第八章 DMA技术

华中科技大学
电子信息与通信学院
School of Electronic Information and Communications

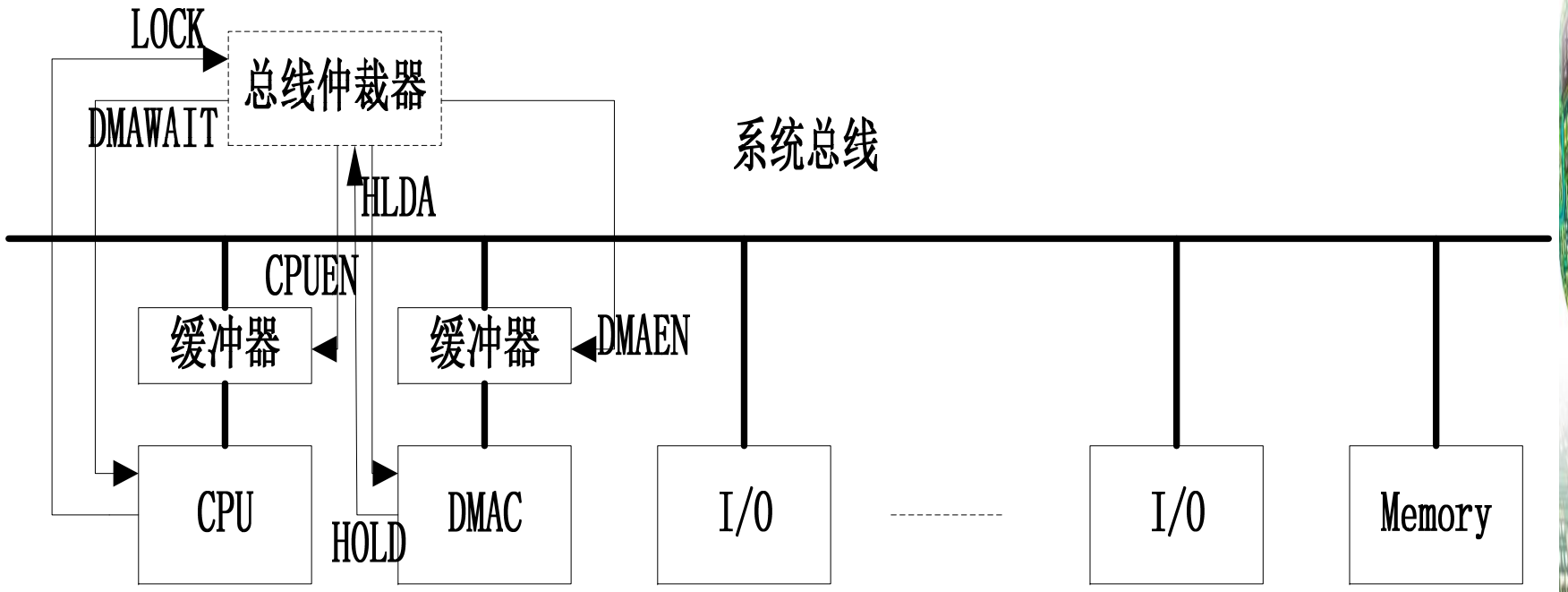
学习目标

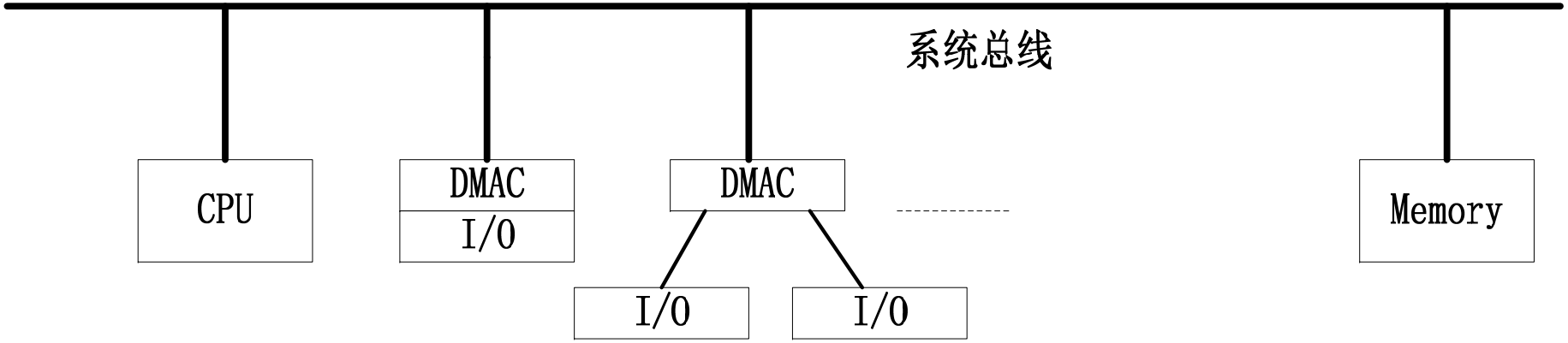
- 了解DMA传送的基本原理
- 了解DMA传送流程
- 了解通道的基本原理
- 了解DMAC 8237A的工作原理(适用于通用PC机)
- 掌握Xilinx XPS DMA控制器的使用（适用于嵌入式）
- 掌握DMA传输初始化编程

