摘 要

随着汽车的普及和大量使用，国内汽车制造业得到了快速的发展，汽车电子在汽车制造业中占有十分重要的地位，而电子控制单元（ECU）是汽车电子核心器件，实现对发动机、车身、底盘的控制并具有通信功能。

目前，国内部分 ECU 采用单片机加外围器件来实现，具有集成度低、体积大、可靠性差等不足；ECU 芯片大部分依赖进口，价格昂贵，只适合在高档轿车中使用。

本文设计了基于 CAN 总线的汽车电子控制单元；依据设计要求，提出设计方案。采用流水线技术，裁减并完善了兼容标准的 8051MCU 核，减少了设计面积，又提高了处理速度。按照自顶而下的设计原则，分别设计了算术逻辑单元、中心控制器、定时/计数器、串行口和 2.0A 标准 CAN 总线单元。

为了确保设计的正确性，按照各部分的特点，分别设计仿真、测试文件及引导程序，实现了在 FPGA 验证过程中的自动装载功能。

该电子控制单元可以实现对车身、底盘等低速设备的控制，具有 CAN 总线通信功能，具有使用方便、处理速度快、集成度高、可靠性好、价格低廉等优点，具有极其广泛的应用前景。

关键词：汽车电子；电子控制单元；IC 设计；MCU；CAN 总线；

Abstract

With the popularization and widely using of automobile, the automobile manufacturing is developing quickly in our country; the automobile electronics has occupied very important position in the field of the automobile manufacturing. Electronic Control Unit (ECU) is the core component in automobile electronics; it acts a part in some aspects such as the engine, bodywork, chassis controlling systems and the communication function.

At present, most of the ECU is designs using singlechip with peripheral circuit, it has some defects such as low integration rate, bulky, bad dependability; most of ECU chip depends on import, it is so costliness that only suit to use on top grade cars.

The thesis discusses the automobile ECU including the CAN bus, according as the design demand, it puts forward the design scheme, adopts the pipeline technique, designs and reduces the standard 8051 MCU core, not only cuts down the area but also increases the processing speed. According Top-Down designing principle, designs the units such as ALU, central controller, timer/counter, the serial port and the standard CAN bus unit based on 2.0A .

In order to insure the correctness, the thesis designs test files respectively bases on the characteristic of each unit, designs the guiding program, realizes the automatic loader function in the FPGA verification procedure.

This design realizes to control the lows speed equipment such as bodywork、chassis and so on, obtains CAN bus communication function, it has some advantages such as easing to use, fast processing speed, high integration, good dependability and the low price etc, has a good prospect of application.

Keywords: Automobile Electronics; Electronic Control Unit; IC design; MCU; CAN Bus

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，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 天津工业大学 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。

学位论文作者签名：杜向军　　签字日期：2007年3月1日

学位论文版权使用授权书

本学位论文作者完全了解 天津工业大学 有关保留、使用学位论文的规定。特授权 天津工业大学 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：杜向军　　导师签名：

签字日期：2007年3月1日　　签字日期：2007年3月1日

学位论文的主要创新点

一、本文利用流水线技术把标准 51 IP 核应用到 ECU 中，提高处理速度，并加入了现场 CAN 总线，用于通信和远程控制。

二、设计引入了自动引导程序，实现了验证过程的自动装载。

第一章 前言

1.1 电子控制单元（ECU）的发展现状

由于经济的快速发展，人们生活水平的进一步改善，对汽车安全、舒适等方面的要求不断提高。传统的机械装置与技术在汽车领域的应用已趋于成熟，有些甚至已达到其物理极限，要进一步发展具有局限性，在这些因素的影响下，使得汽车电子技术在汽车整车和零部件上的应用越来越广泛。现在，每一辆汽车几乎都是机械、电子和信息一体化装置，以至有人认为汽车正在由一个拥有大量电子技术与装置的机械系统向一个由一定机械装置支撑的电子电气系统转变。与五年前相比，如今车内增加了许多电子部件。除了 AM/FM 收音机、CD 播放机以及电动门窗和后视镜以外，还有 GPS、卫星收音机、DVD/视频屏幕、自动气候控制以及油耗/里程监控等功能。

由于电子控制、计算机、通信等技术的迅猛发展，使汽车电子产品技术和产品的开发日新月异。汽车的电子化程度是衡量一个国家汽车工业发展水平的重要标志。汽车电子可分为发动机电子、底盘电子、车身电子、信息通信与娱乐系统四大类。从汽车技术的发展现状看，汽车电子技术是支撑现代汽车发展的基础技术之一，已不是简单地对汽车的单个零部件进行电子控制，而是汽车进行优化综合控制。因此智能化、综合化、网络化控制是汽车电子控制的重要发展方向[1]。电子技术已在车辆发动机控制、底盘控制，故障诊断以及音响、导航等各个方面得到广泛应用，显著提高了车辆的整体性能。因此近些年来，电子控制单元（Electronic Control Unit-ECU）格外受到重视[2]。

所谓电子控制单元，实际上就是一部单片机，有自己的处理器、I/O 设备和简单的存储器。电子控制单元正向系统综合化、网络化、高度集成化方向发展。采用计算机网络技术，通过数据总线将各个 ECU 连接在一起，产生综合电子控制系统，有些高档的轿车，往往拥有几十个甚至上百个 ECU，这些 ECU 通过数字总线（CAN、FlexRAY、LIN 等）结构连接在一起，形成一个控制系统。电子控制单元在汽车上的广泛应用，使得汽车的动力性、经济性、安全性、舒适性、可靠性都得到了显著的改善和提高[3]。

电子控制单元是汽车电子中的核心器件，是中央控制单元，通过车内网络完成协调与连接，完成对发动机、车身、底盘、通信等系统的控制。本文设计一种通用型电子控制单元，采用流水线技术、优化了系统结构，利用现场 CAN 总线作为通信、控制总线，减少了线束，避免了汽车因功能的增加而导致的电子装置数量膨胀，

实现了对汽车各个部分的控制。

1.2 课题研究的目的及意义

本课题是天津市“十五”05 科技创新项目——汽车中电子控制单元（ECU）的集成电路设计。

电子控制单元是汽车电子中的核心器件，实现对发动机、车身、底盘的控制并具有通信功能，所以电子控制单元关系到控制系统甚至整个车的稳定性和安全性。目前国内电子控制单元部分依赖进口芯片或者应用单片机来实现。一种应用简单、稳定性好、集成度高的电子控制单元可以减少汽车的设计成本、增强稳定性、减少设计面积，有利于汽车工业的发展。

课题设计了兼容标准 51 的控制器，实现了应用简单、方便等特点，又采用了流水线技术、优化的系统结构，提高了处理速度。可对车身系统进行控制，如：车门、车锁、后视镜等低速控制部分，具有面积小、速度快等优点，可以缓解国内 ECU 市场依赖国外产品的现状，增强国内产品的市场竞争力。

1.3 课题研究的内容

本课题主要开发一种用于汽车的、新型的、快速的、结构简单的电子控制单元，为此本文分析了 ECU 的要求，提出了设计原则与方案，并以此为基础，设计了 ECU 的各个组成部分：中央控制单元、串行口单元、CAN 总线单元等等。并对各个单元进行了分析、模拟以及 FPGA 验证。

归纳起来，本文主要的研究工作如下：

1. 提出了设计原则与方案；
2. 总结出 ECU 的性能与指标；
3. 依据自顶而下的设计方案，裁减并完善 MCU 核的各个组成部分；
4. 设计 CAN 总线单元；
5. 设计各个部分的仿真模块，并完成仿真设计；
6. 设计验证模块，引导程序，完成整体设计的 FPGA 级验证。

第二章 电子控制单元（ECU）的设计原则与方案

2.1 要求分析

电子控制单元是汽车电子的核心部件，不但发动机上应用 ECU，在其它许多地方如防抱死制动系统、四轮驱动系统、电控自动变速器、主动悬架系统、安全气囊系统、多向可调电控座椅等都配置有各自的 ECU。ECU 主要是由 MCU、存储器和输入/输出接口电路构成。输入电路接受传感器和其它装置输入的信号，对信号进行调理和放大。MCU 将上述已经预处理过的信号进行运算处理，并将处理数据送至输出电路。输出电路是将数字信号的驱动功率放大，有些还要还原为模拟信号，使其驱动被控的调节伺服元件工作。汽车控制系统是一个复合的大系统，由各部件 ECU 互连而成。随着汽车 ECU 的增多，线路会日益复杂。为了简化线束、降低成本，汽车上多个 ECU 之间的信息传递需要可靠、实时的网络总线（如 CAN 总线）做支撑，将整车的 ECU 形成一个网络系统。如图 2-1 所示是一个 ECU 的具体应用[4]。

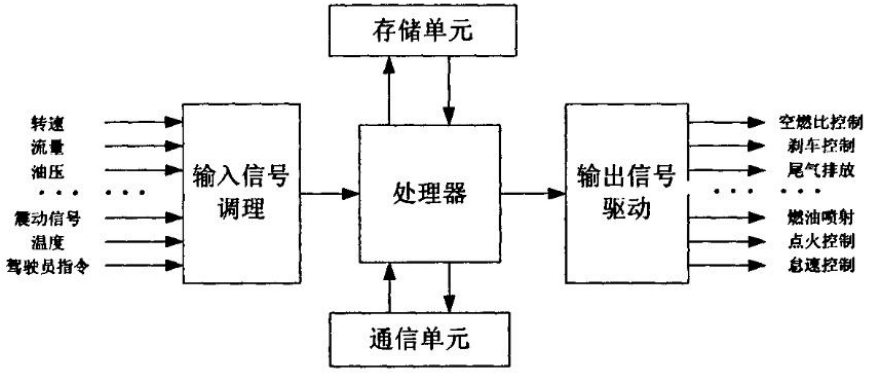


图 2-1 ECU 应用

ECU 的输入信号包括驾驶员控制指令和控制对象的状态检测信号，状态检测信号是通过相应的传感器送入的。这些信号基本可分为电压信号、电平信号和脉冲计数信号。电压信号需要通过模数转换，以数字的方式传送给微处理器；电平信号要求采用开关量输入的方式接入；汽车的脉冲检测信号具有高频率、强实时的特点，因此要求被处理器实时处理[5]。另外此类信号通道较多，处理器需要有多个独立的定时/计数模块进行并行处理。对于 ECU 的控制输出来说，微处理器输出的信号往往被用作控制电磁阀的电流信号、指示灯的开/关信号、规定的周期脉冲信号（如用于发动机燃油喷射控制）、用于驱动步进电机的一系列固定周期的脉冲信号等。一般

很少要求直接由ECU提供模拟的控制量实现对执行器的控制，因此输出信号主要是开关量输出和频率/脉冲量输出，同样对于频率/脉冲输出量也要求独立的定时/计数模块进行并行处理。

2.2 设计原则

（1）计算机技术与微电子技术紧密结合

带微处理器的ECU开发是一个系统工程，需要有计算机软硬件、微电子集成电路、系统开发和应用等方面的知识结合、应用。本设计采用了工作站作为设计平台，IC设计软件为设计手段，FPGA开发板为验证工具来实现整体设计。

（2）采用软硬件协同设计方法

在设计过程中，采用流行的软硬件协同设计方法，通过对应用特征分析和提取，使得结构设计和符号指令设计并行设计，软件开发和硬件实现并行进行。软硬件协同的方法有助于对系统进行全面性能评价、建立系统软件环境和功耗模型、便于对系统和局部功耗进行模拟和评价。

