



ThinPAD指令计算机系 统设计和实现

2022年秋

内容提要

- 我们要做什么？
- 我们已经有了什么？
- 我们可以怎样去做？
- 我们还能做什么？
- 我们最后要交什么？
- 具体时间安排

我们要做什么？

奋战三星期
做台计算机

我们要做什么？

□ 实验任务

- 支持指令流水的CPU
- 使用基本存储和扩展存储以及输入/输出
- 进行扩展

□ 实验目标

- 80分：完成个人实验+流水线处理器运行监控程序的基础版本
- 80分-100分：鼓励额外功能性能特点，包括但不限于中断、虚拟地址、外设驱动、ucore执行、特色应用、指令多发射、SIMD指令、动态分支预测、缓存等
- 100分：能够运行ucore或同等水平

我们已经有了什么？

□ THINPAD硬件平台

- FPGA
- SRAM Memory
- FLASH Memory
- UART/USB/Ethernet/DVI

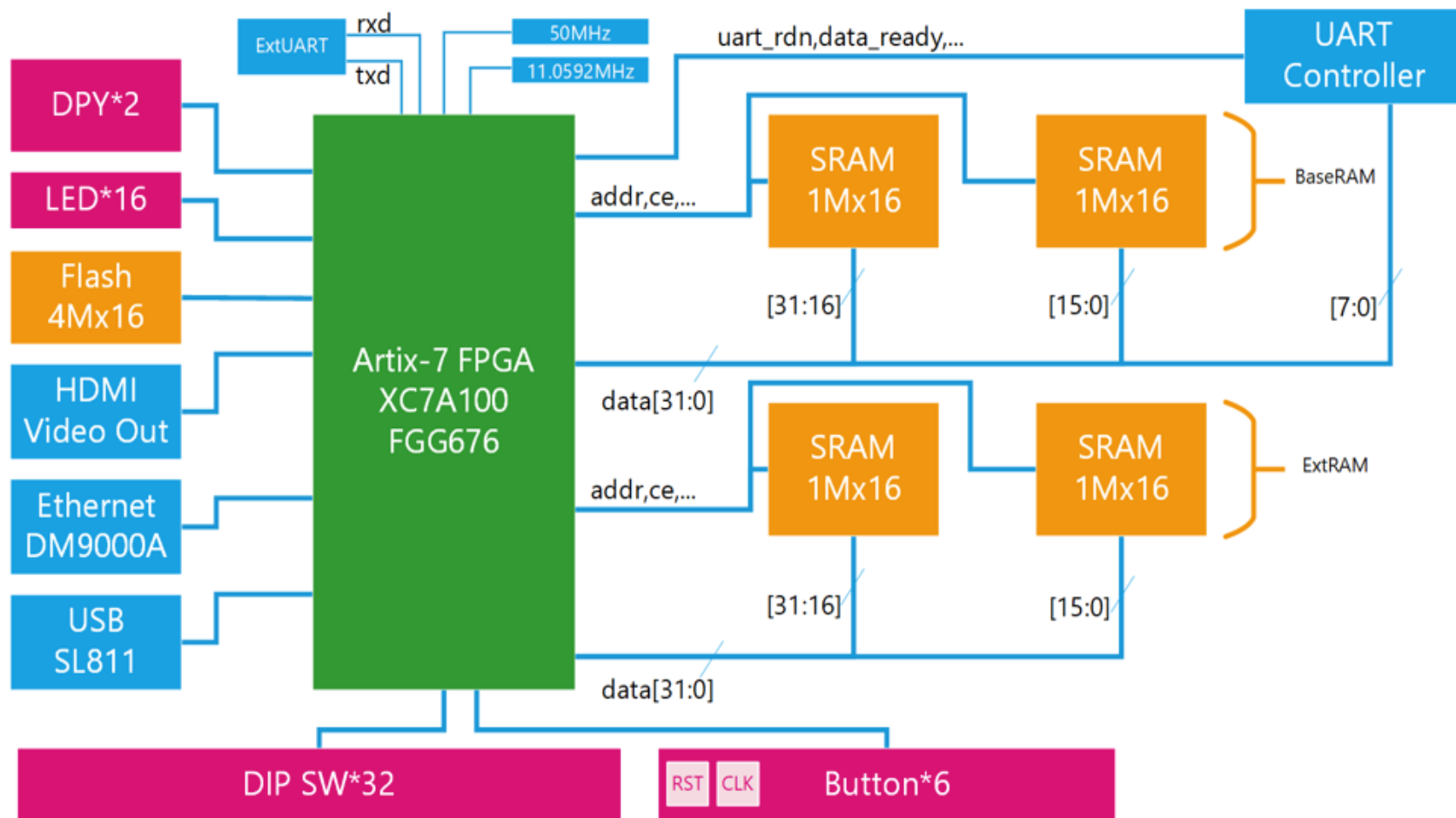
□ RISCV指令系统

□ 监控程序等辅助软件

□ 教材和实验指导材料

实验平台简介

硬件组成与结构



主要特点

- ❑ 搭载Xilinx Artix-7系列FPGA: XC7A100T
 - 100K LEs, 4.8Mb BRAM, 28nm制程
 - 支持最新的Vivado设计工具
- ❑ 两组SRAM内存, 每组4MB容量、32位数据线
- ❑ 8MB NOR Flash闪存
- ❑ 外围接口: SL811 USB, DM9000网卡, TFP410 DVI图像输出
 - USB可在软件配合下支持键盘
- ❑ 板载控制芯片, 支持在本地或远程完成实验