8.1 DMA传送基本原理

1) DMA 传输计算机系统构成
2) DMA 传输步骤
3) DMA 传输方向
4) DMA 传输模式

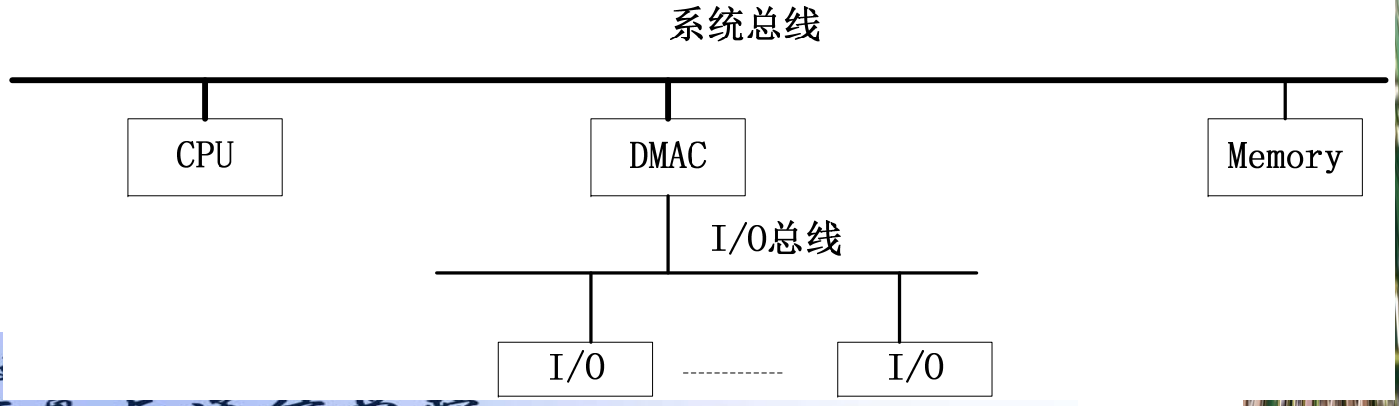
DMA 传输计算机系统构成





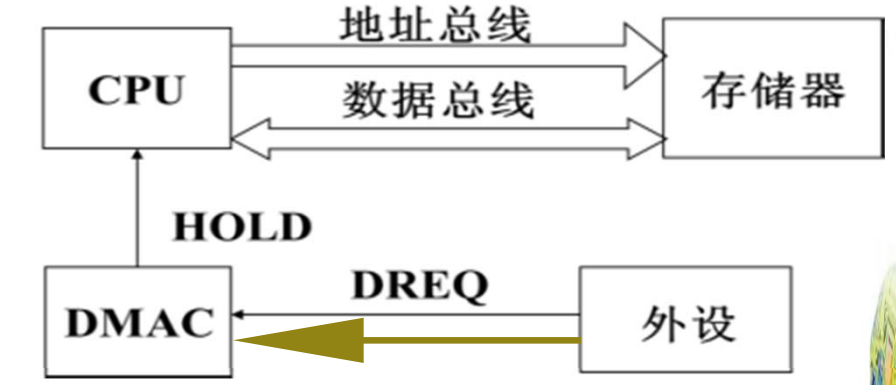
DMA控制器与IO接口集成

DMA控制器提供专门I/O总线

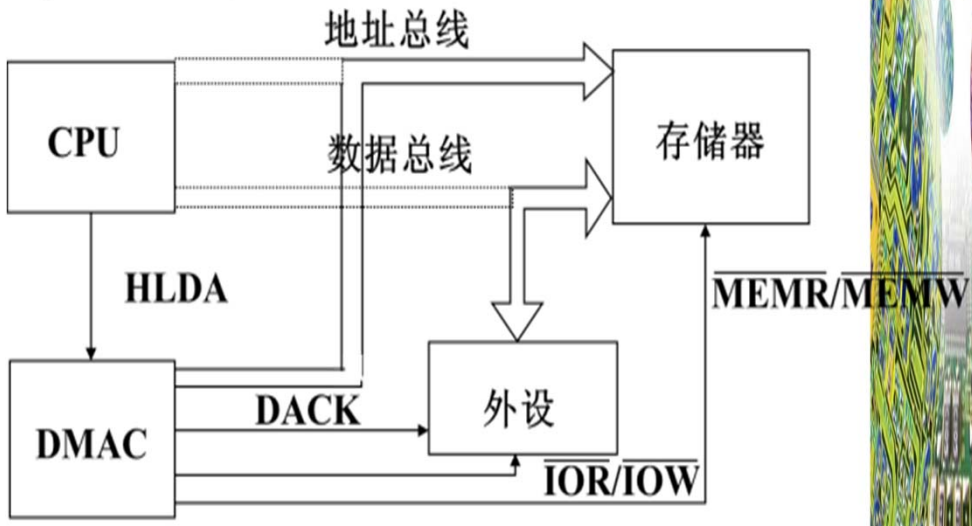


DMA传输步骤

- DMA请求阶段



- **DMA响应及传输阶段**



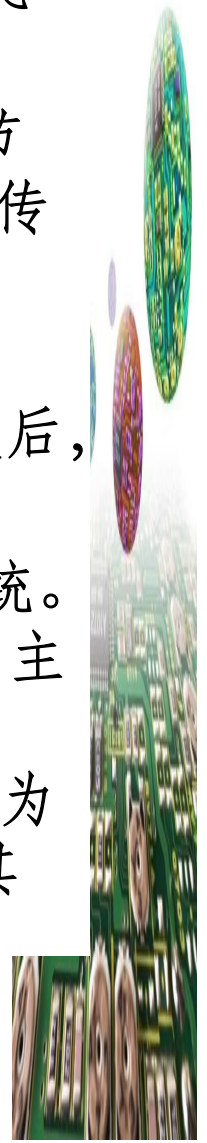
- **DMA过程结束时，DMAC向CPU发出结束信号INT（撤消HOLD请求），将总线控制权交还CPU。**

DMA传输方向

- I/O接口到存储器、
