分类号：TN431.2 密级：保密 2006.03
UDC：621.3 学号：010530

东 南 大 学
硕 士 学 位 论 文

基于 Garfield 的 SoC 功耗估计与分析

研究生姓名：殷 宏
导师姓名：陆生礼 教授

申请学位级别 硕 士 学科专业名称 微电子学与固体电子学
论文提交日期 2004 年 3 月 6 日 论文答辩日期 2004 年 3 月 14 日
学位授予单位 东 南 大 学 学位授予日期
答辩委员会主席 葉建荣 评 阅 人 葉建荣
胡 晨

2004 年 3 月 1 日

摘　要

嵌入式微处理器中存储控制器的设计非常重要，其功能与性能决定着嵌入式微处理器所支持的外部存储器的类型以及对外部存储器的访问速度，进而决定着整个嵌入式系统的处理速度。随着 AMBA 总线规范成为嵌入式微处理器内片上总线的事实标准，设计一个基于 AMBA-AHB 总线规范，支持嵌入式系统常用的存储器类型的存储控制器 IP，具有非常大的实际意义。

本文提出了一个基于 AMBA- AHB 总线规范的存储控制器的整体设计方案，支持当前嵌入式系统中最普遍使用的几种存储设备：SRAM、ROM、FLASH、SDRAM、CF 卡。在完成了整个存储控制器的整体框架设计的基础上，给出了以上各类存储器的控制器的实现原理以及各子模块的划分、设计实现。该控制器已用 Verilog HDL 语言实现并已通过基于 ARM7TDMI CPU 核和 AMBA 总线的系统仿真。

论文首先分析了各类存储器的结构、控制特点，综述了 AMBA Specification 2.0 的有关知识，及其在本文中相应的实现策略。在此基础上，进行了存储控制器的系统定义及整体结构设计。在阐述三类存储器的控制器的设计时各有侧重：SRAM（包括 FLASH、ROM）控制器模块的设计侧重于 AHB 总线规范与存储控制器的时序之间的转换解决策略；在进行 SDRAM 控制器模块的设计时，针对 SDRAM 特有的读写方式，特别是在作 BURST 操作时如何满足 AHB 总线规范，并最大程度的提高总线利用率，以及如何充分利用 SDRAM 特有的读写方式来提高总线读写速度等方面，作者提出了自己的实现方法；CF 卡控制器模块的设计侧重于如何实现 CF 控制器与 CF 卡设备之间的时序配合。最后作者总结了整个存储控制器的系统联调和仿真情况，并对存储控制器的设计技术作了展望。

关键词：嵌入式微处理器，AMBA 总线规范，存储控制器，IP

ABSTRACT

The design of Memory Controller in the embedded microprocessor is very important, because it's function and performance have very important influence to the performance of the embedded system. With the AMBA Specification becoming the de facto standard of Bus on Chip(BOC), designing a memory controller IP complying to AMBA Specification has significant practical meaning.

In this dissertation the author brings forward an entire scheme of Memory Controller complying to AMBA-AHB Specification 2.0, which could be connected to such memory as SRAM, ROM, FLASH, SDRAM, and CF card. The author gives the frame design of whole memory controller and the theory of design of every kind of memory controller, as well as the partition and design of sub-modules. Especially the author provides his own implementation method to improve bus accessing speed and bus utilizing ratio through unique access method of SDRAM.

In the first chapter, background of this dissertation and work that have been done by the author are introduced.

In the second chapter, basic structures of every kind of Random Access Memory and correlated knowledge of AMBA Specification 2.0 are introduced, and on the basis of which, the system specification and the whole structure designing of memory controller are introduced.

In the third chapter the design of SRAM controller is introduced.

In the fourth chapter the design of SDRAM controller is introduced, especially the author gives his implementation method on how to improve the bus accessing efficiency by utilizing the characteristics of SDRAM.

In the fifth chapter the design of CF card controller is introduced.

At last chapter the author summarizes the simulation of the design, and makes expectation to the design of memory controller.

KEY WORD: Embedded microprocessor, AMBA Specification, Memory Controller, IP

学位论文独创性声明

本人声明：所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签名：[illegible] 日期：2003.4.1

关于学位论文使用授权的说明

东南大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交的学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查询和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学研究生院办理。

签名：_____ 导师签名：[illegible] 日期：03.4.1

第一章 绪 论

1.1 本文的背景

当前，嵌入式计算机在数量上远远超过各种通用计算机，在工业流水线控制、通讯、仪器仪表、汽车、船舶、航空航天、军事装备、消费类产品等领域更是嵌入式计算机的天下。基于嵌入式计算机系统的全数字电视、第三代移动通讯等产品，都将在不远的将来占据相关领域市场的绝对份额。现在，嵌入式系统带来的工业年产值已超过了一万亿美元，这一数字还将继续大幅度提高。

一个嵌入式系统硬件一般是由嵌入式微处理器、外部存储器、以及相应的输入输出设备组成。在这个系统中存储器是必不可少的，系统的启动信息，系统运行所需的软件，都是保存在存储器中。随着嵌入式 CPU 运行主频越来越高，一个嵌入式系统的运行速度与其外部存储器的访问速度的关系越来越紧密。

由于嵌入式系统的特定要求，嵌入式微处理器片内必须包含一个实现对于外部存储器的访问功能的存储控制器模块。考虑到成本、面积、功耗等方面的限制，嵌入式系统使用较少的 CACHE（缓存）和较简单的存储器访问机制，因此嵌入式微处理器中的存储控制器模块的功能及性能极大的制约着嵌入式微处理器所能访问的外部存储器的类型及访问速度，进而制约着整个系统的访问外存的速度，并最终影响整个系统的处理速度[14]。因此任何嵌入式微处理器的开发设计都离不开设计一个功能完整、性能优秀的存储控制器模块。

随着嵌入式微处理器设计技术的发展，特别是随着 ARM 公司的微处理器越来越成为精简指令集计算机（RISC）设计中事实上的标准，以及 AMBA 总线规范成为最通用的片上总线规范，现在的片上系统（SOC）设计中所采用的片上总线越来越倾向于使用 AMBA 总线规范，这与 ARM 微处理器体系结构的普及也有着很大关系。AMBA 总线规范为 SOC 的设计提供了以下优点：较好的可移植和可重用设计、低功耗设计、高性能和结构可移植的系统设计、以及较好的可测性设计[1]。 而总线规范的普及又推动着可重用设计技术及 IP（Intellectual Property）设计的发展，使得在一个 SOC 设计流程中，IP 核扮演着越来越重要的作用[2]。

所谓 IP 核，就是一些经过验证的，实现一定功能的设计单元（Design Unit）。比如 ARM 公司的 CORE—ARM7TDMI，就是一个典型的 IP。IP 分为硬核、固核、软核。硬核由流片厂提供，只提供仿真模型供开发之用；固核是以网表的形式提供的；软核是以可综合的硬件描

述语言的形式给出的，可以供仿真、综合、优化[3]。 IP 构成了 SOC 设计中可重用技术的主体，在 SOC 的设计过程中，丰富的 IP 资源可以极大的缩短系统的设计、实现周期[2]。

在以上这些背景下，使得开发一个基于 AMBA-AHB 总线规范的存储控制器（IP）成为可能，并具有非常大的实际意义。因为只要完全遵照 AMBA 总线规范设计，那么对这个存储控制器设计只要作较小的改动，就可以应用在任何符合 AMBA 总线规范的微处理器设计中。

作者参与了所在的东南大学 ASIC 系统工程技术研究中心承担的一个项目：《基于目标产品的 SOC 芯片设计》，并在其中承担了存储控制器模块的设计任务，因此有机会在这一方面进行了一些探索实践。

1.2 本文的主要工作

存储控制器是嵌入式微处理器中 AMBA-AHB 总线与片外存储设备之间的接口，以完成总线主设备（如 CPU CORE 或 DMA）与片外存储设备的数据传输。本课题的目标是利用 Verilog HDL 语言设计一个基于 AMBA-AHB 总线规范，支持对片外各类存储设备实现读写控制的存储控制器。

要使所设计的存储控制器具有实用性和通用性，就必须支持当前嵌入式系统中最广泛使用的存储器类型。基于这一出发点，作者调研了当前嵌入式系统中一些常用的嵌入式微处理器所支持的各种存储器类型及当前嵌入式系统中常用的存储器类型，得到如下结论：

1、SRAM、FLASH、ROM 是嵌入式系统中常用的存储器类型，一些系统掉电后必须保存的数据一般都是保存在 FLASH 中，而系统引导程序是保存在 ROM 中。它们的读写时序与 SRAM 基本一致，所以在这个设计中提供了对 SRAM、FLASH、ROM 的支持。

2、SDRAM 由于其大存储容量和快速的读写速度，是当前普遍使用的随机存取存储器，因此必须提供对 SDRAM 的支持。

3、CF 卡是当前较普遍使用的外部存储设备，而且 CF 卡接口是一种较流行的 I/O 设备接口，许多外设都采用 CF 卡规范作为接口规范，因此提供对 CF 卡接口的支持，也就给系统提供了较强的对外设的扩展能力。

基于以上几点，要实现的存储控制器应该提供对 SRAM（包括 ROM、FLASH）、SDRAM、CF 卡设备的支持。具体工作包括进行整个存储控制器的整体设计规划及各个功能模块的实现。特别是，SDRAM 的读写时序较复杂，针对 SDRAM 特有的读写方式，特别是在作 BURST 操作时如何满足 AHB 总线规范并最大程度的提高总线利用率，以及充分利用 SDRAM 特有的读写方式来提高总线读写速度等方面，也需提出合理的解决方法。本文对以上一些问题都有具体论述。

本文首先分析了嵌入式系统中常用存储器的结构及操作特点，以及当前 SOC（片上系统）设计中的主流总线规范：AMBA 的特点，在此基础上，针对它们的特点，进行了存储控制器整体框架的设计。在这之后，分三章分别阐述了针对三类存储设备的控制模块的设计，在每个控制子模块的阐述中，仍旧采用先介绍模块顶层设计，然后介绍各个底层子模块的顺序，全文都是采用这种自顶向下的思路来展开的。在本文最后对所实现的存储控制器的仿真验证情况进行了总结，并对存储控制器的设计技术作了展望。

第二章　整体设计框架的确定

2.1 存储器知识

我们来回顾一下存储器的发展历程：

存储器可分为只读存储器(ROM)和随机读写存储器(RAM)，随机读写存储器包括 SRAM、DRAM、SDRAM 等。另外，FLASH 也是一种可读写的，非挥发性存储器，也即切断电源后，仍能保存已经写入存储器中的数据，而其他随机读写存储器不具备这一特点，即断电后所保存的数据就丢失了。

SRAM 也就是静态 RAM，是一种挥发性存储器，切断电源后所写入的数据也将丢失。一般 SRAM 采用 4（晶体）管或 6（晶体）管结构，在保持电源的情况下，每个存储单元采用内部反馈电路来保持所写入的数据。下图为一个 SRAM 存储单元的示意图[4]。

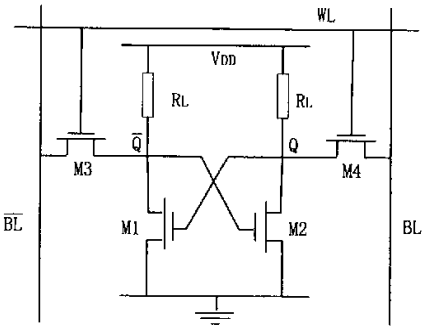


图 2.1　电阻负载 SRAM 单元

从上图可以看出，只要提供 VDD，则位线 BL 上的数据通过 M4 写入存储单元，即数据 Q，并通过由 M1 管构成的反馈电路得以保持。读操作时数据通过 M3 输出，是 Q 的取反。

ROM、FLASH、SRAM 这几种存储器的读写时序基本一致，可以通过同一个存储控制器实现读写。

上图中负载电阻 RL 的作用是补充存储单元中因为漏电流而丢失的电荷，如果能够通过周期性的向存储单元中写入数据，也即通过刷新操作来弥补丢失的电荷，那么这部分负载电路就可以去掉。为了实现刷新操作而增加的那部分电路规模如果远小于去掉的负载电路，那么就可以大大减小存储器电路的复杂性，从而减小存储器单个存储单元的面积，进而提高存储器的存储容量。这就是动态存储器（DRAM）的原理。

下面两图为动态存储器的两种存储单元结构。

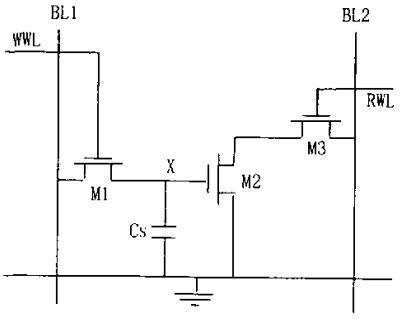


图 2.2 三管动态存储器单元

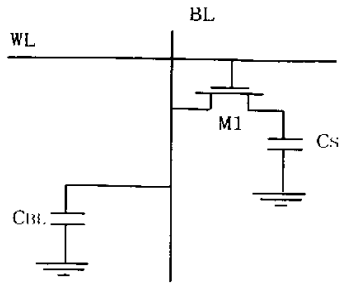


图 2.3 单管动态存储器单元

以三管 DRAM 存储单元为例，作写操作时，字写控制线 WWL 有效，数据线 BL1 上的数据被写入存储单元，写入的数据是通过电容 C_S 上的电荷来保存的。作读操作时，字读控制线 RWL 有效，晶体管 M2 是否导通取决于所保存的数据值，数据线 BL2 被提前预充电，使其电平被拉升至 VDD。因此，当所保存的数据为 1 时，M2、M3 导通，BL2 为低，当所保存的数据为 0 时，BL2 保持为高，即 BL2 上输出的数据为存储单元所保存的数据的反相，这个数据再经过一次反相后输出至存储器片外即可。实现刷新的方法是周期性的读出所保存的数据，再写回存储单元以弥补电荷的泄漏。

可以将三管结构中的 WWL 线和 RWL 线合并，这样在读出数据的同时，就完成了刷新操作。

单管 DRAM 存储单元是最普遍使用的 DRAM 结构，它更大程度的减小了电路的复杂性，从而使单片 DRAM 存储器的存储容量更大。以图 2.3 中的单管存储单元为例，当字线 WL 为高时，根据 BL 线上的数据，电容 C_S 被充电或放电，从而写入 1 或 0。作读操作之前，位线 BL 被预充电至电平 V_{PRE}。作读操作时，字线被置为有效，从而导致位线 BL 和存储电容 C_S 之间发生电荷重新分配，进而导致位线上的电平的改变。电平改变的大小由下式决定[5]：

$$\triangle V=V_{BL}-V_{PRE}=(V_{BIT}-V_{PRE})*C_S/(C_S+C_{BL})$$

上式中，V_{BL} 是电荷重新分配后位线 BL 上的电平，V_{BIT} 是读操作之前电容 C_S 上的电平。由于 C_S 较之 C_{BL} 小的多，因此这一电平的改变值并不大，必须通过差分敏感放大器放大至 V_{DD}。下图为一个示意图。

作读操作时，位线 BL 上的电平变化经差分敏感放大器放大后，被读出，同时又写回存储单元，以防止存储单元内的数据丢失。

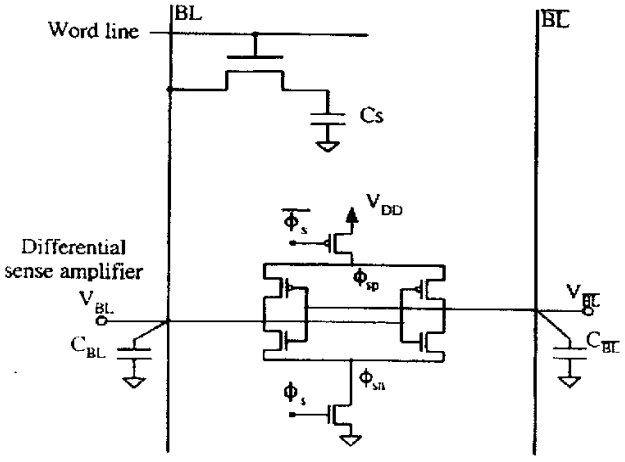


图 2.4　单管 DRAM 读操作示意图

动态 RAM（DRAM）的地址线与 SRAM 相比有很大区别，DRAM 通过将地址截为行地址和列地址，分两次发到片内，并且复用同一组地址线，从而大大减少了芯片封装的管脚数。

DRAM 先后经历了以下几种类型的变革：

最早出现 FPM (FAST PAGE MODE) DRAM，允许在打开一行（row）以后，只要通过提供累加的列（column）地址，就可以依次访问整行，乃至整页（page）的内容，这样可以极大的提高对存储器的访问速度。

EDO (EXTENDED DATA OUT) DRAM 在与 FPM DRAM 兼容的基础上，扩展了读操作（read）时数据在存储器数据线上的有效时间，这就使得 DRAM 控制器有更多时间从数据线上得到数据。

当 CPU 与系统的速度飞速向前发展时，这些老的内存技术显得无力跟上时代潮流，SDRAM 技术的出现正是基于如此背景之下而产生的。

SDRAM 是在标准动态存储器中加入同步控制逻辑(一个状态机)，利用一个单一的系统时钟同步所有的地址数据和控制信号。使用 SDRAM 不但能提高系统表现，还能简化设计、 提供高速的数据传输。 在功能上，它类似常规的 DRAM，且也需时钟进行刷新。 可以说，SDRAM 是一种改善了结构的增强型 DRAM。SDRAM 使用一个外部时钟来同步对它的命令输入，及数据读写，这就使得 SDRAM 与 SDRAM 控制器配合的时序更加精确。另外，在 SDRAM 芯片内部提供了一个模式寄存器，用户可以配置访问模式及访问时序。

总之，在各系统中主存储器的主要目标就是以最低、最有效的成本，提供最大的容量供系统使用。

2.2 存储控制器与 AMBA-AHB 总线的关系

2.2.1 AMBA-AHB 总线介绍

AMBA(Advanced Microcontroller Bus Architecture)总线规范为设计嵌入式微控制器定义了一套片上通讯标准，用户可以各自独立设计基于这个规范的微处理器以及外围 IP，提高了系统的开发效率及模块的可重用性，给嵌入式微控制器的设计带来了一场革命[6]。同时也使本课题具有了更广的实用价值。