THINPAD-CLOUD教学计算机

控制芯片，实现控制面板上的设计上传、Flash读写功能

数码管与LED

图像输出

以太网

USB接口

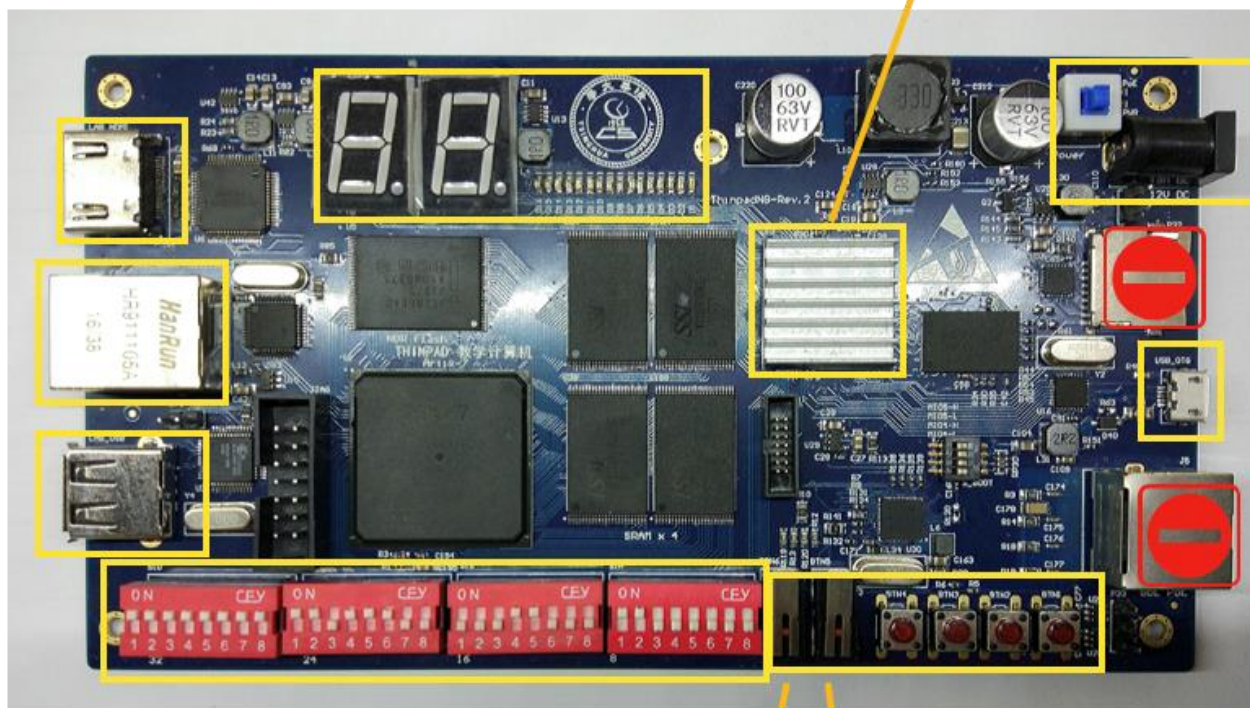
12V电源
及开关

控制端
USB接口

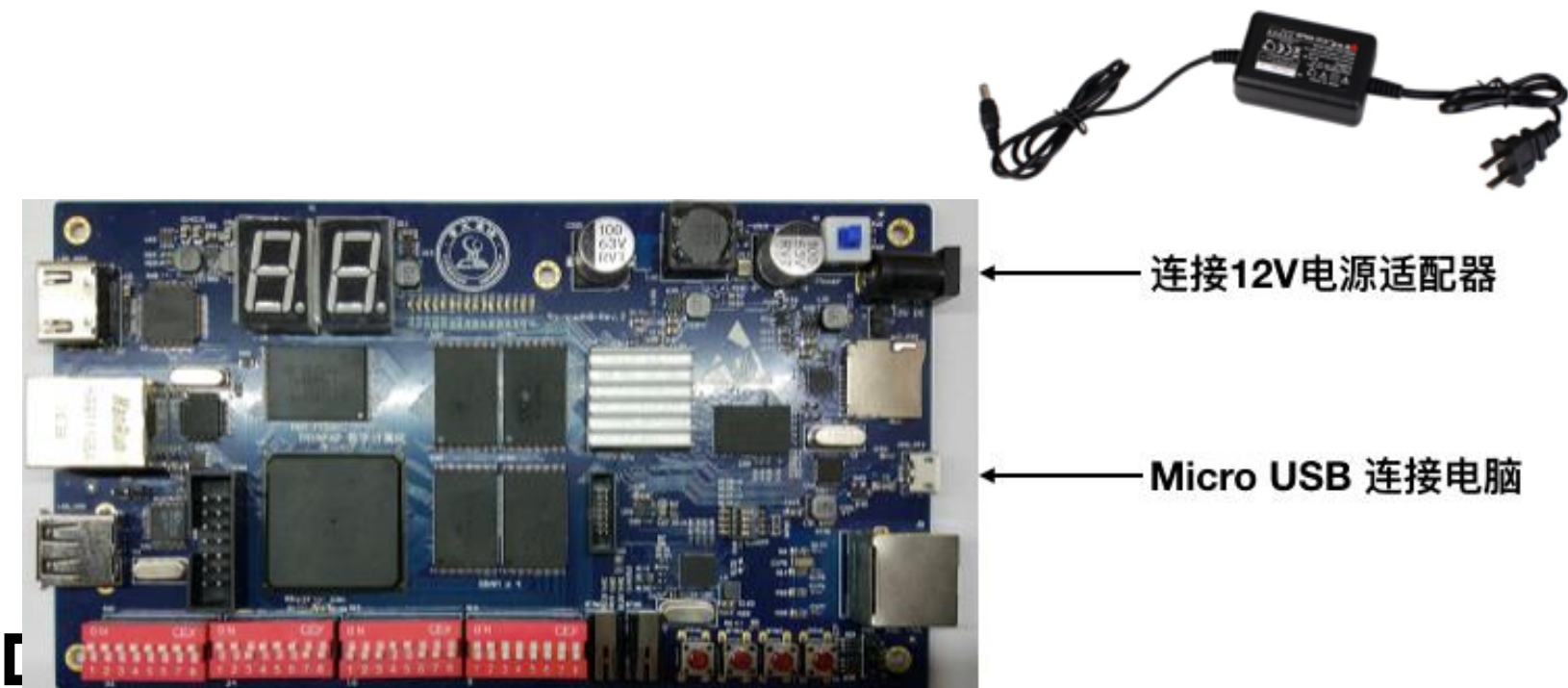
32*DIP开关 (推到ON为“1”)

6*按钮开关 (按下为“1”)

复位 单步时钟



实验平台使用方法——本地模式



- 使用USB与电脑相联之后可通过电脑进行各种操作

检查设备状态

在网络适配器设置中确认发现虚拟网卡，虚拟网卡获得192.168.8.x的IP。



属性	值
连接特定的 DNS 后缀	local
描述	Remote NDIS Compatible Device
物理地址	56-3A-2E-C1-B0-81
已启用 DHCP	是
IPv4 地址	192.168.8.100
IPv4 子网掩码	255.255.255.0

确认串口驱动安装成功，如果串口安装不成功，稍后按照控制面板页面上的说明安装驱动。

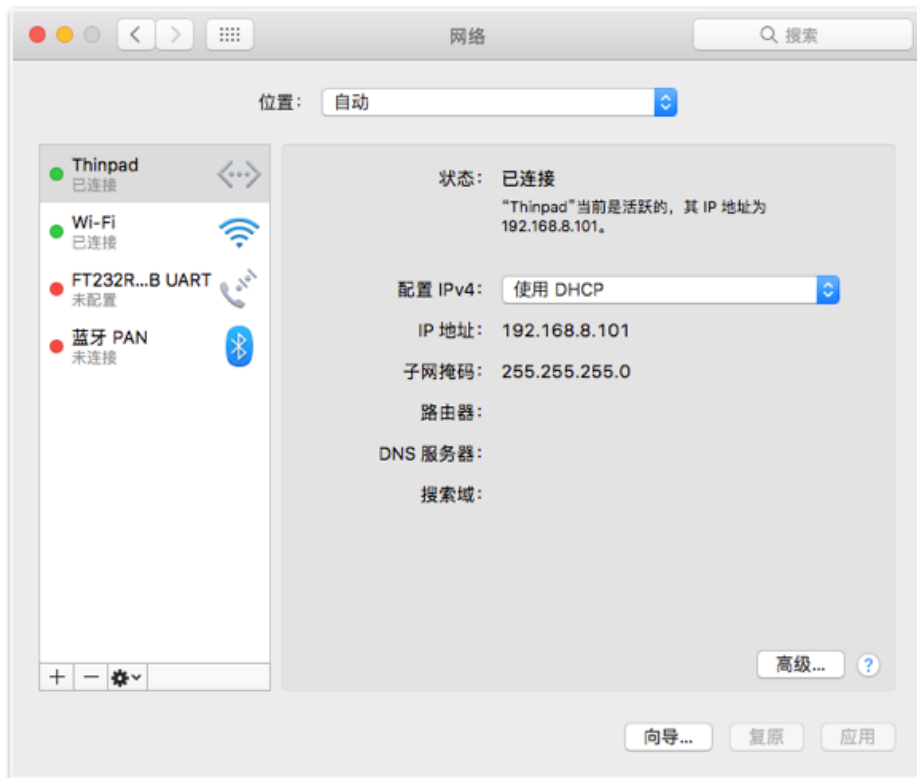
检查设备状态 (Linux)

```
ault qlen 1000
    link/ether 3e:d3:05:e7:68:b3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.8.101/24 brd 192.168.8.255 scope glo
        valid_lft 863918sec preferred_lft 863918sec
    inet6 fe80::1413:9395:9990:8b71/64 scope link
        valid_lft forever preferred_lft forever
4: ens160u2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 15
lt qlen 1000
    link/ether 56:3a:2e:c1:b0:81 brd ff:ff:ff:ff:ff:ff
    inet 192.168.8.100/24 brd 192.168.8.255 scope glo
        valid_lft 863918sec preferred_lft 863918sec
    inet6 fe80::5ca4:923b:54f:8e4c/64 scope link
        valid_lft forever preferred_lft forever
zhang@mint64 ~ $ ls /dev/ttyACM*
/dev/ttyACM0
```

串口 /dev/ttyACM*

虚拟网卡获得192.168.8.x的IP地址 (可能看到两个网卡, 无影响)

检查设备状态 (Mac)



```
[> ls /dev/cu.usbmodem*  
/dev/cu.usbmodem142123
```

串口 `/dev/cu.usbmodem*`
虚拟网卡Thinpad, 获得
192.168.8.x的IP地址

访问控制页面

用浏览器访问 <http://192.168.8.8/> 打开板子上的控制面板

上传 .bit 设计文件

选择串口

Flash 和 RAM 读写

Windows如果上一步没有成功安装串口，请按页面指示，下载驱动文件安装

上传设计文件

选择文件

 未选择任何文件
请选择 .bit 格式设计文件

写入实验FPGA

串口选择

CPLD串口控制器

应用

⚠每次重启开发板后默认为CPLD

将板上的 Micro USB 接口连接到电脑，可检测到一个虚拟串口。如果在旧版 Windows 系统上不能自动识别，可尝试下载[驱动文件](#)，并打开设备管理器，用更新驱动程序向导安装。

启动自检

FPGA外围状态

内存扫描:	测试通过
Flash:	通讯正常
USB:	通讯正常
Ethernet:	通讯正常
I/O:	测试通过
ID Code:	00:0a:35:00:1e:1f

Flash与RAM读写

存储选择

 Flash

容量 8MB

起始地址

 0x

(Byte)

起始地址应当按16位对齐（即为偶数）

读取数据

 0x

读取长度

(Byte)

读取

读取长度应当按16位对齐（即为偶数）

写入数据

选择文件

 未选择任何文件

写入

写入长度为数据文件大小，按16位对齐（即为偶数）

JTAG调试器连接

HARDWARE MANAGER - Unconnected

No hardware target is open. Open target

Name	Status
localhost (0)	Connected

Hardware Server Properties...

Refresh Server

Add Xilinx Virtual Cable (VVC)...

Create VUP Target...