- 存储器到I/O接口、
- 存储器到存储器

DMA传输模式

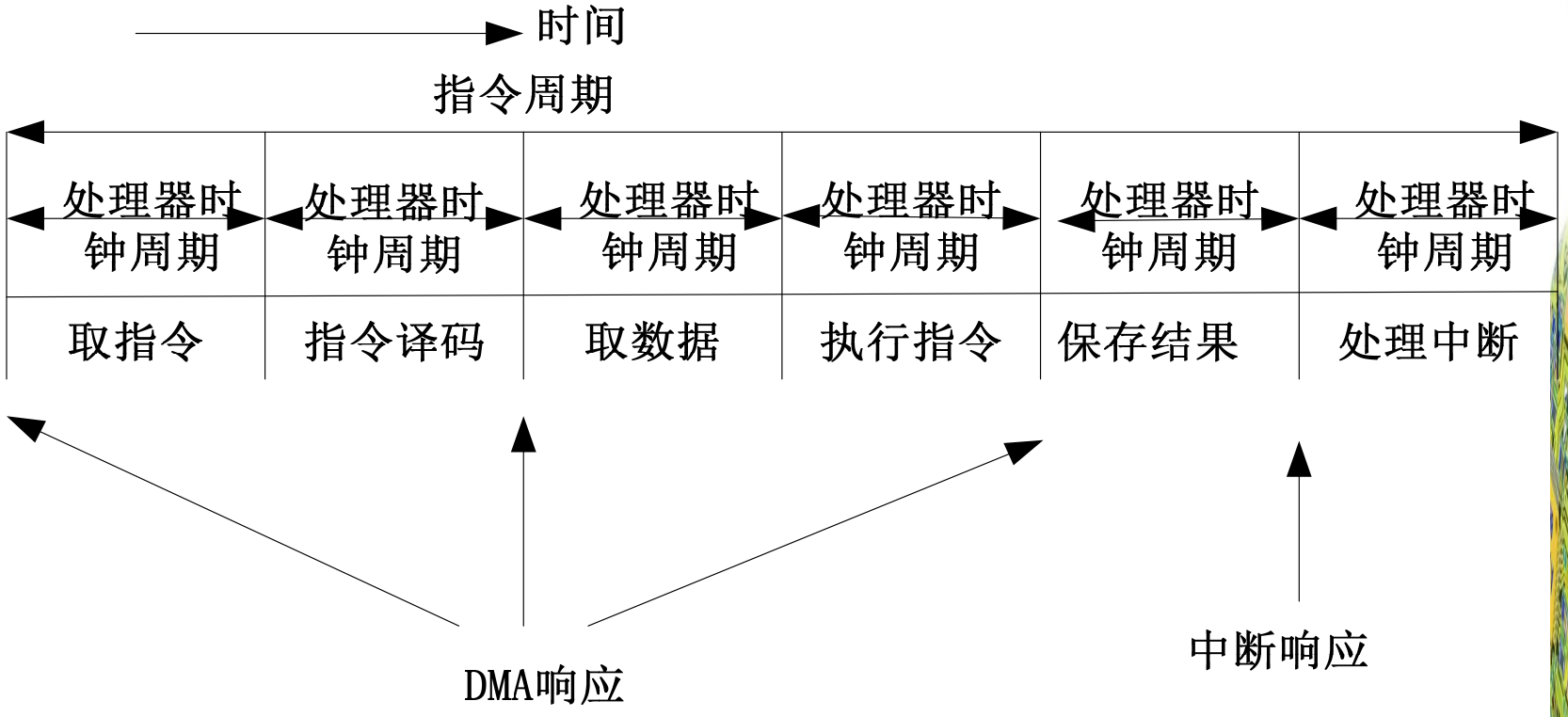
- 单字节传输模式：每次DMA操作传送一个字节后，接着释放总线。
- 块传输模式：连续传送多个字节，每传输一个字节，当前字节计数器减1，当前地址寄存器加1或减1，直到所要求的字节数传输完（当前字节计数器减至0），然后释放总线。
- 请求传输模式：DMAC要检测DREQ信号（询问外设），当DREQ为低时，暂停传输（不释放总线），当DREQ再次有效后，继续进行传输。
- 级联传输模式：多片DMAC级联时，可以构成主从式DMA系统。级联的方式是把从片的请求线HOLD连至主片的DREQ引脚，主片的DACK联至从片的HLDA引脚。若主DMAC的某通道（DREQ）连接从DMAC的HOLD，主DMAC的该通道应设置为级联传输模式，但从DMAC不设置级联传输模式，而是设置其它三种模式之一。