在这个规范中包括三种总线：

- the Advanced High-performance Bus(AHB)
- the Advanced System Bus(ASB)
- the Advanced Peripheral Bus(APB)

其中 AHB 总线用于高性能、高时钟频率的系统模块，它保证了处理器与片内存储器、片外存储器的有效连接。因此本课题要设计的存储器控制器就与 AHB 总线连接。

一个典型的基于 AMBA 总线规范的嵌入式微控制器结构如下：

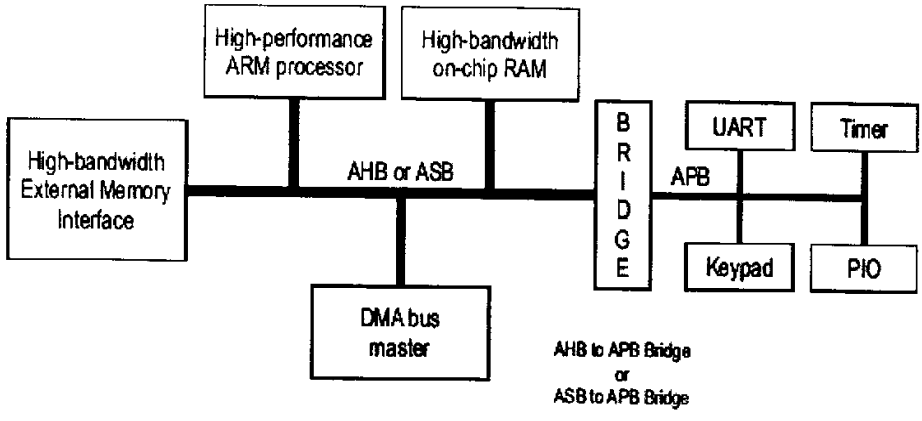


图 2.5 一个典型的 AMBA 系统

从上图可以看出，典型的基于 AMBA 总线的微控制器包含一个作为系统骨架的 AHB（或 ASB）总线，通过它来提供 CPU 或 DMA 模块与片内存储器或片外存储器之间的大数据量的通讯，以获得大的带宽。另外在 AHB 上还有一个桥（Bridge），用来连接 AHB/ASB 与 APB。大部分外部设备模块都挂在 APB 总线上。

一个高性能、高时钟频率的系统要求要实现包括 Burst 传输等特点，可以有多个主设备(Master)发起传输请求，这些，AHB 都实现了。一般嵌入式微控制器中主设备包括 CPU 及

DMA(Direct Memory Access)。

一个典型的 AHB 系统包括以下部分：

- AHB Master　主设备发起读写请求，发出读写地址及控制信号，任何时间只能有一个主设备使用 AHB 总线。
- AHB Slave　从设备响应对特定地址空间的读写请求，并通过响应信号表示读写传输的成功、失败或等待状态。
- AHB Arbiter　总线裁决器用来决定某个时刻由哪个主设备占有总线，并确保在某个时刻只有一个主设备占有总线。
- AHB Decoder　总线译码器对每次总线传输的地址进行译码，以决定当前是对哪个从设备进行读写，从而控制 MUX，来选择合适的从设备发出的读数据及响应信号，并提供从设备选中信号。

下图为 AHB 总线中各组成部分的示意图。

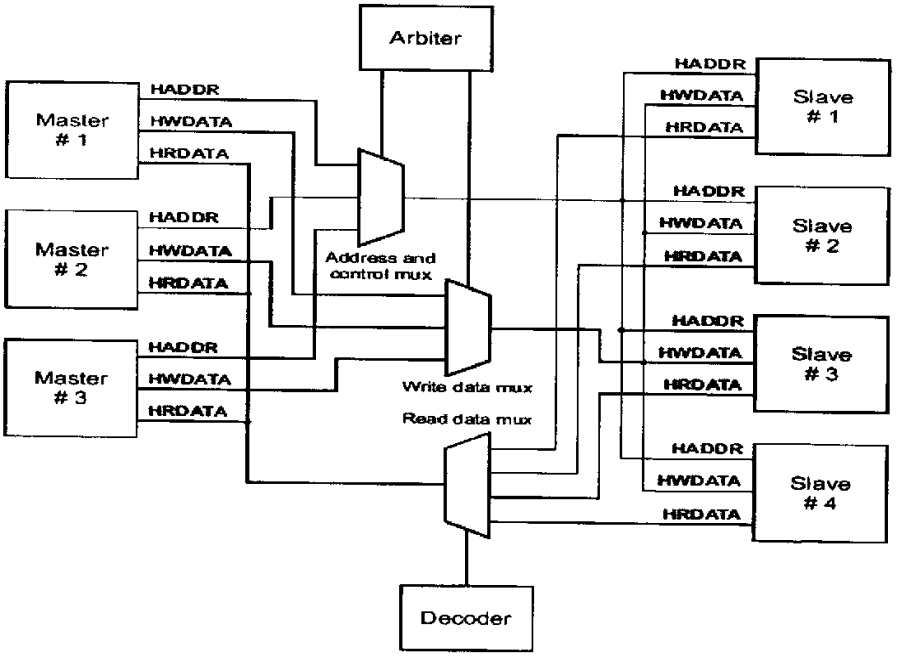


图 2.6 一个 AHB 总线系统的例子

本文要实现的即是一个挂在 AHB 总线上的从设备的设计。

2.2.2 AMBA-AHB 相关信号介绍

AHB 信号中与从设备有关的信号介绍如下：

信号名	来源	功能
HCLK	PLL	总线时钟，以上升沿为触发沿。
HRESETn	Reset controller	总线复位信号，低有效。
HADDR[31:0]	Master	32 位总线地址。
HTRANS[1:0]	Master	表示当前的传输类型，可以是 NONSEQUENTIAL、SEQUENTIAL、IDLE 或 BUSY。
HWRITE	Master	表示当前的总线传输是读还是写操作：为高时是写，为低时是读。
HSIZE[2:0]	Master	表示当前总线传输的位宽，可以是 8 位,16 位,32 位等。
HBURST[2:0]	Master	表示当前传输是否是一种 BURST 传输，可以是 4、8、16 拍的递增或卷页传输。
HPROT[3:0]	Master	表示当前总线传输是指令预取还是数据访问，是特权模式访问还是用户模式访问。
HWDATA[31:0]	Master	写操作时传输主设备到从设备的写数据。
HSEL	Decoder	每一个 AHB 从设备都有自己的选中信号，表示当前传输是针对哪个从设备的。
HRDATA[31:0]	Slave	读操作时传输从设备到主设备的读数据。
HREADY	Slave	当此信号为高时，表示一次传输完成。从设备通过将此信号置低可以延长一次传输。
HRESP[1:0]	Slave	从设备用此信号来表示一次传输的状态，可以是 OKAY、ERROR、RETRY、SPLIT，这里只实现前两种。

AMBA 总线规范通过分别使用写数据总线 HWDATA 和读数据总线 HRDATA 来传送读写数据，可以避免片内三态总线的存在，为芯片的后续设计提供便利。

一、关于传输类型（HTRANS）

AHB 总线规范中共有四种传输类型，由总线主设备决定发起哪一类型传输，用 HTRANS 信号线表示，具体有以下几种：

HTRANS[1:0]	类型	描述

00	IDLE	表示当前没有传输请求，从设备以 OKAY 作为响应信号。
01	BUSY	主设备在一次 BURST 访问中间插入 BUSY 状态，表示主设备还在继续进行 BURST 操作，但当前无法进行下一个传输。从设备以 OKAY 作为响应信号，但不动作。
10	NONSEQ	表示当前传输是一次 BURST 传输的第一拍，或者是一次单拍的传输。当前的地址和控制信号与前一拍的地址和控制信号无关。单拍的传输可以看作是一次单 拍的 BURST 传输。
11	SEQ	一次 BURST 传输中第一拍以外的其他传输都是 SEQ，此时的地址与前一拍的地址相关（累加），控制信号与前一拍相同。

二、关于 BURST 操作

BURST 操作是 AHB 总线特有的一种传输形式，是对一块地址累增的地址空间的连续的读写操作，这一组传输的位数相同，每次传输的位数由 HSIZE[2:0]决定。BURST 传输模式对于 AHB 总线的好处在于：当总线主设备发起一组 BURST 传输请求，并得到总线使用权之后，就可以一直占用总线，而不需反复向总线 ARBITER 申请总线，这样可以极大提高总线效率。

有八种类型的 BURST 传输，由 HBURST[2:0]信号区分，具体如下：

HBURST[2:0]	Type	Description
000	SINGLE	Single transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

又可以分为三类：Single、Incrementing Bursts、Wrapping Bursts。其中 Incrementing Bursts 每次的地址是在前一次传输的地址基础上增加得到的，而 Wrapping Bursts 在地址到达某个边界后，将发生卷页，即地址回到起始点。例如，一个 4 拍的宽度为字(WORD)的 BURST 传输，在地址到达 16 个字节的边界时，将发生卷页。

可以看出，BURST 传输这种模式非常适用于 SDRAM，因为 SDRAM 可以实现连续的、

单时钟周期的、地址累加的读写操作，通过 BURST 传输这种模式可以极大的利用 SDRAM 的这种特性，从而提高总线的传输速率和效率。但 SDRAM 不支持卷页模式的 BURST 读写，因此，在本课题的实现当中，不支持卷页这种 BURST 传输模式。

一次 BURST 传输是以 HTRANS 信号为 NONSEQ 状态开始的，后面应该依次为 SEQ 或 BUSY，直到一次传输完成。但如果 HTRANS 在中间变为 NONSEQ 或 IDLE，则表示一个新的 BURST 操作已经开始，上次的 BURST 操作被中止了。

从设备应能根据 HTRANS 的变化来判断 BURST 操作的状态。

三、关于从设备传输响应

HRESP[1:0]信号用来表示当前传输是否成功完成，有四种响应：OKAY、ERROR、RETRY、SPLIT。其中 OKAY 表示传输成功完成，另外，当从设备将 HREADY 置为低以延迟传输时，也同时发出 OKAY 响应。ERROR 表示当前传输发生错误。根据实际需要，在本文中只实现这两种响应，因为总线规范中的另两种响应不适用于存储控制器的设计。

OKAY 响应可以在一个时钟周期内完成，而 ERROR 响应至少需要两个周期。从设备可以通过将 HREADY 置为低来插入等待周期，以获得时间来决定应该发出何种响应。在倒数第二个周期，从设备将 HRESP 置为合适的响应状态，同时将 HREADY 置为低，以再延迟一拍。在倒数第一个周期，从设备维持 HRESP 为合适响应状态，同时将 HREADY 置为高，结束本次传输。

四、其他相关信号

传输位宽

HSIZE[2:0] 表示每次传输的大小（位宽），AMBA 2.0 规范支持从 8 位到 1024 位传输宽度，在本课题中只支持 8、16、32 位三种传输宽度。

保护控制

HPROT[3:0] 提供传输的保护信息，指出本次传输是取指令，还是数据传输，是用户模式，还是特权模式。

2.2.3 AHB 从设备

一、AHB 从设备接口

如下图所示：

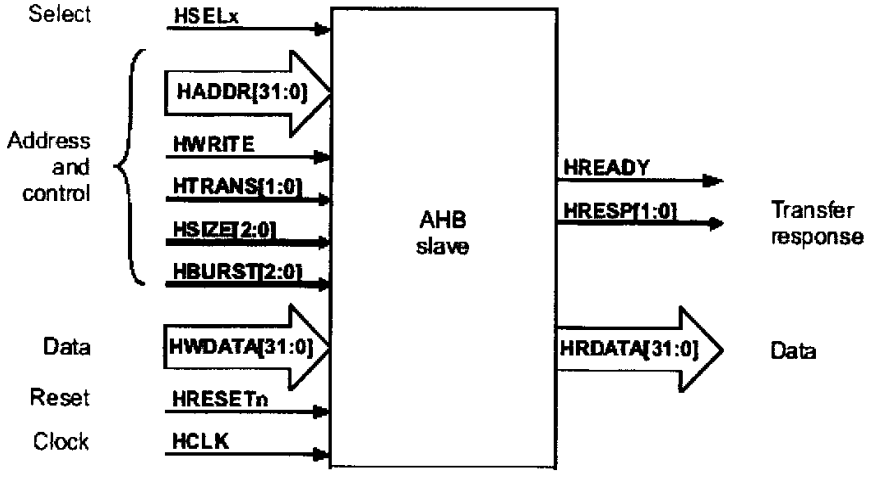


图 2.7　AHB 总线从设备接口示意图

上图中，左边部分的信号为从设备（SLAVE）的输入信号，右边部分为输出信号。这些只是从设备与 AMBA-AHB 总线之间的交互信号，不包括从设备与片外设备的交互信号。

二、AHB 从设备接口时序

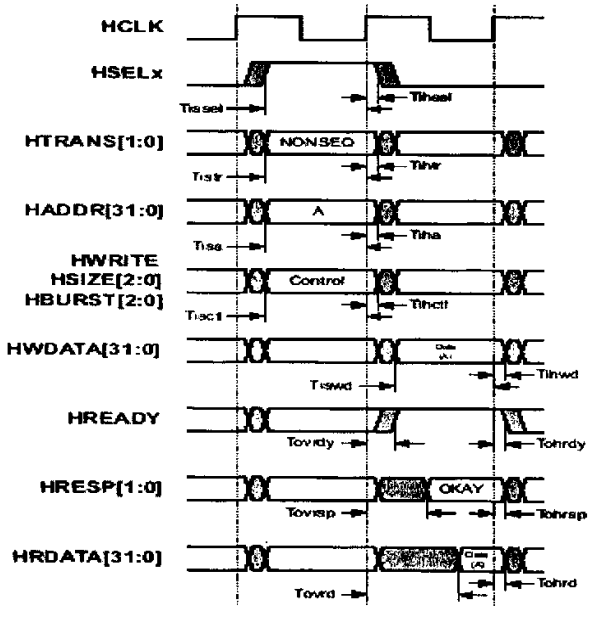


图 2.8　AHB 从设备接口时序示意图

从上图可以看出，HSEL、HTRANS、HWRITE、HSIZE、HBURST 等控制信号在第一个

时钟周期中发出，如果是写操作，则写数据 HWDATA 是在第二个时钟周期发出。被选中的从设备在第二个时钟周期的上升沿锁存到这些控制信号后，要判断是否能在一个时钟周期内完成读写操作，若能，则将 HREADY 置为高，表示要求主设备在下一周期继续发下一次的读写请求；若从设备不能在一个时钟周期内完成读写，则将 HREADY 置为低，表示要将本次读写操作周期延长。AHB 主设备在第三个周期上升沿锁存到这个 HREADY 的低电平后，将继续维持本次传输的数据不变，一直到从设备完成本次传输后，将 HREADY 置为高，则主设备在下一个时钟周期的上升沿发出下一个传输请求。

上图所示，也是一次基本的传输，没有等待周期，从设备在一个时钟周期即完成了读写操作。下图为一次插入两个等待周期的传输。

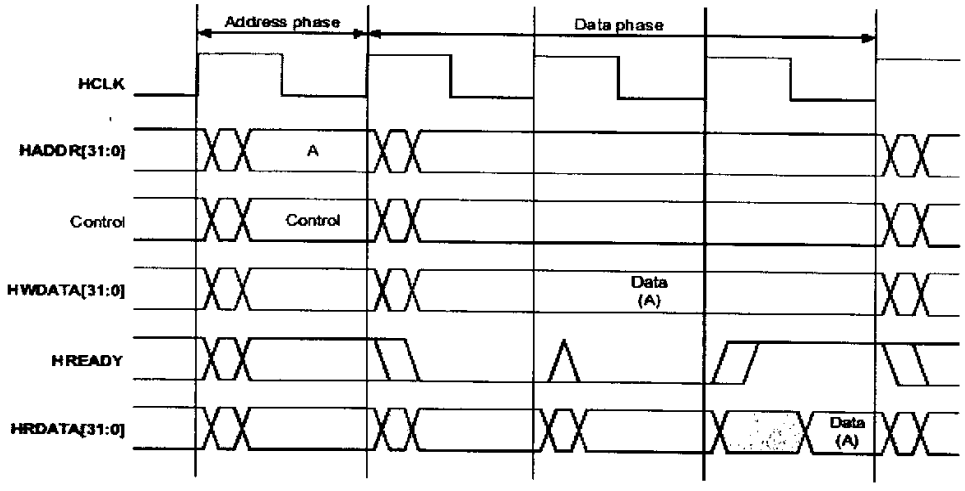


图 2.9 插入等待周期的传输

2.3 存储控制器的构成

存储控制器处在片内 AHB 总线与片外存储设备之间，作为 AHB 总线从设备，它一边需要与 AHB 总线通讯，因此必须符合 AHB 总线规范，另一边又需要给外部存储设备提供控制信号接口，以实现对外部存储设备的读写操作。存储控制器接收到来自总线主设备的符合 AMBA - AHB 总线规范的数据传输请求，对地址译码后，在所有的外部存储设备中选择当前待操作的设备，并对其产生正确的片选信号及读、写控制信号，以完成总线的数据传输请求。因此，存储控制器的设计包括以下几部分：（见图 2.10）

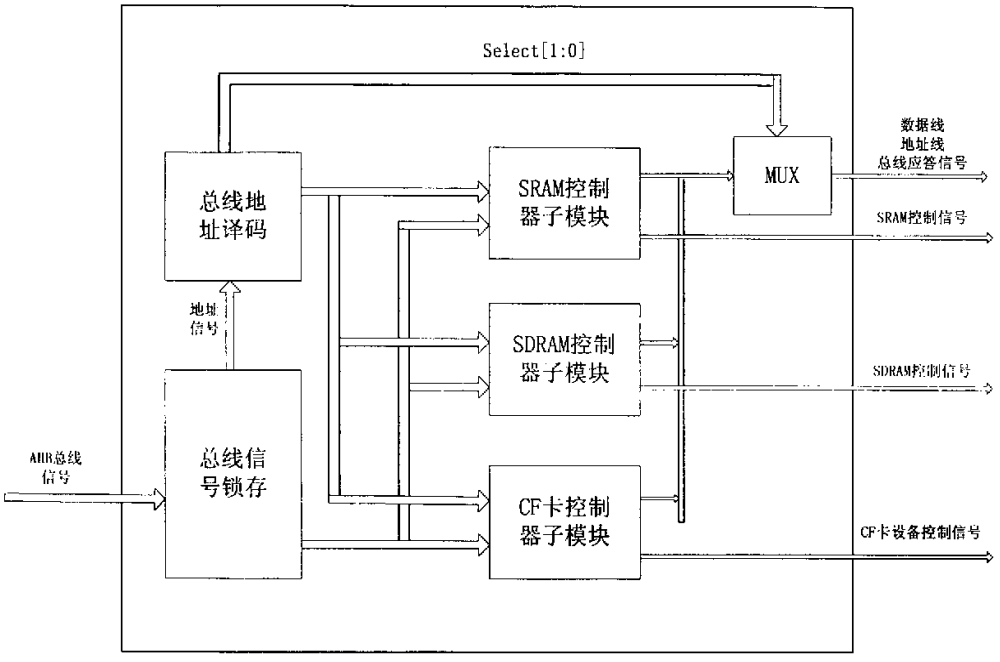


图 2.10 存储控制器整体结构框图

1、总线信号锁存

按照 AMBA 总线规范要求，AHB 总线从设备(Slave)在总线时钟上升沿，在 HREADY 信号（由从设备发出）为高的情况下，必须锁存来自 AHB 总线的总线控制信号、地址信号，以供内部译码模块及其他各模块使用。