（3）正向设计、测试方法紧密结合

采用正向设计方法，分别进行了ECU的系统描述、功能设计和逻辑设计等过程，并同时进行了各个部分的测试、模拟仿真工作[6]。

2.3 设计方案

本设计采用自顶而下的设计方案，把设计分为系统级、功能级和门级，按照自上而下的设计顺序，在不同层次上对系统进行了描述、设计和仿真[7]。

设计分为以下几个阶段：

（1）初步设计阶段

在这个阶段收集设计实现的方法，确定处理器的功能和相应的体系结构。

（2）外部描述阶段和解释器建模阶段

此阶段要对处理器外部功能和行为进行描述，处理器的行为功能主要是由指令集确定的。要完成指令的文字解释，既解释器，还要完成解释器相应的测试程序。因本设计采用51兼容的指令集，所以解释器可以用通用型解释器。

（3）内部深入描述阶段

在内部描述中，需要考虑处理器主要的体系结构特性，如数据通路、流水线、寄存器、主要的总线、中断处理机制和时续关系问题等。

（4）粗略结构模型设计阶段

在这个阶段设计过程中，用硬件描述语言建立模型，此过程涉及到处理器模型的内部描述，包含最终实现门级网表时所需的主要电路。

（5）门级建模阶段

用基于一些 Logic 公司提供的标准单元库来实现设计，从而得到门级网表。

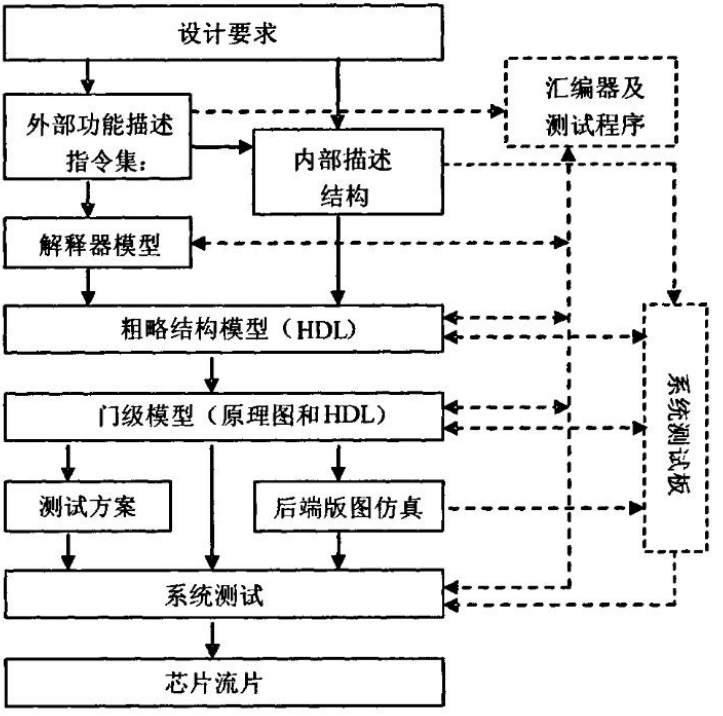


图 2-2 设计流程

第三章 8051 MCU 核的单元设计

3.1 集成电路的设计方法与流程

目前，以集成电路为核心的电子信息产业超过了以汽车、石油、钢铁为代表的传统工业成为第一大产业，成为改造和拉动传统产业迈向数字时代的强大引擎和雄厚基石。1999 年全球集成电路的销售额为 1250 亿美元，而以集成电路为核心的电子信息产业的世界贸易总额约占世界 GNP 的 3%，现代经济发展的数据表明，每 1~2 元的集成电路产值，带动了 10 元左右电子工业产值的形成，进而带动了 100 元 GDP 的增长。作为当今世界经济竞争的焦点，拥有自主版权的集成电路已日益成为经济发展的命脉、社会进步的基础、国际竞争的筹码和国家安全的保障[8]。

3.1.1 设计方法

集成电路设计有许多具体的方法，这些方法适用于不同的设计要求。设计者可以根据产品的产量、设计复杂度、设计周期的约束等决定使用哪一种方法[9]。

3.1.1.1 全定制法

全定制法的特点是针对每个晶体管进行电路参数和版图的优化，以获得最佳的性能（包括速度和功耗）以及最小的芯片面积。它适用于要求得到最高速度、最低功耗和最省面积的芯片设计。这种设计方法的工作效率较低，设计周期长。全定制法适用于要求得到最高速度、最低功耗和最小面积的芯片设计。

3.1.1.2 定制法

定制法适用于芯片性能指标比较高、批量生产、面积比较大的芯片设计。通常分为两大类：

（1）标准单元法（Standard Cell Method）。所以称之为“标准单元”是由于电路中各单元的高度是相等的，只是在宽度上有差别。

（2）通用单元法（General Cell Method）。又分为积木块法和混合法两种。在这类设计中各单元的高度和宽度不再相等。

定制法的特点是设计上自由度较大，芯片中没有无用的单元或晶体管，芯片面积较小。但建立一个物理单元库需要很大的初始投资。此外，制造周期较长，成本也较高。

3.1.1.3 半定制法

半定制法使用于要求设计成本低、设计周期短、批量生产、面积比较小的芯片设计。一般采用此方法设计出产品并投入市场，在占领市场后再用其它方法进行一

次再设计。

半定制法包括数字电路门阵列和线性阵列（Linear Array）两大类。前者简称为门阵列，它又分为通道门及门海两种。门阵列和线性阵列都是预先在芯片上已生成了由基本门（或单元）所组成的阵列，即完成了连线以外的所有芯片加工工序。设计完成后只需加工接触孔和连线即可实现设计。通常这种方法对门阵列的利用率较低，芯片的面积较大

3.1.1.4 模块编译法

模块编译法是一种全自动的设计方法。先对设计模块的性能进行描述，再通过编译直接得到电路的掩膜版图。这种方法适用于 ROM、RAM、ALU、移位寄存器、乘法器等规则结构和模块式结构的芯片设计。

3.1.1.5 可编程逻辑器件法

FPGA（Field Programmable Gate Array，现场可编程门阵列）与 CPLD（Complex Programmable Logic Device，复杂可编程逻辑器件）都是可编程逻辑器件，它们都可以编程写入复杂的组合、时序逻辑，并且在实验室内就可以实现，逻辑修改也非常方便。用可编程逻辑器件进行设计可以大大缩短设计的周期，降低设计风险，对于产量小的设计，成本可以得到较大的降低。

3.1.1.6 基于 IP 重用的设计方法

SOC 设计一般采用多个 IP 模块，包括 DSP IP、MCU IP 及存储器 IP 等，设计者把这些 IP 组合起来实现一个系统的功能，然后对整个设计进行仿真验证，对局部进行修改。这种设计方法相当于把以往的系统集成放到芯片设计中完成，各个 IP 就像以往的芯片。运用 IP 的方法使设计提高到一个较高的层次，设计者可以把注意力放到系统级设计上。从降低设计成本和风险的角度考虑，本设计在开发设计阶段就采用了 FPGA 验证设计。测试系统设计没有问题后，可以基于标准单元法进行版图的设计。在本论文中仅详细讨论系统在 FPGA 设计阶段所涉及的内容[10]。

3.1.2 集成电路设计流程

在数字集成电路设计过程中，要最先写出设计规范，它能抽象地描述所设计电路的功能、接口和整体结构。通过分析电路的功能、性能所要满足的标准以及其它高级的问题后，才能进行行为级描述。行为级的描述可以用硬件描述语言（HDL）来撰写。详细的设计流程如图 3-1 所示[11]。

完成了行为级描述的行为算法优化与功能仿真之后，由于现有的电子设计自动化（EDA）工具只能对 RTL 级描述的 HDL 文件进行自动逻辑综合，因而需要将行为级描述用手工转换成寄存器传输级（RTL）的 HDL 描述。转换后的 RTL 描述同样需要进行仿真验证。从这之后，设计过程是在计算机辅助设计工具的帮助下完成

的。

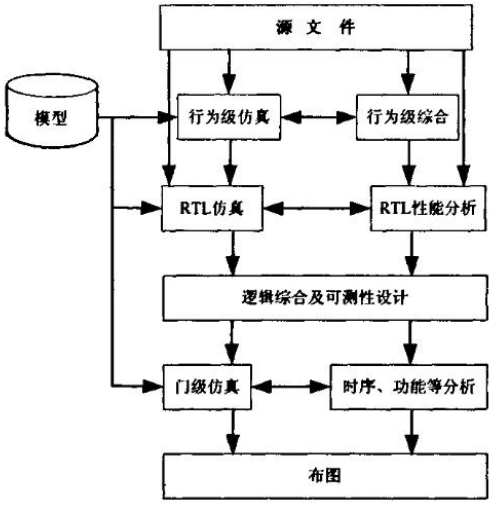


图 3-1 集成电路的设计流程

逻辑综合的目标是将 RTL 的 HDL 代码映射到具体的工艺上加以实现，逻辑综合的仿真叫做门级仿真。逻辑综合工具把 RTL 描述转换成门级网表[12]。门级网表是使用门电路以及门电路之间的连接来描述电路的方式，是产生版图的自动布局布线工具的输入。在版图的布局、布线都已确定后，可以从版图中进一步提取出连线电阻、电容等参数。

在版图生成之后，把从版图中提取的参数反标到门级网表中，进行包含门延时、连线延时的门级仿真，称作后仿真。这一步主要是进行时序模拟，如果时序不能满足设计要求，通常需要修改版图的布局布线、逻辑综合的约束条件，有时也可能回到 RTL 描述、行为级描述甚至设计规范或算法实现上加以调整。版图得到验证后就可以做到硅片上。

3.2 8051 核的特点及特性

3.2.1 8051 MCU 的结构

8051MCU 包含中央处理器、程序存储器（ROM）、数据存储器（RAM）、定时/计数器、并行接口、串行接口和中断系统等几大单元及数据总线、地址总线和控制总线等三大总线，图 3-2 是 8051MCU 的内部结构示意图[13]。

3.2.1.1 微控制器（MCU）

微控制器是核心部件，具有 8 位数据宽度的处理器，能处理 8 位二进制数据或

代码，MCU 负责控制、指挥和调度整个单元系统协调的工作，可以完成指令译码，资源分配，中断现场保存和恢复，控制输入输出功能等操作。8051MCU 中的 ALU 单元负责数据运算，如加减乘除、布尔操作、十进制调整等。

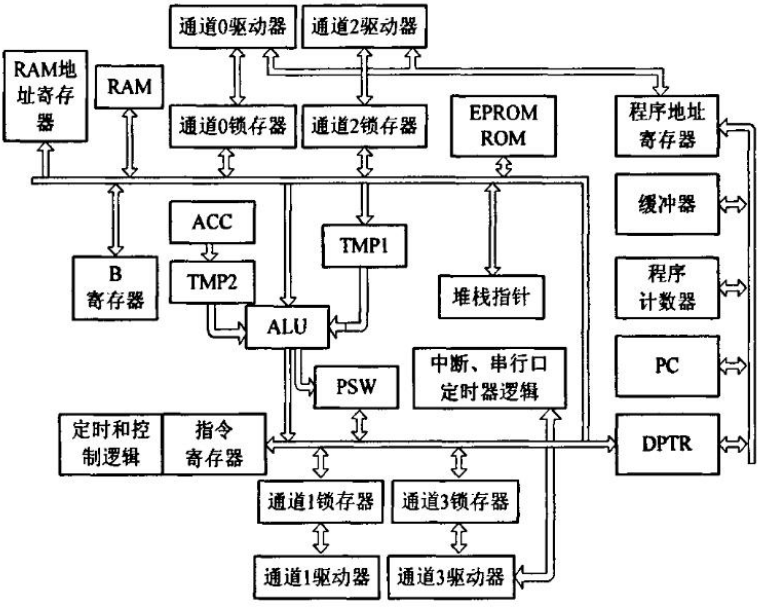


图 3-2 MCU 内部结构图

3.2.1.2 存储器结构

MCU 片内有 4KB 的程序存储单元，其地址为 0000H-0FFFH。启动复位后，程序计数器的内容为 0000H，系统将从 0000H 单元开始执行程序。但在程序存储器中有些特殊的单元。

