MIPSfpga

by Imagination

概要

- **材料组织**
- **MIPS体系结构的历史**
- **MIPSfpga**
  - 背景
  - 内核和系统
  - 接口
    - 系统接口
    - AHB-Lite总线
    - EJTAG

概要

- **材料组织**
- **MIPS体系结构的历史**
- **MIPSfpga**
  - 背景
  - 内核和系统
  - 接口
    - 系统接口
    - AHB-Lite总线
    - EJTAG

实验概览

- **实验 1:** 创建一个 MIPSfpga的Vivado 项目
- **实验2:** 使用Codescape用C语言对MIPSfpga编程
- **实验 3:** 使用 Codescape用汇编语言对MIPSfpga编程
- **实验 4:** MIPSfpga的更多编程实践
- **实验 5:** 添加外围设备: 7 段数码管显示
- **实验 6:** 添加外围设备: 毫秒计时器
- **实验 7:** 添加外围设备: 蜂鸣器
- **实验 8:** 添加外围设备: SPI 液晶显示屏
- **实验 9:** 将MIPSfpga移植到其它 FPGA 板上

概览

- 材料组织
- **MIPS体系结构的历史**
- MIPSfpga
  - 背景
  - 内核和系统
  - 接口
    - 系统接口
    - AHB-Lite总线
    - EJTAG

MIPS体系结构的历史

- 在上世纪 80 年代由**John Hennessy**和他的同事在斯坦福大学提出
- 第一批商业**精简指令集计算机**(RISC) 架构中的一员
- Hennessy共同创立 MIPS 计算机系统 — — 后来被称为 MIPS 科技
- 在许多商业系统中使用，包括硅图工作站、任天堂机和思科服务器
- 在多数大学中被研究
- 出售超过 50亿的 MIPS 微处理器

MIPS体系结构的历史

- Imagination科技在 2013 年 2 月收购了 MIPS 科技
  - 公司总部设在英国
  - 其他产品包括: PowerVR 移动图形处理器，消费类电子产品和音频设备

MIPS内核

- **MIPS R3000, R4000, R10000**
  - 1980's 和1990's
  - 例如，硅图工作站中使用
- **Embedded: M4K, M14K**
  - 例如， Microchip的PIC32微控制器就是基于M4K内核的

MIPS内核

- **microAptiv**
  - 高效、简洁、嵌入式内核
  - 基于M14K架构
- **interAptiv, proAptiv**
  - 更高的性能
  - 多处理器、超标量体系结构、多线程
- **Warrior**
  - 最新的Imagination MIPS 核产品线
  - 属于高性能嵌入式内核范畴

MIPS内核

- **microAptiv :MIPSfpga 采用microAptiv 核**
  - 高效、简洁、嵌入式内核
  - 基于M14K架构
- **interAptiv, proAptiv**
  - 更高的性能
  - 多处理器、超标量体系结构、多线程
- **Warrior**
  - 最新的Imagination MIPS 核产品线
  - 属于高性能嵌入式内核范畴

MIPSfpga概览

- MIPS 体系结构的历史
- **MIPSfpga**
  - 背景
  - 内核和系统
  - 接口
    - 系统接口
    - AHB-Lite总线
    - EJTAG
    - FPGA 板

MIPSfpga背景

MIPSfpga是什么？

- 是一个在 FPGA 上实现的商业 MIPS 处理器软核
- Imagination Technologies公司提供给高校使用

MIPSfpga协议

- 只供**学术用途**
- **每个大学教师必须注册**以获得对 MIPSfpga的使用 (不通过教师之间传递)

  http://community.imgtec.com/university/university-registration
- 最好**不放在硅芯片中**
- 任何**出版物**应承认 MIPSfpga
- 把工作结果的**副本**发给**Imagination** — — 他们有兴趣看你都做了什么!

MIPSfpga： microAptiv 内核

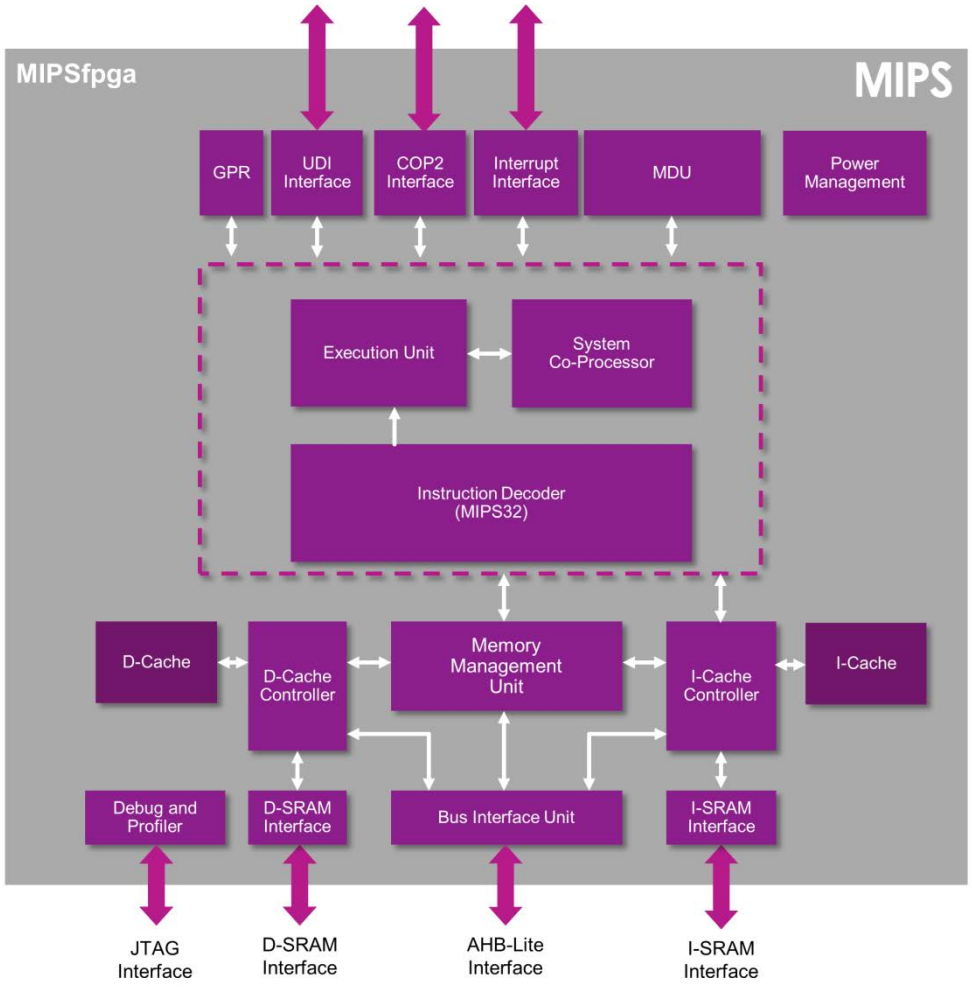
商业 microAptiv 内核

- 5 级流水线
- 1.5 Dhrystone MIPS/MHz
- 2 路相联的指令和数据Cache，每个 2 KB
- 拥有 16 项 TLB 的MMU (内存管理单元)

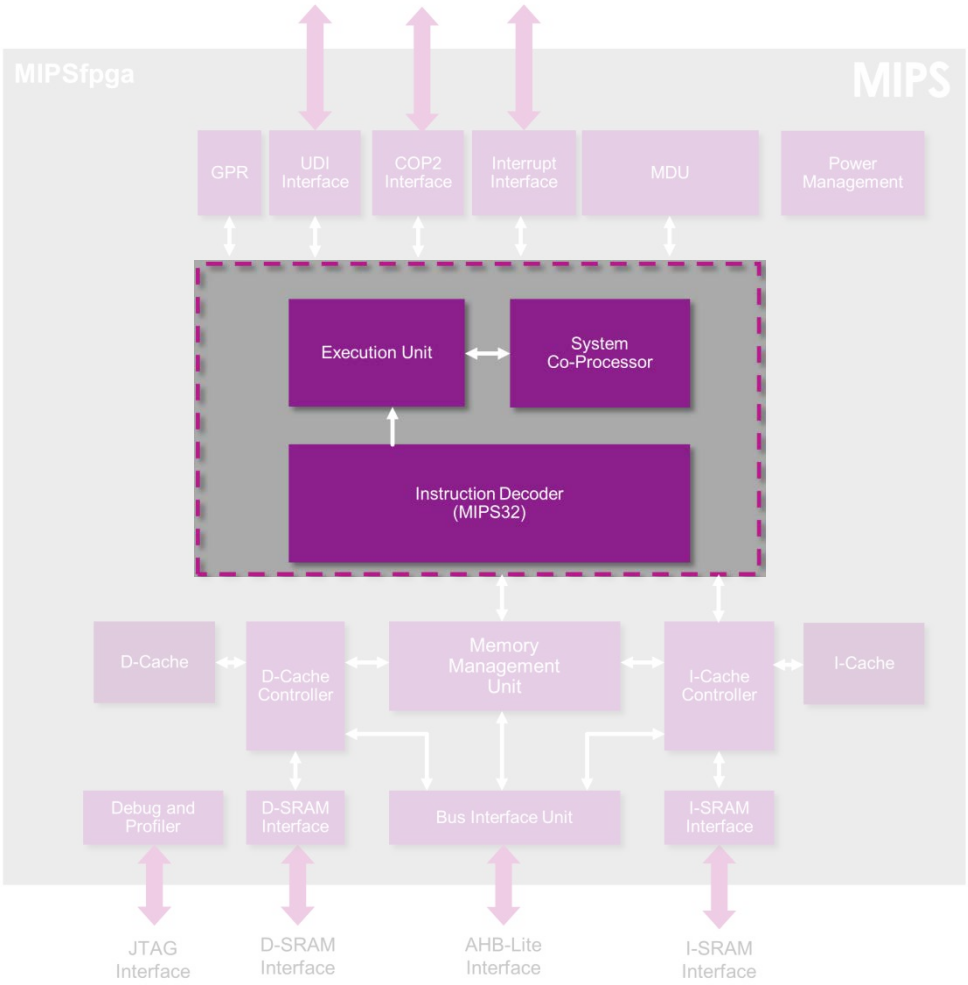
MIPSfpga 概览

- MIPS架构的历史
- **MIPSfpga**
  - 背景
  - **内核和系统**
  - 接口
    - 系统接口
    - AHB-Lite总线
    - EJTAG
    - FPGA 板

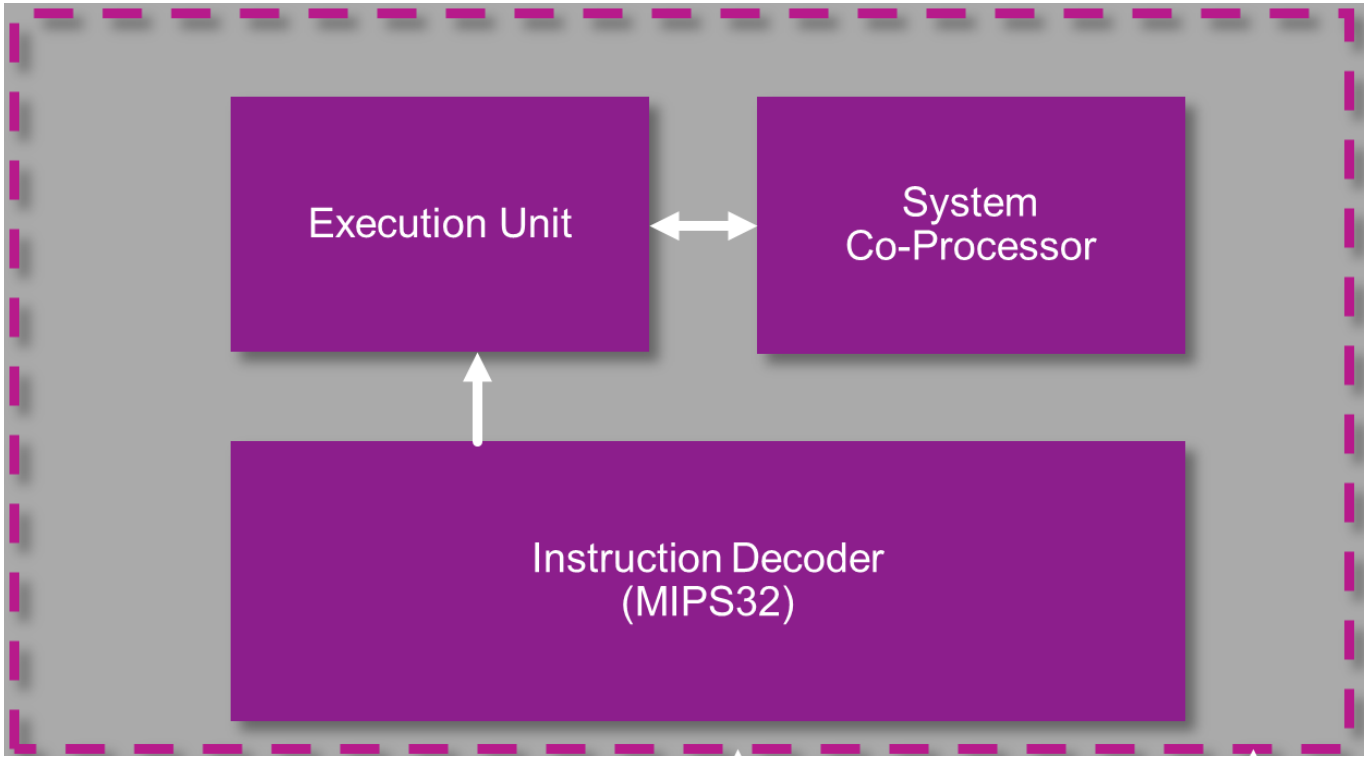
MIPS内核


MIPSfpga
MIPS
GPR
UDI Interface
COP2 Interface
Interrupt Interface
MDU
Power Management
Execution Unit
System Co-Processor
Instruction Decoder (MIPS32)
D-Cache
D-Cache Controller
Memory Management Unit
I-Cache Controller
I-Cache
Debug and Profiler
D-SRAM Interface
Bus Interface Unit
I-SRAM Interface
JTAG Interface
D-SRAM Interface
AHB-Lite Interface
I-SRAM Interface

MIPS内核

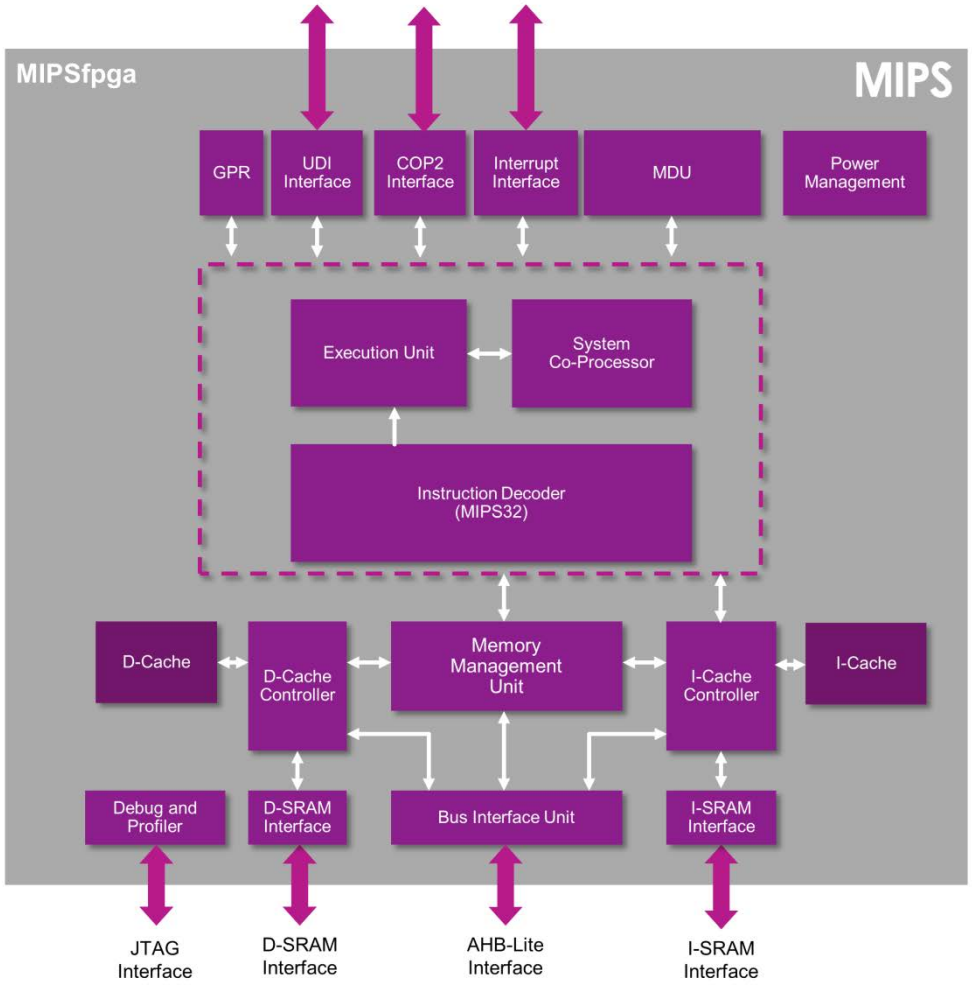

MIPSfpga
MIPS
GPR
UDI Interface
COP2 Interface
Interrupt Interface
MDU
Power Management
Execution Unit
System Co-Processor
Instruction Decoder (MIPS32)
D-Cache
D-Cache Controller
Memory Management Unit
I-Cache Controller
I-Cache
Debug and Profiler
D-SRAM Interface
Bus Interface Unit
I-SRAM Interface
JTAG Interface
D-SRAM Interface
AHB-Lite Interface
I-SRAM Interface

