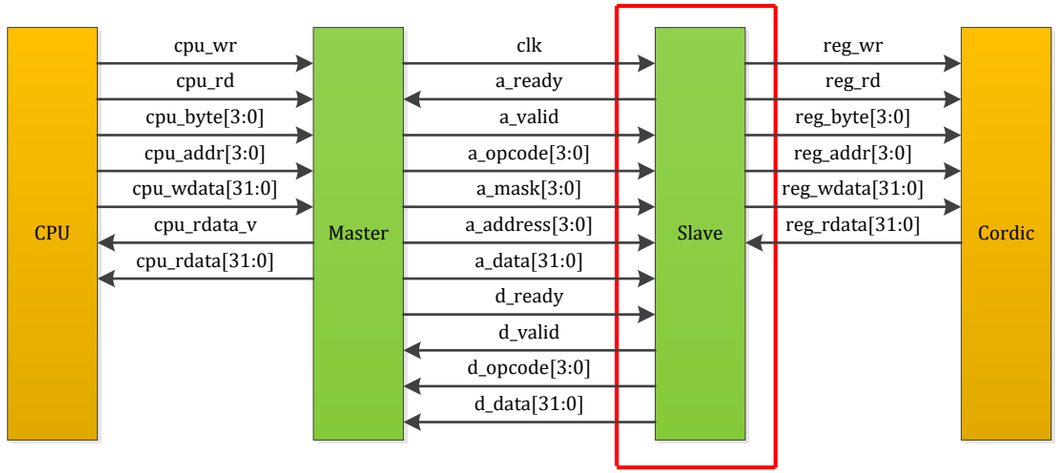
TL-UL支持的5种消息类型：

操作	Message	编码	A	D	响应
Get	Get	4	Y		AccessAckData
	AccessAckData	1		Y	
Put	PutFullData	0	Y		AccessAck
	PutPartialData	1	Y		AccessAck
	AccessAck	0		Y	

实验原理

• 2、TileLink无缓存轻量级(TL-UL)