其中一组特殊单元是 0000H-0002H 单元，系统复位后，PC 为 0000H，单片机从 0000H 单元开始执行程序，如果程序不是从 0000H 单元开始，则应在这三个单元中存放一条无条件转移指令，让 MCU 直接去执行用户指定的程序。

另一组特殊单元是 0003H-002AH，这 40 个单元各有用途，它们被均匀地分为五段，专门用于存放中断处理程序的地址单元，中断响应后，按中断的类型，自动转到各自的中断区去执行程序。因此以上地址单元不能用于存放程序的其他内容，只能存放中断服务程序。

第二个存储区是 MCU 内 128 字节的内部 RAM，这部分主要是作为数据段，存放执行的中间结果和过程数据的。但是必须节省使用，因为它的空间毕竟有限。在数据段中也可通过特殊寄存器 R0 和 R1 采用间接寻址，R0 和 R1 被作为数据区的指

针，将要恢复或改变字节的地址放入 R0 或 R1 中，根据源操作数和目的操作数的不同，执行指令需要一个或两个周期[14]。

数据段中可以分为两个小段，第一个子段包含四组寄存器组，每组寄存器组包含八个寄存器，可在任何时候通过修改 PSW 寄存器的 RS1 和 RS0 这两位来选择四组寄存器的任意一组作为工作寄存器组。MCU 也可默认任意一组作为工作寄存器组。工作寄存器组的快速切换不仅使参数传递更为方便，而且可在 MCU 中进行快速任务转换。另外一个子段叫做位寻址段，包括 16 个字节共 128 位。每一位都可单独寻址。

特殊功能寄存器（SFR）也称为专用寄存器，特殊功能寄存器反映了 MCU 的运行状态。很多功能也通过特殊功能寄存器来定义和控制程序的执行。

MCU 有 21 个特殊功能寄存器，它们被离散地分布在内部 RAM 的 80H-FFH 地址中，这些寄存的功能已作了专门的规定，用户不能修改其结构[15]。

3.2.1.3 定时/计数器

MCU 有两个 16 位可编程的定时/计数器，它们具有四种工作方式，其控制字和状态均在相应的特殊功能寄存器中，通过对控制寄存器的编程，就可方便地选择适当的工作方式。下面对定时/计数器的特性进行阐述。

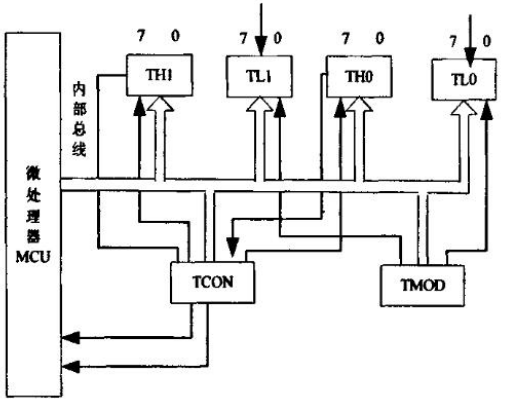


图 3-3 MCU 内部定时/计数器结构图

MCU 内部的定时/计数器的结构如图 3-3 所示，定时器 T0 特性功能寄存器 TL0（低 8 位）和 TH0（高 8 位）构成，定时器 T1 由特性功能寄存器 TL1（低 8 位）和 TH1（高 8 位）构成。特殊功能寄存器 TMOD 控制定时寄存器的工作方式，TCON 则用于控制定时器 T0 和 T1 的启动和停止计数，同时管理定时器 T0 和 T1 的溢出标志等。程序开始时需对 TL0、TH0、TL1 和 TH1 进行初始化编程，以定义它们的工作方式和控制 T0 和 T1 的计数[16]。

3.2.1.4 中断结构

MCU 有 5 个中断源，它们是两个外中断 INT0（P3. 2）和 INT1（P3.3），两个片内定时/计数器溢出中断 TF0 和 TF1，一个是片内串行口中断 TI 或 RI，这几个中断源由 TCON 和 SCON 两个特殊功能寄存器进行控制。

MCU 有两个中断优先级，即高优先级和低优先级，每个中断源都可设置为高或低中断优先级。如果有一低优先级的中断正在执行，那么高优先级的中断出现中断请求时，MCU 则会响应这个高优先级的中断，也即高优先级的中断可以打断低优先级的中断。而若 MCU 正在处理一个高优先级的中断，此时，就算是有低优先级的中断发出中断请求，MCU 也不会理会这个中断，而是继续执行正在执行的中断服务程序，一直到程序结束，执行最后一条返回指令，返回主程序然后再执行一条指令后才会响应新的中断请求。

3.2.1.5 串行口

MCU 内部有一个功能很强的全双工的串行口，该串行口有四种工作方式，波特率可以由软件设置，由片内的定时器/计数器产生。串行口有两个物理上独立地接收、发送缓冲器 SBUF，可以同时发送、接收数据，发送缓冲器只能写入不能读出，接收缓冲器只能读出不能写入，两个缓冲器公用一个字节地址（99H）。串行口的接收、发送数据均可触发中断系统。同时，还有两个控制寄存器来控制串行口，它们是特殊功能寄存器 SCON 和 PCON。有关串行口的详细情况将在设计中描述。

3.2.2 8051 指令系统

8051 系列单片机的指令系统是一种简明高效的指令系统，其基本指令共有 111 条，其中单字节指令 49 条，双字节指令 45 条，三字节指令 17 条。如果按功能可以讲这些指令分为五类：数据传送类（29 条）、算术操作类（24 条）、
逻辑操作类（24 条）、控制转移类（17 条）以及位变量操作类（17 条）。8051 系列单片机的 111 条指令一共采用了 5 种寻址方式。5 种寻址方式以及它们的寻址空间如表所示。

表 3-1 寻址方式

寻址方式	寻址空间
寄存器寻址	R0-R7、A、B、Cy、DPTR、AB
直接寻址	内部 RAM、特殊功能寄存器、所有的可寻址位
寄存器间接寻址	内部 RAM（@R0，@R1，@SP），内部 RAM 单元的低 4 位（@R0，@R1），外部数据存储器（@R0，@R1，@DPTR）
立即寻址	程序存储器
基址寄存器加变址寄存器间接寻址	程序存储器（@DPTR+A，@PC+A）

3.2.2.1 寄存器寻址

寄存器寻址方式可用于访问选定寄存器区的 8 个工作寄存器 R0--R7。由指令操作码的低三位指示所用的寄存器，寄存器 A、B、DPTR，AB 和 Cy 位（位处理机的累加器）也可作为寻址对象。在这种寻址方式中被寻址的寄存器的内容就是操作数。

在实现这类寻址方式时，确定被寻址寄存器的物理地址时关键。由于选定寄存器区由 PSW 的相关位来决定，指令的低三位又指示了具体的寄存器，所以可以用下面的 VHDL 语句来确定相关寄存器的物理地址：

rr_adr<=unsigned((psw and "00011000") or (rom_data_i and "00000111"));

其中 rr_adr 表示的是寄存器的物理地址，rom_data_i 表示的是指令代码。物理地址确定以后，对 RAM 的操作就很好进行。

3.2.2.2 直接寻址

直接寻址是访问特殊功能寄存器的唯一方法。它也可以用于访问内部 RAM（0128 个字节）。采用直接寻址方式的指令是双字节指令，其中第一个字节是操作码，第二个字节是内部 RAM 或特殊功能寄存器的直接地址。地址已经给出，显然不需要象寄存器寻址那样先计算地址，直接针对由 ROM 给出地址就可以对 RAM 进行相关操作。

3.2.2.3 寄存器间接寻址

寄存器间接寻址可用于访问内部 RAM 或者外部数据存储器。访问内部 RAM 或者外部数据存储器的低 256 个字节时，可以采用 R0 或 R1 作为间址寄存器。这类指令为单字节指令，其最低位表示采用 R0 还是 R1 作为间址寄存器。访问内部 RAM 和外部数据存储器时采用不同的指令，所以不会引起混淆[13]。

访问外部数据存储器，还可用数据指针 DPTR 作为间址寄存器，DPTR 是 16 位寄存器，故它可对整个外部数据存储器空间（64K）寻址。

在执行 PUSH（压栈）、POP（出栈）指令时，也采用寄存器间接寻址，这时堆栈指针 SP 用作间接地址寄存器。

实现这类寻址方式必须分为两步，首先确定 4 个通用工作寄存器区中可以作为间接寻址寄存器的 8 个单元的地址，然后读出所选定的寄存器中的值，这个值就是当前指令要寻址的空间的物理地址。确定间接寻址寄存器的地址可以由下面的 VHDL 语句来实现：

ri_adr<=((psw and "00011000") or (s_command(7 downto 0) and "00000001"));

其中 ri_adr 表示的是用于间接寻址的寄存器的物理地址，s_command 表示的是当前指令的操作码。然后通过另外一个读 RAM 的进程就可以确定间址寄存器中的值，从而得到指令需要的存储器单元的地址。至于其他特殊功能寄存器作为间址寄存器的情况，由于间址寄存器的地址事先已经能够确定，所以这种方式下寻找存储

器单元的地址就仅需要上述两步中的后一个步骤就可以确定指令需要的存储器单元的地址。

3.2.2.4 立即寻址

采用立即寻址方式的指令是双字节的，第一个字节是操作码，第二个字节是立即操作数。因此，这种寻址方式实现起来比直接寻址还要容易，操作数就是放在程序存储器内的常数。

3.2.2.5 基址寄存器加变址寄存器间接寻址

这种寻址方式用于访问程序存储器的一个单元，该单元的地址是基址寄存器（DPTR 或 PC）与变址寄存器 A 的内容之和。虽然这类寻址方式也是间接寻址，对于使用 DPTR 作为基址寄存器的情况，它的实现方法和用数据指针 DPTR 作为间址寄存器时的间接寻址的情况很相似，因为其间址寄存器事实上还是确定的。与用数据指针 DPTR 作为间址寄存器时的间接寻址的情况有所不同的是，确认最终需要的地址还要进行一次加法运算，这在使用 VHDL 语言描述的时候是很容易实现的。用 PC 作为基址寄存器时，需要知道 PC 当前值，但是 PC 和 DPTR 时不同的，DPTR 是特殊功能寄存器，利用它的地址就可以读出其值，PC 并没有被分配地址，不能使用读 RAM 的方式取得其值，所以内部有必要设置编写这样一个进程，它用于读出那些位于 CPU 内部、没有分配地址的辅助寄存器的值[19]。

3.2.3 流水线技术

3.2.3.1 流水线的概念和特性

流水线是一种能够使多条指令重叠操作的微控制器实现技术，它已成为现代微控制器设计中最关键的技术。流水线结构把一条指令的执行分成几个步骤，或称为级。每一级在一个时钟周期内完成。在每个时钟周期，微控制器启动执行一条指令。如果微控制器的流水线有 m 级，则同时可重叠执行的指令总条数将为 m，每条指令处于不同的执行阶段。如果分级分得好的话，那么每一级都没有时间上的浪费。在这种理想情况下，微控制器在采用流水线结构与不采用流水线结构的性能加速比为：

$$S = \frac{I \times CPI_{np} \times T}{I \times CPI_{p} \times T} = \frac{CPI_{np}}{CPI_{p} / m} = m \tag{3-1}$$