MIPS内核



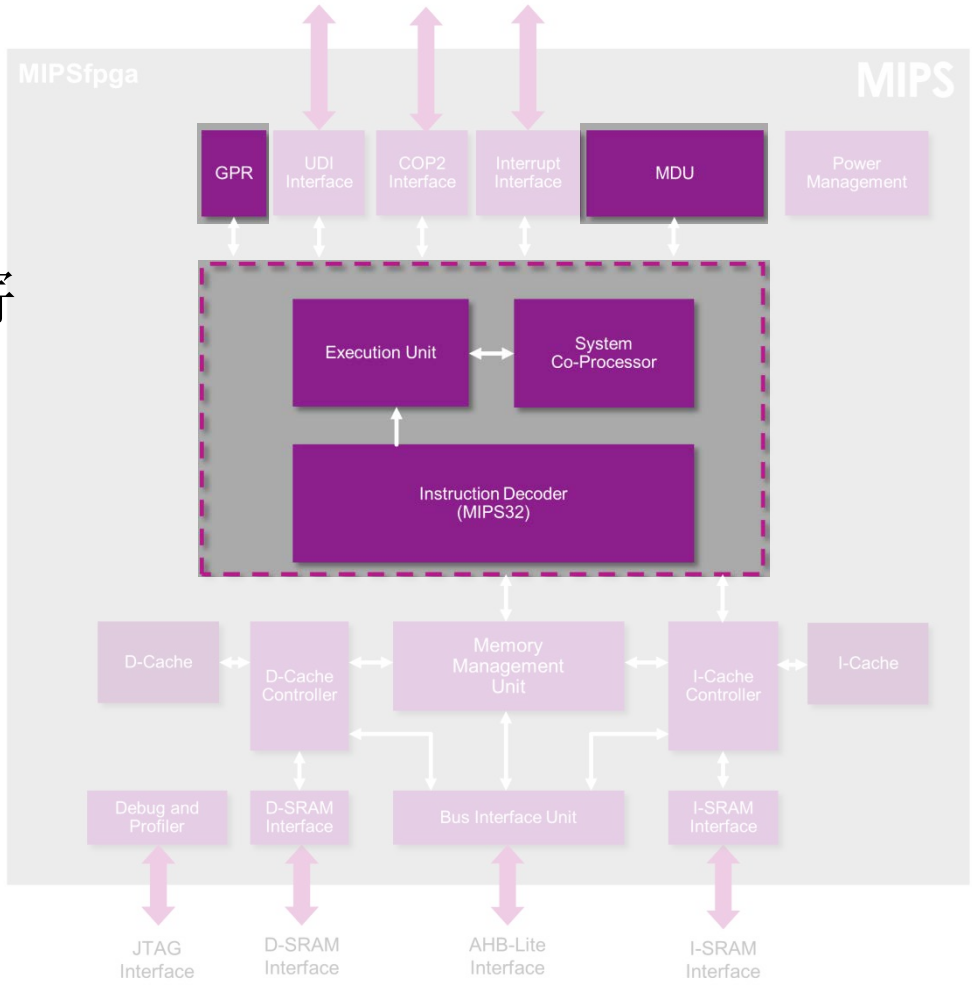
- 处理指令
- 协处理器：系统寄存器，复位处理

MIPSfpga:寄存器, MDU



MIPSfpga:寄存器, MDU

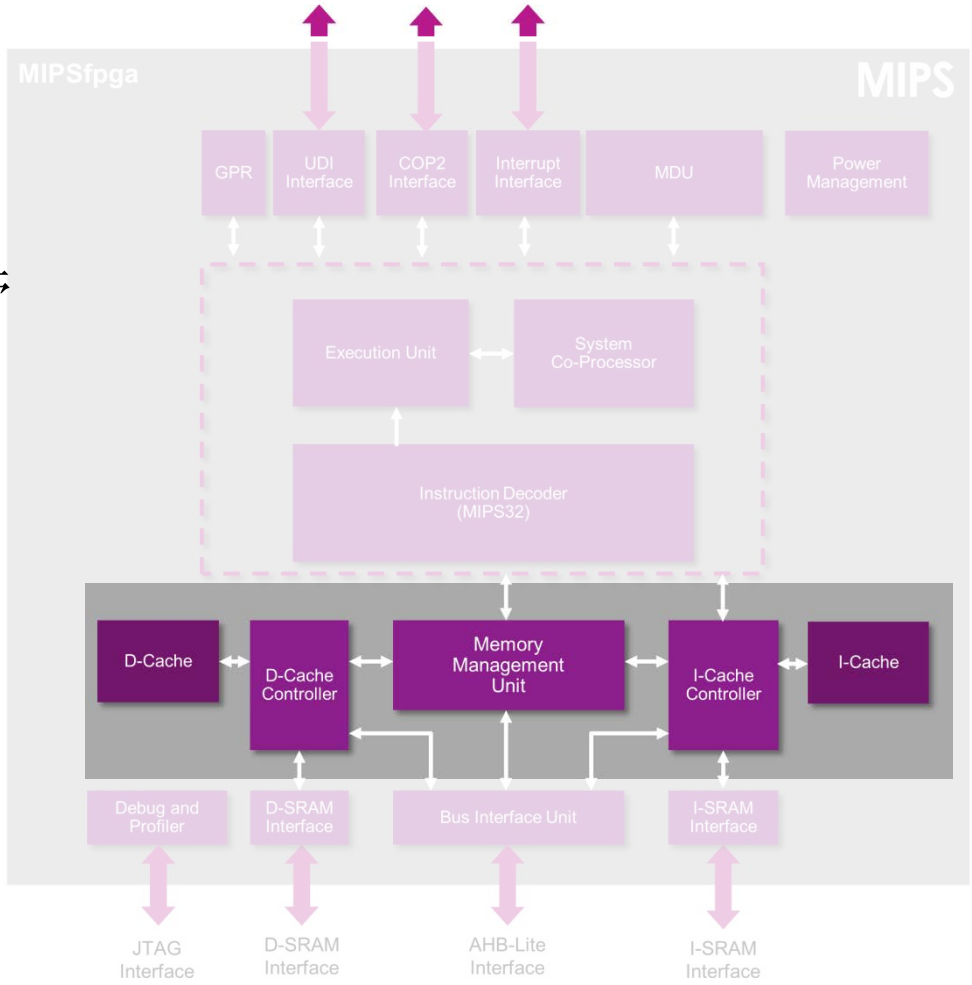
GPR: 通用寄存器



MDU: 乘/除法单元

MIPSfpga: MMU,缓存

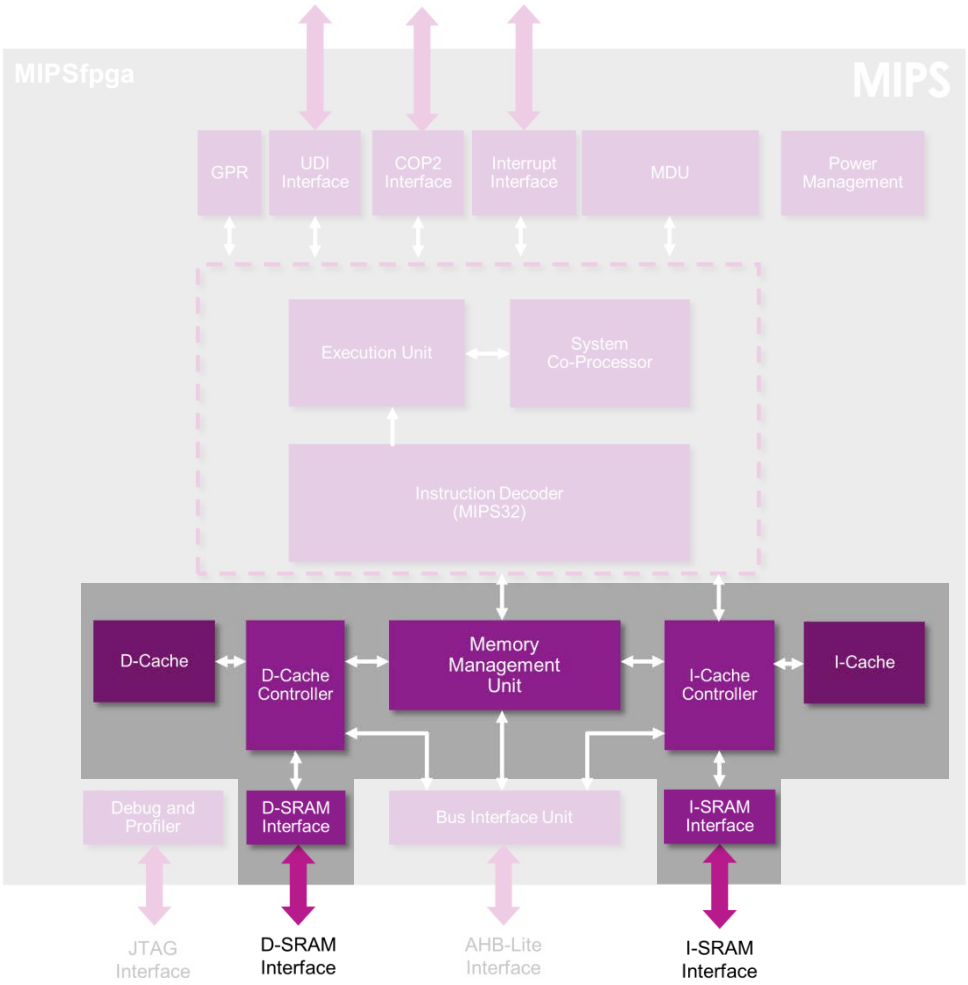
MMU: 存储管理单元



高速缓存（Cache）：指令和数据

高速缓存控制器：指令和数据缓冲

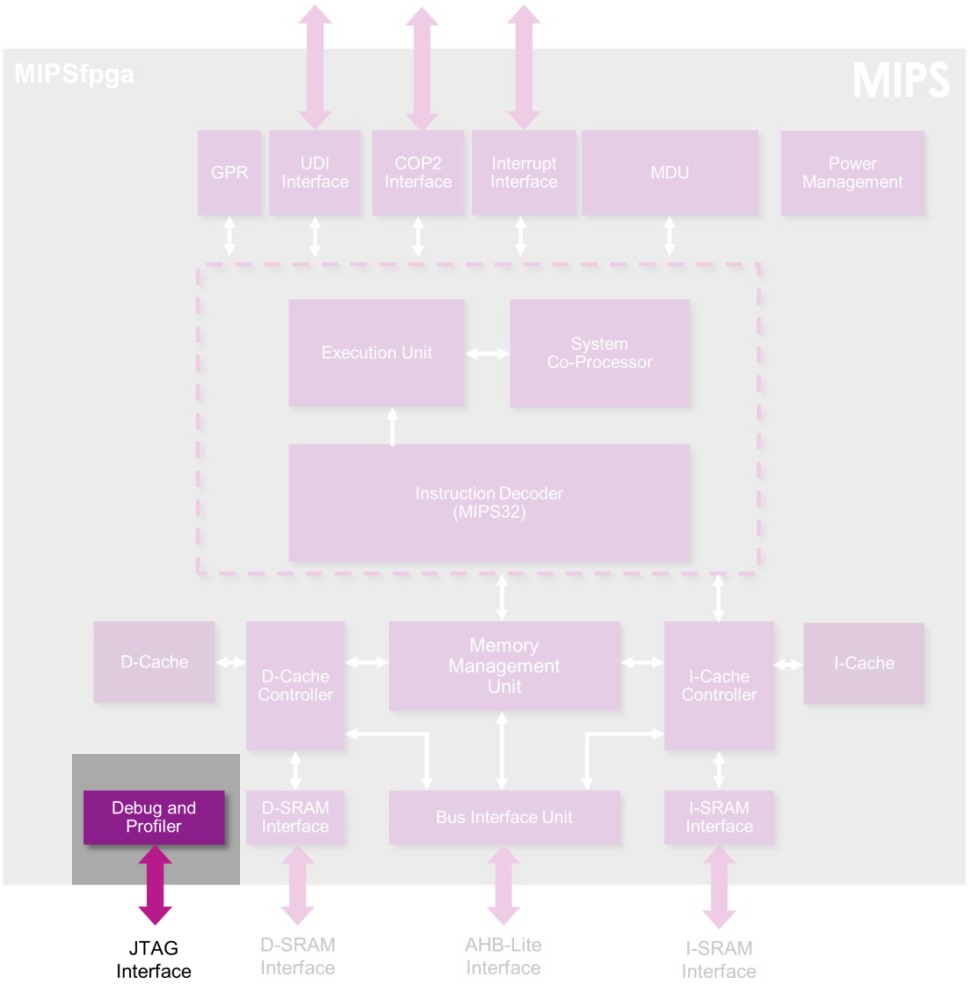
MIPSfpga: 缓存控制器



缓存控制器：指令和数据缓存的接口和外部存储器（称为scratch RAM，SRAM）

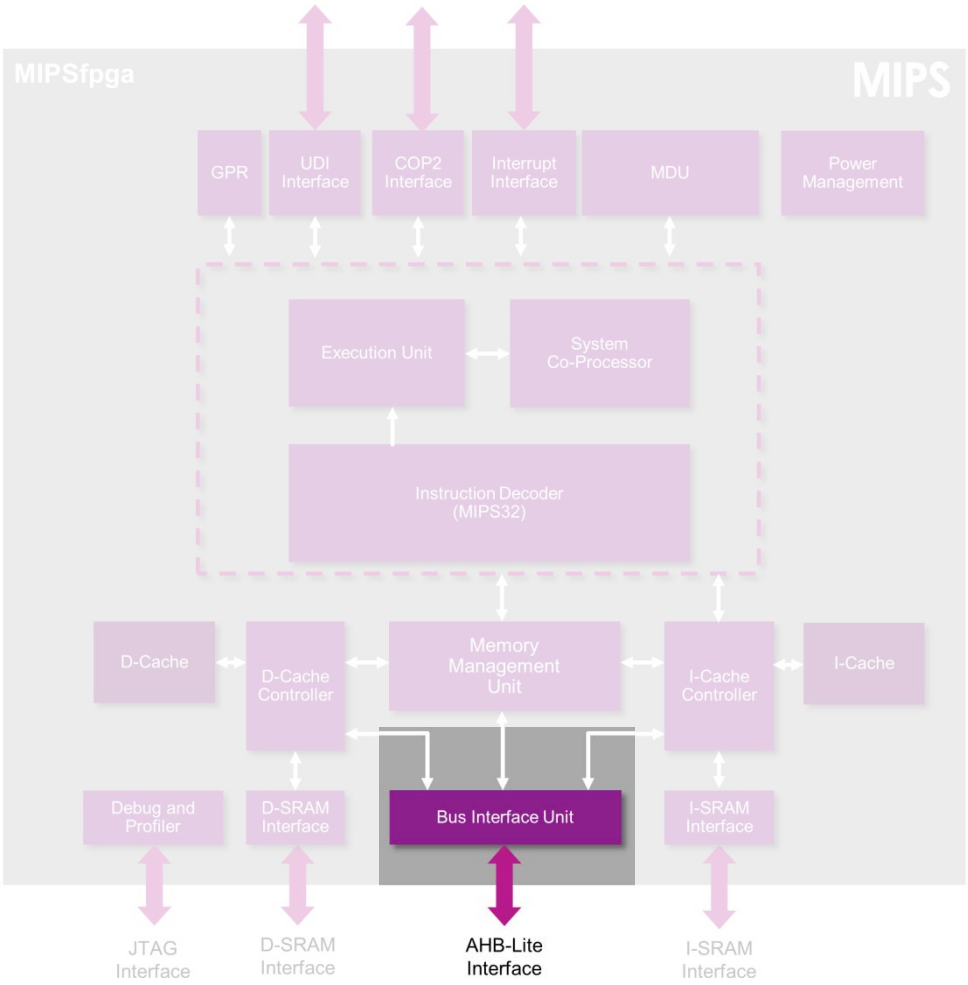
MIPSfpga (E)JTAG 接口

JTAG（也叫EJTAG）：
对内核进行编程和实时调试



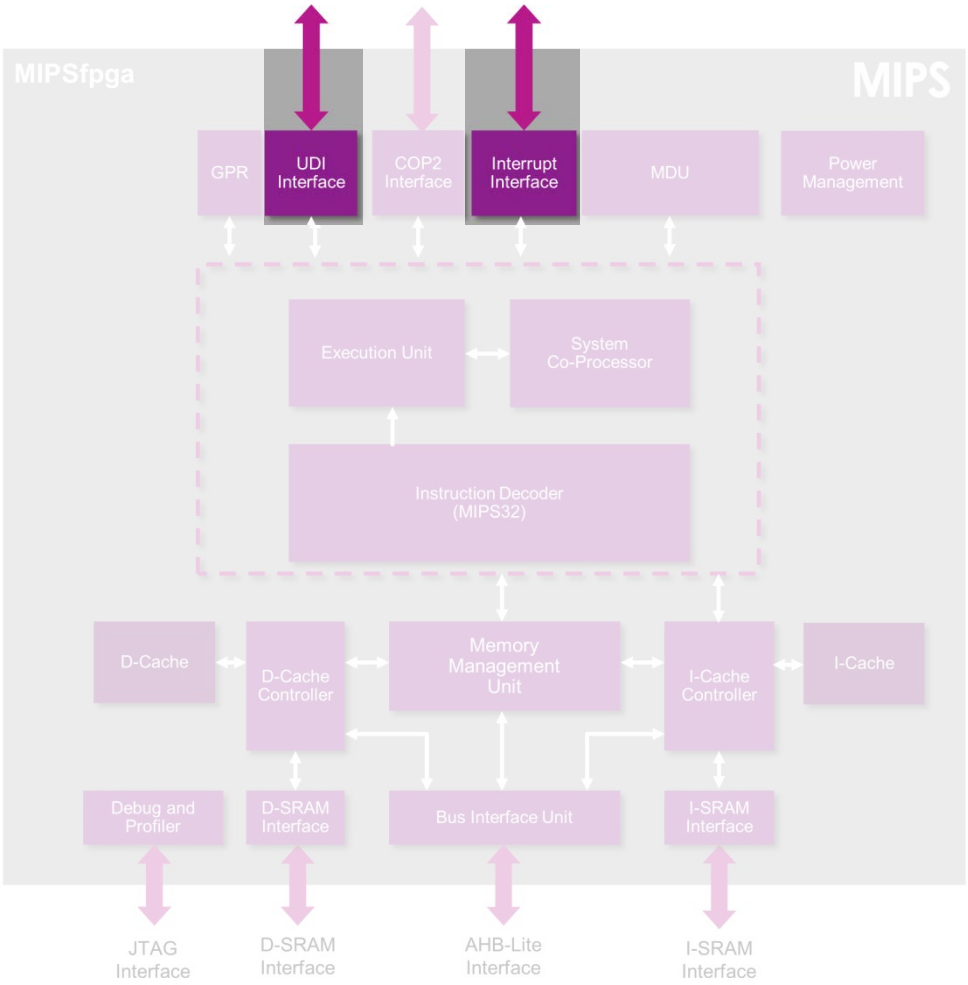
MIPSfpga AHB-Lite 总线

AHB-Lite总线：
用于内存和外围设备的交互



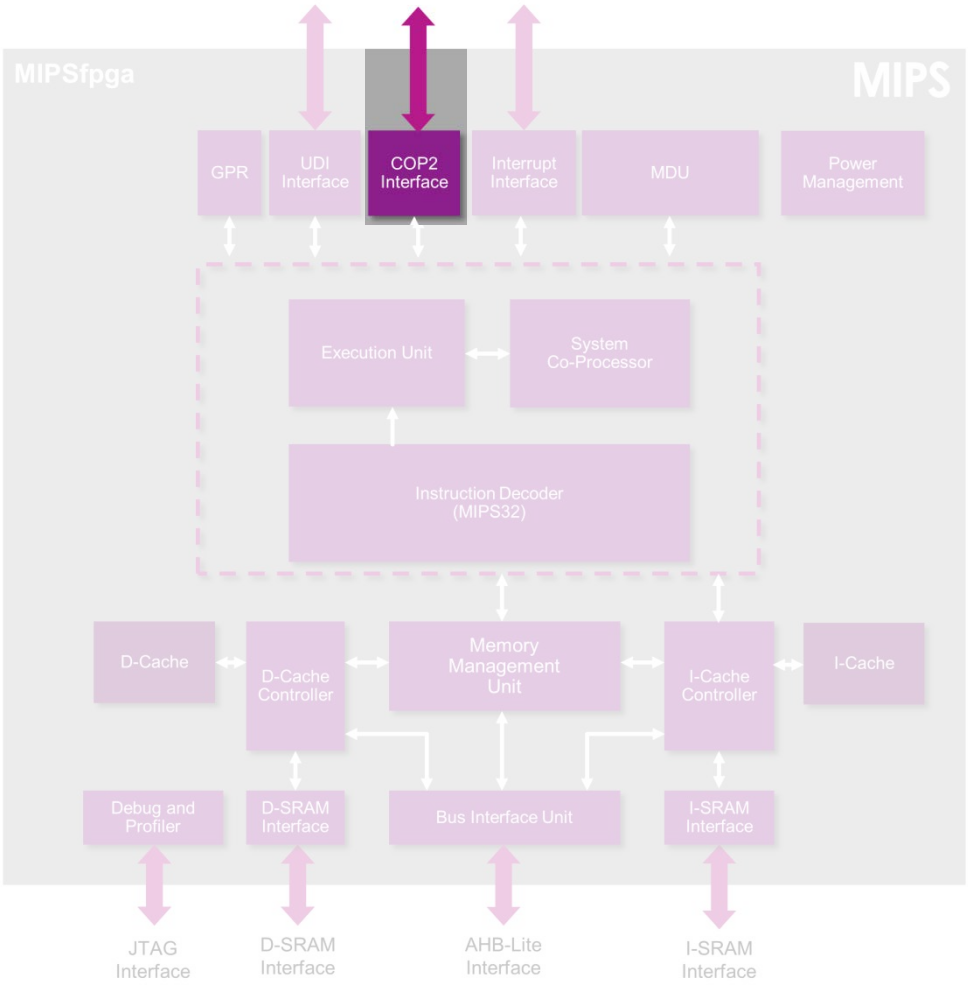
MIPSfpga UDI 接口, 中断

UDI（用户定义接口单元）：允许用户自定义指令



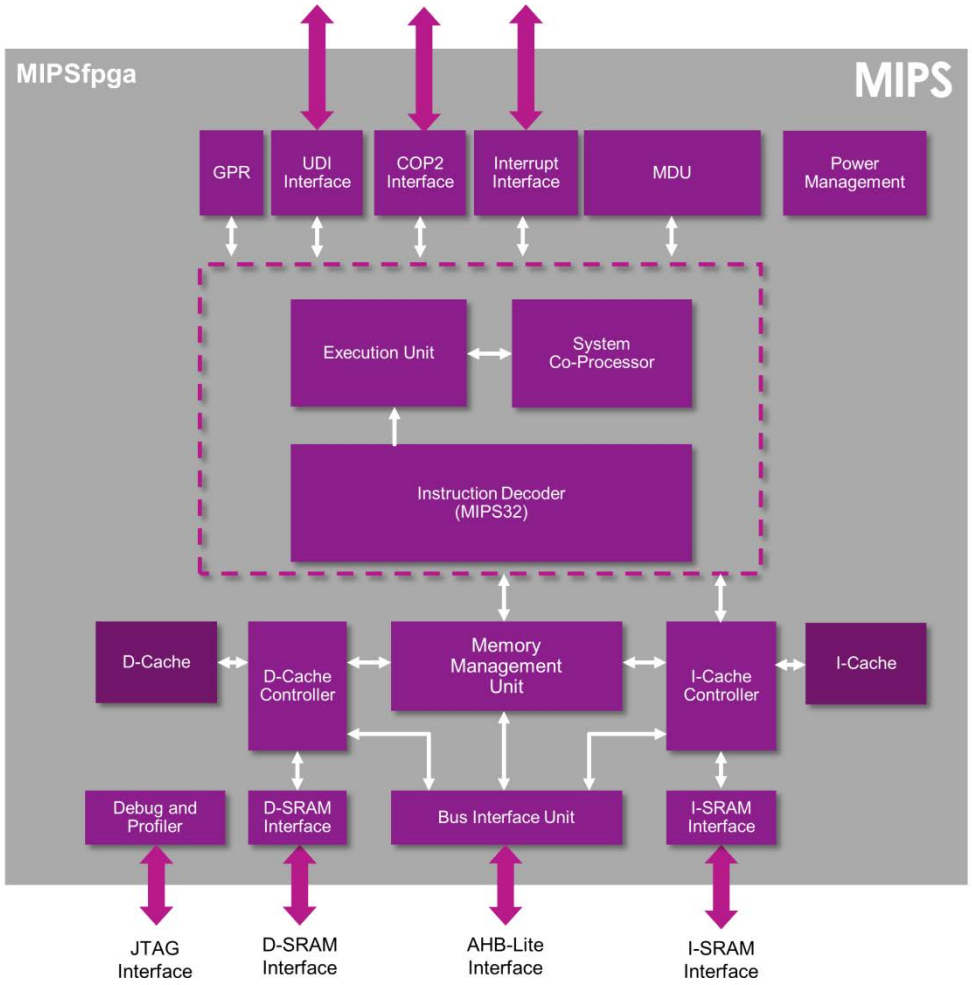
中断接口：用于硬件中断

MIPSfpga UDI 接口, 中断



COP2接口：
协处理器2的接口

MIPSfpga 内核


MIPSfpga
MIPS
GPR
UDI Interface
COP2 Interface
Interrupt Interface
MDU
Power Management
Execution Unit
System Co-Processor
Instruction Decoder (MIPS32)
D-Cache
D-Cache Controller
Memory Management Unit
I-Cache Controller
I-Cache
Debug and Profiler
D-SRAM Interface
Bus Interface Unit
I-SRAM Interface
JTAG Interface
D-SRAM Interface
AHB-Lite Interface
I-SRAM Interface