Specify the host name and the port of the virtual cable

Host name: 192.168.8.8

Port: 2542 [default is 2542]

OK Cancel

实验平台使用方法——远程模式

- ❑ 学生从系统管理员处得到服务器地址和账号、密码后，可以通过浏览器登录到远程实验系统，成功连接到一块远程实验板，并在板上完成实验。
- ❑ 可以上传设计文件
- ❑ 通过实验平台进行基本操作
- ❑ 存储器读写

设计文件上传与更新

ThinpadCloud 文件上传 工作区域 admin 登出

Thinpad rev.3

ThinPAD-Cloud 教学计算机硬件平台，硬件版本3。

参考文件：
[工程模板下载（含引脚约束） rev.3](#)

开发板分配

Bitstream hw_test3.bit
编译得到的 .bit 文件

文件信息 soc_toplevel;UserID=0xFFFFFFFF;COMPRESS=TRUE;Version=2017.3

编译时间 2018/08/12 10:08:09

□ 设计文件上传

更新设计文件

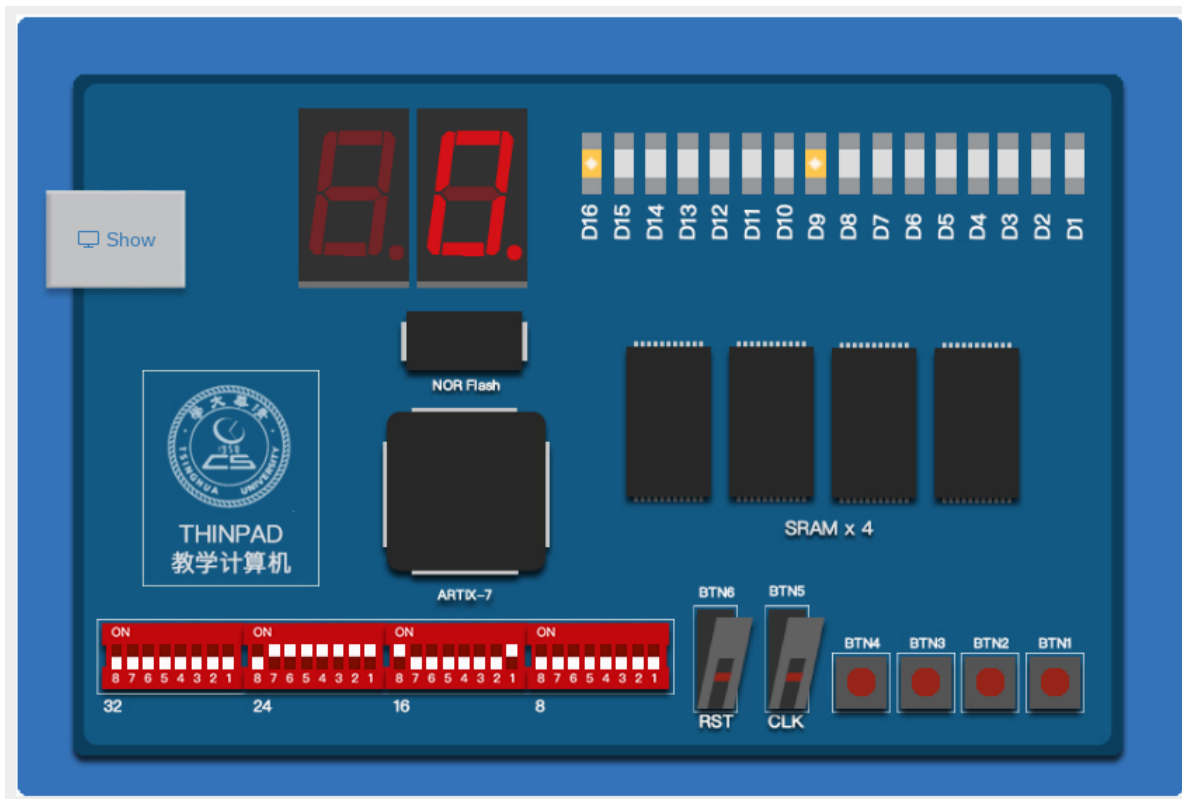
hw_test3.bit

Choose a "*.bit" bitstream file

Design Name	soc_toplevel;UserID=0xFFFFFFFF;COMPRESS=TRUE;Version=2017.3
Build Time	2018/08/12 10:08:09

- 设计文件更新

实验平台基本操作



串口输入/输出

换行符
选择发送数据时添加的换行符

输入

位置

波特率

校验位

数据位

停止位

• 串口配置

□ 面板操作

存储器读写

Flash&RAM 读写 使用本功能将重置FPGA

存储选择 BaseRAM 容量 4MB

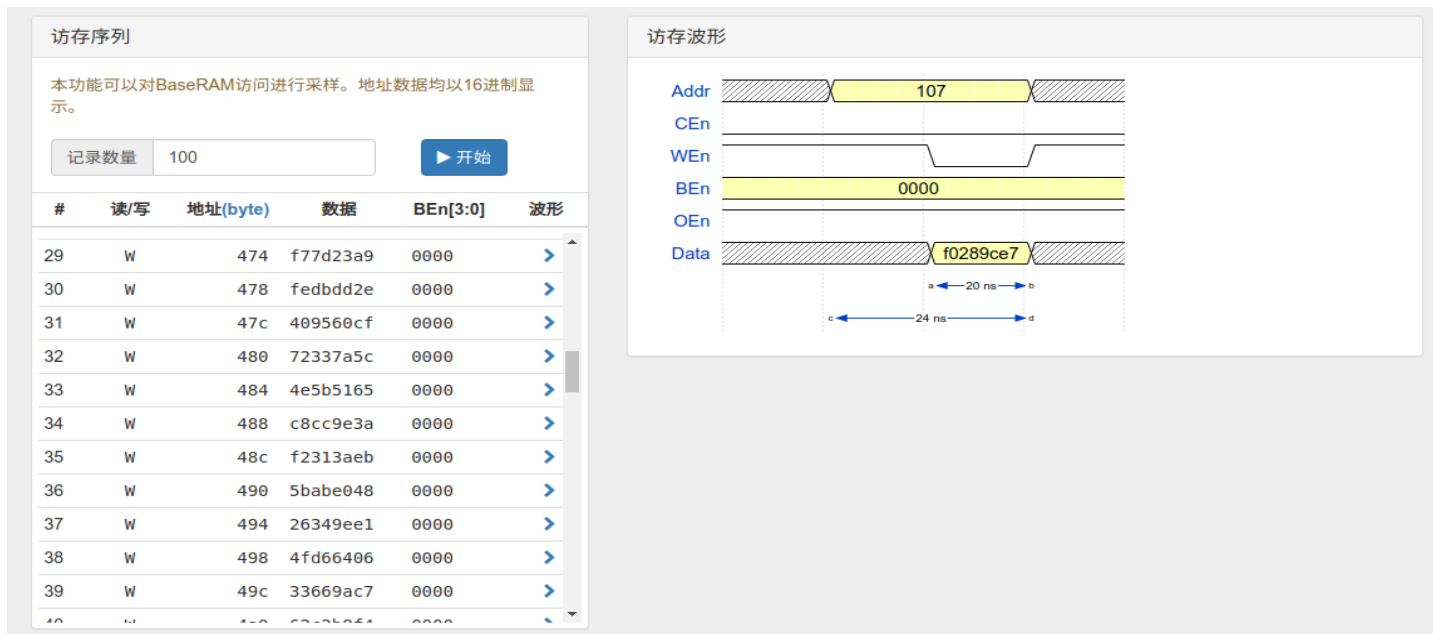
起始地址 0x 0 (Byte)
起始地址应当按16位对齐（即为偶数）

读取数据 0x 读取长度 (Byte) 读取
读取长度应当按16位对齐（即为偶数）

写入数据 Choose File No file chosen 写入
写入长度为数据文件大小，按16位对齐（即为偶数）

❑ 读写Flash和RAM时，为了避免信号电平冲突，系统会清除实验FPGA中的用户设计，并在读写完成后自动重载

SRAM访问地址序列记录



□ 远程实验平台增加了一个功能，能够记录程序运行过程中，对SRAM访问时的地址序列

FPGA样例工程说明

□ 提供创建好的Vivado工程模板

- 已包含全部的引脚约束（不用根据表格手工分配）
- 包含LED、数码管、图像输出的示例代码
- 在FPGA内部实现串口收发的示例代码

□ 所有信号定义、FPGA型号等信息以工程模板为准

```

input wire clk_50M,           //50MHz 时钟输入
input wire clk_11M0592,      //11.0592MHz 时钟输入

input wire clock_btn,        //BTN5手动时钟按钮开关，带消抖电路，按下时为1
input wire reset_btn,        //BTN6手动复位按钮开关，带消抖电路，按下时为1

input wire[3:0] touch_btn,   //BTN1~BTN4，按钮开关，按下时为1
input wire[31:0] dip_sw,     //32位拨码开关，拨到“ON”时为1
output wire[15:0] leds,      //16位LED，输出时1点亮
output wire[7:0] dpy0,       //数码管低位信号，包括小数点，输出1点亮
output wire[7:0] dpy1,       //数码管高位信号，包括小数点，输出1点亮

//CPLD串口控制器信号
output wire uart_rdn,        //读串口信号，低有效
output wire uart_wrn,        //写串口信号，低有效
input wire uart_dataready,   //串口数据准备好
input wire uart_tbre,        //发送数据标志
input wire uart_tsre,        //数据发送完毕标志

//BaseRAM信号
inout wire[31:0] base_ram_data, //BaseRAM数据，低8位与CPLD串口控制器共享
output wire[19:0] base_ram_addr, //BaseRAM地址
output wire[3:0] base_ram_be_n, //BaseRAM字节使能，低有效。如果不使用字节使能，请保持为0
output wire base_ram_ce_n,      //BaseRAM片选，低有效
output wire base_ram_oe_n,      //BaseRAM读使能，低有效
output wire base_ram_we_n,      //BaseRAM写使能，低有效