其中：I 为一个程序被执行的总的指令条数。在微控制器采用流水线结构与非流水线结构的情况下，它的值是相等的。

CPI 是每条指令总体平均所需的时钟周期数。CPI_{np} 是采用非流水线结构的微控制器的 CPI，CPI_{p} 是采用流水线结构的微控制器的 CPI。因为流水线把指令的执行时间理想地分成了 m 级。有 m 级指令同时执行，则

$$CPI_p = \frac{CPI_{np}}{m} \tag{3-2}$$

若CPI_p为 1，则CPI_{np}，等效为m。

其中T是每个时钟周期的时间长度，假设它在两种结构的微控制器中也是相同的。

最后计算得出的加速比等于流水线的级数。但并不是把流水线级数分得越多，微控制器的性能就越好，这是因为实际的情况要受很多条件的限制。微控制器性能提高的关键在于每个时钟周期都能启动一条指令的执行。这意味着流水线每级中的执行部件要有能力在每个时钟周期接受一条新的指令。例如，如果 ALU 完成一次操作要用 10ns 的话，那么 ALU 接收一条指令的操作周期不能长于 10ns。即每隔 10ns，可以从指令存储器中取出一条指令；或者每隔 10ns 可以完成一次读写数据存储器的操作。

流水线设计是将一个顺序处理过程分解成若干个处理过程。这些子处理过程叫做段或级。每段完成一个具体的功能并产生一个（中间）结果。每级由一个输入锁存器和随后的一个处理电路组成，该处理电路又与下一级的输入锁存器相接。时钟信号可以同时送到各级的输入锁存器。每个时钟脉冲，都促使各级将完成的结果同时送到下一级。各级完成指定的处理任务需要的时间可能不同，但时钟周期 T 的选取就需要保证最慢的哪一级也能将任务完成。

在采用了流水线设计后，单条指令的执行时间并没有被缩短；但每个时钟周期都会有一条指令执行完毕。流水线技术的好处在于它既能把指令间的操作并行性充分发挥出来，同时又不对用户编程提出新的要求。

3.2.3.2 流水线竞争

控制竞争，由程序 PC 指针值的改变引起。当执行跳转指令时，PC 指针值要到执行级才能改变，这将会使下一拍的取指操作出错。这时必须由硬件插入一条空操作 NOP 指令，等待 PC 指针的值改变后再取下一条指令。

数据竞争，由指令间数据相关引起。存储器访问存在先写后读相关（Read after Write），前一条指令的写操作要到回写级才能完成。若紧接的下一条指令需要读取同一地址的内容时，必须使用旁路（Bypassing）技术，从 ALU 的输出结果直接反馈到 ALU 的输入端供下一条指令的执行级使用。

3.2.3.3 三级流水线结构实现

考虑到流水线实现的复杂程度和必要性，在本设计中的微控制器使用三级流水线来实现。第一级为取指令和指令译码周期，第二级为指令执行，第三级为指令写回运算结果。如图 3-4 所示。

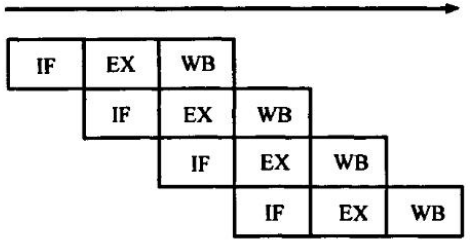


图 3-4 三级流水线示意图

取指级（IF）——从程序存储器中取出一条指令，同时进行指令译码．准备寄存器、存储器的读地址，读 / 写控制信号；

在每个 Q1 拍，即 clkl 的上升沿，程序指针 PC 更新为目标地址，目标地址在正常情况下是当前地址加 1，当跳转时由译码器给出目标地址，当中断时，目标地址是对应的中断向量；在 Q4 拍，即 clk4 的上升沿，把指令从程序存储器中取出，并将指令锁存在临时指令寄存器中。

执行级（EX）——数据输入 ALU 单元运算，同时准备寄存器或存储器的写地址；

指令执行阶段包括译码、取操作数、执行运算和结果写回寄存器，以上所有操作是在下一个 Q1 到 Q4 拍之间完成，并与取指令阶段是并行工作的。

回写级（WB）——将 ALU 输出的运算结果写入寄存器或存储器中。

3.2.4 典型指令举例

下面就几个典型例子说明指令执行过程。

3.2.4.1 单字节单周期指令 XCH A，@R1

这条是典型的单字节单周期实现的指令，涉及到的操作非常多，有两次读 RAM 一次写 RAM，对 ACC 的一次读和写，具体实现过程如下：

Sl：在第一个时钟节拍的上升沿，PC 指针更新为新的目标地址。

由指令临时寄存器驱动总线，译码器从总线读入新的指令，此后整个指令周期不再变化。RAM 模块从总线的指令中分离出地址信息并送入 RAM 的地址中。

S2：在第二个时钟节拍的上升沿，取指令模块给出目标指令的地址；

译码器要给出 RAM 的读使能信号，此时由它们驱动数据总线，RAM 模块仍然从总线的指令中分离出地址信息并送入 RAM 的地址中（用于@Ri 指令时）；同时要读 ACC，将 ACC 中数经 TEMP2 进入 ALU 中。

S3：在第三个时钟节拍的上升沿处，取指令模块给出 ROM 的读信号；

译码器要给出 RAM 的读使能信号（用于@Ri 指令时）由它们驱动数据总线，

送入 ACC。

S4：在第四个时钟节拍的上升沿，取指令模块把指令从 ROM 中取出来并存在指令临时寄存器中；

ALU 将 TEM2 中的数输出驱动总线，要在第四个时钟节拍中间给出 RAM 的写信号。

此条指令标准 51 需要一个机器周期 12 个 clock，本文中只需要一个机器周期 4 个 clock，速度是原来的 3 倍。

3.2.4.2 查表的单字节三周期指令 MOVC A， @A+PC

此指令主要用来查表，涉及到 PC 的操作比较多，完整的执行需要 3 个机器周期。

S1：在第一个时钟节拍的上升沿，PC 指针更新为新的目标地址。

由指令临时寄存器驱动总线，译码器从总线读入新的指令，此后整个指令周期不再变化。

S2：在第二个时钟节拍的上升沿，取指令模块给出目标指令的地址；Buff2(l6)从 PC 读入 PC 值；ACC 中数驱动数据总线，并进入 Buffl (8)中；

S3：在第三个时钟节拍的上升沿处，取指令模块给出 ROM 的读信号；

Add 将 Buffl 与 Buff2 中的值求和，并送到 PC。取指令模块给出 ROM 的读信号，ROM 中的值送入 IR1 中，

S4：在第四个时钟节拍的上升沿，取指令模块把指令从 ROM 中取出来并存在指令临时寄存器中；IR1 驱动总线，并进入 ACC。

此条指令标准 51 需要两个机器周期 24 个 clock，本文中需要两个机器周期就可以取得查表的值，但由于中断了流水线，故需要增加一个机器周期，即花了 12 个 clock，速度是原来的 2 倍。

3.2.4.3 读外部数据存储器的单字节双周期指令 MOVX A，@DPTR

这条是用来读写外部 RAM 的典型指令，需要注意的是外部 RAM 的地址也是靠 PC 模块来送出地址的。

S1：在第一个时钟节拍的上升沿，PC 指针更新为新的目标地址；

由指令临时寄存器驱动总线，译码器从总线读入新的指令，此后整个指令周期不再变化。

S2：在第二个时钟节拍的上升沿，PC 模块输出目标指令的地址；

DPTR 中值通过 PC 模块将 DPTR 中的数据作为地址输出。

S3：在第三个时钟节拍的上升沿处，取指令模块给出 ROM 的读信号；

给出外部 RAM 的读信号。

S4：在第四个时钟节拍的上升沿，取指令模块把指令从 ROM 中取出来并存在寄存器 P0 中；

外部 RAM 中数据进入总线，中间节拍给出 ACC 的锁存信号。

此条指令标准 51 需要两个机器周期 24 个 clock，本文中断了流水线，需要两个机器周期 8 个 clock，速度是原来的 3 倍。

3.3 MCU 的状态描述

MCU 的主要工作过程可以描述为以下几个状态，复位、取指、执行、中断。如图 3-5 所示。首先，在复位信号到来后，进行系统复位，主要来完成各寄存器初始值的设置。在复位完成以后，从指定的地址开始取指、执行。每次执行完毕，都需要进行中断判断，如果中断有效就进入中断状态，主要任务是将重要的寄存器值保存在堆栈中，以便中断返回后恢复。

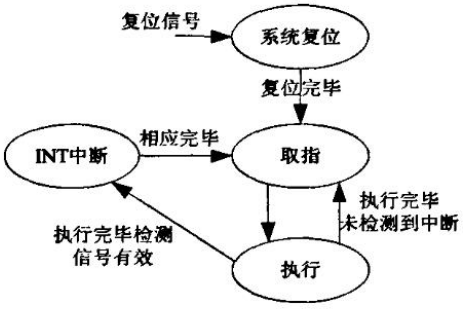


图 3-5 MCU 工作状态

下面对各个状态进行详细的描述。

（1）系统复位

首先，系统加电，时钟稳定以后，RESET 信号由低变高。系统进入复位状态，将特殊功能寄存器和程序计数器初始化，但不影响片内 RAM 中存放的内容。然后处理器从 PCH，PCL 所指的地址开始运行。

（2）指令取指及执行状态

微处理器的工作是一个周而复始的取指令、分析指令、执行指令的过程。对于 MCU 来说，所有的指令操作都可以分解成为一系列的状态，如下图 3-6 所示。

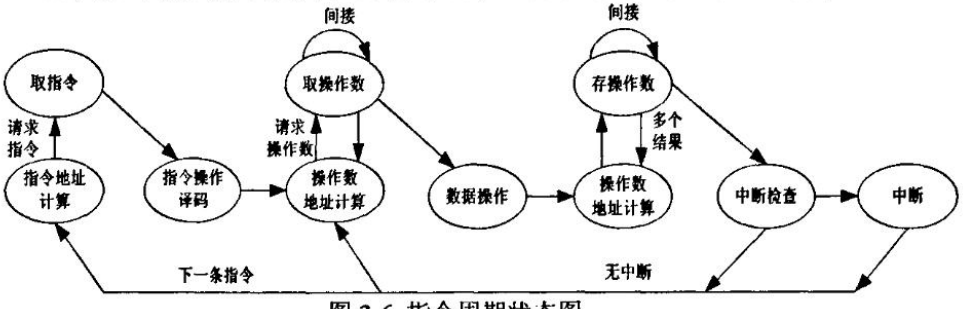


图 3-6 指令周期状态图

这些状态不一定都要执行。对某些指令来说，可能不执行图中的某些状态，而有些指令可能会重复执行某些状态。

下面对这些状态进行说明：

PC 值计算：计算下条指令的存放的地址。一般来说，都是前一条指令的地址加一。遇到跳转指令和中断指令时，则可能会有所不同，需要判断和计算。

（1）取指：从存储器 ROM 读入指令操作码，送入指令寄存器 IR。

（2）译码：根据指令操作码，决定所要执行的操作类型以及所使用的操作数。具体的实现是根据指令操作码来决定指令节拍码的顺序和内容。

（3）寻址：当执行的指令需要复杂寻址时，首先计算出操作数的具体地址。

（4）取数：从存储单元或者其他部件中读取操作数（非 ROM ）。

（5）数据操作：对数据完成运算或操作。

（6）回写：将结果写入存储单元或指定的寄存器。

（7）检测中断：每次指令执行完毕都要检查中断信号，检查是否需要执行中断。

（8）执行中断：当中断有效后，进入中断处理过程。

一般来说，指令完成的时钟周期不同，每个周期执行的操作也不相同。经过对不同指令的分析和总结，可以发现指令的执行过程可以分解成一系列的不同类型的微状态，在不同的状态下，完成不同的操作。通过一系列状态的顺序执行，就能够完成不同指令的功能了。由于各个指令执行过程复杂，在这里就只以寄存器间接寻址 MOV A，@Ri 为例，简要说明指令的操作过程和执行状态的转移。