MIPSfpga 内核规格

- **商业 microAptiv 内核**
  - 5 级流水线
  - 4 KB 2 路组相联的指令和数据缓存
  - 16项TLB的MMU（内存管理单元）
  - 性能计数器、 输入同步器
  - 没有 DSP、 协处理器 2 或影子寄存器
  - 接口:
    - AHB-Lite总线
    - EJTAG编程器/调试器
    - 用于用户自定义指令的扩展

MIPSfpga五级流水

#	流水级	名称	描述
1	I	取指	取指令
2	E	执行	从RF取操作数&执行ALU操作
3	M	访存	访问内存
4	A	对齐	数据字边界对齐
5	W	回写	结果回写到RF

MIPSfpga操作模式

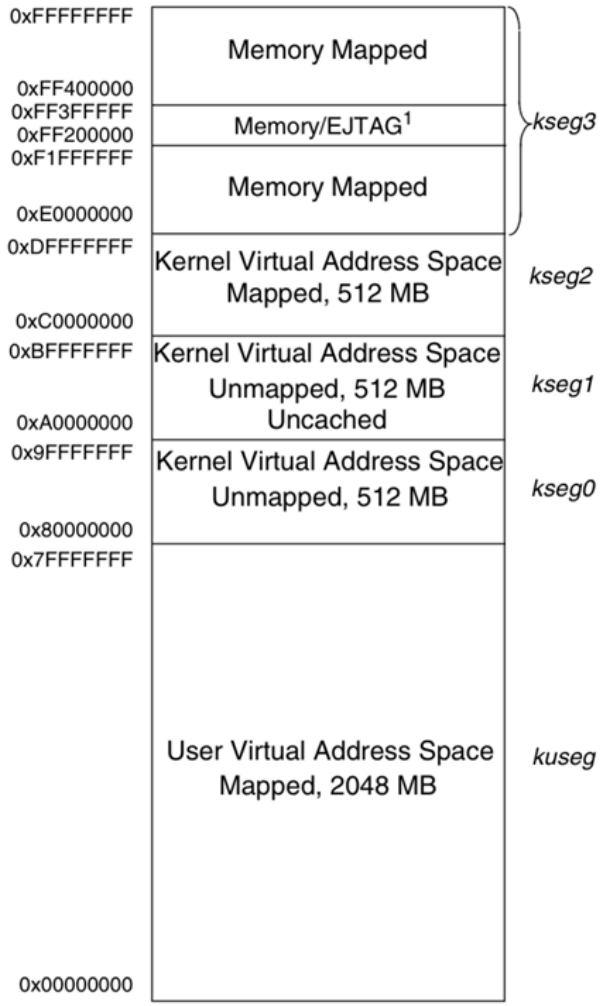
- 内核
- 用户
- 调试

MIPSfpga操作模式

- 内核
- 用户
- 调试

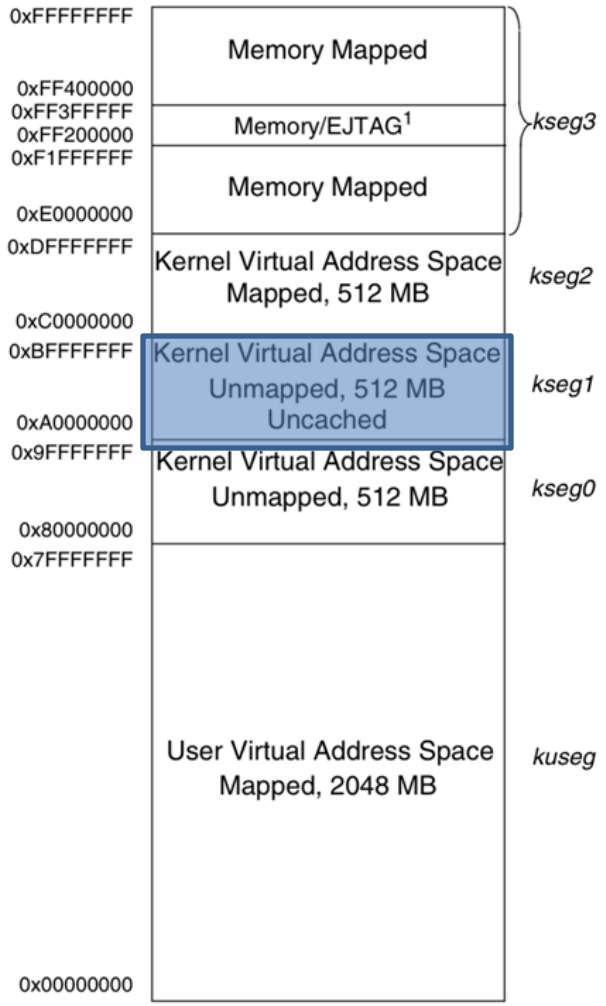
复位时，处理器在内核模式中开始，然后跳转到地址为 0xbfc00000的复位向量

MIPSfpga存储映射



- **32-bit**虚拟内存空间 (0x00000000 – 0xFFFFFFFF)
- 拆分为不同的段
- **kseg0** 和**kseg1** 都映射到以0x0开始的物理地址, 即:
  - **0xA0000000** 映射到物理地址 **0x00000000**
  - **0xBFC00000** => **0x1FC00000**
  - **0x80000000** => **0x00000000**

MIPSfpga存储映射

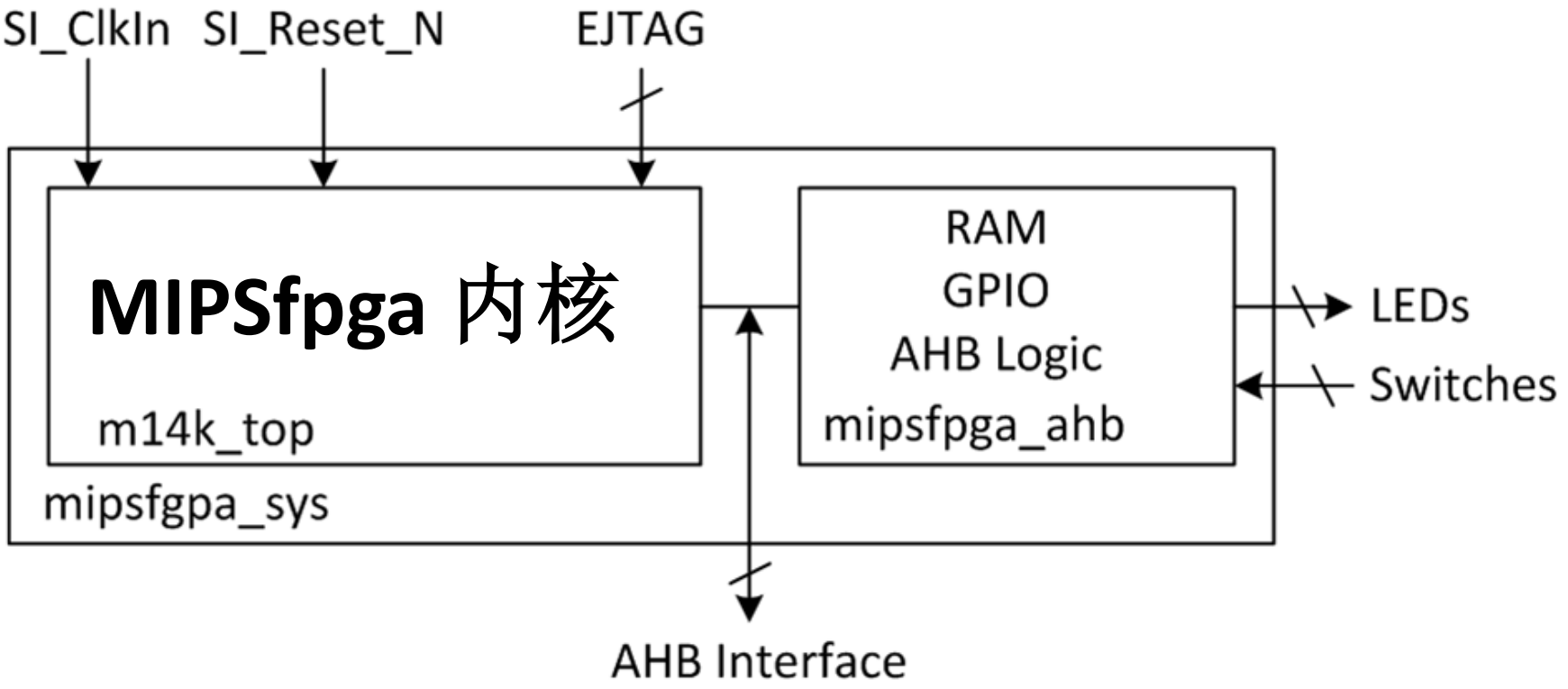


- 复位时，处理器在内核模式中开始，然后跳转到地址为**0xBFC00000**的复位向量
- 在**kseg1**：在TLB中没有缓存映射和地址映射（此时还没有初始化）
  - 所有指令**取自外存**（而不是缓存）
  - **0xBFC00000**映射到物理地址**0x1FC00000**

MIPSfpga概览

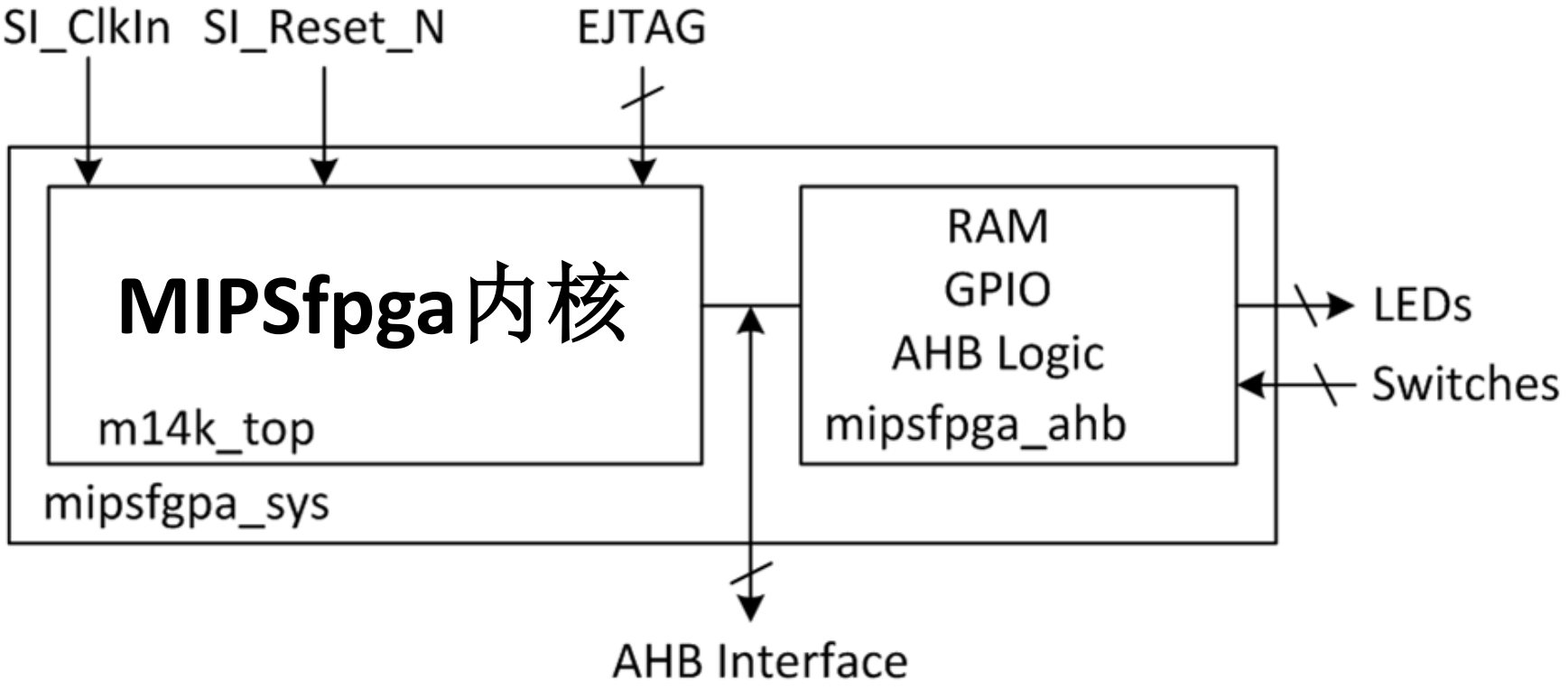
- MIPS架构的历史
- **MIPSfpga**
  - 背景
  - 内核和**系统**
  - I接口
    - 系统接口
    - AHB-Lite 总线
    - EJTAG

MIPSfpga 系统

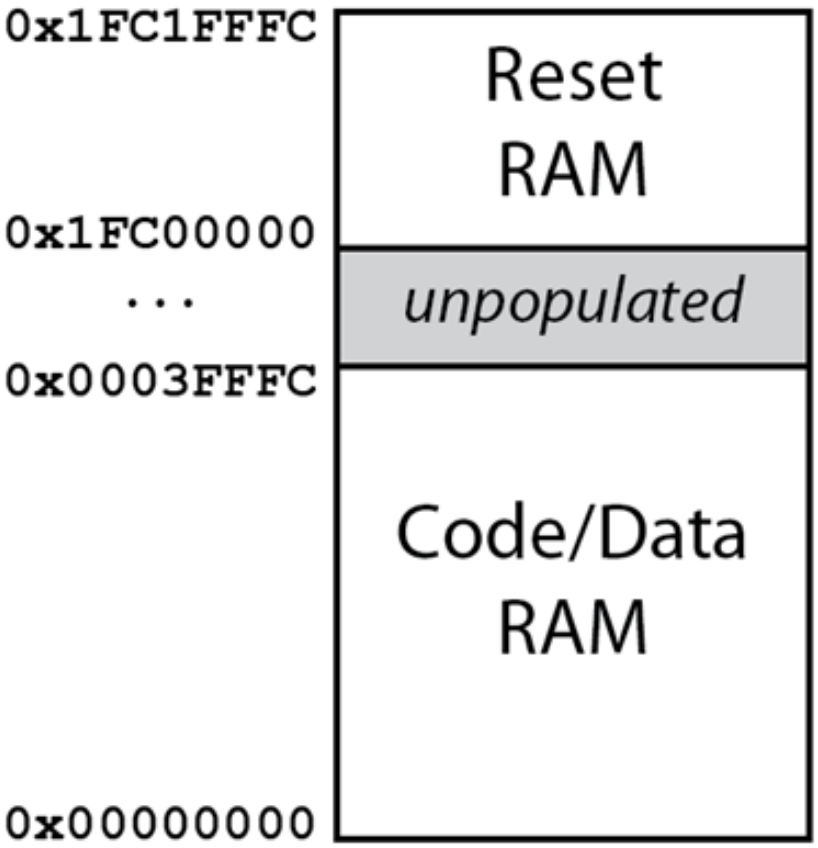

SI_ClkIn
SI_Reset_N
EJTAG
MIPSfpga 内核
m14k_top
RAM
GPIO
AHB Logic
mipsfpga_ahb
LEDs
Switches
mipsfgpa_sys
AHB Interface

MIPSfpga System

RAM: 128 KB (引导代码)
256 KB (用户代码)



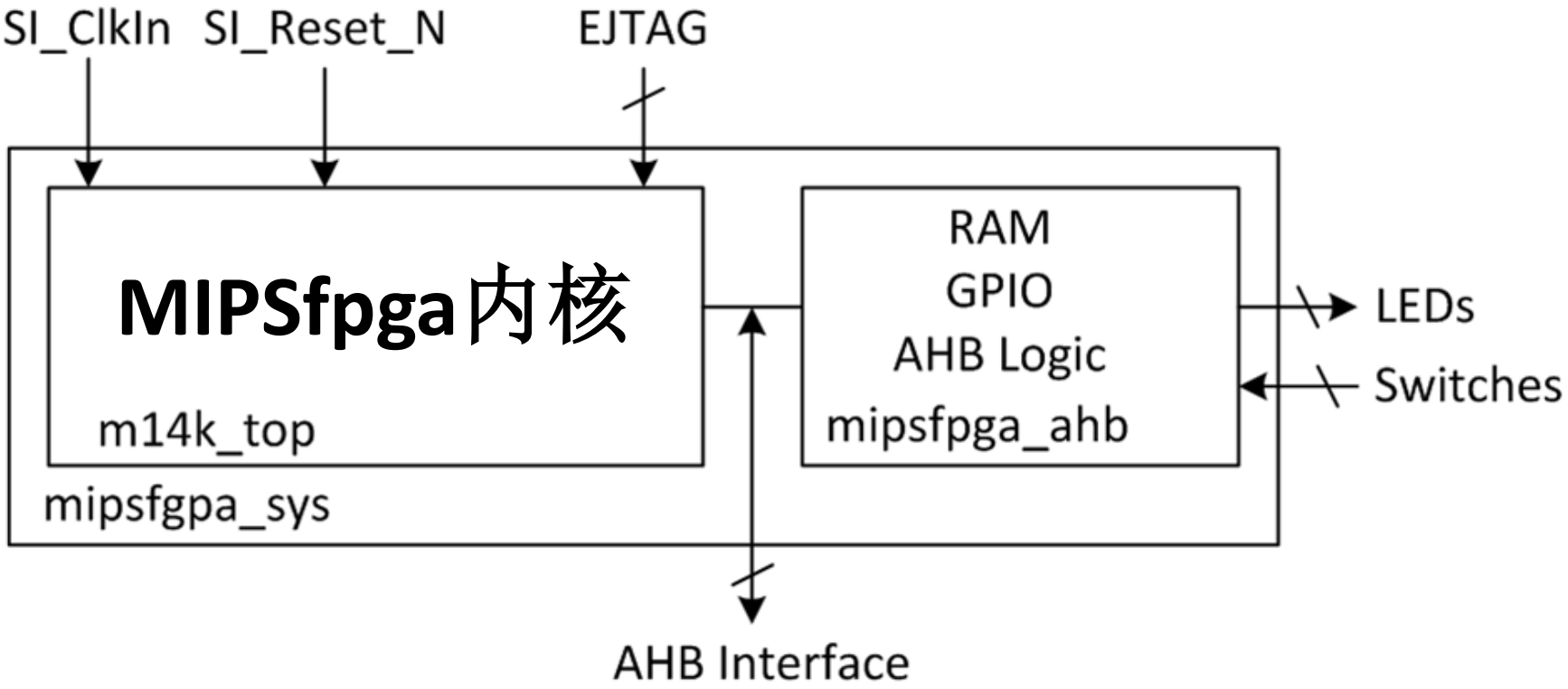
MIPSfpga 系统: 物理内存



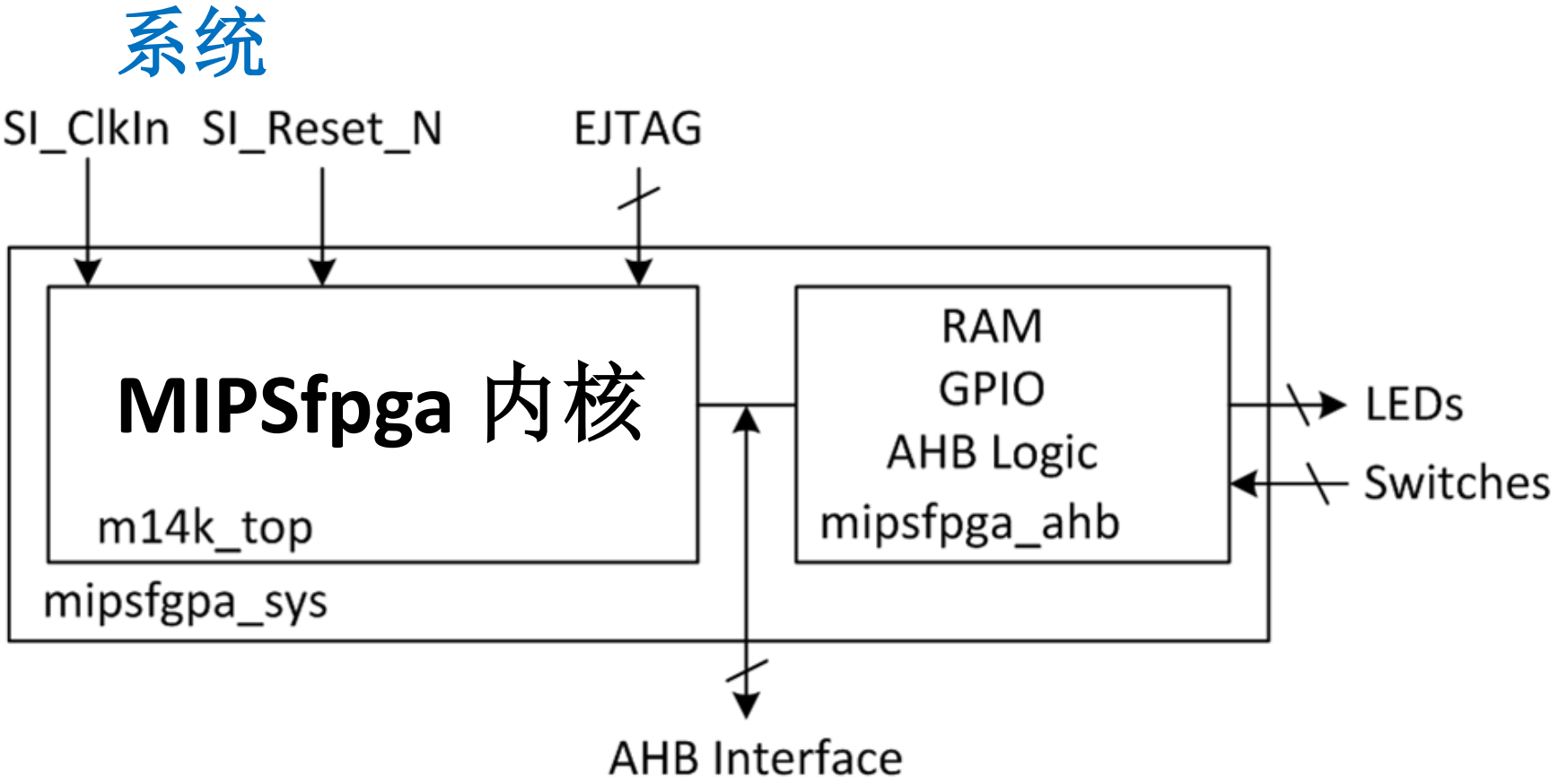
引导代码:
系统启动时执行(128 KB)