```

```

//ExtRAM信号
inout wire[31:0] ext_ram_data, //ExtRAM数据
output wire[19:0] ext_ram_addr, //ExtRAM地址
output wire[3:0] ext_ram_be_n, //ExtRAM字节使能, 低有效。如果不使用字节使能, 请保持为0
output wire ext_ram_ce_n, //ExtRAM片选, 低有效
output wire ext_ram_oe_n, //ExtRAM读使能, 低有效
output wire ext_ram_we_n, //ExtRAM写使能, 低有效

//直连串口信号
output wire txd, //直连串口发送端
input wire rxd, //直连串口接收端

//Flash存储器信号, 参考 JS28F640 芯片手册
output wire [22:0]flash_a, //Flash地址, a0仅在8bit模式有效, 16bit模式无意义
inout wire [15:0]flash_d, //Flash数据
output wire flash_rp_n, //Flash复位信号, 低有效
output wire flash_vpen, //Flash写保护信号, 低电平时不能擦除、烧写
output wire flash_ce_n, //Flash片选信号, 低有效
output wire flash_oe_n, //Flash读使能信号, 低有效
output wire flash_we_n, //Flash写使能信号, 低有效
output wire flash_byte_n, //Flash 8bit模式选择, 低有效。在使用flash的16位模式时请设为1

```

实验平台上的器件

开关与LED

- ❑ 高低两位7段数码管，对应信号名称dpy1、dpy0，输出高电平点亮笔段。模板中提供了数码管译码逻辑
- ❑ 16位LED对应信号名称leds，输出高电平点亮笔段
- ❑ 32位拨码开关对应dip_sw，拨到“ON”位置时输入信号为1
- ❑ 4位按钮开关(BTN1~4)，对应touch_btn信号，按下输入为1
- ❑ 微动开关BTN5、BTN6，对应信号clock_btn、reset_btn，自带消抖，按下时输入为1，用于手工输入时钟和复位

板载存储芯片SRAM

- ❑ THINPAD-Cloud硬件平台提供了SRAM和Flash两类存储器
- ❑ 板上共有4片ISSI公司生产的IS61WV102416型高速异步SRAM芯片。每片SRAM存储容量为 $1024K \times 16\text{bit}$ ，即16位宽，1M深度，容量2MB。板上总共有8MB的SRAM存储空间。
- ❑ 将4片SRAM分为两组，一组内有两片。组内的两片SRAM地址和读写控制信号连接到一起，数据线分开，这样相当于把两片SRAM拼接成一个32位宽的SRAM，以便实现32位的计算机系统实验。

内存信号

□ RAM分为Base和Ext两组，地址、数据和控制信号完全独立

□ 每组RAM 4MB，32bit数据线，20bit地址线。实际是用两片16位SRAM芯片拼成的

□ 控制信号

- be_n[3:0]: RAM字节使能，低有效，对应32位中的4个字节

- ce_n: RAM片选，低有效

- oe_n: RAM读使能，低有效

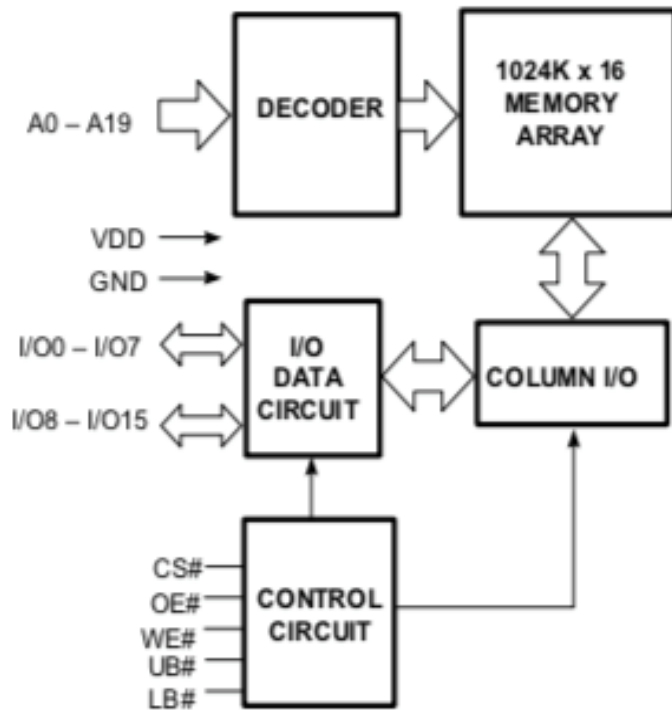
- we_n: RAM写使能，低有效

□ 读写时序（详细时序图见芯片手册）

- 读：设置地址，置控制信号ce_n=oe_n=be_n=0, we_n=1，等待一段时间后读数据

- 写：设置数据、地址，置控制信号ce_n=we_n=be_n=0，等待一段时间后置we_n=1

板载存储芯片Flash



- 一片8MB容量的NOR Flash作为非易失存储，可用于存储实验用的程序，比如监控程序等

实验平台外部接口

- 串行接口
- DVI图像输出接口
- 100M以太网接口
- USB接口

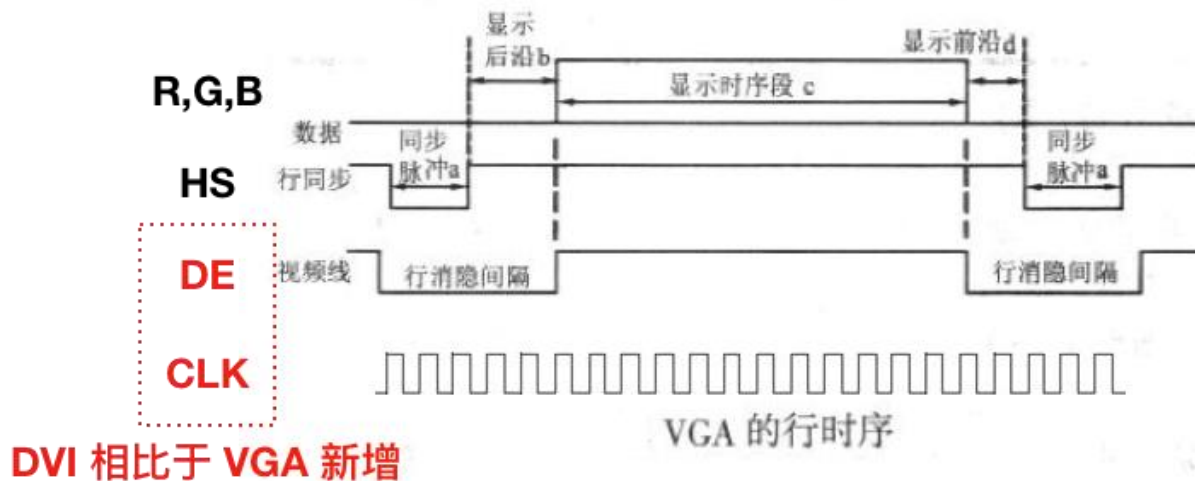
串口信号

- 串口有两种，可在控制面板（网页上）选择
- CPLD串口控制器：串口控制器在FPGA外部。5根控制信号，数据线与Base RAM的低8位共享
 - 串口参数固定为波特率 9600，无校验位
 - 设计中应当正确处理串口、内存数据线共享
- 直连串口：使用直接与FPGA相连的txd/rxd信号，学生在FPGA中自行实现串口控制器
 - 灵活，可以自己实现串口缓冲，可变波特率

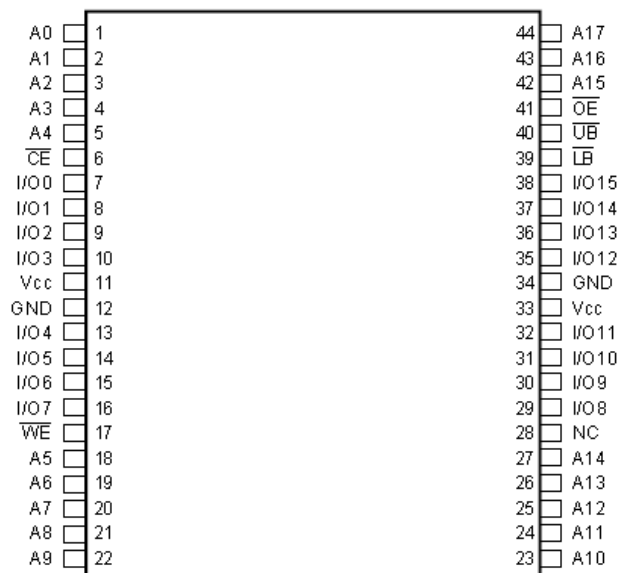
视频输出

- 图像输出由VGA变为DVI协议的数字信号（接口是HDMI）
 - 板上有DVI信号转换芯片
 - FPGA设计主要区别：输出与像素同步的时钟信号和DE信号（参考工程模板）
 - DE信号用于标记有效的图像数据，在消隐区为0，显示区为1
 - 像素数据为8位，R:G:B=3:3:2