寄存器间接寻址 MOV A，@Ri，首先 PC 值更新后，在取指状态，程序存储器在读指令控制下将指令送往指令寄存器；译码后，从 Ri 中读取数据即间接地址，送往数据存储器的地址译码器；下一状态再将此间接地址中的数读出即为所需的数，最后送入 ACC 中。如图 3-7 所示的指令的执行状态转移过程。

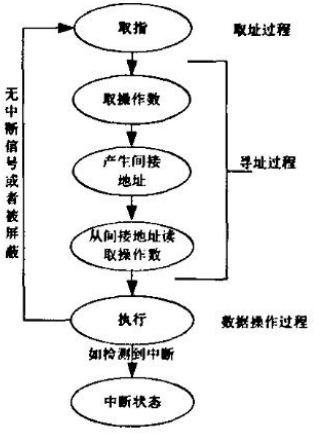


图 3-7 间接寻址的状态转移过程

（3）中断过程

从执行程序流程角度来看，所谓中断就是一种可以临时中断微处理器正常程序执行并引发另一个程序开始执行的特殊信号。在指令执行完毕的最后一个状态，判断中断标志决定是否进入中断状态。如果中断有效，系统开始进行中断操作，首先要保护现场，把下一条指令地址压入堆栈进行保存（若要保护 ACC 和标志寄存器等需用户在中断程序里进行压栈保存），再把存储器中的中断向量读入 16 位地址送到程序计数器，然后就可以开始运行中断服务程序了。

当中断返回时，执行一条 RETI 指令，系统会恢复现场，把堆栈中的 PC 值数据弹出来继续执行中断前的下一条指令，被中断的程序就像没有发生中断那样继续执行。中断处理和返回的状态图见图 3-8 和图 3-9 所示。

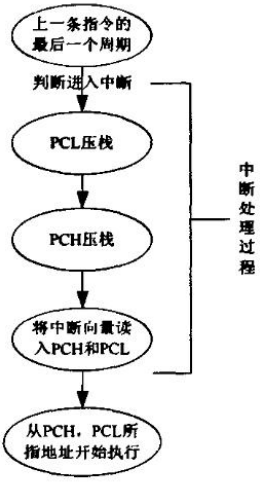


图 3-8 中断处理过程的状态图

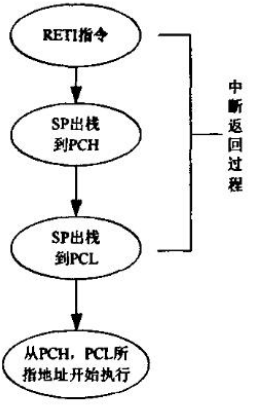


图 3-9 中断返回过程的状态图

3.4 MCU 核组成

3.4.1 MCU 总体设计

3.4.1.1 MCU 主要特性

本文设计的 MCU 具有如下特性：

（1）全同步设计 ——同步于时钟信号；

（2）指令集完全兼容工业界标准 8051 微处理器；

（3）优化结构——执行每条指令只需 1－4 个时钟周期内；

（4）I/O 口独立设置，适合综合、仿真，接口简单；

（5）256 位内部 RAM，可达 64K ROM，64K 外 RAM；

（6）具有 CAN 总线接口，适合汽车工业应用。

其端口定义为：

CLK：系统时钟输入，在时钟的上升沿有效；

RESET：全局复位输入，采用异步复位方式；

T0_I：Timer/Counter0 输入；

T1_I：Timer/Counterl 输入；

RXD_I：串行口数据接收端；

INT0_I：外部中断 0 输入；

INTI_I：外部中断 1 输入；

P0_I：P0 并行口输入；

PI_I：P1 并行口输入；

P2_I：P2 并行口输入；

P3_I：P3 并行口输入；

RXDWR_O：串口 RXD 输入/输出指示，'1'表示 RXD_O 数据有效，'0'表示 RXD_I 数据有效；

TXD_O：串行口数据发送端；

RX_O：串口模式。数据输出端；

P0_O：P0 并行口输出；

Pl_O：P0 并行口输出；

P2_O：P0 并行口输出；

P3_O：P0 并行口输出。

关键部分说明：

CLK：由于该 8051 IP 采用了全局同步设计，因此只需要一个时钟输入就可以为每个存储元素提供时钟沿控制。在设计中，并没有使用门控时钟。对于外部中断输入，由于它们可能是由外部不同的时钟源控制，所以在进入中断端口后，由 8051 IP 的全局时钟使用两级寄存器进行同步化。

并行 IO 口：该 8051 IP 提供了与 Intel 8051 相同的 4 个 8 位的 I/O 并行口。但为了方便进行 IC 集成，该 IP 没有采用 I/O 复用的方式，而是将每个端口都是设计为单向的，这样的设计符合了 FPGA 设计中慎用三态的原则。

存储器接口：为了优化 IP 的结构，该 IP 模块的信号在与存储器接口连接时没有使用寄存器输入输出，因而在综合时需要进行相关端口的时间约束。

此外，该 IP 还可以通过修改其参数来扩展多个定时/计数器、串口通信、外部中断等，以实现不同的应用需求。

3.4.1.2 设计层次

本设计应用 VHDL 语言设计一个 8051 兼容的 MCU 核，设计层次和相关的 VHDL

文件如图 3-10 所示，微控制器核是由定时器/计数器、逻辑运算单元、串口和控制单元四个大部分组成的。并配合 RAM、ROM 及其 RAMX 组成整个处理器整体。

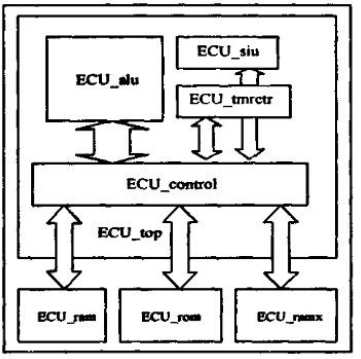


图 3-10 MCU 核顶层设计

本设计的详细组成模块如图 3-11 所示，顶层是用于程序测试的测试文件（Testbench Files），测试包括外部 RAM、内部 RAM、ROM 和处理器核。处理器核是设计的核心部件，可以实现数据的存储、处理，指令解码、执行等操作。具体包括定时/计数器、串行口、算术逻辑单元和控制单元。控制单元又包括存储单元和无限状态机（FSM）两部分，用于实现指令译码功能。算术逻辑单元（ALU）用于实现算术运算（加、减）和逻辑运算（与、或、非）的单元，主要包括 ALU 多路选择器、加减核以及 ALU 核三部分组成。下面各节将会详细介绍各个组成部分及其功能。

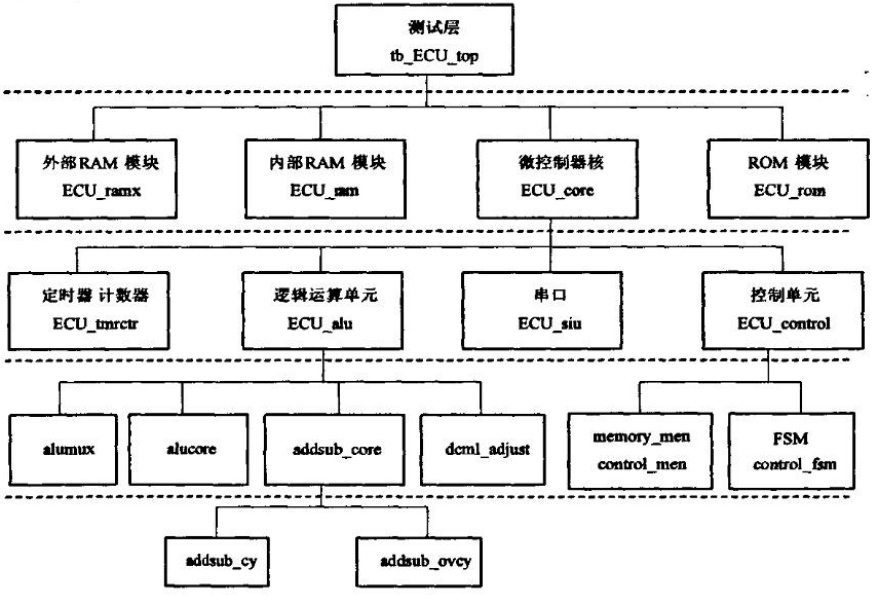


图 3-11 微控制器 IP 核的各个模块

3.4.2 算术逻辑单元（ALU）

ALU 是 MCU 的核心部分，它是完成数据的算术运算、逻辑运算的功能单元，是一种功能较强的组合逻辑电路单元。事实上，系统中的其他部分，比如控制器、寄存器、存储器、如接口等，都是为 ALU 准备数据和传输运算结果而设置的辅助单元。MCU 工作时，数据由寄存器送入 ALU 中，在控制信号的作用下 ALU 进行运算，并将结果存储到指定的寄存器中，同时根据运算的结果设置相关的标志位。控制寄存器为 ALU 提供两类控制信号，一类控制信号保证数据在 ALU 和寄存器之间正确的移动，另一类信号保证 ALU 进行正确的运算。

ALU 模块是由算术运算和逻辑运算电路组成，几乎所有的指令都要在 ALU 中执行。ALL 包含了各种算术运算和逻辑运算功能，包括加、减、乘、除、逻辑运算、移位运算和位操作等。接收从 ROM、RAM 传来的数据及其相应的指令，产生新的进位、溢出标志位和结果，从而实现算术逻辑功能。

ALU 分别由加减法核、ALU 核、十进制调整及其 ALU 多路选择四部分组成，如图 3-12 所示。

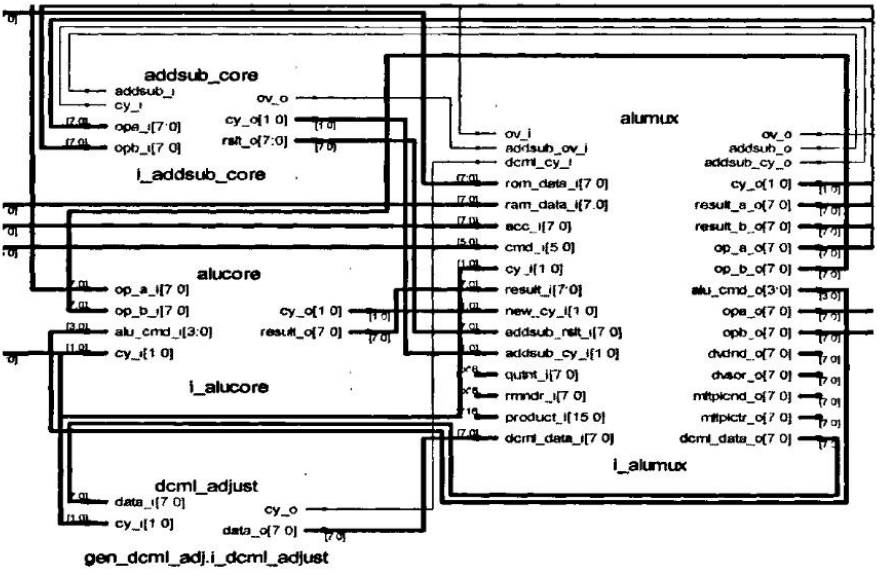


图 3-12 算术逻辑单元（ALU）核的组成

ALU 模块支持以下几种运算操作：

NOP——空操作

ADD——加法

SUB——减法

DA——十进制调整

INV——逻辑非（操作数为字节位）

LAND——逻辑与（操作数为字节位）

LOR——逻辑或（操作数为字节位）

LXOR——逻辑异或（操作数为字节位）

RL——循环左移

RLC——带进位循环左移

RR——循环右移

RRC——带进位循环右移

COMP——位比较

3.4.2.1 加减法运算

加减法运算单元由两部分组成，如图 3-13 所示，一部分是加减运算单元（addsub_cy），另一部分负责产生进位、溢出标志位（addsub_ovcy），分别对操作数 A、B 进行 addsub_i 表示的加减运算。可以实现 ADD、SUB、 ADC、SUBB、INC、DEC 等指令。

```
if addsub_i = '1' then                    -- 加法运算
   v_a(0) := '1';
   v_b(0) := cy_i;
   v_result := conv_unsigned(v_a,DWIDTH+2) + v_b;
else                                      --减法运算
   v_a(0) := '0';
   v_b(0) := cy_i;
   v_result := conv_unsigned(v_a,DWIDTH+2) - v_b;
end if;
cy_o <= v_result(DWIDTH+1);               -- 进位标示
rslt_o <= v_result(DWIDTH downto 1);      --产生运算结果
```