用户代码/数据
(256 KB)

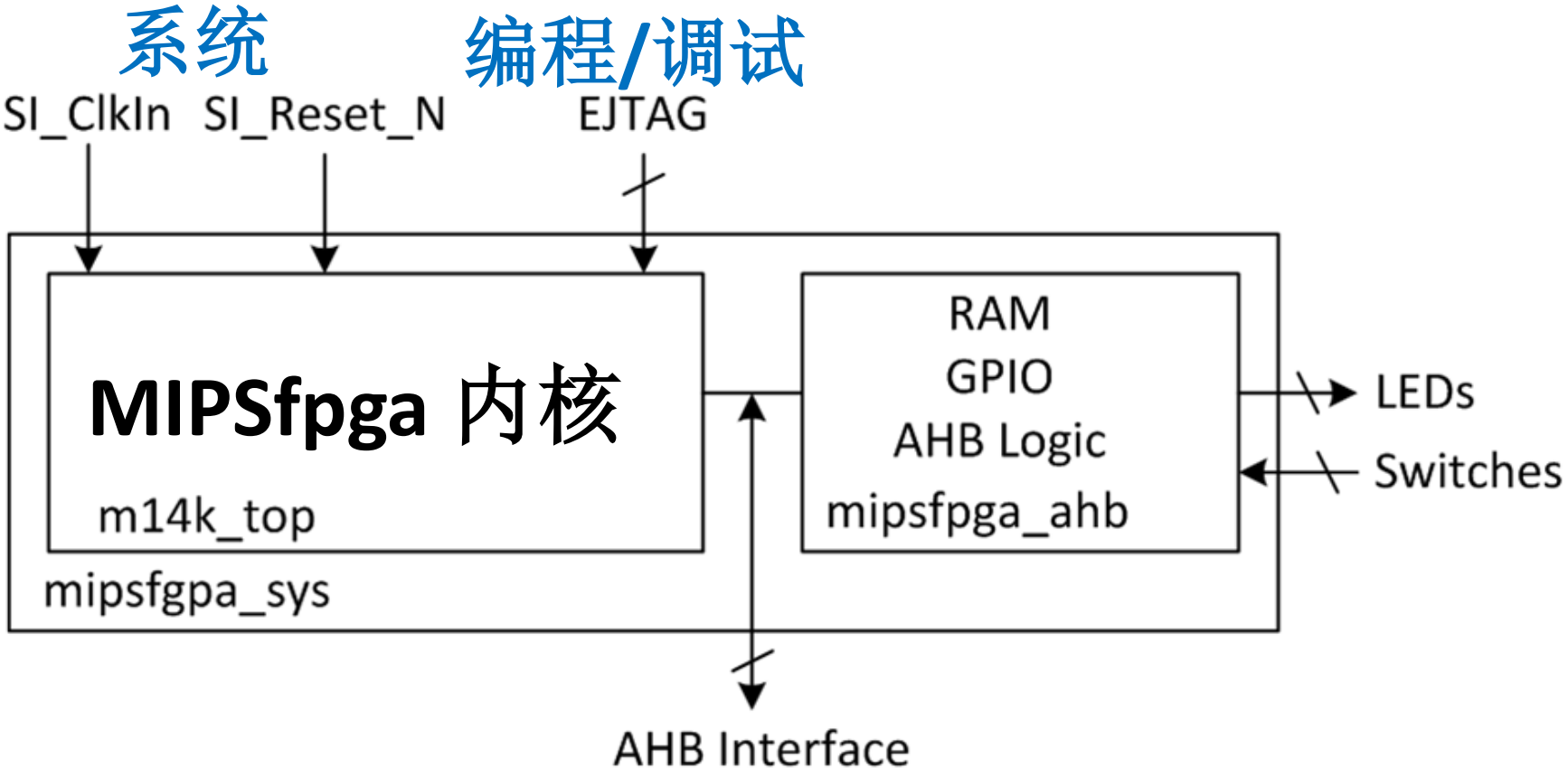
MIPSfpga 系统


SI_ClkIn
SI_Reset_N
EJTAG
MIPSfpga内核
m14k_top
mipsfgpa_sys
RAM
GPIO
AHB Logic
mipsfpga_ahb
LEDs
Switches
AHB Interface

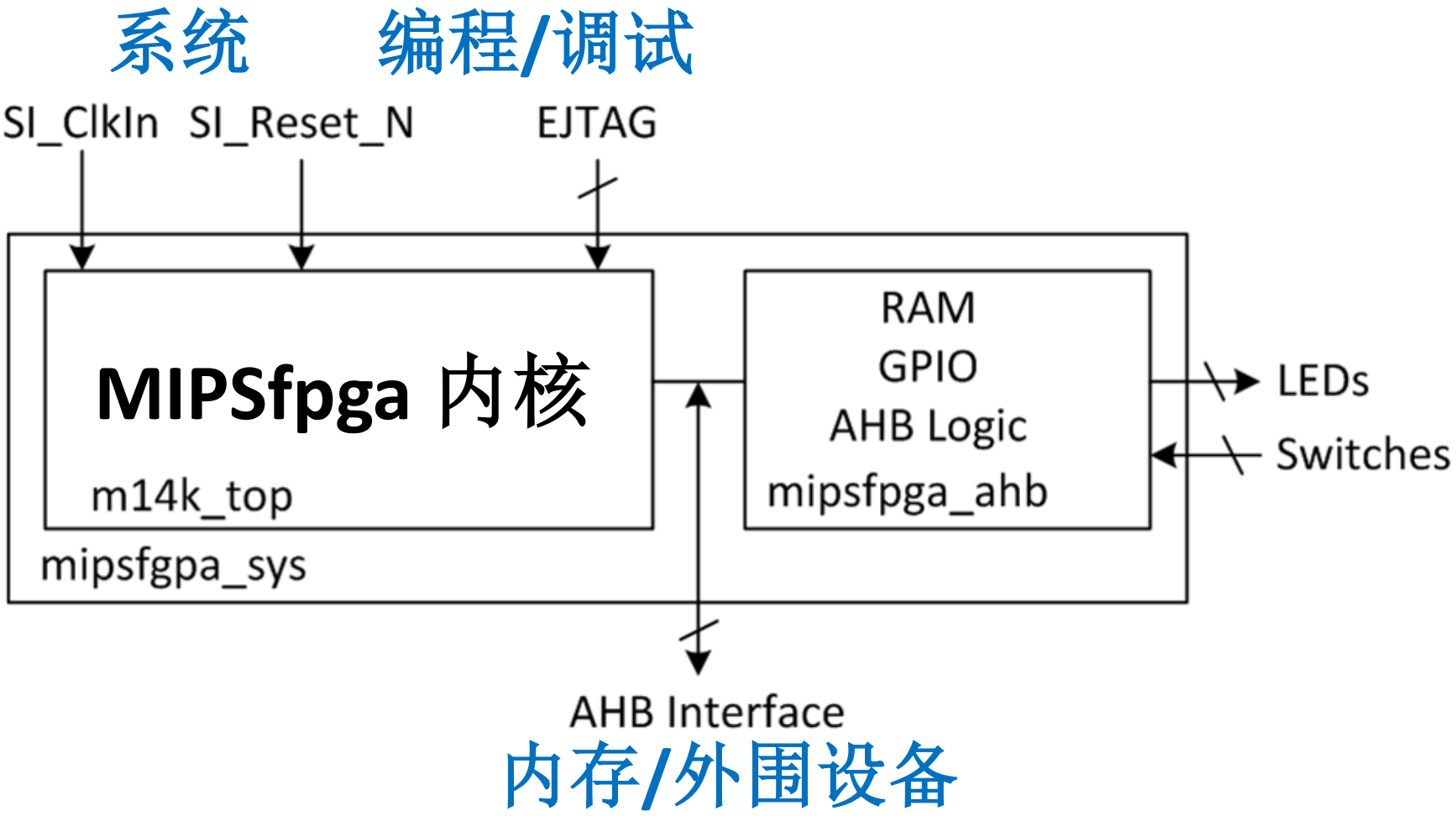
MIPSfpga 系统


系统
SI_ClkIn
SI_Reset_N
EJTAG
MIPSfpga 内核
m14k_top
mipsfgpa_sys
RAM
GPIO
AHB Logic
mipsfpga_ahb
LEDs
Switches
AHB Interface

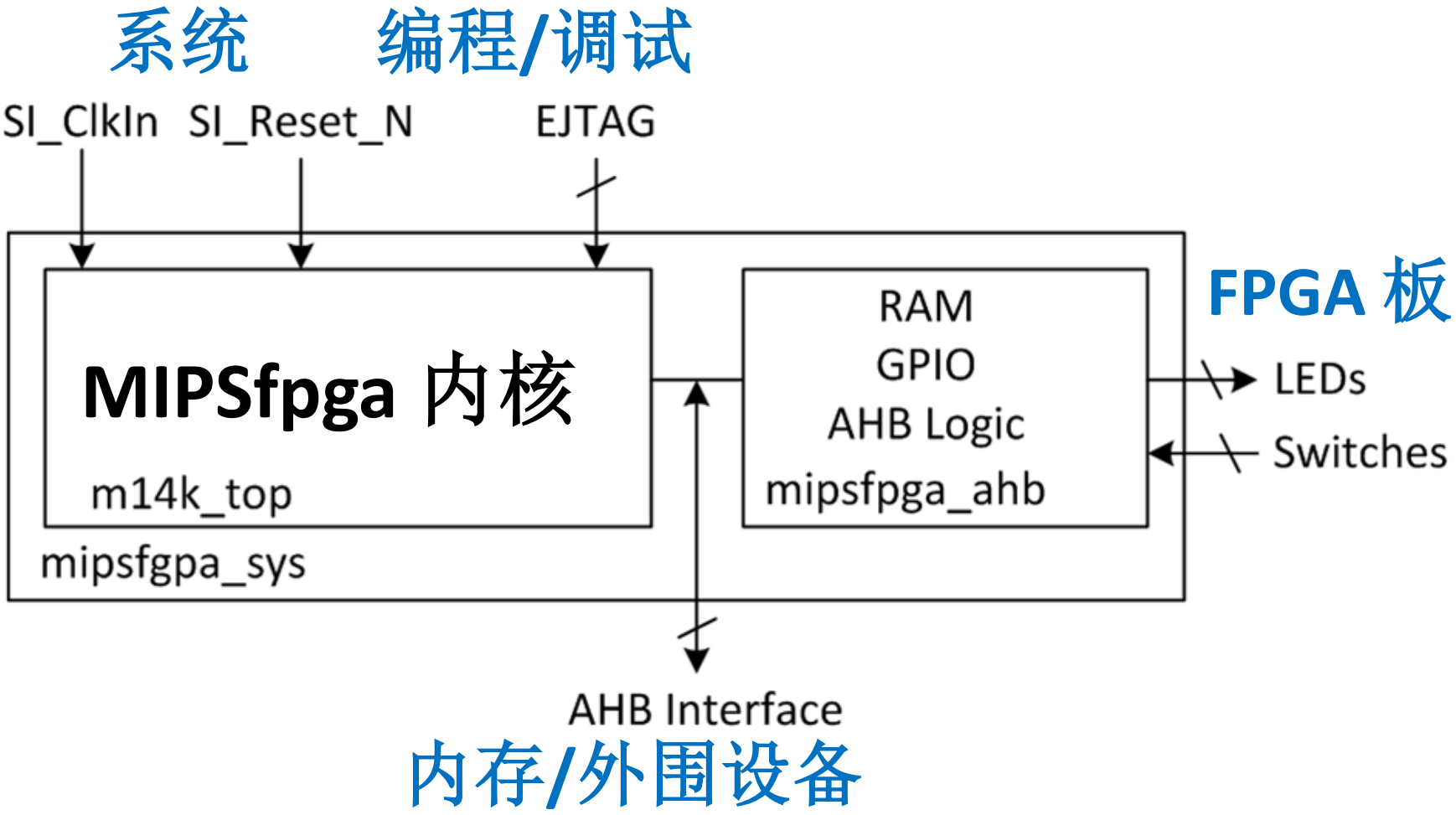
MIPSfpga 系统


系统
编程/调试
SI_ClkIn
SI_Reset_N
EJTAG
MIPSfpga 内核
m14k_top
mipsfgpa_sys
RAM
GPIO
AHB Logic
mipsfpga_ahb
LEDs
Switches
AHB Interface

MIPSfpga 系统


系统
编程/调试
SI_ClkIn
SI_Reset_N
EJTAG
MIPSfpga 内核
m14k_top
mipsfgpa_sys
RAM
GPIO
AHB Logic
mipsfpga_ahb
LEDs
Switches
AHB Interface
内存/外围设备

MIPSfpga 系统


系统
编程/调试
SI_ClkIn
SI_Reset_N
EJTAG
MIPSfpga 内核
m14k_top
mipsfgpa_sys
RAM
GPIO
AHB Logic
mipsfpga_ahb
FPGA 板
LEDs
Switches
AHB Interface
内存/外围设备

MIPSfpga 概览

- MIPS架构的历史
- **MIPSfpga**
  - 背景
  - 内核和系统
  - **接口**
    - **系统接口**
    - **AHB-Lite 总线**
    - **EJTAG**

MIPSfpga 概览

- MIPS架构的历史
- **MIPSfpga**
  - 背景
  - 内核和系统
  - 接口
    - 系统接口
    - **AHB-Lite 总线**
    - **EJTAG**

MIPSfpga 接口: 系统

信号名	描述	Nexys4 DDR 板
SI_Reset_N	为0时处理器复位	CPU复位按钮
SI_ClkIn	系统时钟	50 MHz (产生于板载100MHz时钟)

SI：Verilog文件中系统接口信号的前缀

MIPSfpga 概览

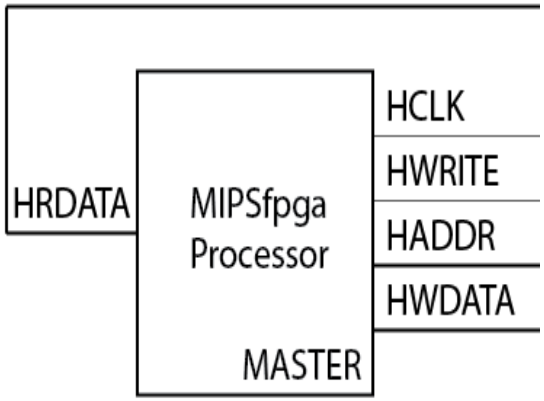
- MIPS架构的历史
- **MIPSfpga**
  - 背景
  - 内核和系统
  - 接口
    - 系统接口
    - **AHB-Lite 总线**
    - **EJTAG**

MIPSfpga 接口: AHB-Lite

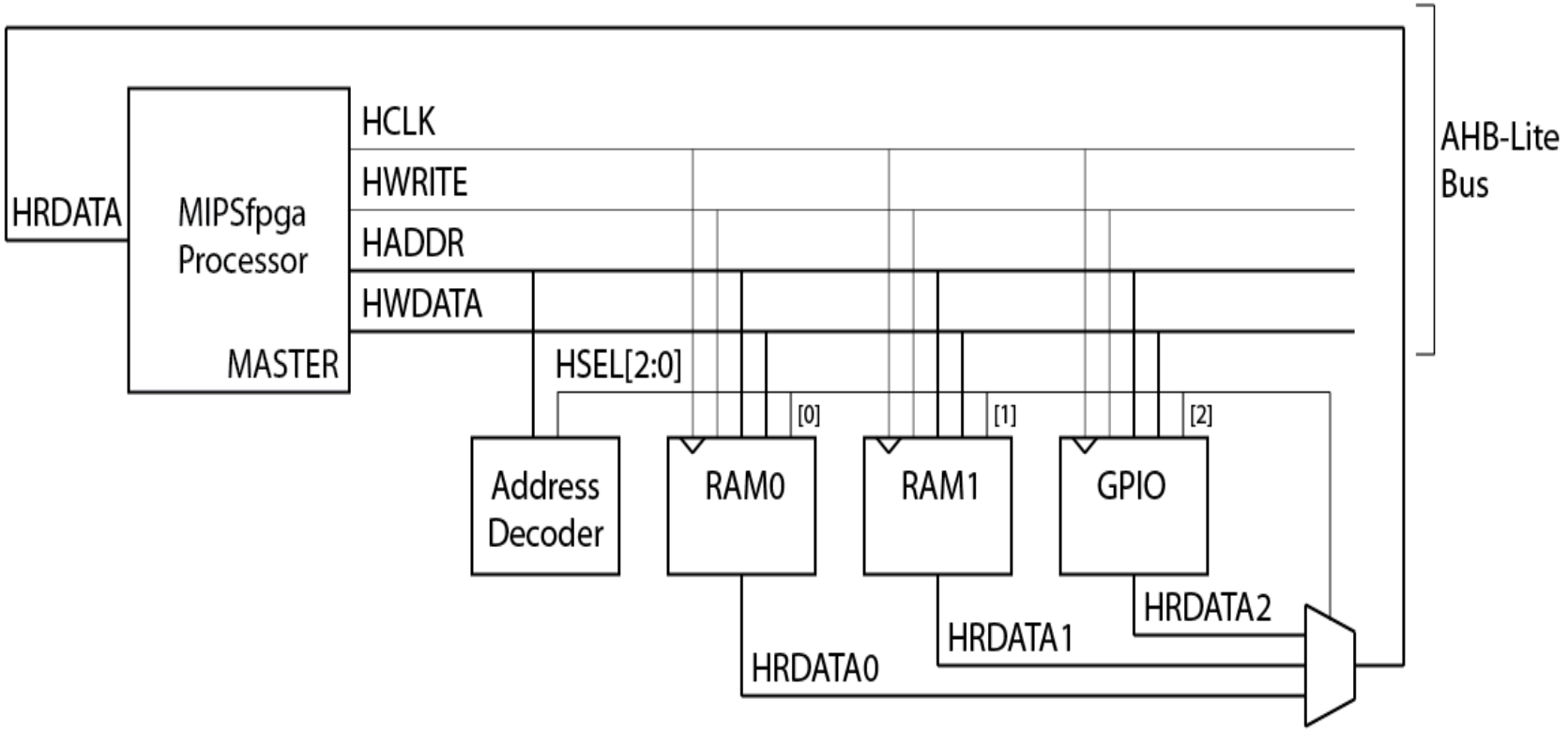
信号名	描述
HADDR[31:0]	地址总线
HRDATA[31:0]	读数据总线
HWDATA[31:0]	写数据总线
HWRITE	写使能
HCLK	时钟