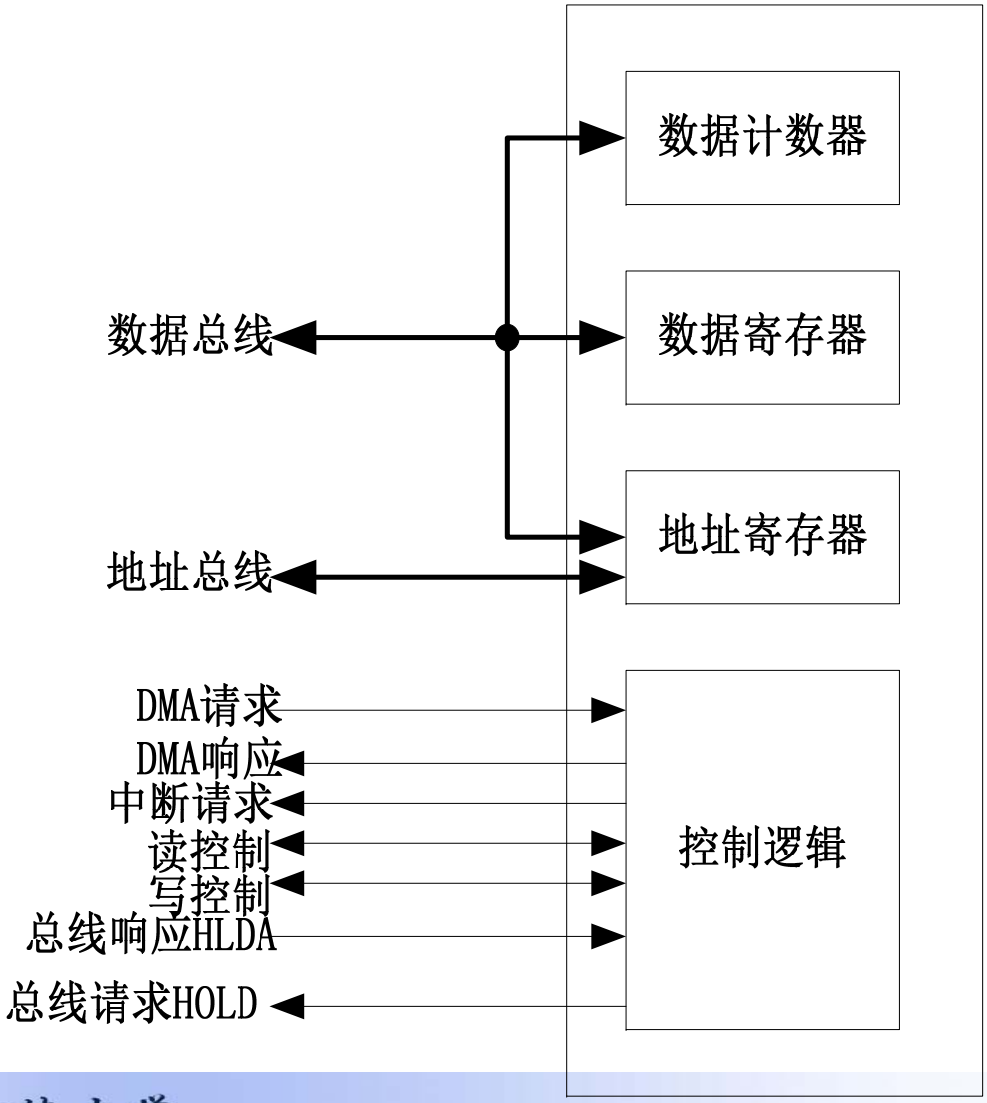
DMA传送基本流程

- 初始化阶段：在此阶段实现对DMA控制器的初始化，如配置数据传送模式，数据传送方向，内存区域的首地址、地址是递增还是递减、传送的总字节数等。
- DMA传送请求阶段：请求DMA传送由外设产生DREQ信号或者由CPU向DMAC写入启动DMA传送控制字，DMAC向CPU发出总线请求信号(HOLD)。
- DMA传送响应阶段：若CPU接收到总线请求信号(HOLD)，并且可以释放系统总线，则发出HLDA信号。
- DMA传送阶段：向地址总线发出地址信号，指出传送过程需使用的内存地址，同时向外设发出DMA应答信号(DACK)，实现该外设与内存之间的DMA传送。在DMA传送期间，DMAC发出内存和外设的读/写信号。
- DMA传送结束阶段：当DMA传送的总字节达到初始化时的字节数或者达到某种终止DMA传输条件时，DMAC向CPU发出结束信号INT（撤消HOLD请求），将总线控制权交还CPU。