这一部分的设计很容易实现，为一组触发器，以总线时钟 HCLK 作为触发时钟，在时钟上升沿锁存数据端数据。此时若 HREADY 为高，则锁存来自总线的信号；若 HREADY 为低，则锁存本触发器的输出，即维持原值。

2、总线地址译码

在一次读写操作中，对来自总线的读写地址进行译码，以判断是对哪个片选进行访问，

并产生相应的内部片选信号。

3、对应于三类存储设备各自控制、应答信号的产生

根据地址译码结果，相应的控制电路产生各自对应的存储设备的控制信号，以完成对存储器的读写操作请求，并且同时产生正确的 AHB 总线信号，以满足 AHB 总线规范。

这一部分可分为三部分，分别是 SRAM（包括 ROM、FLASH）控制部分、SDRAM 控制部分、CF 卡设备控制部分。这部分是整个存储控制器的主体，在本文中分三章分别予以介绍。

4、各子模块输出信号的选择

三个控制信号产生子模块（SRAM 控制器子模块、SDRAM 控制器子模块、CF 卡控制器子模块）会分别产生自己的 AHB 总线应答信号和送往外部存储器的地址、数据信号，而在某个时间，只能是对某一种存储设备进行访问，即这些信号只能来自一个控制子模块，因此需要在这几组信号中进行选择（仲裁）。这部分的设计采用以下方案：

在三类存储设备的控制模块内分别产生各自的 HREADY 信号，然后将这三个 HREADY 信号作“与”操作（AND）后送至 AHB 总线，作为整个存储控制器的 HREADY 信号。因为在某类存储设备没有被访问时，相应控制器模块一直将 HREADY 置为高；被访问的存储设备的控制器在没有完成当前的读写操作之前会将 HREADY 置为低，直到当前读写操作完成后即将 HREADY 置为高。因此作“与”之后的 HREADY 信号可以表示当前整个存储控制器的状态，而且可以满足在一个时钟周期内对总线传输请求产生应答的要求。

对于其他的几个复用信号，各个子模块都分别产生各自的 AHB 总线响应信号 HRESP，数据线，地址线，然后经过选择后作为整个存储控制器的输出。选择信号来自存储控制器的地址译码模块，根据当前来自 AHB 总线的数据传输请求的地址，译码后得到选择信号 SELECT[1:0]，由这个信号决定当前选择来自哪个控制器子模块的信号作为整个存储控制器的输出。

微处理器外部与存储器之间的数据线都是双向的，但芯片内部的数据线都采用单向的。在内部使用三组相关信号：DOUT（单向输出），DIN（单向输入），PADDIR（方向），将这三组信号送到芯片的 PAD 上来完成单向转双向，这样可以避免在芯片内部使用三态结构。因此在存储控制器的设计中只要提供这三组信号接口就可以了。

本文介绍的存储控制器最终实现了以下特性：

- 符合 AMBA Specification 2.0 总线规范
- 提供 8 个外部片选，支持 ROM、SRAM、FLASH 等存储器。

- 在这 8 个片选中，有 4 个片选支持 SDRAM 存储器，支持内部有 4 个 BANK，并允许同时激活最多达 4 个 BANK 的 SDRAM 存储器，。
- 以上所有片选支持的外部存储器的数据总线宽度可以是 16 位和 32 位两种宽度。
- 根据选用的 SDRAM 型号，tRC、tRP、tRCD、CAL latency 等时间参数可配置。
- 支持 SDRAM 的低功耗模式，及自刷新功能。
- 支持单个 16 位 CF 卡插槽，支持对卡的 8 位、16 位访问。

2.4 关于内存地址映射

分配给外部存储器的内存地址空间在不同的系统中是不相同的，因此要求是可配的，配置工作是通过写一组地址配置寄存器来完成的。具体来说就是配置每组外部存储器的基地址（即起始地址）和地址空间的大小，以及每组存储器对应的存储器类型，因为存储控制器的地址译码模块是根据这些参数来进行地址译码的。内存地址的配置工作是在系统启动后由用户完成的。在没有对存储空间进行配置之前，各个存储设备是不能使用的（启动时的引导程序所在的存储器较特殊，解释见下面）。对于 CF 卡，有特别的处理方式，具体见 CF 卡控制器模块的设计部分。

存在一个问题：存储控制器中的各个内部寄存器（配置/控制寄存器）如何被寻址？因为在不同系统中， 分配给 AMBA 总线的各个从设备（slave）的地址是不同的，为了增强存储控制器（IP）的硬件可重用性和软件可移植性，在存储控制器硬件设计中，这些内部配置寄存器的准确地址没有被指定，但它们相对于存储控制器模块在系统中的起始地址的相对地址偏移量是固定的，这样也有利于存储控制器模块的硬件设计。

另一个问题是系统启动时，系统如何加载启动程序，因为这一工作也是由存储控制器来完成的。这里采用的方式是在硬件上将片选 CSA0 进行双映射，同时用 CSA0 作为引导程序所在的存储器的选中信号（也就是片选）。所谓双映射，是指 CSA0 这个片选所对应的地址空间，除了可以通过地址配置寄存器配置以外，还自动将整个系统地址空间的最前面一段地址（例如 0～1M 空间）分配给 CSA0。一般系统复位后，CPU CORE 都会发出对这段地址空间内的读请求，以取得启动程序。当 CPU CORE 通过总线发起对这段地址空间的读请求后，存储控制器便会使 CSA0 选中，则系统从连接在 CSA0 上的存储器里读取启动程序。这样，当对片选 CSA0 的地址空间作完配置以后，片选 CSA0 将对应于两段地址空间，这就是所谓的双映射（DOUBLE MAPPING）。

第三章　SRAM 控制器模块设计

因为 SRAM、ROM、FLASH 的读写时序基本一致，在后面的介绍中就用 SRAM 代表，不同之处将特别指出。

3.1 SRAM 的时序要求

SRAM 的读写时序较简单，下图为读时序：

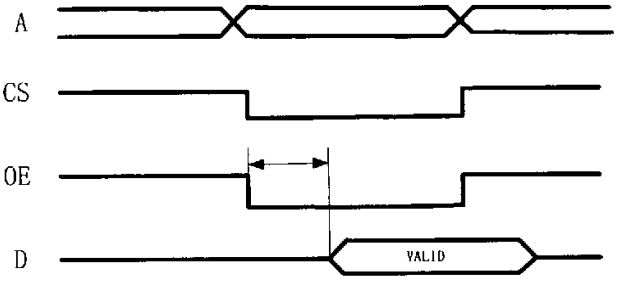


图 3.1　SRAM 读时序

下图为写时序：

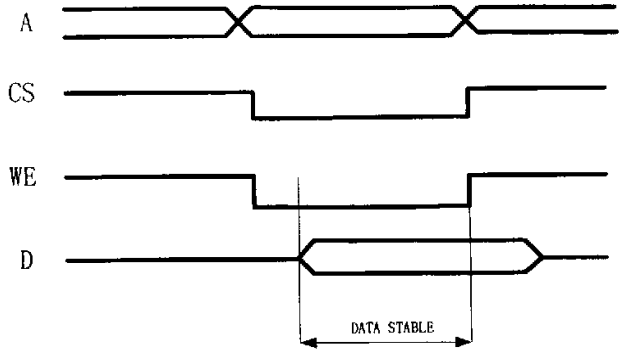


图 3.2　SRAM 写时序

在读操作中 OE 信号有效后，等待一段时间后，输出数据才有效；在写操作中，数据端数据必须维持稳定一段时间，才能写入片内。这两个等待时间必须满足。

存储控制器提供了 8 个片选信号，作为这一类存储器的片选信号。

3.2 SRAM 控制器整体设计介绍

为了提供足够的外部存储器访问能力，所设计的存储控制器提供了 8 个片选（CSA0、CSA1、CSB0、CSB1、CSC、CSD、CSE、CSF），可以用来作为 8 组外部 SRAM（FLASH、ROM）的片选，其中 CSC、CSD、CSE、CSF 也可以作为 SDRAM 的片选（具体见 SDRAM 控制器部分）。提供了 25 位地址线及 32 位数据线，这样单个片选最大可以寻址 32M 空间，实际上当前 SRAM、FLASH 等这类存储器的存储容量较小，并不会使用象 32M 这么大的空间，这样在保证实际系统的应用要求的基础上，使本设计支持的系统配置具有一定的可扩展能力。

在进行嵌入式微处理器设计时，如果需要使用这个存储控制器 IP，可根据具体系统规划对它的这些参数进行调整，而不会对它的功能、性能产生影响，这样就增强了它作为 IP 设计的可重用性。

如何将来自 AHB 总线的对外部存储设备的读写请求转换成对相应存储器的读写操作呢？显然存储控制器应能得到各个存储器的相关信息，包括存储器类型（是否是 DRAM 等）、所分配的地址空间、数据线宽度、读写延时信息等。片选信号基址寄存器和片选信号配置寄存器就是应这些要求而设置的。片选信号配置寄存器各个配置位与实际连接的外部存储器类型相关，用来配置外部存储器的类型、数据线宽度、读写时间要求、存储容量大小、存取权限以及使能位（即某个片选是否接外部存储器）。基址寄存器用来设置各个片选信号对应的地址空间的起始地址，以供地址译码模块根据这个起始地址及各个存储器大小，来产生正确的地址空间译码信号，表明当前是对哪个片选连接的存储器进行访问。

3.2.1 接口信号描述

主要包括两部分接口信号：AHB 总线信号和输出给 SRAM 芯片的控制信号。另外，整个存储控制器的总线信号锁存功能也在本模块中实现，因此锁存后的总线信号也由此模块提供；另外总线地址被锁存后，要经过译码，以判断当前是对哪个片选对应的存储空间进行访问，这部分功能也在 SRAM 控制器中实现，因此这些译码信号也由本模块输出给其他模块（包括 SDRAM 控制器、CF 控制器）。

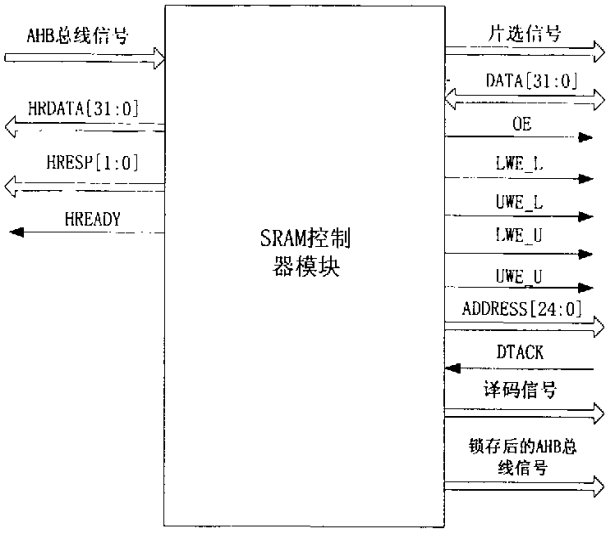


图 3.3　SRAM 控制器模块接口信号

3.2.2 片选信号配置寄存器

具体内容如下：

CSAB —— 配置片选信号 CSA0、CSA1、CSB0、CSB1 的相关信息

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	RO	-				BW		FLASH	WS3	WS2	WS1	WS0	SIZ2	SIZ1	SIZ0	EN
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	RO	SOP	-			BW		FLASH	WS3	WS2	WS1	WS0	SIZ2	SIZ1	SIZ0	EN

其中：高 16 位用来配置 CSBx、低 16 位用来配置 CSAx。

CSCD、CSEF —— 前者配置片选信号 CSC/CSD，后者配置 CSE/CSF 的相关信息

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	RO	SOP-				BW	SDRAM	FLASH	WS3	WS2	WS1	WS0	SIZ2	SIZ1	SIZ0	EN
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	RO	SOP	-			BW	SDRAM	FLASH	WS3	WS2	WS1	WS0	SIZ2	SIZ1	SIZ0	EN

这两个寄存器各位意义相同，其中：高 16 位用来配置 CSD/CSF、低 16 位用来配置 CSC/CSE。

释：

RO —— 只读标识位：

当 RO=1 时，表示 CSx 所选的存储区域全都是只读的；

当 RO=0 时，表示 CSx 所选的存储区域是可读可写的。

设置这一位使用户可以将某个片选所接的存储器设为只读属性，这样可以防止因为意外的写操作而使其中的数据丢失。

SOP —— 超级用户标识位：

当 SOP=1 时，表示 CSx 所选的存储区域是超级用户区；

当 SOP=0 时，表示 CSx 所选的存储区域是普通用户区。

设置这一位是针对总线规范中对于传输请求保护模式的定义，AHB 总线通过 HPROT [1] 这一位信号来标识当前的数据传输请求是特权模式还是普通用户模式，只有当被访问的存储器的访问权限设置与总线传输请求的权限吻合，访问才会继续进行，否则存储控制器会发出错误响应给 AHB 总线，并结束本次传输。当然 AHB 总线规范中还规定并区分了其他的保护模式，比如是指令预取还是数据传输等，但因为存储器对于所保存的数据并不区分是指令还是数据，因此在这里不提供对其他保护模式的支持，而只区分特权模式和普通用户模式。

注意：片选 CSA0 作为启动时从其中加载启动程序的存储器的片选，不能将存取权限设置为超级用户区，否则在处理器复位后，CPU 的 CORE 发出取指令请求，存储控制器会因为所要读取的地址空间是超级用户区而拒绝取指令请求，因此 CSAB 寄存器中没有提供相应配置位。

BW —— 外部数据线宽度：

当 BW＝1 时，表示此片选所连接的存储器总共为 32 位；

当 BW＝0 时，表示此片选所连接的存储器总共为 16 位；

提供这个配置位很必要。虽然存储控制器提供给 8 个片选的数据线都是 32 位的，但根据应用系统的实际需要，某个片选所接的存储器不一定构成 32 位，可能只有 16 位，因为在满足需要的情况下，这样可以节约系统 PCB 板的面积。所以提供对片选的 16 位访问模式的支持可以扩展系统的灵活性。

SDRAM——为 1 表示本片选外接 SDRAM，因为 CSC、CSD、CSE、CSF 也可以作为 SDRAM 的片选。

FLASH —— 表示 CSx 是否选的是 FLASH：

当 FLASH=1 时，表示 CSx 所接的存储器为 FLASH；

当 FLASH=0 时，表示 CSx 所选的存储器不是 FLASH。

WS3、WS2、WS1、WS0 —— 指示需要多长时间才能准备好数据。

从图 3.1 和 3.2 中知道，在对 SRAM 作读操作时，读有效信号（OE）有效后需等待一定时间，SRAM 才能准备好数据；在作写操作时，写有效信号（WE）及数据有效后也需等待一定时间，数据稳定后才能正确写入 SRAM 中。这些等待时间的长度即由这四位设置。

例如：

0	0	0	0	不需等待
0	0	0	1	等待 1 个周期
0	0	1	0	等待 2 个周期

依次类推。特别的，当这四位配置为四个 1 时，表示等待时间由芯片外部管脚送来的 DTACK 信号控制。

在这些等待时间内，存储控制器应该将总线应答信号 HREADY 置为低，以相应延迟总线读写周期，直到读写完成后再将 HREADY 置为高，继续进行下一拍数据传输。

SIZ2、SIZ1、SIZ0 —— 所选存储空间大小，例如：

0	1	0	512K(CSAx and CSBx)	1M(CSC/D and CSE/F)
0	1	1	1M (CSAx and CSBx)	2M(CSC/D and CSE/F)

地址译码模块是如何对总线地址进行译码的呢？首先用户要在片选信号基址寄存器中配置相应片选的起始地址，在片选信号配置寄存器中通过 **SIZ2、SIZ1、SIZ0** 这几位配置相应片选的地址空间大小，通过这两个信息就可以给某个片选指定特定的一块内存地址空间。这样存储控制器在接到 AHB 总线传输请求后，将总线地址与各个片选所配置的地址空间进行比较，如果总线地址落在某个地址空间内，则相应片选被选中。

CSC、CSD、CSE、CSF 这四个片选也可作为 SDRAM 的片选，因为 SDRAM 芯片存储容量较 SRAM 等比较大，所需分配的地址空间也要较 SRAM 大，所以 CSCD、CSEF 寄存器中这几位的值对应的存储空间较 CSAB 寄存器中的大。

举例说明：当各寄存器中 SIZ[2:0]=010 时，对于 CSAx 而言，CSA0 所选区域是在 CSGBAB[15:0]（见后）指定的基址后的第一个 512K 的区域，CSA1 所选区域则是紧接着 CSA0 后面的 512K 区域。CSB0 所选区域是在 CSGBAB[31:16] 指定的基址后的第一个 512K 的区域，CSB1 所选区域是紧接着 CSB0 后面的 512K 区域。而 CSC 所选区域是在 CSGBCD[15:0] 指定的基址后 1M 区域，CSD 所选区域是在 CSGBCD[31:16]指定的基址后 1M 区域，CSE、CSF 与 CSC/D 相似。

EN —— 片选信号有效位

当 EN=1 时，表示该组片选信号可用；

当 EN=0 时，表示该组片选信号无效。

3.2.3 片选信号基址寄存器

这个寄存器保存各个片选的基址（即起始地址），且只保存其中的 12 位（28~17 位）。因为在嵌入式系统中，分配给外部存储器的地址空间一般位于低地址空间，此时起始地址的 31～29 各位都为 0，表示各个片选的基址时无需用到这三位，而表示基址的地址线的低位也都是 0，因此只需用地址线的 28～17 位即可表示各个片选的基址。这样做的好处是可以节省所需的寄存器数目，这样，每个片选的基址寄存器只需 12 位即可，可以大大节约硬件开支，节省面积。