H： Verilog文件中AHB-Lite接口信号的前缀

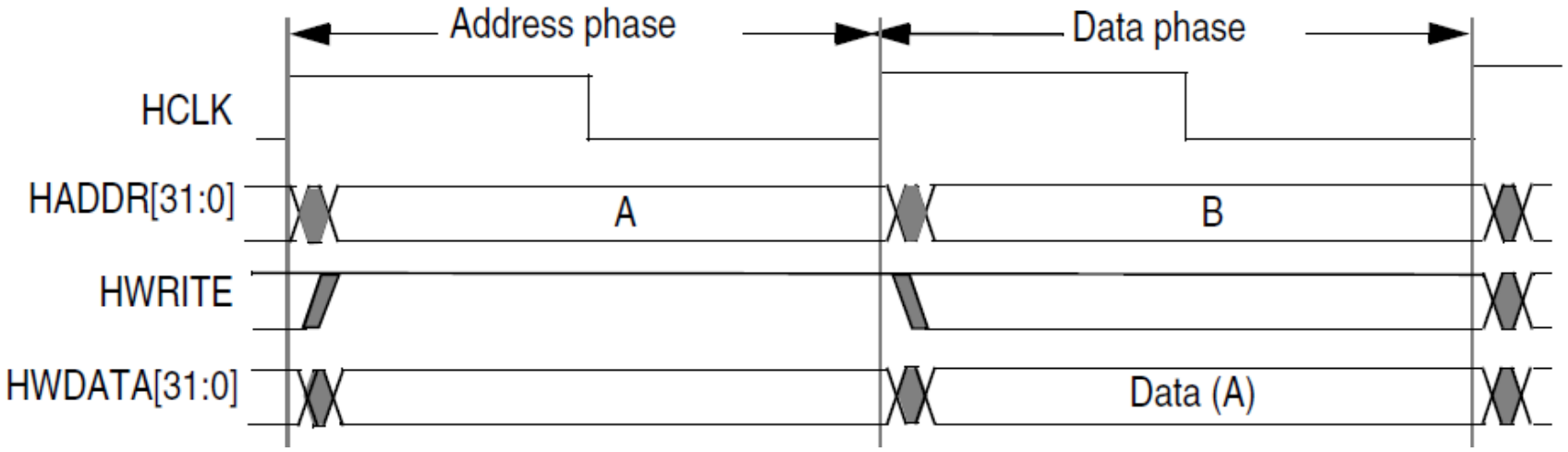
MIPSfpga 接口: AHB-Lite



AHB-Lite 内存/外围设备

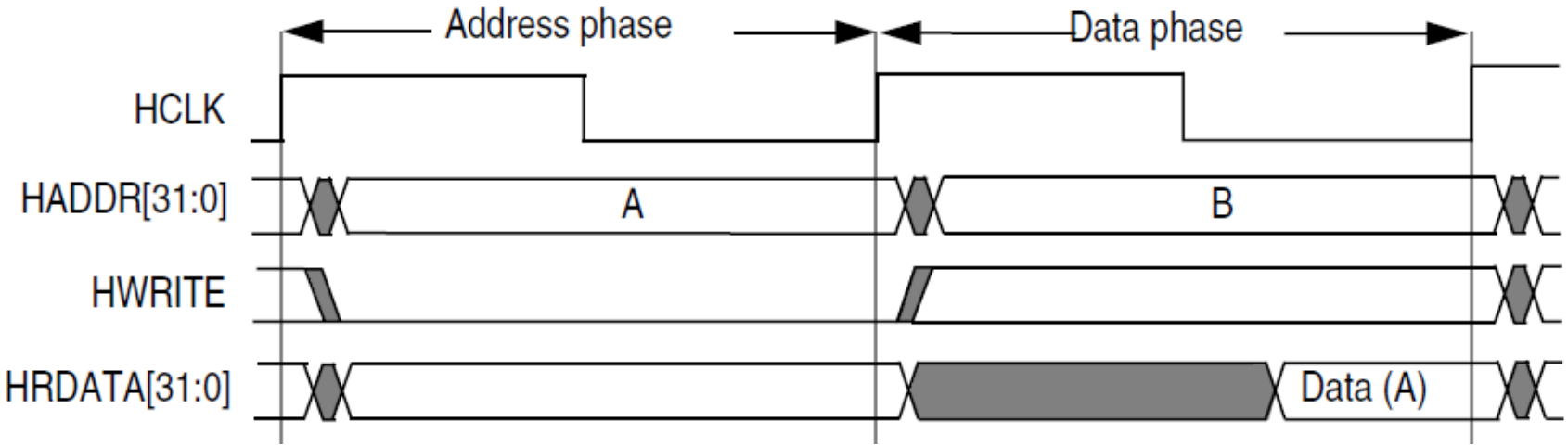


AHB-Lite 写时序



周期1: 地址和写使能(HADDR & HWRITE)
周期2: 写数据(HWDATA)

AHB-Lite 读时序



周期 1: 地址 (HADDR) (即, HWRITE = 0)
周期 2: 读数据(HRDATA)

MIPSfpga 概览

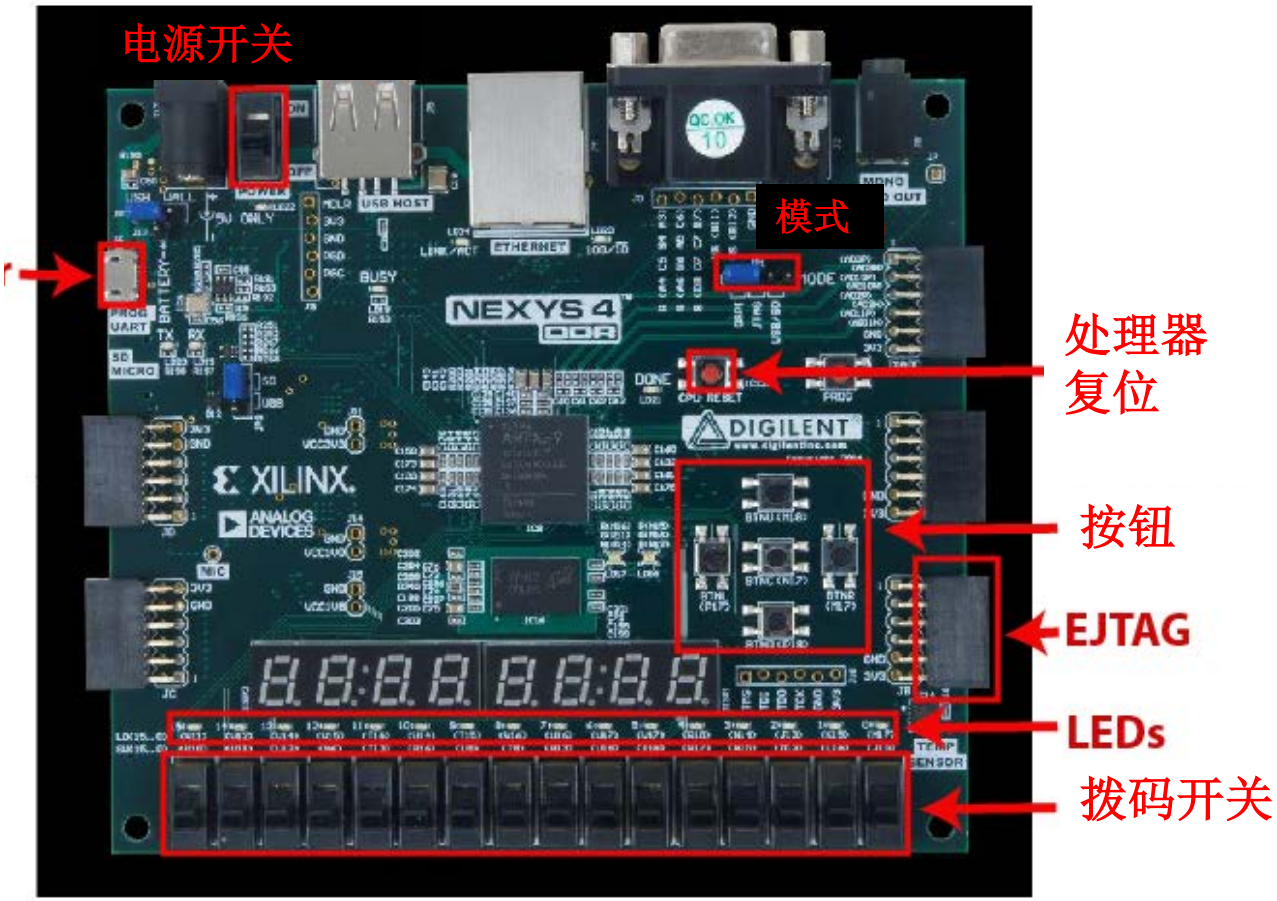
- MIPS架构的历史
- **MIPSfpga**
  - 背景
  - 内核和系统
  - 接口
    - 系统接口
    - **AHB-Lite 总线: FPGA板 I/O**
    - **EJTAG**

MIPSfpga 接口: Nexys4 板

信号名	描述
IO_LEDR[15:0]	LEDs
IO_Switch[15:0]	拨码开关
IO_PB[4:0]	按钮(BTNU, D, L, R, C)

IO： Verilog文件中FPGA板I/O的信号的前缀

Nexys4 DDR FPGA 板

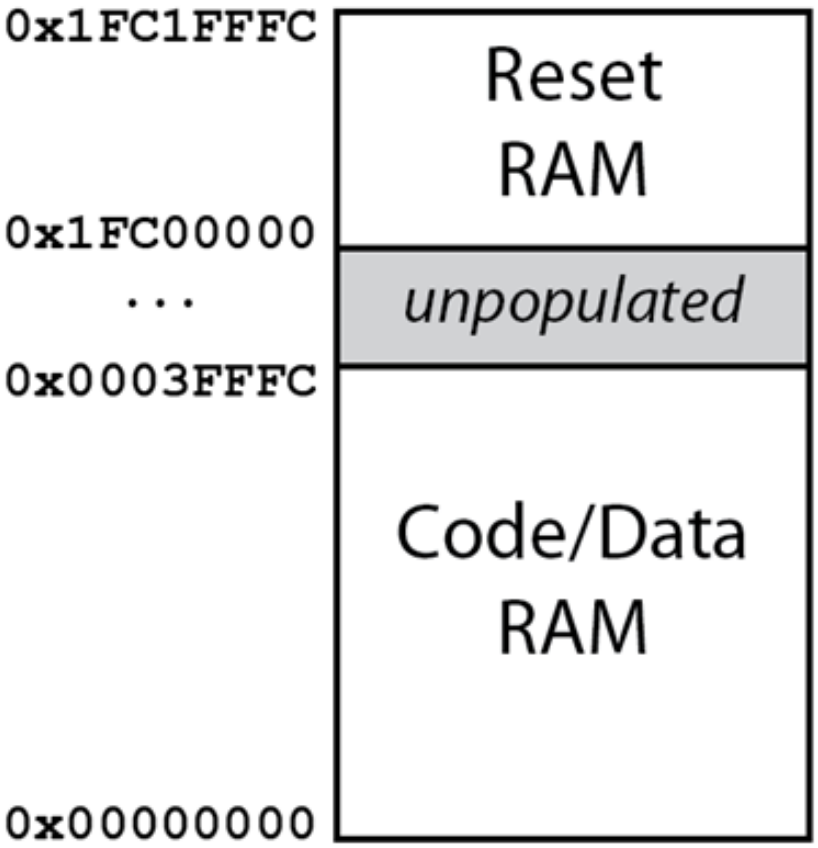


存储映射 I/O

信号名	虚拟地址	物理地址
IO_LEDR[15:0]	0xbf800000	0x1f800000
IO_Switch[15:0]	0xbf800008	0x1f800008
IO_PB[4:0]	0xbf80000c	0x1f80000c

- 对地址0xbf800000的写将**写到LEDs**
- 对地址0xbf800000的读将**读取拨码开关的数值**

MIPSfpga 系统: 物理内存



在存储映射地址(0x1f800000+)处没有物理存储

存储映射I/O

信号名	虚拟地址	物理地址
IO_LEDR[15:0]	0xbf800000	0x1f800000
IO_Switch[15:0]	0xbf800008	0x1f800008
IO_PB[4:0]	0xbf80000c	0x1f80000c

- 对地址0xbf800000的写将**写到LEDs**

存储映射I/O

信号名	虚拟地址	物理地址
IO_LEDR[15:0]	0xbf800000	0x1f800000
IO_Switch[15:0]	0xbf800008	0x1f800008
IO_PB[4:0]	0xbf80000c	0x1f80000c

- 对地址0xbf800000的写将**写到LEDs**

```
//把0x543写到LEDs
addiu $7, $0, 0x543      # $7 = 0x543
lui   $5, 0xbf80         # $5 = 0xbf800000 (LED address)
sw    $7, 0($5) # LEDs = 0x543
```

存储映射I/O

信号名	虚拟地址	物理地址
IO_LEDR[15:0]	0xbf800000	0x1f800000
IO_Switch[15:0]	0xbf800008	0x1f800008
IO_PB[4:0]	0xbf80000c	0x1f80000c

- 对地址**0xbf800000**的写将写到**LEDs**
- 对地址**0xbf800008**的读将读取开关的数值

存储映射I/O

信号名	虚拟地址	物理地址
IO_LEDR[15:0]	0xbf800000	0x1f800000
IO_Switch[15:0]	0xbf800008	0x1f800008
IO_PB[4:0]	0xbf80000c	0x1f80000c

- 对地址**0xbf800000**的写将写到**LEDs**
- 对地址**0xbf800000**的读将读取开关的数值

```
//把开关的值读到10号寄存器
lui   $5,  0xbf80        # $5 = 0xbf800000
lw    $10, 8($5)# $10 = value of switches
```

MIPS程序示例

C代码

```
unsigned int val = 1;
volatile unsigned int* dest;
dest = 0xbf800000;

while (1) {
  *dest = val;
  val++;
}
```

MIPS程序示例

C代码

```
unsigned int val = 1;
volatile unsigned int* dest;
dest = 0xbf800000;

while (1) {
  *dest = val;
  val++;
}
```

汇编代码

```
# $9=val, $8=0xbf800000
    addiu $9, $0, 1   # val=1
    lui   $8, 0xbf80  # address

L1: sw    $9, 0($8)   # write to addr
    addiu $9, $9, 1   # val++
    beqz  $0, L1      # loop
    nop               # branch delay
                      # slot
```

程序示例

C代码

```
unsigned int val = 1;
volatile unsigned int* dest;
dest = 0xbf800000;

while (1) {
  *dest = val;
  val++;
}
```

汇编代码

```
# $9=val, $8=0xbf800000
    addiu $9, $0, 1    # val=1
    lui   $8, 0xbf80   # address

L1: sw    $9, 0($8)    # write to addr
    addiu $9, $9, 1    # val++
    beqz  $0, L1       # loop
    nop                # branch delay
                       # slot
```

把递增的数值写到内存地址**0xbf800000（LEDs）**

怎样在MIPSfpga上运行程序?

- 仿真
- 硬件上：
  - 在综合时将程序加载到内存
  - 使用**EJTAG**接口将程序加载到内存

怎样在MIPSfpga上运行程序?

- 仿真
- 硬件上：
  - 在综合时将程序加载到内存
  - 使用**EJTAG**接口将程序加载到内存

MIPSfpga 概览

- MIPS架构的历史
- **MIPSfpga**
  - 背景
  - 内核和系统
  - 接口
    - 系统接口
    - **AHB-Lite 总线**
    - **EJTAG**

MIPSfpga 接口: EJTAG

- 应用于对MIPSfpga内核的编程和调试
- 借助JTAG协议的信号名和功能

MIPSfpga 接口: EJTAG

信号名	描述
EJ_TDI	测试数据输入
EJ_TDO	测试数据输出
EJ_TMS	测试模式选择
EJ_TCK	测试时钟
EJ_DINT	调试中断请求
SI_ColdReset_N	处理器复位
EJ_TRST_N_probe	EJTAG控制器复位

EJ：Verilog文件中EJTAG接口信号前缀

如何在MIPSfpga上运行程序?

- 仿真上
- 硬件上：
  - 在综合时将程序加载到内存
  - 使用**EJTAG**接口将程序加载到内存

MIPS程序示例

C代码

```
unsigned int val = 1;
volatile unsigned int* dest;
dest = 0xbf800000;

while (1) {
  *dest = val;
  val++;
}
```

汇编代码

```
# $9=val, $8=0xbf800000
    addiu $9, $0, 1    # val=1
    lui   $8, 0xbf80   # address

L1: sw    $9, 0($8)    # write to addr
    addiu $9, $9, 1    # val++
    beqz  $0, L1       # loop
    nop                # branch delay
                       # slot
```

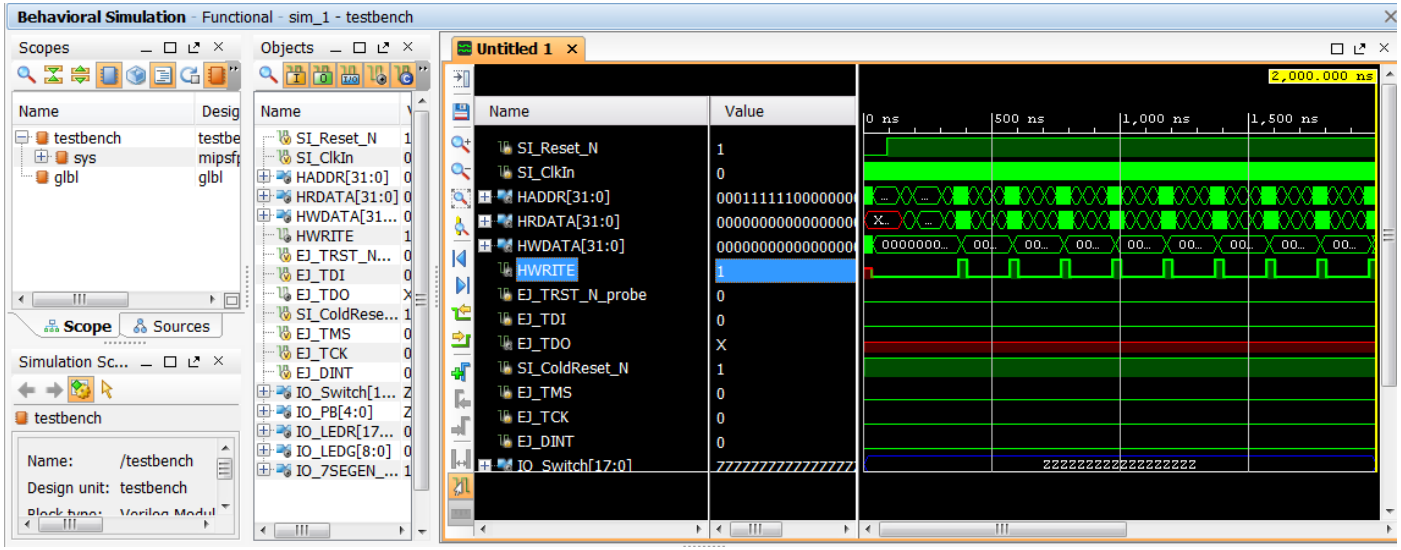
将递增的在数值写到地址**0xbf800000（LEDs）**

机器码

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	// bfc00008:	L1: sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop

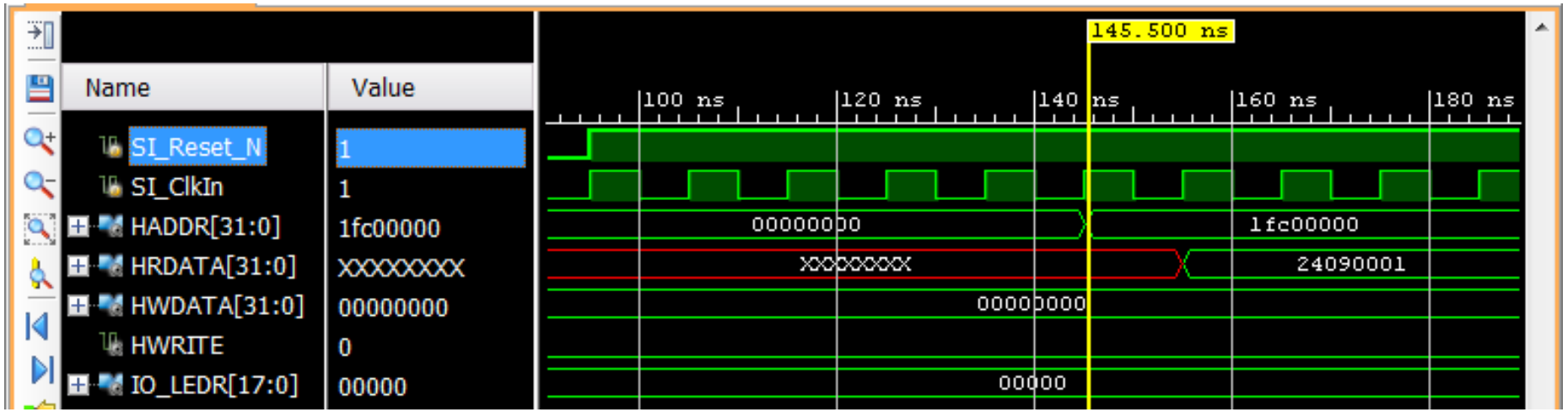
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	// bfc00008:	L1: sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



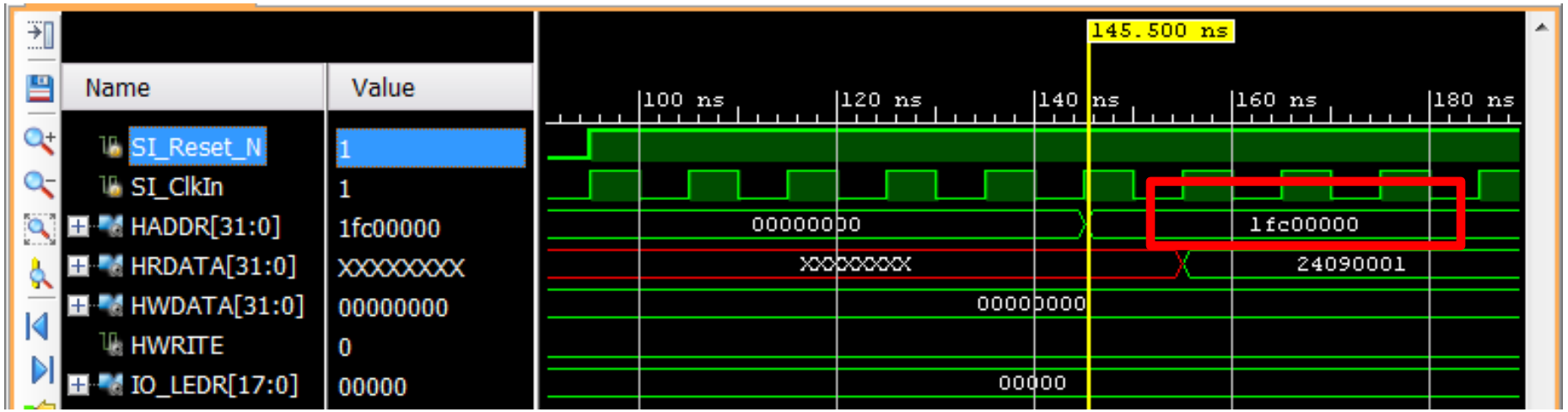
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	**// bfc00000:**	**addiu $9, $0, 1**
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	// bfc00008:	L1: sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