Slave的接口定义：



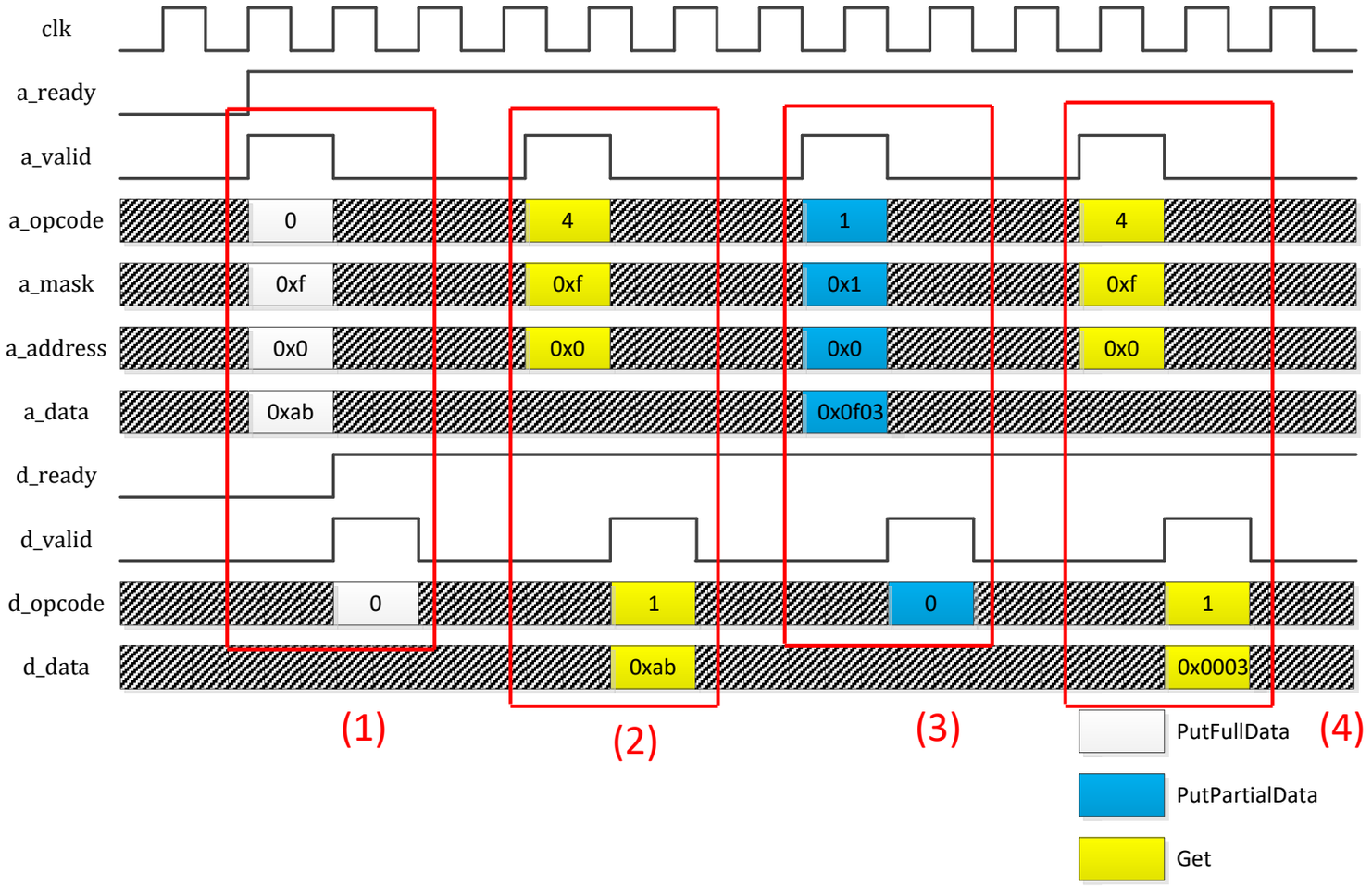
cpu_byte：选择写入的数据哪个字节有效；

信号	通道	宽度	描述
a_ready	A	1	通道数据是否可以被接收
a_valid	A	1	通道携带的数据是否有效
a_opcode	A	3	操作码。识别该通道携带消息的类型
a_mask	A	4	选择消息数据的哪个字节
a_address	A	4	操作的按字节寻址的地址目标
a_data	A	32	可随消息携带的数据
d_ready	D	1	通道数据是否可以被接收
d_valid	D	1	通道携带的数据是否有效
d_opcode	D	3	操作码。识别该通道携带消息的类型
d_data	D	32	可随消息携带的数据
reg_wr	Null	1	Cordic模块寄存器写使能
reg_rd	Null	1	Cordic模块寄存器读使能
reg_byte	Null	4	Cordic模块寄存器读写操作的字节选择
reg_addr	Null	4	Cordic模块寄存器读写地址
reg_wdata	Null	32	Cordic模块寄存器写数据
reg_rdata	Null	32	Cordic模块寄存器读数据

实验原理

- 2、TileLink无缓存轻量级(TL-UL)

Get和Put操作的波形：



(1) PutFullData写数据0xab

(2) Get读回数据0xab

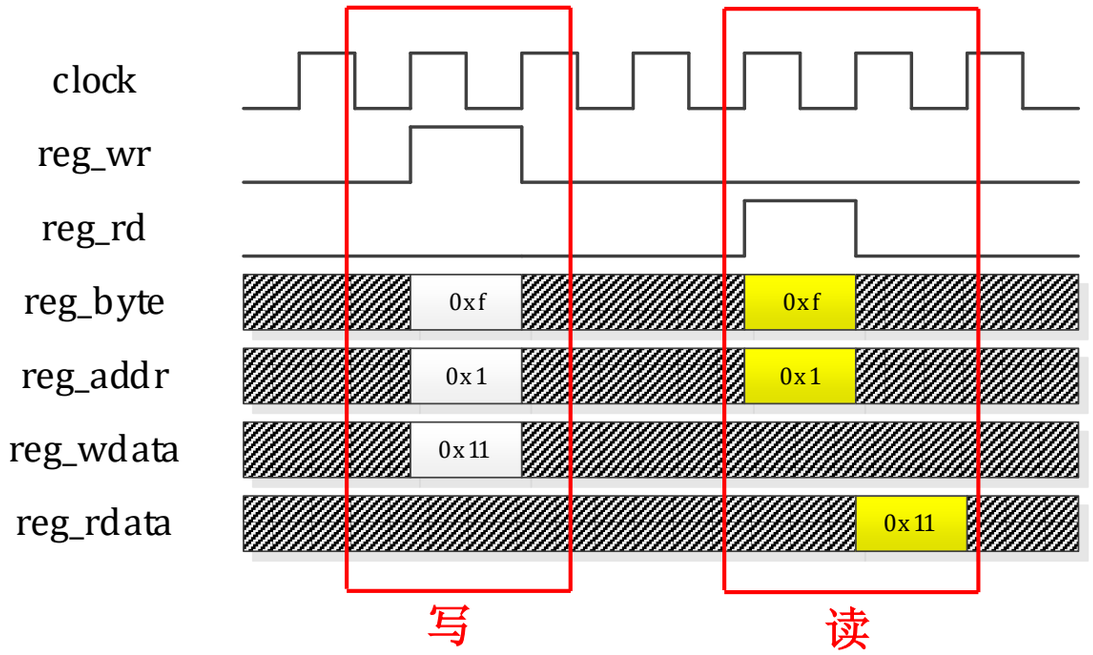
(3) PutPartialData写入0x0f03

(4) Get读回0x0003

实验原理

- 2、TileLink无缓存轻量级(TL-UL)