视频输出时序



内存访问

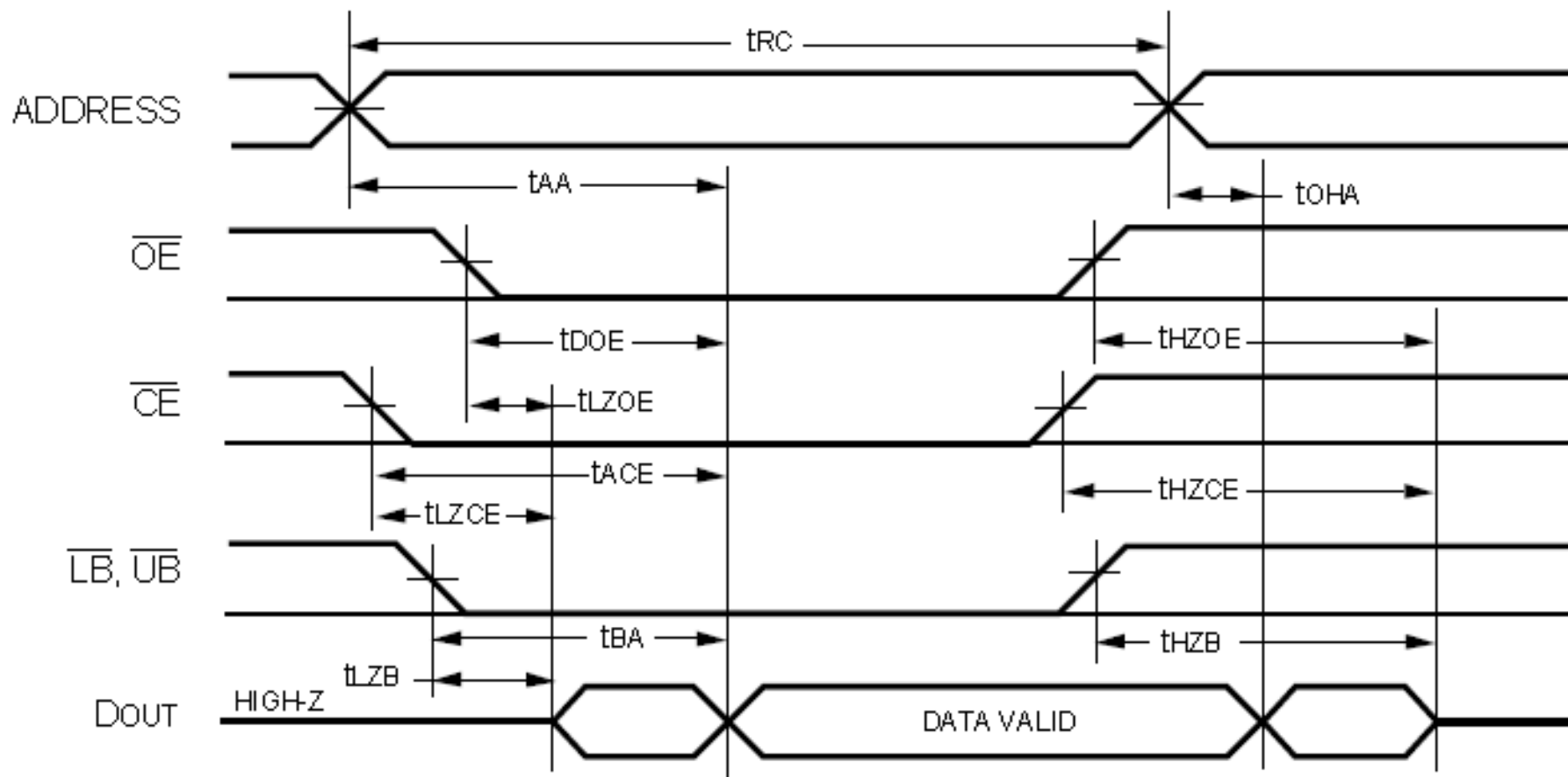


- 控制信号
- \overline{CE} 、 \overline{WE} 、 \overline{OE}
- 访问时序
- 如何保证？

TRUTH TABLE

Mode	\overline{WE}	\overline{CE}	\overline{OE}	\overline{LB}	\overline{UB}	I/O PIN		Vcc Current
						I/O0-I/O7	I/O8-I/O15	
Not Selected	X	H	X	X	X	High-Z	High-Z	IsB1, IsB2
Output Disabled	H	L	H	X	X	High-Z	High-Z	Icc
	X	L	X	H	H	High-Z	High-Z	
Read	H	L	L	L	H	DOUT	High-Z	Icc
	H	L	L	H	L	High-Z	DOUT	
	H	L	L	L	L	DOUT	DOUT	
Write	L	L	X	L	H	DIN	High-Z	Icc
	L	L	X	H	L	High-Z	DIN	
	L	L	X	L	L	DIN	DIN	

内存读时序



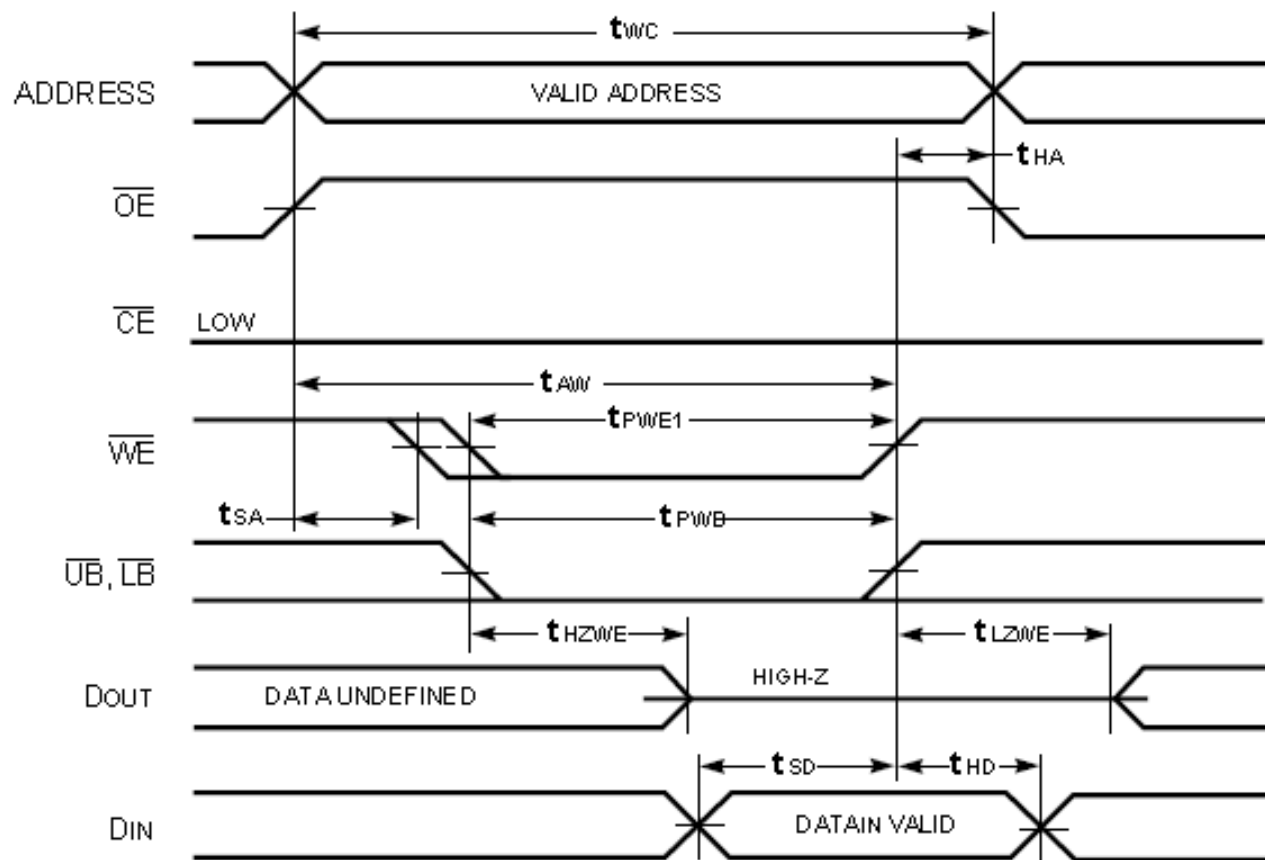
内存读参数

READ CYCLE SWITCHING CHARACTERISTICS⁽¹⁾ (Over Operating Range)

Symbol	Parameter	-8		-10		-12		-15		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
t _{RC}	Read Cycle Time	8	—	10	—	12	—	15	—	ns
t _{AA}	Address Access Time	—	8	—	10	—	12	—	15	ns
t _{OH}	Output Hold Time	3	—	3	—	3	—	3	—	ns
t _{ACE}	$\overline{\text{CE}}$ Access Time	—	8	—	10	—	12	—	15	ns
t _{DOE}	$\overline{\text{OE}}$ Access Time	—	4	—	5	—	6	—	7	ns
t _{HZOE} ⁽²⁾	$\overline{\text{OE}}$ to High-Z Output	0	4	—	5	—	6	0	6	ns
t _{LZOE} ⁽²⁾	$\overline{\text{OE}}$ to Low-Z Output	0	—	0	—	0	—	0	—	ns
t _{HZCE} ⁽²⁾	$\overline{\text{CE}}$ to High-Z Output	0	4	0	5	0	6	0	6	ns
t _{LZCE} ⁽²⁾	$\overline{\text{CE}}$ to Low-Z Output	3	—	3	—	3	—	3	—	ns
t _{BA}	$\overline{\text{LB}}, \overline{\text{UB}}$ Access Time	—	4	—	5	—	6	—	7	ns
t _{HZB}	$\overline{\text{LB}}, \overline{\text{UB}}$ to High-Z Output	0	4	0	5	0	6	0	6	ns
t _{LZB}	$\overline{\text{LB}}, \overline{\text{UB}}$ to Low-Z Output	0	—	0	—	0	—	0	—	ns

内存写时序

WRITE CYCLE NO. 2 ($\overline{\text{WE}}$ Controlled. $\overline{\text{OE}}$ is HIGH During Write Cycle) ^(1,2)



内存写参数

WRITE CYCLE SWITCHING CHARACTERISTICS^(1,3) (Over Operating Range)

