第八章习题解答

1. DMA 控制器与总线仲裁器之间的联络信号有哪些？分别代表什么含义？
解答：HOLD，HLDA。HOLD：总线请求信号，DMA 控制器向总线仲裁器发出总线请求信号；HLDA：总线请求应答信号，当允许 DMA 控制器占用总线时，由总线仲裁器向 DMA 控制器发出的允许信号。

2. DMA 控制器与外设之间的联络信号有哪些？分别代表什么含义？
解答：DREQ，DACK。请求 DMA 传送由外设产生 DREQ 信号，当可以进行 DMA 传送时，DMA 控制器向外设发出 DMA 应答信号(DACK)，

3. DMA 传输的基本流程分为哪几个阶段？各个阶段分别产生什么信号？
解答：

DMA 传送可以分为以下几个阶段：

a) 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。
b) 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)。
c) 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。
d) DMA 传送阶段：DMAC 获得总线的控制权后，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(DMAC 内部设有“地址寄存器”，在 DMA 操作过程中，每传送一字节数据，DMAC 自动修改地址寄存器的值，以指向下一个内存地址)。同时，向外设发出 DMA 应答信号(DACK)，实现该外设与内存之间进行 DMA 传送。在 DMA 传送期间，DMAC 发出内存和外设的读/写信号。
e) 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。

4. CPU 在何时可以响应 DMA 传输请求？
解答：CPU 在每条指令执行的各个阶段之中都可以让给 DMA 使用，是立即响应，如取指令、取数据以及保存结果之前都可以响应 DMA 请求

5. DMA 传输与中断传输的差别是什么？
解答：两个主要的区别：
1）中断请求不但使 CPU 停下来，而且要 CPU 执行中断服务程序为中断请求服务，这个请求包括了对断点和现场的处理以及 CPU 与外设的传送，所以 CPU 付出了很多的代价。DMA 请求仅仅使 CPU 暂停一下，不需要对断点和现场的处理，并且是由 DMA 控制器控制外设与主存之间的数据传送，无需 CPU 的干预，DMA 只是借用了一点 CPU 的时间而已。
2）CPU 对这两个请求的响应时间不同，对中断请求一般都在执行完一条指令的时钟周期末尾响应，而对 DMA 的请求，考虑到它的高效性，CPU 在每条指令执行的各个阶段之中都可以让给 DMA 使用，是立即响应。

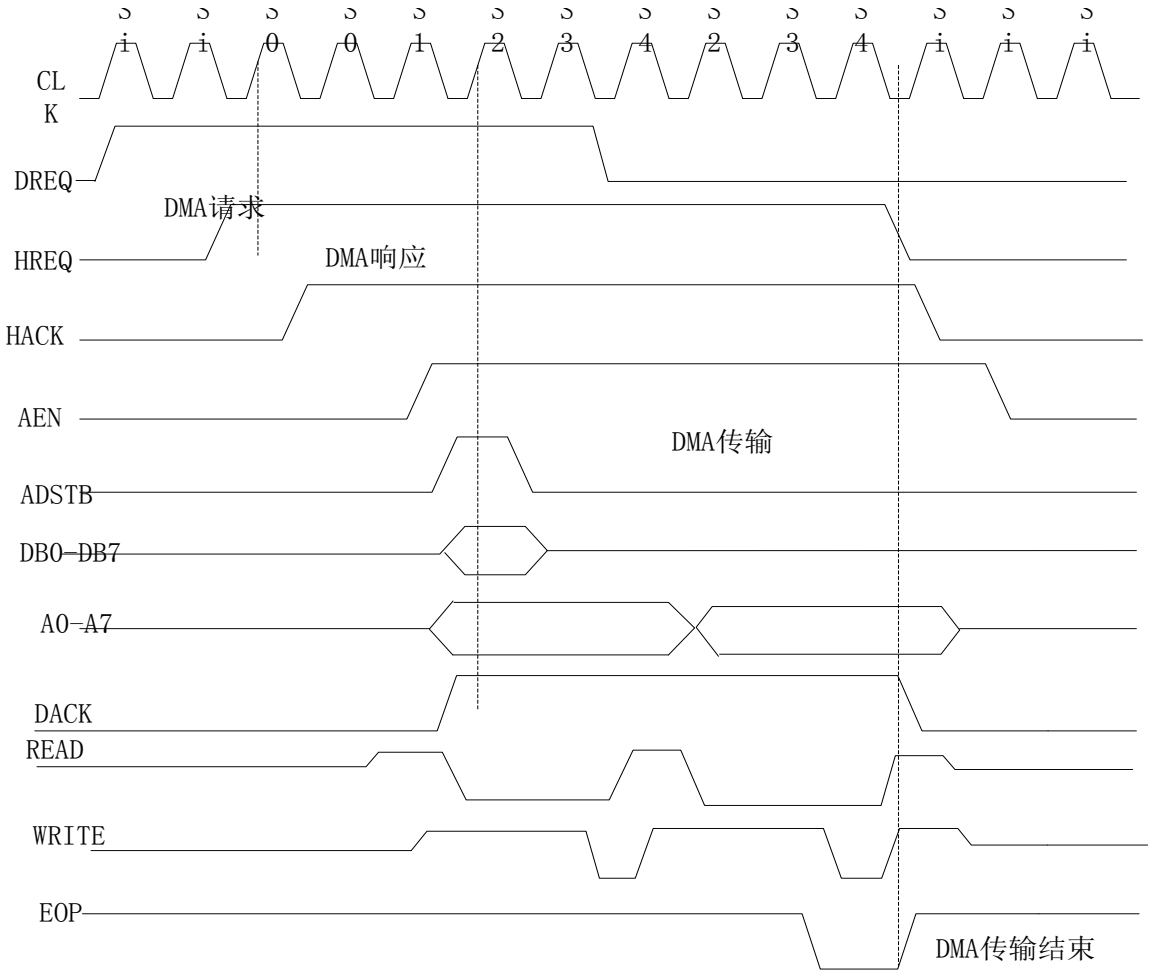
6. 基于例 8.1 所示计算机系统，试采用伪指令初始化该系统中的第二个 8237DMA 控制器，利用该 DMA 控制器的第三个通道采用 DMA 方式从某特定外设读入 0x100 个数据，并保存到地址 0x9003 7865 开始的连续内存空间。