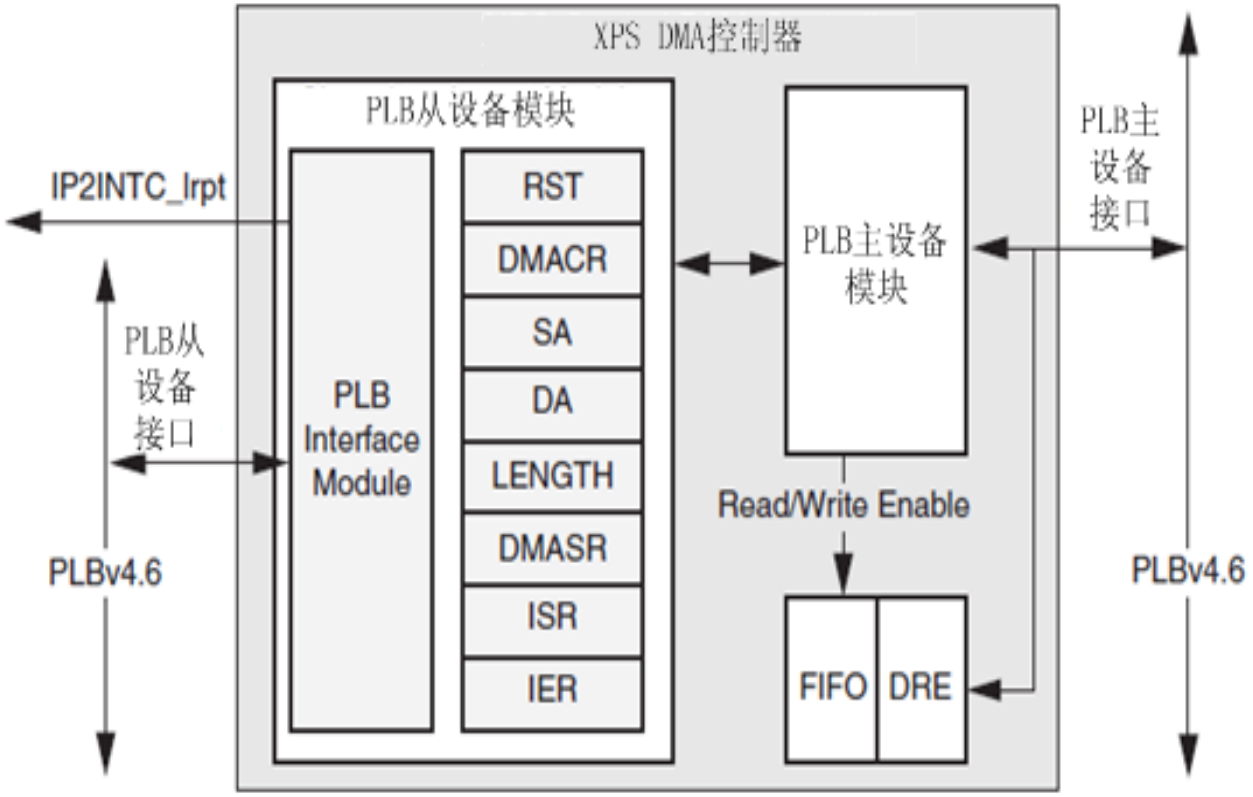
DMA以及中断在指令周期中的响应位置



8.3 DMA控制器


数据计数器
数据寄存器
地址寄存器
控制逻辑
数据总线
地址总线
DMA请求
DMA响应
中断请求
读控制
写控制
总线响应HLDA
总线请求HOLD

Xilinx XPS DMA控制器简介



DMA传输时在嵌入式系统内都可以认为是存储器到存储器的传输方式，
存储器与**IO**接口之间**DMA**传输不同于存储器之间的**DMA**传输主要表现在**IO**接口不需要修改端口地址，而存储器需要修改地址。