Symbol	Parameter	-8		-10		-12		-15		Unit
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
t _{WC}	Write Cycle Time	8	—	10	—	12	—	15	—	ns
t _{SC_{CE}}	\overline{CE} to Write End	7	—	8	—	9	—	10	—	ns
t _{AW}	Address Setup Time to Write End	7	—	8	—	9	—	10	—	ns
t _{HA}	Address Hold from Write End	0	—	0	—	0	—	0	—	ns
t _{SA}	Address Setup Time	0	—	0	—	0	—	0	—	ns
t _{PWB}	\overline{LB} , \overline{UB} Valid to End of Write	7	—	8	—	9	—	10	—	ns
t _{PWE}	\overline{WE} Pulse Width	7	—	8	—	9	—	10	—	ns
t _{SD}	Data Setup to Write End	4.5	—	5	—	6	—	7	—	ns
t _{HD}	Data Hold from Write End	0	—	0	—	0	—	0	—	ns
t _{HZWE⁽²⁾}	\overline{WE} LOW to High-Z Output	—	4	—	5	—	6	—	7	ns
t _{LZWE⁽³⁾}	\overline{WE} HIGH to Low-Z Output	3	—	3	—	3	—	3	—	ns

实验概述

基本步骤

□ 确定目标

- 流水线、扩展方案、展示方案
- 工作计划
- 组内分工

□ 概要设计

- 数据通路设计：每条指令在数据通路上的实现过程
- 指令流程表：每条指令每个步骤所完成的功能
- 流水段设计
- 控制信号设计
- 模块设计（各模块基本功能和接口）

□ 详细设计

- 各模块编码实现

□ 编码实现

□ 模拟分析

□ 硬件调试

□ 总结报告

我们还能做什么

□ 硬件

- 对流水线进行完善
 - 结构冲突、数据冲突、控制冲突
- 中断
- 分时操作

□ 软件

- 监控程序的完善
- 有创意的应用
- 性能分析和评价

时间安排

□ 截止时间

- 实验检查：12月5日~ 12月8日（只能检查一次）
- 实验测试：12月9日
- 实验报告上交：12月13日
- 大实验分班分组答辩：12月16日

□ 实验室开放及助教

- 答疑日期：11月17、21、24、28号；12月1号
- 答疑时段：14:00~16:30
- 地点：东主楼 9-206
- 实验室白天开放

注意事项

- ❑ 如果需要实验箱的话，携带装有开发板的盒子时避免剧烈撞击
- ❑ 拿板子时请拿边缘，不要触碰元件，防止静电损坏元件
- ❑ 使用时应当避免板子形变，造成芯片脱焊
- ❑ 板子背面有裸露的焊盘，不要把板子放到导电物体上（如金属外壳）

实验成果

□ 完整的实验报告

- 实验目标
- 实验内容
- 实验展示
- 实验心得体会

□ 完整的源代码

- 设计框图
- 指令流程表
- Verilog代码
- 其他代码

大实验报告要求

□ 实验目标概述

□ 主要模块设计

- 硬件：Datapath、Controller、Memory、I/O
- 软件：对系统软件的修改、应用软件

□ 主要模块实现

□ 实验成果展示

□ 实验心得和体会

□ 提交材料

- 实验报告
- 视频材料
- 完整源代码及相关说明

实验过程与注意事项

实践

□ 分析ThinPAD RISC-V指令系统

■ 指令格式

■ 操作码、操作数地址

■ 寻址方式

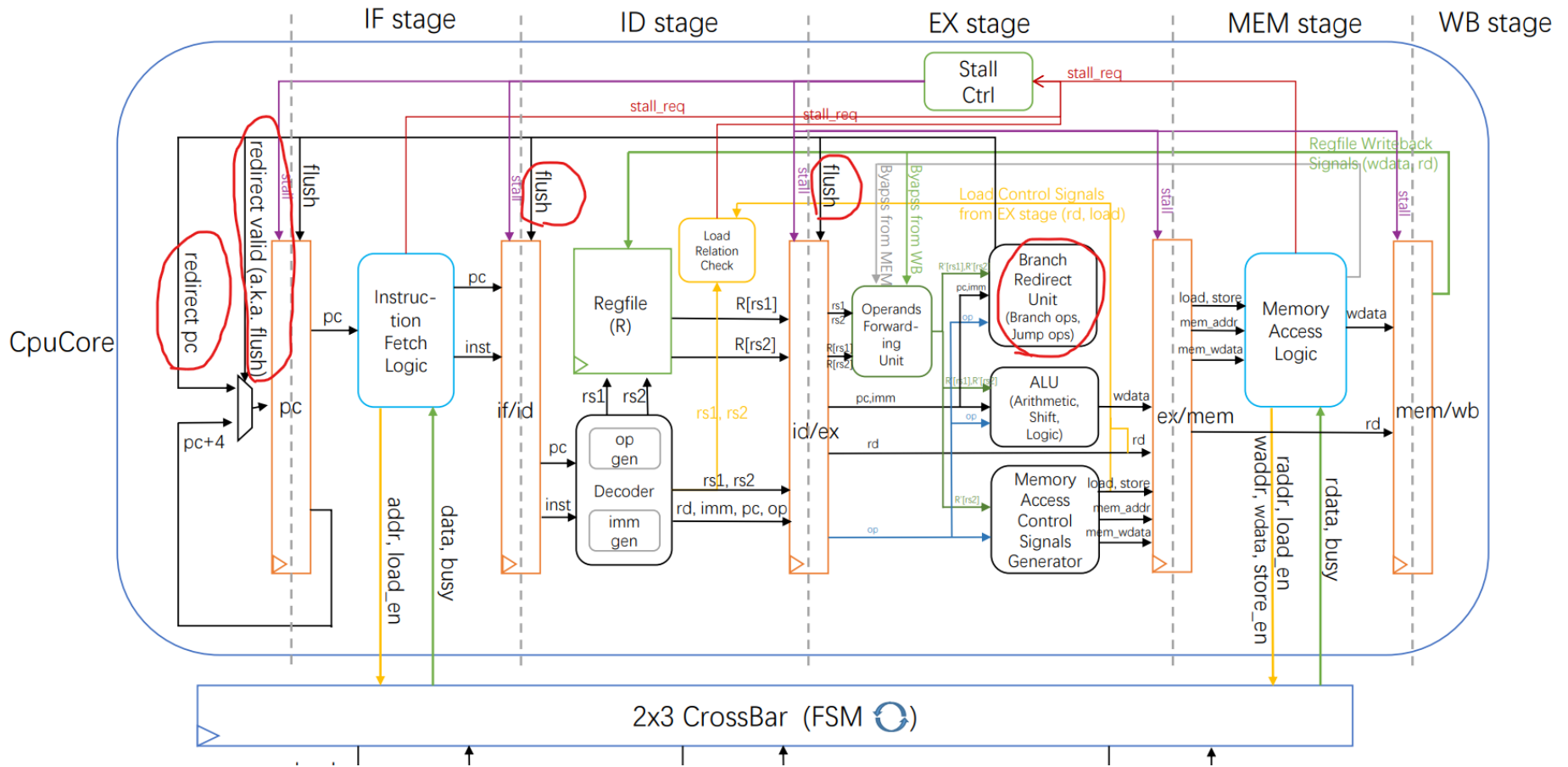
■ 寄存器寻址、立即数寻址、基址寻址

□ 设计数据通路

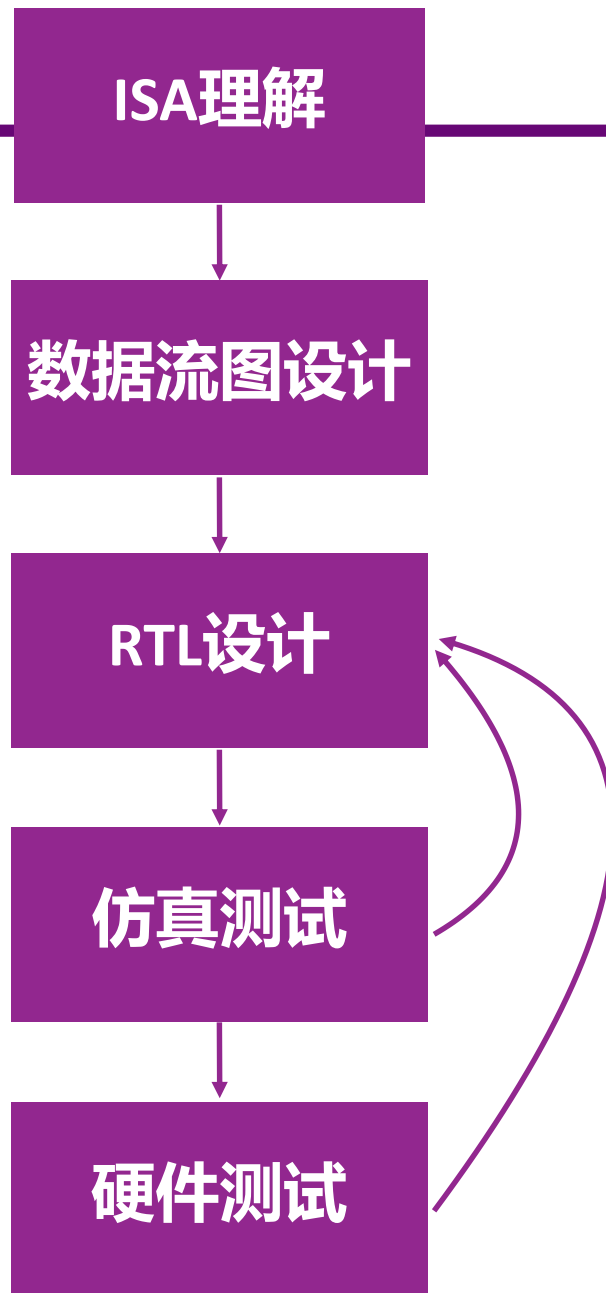
□ 划分指令执行步骤

■ 指令流水

流水线处理器框架结构



设计步骤



ISA 理解

- 19+x条指令如何编码？
指令的语义？
- 每条指令在各阶段/
状态要做什么事情？
- 监控程序如何使用这
些指令
 - 尤其注意访存指令
- 地址空间划分规则
 - 串口读写方法