总共需要三个片选信号基址寄存器来配置 8 个片选的起始地址，分别是：**CSGBAB**、**CSGBCD**、**CSGBEF**。

CSGBAB —— CSAx 和 CSBx 片选信号所选区域基址寄存器

BIT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIELD	GBA28	GBA27	GBA26	GBA25	GBA24	GBA23	GBA22	GBA21	GBA20	GBA19	GBA18	GBA17				
BIT	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FIELD	GBA28	GBA27	GBA26	GBA25	GBA24	GBA23	GBA22	GBA21	GBA20	GBA19	GBA18	GBA17				

其中：高 16 位用来指示 CSBx 的基址，低 16 位用来指示 CSAx 的基址。

CSGBCD、**CSGBEF** 与 CSGBAB 相似，CSGBCD 的高 16 位用来指示 CSD 的基址，低 16 位用来指示 CSC 的基址；CSGBEF 的高 16 位用来指示 CSF 的基址，低 16 位用来指示 CSE 的基址。

3.3 各子模块设计

来自 AHB 总线的、对存储控制器的读写请求有两类：一类是对存储控制器中的控制寄存器的读写，另一类是对外部存储器的读写。因为在一个嵌入式系统中分配给控制寄存器和外部存储器的地址空间是不同的，所以存储控制器中的地址译码模块可以根据来自总线的读写地址来判断当前读写操作的对象。控制寄存器是一组内部寄存器，对它们的读写可以在一个时钟周期内完成，而对外部存储器的读写访问一般需要多个时钟周期才能完成。SRAM 控制器的动作流程如下：

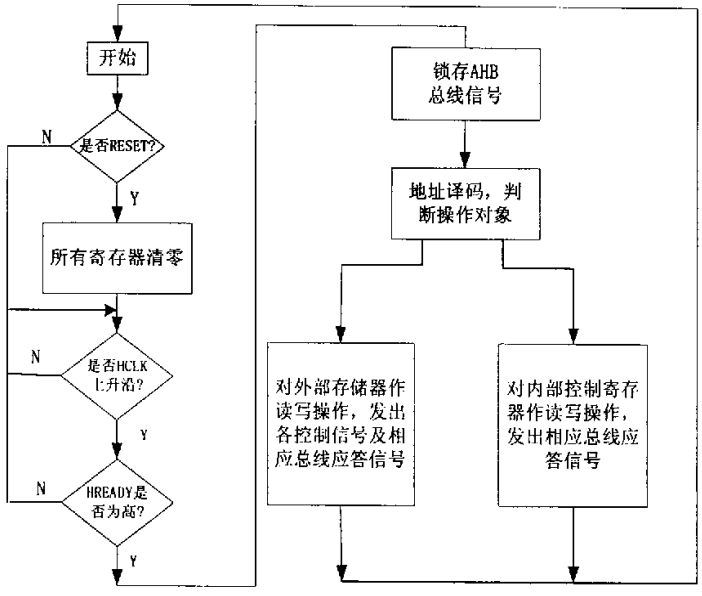


图 3.4 SRAM 控制器的动作流程图

下面介绍 SRAM 控制器模块中各个子模块的设计：

3.3.1 总线信号锁存子模块

本模块实现对 AHB 总线信号的锁存。锁存后的总线信号不但被 SRAM 控制器模块使用，同时也输出给 SDRAM 控制器模块及 CF 控制器模块使用，因此本模块为存储控制器的公共子模块。

根据 AHB 总线规范要求，AHB 从设备在 HREADY 信号为高时，在 HCLK 的上升沿锁存总线信号，并对锁存后的信号进行译码。要注意此处控制锁存动作是否发生的 HREADY 信号是由从设备（此处是存储控制器）产生的。

根据 AHB 总线规范，需要锁存的总线信号包括与从设备相关的总线地址信号、总线控制

信号，而总线数据信号不需锁存。因为 AHB 总线采用流水结构，在前一个总线周期的上升沿，AHB 总线发出地址、控制信号，在后一个总线周期的上升沿发出数据信号。从设备在这个周期锁存到总线地址信号、总线控制信号后，如果将 HREADY 置为高，表示从设备可以在一个周期内完成数据传输，并将在下一周期的上升沿完成这次数据传输，而总线也将在下一上升沿同步发出新的数据及下一拍的传输请求信号；如果从设备在锁存到总线地址信号、总线控制信号后，将 HREADY 置为低，表示从设备无法在一个周期内完成数据传输，那么总线会一直将这个数据信号保持在数据总线上，直到从设备完成数据传输后，将 HREADY 置为高，总线才会在下一时钟上升沿发出新的总线数据和地址、控制信号。因此在 HREADY 置为低的这段时间，数据一直在数据总线上，从设备无需锁存它。

3.3.2 地址译码子模块

根据 AHB 总线规范，从设备在锁存到总线信号后，根据 HSEL 信号来判断当前的数据传输请求是否是针对本设备的。如果是，那么地址译码模块就根据锁存后的总线地址信号，及内部片选配置寄存器、片选基址寄存器，进行地址译码，产生正确的片选信号，如果相应片选外接 SDRAM，则产生内部 SDRAM 片选信号给 SDRAM 控制模块，如果访问的地址空间对应 CF 卡地址空间，则产生内部 CF 片选信号，给 CF 控制模块。另外，如果当前是针对内部寄存器的读写请求，则译码模块会产生相应信号给内部寄存器读写子模块，以表明当前是对内部寄存器的读写操作。

3.3.3 内部寄存器读写子模块

本模块实现对各个片内控制寄存器的上电复位及读写操作，并将各寄存器的各位输出给其他模块。本模块为一组同步的寄存器，以 HCLK 作为所有寄存器的同步时钟，在 HCLK 上升沿完成读写操作。由来自地址译码子模块的译码信号和锁存后的总线信号 HWRITE 决定是否作读写操作，是作读操作还是写操作。

3.3.4 HREADY、HRESP 信号产生模块

本模块产生正确的 HREADY 和 HRESP 信号。

将 HREADY 信号置低可以用来延长总线等待时间，以等到外部 SRAM 准备好数据（具体等待时间由配置寄存器中相应位决定）。HREADY 的产生有以下几种情况：

1、当前是对内部寄存器进行读写：由于寄存器的读写可以在一个时钟周期内完成，因此 HREADY 不用置低，可以保持为高。

2、如果当前总线发起的读写操作有错误，则将 HREADY 置低一个周期后，再将 HREADY 置高，以配合 HRESP 完成一次错误响应。

3、如果当前总线传输状态（HTRANS）为 IDLE 或 BUSY，则也不需要将 HREADY 置低，保持为高即可。

4、正常的对外部存储器的读写：此时要将 HREADY 置低，必须等到外部存储器准备好数据，并且此数据已由 SRAM 控制器放到 AHB 数据总线（HRDATA）上以后，才能将 HREADY 置高，通知总线在下一拍锁存数据。要注意的是当外部存储器数据线为 16 位，而 AHB 总线发起一次 32 位的读写请求这种情况，此时要对存储器进行两拍 16 位的读写后，才完成一次完整的总线读写请求。因此在整个读写过程中必须一直将 HREADY 置低，直到第二拍读写完成以后，才将 HREADY 置高。

AMBA 总线规范中定义的 HRESP－即从设备的总线应答有四种：OKAY、ERROR、RETRY、SPLIT，其中后两种并不适用于存储控制器，因此在这里只需实现前两种。如果存储控制器能够正确完成当前的总线数据传输请求，则 HRESP 应答为 OKAY。当出现以下几种情况时，HRESP 应答为 ERROR：

1、总线对一块属性为只读的存储空间发起写操作，则存储控制器响应为 ERROR。

2、总线读写请求的模式为普通用户模式，而被访问的地址空间的权限被配置为超级用户，则存储控制器响应为 ERROR。

3、外部存储器的等待时间设置为 DTACK 应答模式，但是一定时间后还未接到 DTACK 应答信号，则存储控制器也将总线响应置为 ERROR。

3.3.5 DATAPATH 子模块

本模块实现 AHB 总线上的数据总线（HRDATA、HWDATA）与连接外部存储器的数据线（DATA）之间的数据转换。包括以下几种操作：

1、总线发起 BYTE、HALFWORD、WORD 读写操作时，由本模块进行正确的数据总线扩展。例如当总线发起一次宽度为 BYTE 的读操作时，从安全性考虑，存储控制器应该将所需要的数据字节扩展至整个数据总线，即将读到的数据字节扩展四次，扩展为 32 位数据后再送到 AHB 数据总线上。类似的，对于总线的半字（HALFWORD）操作请求，数据需要扩展一次，扩展为 32 位后送到 AHB 数据总线上。

2、当外部数据总线为 16 位，而内部总线发起 32 位读写操作时，此时一次读写分为两拍完成，由本模块正确实现数据总线的转换。

3.3.6 SRAMCON 模块

本模块正确产生 SRAM 的写有效信号 LWE_L、UWE_L、LWE_H、UWE_H（分别对应外部 32 位数据线上从低到高四个字节），及输出有效信号 OE。可以实现 AHB 总线对外部存储器单个字节的写操作，从而给用户提供了很大的灵活性。在对外部存储器作读操作时，不论 AHB 总线要求是要读一个字节，一个半字，还是一个字，存储控制器的作法是一次读取 32 位（或 16 位）数据，在存储控制器内部进行必要的数据扩展后送至 AHB 总线，这也是为了符合 AHB 总线规范。产生这几个信号的依据是来自总线的 HADDRESS 和 HSIZE。

3.4 本章小结

本章在介绍 SRAM 控制器的设计时，针对 AHB 总线和存储控制器的时序特点，结合 SRAM 控制器各个子模块的设计，重点在于介绍设计基于 AHB 总线规范的存储控制器的基本策略及实现方法，这些策略和方法对以下几章的内容都适用。

第四章　SDRAM 控制器模块设计

4.1 SDRAM 控制器的功能与性能优化

SDRAM 通过一个内部接口模块，使内部存储单元与外部总线信号之间隔离开，SDRAM 控制器是通过总线信号给 SDRAM 发各种命令来正确实现对 SDRAM 的操作。SDRAM 内部可以包含多个 BANK（比如 4 个）。对 SDRAM 的读写访问分两步完成：第一步先对存储器某个 BANK 中的某一行进行寻址，即作激活操作；第二步才能对这一行中的某一列进行访问。之后对这一行中的其他列的访问就很快了，只要直接访问那一列单元即可。但如果要对这一个 BANK 中的其他行进行访问就会比较慢了，必须先对这个 BANK 作一次 PRECHARGE 操作，之后才能对新的行进行访问。SDRAM 存储单元中的一行也称为“页”。在 SDRAM 内的每个 BANK 中都可以同时激活一个行，这样，如果有四个 BANK，则一个 SDRAM 片内可以同时有四个行（处于不同的 BANK 内）处于激活状态。

SDRAM 控制器的功能是将来自 AHB 总线主设备的读写请求转换成对 SDRAM 的控制信号，并且这些控制信号必须满足 SDRAM 的时序要求。

SDRAM 控制器必须支持以下命令：

1、激活命令（active）：

在对 SDRAM 作读写操作时，必须先作激活操作。这一操作用来激活 SDRAM 中某个 BANK 中的某一行，然后才能对其中的某一列进行读写。之后，对这一行的某个列地址的读写操作之前不再需要作激活操作。若要对这个 BANK 中的另一行进行读写，必须先对这个 BANK 作 PRE-CHARGE 操作后，才能重新作激活操作。激活操作与紧接着后面的读写操作之间要有一定的间隔时间，具体见后面。

2、读、写命令（read、write）：

在作完激活操作后，即可对这个 BANK 内的被激活的行进行读、写操作。若要对同一个 BANK 内另一行进行读写，则必须先对这个 BANK 进行 PRE-CHARGE 操作，再使用激活命令来激活另一行，然后才能进行读写操作。

激活某一行后，在作 BURST 读、写操作时，SDRAM 控制器只需提供起始地址，SDRAM 会在其后的连续同步时钟的每个上升沿，自动实现对后续地址的读、写操作，直到完成一次 BURST 操作，或遇到 BURST 中止命令。

一般 SDRAM 可以进行 1、2、4 或 8 拍的 BURST 操作，但考虑到 SDRAM 控制器的设计复杂性及实际需要，本 SDRAM 控制器实现 4 拍的 BURST 读操作，写操作为单拍。

3、BURST 读中止命令（TBST）：

当一次读操作实际需要的拍数小于 4 拍时（即小于一次完整 BURST 读操作的拍数），则需要发出 Burst Terminate 命令来使 SDRAM 的 BURST 读操作中止。

4、PRE-CHARGE 命令：

在激活 SDRAM 中某个 BANK 内的某一行后，若要对这个 BANK 中的另一行进行访问，则必须要先对这个 BANK 作一次 PRE-CHARGE 操作，然后才能再激活要访问的那一行，进行访问。此时前一行是处于非激活状态，即 SDRAM 中的每个 BANK 内只允许同时激活一行，但一个 SDRAM 可允许片内每个 BANK 内都有一个行处于激活状态。

5、自动刷新命令(auto-refresh)：

SDRAM 芯片最多每隔 64ms 就必须将所有存储单元刷新一遍，也称为 CBR(column before row)刷新。刷新操作可以以两种方式进行：一种是每隔 64ms 作一次刷新，每次将所有的存储单元刷新一次；另一种是在 64ms 内每隔一段时间进行一次刷新操作，每次只刷新一行或几行，在 64ms 内保证将所有存储单元刷新一次。在这里使用第二种方式。

刷新操作以行（页）为单位进行，每次 SDRAM 锁存到刷新命令后，自动刷新一行，并在内部将刷新地址自动累加一。所以 SDRAM 控制器发自动刷新命令时不需给出刷新地址。

6、自刷新命令(self-refresh)：

SDRAM 具有自刷新功能，这是一种省电模式，在锁存到这个命令后，SDRAM 进入自刷新状态，此时除了 CLKEN 信号（时钟使能信号）维持为低以外，SDRAM 的所有其他外部控制信号都无效，SDRAM 在内部自动进行自刷新操作，以保持片内数据不丢失。因此要求 SDRAM 控制器能够正确的发出自刷新命令。当要对 SDRAM 进行读写时，要对 SDRAM 发退出自刷新命令。

7、低功耗命令（PWRDN）：

在低功耗模式，SDRAM 芯片除了 CLKEN 信号维持为低，其他信号都被关闭，即无效，这一点与自刷新（SELF-REFRESH）模式相似。但在低功耗模式时，SDRAM 芯片内部不作任何操作，包括刷新操作，因此也就无法保持片内数据不丢失。所以 SDRAM 芯片不能长时间处于低功耗模式，以免片内数据丢失，由控制器保证 SDRAM 处于低功耗模式下的时间不长于刷新周期。在对 SDRAM 读、写操作之后，如果在一段时间后没有对 SDRAM 再次进行访问，控制器将会发出低功耗命令，使 SDRAM 进入低功耗模式。当要对 SDRAM 进行读写，

或需要进行 CBR 刷新时，要对 SDRAM 发退出低功耗命令（PWRDNX）。

8、空操作命令（NOP）：

当 SDRAM 处于空闲状态或等待状态时，SDRAM 控制器应该向 SDRAM 发空操作命令，这可以防止 SDRAM 在进行某种操作时锁存到错误命令，从而引起操作失败。

9、不选中命令（DSEL）：

此命令与空操作命令相似，只是片选信号线置为高，此时其他信号线无效。

充分利用 SDRAM 的特性，控制器可以大大减少对 SDRAM 的操作时的延迟[14]。可以采用以下的一些方法来提高对 SDRAM 的操作效率：

1、在对 SDRAM 中的某一行作完读写操作之后，使这一行保持尽量长时间的激活状态，这样可以节省一次 PRECHARGE 操作时间，而且在下次对这一行的访问时，可以节省一次激活操作时间，这种策略称为“开页策略”。

2、控制器将来自 AHB 总线的地址进行适当的调整后再送至 SDRAM（即 Interleaving Address），使对 SDRAM 的连续访问发生在各个 BANK 之间，从而利用 SDRAM 内部各个 BANK 可以同时激活的特性，来提高对 SDRAM 的读写效率。具体细节见 4.2.5。

3、利用 SDRAM 的接口的“流水”特性，控制器可以将发给 SDRAM 的前后两条命令重叠起来，从而减小延时，提高对 SDRAM 的读写效率。解释如下：根据 SDRAM 的特性，在对 SDRAM 发出读命令之后，经过一段延时之后数据才会出现在 SDRAM 的数据线上，这段延时表示为 Tscl。在对 SDRAM 发出一次 BURST 读命令后，SDRAM 一次进行四拍的 BURST 读，如果要继续进行 BURST 读操作，必须再发一次读命令。例如当 AHB 总线发起一次 16 拍的、宽度为 WORD 的 BURST 读操作时，SDRAM 控制器需要向 SDRAM 发出多次读命令。如果每次都是在前一次 BURST 读操作完成之后再发下一次读命令，那么每次读操作都有 Tscl 延时，这会造成效率下降。而如果在前一次读操作进行当中就发出下一次读命令，则可以将这个延时省掉。见下图。