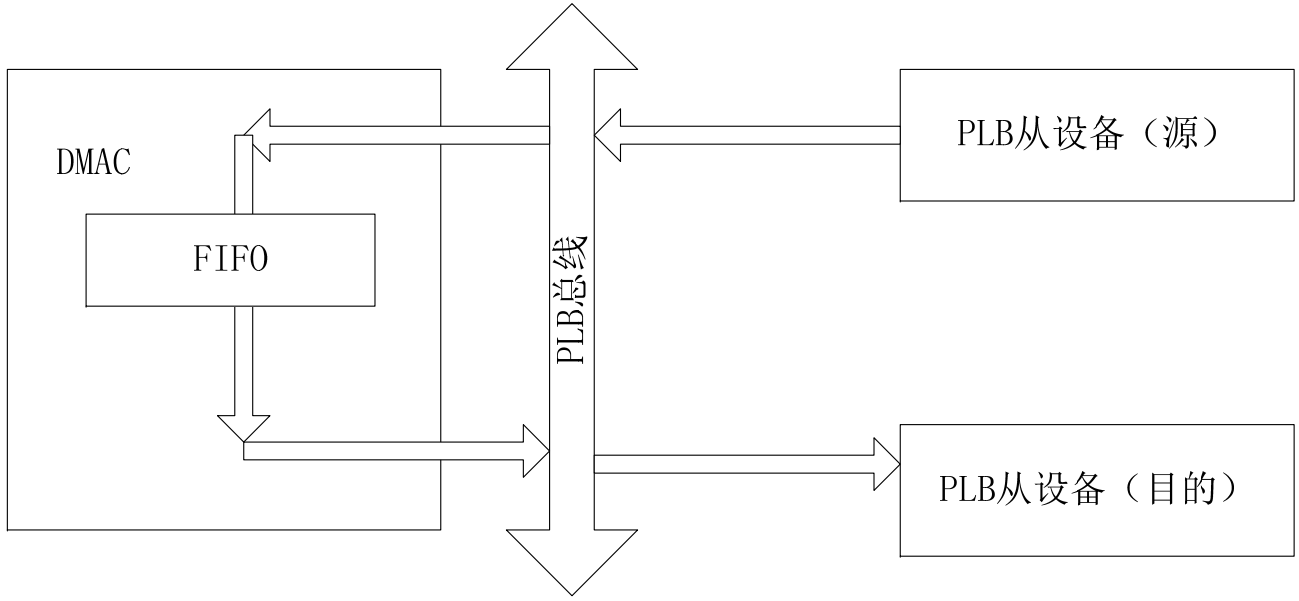
Xilinx XPS DMA控制器内部寄存器

寄存器名称	偏移地址	功能描述
RST	0X0	向该寄存器写0x0000000A，将软件复位DMA控制器
DMACR	0X4	Bit0为SINC:1,源地址增加；0,源地址不变；bit1为DINC：1,目的地址增加；0,目的地址不变
SA	0X8	保存源地址
DA	0XC	保存目的地址
LENGH	0X10	传送数据长度，一旦往该寄存器写入字节长度，DMA控制器就开始进行DMA传输
DMASR	0X14	Bit0，DMA传输忙闲状态：0没有DMA传输，1正在进行DMA传输；bit1，DMA总线出错状态：1有错，0没有错误
ISR	0X2C	Bit30，DMA出错中断；bit31，DMA传输结束中断，1表示产生了中断，向相应的位写1将清除该中断状态
IER	0X30	Bit30，DMA出错中断使能；bit31，DMA传输结束中断使能，写1允许中断，写0不允许产生中断