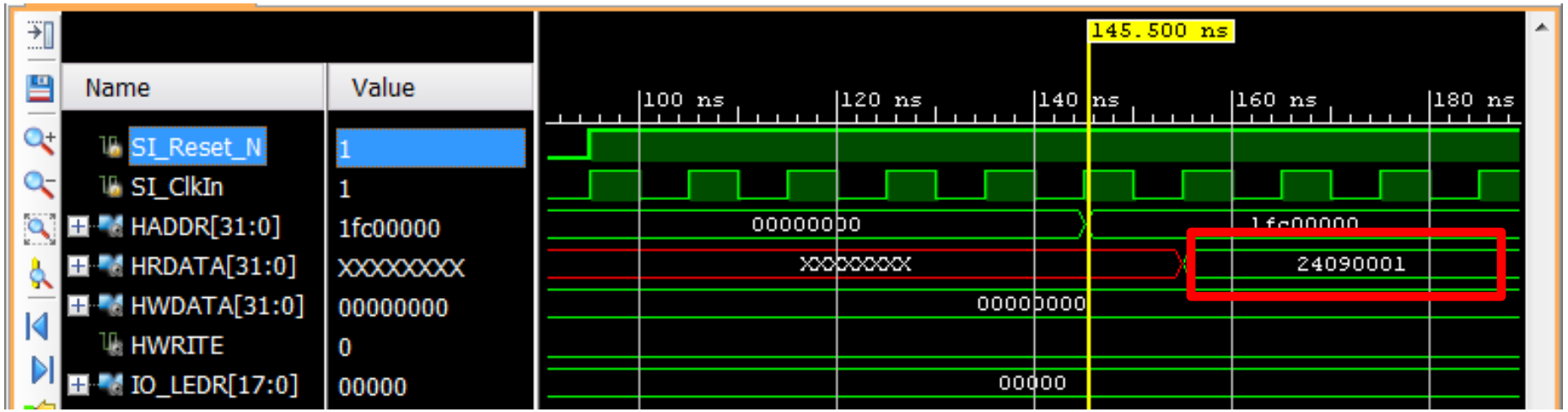
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	**// bfc00000:**	**addiu $9, $0, 1**
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	// bfc00008:	L1: sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



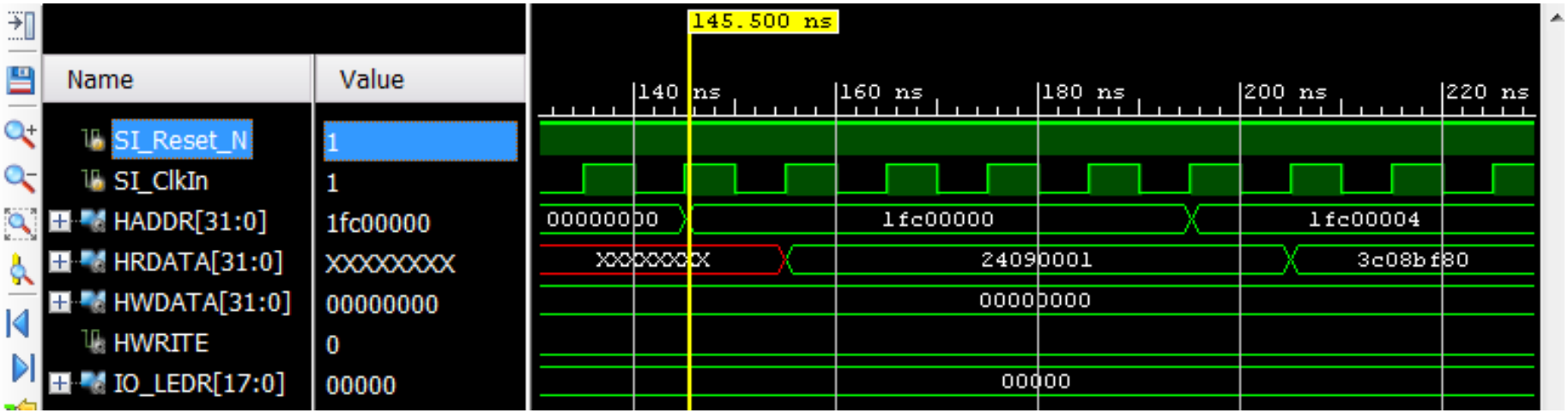
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	**// bfc00000:**	**addiu $9, $0, 1**
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	// bfc00008:	L1: sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



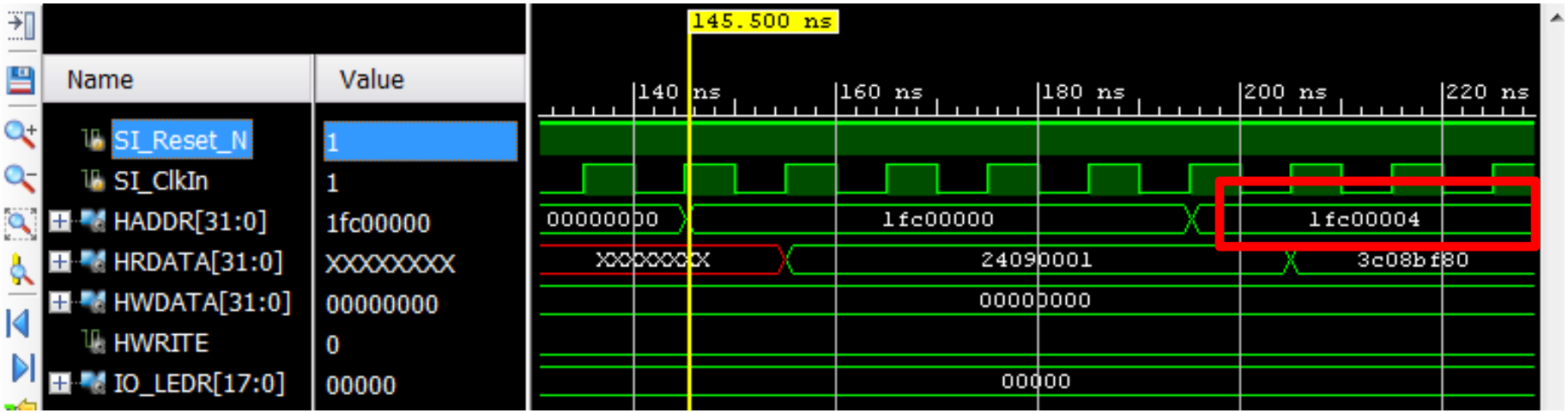
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	**// bfc00004:**	**lui $8, 0xbf80**
ad090000	// bfc00008:	L1: sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



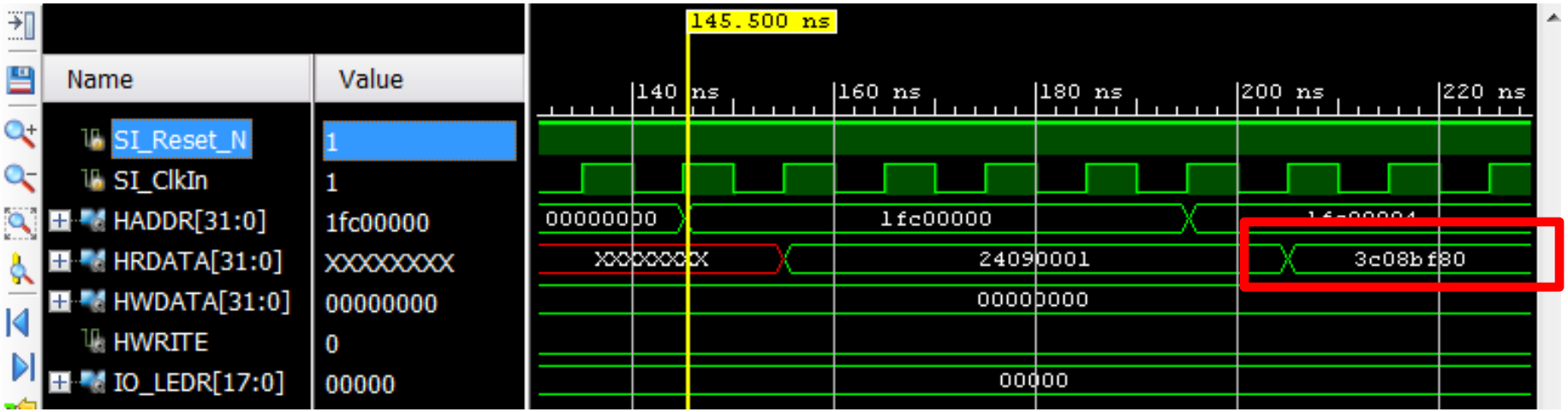
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	**// bfc00004:**	**lui $8, 0xbf80**
ad090000	// bfc00008:	L1: sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



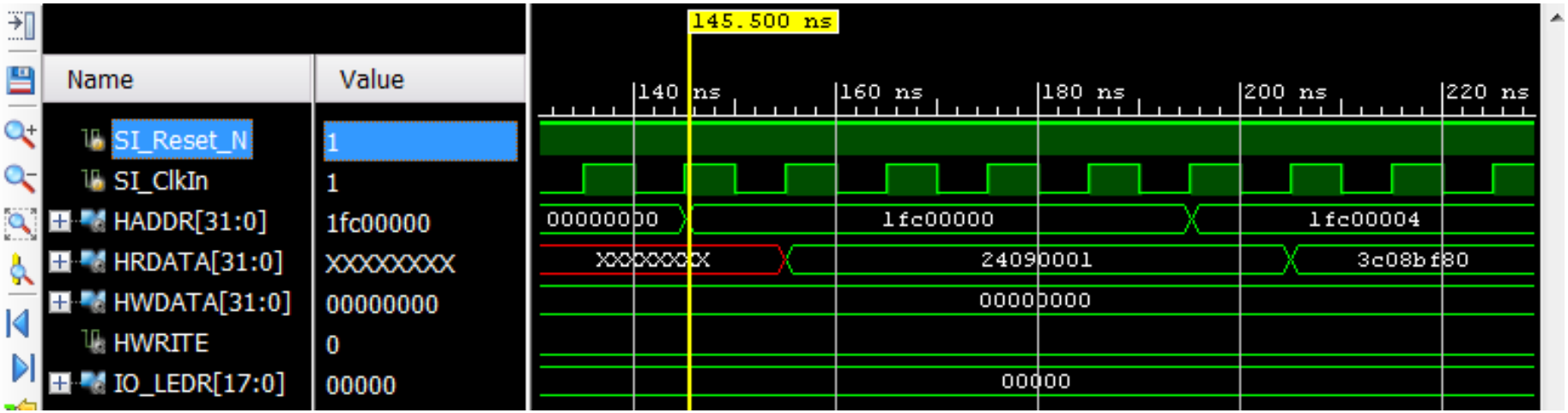
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	**// bfc00004:**	**lui $8, 0xbf80**
ad090000	// bfc00008:	L1: sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



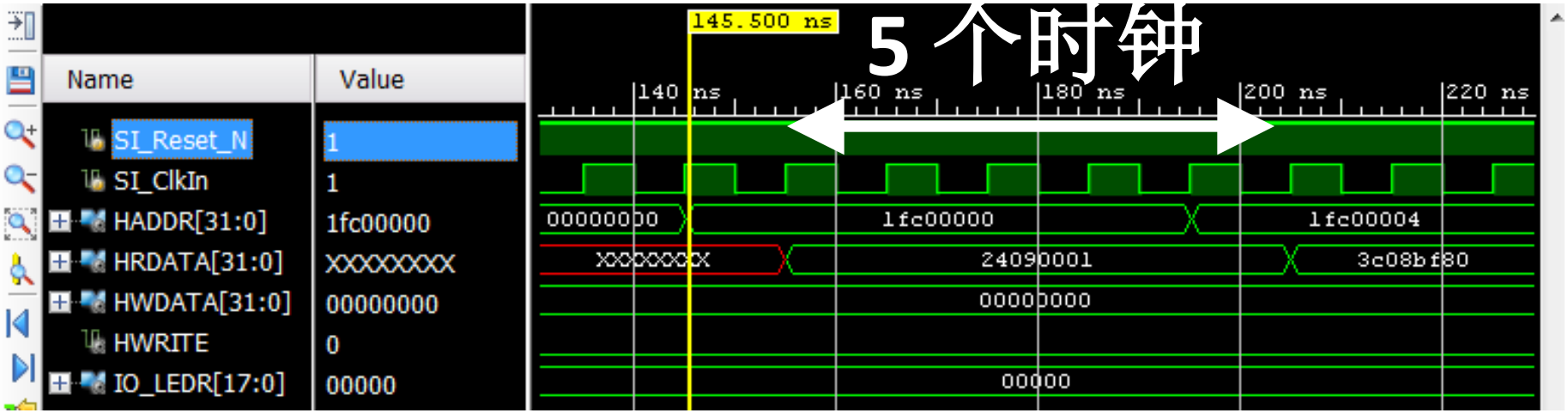
MIPSfpga仿真

在缓存初始化前，每条指令花费5个周期而不是1个周期



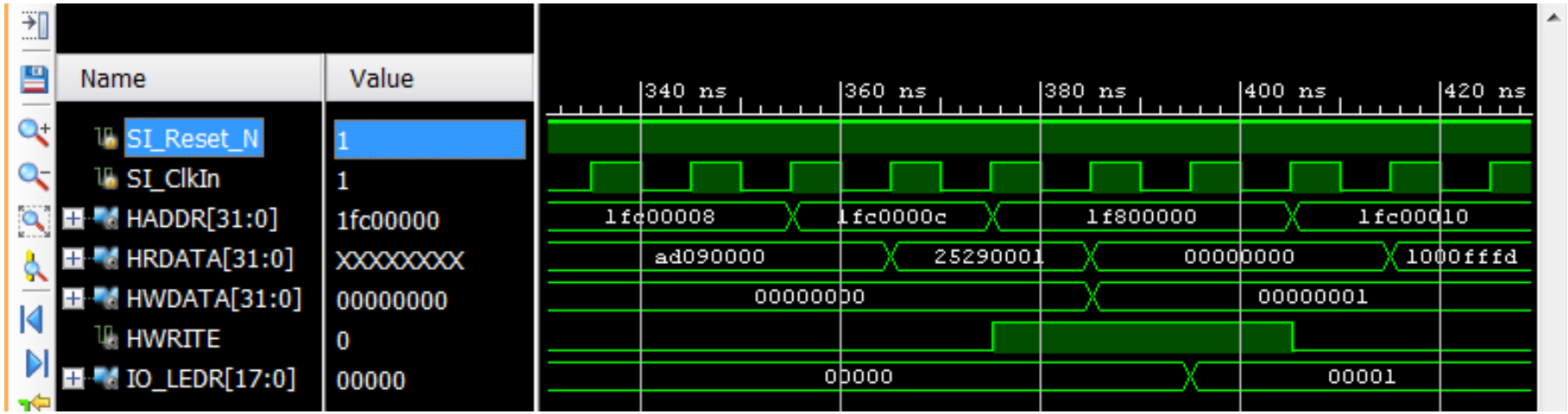
MIPSfpga仿真

在缓存初始化前，每条指令花费5个周期而不是1个周期



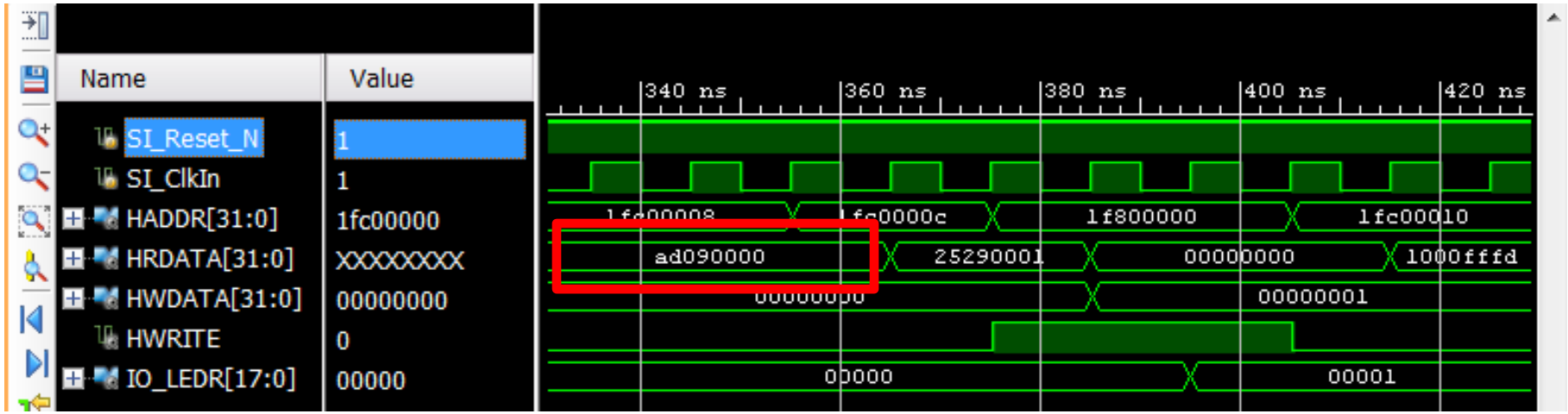
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	**// bfc00008:**	**L1: sw $9, 0($8)**
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



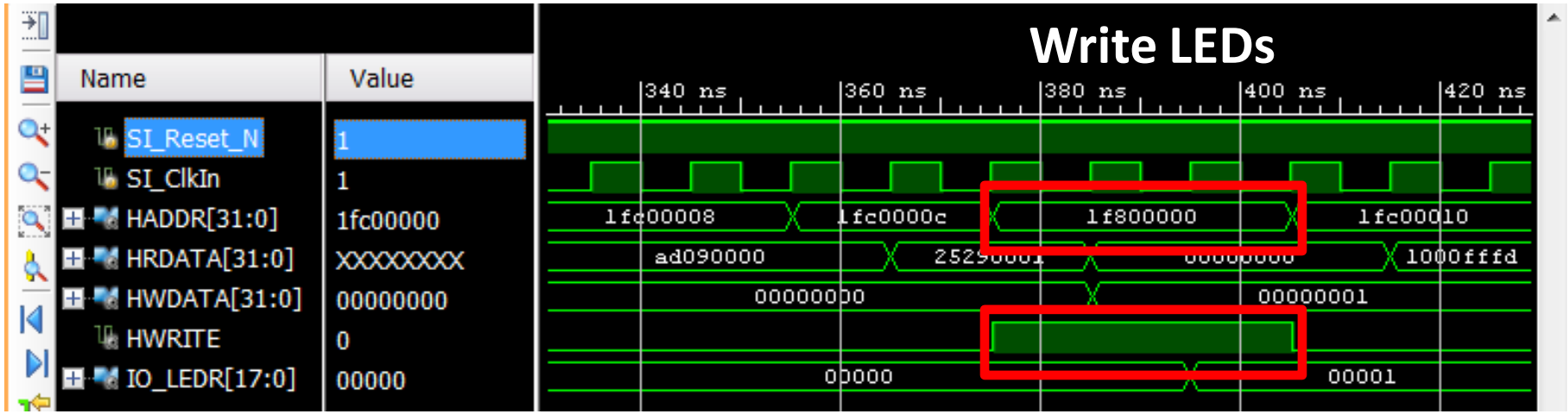
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	**// bfc00008:**	**L1: sw $9, 0($8)**
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



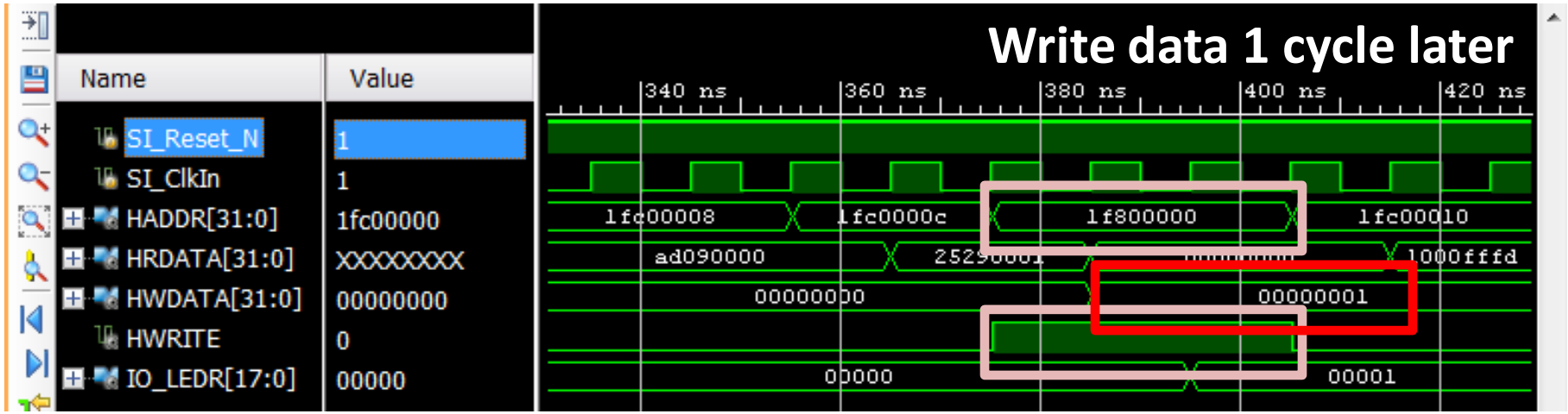
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	**// bfc00008:**	**L1: sw $9, 0($8)**
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