Cordic模块的读写操作波形：

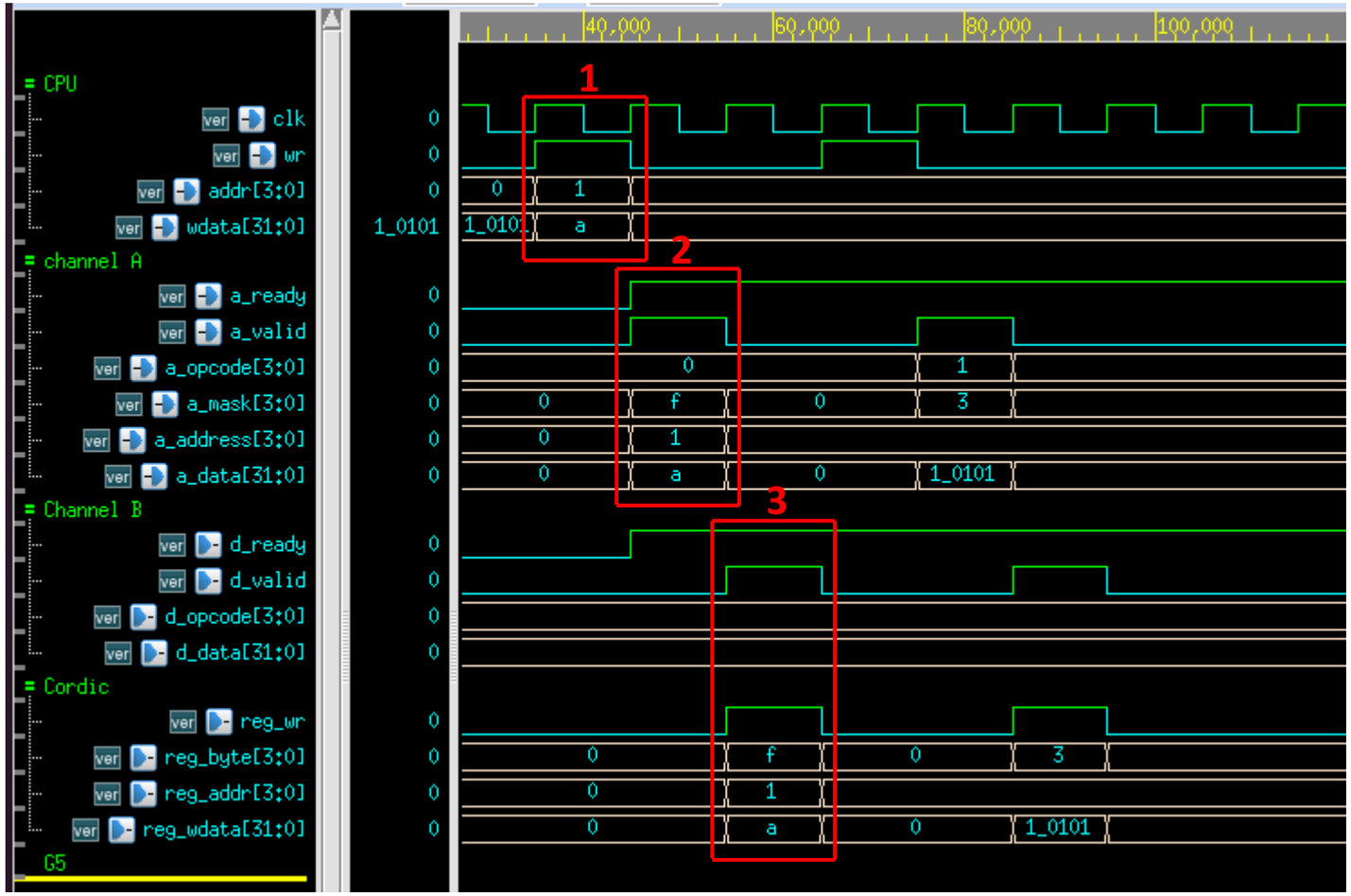


对地址0x1，写入数据0x11；

对地址0x1读取，读回0x11；
注：读数据会在读使能信号的下一个时钟周期才有效；

实验结果

- 实验结果展示



(1) CPU发起对地址0x1的写操作，写数据为0x0000000a；

(2) Master发出PutFullData操作；

(3) Slave向Cordic模块的地址0x1寄存器，写入了0x0000000a，同时向Master返回了AccessAck消息；

要求：Master发出请求后，5个周期内返回AccessAck/AccessAckData

作业提交

- 提交网址：
  http://10.249.12.98:6620/auth/users/sign_in

- slave.v文件（4分）
- 实验作业 （3分）

- 课上完成作业的第1部分

- 提交时间：5月30日（星期日）24：00

开始实验