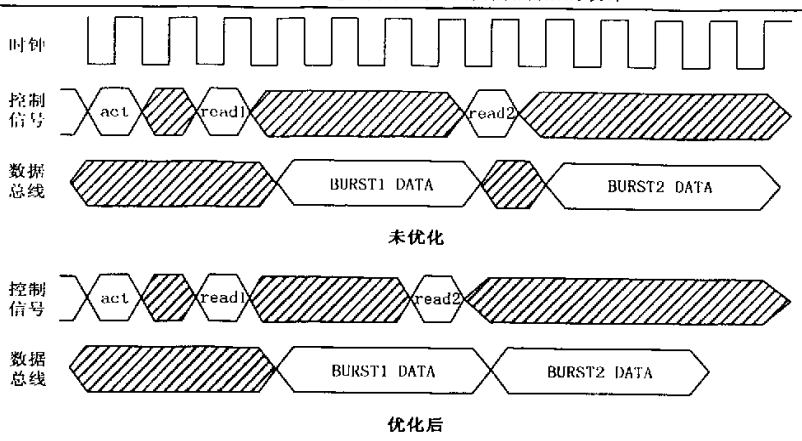


图 4.1　对 SDRAM 读指令的重叠优化

4.2 SDRAM 控制器整体设计介绍

4.2.1 整体结构介绍

此模块组成框图如图

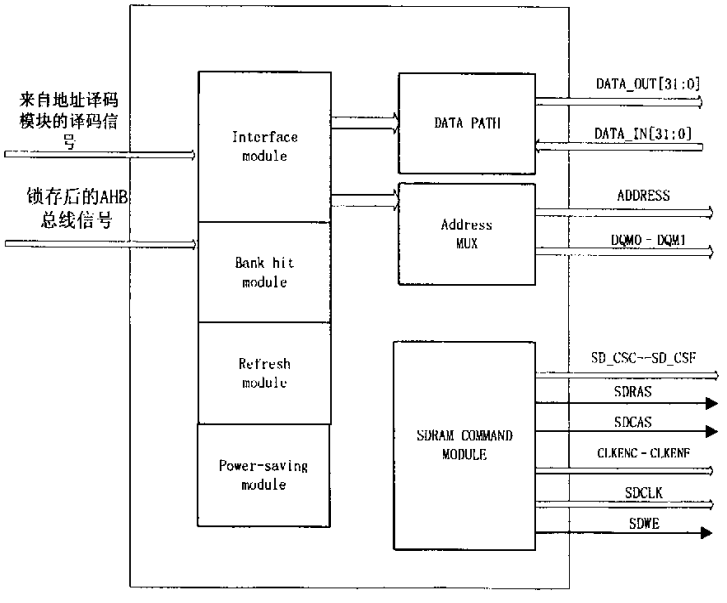


图 4.2　SDRAM CONTROLLER 组成框图

包括以下几个组成部分：

接口部分（INTERFACE-MODULE）：总线信号和来自 SDRAM 控制寄存器的信号在这里处理后，相关信息被送至其他模块使用。

SDRAM 命令产生部分（SDRAM-COMMAND MODULE）：根据总线指令及当前控制

寄存器的配置，产生正确的 SDRAM 控制信号，以实现对 SDRAM 的访问。

REFRESH MODULE：以 32kHz 左右频率的时钟作为参考时钟，根据控制寄存器的配置，产生正确的刷新请求信号，送至 SDRAM COMMAND MODULE。

BANK/ROW HIT MODULE：本模块保存已经打开的 BANK 和 ROW 地址，并将新的访问地址与它们比较，判断当前是否是对已经激活的 BANK/ROW 的访问，并发出相应指示信号。如果是，则不需进行激活操作，这样可以提高访问速度。

POWER-SAVING MODULE：对 SDRAM 最后一次读写操作之后一定时间后，如果再没有对 SDRAM 的读写，本模块将发出进入低功耗模式请求。

ADDRESS-MUX MODULE：因为 SDRAM 的行、列地址线是复用的，本模块的作用是从总线地址中截取出行、列地址及 BANK 地址，并分别送至与 SDRAM 的接口。

DATAPATH MODULE：本模块完成 AHB 总线数据与外部存储器数据之间的传输，其中使用一个 FIFO 来实现这个功能，并且这个 FIFO 的设计对于整个 SDRAM 控制器的性能都起着至关重要的作用[8]。

4.2.2 接口信号描述

SDCTLC、SDCTLD、SDCTLE、SDCTLF：(in)SDRAM 相关的控制寄存器位，来自 SRAM 控制模块。

CSC_DRAM、CSD_DRAM、CSE_DRAM、CSF_DRAM：(in)来自 SRAM 控制模块的地址译码信号，表示当前对哪个 SDRAM 片选对应的地址空间进行操作。

HTRANS、HBURST、HWRITE、HADDR 等：(in)来自 SRAM 控制模块的总线锁存信号。

self-refresh-require：(in)进入自刷新模式的请求信号，在一个嵌入式 CPU 中是来自一个功耗管理模块。

self-refresh-ack：(out)进入自刷新模式的应答信号，送给功耗管理模块。

Hready、Hresp、Hrdata：(out)输出到 AHB 总线的信号。

SD_CSC-SD_CSF：(out)四个 SDRAM 片选信号。

SDRAS：(out)SDRAM 行选择信号。

SDCAS：(out)SDRAM 列选择信号。

CLKENC-CLKENF：(out)SDRAM 时钟使能信号。

SDCLK：(out)SDRAM 时钟信号。

WE：(out)SDRAM 写有效信号。

DQM0-DQM3：(out)表示 SDRAM 在数据线上传输的数据宽度，为 1 时对应数据线上的数据无效。DQM0 对应 D[7:0]，DQM1 对应 DA[15:8]，DQM2 对应 DA[23:16]，DQM3 对应 DA[31:24]

BA0-BA1：(out)BANK 地址。

ADDRESS：(out)地址信号。

DATA-IN、**DATA-OUT**：单向数据信号线，与 SDRAM 交换数据。

PAD-DIR：(out)数据线的方向信号线。

4.2.3 控制寄存器介绍

有四个控制寄存器：SDCTLC、SDCTLD、SDCTLE、SDCTLF。

这四个寄存器分别对应四个 SDRAM 片选，所以他们各位的定义相同。寄存器中相关配置信息介绍如下：（其中最左边一列为寄存器中的各个配置位）

表 4.1　SDRAM 控制寄存器配置位信息

名称	描述	说明
SDE	SDRAM 控制器使能位	为 1 时有效
SMOD	这些位决定控制器的当前工作模式，当对 SDRAM 读写时，控制器根据这几位决定工作模式。	具体见后面介绍
SROW	这几位决定 SDRAM 的行地址位数	最多支持 13 位行地址宽度
SCOL	这几位决定 SDRAM 的列地址位数	最多支持 11 位列地址宽度
IAM	SDRAM Interleaved Address Mode 控制位	0=非 Interleaved Address Mode 1= Interleaved Address Mode
SREF	这两位决定 SDRAM 刷新率，即在每个刷新时钟沿到来时，对 SDRAM 作刷新操作的行数。对这两位的设置要参考刷新时钟频率和 SDRAM 参数	00=不刷新 01=1 行 10=2 行 11=4 行
CLKST	这两位决定是否将 SDCLK 挂起，将 SDRAM 置于低功耗模式	可以不挂起，或在没有对 SDRAM 访问时挂起，或在最后一次对 SDRAM 访问之后一定周期后挂起
DSIZ	决定外部 SDRAM 的数据线位数	支持 16、32 两种数据线宽
SCL	这两位决定 CAS 延迟，即读命令和 SDRAM 数据有效之间的间隔	以时钟周期为单位
SRP	这一位决定对某个 BANK 的 PRE-CHARGE 命令和下一个对同一个 BANK 的 active 命令之间要插入的时钟个数	以时钟周期为单位
SRCD	这一位决定 active 命令和对同一个 BANK 的读/写命令之	以时钟周期为单位

	间要插入的时钟个数	
SRC	这几位决定 REFRESH 命令与后续的命令之间需插入的时钟个数	以时钟周期为单位

4.2.4 关于工作模式

要理解 SDRAM 控制器的工作模式，先介绍 SDRAM 的初始化过程。

SDRAM 芯片在初始化时有特殊的时序要求，见下图：

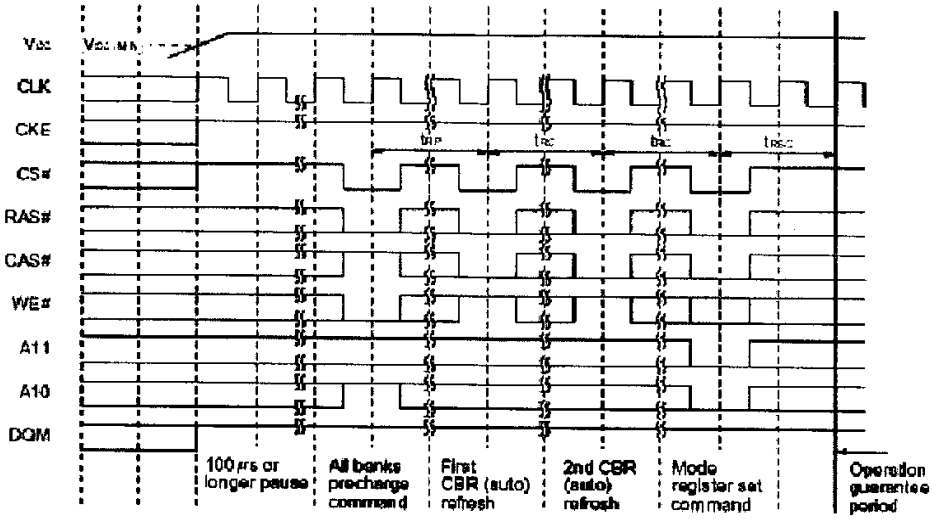


图 4.3 SDRAM 初始化示意图

即首先对所有 BANK 进行 PRECHARGE-ALL 操作，然后作 CBR 刷新，然后写 SDRAM 片内的模式寄存器（MODE REGISTER），做完所有这些操作后，才能对 SDRAM 进行正常读写操作。

根据 SDRAM 的这一要求，将 SDRAM 控制器的工作模式分为四种：PRECHARGE 模式、AUTO REFRESH 模式、SET MODE REGISTER 模式、正常读写模式，分别用来产生 SDRAM 在初始化和正常工作时所要求的各项命令。SDRAM 控制器的工作模式具体如下：

第一步：用户先将 SDRAM 控制寄存器中的 SMOD 位设置为 PRECHARGE 模式，即将 SDRAM 控制器置为 PRECHARGE 模式，然后对 SDRAM 地址空间进行一次读/写操作，这时控制器就会产生一次对 SDRAM 的 PRECHARGE-ALL 操作；

第二步：用户将 SDRAM 控制寄存器中的 SMOD 位置为 AUTO REFRESH 模式，然后对 SDRAM 地址空间进行多次读/写操作，这时控制器就会产生多次对 SDRAM 的 CBR 刷新操作，以满足 SDRAM 的初始化要求；

第三步：用户将 SDRAM 控制寄存器中的 SMOD 位置为 SET MODE REGISTER 模式，

然后对 SDRAM 地址空间进行一次写操作，这时控制器就会产生一次对 SDRAM 的 SET MODE REGISTER 操作。因为 SDRAM 在作 SET MODE REGISTER 操作时，是从地址线 A0~A10 上得到要设置的模式寄存器的值，因此在对 SDRAM 作写操作时，要求将要设置的模式寄存器的值放到相应的地址线上。

第四步：用户将 SDRAM 控制寄存器中的 SMOD 位置为正常读写模式，此后即可以实现正常的对 SDRAM 的读写操作，SDRAM 控制器也可以自动实现对 SDRAM 的自动刷新操作等。

4.2.5 关于 Interleaved Address Mode

在 SDRAM 控制寄存器中有一位：IAM。

当 IAM=0 时，为非 Interleaved address mode，此时认为来自 AHB 总线的内部 32 位地址的各位从低位到高位依次为列地址，行地址，BANK 地址。在这种模式下，当地址从基地址开始增加时，先依次访问第一个 BANK 的所有 PAGE，然后访问第二个 BANK 的所有 PAGE，依次进行下去。这种模式适合于对 LCD 显示内存的那种连续线性访问

当 IAM=1 时，为 Interleaved address mode，此时认为来自总线的内部地址从低位到高位依次为列地址，BANK 地址，行地址。在这种模式下，当地址从基地址开始增加时，先依次访问各个 BANK 的 PAGE0，然后访问各个 BANK 的 PAGE1,依次进行下去。

见下图

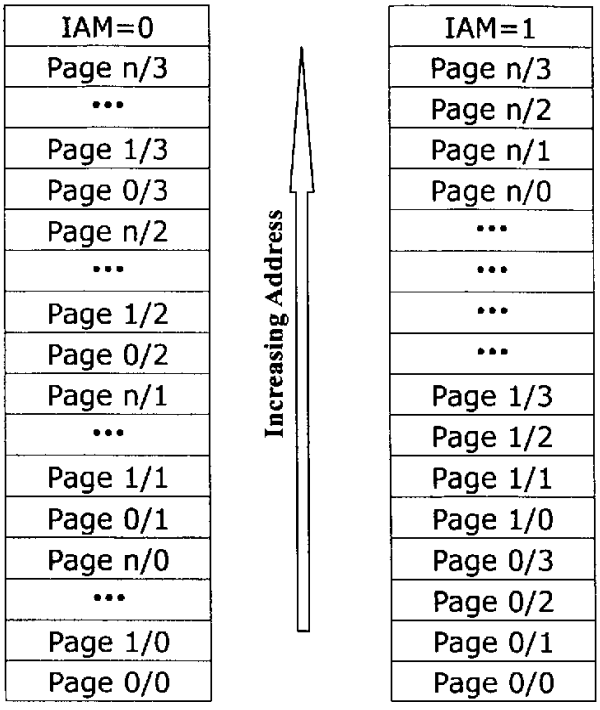


图 4.4　两种地址模式

上图中 Page N/M 表示 BANK M 中的 Page N。

4.3 各子模块设计

4.3.1 接口部分

这部分的一个重要功能是根据总线传输请求的各个控制信号，包括传输位宽，BURST 类型等，再根据相应片选外部所接的 SDRAM 的数据线宽度，来决定将要对外部 SDRAM 发起何种读写操作，需要发起几次 BURST 操作，是否需要中止 BURST 操作，以及何时中止一次 BURST 操作等。

另外，这部分还要产生总线应答信号，如 HREADY、HRESP 等。

在介绍接口部分的设计实现之前，我们先来分析一下 SDRAM 控制器在控制 SDRAM 实现 AHB 总线的传输请求时会碰到的问题。

SDRAM 芯片内有一个可以由用户配置的模式寄存器，其中有几位决定每次 SDRAM 作 BURST 读写操作的拍数，可以设置为 1 拍、2 拍、4 拍或 8 拍。

SDRAM 的特点是在锁存到读写操作命令，并开始一次读写操作之后，SDRAM 会在其后的连续的同步时钟的每个上升沿自动进行读写操作，直到锁存到 BURST 中止命令或本次 BURST 操作完成为止。以读操作为例，SDRAM 在开始读操作之后的每个同步时钟上升沿，会自动将下一个地址单元的数据读出，放到数据线上，一直进行下去，直到完成一次完整的 BURST 读操作的拍数为止。但 SDRAM 如果在中间锁存到 BURST 中止命令，就会提前结束这一过程。

在结束一次 BURST 读操作之后，如果要开始另一次新的 BURST 读操作，则 SDRAM 控制器必须再发一次 BURST 读命令。前后连续的两次 BURST 读操作可以在时序上连起来，实现的方法是在前一次 BURST 读操作结束之前，再发一次 BURST 读命令，只要发出前后两次 BURST 读命令的间隔周期数合适，就可以保证对 SDRAM 的前后两次 BURST 读操作连续进行，这样可以提高对 SDRAM 的读操作效率，进而提高 AHB 总线的传输效率。

因此，SDRAM 控制器在锁存到 AHB 总线发起的一次数据传输请求时，及早决定要对 SDRAM 进行几拍的 BURST 操作，需要向 SDRAM 发出几次 BURST 读写命令，以及在一次 BURST 操作当中是否需要插入 BURST 中止命令，对于提高读写效率有很大帮助。

在对 SDRAM 作写操作时，在同步时钟的上升沿，除了要发出写命令，还要在数据线上同时发出要写入的数据，这个数据是来自 AHB 总线的。如果将 SDRAM 设置为单拍写模式，那么不论 AHB 总线发起何种的 BURST 写操作请求，控制器都无需向 SDRAM 发 BURST 中

止命令，这样可以简化 SDRAM 控制器的设计，却不会对 SDRAM 的写速度产生影响。

另外，可以看出，AHB 总线发起一次读传输请求之后，存在两个数据传输过程：一个是 SDRAM 到 SDRAM 控制器的数据传输，另一个是 SDRAM 控制器到 AHB 总线的数据传输。如果等到前一个数据传输过程完成之后，才进行后一个传输过程，即将从 SDRAM 读取的数据都锁存进控制器之后才传输给 AHB 总线，那么将极大的降低 AHB 总线的传输效率。而如果能将这两个数据传输过程最大程度的并行起来，就可以减小 AHB 总线的等待周期，从而提高 AHB 总线的传输效率。

下面详细介绍在这个 SDRAM 控制器中是如何实现上面提到的几点的。

在这个 SDRAM 控制器中，默认外部 SDRAM 为 4 拍的 BURST 读模式和单拍的写模式，这一点很容易做到，只要在每次 SDRAM 芯片的初始化过程中写模式寄存器时，将 SDRAM 配置为 4 拍的 BURST 读模式和单拍的写模式。在这种模式下，锁存到一次读命令后，SDRAM 默认为是一次 4 拍的 BURST 读操作。若要提前结束本次 BURST 读操作，则 SDRAM 控制器必须发 Burst Terminate 命令(TBST)，SDRAM 才会结束本次 BURST 读操作。

在设计时，使用了 4 个变量，来表示当前的传输类型。这四个变量由本模块根据总线传输控制信号来决定。

Trans-Length：表示 SDRAM 控制器实际需要的对 SDRAM 的读拍数；

Burst-Length：表示第一个读命令发出后间隔几个周期向 SDRAM 发 TBST 命令。若最后一次读命令对于完整的 4 拍读，则此值为 0，表示不需要向 SDRAM 发 TBST 命令。

Burst-Times：表示针对 AHB 总线的一次 BURST 传输请求，SDRAM 控制器需要向 SDRAM 发几次读命令。

Trans-Type：表示当前总线传输请求的位宽与外部 SDRAM 数据位宽的对应关系。

我们先以来自 AHB 总线的 4 拍的 BURST 传输为例，来说明本模块的工作原理。

1、AHB 总线发起 4 拍的，位宽为 BYTE 的 BURST 传输：

此时总线总共要读 4 个 BYTE 的数据。

当 SDRAM 数据线位宽为 32 位时，对 SDRAM 一个读周期可以读到一个 WORD(4 个 BYTE)，而发一个读命令可以连续有 4 个读周期，因此只要发一次读命令（READ）就可以。考虑到 AHB 总线传输请求的起始地址最低两位有可能不是 00，而对 SDRAM 的读写是以 WORD 为单位的，因此此时最多可能需要两个读周期才能读到满足总线 BURST 读请求的数据量。因此，在这种情况下，**Trans-Length** 置为 2，**Burst-Length** 置为 2，**Burst-Times** 为 1，**Trans-Type** 为 BYTE_32。