The RISC-V Instruction Set Manual
Volume I: Unprivileged ISA
Document Version 20191214-draft

Editors: Andrew Waterman¹, Krste Asanović^{1,2}
¹SiFive Inc.,
²CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu
July 27, 2020

数据流图设计

□流水线CPU：各流水阶段设计

- 几个流水阶段？每个阶段的功能？
- 三种冲突怎么解决？
 - 控制流水线暂停的真值表

面向硬件的编码

❑ 综合→优化→布局→布线→生成 *.bit

❑ 带着硬件思维编写RTL代码

■ 考虑你的代码会被综合成什么电路

❑ 哪些RTL是可综合的？

■ Verilog的很小一部分是可综合的

■ 时延标识不可综合 (a = #5 b;)

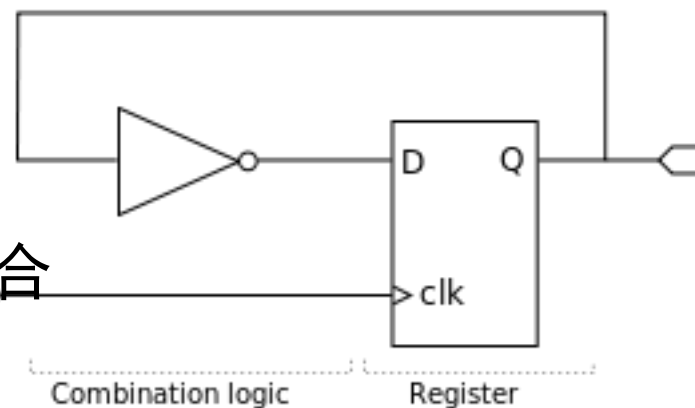
■ initial 块严格来说不可综合

■ 循环语句严格来说不可综合

■ 用来generate代码除外

■ always块指定多个敏感源不可综合

```
reg Q;  
always @(posedge clk) begin  
    if(Q) begin  
        Q <= 0;  
    end else begin  
        Q <= 1;  
    end  
end
```



FPGA中的硬件资源

□ LUT: 6输入-2输出查找表, 可级联

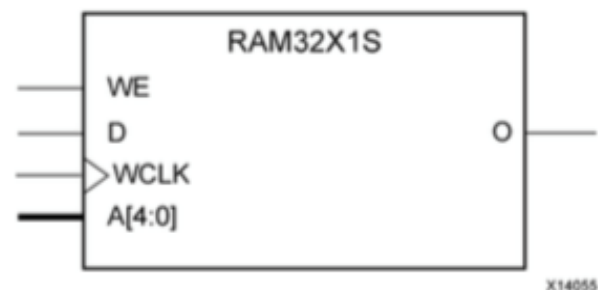
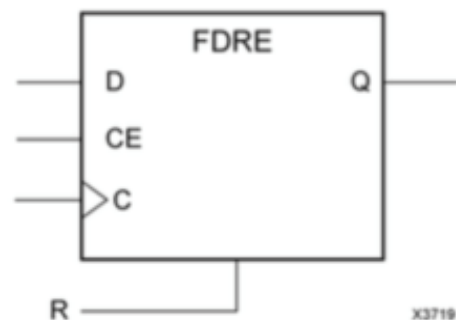
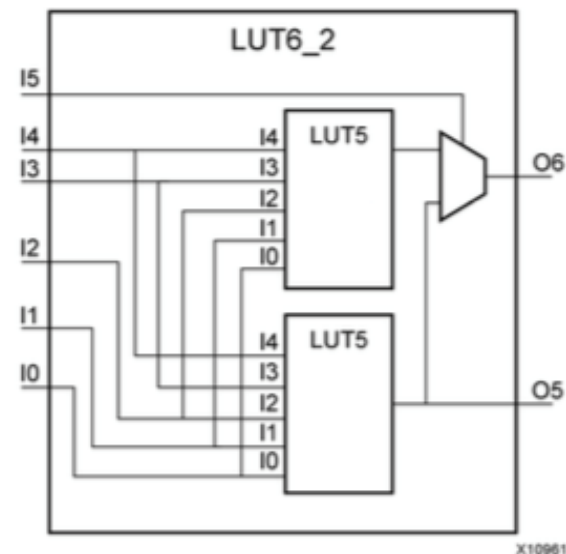
- 任意组合逻辑

□ FDRE: 带有时钟使能和服务的D触发器

- 即一位的“reg”

□ 片内存储单元: 单/双端口读写

- 实现CPU寄存器、Cache等



常见的编码问题-多源驱动

❑ Critical Warning :
[Synth 8-3352] multi-driven net
<signal_name> with
1st driver pin
'<pin_name1>' [xxx.v.3]

❑ 问题原因

- 同一触发器（reg）存在多个驱动源

❑ 解决方法

- 一个reg只能在一个always块中赋值

```
always @(posedge clk) begin
    if (xxx) begin
        s <= 1;
    end
end
always @(posedge clk) begin
    if(yyy) begin
        s <= 0;
    end
end
```

常见的编码问题-电平锁存器

❑ WARNING:
[Synth 8-327]
inferring latch for
variable 's_reg'
[*..v:12]

❑ 问题原因

- 组合逻辑敏感信号列表不全

❑ 解决方法

- 描述组合逻辑时用always@(*)

```
always @(a) begin
    if (a) begin
        s <= 1;
    end else if(b) begin
        s <= 0;
    end
end
```

```
always @(*) begin
    if (a) begin
        s <= 1;
    end else if(b) begin
        s <= 0;
    end
end
```

?

s	a	b
1	1	x
0	0	1
-	0	0

常见编码问题-电平锁存器

❑ WARNING: [Synth 8-327] inferring latch for variable 's_reg' [*.v:12]

❑ 问题原因

- 组合逻辑敏感信号列表不全，空分支

❑ 解决方法

- 描述组合逻辑时用 `always@(*)`，并且保证所有分支均被覆盖

```
always @(a) begin
    if (a) begin
        s <= 1;
    end else if(b) begin
        s <= 0;
    end
end
```

```
always @(*) begin
    if (a) begin
        s <= 1;
    end else if(b) begin
        s <= 0;
    end else begin
        s <= 0;
    end
end
```

s	a	b
1	1	x
0	0	1
0	0	0

常见编码问题-跨时钟域

□ 问题原因

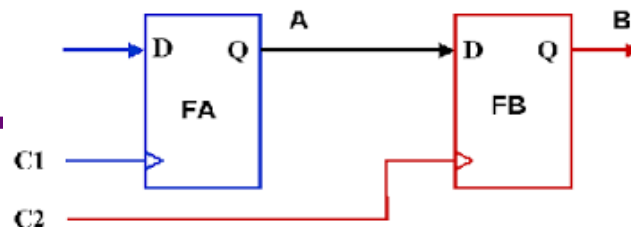
- 捕获一个来自不同的时钟域的信号时，建立时间不满足，出现亚稳态

□ 解决方法

- 使用两级触发器，构造一个同步器，消除亚稳态

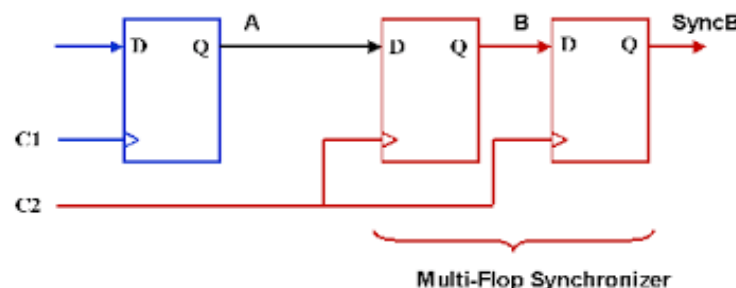
□ 扩展阅读：

<https://www.fpga4fun.com/CrossClockDomain.html>



```
reg in, a, b;
always @(posedge c1) begin
    a <= in;
end
always @(posedge c2) begin
    b <= a;
end
```

```
reg in, a, b, sync_b;
always @(posedge c1) begin
    a <= in;
end
always @(posedge c2) begin
    b <= a;
    sync_b <= b;
end
```