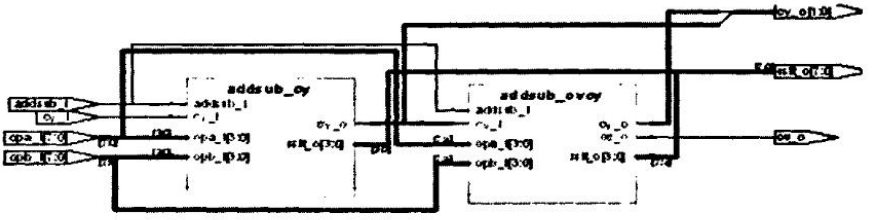


图 3-13 加减法单元

3.4.2.2 十进制调整运算

在单片机中对于二进制数的运算，可以直接使用算术运算指令，因为算术运算功能的硬件实现都是基于二进制数的；当对于十进制数的运算，只使用算术运算指令是不够的，必须配合使用一套运算调整指令，才能得到正确的运算结果。十进制调整运算就是实现此功能的模块。

3.4.2.3 ALU 数据选择模块（ALUMAX）

ALUMAX 是根据不同的命令选择不同的数据通道，依据从 cmd_i 中得到的指令分别从 RAM、ROM、ALU 中取得数据并送到相应的输出端口中。支持的指令包括 DA、ADD、SUB、OR、RL 等等。

3.4.3 中心控制器（ECU_Control）

中心控制器是 ECU 处理器的核心器件，接收 ALU、SIU、存储器等外围模块传来的数据，并依据相应的命令对数据进行处理，并把产生的结果传给相应的输出端口进行输出。控制器的输入信号包括指令寄存器存储的指令代码，状态寄存器的标志位，外部模块输入的控制总线的控制信号（比如中断请求信号）。产生的输出信号，包括各个寄存器的控制信号，总线的控制信号，ALU 的控制信号以及外部存储器的读写控制信号等等。

控制器主要由中断控制电路，时钟产生电路，指令节拍计数器，指令寄存器、控制信号发生器、标志寄存器这几部分组成，如图 3-14 所示。

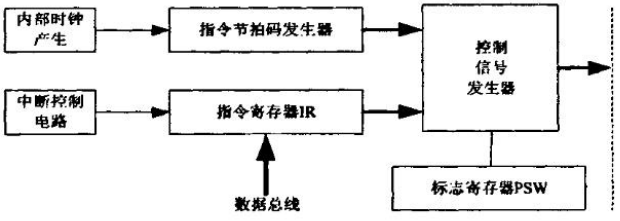


图 3-14 控制器的结构图

中断控制电路实现对中断信号的采样，以及是否屏蔽中断的判断。产生的控制信号，控制指令寄存器的清零，来决定是否进行中断操作。

指令寄存器 IR 用于保存当前正在执行的指令操作码。每一条指令完成以后，根据指令节拍码产生的控制信号加载从数据总线送来的 8 位操作码，或者受中断控制电路发来的控制信号进行清零操作。

状态寄存器是 7 位的寄存器，这个寄存器中的位的值的修改依赖于数据通道的影响，也可以由控制信号发生器产生的相关控制信号来改变状态。同时它又指示了

程序运行的结果，控制影响数据通道的工作。

指令节拍码产生电路，完成根据指令寄存器的指令操作码，确定指令执行所需的周期数，以及产生指令节拍代码。产生的指令节拍码送到控制信号产生电路，和指令操作码、状态寄存器的状态标志一起来产生控制信号，来决定本周期内完成什么操作。

中心控制器主要由状态机（control_fsm）和控制存储器 （control_mem）两部分组成。如图 3-15 所示。

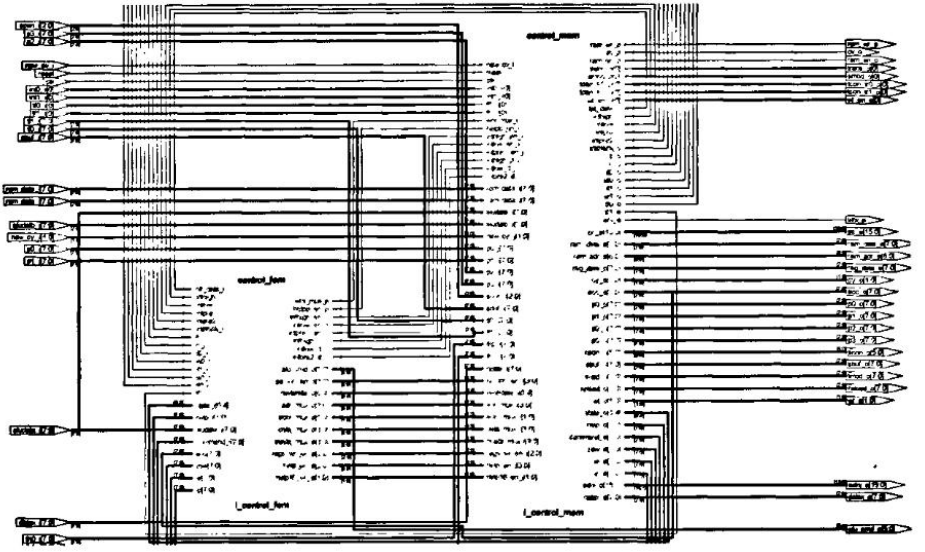


图 3-15 中心控制器组成图

3.4.3.1 控制状态机（control_fsm）

状态机主要是用于对指令进行解码并执行的模块。通过状态机来实现对 MCU 状态转移控制，如前面提到流水线处理，这里把指令处理过程分为：FETCH、EXEC1、EXEC2、EXEC3 三级流水线机构。

```
if state=FETCH then              --处于取指状态
  s_intpre2_en <= '1';
      ...
  s_nextstate <= EXEC1;          --下状态 执行 1
elsif state=EXEC1 then
  if (PX0 and EX0 and s_ie0)='1' then
    s_help_en <= "0101";
  else
```

```
        ...
        end if;
        s_nextstate <= EXEC2;
      elsif state=EXEC2 then
        s_adr_mux <= "0101";
        ....
        s_nextstate <= EXEC3;
      elsif state=EXEC3 then
    s_nextstate <= FETCH;
      else
        s_nextstate <= FETCH;
      end if;
```

3.4.3.2 控制存储器 （control_mem）

控制存储器是用于从特殊功能寄存器或中断源中读取数据并写入到相应的一些寄存器中，或者从可位读写的存储器地址读写数据信息。

分为以下几个处理进程：

（1）读特殊功能寄存器（SFR），RAM 的可位读写区；

（2）写中断源的上升/下降沿的检测信号寄存器；

（3）读写寄存器的多路选择器；

（4）写内部寄存器，用户不能直接访问的寄存器；

（5）计算下一个 PC 的地址；

（6）写 RAM、SFR 或者可位读写寄存器。

3.4.4 定时/计数器

本设计具有 2 个定时器/计数器，即定时器/计数器 0 和 1。在专用寄存器 TMOD（定时器方式）中，各有一个控制位（C/T），分别用于控制定时器/计数器 0 和 1 是工作在定时器方式还是计数方式。

选择定时器工作方式时，计数输入信号是内部时钟脉冲，每个机器周期使寄存器的值增 1。每个机器周期等于 12 个振荡器周期，故计数速率为振荡器频率的 1/12。

选择计数器工作方式队时，计数脉冲来自相应的外部输入引脚 T0 或 Tl。当输入信号产生由 1 到 0 的负跳变时，计数寄存器（TH0、TL0 或 THl、TLI）的值增 1。每个机器周期的 S5P2 期间，对外部输入进行采样。如果在第一个周期中采得的值为 1，而在下一个周期采得的值为 0，则在紧跟着的再下一个周期的 S3P1 期间，计数值就增 1。由于确认一次下跳变要花 2 个机器周期，即 24 个振荡器周期，因此外

部输入的计数脉冲的最高频率为振荡器频率的 1/24。对外部输入信号的占空比并没有什么限制，但为了确保某一给定的电平在变化之前至少被采样一次，则这一电平至少要保持一个机器周期。

由以上对定时器/计数器功能和原理的一个整体性的描述可知，本单元将必须以时序逻辑电路的方式来实现。图 3-16 是本模块设计完成后形成的符号文件。图中的输入除时钟 CLK、复位 RESET、外部中断 0 INT0_I和外部中断 1 INT1_I 直接来自芯片的外部输入之外，其余的输入信号全部来自控制器。其中的 RELOAD_I[7 to 0]、WT_ EN 以及 WT_I[7 to 0]用于重写定时器/计数器内部寄存器。所有的输出信号全部送往控制器，用于及时更新特殊功能寄存器的内容。

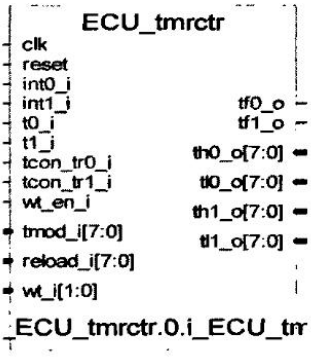


图 3-16 定时/计数器符号图

定时器/计数器的两种工作方式决定了要设计好定时器/计数器首先必须设计一个分频器和两个（因为有两个定时器/计数器）负跳变的检测器。因此本设计实体的结构体将由 3 个进程来实现。除了分频器和两个负跳变的检测器各占用一个进程外，第三个进程用于实现时器/计数器的四种工作模式。各个进程之间相互关联，内部信号将作为进程之间传输信息的纽带。

下面的代码就是设计的 12 分频器的进程，信号 s_count_enable 的频率为输入时钟频率的 1/12，其高电平维持时间与一个输入时钟的周期相等。

```
s_count_enable <= '1' when s_pre_count = conv_unsigned(11,4) else '0';
  p_divide_clk: process (clk, reset)
  begin
    if reset = '1' then                    --复位
      s_pre_count <= conv_unsigned(0,4);
    else
      if clk'event and clk='1' then
   if (s_pre_count = conv_unsigned(11,4)) then --满 12 时 清零
```

```
            s_pre_count <= conv_unsigned(0,4);
        else
          s_pre_count <= s_pre_count + conv_unsigned(1,1);--不满时加 1
        end if;
          end if;
        end if;
    end process p_divide_clk;
```

负跳变的检测器，以一个双稳态的触发器来实现，实现此触发器的进程如下：

```
s_ext_edge0 <= '1' when (s_t0ff1 = '0' and s_t0ff2 = '1') else '0';
    p_sample_t0: process (clk, reset)
      begin
        if reset = '1' then                    --复位
          s_t0ff0 <= '0';
          s_t0ff1 <= '0';
          s_t0ff2 <= '0';
        else
          if clk'event and clk = '1' then
            if s_pre_count = conv_unsigned(6,3) then
              if s_c_t0 = '1' then
                s_t0ff0 <= t0_i;
                s_t0ff1 <= s_t0ff0;
                s_t0ff2 <= s_t0ff1;
              end if;
            end if;
          end if;
        end if;
    end process p_sample_t0;
```