当 SDRAM 数据线位宽为 16 位时，对 SDRAM 一个读周期可以读到一个 HALF-WORD(2 个 BYTE)，只要发一次读命令（READ）就可以。考虑到 AHB 总线传输请求的起始地址最低一位有可能不是 0，而对 SDRAM 的读是以 HALF-WORD 为单位的，因此此时最多可能需要对 SDRAM 进行三个读周期才能完成总线 BURST 读请求的数据量。因此，在这种情况下，**Trans-Length** 置为 3，**Burst-Length** 置为 3，**Burst-Times** 为 1，**Trans-Type** 为 BYTE_16。

2、AHB 总线发起 4 拍的，位宽为 HALF-WORD 的 BURST 传输：

此时总线总共要读 8 个 BYTE 的数据。

当 SDRAM 数据线位宽为 32 位时，考虑到 AHB 总线传输请求的起始地址最低位一定为 0，但次低位不一定为 0，因此此时最多可能需要三个读周期才能完成总线 BURST 读请求的数据量。因此，在这种情况下，**Trans-Length** 置为 3，**Burst-Length** 置为 3，**Burst-Times** 为 1, **Trans-Type** 为 HALFWORD_32。

当 SDRAM 数据线位宽为 16 位时，此时需要四个读周期才能完成总线 BURST 读请求，这正好构成一次完整的读命令，因此，在这种情况下，**Trans-Length** 置为 4，**Burst-Length** 置为 0，**Burst-Times** 为 1, **Trans-Type** 为 HALFWORD_16。

3、AHB 总线发起 4 拍的，位宽为 WORD 的 BURST 传输：

当 SDRAM 数据线位宽为 32 位时，此时需要四个读周期才能完成总线 BURST 读请求的数据量，也正好构成一次完整的读命令，因此，在这种情况下，**Trans-Length** 置为 4，**Burst-Length** 置为 0，**Burst-Times** 为 1, **Trans-Type** 为 WORD_32。

当 SDRAM 数据线位宽为 16 位时，需要八个读周期才能完成总线 BURST 读请求，这时需要向 SDRAM 发两次读命令，因此，在这种情况下，**Trans-Length** 置为 8，**Burst-Length** 置为 0，**Burst-Times** 为 2, **Trans-Type** 为 WORD_16。

对于其他几种 AHB 总线传输请求类型，都可参照以上原理处理。

4.3.2 BANK-HIT 模块

此模块的功能是在每次总线发起数据传输请求时，判断当前总线地址对应的 SDRAM 中的 BANK/行（ROW）是否已经激活（ACTIVE）。每个片选允许同时最多激活 4 个 ROW，并且这四个 ROW 必须位于不同的 BANK。因此针对每个片选设置了四个一位的标志寄存器（用来表示每个 BANK 是否已激活）和四个多位的行地址寄存器（保存已经被激活的行地址）。

若总线要读写的相应 BANK 已被激活，则将总线地址中的行(ROW)地址与模块内部保存的相应片选的相应 BANK 中的行地址比较，若二者相同，说明要读写的行已经激活，则将相

应标志（HIT）置为 1；若比较结果不同，说明要读写的行未激活，则将 HIT 置为 0。若相应 BANK 还未激活，则直接将 HIT 置为 0。.

4.3.3 REFRESH 模块

本模块的功能是正确发出刷新请求，并决定刷新操作的次数。

在三种情况下，SDRAM 要求作刷新操作（此处是指 CBR－AUTO REFRESH）：

第一种情况是在上电初始化 SDRAM 时，要求对 SDRAM 发固定次数的刷新命令。

第二种情况是在 SDRAM 进入自刷新（SELF-REFRESH）和退出自刷新时，要求将所有行刷新一次或者只刷新一行（不同的 SDRAM 信号有不同的要求，由 SDRAM 控制寄存器配置）。

第三种情况是为了保持 SDRAM 片内数据不丢失，SDRAM 要求必须在 64 毫秒内将所有存储单元以“行”为单位刷新一次。

作刷新操作时，SDRAM 控制器无需给出要刷新的行地址，只需发刷新命令。

前两种刷新操作不是由本模块发起请求的，其刷新次数是固定的。

对于第三种情况，要求在 64 毫秒内发刷新命令的次数至少等于行数。为实现这一功能，在设计中将总线时钟 HCLK 进行分频，以得到接近 32K 频率的时钟（频率应大于 32K），以此时钟作为刷新时钟[8]，每次在这个时钟上升沿到来时，发起一次刷新请求，这样，在 64 毫秒内，至少可以发起 2048 次刷新请求。如果根据所连接的外部 SDRAM 芯片的行数，每次有刷新请求时作 1 行、2 行或 4 行刷新操作（这可以由控制寄存器配置决定），则 64 毫秒内可以对 2048、4096 或 8196 行进行刷新，这样就满足了 SDRAM 关于刷新操作的要求。

4.3.4 POWER SAVING 模块

SDRAM 芯片有一种低功耗模式，在这种模式下，SDRAM 处于省电状态。用户有三种选择：1、可以要求不使 SDRAM 进入低功耗模式 2、当没有对 SDRAM 进行读写操作时就进入低功耗模式 3、对 SDRAM 最后一次读写操作之后一定周期后使 SDRAM 进入低功耗模式。用户可以配置 POWER SAVING 模块，对 SDRAM 按以上任何一种模式来处理。为此，本模块包括了四个计数器，当采用第三种方式，而且当前没有对 SDRAM 进行读写操作时，会启动计数器计数，计数到设置值后，发出进入低功耗请求给状态机模块，使其发出进入低功耗命令。

4.3.5 DATA-PATH 模块

本模块负责 SDRAM 控制器与外部 SDRAM 之间的数据交换，具体说，就是在对 SDRAM 写操作时将来自 AHB 总线的数据放到外部数据线上，在对 SDRAM 读操作时，将来自 SDRAM 的数据正确送到 AHB 总线上。

在写操作时较简单，因为对 SDRAM 进行的是单拍的写操作，在一个写周期内，来自 AHB 总线的数据一直在总线上，无需缓存，可直接送到外部数据线上，只需进行必要的数据调整。在读操作时较复杂，因为对 SDRAM 发出读命令后，数据在每个同步时钟的上升沿被连续读出（即是 BURST 读），而从 SDRAM 读出的数据宽度不一定与 AHB 总线传输请求的宽度一致，因此需要缓存并调整以后，然后按照 AHB 总线的要求送到总线上。

为实现这个缓存功能，在这个模块内，设计了一个深度为 32、宽度为 32bits 的同步 FIFO（先入先出缓存），来实现对 SDRAM 的读操作。这个 FIFO 采用 AHB 总线时钟 HCLK 作为同步时钟。宽度采用 32 位是为了与 AHB 总线的宽度匹配，深度采用 32 级的原因如下：来自 AHB 总线的 BURST 类型的传输请求最大可能是 16 拍的“字”传输，而外部 SDRAM 的数据线宽度有可能是 16 位，这种情况下，一次完整的 BURST 读操作最多需要有 32 个来自 SDRAM 的数据被缓存，因此 FIFO 的深度设计为 32 级。来自外部 SDRAM 的数据先进入这个 FIFO，再按照 AHB 总线的要求送到 AHB 总线上。在 FIFO 的每个单元（32bits）内存入每个 AHB 总线读周期内要送到 AHB 总线的数据（**Trans-Type** 为 WORD_16 时例外，见下面 5），因此在将外部数据存入 FIFO 时要根据 AHB 总线读请求的 HSIZE 和外部 SDRAM 的数据线的宽度，进行必要的转换。转换方式如下：

1、若外部 SDRAM 数据线为 32 位（或 16 位），AHB 总线传输请求宽度为 BYTE，则将外部数据线上低字节的数据扩展为 32 位后存入 FIFO 低地址单元，外部数据线上高字节的数据扩展为 32 位后存入 FIFO 高地址单元。

2、若外部 SDRAM 数据线为 32 位，AHB 总线传输请求宽度为 HALF-WORD，则将外部数据线上低半字的数据扩展为 32 位后存入 FIFO 低地址单元，外部数据线上高半字的数据扩展为 32 位后存入 FIFO 高地址单元。

3、若外部 SDRAM 数据线为 32 位，AHB 总线传输请求宽度为 WORD，此时不需进行数据位扩展，直接将外部数据线上的数据存入 FIFO。

4、若外部 SDRAM 数据线为 16 位，AHB 总线传输请求宽度为 HALF-WORD，则将外部数据线上的有效数据扩展为 32 位后存入 FIFO。

5、若外部 SDRAM 数据线为 16 位，AHB 总线传输请求宽度为 WORD，此时仍将每次

从外部数据线上得到的有效数据扩展为 32 位后存入 FIFO。只是在送到 AHB 数据总线之前要将 FIFO 内两个连续单元所存的数据进行合并，合并成一个 32 位的有效数据，再送到 AHB 总线上。

这里的 FIFO 有两个指针：一个读指针（ip_read），一个写指针（ip_write）。当外部数据线的数据存入 FIFO 时会相应修改写指针，当 FIFO 内的数据送到 AHB 总线时会相应修改读指针。读写指针的修改与当前的传输类型（用 trans-type 表示）有关，具体如下：

当 trans-type 为 BYTE_32 时，一次向 FIFO 内 4 个单元写入数据，因此每次存入数据后 FIFO 写指针加 4。

当 trans-type 为 BYTE_16 或 HALFWORD_32 时，一次向 FIFO 内 2 个单元写入数据，因此每次存入数据后 FIFO 写指针加 2。

当 trans-type 为 HALFWORD_16、WORD_16 或 WORD_32 时，一次向 FIFO 内 1 个单元写入数据，因此每次存入数据后 FIFO 写指针加 1。

读指针的维护较简单：当 trans-type 为 WORD_16 时，每次从 FIFO 内取出两个单元的数据，所以读指针每次加 2；对于其他几种传输类型，每次从 FIFO 内取出一个单元的数据，所以读指针加 1。

在每次开始一次新的 BURST 读操作时，写指针都要重新置为 0。由于 AHB 总线 BURST 读操作的起始地址不一定是 0，而对 SDRAM 只能发起半字或字操作，这样可能会读到多余数据，因此在每次开始一次新的 BURST 读操作时，需要将读指针初始化到正确初始值。这个动作取决于 trans-type 和总线 BURST 读操作的起始地址，具体代码如下：

```
case (byte-type)
  `BYTE_32:
    case (latch_address[1:0])
      2'b00:
        ip_read <= 5'b0;
      2'b01:
        ip_read <= 5'b01;
      2'b10:
        ip_read <= 5'b10;
      2'b11:
        ip_read <= 5'b11;
```

```
        endcase
    `BYTE_16:
      if (latch_address[0] )
        ip_read <= 5'b01;
      else
        ip_read <= 5'b0;
    `HALFWORD_32:
      if (latch_address[1] )
        ip_read <= 5'b01;
      else
        ip_read <= 5'b0;
    default:
        ip_read <= 5'b0;
  endcase
```

关于 FIFO 的空、满判断机制：

由于 FIFO 的深度足够大（32 级），即使 AHB 总线发起一次 16 拍（最大）的，宽度为 32 位的 BURST 读请求，在一次完整的 BURST 传输过程中，来自 SDRAM 的数据也最多只能刚好充满 FIFO，而不会覆盖已经写入 FIFO 但还未被读出的数据，也即不会发生所谓的“满”。但在从 FIFO 中读数据时有可能会发生所谓的“空”，且只会发生在 trans_type 为 WORD_16 时的情况下，因为这种情况下每个时钟周期只能从 SDRAM 读到 16 位有效数据，而 AHB 总线却需要一次读取 32 位数据，也即向 FIFO 作两次写入操作，才能作一次读出操作，所以写指针地址必须大于读指针地址两个位置，否则发出“空”信号。

4.3.6 状态机模块

本模块通过一个状态机来发出 SDRAM 命令，实现对 SDRAM 的所有操作，状态转换图如下图所示。

系统复位后，状态机处于 DSEL 状态，在这个状态时，不产生任何操作命令。当要产生对 SDRAM 的任何操作时，都会使状态机离开 DSEL 状态，发生翻转，以产生各种命令及必要的延时。在完成所有的操作后，状态机必定又回到 DSEL 状态。

在所有的状态中，READ、WRITE、ACT、TBST、PRE、PALL、CBR、SLFRSH、SLFRSHX、

PWRDN、PWRDNX、MRS 这几个状态用来产生各种对 SDRAM 的操作命令，WAITBURST、WAITCAS、WAITTBST、WAITPRE、WAITPALL、WAITCBR、WAITSLFRSH、WAITSLFRSHX、WAITMRS 这几个状态用来产生各种命令之后与下一条命令之前必要的间隔。

对状态转换图的说明：

1、在 DSEL 状态下，接到对 SDRAM 的读写请求，如果 SDRAM 处于低功耗状态或自刷新(SELF-REFRESH)状态，则状态机先转换至 PWRDNX 或 SLFRSHX 状态，发命令使 SDRAM 退出低功耗或自刷新状态，然后才进行正常的读写操作。

2、SDRAM 在出了自刷新（SELF-REFRESH）状态后，要紧接着作 CBR 刷新操作，因此有 SLFRSHX 状态→CBR 状态的转换。

3、当需要进行周期性的 CBR 刷新操作时，如果有某个 BANK/ROW 已被激活，则必须先作 PRECHARGE-ALL 操作，因此有 DSEL→PALL→WAITPALL→CBR 的转换。

4、SDRAM 在进入自刷新（SELF-REFRESH）状态之前，必须先作 CBR 操作。因此有 DSEL→CBR→WAITCBR→SLFRSH 的转换。

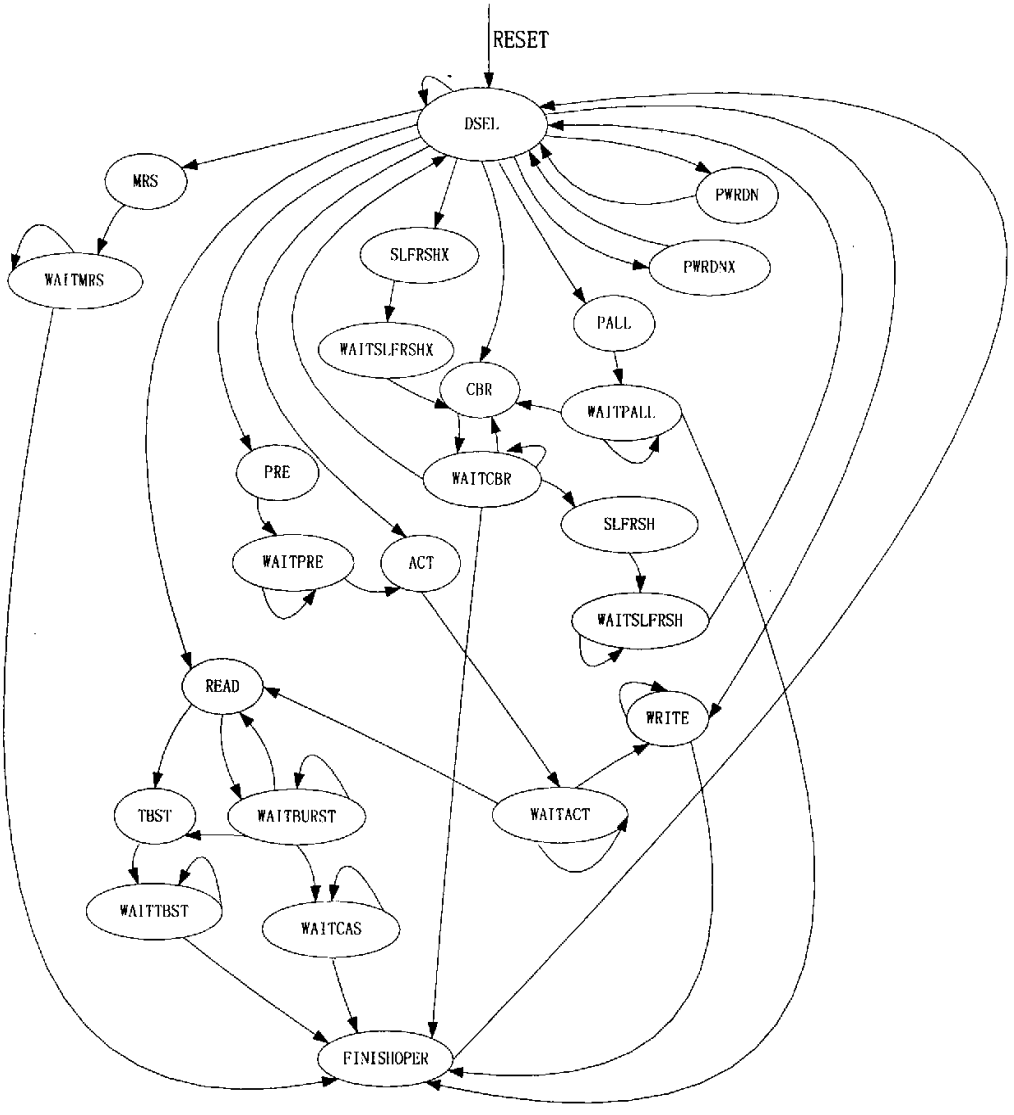


图 4.5　SDRAM 控制器状态机状态转换图

4.4 本章小结

SDRAM 内部的存储体可以分为多个 BANK，对每个 BANK 内的某个存储单元的访问分两步进行：先对行地址访问，即所谓的激活（active）操作，然后才能对列地址作读写操作，这前后两次访问之间存在延时。因此在对存储器进行单拍的访问时，SDRAM 的读写速度不如普通的 RAM 快。SDRAM 的优越性体现在对存储器的连续访问时，在激活某一行之后，就可以对 SDRAM 进行与同步时钟同步的、连续的、单时钟周期的读写访问，这样可以大大提高读写速率，而且这种对存储器的连续访问模式正好与 AHB 总线规范中的 BURST 操作吻合。但同时也给存储控制器的设计带来了困难：一方面是如何保证对 SDRAM 的读写操作时序与 AHB 总线时序相协调，另一方面是如何利用 SDRAM 的时序特性，尽量提高数据的访问速率，从而减少存储控制器占用 AHB 总线的时间，提高总线利用率。这些都是 SDRAM 控制器设计过程中要重点考虑的问题。