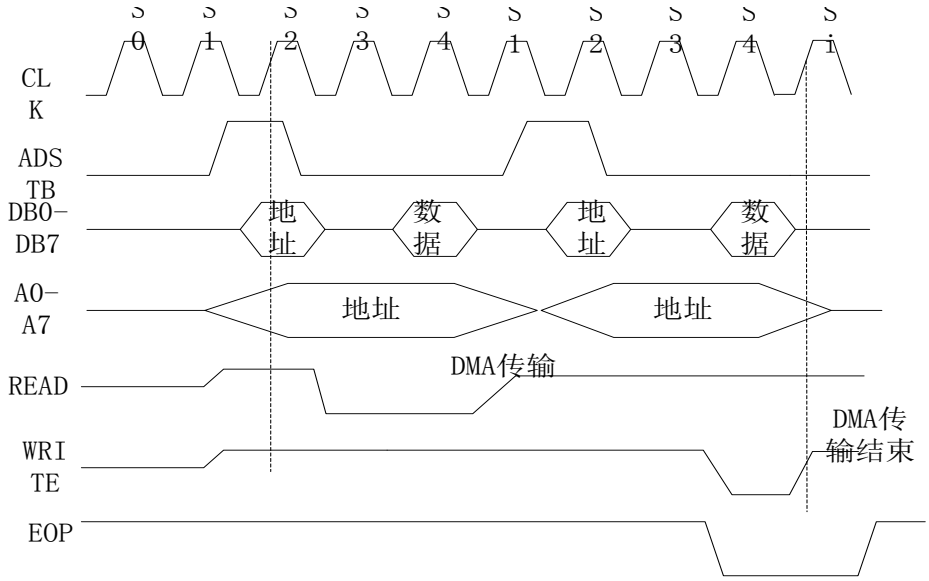
解答：第二个 8237DMA 控制器地址范围：基地址为 0x0080，使用 I/O 地址 0x0080~0x008F

```
PORTWrite(0x88,0);//对控制寄存器进行初始化
PORTWrite(0x8A,7);//屏蔽通道 3
PORTWrite(0x8C,0);//清除先/后寄存器
PORTWrite(0x86,0x65);
PORTWrite(0x86,0x78);//写基地址寄存器
PORTWrite(pageAddr,0x9003);//写页面寄存器，保存高位地址
PORTWrite(0x87,0x00);
PORTWrite(0x87,0xFF);//写基本字节寄存器
PORTWrite(0x8B,0X47);//设置模式寄存器,单字节传输，
PORTWrite(0x8A,3);//解除通道 3 屏蔽
```

7. 试将图 8- 12，图 8- 13DMA 传输时序中的各个时钟周期对应划分到 DMA 传输流程的不同阶段。

解答：





8. 基于例 8.2 编程控制采用 DMA 方式将内存起始地址为 0x8320000A 的 1K 字节的内存数据传输到地址为 0x8310000B 的 IO 接口，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。