3.4.4.1 模式 0

在模式 0 环境下，16 位寄存器（TH1+TL1）只用了 13 位，TL1 高 3 位未用。C/T 是专用寄存器 TMOD 中的控制位。当 C/T=0 时，选择定时器方式，C/T =1 时选择计数器方式。引脚 T1 接外部输入信号。TR1 是专用寄存器 TCON 中的一个控制位，GATE 是 TMOD 中的另一个控制位，引脚 INT1 是外部中断 1 的输端，在此另有它用。TF1 是定时器溢出标志。当满足条件（TR1=1）AND （GATE=0 OR INT1 =1）为真时，接通计数输入。当计数值由全 1 再增 1 变为全 D 时，使 TF1 置 1，请求中

断。若 TRI=1 和 GATE=1，则（TH1+TL1）是否计数取决于 INT1 引脚的信号，当 INT1 引脚由 0 变 1 时，开始计数，当 INT1 山 1 变 0 时，停止计数。

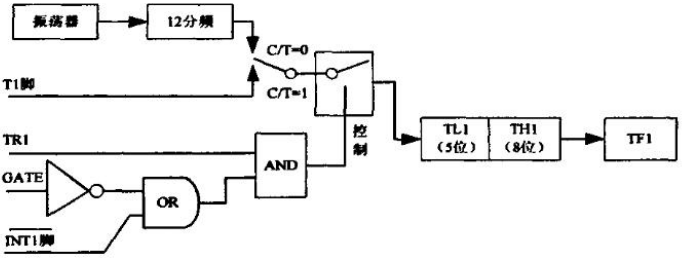


图 3-17 定时器/计数器 1 操作模式 0

3.4.4.2 模式 1

模式 1 和模式 0 几乎完全相同，唯一的差别是：模式 1 中，定时器寄存器（TH+TL）是以全 16 位参与操作的。

3.4.4.3 模式 2

模式 2 把定时器/计数器寄存器 TL1（或 TL0）配置成一个可以自动重装载的 8 位计数器，如图 3-18 所示。TL1 计数溢出时，不仅使溢出标志 TFl 置 1，而且还自动把 TH1 中内容装载到 TLl 中。TH1 的内容可以靠软件顶置，重新装载后其内容不变。由于模式 2 与模式 0 到的控制结构是一样的，所以其电路的实现方式也是相同的。由于其是 8 位的计数方式，所以实现起来更简便—用于写定时器/计数器的高 8 位计数寄存器 TH 的部分将不在如模式 0 中那样复杂。

只需简单根据控制器的指令将特殊功能寄存器中的 TH 的值写入即可。至于自动重新装载计数值，只要在 8 位计数器的值为 255 时，其下一个计数值不为 0 而是将 TH 中的值装入即可实现。

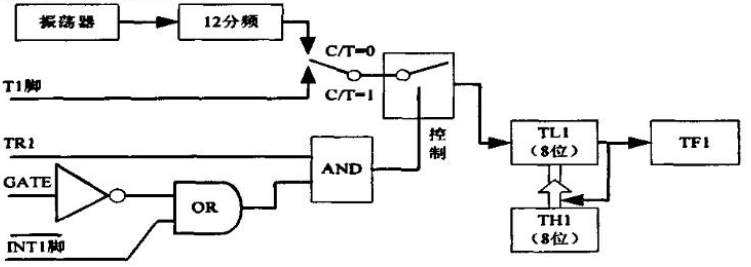


图 3-18 13 位计数器模式 2：8 位自动装载

3.4.4.4 模式 3

操作模式 3 对于定时器/计数器 0 和定时器/计数器 1 是大不相同的。

对于定时器/计数器 0，设置为模式 3，将使 TL0 和 TH0 成为两个相互独立的 8 位计数器，如图 3-19 所示。其中 TL0 利用了定时器/计数器 0 本身的一些控制位：

C/T、 GATE、 TR0、INT0 和 TF0。它的操作情况与模式 0 和模式 1 类似。但 TH0 被规定只用作定时器，对机器周期计数，它借用定时器/计数器 1 的控制位 TR1 和 TF1，故，此时 TH0 控制了定时器 1 的中断。

由上述说明以及图 3-19 可知，定时器/计一数器 0 工作于模式 3 的这部分电路的设计应该分成两个部分：一个 8 位的定时器/计数器和一个 8 位的定时器。8 位的定时器/计数器的设计方法与 13 位（或 16 位）定时器/计数器的设计方法一样，在此不加详述。至于 8 位定时器的设计，可以参考模式 2 中的设计方法，需要注意的是它借用的是定时器/计数器 1 的控制位和溢出标志位。

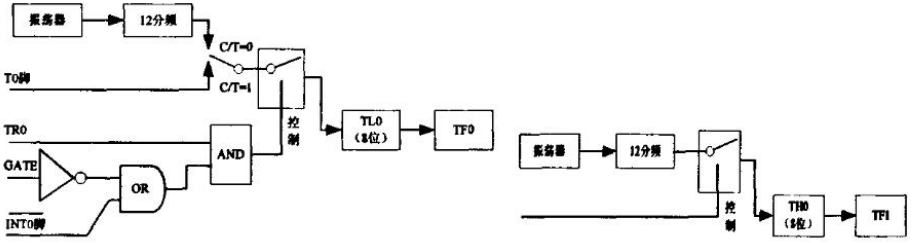


图 3-19 定时器/计数器 0 模式 3：2 个 8 位计数器

3.4.5 串行接口单元（SIU）

本设计串行接口是一个全双工通信接口，即能同时进行发送和接收。它可以作 DART（通用异步接收和发送器）用，也可以作同步移位寄存器用。

串行口缓冲寄存器 SBUF 是可直接寻址的专用寄存器。在物理上，它对应着两个寄存器，一个发送寄存器，一个接收寄存器。CPU 写 SBUF，就是修改发送寄存器；读 SBUF，就是读接收寄存器。接收器是双缓冲的，以避免在接收下一帧数据之前 CPU 未能及时响应接收器的中断，没有把上一帧数据读走，而产生两帧数据重叠的问题。对于发送器，为了保持最大的传输速率，一般不需要双缓冲，因为发送时 CPU 是主动的，不会产生写重叠的问题。

串行口控制寄存器 SCON 用于控制和监视串行口的工作状态。它的各位的功能说明如下：

SM0（SCON.7）和 SMl（SCON.6 ）：串行口模式选择位。

SM2（SCON.5）：在模式 2 和模式 3 中多处理机通信使能位。在模式 2 和模式 3 中，若 SM2=1，且接收到的第九位数据（RB8）是 0，则接收中断标志位（RI）不会被激活。在模式 1 中，若 SM2=1 且没有接收到有效的停止位，则接收中断标志位（RI）不会被激活。在模式 0 中，SM2 必须为 0。

REN（SCON.4 ）：允许接收位。由软件置位或者清除。REN=1，允许接收，REN=0，禁止接收。

TB8（SCON.3 ）：待发送的第 9 位数据

RB8（SCON.2 ）：接受到的第 9 位数据

TI（SCON.1）和 RI（SCON.1）：发送和接收中断标志位（注：必须靠软件清零）。

另外功耗控制寄存器 PCON 的最高位 D7 为波特率系数选择位（SMOD）。

图 3-20 就是本模块设计完成后形成的符号文件。由于本设计不支持 I/0 口的复用，所以为串行口工作于模式 0 下增加了 RXD_O 和 RXDWR_O 两个输出引脚，前者用于输出，后者为输出有效控制位（RXDWR 为高电平时，指示输出）。

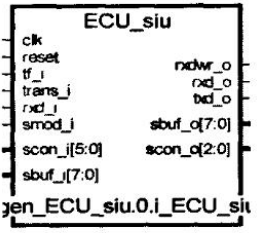


图 3-20 SIU 模块符号图

至于接收和发送，这里采用有限状态机的方式来实现。由于要工作在全双工模式下，所以接收和发送要采用两个有限状态机。有限状态机的每一个状态用来发送/接收移位数据。发送和接收的具体实现方式是移位。移位的时钟来自上面所述的分频器的，也就是根据各自的波特率进行

3.4.6 内部 RAM、ROM

本文设计的的是 MCU 核是基于 Altera 公司的 Stratix 系列 FPGA 仿真、验证的，所以 RAM、ROM 就应用 Altera 公司提供的参数化模块（LPM）器件，改变参数以适合本设计应用。

3.4.6.1 ROM 设计

ROM 是用于存储应用程序的存储器，具有 13 位地址输入，8192 位深度，8 位数据输出。

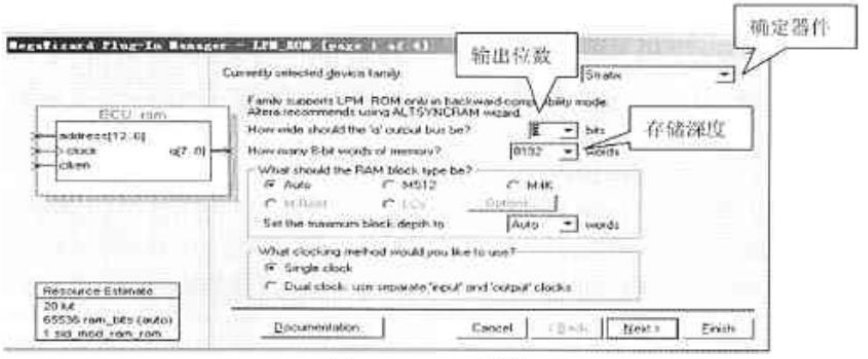


图 3-21 ROM 配置一

在此设计中增加了时钟允许信号，这样所有寄存器都由允许信号统一控制读写。

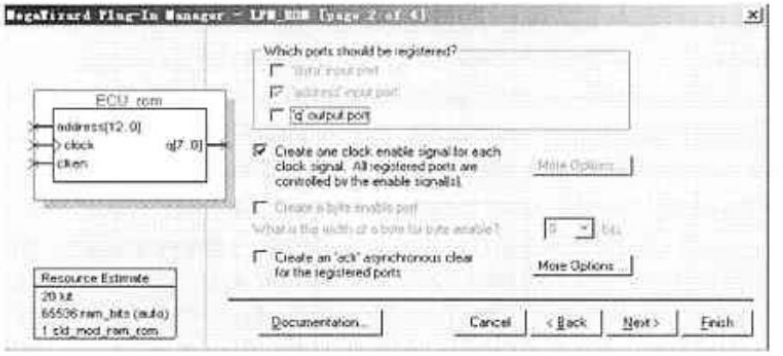


图 3-21 ROM 配置二

指定 ROM 里的初始程序，本设计在 ROM 的 0000H 地址里装载了引导程序，可以实现与上位机通信，并把用户程序引导到程序 RAM（PRAM）中，并开始在 2000H 地址开始执行用户程序。具体执行过程在第五章 FPGA 验证部分会详细介绍。

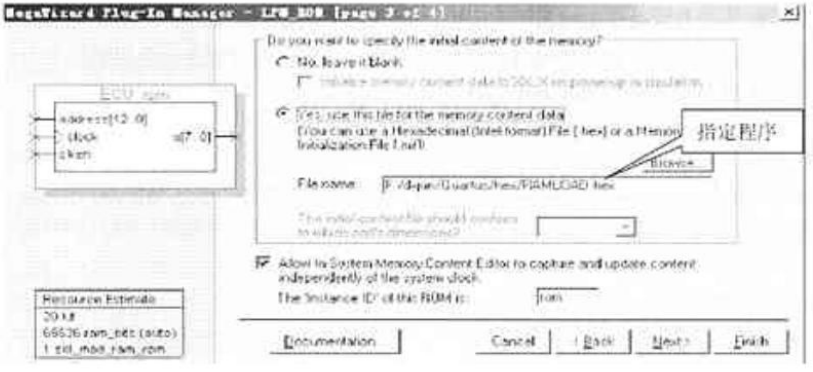


图 3-21 ROM 配置三

3.4.6.2 RAM 设计

本设计 RAM 包括包括三个，分别是内部 RAM、外部 RAM（RAMX）、和用于存储用户程序的程序 RAM（PRAM），其参数设计过程与 ROM 大致相同，在此就不在赘述了。各自特性如下：