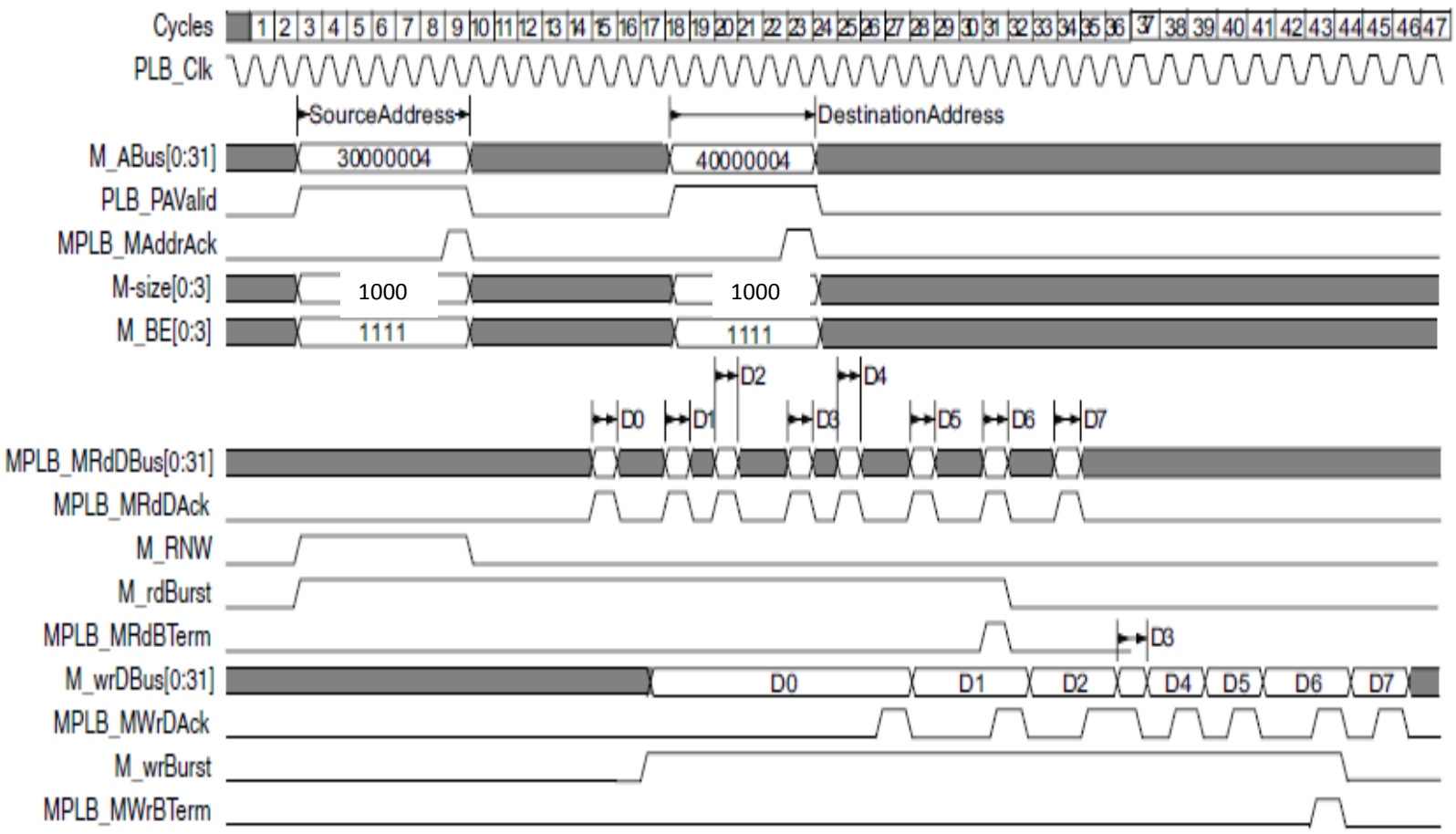
控制Xilinx XPS DMA控制器进行DMA数据传输的流程

- 写源地址寄存器SA，配置DMA传输源端起始地址；
- 写目的地址寄存器DA，配置DMA传输目的端起始地址；
- 写控制寄存器DMACR，通过写SINC和DINC控制源和目的端地址是否发生改变。
  - 若是存储器到IO接口的传输，则SINC=1，DINC=0；
  - 若是IO接口到存储器的传输，则SINC=0，DINC=1；
  - 若是存储器到存储器的传输，则SINC=1，DINC=1；
- 写传输字节长度寄存器LENGTH，将要传输的字节数写入该寄存器就启动了DMA传输。