本章首先介绍了 SDRAM 控制器应该实现的功能，然后根据控制器要实现的功能，并针对以上提到的几点，对 SDRAM 控制器的设计进行整体模块划分，并介绍了相关内部控制寄存器的设置。接下来通过详细介绍各个功能子模块的设计，给出了整个 SDRAM 控制器的设计实现策略，这部分是全文的重点之一。在这一部分，作者详细分析了 SDRAM 的各种控制时序，及与 AHB 总线规范的关系，对于如何提高 AHB 总线中对 SDRAM 的读写操作的效率提出了自己的实现方法。

第五章 CF 控制器模块设计

5.1 CF+ and CompactFlash Specification Revision 1.4

5.1.1 CF 卡简介

CF 卡给数字设备提供了一种方便的存储各种数据、音频、图像信息，以及在不同的数字设备之间传输这些信息的手段，它具有重量轻、功耗低，可移动等优点。成立于 1995 年 10 月的 CompactFlash 协会正是以推广 CF 卡的使用，及制定 CF 卡相关规范，并保证兼容性为宗旨，到现在，这个协会已制定出 CF+规范，来扩展 CF 的概念，使其不但支持 FLASH 数据存储，而且支持 I/O 设备及磁盘数据存储。CF+卡提供大容量的数据存储和 I/O 功能，并与 PCMCIA 标准兼容（二者有少许差异，CF+卡可通过转接口连接至 PC 卡插槽）。也就是说，在这里所说的 CF 卡实际上是一个广义的概念，是指所有符合 CF 卡规范的外部设备，包括存储设备，I/O 设备等。

在 CF 卡中有一个内置的控制器，用于接口协议的实现、数据传输控制、错误校验、电源管理、时钟控制等。CF 卡控制器实际上就是与此内置控制器通信。

5.1.2 CF 卡访问模式

满足 CF 卡规范的设备要求可以工作在三种模式下：MEMORY 模式、I/O 模式、True IDE 模式，CF+卡只要求工作在前两种模式，最后一种工作模式可选，即不是必须要支持的模式。鉴于这一情况，本 CF 卡控制器只支持前两种访问模式。其中，MEMORY 模式又包括两种：Common Memory 模式和 Attribute Memory 模式。

Attribute Memory 是 CF 卡内部的一块空间，在其中保存有关于 CF 卡的配置信息，CF 卡的配置寄存器即位于这段空间中，对 CF 卡的配置及读取配置信息都是通过对这块空间进行读写来完成的。配置寄存器是一组地址固定的、8 位的寄存器，且每个配置寄存器对应的地址都是偶地址，因此对配置寄存器进行访问，只能按照 Attribute Memory 模式的时序，通过对相应偶地址进行 8 位的读写访问来完成。

对 CF 卡设备进行数据交换时，是通过对 CF 卡设备内相应功能寄存器的操作来完成的，根据 CF 卡设备的类型，对这些功能寄存器的访问可采用两种不同的模式，即 Common Memory 模式和 I/O 模式。

5.1.3 CF 控制器与 CF 卡的接口信号

MA[10:0]：CF 卡地址线。CF 卡只有 11 位地址线。

nPWE：在 MEMORY 访问模式，此位有效时是对 CF 卡的写操作；在 I/O 模式，此位有效时是对 CF 卡的配置寄存器的写操作。低有效。

nPOE：在 MEMORY 访问模式，此位有效时是对 CF 卡的读操作；在 I/O 模式，此位有效时是对 CF 卡的配置寄存器的读操作。低有效。

nPREG：在 MEMORY 访问模式，为低时表示对配置寄存器读写，为高时表示对 COMMON MEMORY 读写；在 I/O 访问模式，此位必须为低。

nPIOR：在 I/O 访问模式，此位为低时表示对 CF 卡读操作。在 MEMORY 模式此位无效。

nPIOW：在 I/O 访问模式，此位为低时表示对 CF 卡写操作。在 MEMORY 模式此位无效。

nPCE[1:0]：既作为 CF 卡的选择信号，同时与 AD[0]配合表示当前的传输是 word,还是 byte，是在 DA[7:0]上还是在 DA[15:8]上。

以上信号由 CF 控制器发出。

nPWAIT：由 CF 卡发出的信号，用来在访问过程中插入一些延时。

nIOIS16：由 CF 卡发出的信号，在 I/O 模式，此信号表示传输数据是 16 位的；在 MEMORY 模式，CF 卡作完复位初始化后，将此位置为低。

DO[15:0]：CF 控制器输出数据线，有 16 位。

DI [15:0]：CF 控制器输入数据线，有 16 位。

PADDIR：表示当前数据传输的方向，也即外部数据线的方向。

另有一些 CF 卡信号，可通过 CPU 上的通用 I/O 口来与其通讯，而不必在 CF 控制器中设置专用管脚，这样可以减少管脚数量，从而降低芯片成本。这些信号如下：

nPCD1~nPCD0：由 CF 卡发出，在卡内部与地连接，这两位用来检测卡是否已完全插入插槽中，若是，则这两位都为低。

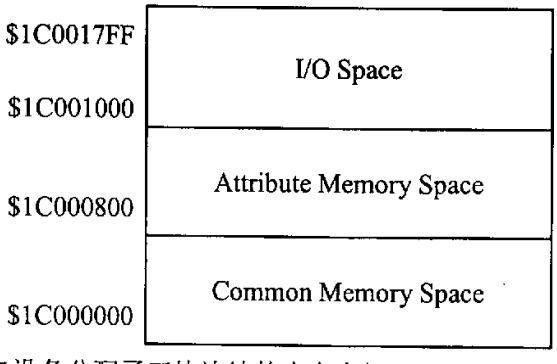
PRDY/~BSY：由 CF 卡发出。在 MEMORY 模式，当 CF 卡准备好进行新的数据传输时，将此位置高；当 CF 卡处于 BUSY 状态时，将此位置低。在 CF 卡处于上电和复位状态时，此位被置为低，表示此时不能对 CF 卡进行读写操作，直到此位被置为高以后。当 CF 卡被配置为 I/O 模式以后，这一位用来表示中断请求，可产生电平中断或沿中断。

5.2 CF 控制器整体设计介绍

CF 控制器的功能是当 AHB 总线上发出对 CF 卡的读写访问时，在外部管脚上产生满足 CF+ and CompactFlash Specification Revision 1.4 规范的时序信号，以正确完成对 CF 卡的读写。

因为要实现对 CF 卡的三种访问模式：Attribute Memory 模式、Common Memory 模式、I/O 模式，每种模式下访问时序各异，而具体每一次针对 CF 卡设备的访问采用何种模式，取决于外部 CF 卡设备的类型和当前要进行的操作类型。比如，如果要对 CF 卡设备进行配置或读取配置信息，则 CF 控制器需按照 Attribute Memory 模式产生控制信号；如果要对 CF 存储卡进行数据读写，则需按照 Common Memory 模式工作；而如果要对 I/O 类型的 CF 卡设备进行读写，则需按照 I/O 模式工作。而对于 AHB 总线来说，并没有办法区分当前是何种模式的读写操作，这就需要设计一种映射方式，使 CF 控制器能够辨别当前来自 AHB 总线的针对 CF 卡设备的数据传输请求是何种模式。

我们知道，对 CF 卡设备的操作是通过对其内部的各个特定地址的寄存器的读写操作来完成的，这些地址的偏移量在 2K 范围之内。如果在系统中专为 CF 卡设备开辟三块内存空间，CF 控制器在接到来自 AHB 总线的对这三块地址空间的读写访问请求时，会分别产生对 CF 卡的三种访问模式，这样就可以解决 AHB 总线的传输请求到 CF 卡设备三种访问模式的映射[19]。下面为一个例子。

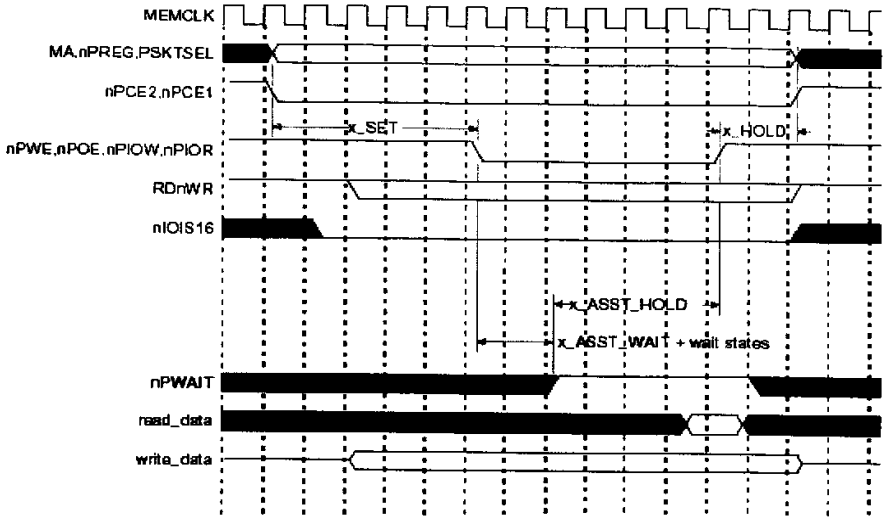


在这个例子中，为 CF 卡设备分配了三块连续的内存空间：1C000000~1C0007FF 分配给 Common Memory 空间，1C000800~1C000FFF 分配给 Attribute Memory 空间，1C001000 ~1C0017FF 分配给 I/O 空间。如果要对 CF 卡中位于 Attribute Memory 中的寄存器进行配置时，只需发起对系统内存空间中 Attribute Memory 地址空间的读写，读写时的地址其高位地址由所分配的地址段决定，低位地址由所要配置的寄存器地址决定。

由于在不同的系统中分配给 CF 卡设备的地址空间可能不同，因此在本设计中每一块内

存空间的起始地址用一个内部寄存器来实现可配置，为方便叙述，假定这些寄存器的命名分别为 COMMEMBASEADDR、ATTMEMBASEADDR、IOBASEADDR。

为了使控制器可以适用于不同频率的 AHB 总线及不同类型的 CF 卡设备，必须能够根据总线时钟频率来配置访问时序。对 CF 卡设备的三种模式的读写时序可以统一用下图来表示



[19]：

图 5.1　Memory 或 I/O 模式 16 位访问

从上图可以看出，在对不同的 CF 卡设备进行读写操作时，有四个时间参数必须针对特定 CF 卡设备进行配置，才能保证正确实现 CF 卡控制器对 CF 卡设备的操作。这四个时间参数分别是：1、从地址、片选信号置为有效到读、写信号置为有效之间的间隔时间。2、从读、写信号置为有效到检测 nPWAIT 信号之间的间隔时间。3、从检测到 nPWAIT 信号无效（为高）到读、写信号置为无效之间的间隔时间。4、从读、写信号置为无效到地址、片选信号置为无效之间的间隔时间。

在设计中采用了三个时序配置寄存器 CFCOMMEM、CFATTMEM、CFIO，来实现这些配置工作。具体介绍在后面。

另外用了一个配置寄存器：CFCONF，实际只有一位：CIT，用来表示 CF 卡插槽中是否已插入 CF 卡。由软件来检测卡是否已正确插入插槽中，若是，则由软件将这一位置为 1，否则为 0。

表 5.1　CF 控制器内部寄存器列表

名 字	描 述
CFCOMMEM	配置 COMMON MEMORY 模式下的读写时序
CFATTMEM	配置 ATTRIBUTE MEMORY 模式下的读写时序
CFIO	配置 I/O 模式下的读写时序
CFCONF	只有一位，说明卡是否已插入
COMMEMBASEADDR	COMMON MEM 空间基址寄存器
ATTMEMBASEADDR	ATTRIBUTE MEM 空间基址寄存器
IOBASEADDR	IO 空间基址寄存器

各寄存器介绍如下。

5.2.1 CFCOMMEM、CFATTMEM、CFIO 寄存器

这三个寄存器用来配置对 CF 卡的读写时序，以满足不同总线频率下对 CF 卡的正确读写。这几个寄存器的各位定义彼此相同。下面以用来配置 Common Memory 模式下的读写时序的 CFCOMMEM 寄存器为例来介绍。这个寄存器应该包括以下一些信息：

MEM_SET：在命令有效之前，地址需要提前发出的时钟个数。

MEM_HOLD：在命令撤销后，地址仍旧维持的时钟个数。

MEM_ASST：命令维持的时间。通过编码对应的方式，使这个参数的任何一个配置值都对应着一个具体的 ASST_WAIT 和 ASST_HOLD 时间，以实现命令维持时间的可配置。

5.2.2 CFCONF 寄存器

这个寄存器实际只有一位，用这一位来表示卡是否已插入插槽中。1 表示卡已插入，0 表示卡没有插入。用户通过读 CF 卡的两个检测位 nCD1，nCD2，来判断 CF 卡是否已插入插槽中，然后正确写入这一位。CF 卡在内部将这两个管脚置低，所以，如果卡已插入插槽，则这两位就会检测到为低。

5.2.3 各种访问模式的实现

CF 卡控制器支持两种访问模式：MEMORY 模式和 I/O 模式。其中 MEMORY 模式又包括两种模式：Attribute Memory 模式和 Common Memory 模式。这三种访问模式分别有自己的时序要求，因此要求 CF 控制器要实现这几种时序。具体见下面的介绍。下面的时序图中 MA 为地址信号，DA 为双向数据线（在设计中实际是两组单向数据线，由方向控制信号控制），nPCE 为 CF 卡选择信号，nPOE、nPWE、nPIOR、nPIOW 为读/写控制信号，nPWAIT、nIOIS16、

nPREG 为专用信号。

1、Attribute Memory 模式写操作

在这种模式下，~REG 和~WE 信号都应为低（有效），在这种模式下，nPWAIT 信号无效，即 CF 卡设备不能通过将这个信号置低来延长读写时间，实现的写时序如下图：

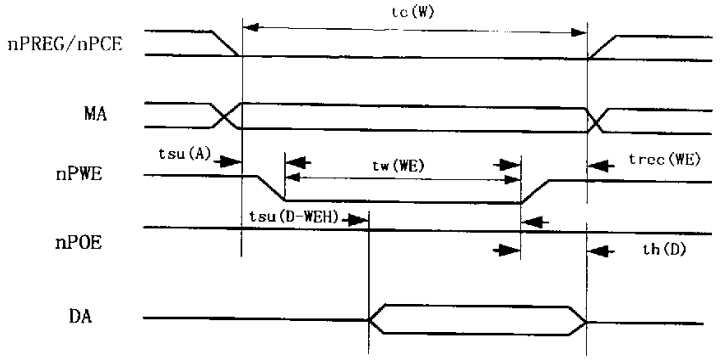


图 5.2　Attribute Memory 模式写操作时序

上图中各时间段定义见下表

项目	符号
写周期时间	tc(W)
nPWE 脉冲宽度	tw(WE)
地址建立时间	tsu(A)
写恢复时间	trec(WE)
相对 nPWE 的数据建立时间	tsu(D-WEH)
数据保持时间	th(D)

下表为 Attribute Memory 模式的写命令，这种情况下，只能对偶字节进行 8 位写操作，表中 X 表示数据无效。

nPCE2	nPCE1	MA[0]	nPOE	nPWE	DA[15:8]	DA[7:0]
1	0	0	1	0	X	偶字节

2、Attribute Memory 模式读操作

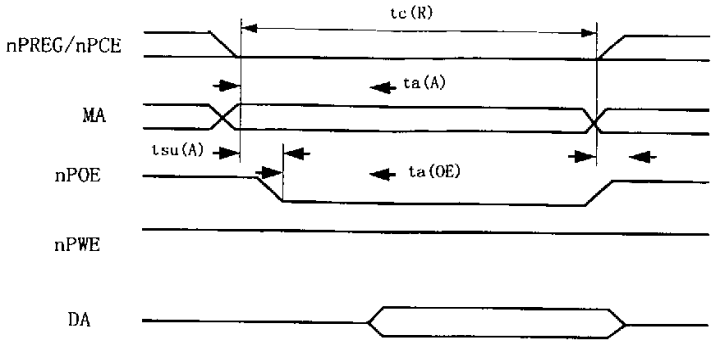


图 5.3 Attribute Memory 模式读操作时序

上图中各时间段定义见下表

项目	符号
读周期时间	tc(R)
地址到数据有效时间	ta(A)
nPCE 到数据有效时间	ta(CE)
地址建立时间	tsu(A)
地址到数据无效时间	tv(A)

下表为 Attribute Memory 模式的读命令。虽然 CF 卡中各个配置寄存器的地址都是偶地址，且只有 8 位，但实际在作读操作时，还是进行 16 位读操作，然后只取低字节数据进行总线扩展后送给 AHB 数据总线。

nPCE2	nPCE1	MA[0]	nPOE	nPWE	DA[15:8]	DA[7:0]
0	0	0	0	1	X	偶字节

3、Common Memory 模式写操作

这种模式下既可以对 CF 卡作 8 位访问，也可以作 16 位访问，CF 卡可以通过将 nPWAIT 信号置低，来要求 CF 控制器延长读写时间，直到 CF 卡已准备好数据。

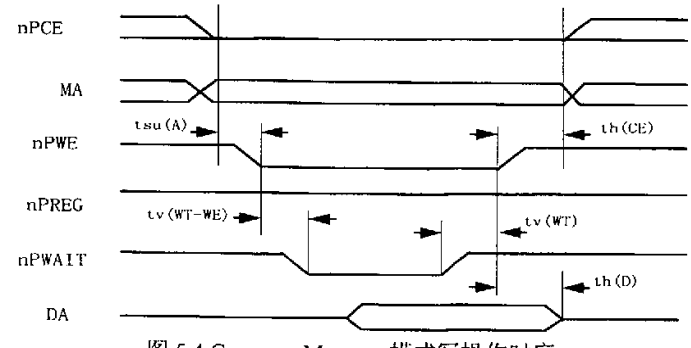


图 5.4 Common Memory 模式写操作时序

上图中各时间段定义见下表

项目	符号
数据相对 nPWE 的保持时间	th(D)
地址建立时间	tsu(A)
nPCE 相对 nPWE 的保持时间	th(CE)
nPWE 到 nPWAIT 下降沿时间	tv(WT-WE)
nPWAIT 释放到 nPWE 变高时间	tv(WT)