解答：

```
#include "xparameters.h"
#include "xdmacentral.h"
#include "xil_cache.h"
#include "xintc.h"
#include "xil_exception.h"
#define DMA_DEVICE_ID      XPAR_DMACENTRAL_0_DEVICE_ID
#define INTC_DEVICE_ID      XPAR_INTC_0_DEVICE_ID
#define BUFFER_BYTESIZE 1024    //定义数据区大小为 1K
intXDmaCentral_SetupIntrSystem(XIntc *IntcInstancePtr, XDmaCentral *DmaInstance, u16 IntrId);//声明建立中断处理系统函数
void XDmaHandler(void * CallBackRef);//声明 DMA 中断处理函数
static XDmaCentralDmaCentral;    // 实例化 DMA 控制器
static XIntcIntcInstance; //实例化中断控制器
intdma_done=0;//进入中断标志
int main()
{
    int Status;
    u8 *SrcPtr;
    u8 *DestPtr;
    u32 Index;
    XDmaCentral_Config *ConfigPtr;
    //查找 DMA 驱动配置项
    ConfigPtr= XDmaCentral_LookupConfig(XPAR_XPS_CENTRAL_DMA_0_DEVICE_ID);
    if (ConfigPtr == NULL) {
        return XST_FAILURE;
```

```
        }
    //初始化驱动参数
    Status = XDmaCentral_CfgInitialize(&DmaCentral,ConfigPtr,ConfigPtr->BaseAddress);
        if (Status != XST_SUCCESS) {
            return XST_FAILURE;
        }
        //复位 DMA 控制器
        XDmaCentral_Reset(&DmaCentral);
        SrcPtr = 0x8320000A;//初始化源地址指针（修改处）
        DestPtr = 0x8310000B;//初始化目的指针（修改处）
        //初始化源内存区域数据为 0~1024 模 256 的余数，目的内存区域数据为 0
        for (Index = 0; Index < BUFFER_BYTESIZE; Index++) {
            SrcPtr[Index] = Index;
                    }
        //设置 DMA 控制寄存器使得 SINC=1.DINC=0
        XDmaCentral_SetControl(&DmaCentral,
                    XDMC_DMACR_SOURCE_INCR_MASK);（修改处）
        //使能 DMA 控制器当传输完毕时产生中断
        XDmaCentral_InterruptEnableSet(&DmaCentral,XDMC_IXR_DMA_DONE_MASK);
        //注册 DMA 中断服务程序
        XDmaCentral_SetupIntrSystem(&IntcInstance, &DmaCentral,
                XPAR_XPS_INTC_0_XPS_CENTRAL_DMA_0_IP2INTC_IRPT_INTR);
        //启动 DMA 传输
        XDmaCentral_Transfer(&DmaCentral, SrcPtr,DestPtr, BUFFER_BYTESIZE);
        while (dma_done==0);//等待 DMA 传输结束中断
        return XST_SUCCESS;
}
//DMA 中断处理函数
VoidXDmaHandler(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);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }
    Status = XIntc_Connect(IntcInstancePtr, IntrId,
                    (XInterruptHandler) XDmaHandler,
                    (void *)DmaCentralInstancePtr);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }
    Status = XIntc_Start(IntcInstancePtr, XIN_REAL_MODE);
    if (Status != XST_SUCCESS) {
    return XST_FAILURE;
    }
    XIntc_Enable(IntcInstancePtr, IntrId);
    microblaze_register_handler((XInterruptHandler)XIntc_InterruptHandler,
                                        IntcInstancePtr);
    microblaze_enable_interrupts();
    return XST_SUCCESS;
}
```

9. 什么是通道？通道的基本功能是什么？

解答：通道是一个具有输入输出控制处理器的输入输出部件。通道控制器有自己的指令，即通道命令，能够根据程序控制多个外部设备并提供了 DMA 共享的功能，而 DMA 只能进行固定数据传输操作。

其基本功能为运行输入输出控制程序，实现多个输入输出设备的数据传送。

10. 通道有哪些类型？它们的数据传送方式有什么差别？

解答：通道可分成字节多路通道、选择通道和数组多路通道三种类型。

差别：

选择通道的输入输出操作启动之后，该通道就专门用于该设备的数据传输直到操作完成。

字节多路通道和数组多路通道都是多路通道，在一段时间内可以交替地执行多个设备的通道程序，使这些设备同时工作。但两者也有区别，首先数组多路通道允许多个设备同时工作，但只允许一个设备进行传输型操作，而其他设备进行控制型操作；而字节多路通道不仅允许多个路同时操作；而且允许它们同时进行传输型操作。其次，数组多路通道与设备之间的数据传送的基本单位是数据块，通道必须为一个设备传送完一个数据块以后才能为别的设备传送数据块，而字节多路通道与设备之间的数据传送基本单位是字节。通道为一个设备传送一个字节之后，又可以为另一个设备传送一个字节，因此各设备与通道之间的数据传送是以字节为单位交替进行的。