Xilinx XPS DMA控制器DMA数据传输通路

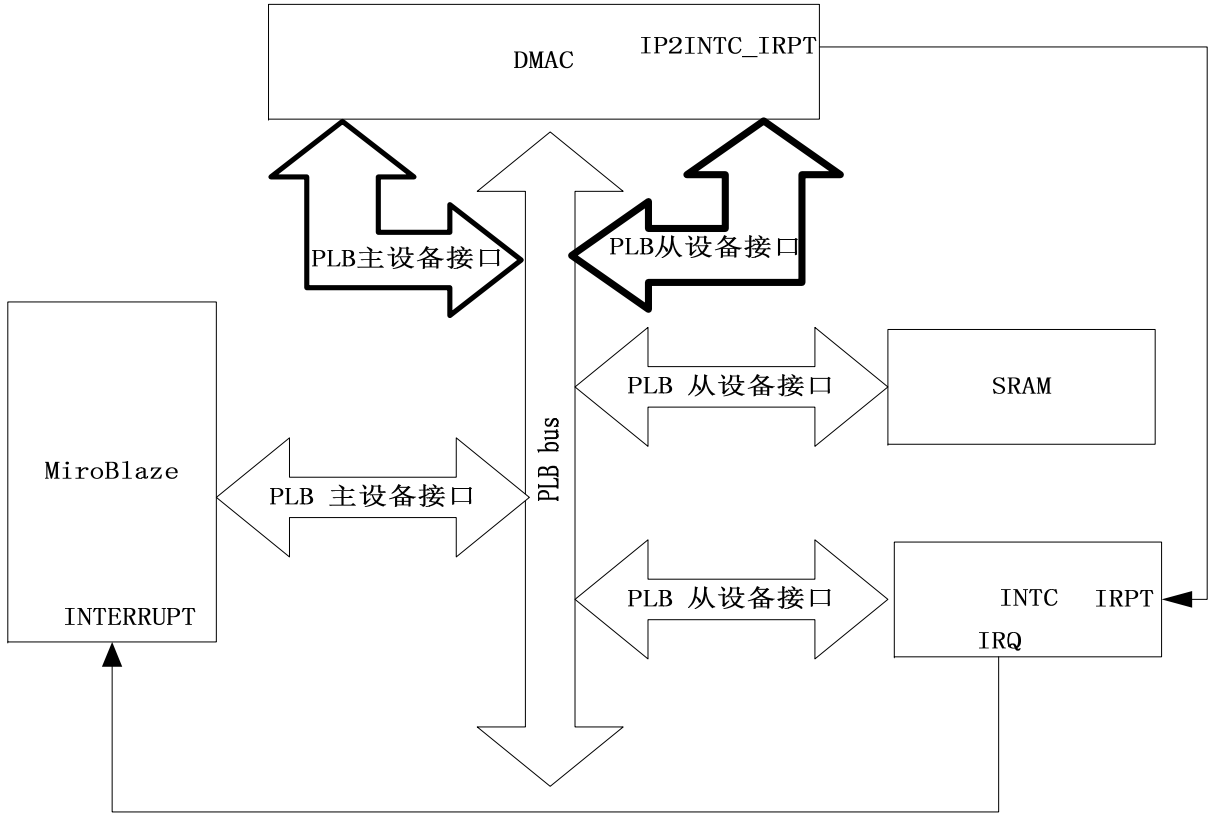


Xilinx XPS DMA控制器突发模式DMA
数据传输时序



- 已知某基于PLB总线的计算机系统采用MicroBlaze微处理器，并带有SRAM存储器，存储器地址范围为0x83000000~0x83ffffff，要求采用DMA方式将起始地址为0x83200000的1K字节的内存数据传输到地址为0x83100000的内存存储区间，传输结束时产生中断信号，当CPU接收到该中断信号后，程序退出。试设计接口电路并编写控制程序(检测数据传输的正确性)。

基于PLB总线支持DMA传输的计算机系统硬件电路连接框图



DMA API

Int XDmaCentral_CfgInitialize(XDmaCentral *InstancePtr,XDmaCentral_Config *DmaCentralCfgPtr,u32 EffectiveAddr);
//初始化配置参数
Void XDmaCentral_Reset(XDmaCentral *InstancePtr);
//复位DMA控制器
Void XDmaCentral_SetControl(XDmaCentral *InstancePtr, u32 Value);
//设置DMA控制寄存器为Value
u32 XDmaCentral_GetStatus(XDmaCentral *InstancePtr);
//读取DMA状态寄存器
voidXDmaCentral_Transfer(XDmaCentral *InstancePtr,void *SourcePtr, void *DestinationPtr, u32 ByteCount);
//设置源地址寄存器为SourcePtr，目的地址寄存器为DestinationPtr，长度寄存器为ByteCount，并启动DMA传输
XDmaCentral_Config *XDmaCentral_LookupConfig(u16 DeviceId);
//操作系统系统的统一设备ID查找DMA配置参数数据结构
Int XDmaCentral_Initialize(XDmaCentral *InstancePtr, u16 DeviceId);
//根据操作系统系统的统一设备ID，初始化DMA控制器实例
void XDmaCentral_InterruptEnableSet(XDmaCentral *InstancePtr, u32 Mask);
//使能Mask对应的DMA中断源
u32 XDmaCentral_InterruptStatusGet(XDmaCentral *InstancePtr);
//获取DMA中断请求源
void XDmaCentral_InterruptClear(XDmaCentral *InstancePtr, u32 Mask);
//清除Mask对应的DMA中断源的中断请求状态

DMA中断处理函数

```
Void XDmaHandler(void * CallBackRef){
        Xuint32 status;
//读取中断状态寄存器
status=XDmaCentral_ReadReg(XPAR_XPS_CENTRAL_DMA_0_BASEADDR,
XDMC_ISR_OFFSET);
//判断是否是DMA传输结束中断
if((status&XDMC_IXR_DMA_DONE_MASK)==XDMC_IXR_DMA_DONE_MASK )
        {dma_done=1;
//清除中断标志
        XDmaCentral_WriteReg(XPAR_XPS_CENTRAL_DMA_0_BASEADDR,

XDMC_ISR_OFFSET,XDMC_IXR_DMA_DONE_MASK);
        }
}
```

注册DMA中断处理函数到中断系统中

```
intXDmaCentral_SetupIntrSystem(XIntc*IntcInstancePtr, XDmaCentral
*DmaCentralInstancePtr, u16 IntrId)
{
        int Status;
        Status = XIntc_Initialize(IntcInstancePtr, INTC_DEVICE_ID);
        Status = XIntc_Connect(IntcInstancePtr, IntrId,
                                (XInterruptHandler) XDmaHandler,
                                (void *)DmaCentralInstancePtr);
        Status = XIntc_Start(IntcInstancePtr, XIN_REAL_MODE);
        XIntc_Enable(IntcInstancePtr, IntrId);
microblaze_register_handler((XInterruptHandler)XIntc_InterruptHandler, IntcInstancePtr);
        microblaze_enable_interrupts();
        return XST_SUCCESS;
}
```

DMA控制主函数

```
while (dma_done==0);//等待DMA传输结束中断
   //检验两块数据区的数据是否相等
for (Index = 0; Index < BUFFER_BYTESIZE;
   Index++) {
   if ( DestPtr[Index] != SrcPtr[Index]) {
   return XST_FAILURE;
   }
   }
   return XST_SUCCESS;
}
```

作业

- 6,8