下表为 Common Memory 模式的写命令，通过 MA<0>和 nPCE 信号的组合，可以只对偶字节或奇字节写，或同时对奇偶字节写。

当 AHB 总线要求只对偶字节或奇字节写时，将总线地址直接送至外部地址线，同时将 nPCE2 置为高（表示无效），将 nPCE1 置为低。另外从 AHB 总线数据线上取得相应字节的数据，送至外部数据线上。

nPCE2	nPCE1	MA[0]	nPOE	nPWE	DA[15:8]	DA[7:0]
0	0	0	1	0	奇字节	偶字节
1	0	0	1	0	X	偶字节
1	0	1	1	0	X	奇字节

4、Common Memory 模式读操作

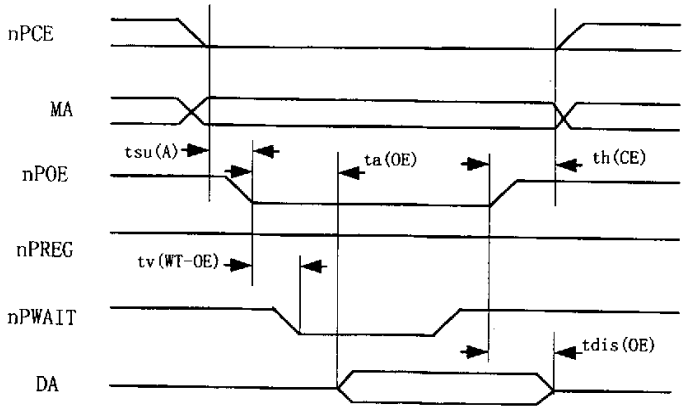


图 5.5 Common Memory 模式读操作时序

上图中各时间段定义见下表

项目	符号
nPOE 到数据有效时间	ta(OE)
nPOE 到数据无效时间	tdis(OE)
地址建立时间	tsu(A)
nPCE 在 nPOE 之后的保持时间	th(CE)
nPOE 到 nPWAIT 下降沿时间	tv(WT-OE)

下表为 Common Memory 模式的读命令，只需实现同时对奇偶字节的读操作，即半字操作，然后在控制器内部进行总线扩展即可。

nPCE2	nPCE1	MA[0]	nPOE	nPWE	DA[15:8]	DA[7:0]
0	0	0	0	1	奇字节	偶字节

5、I/O 模式写操作

对 CF 卡的 I/O 读写访问，可以是 8 位的，也可以是 16 位的。在读写过程中若 CF 卡将~IOIS16 管脚置为低时，表示为 16 位访问，否则为 8 位。若控制器试图进行一次 16 位访问，而 CF 卡将~IOIS16 置为高，则控制器应该进行两次 8 位访问。

另外，CF 卡可以通过将~nPWAIT 管脚置低，来要求控制器延长读写访问时间，直到 CF 卡已准备好数据，将此管脚置为高电平。

在这种模式下，nREG 信号必须置低。

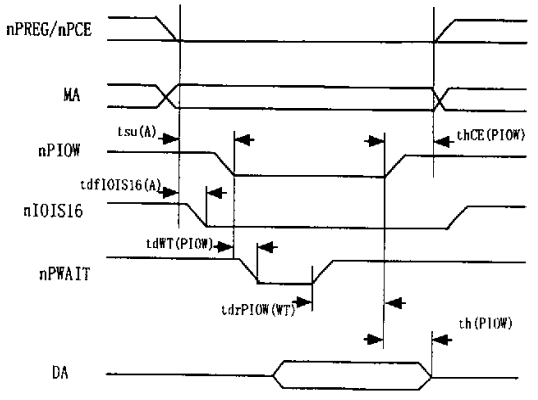


图 5.6　I/O 模式写操作时序

上图中各时间段定义见下表

项目	符号
数据相对 nPIOW 的保持时间	th(PIOW)

地址建立时间	tsu（A）
nPCE 相对 nPIOW 的保持时间	thCE(PIOW)
nPIOW 到 nPWAIT 下降沿时间	tdWT(PIOW)
nPWAIT 释放到 nPIOW 变高时间	tdrPIOW (WT)
地址有效到 nIOIS16 变低	tdfIOIS16(A)

下表为 16 位 I/O 模式的写命令（nIOIS16=0），通过 MA<0>和 nPCE 信号的组合，可以只对偶字节或奇字节写，或同时对奇偶字节写。

nPCE2	nPCE1	MA[0]	nPIOR	nPIOW	DA[15:8]	DA[7:0]
0	0	0	1	0	奇字节	偶字节
1	0	0	1	0	X	偶字节
1	0	1	1	0	X	奇字节

另有一种特殊情况，就是当对 8 位的 I/O 类型的设备进行 16 位的读写操作时，需要将一次 16 位的读写操作分解为连续的，两次 8 位 I/O 模式的读写操作。

下表为 8 位 I/O 模式的写命令（nIOIS16=1）

nPCE2	nPCE1	MA[0]	nPIOR	nPIOW	DA[15:8]	DA[7:0]
1	0	0	1	0	X	偶字节
1	0	1	1	0	X	奇字节

6、I/O 模式读时序

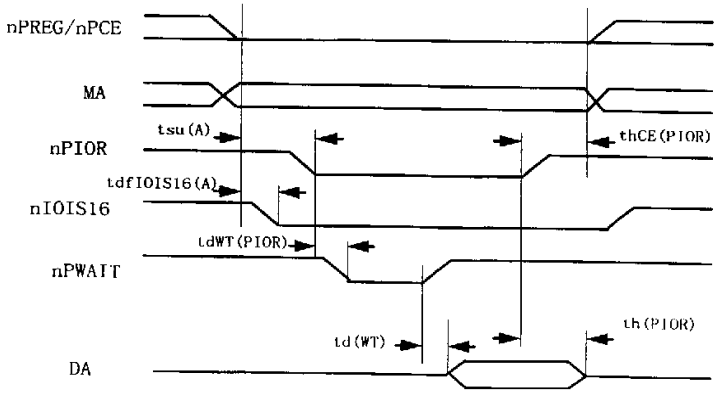


图 5.7　I/O 模式读操作时序

上图中各时间段定义见下表

项目	符号
nPIOR 到数据无效时间	th(PIOR)
地址建立时间	tsu(A)
nPCE 在 nPIOR 之后的保持时间	thCE(PIOR)
地址到 nIOIS16 下降沿时间	tdfIOIS16(A)
nPIOR 到 nPWAIT 下降沿时间	tdWT(PIOR)
nPWAIT 变高到数据有效时间	td(WT)

下表为 16 位 I/O 模式的读命令（nIOIS16=0）

nPCE2	nPCE1	MA[0]	nPIOR	nPIOW	DA[15:8]	DA[7:0]
0	0	0	0	1	奇字节	偶字节

下表为 8 位 I/O 模式的读命令（nIOIS16=1）

nPCE2	nPCE1	MA[0]	nPIOR	nPIOW	DA[15:8]	DA[7:0]
1	0	0	0	1	X	偶字节
1	0	1	0	1	X	奇字节

5.3 各子模块设计

CF 控制器由三个底层子模块文件和一个顶层文件，外加一个参数定义文件构成。三个底层子模块分别完成三部分功能：cf_interface.v 模块按照 AMBA－AHB 总线规范产生相应信号，以及 CF 控制器与 CF 卡之间接口信号时序的产生；cf_decoder.v 模块根据总线地址及各基址寄存器判断当前是对哪块内存地址空间进行访问，进而判断应该以何种模式访问 CF 设备，并根据相应时序控制寄存器的配置信息，产生正确的延时信息，送给状态机；cf_state.v 模块是一个状态机，具体介绍见 5.3.3

5.3.1 接口模块(cf_interface.v)

本模块根据状态机的状态变化，产生正确的 AMBA－AHB 总线信号，以及正确产生 CF 卡各接口信号：nPWE, nPOE, nPREG, nPIOR, nPIOW, nPCE2, nPCE1, MA（地址），DA（数据）。

本模块以组合电路为主。

5.3.2 译码模块(cf_decoder.v)

本模块输入为 AHB 总线地址信号、各基址寄存器及各时序控制寄存器，根据总线地址及各基址寄存器判断当前是对哪块内存地址空间进行访问，也即当前是以哪种模式访问 CF 卡，并根据相应模式的时序控制寄存器的值，产生正确的延时信息，送给状态机。本模块输出为各种延时信号及当前的访问模式。

本模块纯为组合逻辑电路。

5.3.3 状态机模块(cf_state.v)

本模块是 CF 控制器的核心，CF 控制器所有的输出信号都取决于状态机的当前状态。状态机离开`IDLE 状态后，向下一个状态的转变时间决定于时序控制寄存器中各延时时间的设置。

状态机的状态示意图如下：

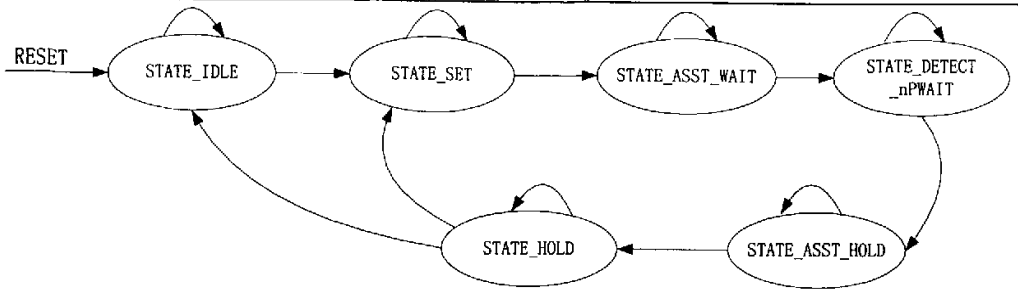


图 5.8 CF 控制器状态机状态转换图

如果是一次对 8 位设备的 8 位读写，或对 16 位设备的 16 位读写，则 STATE_HOLD 状态结束后回到 STATE_IDLE 状态。如果是一次对 8 位设备的 16 位读写，则 STATE_HOLD 状态结束后转到 STATE_SET 状态。

5.4 本章小结

CF 卡接口规范作为当前较普遍使用的计算机外部设备的接口规范，被广泛应用于存储器、网卡、多媒体卡等设备中。本章首先介绍了 CF 卡规范中对于 CF 卡设备的定义，整个 CF 控制器都是围绕这个规范来设计的。然后介绍了 CF 控制器的整体设计方案及所实现的时序。由于按照 CF 卡接口规范的要求，CF 卡设备可以有多种工作模式，这就要求 CF 卡控制器应该提供对各种工作模式的支持，这一点构成了 CF 卡控制器设计的重点和难点，本章主要围绕这一点展开。最后介绍了各个子模块设计及其之间的关系。

第六章 总结与展望

在完成各个模块的 Verilog HDL 代码编写之后，作者进行了大量的软件仿真验证工作，以最大程度的证实本设计的可行性与可靠性。整个验证工作是在一个完整的验证平台中进行的。

在这个验证平台系统中，包括一个 ARM 公司设计的微处理器核（ARM7TDMI），一个 Synopsys 公司提供的完整的 AMBA 总线模型(以 DESIGNWARE 的形式给出)，一个片内 DMA（Direct Memory Access）模块，以及本课题设计实现的存储控制器（此存储控制器与 AMBA-AHB 总线相连，作为总线从设备）。其中 ARM7TDMI 和 DMA 模块用来作为总线主设备，发起对存储控制器的数据传输请求，以验证存储控制器的正确性。另外，为了更客观的验证设计，在进行验证的过程中，使用了 SRAM 及 SDRAM 芯片的 Verilog HDL 模型，将这些模型与存储控制器连接，以验证各种操作的完成情况。

ARM7TDMI 作为总线主设备，只能产生对从设备的单拍读写请求，而不会产生 BURST 类型的传输请求，因此在上面的验证系统中，包括了两个总线主设备：ARM7TDMI 和 DMA。后者在经过配置以后，可以产生所要求的各种 BURST 传输请求。

为了使总线主设备（包括 ARM7TDMI 和 DMA）在验证时产生各种传输请求信号，以增加验证的完备性，在验证过程中编写了大量的汇编程序，使用包括字节操作指令（STRB、LDB）、半字操作指令（STRH、LDH）和字操作指令（STR、LD），并结合 DMA 的功能，来达到增加验证完备性的目的。验证系统使用 ARM 公司的软件开发平台 ARM PROJECT MANAGER 作为汇编器。

经过验证，所设计的存储控制器能正确完成要求的各项操作。

另外，整个设计经过了预综合，证明了设计的可综合性。

展望

随着存储器技术的突飞猛进，各种存储容量大、存取速度快的存储器不断面世，并应用在个人电脑及服务器领域，只是由于受嵌入式系统的实际应用需求及嵌入式处理器的主频所限，嵌入式系统中对于外部存储器的读写速率的要求还没有达到个人电脑及服务器中的要求那么高。比如在 PC 机中已广泛使用的 DDR SDRAM(Doul Data Rate SDRAM)，这种 SDRAM 由于可以在同步时钟的上升沿和下降沿都进行数据传输，因此读写速度得到大幅度提高，但

由于以上提到的缘由，这种存储器还没有在嵌入式系统中得到应用。另外 Rambus 公司推出的 RDRAM（Direct Rambus DRAM）又将 DRAM 的性能提高到一个新的高度，用 RDRAM 代替 SDRAM 可以使系统性能提高三分之一[14]。随着嵌入式微处理器主频的不断提高，及嵌入式系统对系统主频的要求不断提高，相信很快在嵌入式系统中也将要求使用这些种类的存储器，因此使得开发支持这些存储器的控制器的工作变得很有意义和必要。

致　谢

在此，首先要感谢我的研究生导师：陆生礼博士。本课题从选题、方案制定、具体实施，到论文写作完成，整个过程都是在陆老师的指导、帮助和关心之下完成的，陆老师渊博的学识和严谨的治学态度都使我受益匪浅。更让我感动和钦佩的是他高尚的人品和博大的胸怀，以及对工作一丝不苟的作风，他在这方面对我的影响甚至将超过他教给我的具体科学知识对我的影响，必将在我以后的工作、学习、生活中树立一个高尚的榜样。

论文完成之后，也得到了凌明老师的仔细审阅，并提出了许多宝贵的意见和建议，在此表示深深的谢意。

王镇同学在本课题进行过程中承担了部分代码编写工作，我们之间进行了很多次讨论，这些讨论对课题的进行非常重要，对我的帮助很大，在此也谢谢他。

我在ASIC工程中心学习、生活的这三年当中，得到了时龙兴老师、张嗣忠老师、吴建辉老师、张盟老师等各位老师的谆谆教诲和帮助，使我顺利度过了研究生生活，在此向他们表示深深的谢意。

谨以此文献给赐予我养育之恩的父母亲，以及一直默默支持我学业的妻子。

参 考 文 献

1、David Flynn. " AMBA: Enabling Reusable On-Chip Design ", IEEE Micro, July/August 1997

2、Mike Keating, Pierre Bricaud. " Reuse Methodology Manual" Edition 2. Kluwer Academic Publishers, 2000

3、Jacobo Riesco, Juan C. Diaz and Plierre I. Plaza. " The uPP: Design, Methodologies and Tools for a Pay Phone System-On-a-Chip Based on an ARM Core and Design Reuse " , IEEE 1999 Custom Integrated Circuits Conference.

4 、Jan M. Rabaey. " Digital Integrated Circuits: A Design Perspective ", 清华大学出版社, Prentice Hall, 1996

5、Wai-Kai Chen. " The VLSI Handbook ", CRC Press LLC, 2000

6、ARM Ltd . " AMBA Specification (Rev 2.0) ", ARM Ltd. 1999.

7、Wang Zhonghai, Ye Yizheng, Wang Jinxiang, and Yu Mingyan. "Designing AHB/PCI bridge " , ASIC, 2001. Proceedings . 4th International Conference on 2001

8、Sean W. McGee, Robert H. Klenke, James H. Aylor and Andrew J. Schwab. "Design of a Processor Bus Interface Asic For The Stream Memory Controller", ASIC Conference and Exhibit, 1994. Proceedings., Seventh Annual IEEE International , 19-23 Sep 1994

9 、Silvije Jovalekic, Martin Rieger and Ralph Runge. " Design of a Microcomputer Platform With Real-Time Operating System For Laboratory Projects ", Frontiers in Education Conference, 1997. 27th Annual Conference. 'Teaching and Learning in an Era of Change'. Proceedings. , Volume: 3 , 5-8 Nov 1997

10、Mark Muegge, Michael G . Davis. "High performance bi-directional FIFO for bus interface applications " , Northcon/93, Conference Record, 12-14 Oct 1993

11 、David Wyland. " New features in synchronous FIFOS " , Northcon/93, Conference Record, 12-14 Oct 1993

12、S. Osborne, A.T. Erdogan , T. Arslan and D. Robinson. " Bus encoding architecture for low-power implementation of an AMBA based SOC platform " , IEE Proc-Comput Digit. Tec. Vol 149. No.4 July 2002

13 、Andy Nightingale and John Goodenough. " Testing For AMBA Compliance" , ASIC/SOC Conference, 2001. Proceedings. 14th Annual IEEE International , 2001

14 、Matthias Gries. " The Impact of Recent DRAM Architectures on Embedded System Performance ", Euromicro Conference, 2000. Proceedings of the 26th , Volume: 1 , 2000

15、CompactFlash Association. " CF+ and CompactFlash Specification

Revision 1.4 ", CFA 1999

16、J. Bhasker. " Verilog HDL 硬件描述语言 ", 徐振林 等译, 北京, 机械工业出版社, Star Galaxy Publishing . 2000

17、Motorola Inc. " MC68SZ328 INTEGRATED PROCESSOR User's Manual ", Revision 0. Motorola Inc, 2 August, 2001

18、Motorola Inc. " MC9328MX1 Integrated Portable System Processor Reference Manual " , Revision 0, 4/2002

19、Intel Inc. " Intel PXA250 and PXA210 Aplication Processors Developer's Manual" , Intel Inc. February 2002

20、Micro Technology Inc. " Migrating From FPM/EDO To SDRAM ", Technical Note: TN 48-07, Micro Technology Inc.

21、Micro Technology Inc. " Various Methods of Dram Refresh " , Technical Note: TN 04-30, Micro Technology Inc. 1994

22、Winbond Electronics Corp. " W981216BH User Manual " Publication Release Date: September 20, 2001

已发表论文

1、康军，黄克勤，张嗣忠。“同步电路设计中 CLOCK SKEW 的分析” 《电子器件》，2002 年第 25 卷，第 4 期

基于AMBA-AHB总线规范的IP-存储控制器的设计

作者： 康军
学位授予单位： 东南大学