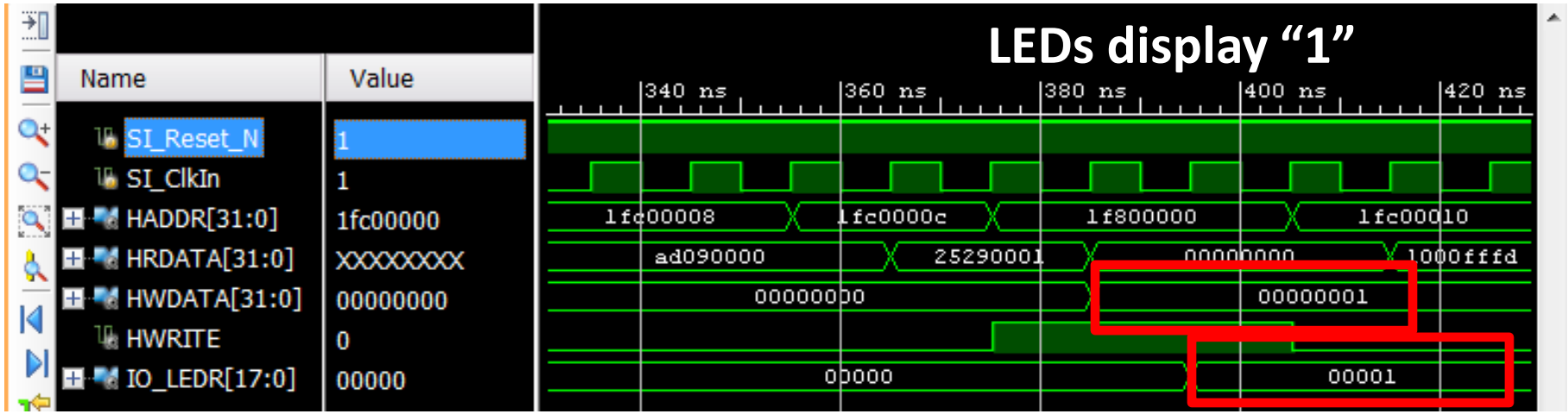
MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	**// bfc00008:**	**L1: sw $9, 0($8)**
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



MIPSfpga仿真

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	**// bfc00008:**	**L1: sw $9, 0($8)**
25290001	// bfc0000c:	addiu $9, $9, 1
1000fffd	// bfc00010:	beqz $0, L1
00000000	// bfc00014:	nop



如何在MIPSfpga上运行程序？

- 仿真
- 硬件：
  - 综合的时候就把程序加载到内存中
  - 通过EJTAG接口把程序下载到内存中

综合时加载程序

内存模块

```
module ram_reset_dual_port
  initial begin
    $readmemh("ram_reset_init.txt",
ram);
  end
```

内存初始化文件

ram_reset_init.txt:

机器码	指令地址	MIPS汇编代码
24090001	// bfc00000:	addiu $9, $0, 1
3c08bf80	// bfc00004:	lui $8, 0xbf80
ad090000	// bfc00008: L1:	sw $9, 0($8)
25290001	// bfc0000c:	addiu $9, $9, 1
3c050026	**// bfc00010: delay:**	**lui $5, 0x026**
34a525a0	**// bfc00014:**	**ori $5, $5, 0x25a0**
00003020	**// bfc00018:**	**add $6, $0, $0**
00a63822	**// bfc0001c: L2:**	**sub $7, $5, $6**
20c60001	**// bfc00020:**	**addi $6, $6, 1**
1ce0fffd	**// bfc00024:**	**bgtz $7, L2**
00000000	**// bfc00028:**	**nop**
1000fff6	**// bfc0002c:**	**beq $0, $0, L1**
00000000	**// bfc00030:**	**nop**

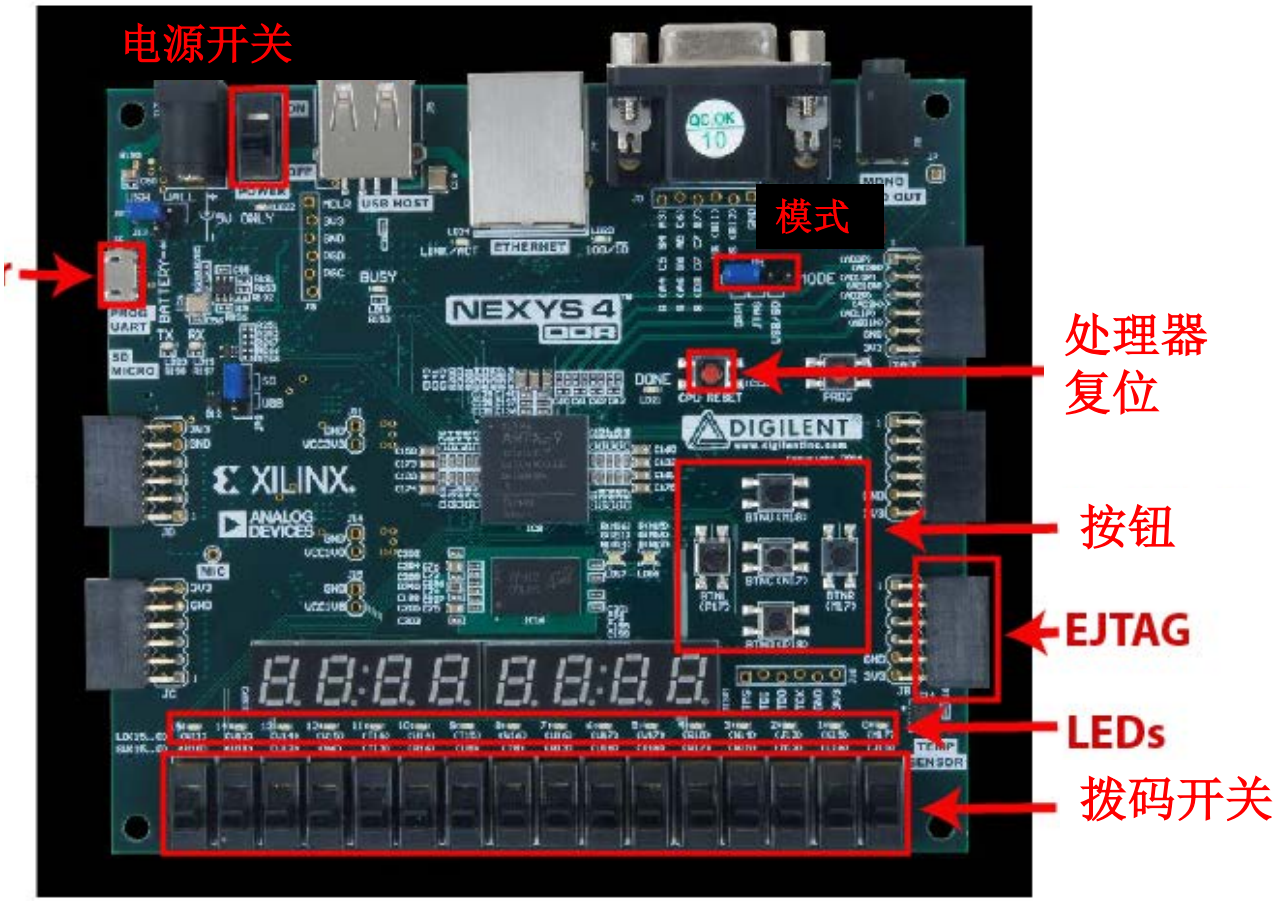
添加延时使得人眼可以看清LED的变化

实验 2和3: 编程

如何在MIPSfpga上运行程序？

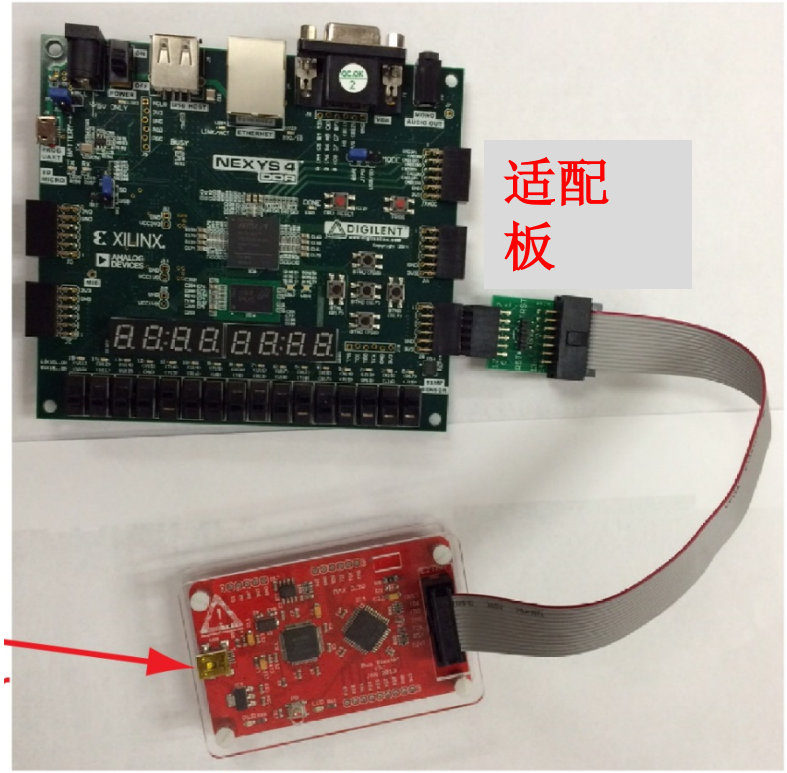
- 仿真
- 硬件：
  - 综合的时候就把程序加载到内存中
  - 通过EJTAG接口把程序下载到内存中

Nexys4 DDR 板上的MIPSfpga



Nexys4 DDR / Bus Blaster 连接器

Nexys4 DDR板



下载代码到MIPSfpga上

1. 编译代码-修正错误
2. 把Bus Blaster 连接到板卡和电脑上
3. 把MIPSfpga系统下载到Nexys4 DDR板卡上
4. 使用脚本下载程序:

   loadMIPSfpga.bat

下载代码到MIPSfpga上

1. 编译代码-修复bug
2. 把Bus Blaster 连接到板卡和电脑上
   - 通过zadig.exe安装电缆驱动
3. 把MIPSfpga系统下载到Nexys4 DDR板卡上
4. 使用脚本下载程序:

   loadMIPSfpga.bat

使用脚本下载程序

1. 打开命令窗口（**cmd.exe**）
2. 把路径设置为脚本所在目录：

   `MIPSfpga_Fundamentals\Scripts\Nexys4_DDR`

3. 紧接着在命令提示符下输入：

```
loadMIPSfpga.bat
C:\MIPSfpga_Fundamentals\Xilinx\Lab02_C\Read
Switches
```

使用脚本下载程序

1. 打开命令窗口（**cmd.exe**）
2. 把路径设置为脚本所在目录：

   `MIPSfpga_Fundamentals\Scripts\Nexys4_DDR`

3. 紧接着在命令提示符下输入：

```
loadMIPSfpga.bat
C:\MIPSfpga_Fundamentals\Xilinx\Lab02_C\ReadSwitches
```

注意：脚本的参数可以为任何路径

实验2和3：编程

使用MIPSfpga_Fundamentals\Xilinx 中的C语言和MIPS汇编来对MIPSfpga编程

- 程序
- 配套文件:

  Lab02_C

  Lab03_Assembly

实验 5: 存储映射 I/O

实验5：添加外围设备

目标: 通过存储映射I/O的方式把7-段数码管显示器添加到MIPSfpga上

实验5:7-段数码管显示器

目标: 通过存储映射I/O的方式把7-段数码管显示器添加到MIPSfpga上

实验过程:

1. 添加硬件驱动7-段数码管显示器
2. 把数码管显示的数字和数码管使能信号映射到内存空间上
3. 修改MIPSfpga接口来驱动数码管的7个段和使能引脚

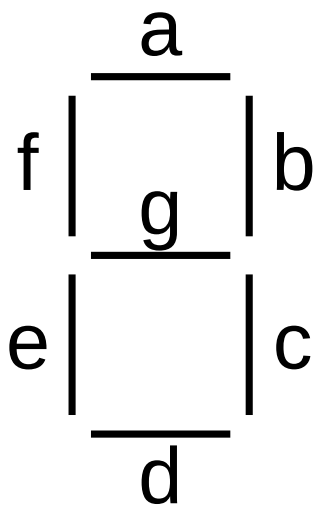
实验5:7-段数码管显示器

目标: 通过存储映射I/O的方式把7-段数码管显示器添加到MIPSfpga上

实验过程:

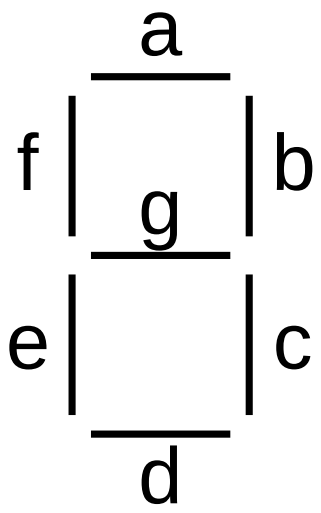
1. **添加7-段数码管显示器硬件驱动**
2. 把数码管显示的数字和数码管使能信号映射到内存空间上
3. 修改MIPSfpga接口来驱动数码管的7个段和使能引脚

7-段数码管显示器



通过点亮某几个段来显示数字

7-段数码管显示器



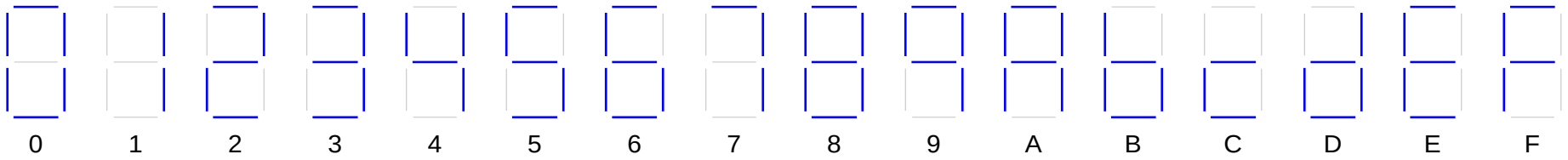
举个例子，

0 点亮: **a,b,c,d,e,f**

1 点亮: **b,c**

2 点亮: **a,b,d,e,g**

等等.



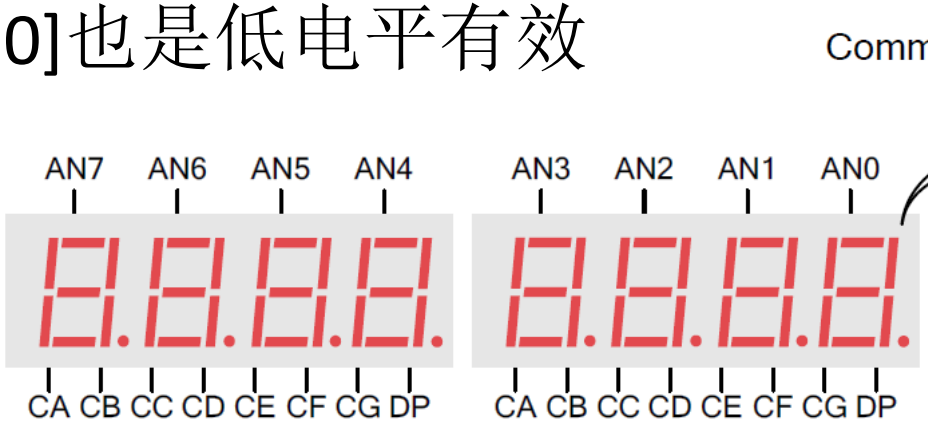
7-段数码管显示器：低电平触发

```
module mipsfpga_ahb_sevensegdec(input      [3:0] data,
                                output reg [6:0] segments);
  always @(*)
    case(data)              // abc_defg
      4'h0: segments = 7'b000_0001;
      4'h1: segments = 7'b100_1111;
      4'h2: segments = 7'b001_0010;
      4'h3: segments = 7'b000_0110;
      4'h4: segments = 7'b100_1100;
      4'h5: segments = 7'b010_0100;
      4'h6: segments = 7'b010_0000;
      4'h7: segments = 7'b000_1111;
      4'h8: segments = 7'b000_0000;
      4'h9: segments = 7'b000_1100;
      4'ha: segments = 7'b000_1000;
      4'hb: segments = 7'b110_0000;
      4'hc: segments = 7'b111_0010;
      4'hd: segments = 7'b100_0010;
      4'he: segments = 7'b011_0000;
      4'hf: segments = 7'b011_1000;
      default:
            segments = 7'b111_1111;
    endcase
endmodule
```

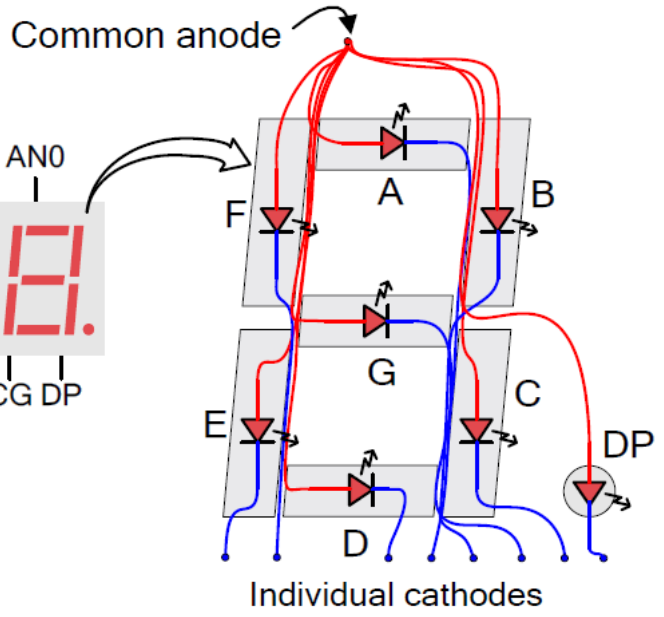
这些段都是低电平有效

Nexys4 DDR 7-段数码管显示器

- 8个7-段数码管显示器
- 每个数码管的段信号都接到同一个输入上（CA-CG）
- 使能信号（AN[7:0]）决定哪个数码管可以点亮，AN[7:0]也是低电平有效



Eight-digit Seven Segment Display

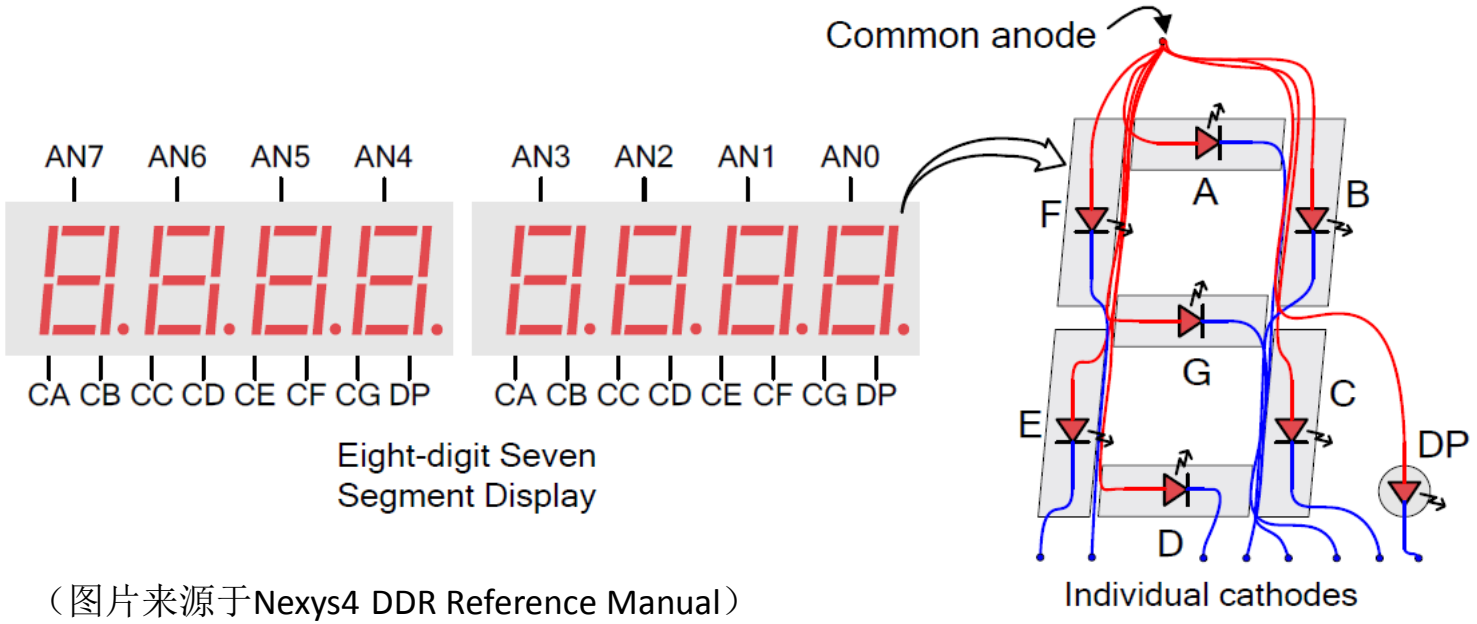


（图片来源于Nexys4 DDR Reference Manual）

Nexys4 DDR 7-段数码管显示器

例子:

- AN[7:0] = 11111110_2 (只有最右边的数码管被打开)
- 显示的数字由CA-CG决定

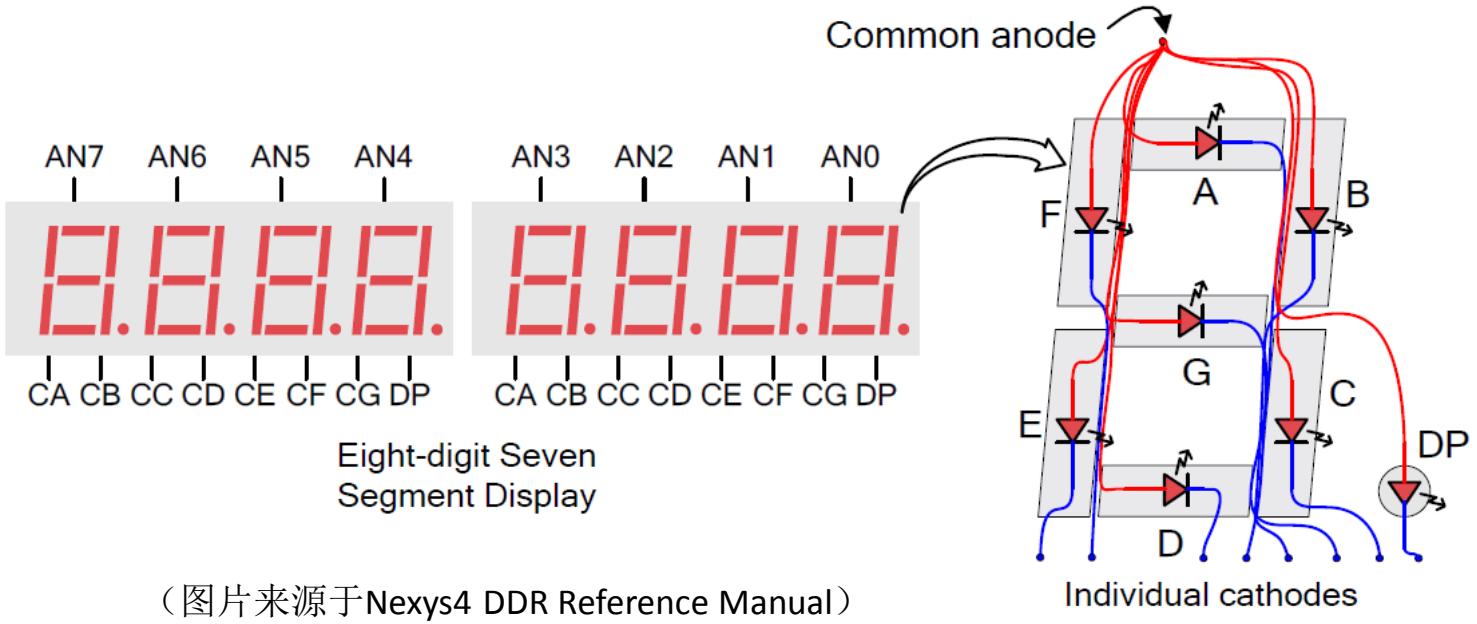


（图片来源于Nexys4 DDR Reference Manual）

Nexys4 DDR 7-段数码管显示器

驱动多个数码管:

- 一次只能驱动一个数码管
- 速度足够快的话就能可以使人眼看不到闪烁

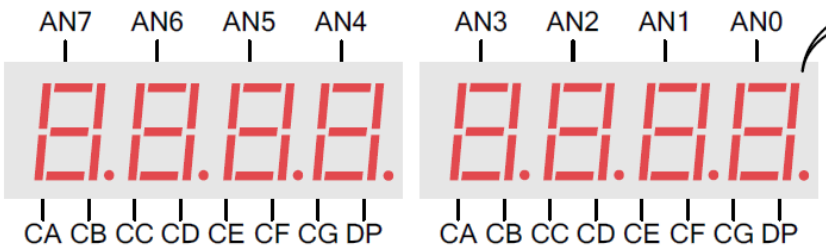


（图片来源于Nexys4 DDR Reference Manual）

Nexys4 DDR 7-段数码管显示器

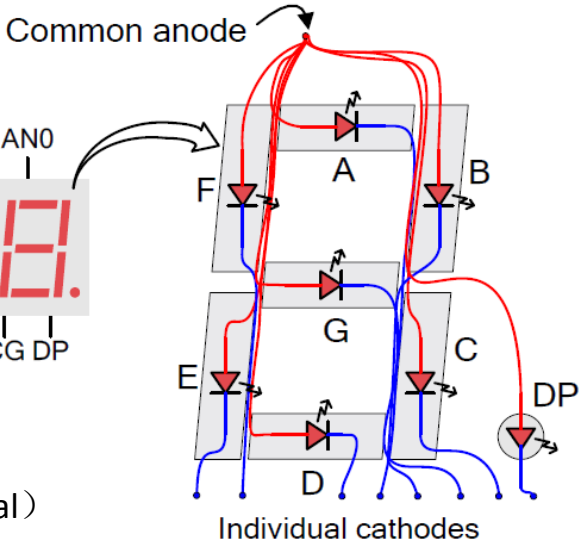
不同数码管显示间隔大约为2ms:

- 当 t=0,　　AN[7:0] = 11111110, CA-CG 显示数字0
- 当 t=2ms, AN[7:0] = 11111101, CA-CG 显示数字1
- 当 t=4ms, AN[7:0] = 11111011, CA-CG 显示数字2
- 等等...



Eight-digit Seven Segment Display

（图片来源于Nexys4 DDR Reference Manual）



Nexys4 DDR 7-段数码管显示器硬件

9个寄存器需要存储映射:

- 一个用来设置哪些数码管可以亮（ SEGEN_N[7:0]) ）
- 其他8个用来设置显示的数字（ SEG0_N[3:0], SEG1_N[3:0], … SEG7_N[3:0]）

Nexys4 DDR 7-段数码管显示器硬件

9个寄存器需要存储映射:

- 一个用设置哪些数码管可以亮（ SEGEN_N[7:0]) ）
- 其他8个用来设置显示的数字（ SEG0_N[3:0], SEG1_N[3:0], … SEG7_N[3:0]）

通过一个3位的计数器（运行在500Hz，周期为2ms）逐个点亮被使能的数码管显示器。

实验5:7-段数码管显示器

目标: 通过存储映射I/O的方式把7-段数码管显示器添加到MIPSfpga上

实验过程:

1. 添加7-段数码管显示器硬件驱动
2. **把数码管显示的数字和数码管使能信号映射到内存空间上**
3. 修改MIPSfpga接口来驱动数码管的7个段和使能引脚

使能信号和数字的存储映射

寄存器名	描述	存储地址
SEGEN_N[7:0]	使能	0xbf800010
SEG0_N[3:0]	Digit 0 的值	0xbf800014
SEG1_N[3:0]	Digit 1的值	0xbf800018
SEG2_N[3:0]	Digit 2的值	0xbf80001c
SEG3_N[3:0]	Digit 3的值	0xbf800020
SEG4_N[3:0]	Digit 4的值	0xbf800024
SEG5_N[3:0]	Digit 5的值	0xbf800028
SEG6_N[3:0]	Digit 6的值	0xbf80002c
SEG7_N[3:0]	Digit 7的值	0xbf800030

实验5:7-段数码管显示器

目标: 通过存储映射I/O的方式把7-段数码管显示器添加到MIPSfpga上

实验过程:

1. 添加7-段数码管显示器硬件驱动
2. 把数码管显示的数字和数码管使能信号映射到内存空间上
3. **修改MIPSfpga接口来驱动数码管的7个段和使能引脚**

MIPSfpga 接口

GPIO模块和MIPSfpga系统输出信号:

```
...
output      [  7: 0] IO_7SEGEN_N,
output      [  6: 0] IO_7SEG_N
```

MIPSfpga 接口

GPIO模块和MIPSfpga系统输出信号:

```
    ...
    output      [  7: 0] IO_7SEGEN_N,
    output      [  6: 0] IO_7SEG_N
```

Nexys4 DDR 顶层模块:

```
module mipsfpga_nexys4_ddr( ...
      output [ 7:0] AN,
      output        CA, CB, CC, CD, CE, CF, CG);
  ...
  mipsfpga_sys mipsfpga_sys(
                    ...
                    .IO_7SEGEN_N(AN),
                    .IO_7SEG_N({CA,CB,CC,CD,CE,CF,CG})
```

MIPSfpga 接口: Nexys4 DDR 引脚

MIPSfpga_Nexys4DDR.xdc:

```
set_property -dict { PACKAGE_PIN T10   IOSTANDARD
LVCMOS33 } [get_ports { CA }];
set_property -dict { PACKAGE_PIN R10   IOSTANDARD
LVCMOS33 } [get_ports { CB }];
...
set_property -dict { PACKAGE_PIN J17   IOSTANDARD
LVCMOS33 } [get_ports { AN[0] }];
set_property -dict { PACKAGE_PIN J18   IOSTANDARD
LVCMOS33 } [get_ports { AN[1] }];
...
```

Lab 9: 移植 MIPSfpga

实验 9: 移植MIPSfpga到其他板卡

目标: 把MIPSfpga移植到其他FPGA板卡上

为什么要移植到其他板卡上?

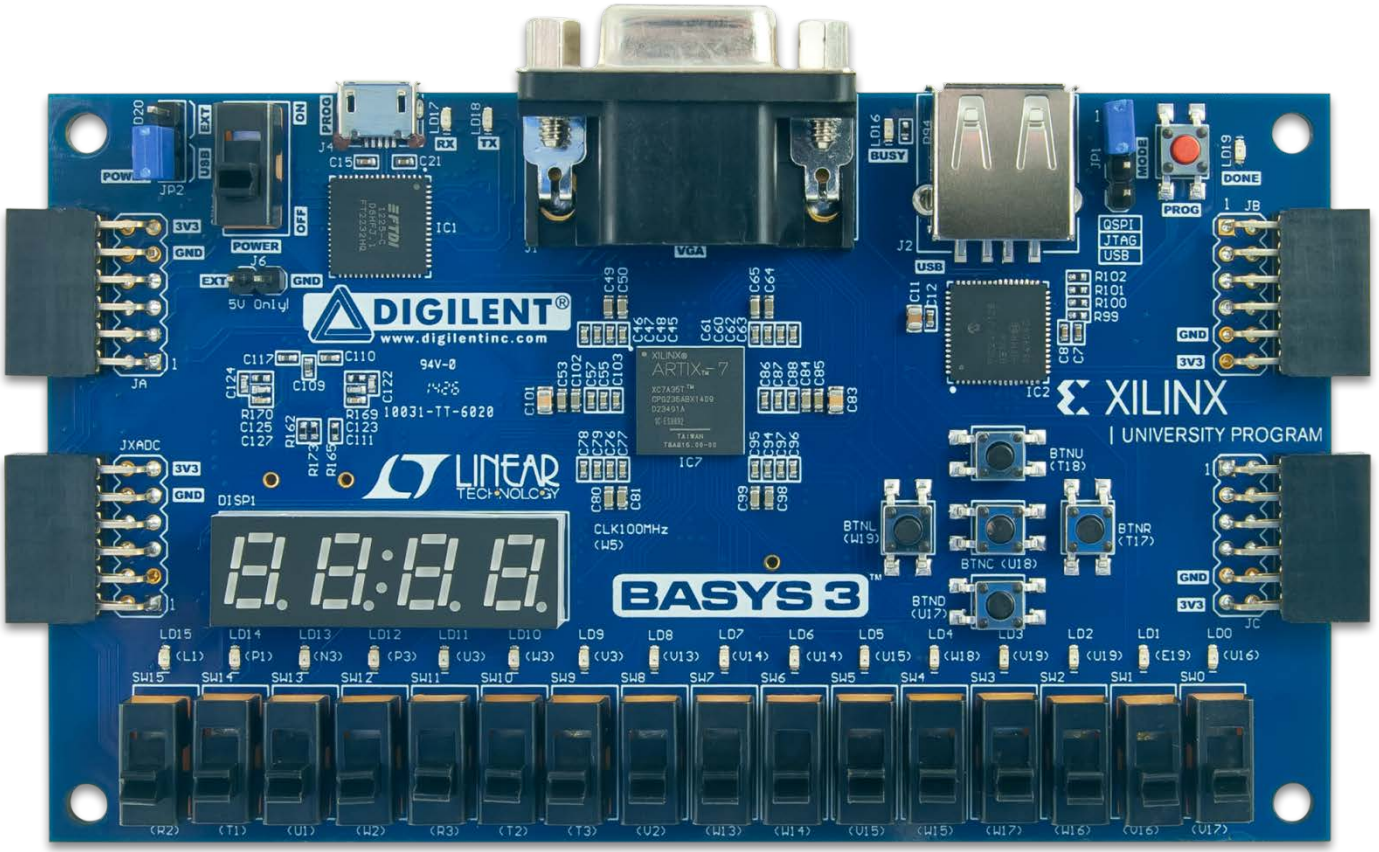
- 可用性
- 更经济

实验 9: 移植MIPSfpga到其他板卡

需要改动的地方:

- 顶层封装模块
- Xilinx 设计约束文件(.xdc) : 映射顶层模块的I/O信号到FPGA引脚
- 可能要改变内存大小

Example: Basys3



实验 9: 移植MIPSfpga到其他板卡

需要改动的地方:

- 顶层封装模块
- Xilinx Design Constraint (.xdc) 约束文件: 映射顶层模块的I/O信号到FPGA引脚
- 可能要改变内存大小

Basys3 与 Nexys4 DDR

	Basys3	Nexys4 DDR
价格	$79 (大学计划), $149	$159 (大学计划), $320
FPGA	Artix-7 (XC7A35T-CPG236C)	Artix-7 (XC7A100T-1CSG324C)
内存 **(block RAM)**	225 KB	607 KB
LE's (logic)	33k	101k
7-段数码管显示器	4	8

Basys3: 顶层封装模块

```
module mipsfpga_basys3(
  input          clk,
  input          btnU, btnD, btnL, btnR, btnC,
  input  [15:0] sw,
  output [15:0] led,
  inout  [ 5:0] JB,
  output [ 3:0] an,
  output [ 0:6] seg
);
```

Nexys4 DDR: 顶层封装模块

```
module mipsfpga_nexys4_ddr(
  input          CLK100MHZ,
  input          CPU_RESETN,
  input          BTNU, BTND, BTNL, BTNR, BTNC,
  input  [15:0] SW,
  output [15:0] LED,
  inout  [ 8:1] JB,
  output [ 7:0] AN,
  output         CA, CB, CC, CD, CE, CF, CG
);
```

顶层封装模块

```
module mipsfpga_nexys4_ddr(
  input          CLK100MHZ,
  input          CPU_RESETN,
  input          BTNU, BTND, BTNL, BTNR, BTNC,
  input  [15:0] SW,
  output [15:0] LED,
  inout  [ 8:1] JB,
  output [ 7:0] AN,
  output         CA, CB, CC, CD, CE, CF, CG );

module mipsfpga_basys3(
  input          clk,
  input          btnU, btnD, btnL, btnR, btnC,
  input  [15:0] sw,
  output [15:0] led,
  inout  [ 5:0] JB,
  output [ 3:0] an,
  output [ 0:6] seg );
```

实验 9: 移植MIPSfpga到其他板卡

需要改动的地方:

- 顶层封装模块
- Xilinx设计约束文件(.xdc) : 映射顶层模块的I/O信号到FPGA引脚
- 可能要改变内存大小

约束文件

mipsfpga_basys3.xdc:

```
set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
...
```

实验 9: 移植MIPSfpga到其他板卡

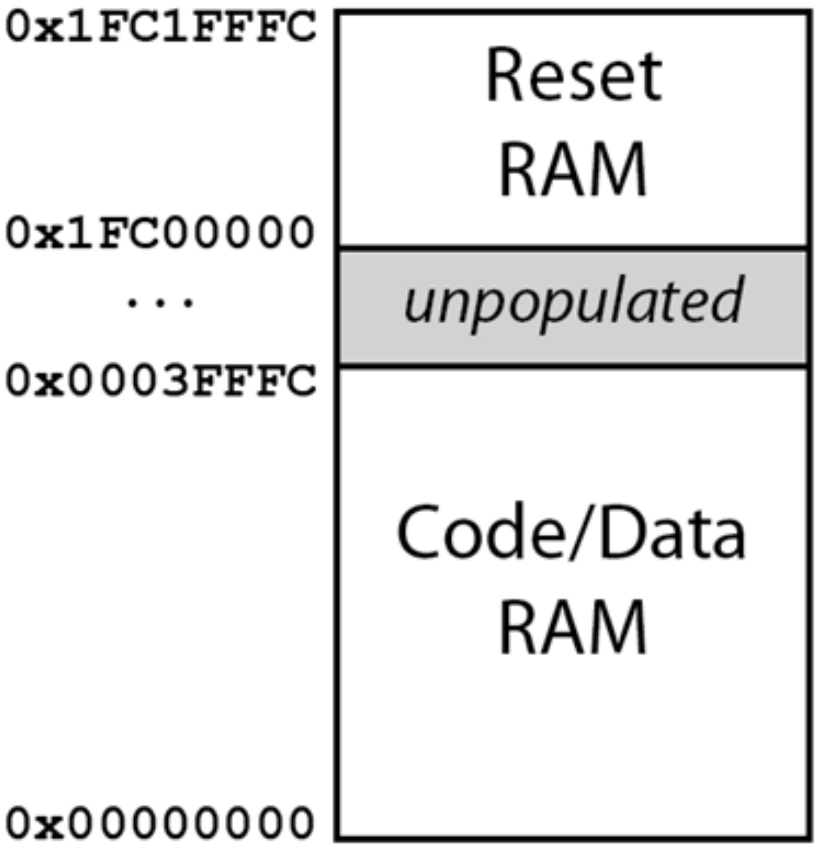
需要改动的地方:

- 顶层模块
- Xilinx 设计约束文件(.xdc) : 映射顶层模块的I/O信号到FPGA引脚
- 可能要改变内存大小

Basys3 与 Nexys4 DDR

	Basys3	Nexys4 DDR
价格	$79 (academic), $149	$159 (academic), $320
FPGA	Artix-7 (XC7A35T-CPG236C)	Artix-7 (XC7A100T-1CSG324C)
内存(block RAM)	**225 KB**	**607 KB**
LE's (logic)	33 k	101 k
7-段数码管显示	4	8

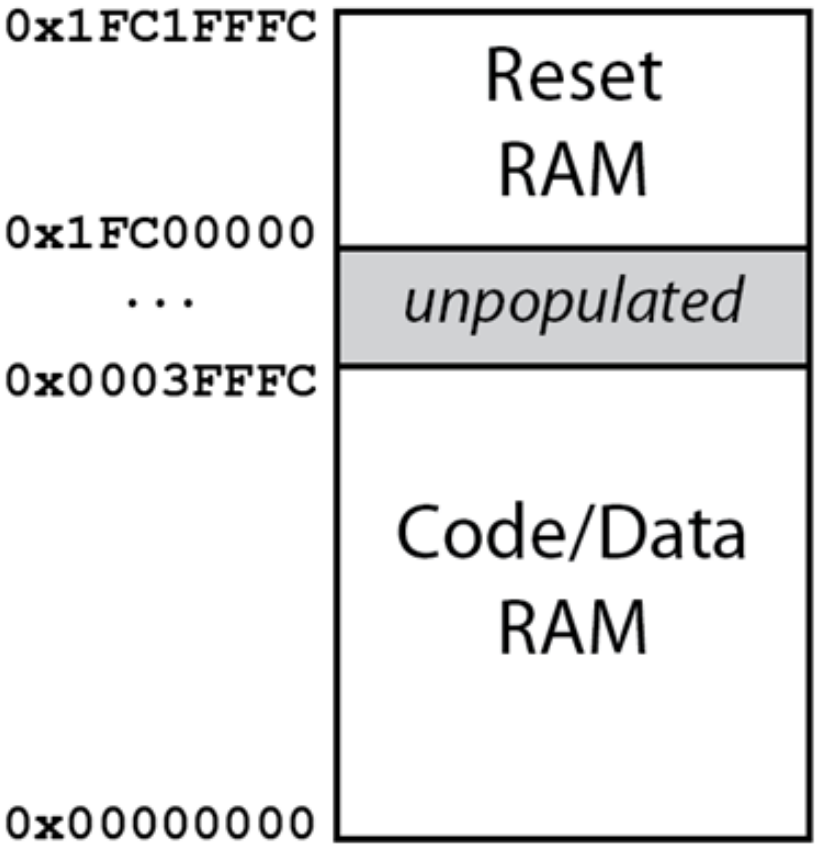
MIPSfpga 系统: 物理内存



引导代码:
启动时运行的代码(128 KB)

用户代码和数据(256 KB)

MIPSfpga 系统: 物理内存



引导代码:
启动时运行的代码(128 KB)

用户代码和数据(256 KB)

Basys3不适用：只有225KB

降低内存大小

引导代码: **32 KB** (2^{15} bytes = 2^{13} words)
用户代码: **64 KB** (2^{16} bytes = 2^{14} words)

降低内存大小

引导代码: **32 KB (**2^{15} bytes = 2^{13} words)
用户代码: **64 KB (**2^{16} bytes = 2^{14} words)

mipsfpga_ahb_const.vh:

```
`define H_RAM_RESET_ADDR_WIDTH(13)
`define H_RAM_ADDR_WIDTH       (14)
```

MIPSfpga 应用

- **教学优质资源:**
  - 数字设计
  - 体系结构
  - 嵌入式系统
  - VLSI
  - 片上系统设计
- **对研究也有很好的资源**

MIPSfpga 教学材料

Imagination 大学计划网站上可以找到相关的教学材料:

http://community.imgtec.com/university/university-registration

支持

MIPSfpga 论坛(技术支持):

http://community.imgtec.com/forums/cat/mips-insider/mipsfpga/

其他论坛

- **MIPS 技术支持(常见问题):**

  http://community.imgtec.com/forums/cat/mips-insider/

- **Imagination 大学计划(课程讨论, IUP 问题, 等等. – 非技术支持):**

  http://community.imgtec.com/forums/cat/university/

致谢

- Robert Owen
- Sarah Harris
- David Money Harris
- Yuri Panchul
- Bruce Ableidinger
- Kent Brinkley
- Chuck Swartley
- Christian White
- Sean Raby
- Rick Leatherman
- Matthew Fortune
- Munir Hasan
- Sachin Sundar
- Michael Alexander
- Sam Bobrowicz
- Cathal McCabe
- Larissa Swanland
- Clint Cole
- Students and faculty at UCL
- Ian Oliver
- Steve Kromer
- Daniel Chaver-Martinez
- Parimal Patel
- Jason Wong