单个模块行为仿真

- 自行编写简单的测试信号生成器，进行单元测试
 - always和时延生成时钟信号
 - initial 表示信号变化过程
 - repeat(n) 返回执行n次
 - @(posedge clk) 等待一个上升沿
- 这些测试代码仅限于仿真，不可综合

```
`timescale 1ns/1ps

reg clock_10M;
reg reset;

initial begin
    clock_10M = 0;
end

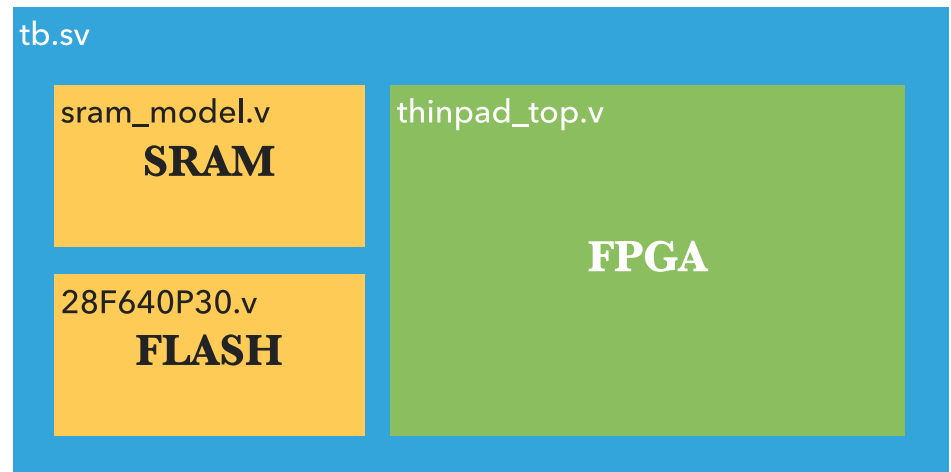
always begin
    #50 clock_10M = ~clock_10M;
end

initial begin
    reset = 1;
    repeat(5) begin
        @(posedge clock_10M);
    end
    reset = 0;
end
```


整板行为仿真

- 在各个模块单元测试通过后，应当进行整个设计的仿真测试
 - 仿真应包括CPU、内存控制器和内存
- 工程模板提供测试代码，Vivado里的“Simulation Sources”源码树中
 - 提供内存Flash模型
 - 模拟实验板上连线

■ **tb.sv**



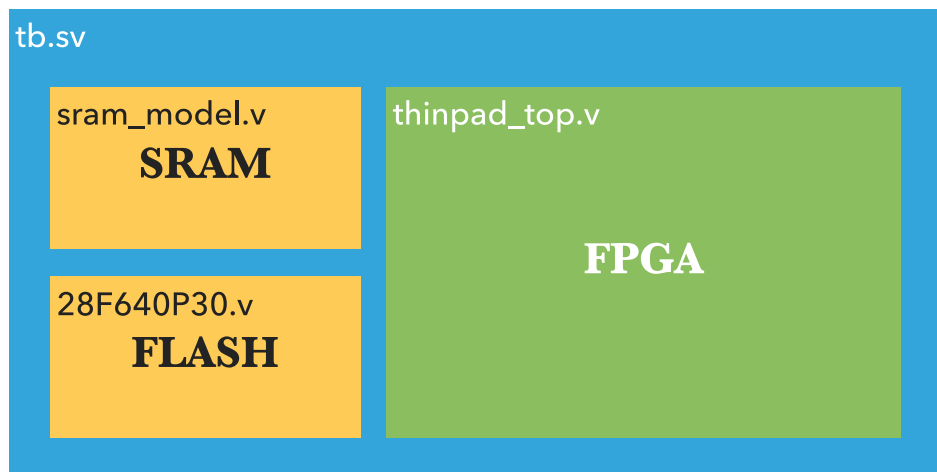
整板行为仿真

- tb.sv 中还可以指定内存与Flash初始化文件
- 将监控程序kernel.bin直接加载到RAM中，即可在仿真器中运行监控
- 仿真比硬件运行慢，但避免了编译过程

```
//Windows需要注意路径分隔符的转义，例如"D:\\foo\\bar.bin"
parameter BASE_RAM_INIT_FILE = "/tmp/main.bin"; //BaseRAM初始化文件,
parameter EXT_RAM_INIT_FILE = "/tmp/eram.bin"; //ExtRAM初始化文件,
parameter FLASH_INIT_FILE = "/tmp/kernel.elf"; //Flash初始化文件,

assign rxd = 1'b1; //idle state

initial begin
    //在这里可以自定义测试输入序列，例如：
    dip_sw = 32'h2;
    touch_btn = 0;
    for (integer i = 0; i < 20; i = i++) begin
        #100; //等待100ns
        clock_btn = 1; //按下手工时钟按钮
        #100; //等待100ns
        clock_btn = 0; //松开手工时钟按钮
    end
end
```



FPGA调试工具



网页上的图示

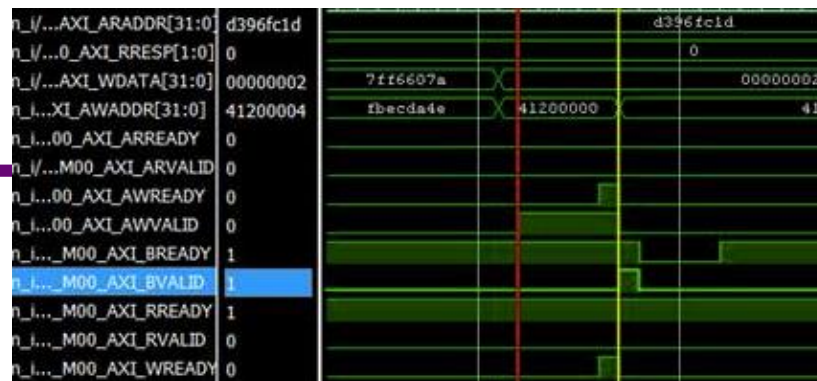
□ Vivado提供了两种调试工具

- ILA：嵌入逻辑分析仪，采样信号在一段时间内的波形
- VIO：虚拟输入输出接口，提供“无限”数量的LED和开关

□ JTAG 连接方法

- Vivado中打开Hardware Manager后，参考Thinpad控制面板上的图片说明配置
- 使用ILA时，信号的采样时钟不能低于4MHz，否则ILA工作异常

Vivado硬件调试工具



□ ILA：嵌入逻辑分析仪

- 编译时设置要观察的信号，声明 (`* MARK_DEBUG = "TRUE" *`)
- 硬件测试时**采样**信号在一段时间内的**波形**

- 可以设置触发条件

□ VIO：虚拟输入输出

Name	Value	Activity	Direction	VIO
fsys_j/GPIO_O[31:0]	[H] 0000_0000		Input	hw_vio_1
LED			Input	hw_vio_1
fsys_j/locked	[B] 1		Input	hw_vio_1
fsys_j/simple_out[31:0]	[H] 0000_000F		Input	hw_vio_1
fsys_j/simple_in[31:0]	[H] 0000_0000		Output	hw_vio_1
RESET	0		Output	hw_vio_1

- 添加 VIO IP 核到设计中，让你拥有“无限”数量的**LED**和**开关**
- 硬件测试时，**实时**观察信号当前状态，或手工设定信号状态

CPU模拟器

❑ 业界广泛使用的开源RISCV CPU模拟器 —— QEMU

- 可作为标准的CPU参考实现
- 当你不知道是软件还是硬件bug时，听QEMU的
- 配合 GDB 单步（单指令）运行程序，观察所有寄存器变化，查看内存数据，了解程序运行流程
- 自行修改QEMU，打印程序用到的指令，每一步执行结果

❑ Project I 中已经使用过QEMU了

- 具体步骤可参考实验指导书

GNU Binutils（详细看实验指导材料）

- ❑ GNU Binutils 是一套针对 elf 可执行文件格式的工具集
- ❑ `addr2line -e fib.elf 0x80000008`: 从 elf 文件中, 根据地址查得代码中的行号
- ❑ `objcopy`: 将 elf 中的某个 section 提取出来
 - `objcopy -O binary -j .text kernel.elf kernel.bin` 提取指令部分到bin文件
- ❑ `objdump -D main.elf` 反汇编程序
- ❑ `readelf`: 获取 elf 中有用的信息
 - `readelf -l main.elf` 代码的加载地址和入口点
 - `readelf -s main.elf` 所有的符号（变量、函数）的地址
 - `readelf -S main.elf` 所有 section 的地址

一些提示和帮助，供参考（1）

□ 模拟器不支持的指令（自定义指令）怎么办？

- 通过二进制写进去，按照指令的格式写进去，不需要自己去修改模拟器或者terminal

□ 关于COM口地址的判断

- 由于COM口占用了一段地址空间，可以在处理器内部做特判，在访问这块内存的时候去控制串口或者去完成串口读写，而不是交给内存

□ 处理器的启动过程

- 处理器的启动过程可以从FLASH中装入监控程序，方法是在启动的时候设置启动状态，读入监控程序，在读入完成的时候再去执行指令

一些提示和帮助，供参考（2）

□ 内存读写时序

- 内存读写的时候，WE有一个拉低的过程，可以自行定义时钟的起始位置，使用时钟信号来设置WE（获得一个下降沿），在当前周期完成内存的读写工作
- 寄存器也可以响应下降沿来进行数据更新

□ 指令和数据分开（在MEM阶段可能需要插入气泡）

- 一般来说，指令和数据分开是可以帮助流水线不断流的。但是，监控程序会将用户程序放到代码的内存中，在这个时候会发生结构冲突（写的时候）。因此，需要插入NOP来解决这个冲突。（正常情况下，两块内存同时使用，不需要插入气泡）

□ 时钟信号的设置

- 时钟信号的设置与设计的流水线处理器密切相关，可以逐步提高，在主板上有模拟时钟信号的按钮，可以提供足够长的时钟周期

一些提示和帮助，供参考（3）

- ❑ 一开始把整个流程想清楚非常重要，否则后期在进行修正会非常繁琐以及崩溃，因此前期不要怕花时间把整个流程想清楚
- ❑ 按照指令分类，相同的指令先去设计数据通路，会降低数据通路设计的难度
- ❑ 指令应该是按产生的各种信号进行分类的，各种类别的指令也会相互影响