内部 RAM 具有 8 位数据输入、8 位数据输出，7 位地址寻址 128 字节深度。

外部 RAM 具有 8 位数据输入、8 位数据输出，13 位地址寻址 8192 字节深度。

PRAM 采用简单双端口模式，具有 8 位数据输入、8 位数据输出，13 位读/写地址。

第四章 CAN总线单元的设计

4.1 控制器局域网CAN

4.1.1 CAN总线

CAN（Controller Area Network）即控制器局域网，是国际上应用最广泛的现场总线之一。最初，CAN被设计用于汽车环境中的微控制器通信，在车载各电子控制单元之间交换信息，形成汽车电子控制网络，如发动机管理系统、变速箱控制器、仪表装备等均嵌入了CAN控制装置[21]。

一个由CAN总线构成的单一网络中，理论上可以挂接无数个节点，实际应用中，节点数目受网络硬件的电气特性所限制。例如，当使用Philips P82C250作为CAN收发器时，同一网络中允许挂接110个节点。CAN可提供高达1Mbit/s的数据传输速率，可以实现实时控制。

4.1.2 CAN的发展

控制器局域网CAN是80年代初博世公司为解决现代汽车中众多控制单元、测试仪器之间的实时数据交换而开发的一种串行通信协议，经多次修订，于1991年9月形成技术规范2.0。该版本包括2.0A和2.0B两部分。其中2.0A给出了报文标准格式，2.0B给出了报文的标准和扩展两种格式。

随着车用电气设备越来越多，为了解决现代汽车内部用少量的线束实现大量的控制测控仪器、微处理器、传感器和执行机构之间的数据交换问题，基于CAN协议的分布式控制网络逐渐在现代汽车上广泛采用。CAN协议的最大特点是废除了传统的站地址编码，而代之以对通信数据块进行编码。理论上站点个数不受限制。数据块的标识码可由11位或29位二进制数组成。采用CRC检验，并可提供相应的错误处理功能、数据通讯可靠性高。这将是电动汽车中数据通讯中最适合的现场总线之一。为了解决现代汽车内部用少量的线束实现大量的控制测控仪器、微处理器、传感器和执行机构之间的数据交换问题，推动汽车电子化、电脑化、电信三电一体化。

基于CAN协议的分布式控制网络，逐渐在现代汽车上广泛采用。目前在汽车设计领域，CAN几乎成为一种必须采用的技术手段，尤其在欧洲，如奔驰、宝马、大众、沃尔沃。美国汽车厂也将控制器联网系统逐步由class2过渡到CAN。据Strategy Analysis市场研究公司公布的一份对为控制器和汽车网络所作的研究表明，大多数客车厂都选用基于CAN的网络。到2007年，CAN将会占据整个汽车网络协议市场的73%，在欧洲大约为88%。对机动车辆总线和对现场总线的要求有许多相似之处，

即较低的成本、较高的实时处理能力和在恶劣的强电磁干扰环境下可靠的工作。奔驰S型轿车上采用的就是CAN总线系统；美国商用车辆制造商们也将注意力转向CAN总线[22]。

4.1.3 CAN 总线的通信协议和特点

CAN总线有标识符长度为11位的标准1.1和长度可以为29位的扩展2.0两个版本。在2.0B版本中规定，CAN控制器的标识符长度可以是11位或29位，遵循CAN2.0B协议的CAN控制器可以发送和接收11位标识符的标准格式报文或29位标识符的扩展格式报文。

4.1.3.1 通信协议 CAN支持四类信息帧类型

CAN通信协议主要描述设备之间的信息传递的方式，其定义与开放系统互连模型（OSI）一致，能够实现每一层与另一设备上相同的那一层之间的通信。实际的通信发生在每一设备上相邻的两层，设备间只通过物理层的物理介质互连。CAN通信协议的规范定义了模型的数据链路层和物理层两层。应用层协议可以由CAN用户定义成适合特别工业领域的任何方案。在汽车工业里，许多制造商都应用自己的标准。

CAN能够使用多种物理介质，例如双绞线、光纤等。最常用的就是双绞线，信号使用差分电压传送，两条信号线被称为“CAN_H”和“CAN_L”，静态时均是2.5V左右，此时状态表示为逻辑“1”，也叫做“隐性”；动态时CAN_H和CAN_L的电压分别为3.5V和1.5V，此时状态表示为逻辑“0”，也称为“显性”。[23] [24]

（1）数据帧 CAN协议有两种数据帧类型标准2.0A和标准2.0B。两者本质的不同在于ID的长度不同。在2.0A类型中，ID的长度为11位；在2.0B类型中ID为29位。一个信息震中包括7个主要的域：

帧起始域——标志数据帧的开始，由一个显性位组成。

仲裁域——内容由标示符和远程传输请求位（RTR）组成，RTR用以表明此信息帧是数据帧还是不包含任何数据的远地请求帧。当2.0A的数据帧和2.0B的数据帧必须在同一条总线上传输时，首先判断其优先权，如果ID相同，则非扩展数据帧的优先权高于扩展数据帧。

控制域——R0、R1是保留位，作为扩展位，DLC表示一帧中数据字节的数目。

数据域——包含0～8字节的数据。

校验域——检验位错用的循环冗余校验域，共15位。

应答域——包括应答位和应答分隔符。正确接收到有效报文的接收站在应答期间将总线值为显性电平。

帧结束——由七位隐性电平组成。

（2）远程帧 接受数据的节点可通过发远程帧请求源节点发送数据。它由 6 个域组成：帧起始、仲裁域、控制域、校验域、应答域、帧结束。

（3）错误指示帧 由错误标志和错误分界两个域组成。接收节点发现总线上的报文有误时，将自动发出“活动错误标志”其他节点检测到活动错误标志后发送“错误认可标志”。

（4）超载帧 由超载标志和超载分隔符组成。超载帧只能在一个帧结束后开始。当接收方接收下一帧之前，需要过多的时间处理当前的数据，或在帧间空隙域检测到显性电平时，则导致发送超载帧。

（5）帧间空隙 位于数据帧和远地帧与前面的信息帧之间，由帧间空隙和总线空闲状态组成。帧间空隙是必要的，在此期间 CAN 不进行新的帧发送，为的是 CAN 控制器在下次信息传递前有时间进行内部处理操作。当总线空闲时 CAN 控制器方可发送数据。

4.1.3.2 CAN 总线的特点

CAN 具有如下特点：

（1）低成本，总线利用率高；

（2）数据传输距离远（长达 10Km）；

（3）数据传输速率高（高达 1Mbit/s）；

（4）可根据报文的 ID，决定接收或屏蔽报文；

（5）可靠的错误处理和检错机制，发送的信息遭到破坏后，可自动重发；

（6）节点在错误严重的情况下具有自动退出总线的功能；

（7）报文不包含源地址和目标地址，用标识符指示功能和优先级信息。

4.1.3.3 CAN 总线技术的应用

（1）CAN 网络上任何一节点均可作为主结点主动地与其他节点交换数据，大大提高系统的性能。

（2）CAN 网络节点的信息帧可分出优先级，且单帧字节长度短，有很好的实时性。

（3）CAN 的物理层及数据链路层采用独特的设计技术，使其在抗干扰，错误监测能力等方面的性能均超过其他总线。

（4）CAN 的通信速率相当高。当网络线的长度不超过 40 米时，其通信速率可达 1Mbit/s。

（5）CAN 总线每帧数据都含有 CRC 校验及其他校验措施，数据出错率低。

（6）CAN 总线节点在严重错误的情况下，可自动切断与总线的通信联系，以使总线上的其他操作不受影响[25]。

4.2 CAN 总线 IP 核的实现

CAN BUS 分为标准 CAN（BasicCAN）和扩展格式 CAN（PeliCAN）两种。标准 CAN 的标志符长度是 11 位，而扩展格式 CAN 的标志符长度可达 29 位。CAN 协议的 2.0A 版本规定 CAN 控制器必须有一个 11 位的标志符。同时，在 2.0B 版本中规定，CAN 控制器的标志符长度可以是 11 位或 29 位。遵循 CAN2.0B 协议的 CAN 控制器可以发送和接收 11 位标识符的标准格式报文或 29 位标识符的扩展格式报文。如果禁止 CAN2.0B，则 CAN 控制器只能发送和接收 11 位标识符的标准格式报文，而忽略扩展格式的报文结构，但不会出现错误。

考虑到设计的复杂性和处理速度等方面，本设计采用了标准 CAN 模式设计。采用自顶而下的设计方法，顶层如图 4-1 所示，分别包括位时序逻辑、位处理逻辑和寄存器三大部分组成。

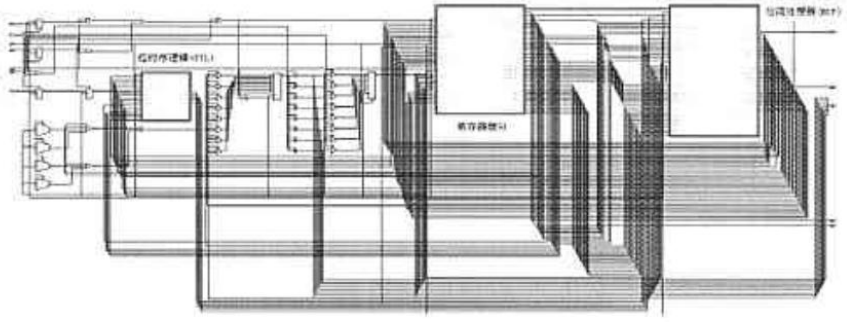


图 4-1 CAN 核顶层逻辑

其内部结构组成关系如图 4-2 所示：

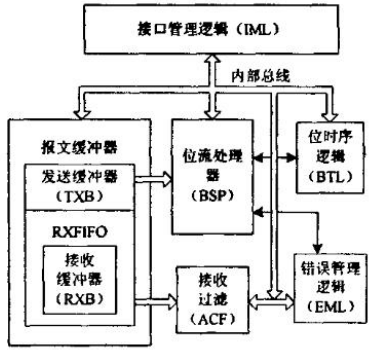


图 4-2 CAN 内部结构

4.2.1 接口管理逻辑（IML）

接口管理逻辑解释来自 MCU 的命令，控制 CAN 寄存器的寻址，向主控制器提

供中断信息和状态信息。

4.2.2 发送缓冲器（TXB）

发送缓冲器是 MCU 和 BSP（位流处理器）之间的接口。它能提供存储器要通过 CAN 网络发送的一条完整报文。缓冲器长度为 13 个字节，有 MCU 写入，BSP 读出。

4.2.3 接受缓冲器（RXB）

接受缓冲器（RXB，RXFIFO）是接收滤波器和 MCU 之间接口，是用来存储从 CAN 总线上接收并被确认的信息。接收缓冲器（RXB，13 个字节）作为接收 FIFO（RXFIFO，长 64 字节）的一个窗口，可被 MCU 访问。

MCU 在此 FIFO 的支持下，可以在处理一条报文的同时接收其他报文。

4.2.4 验收滤波器（ACF）

验收滤波器把它的内容和接收到的标识码相比较，以决定是否接收下这条报文。在验收测试通过后，这条完整的报文就被保存在 RXFFIFO 中。

4.2.5 位流处理器（BSP）

位流处理器是一个在发送缓冲器、RXFIFO 和 CAN 总线之间控制数据流的队列发生器，它还执行总线上的错误检测、仲裁、填充和错误处理。

4.2.6 位时序逻辑（BTL）

位时序逻辑监视串行的 CAN 总线和位时序，它是在一条报文开头，总线传输出现从隐形到显形时同步于 CAN 总线上的位流（硬同步），并且在其后接收一条报文的传输过程中再同步（软同步）。

BTL 还提供了可编程的时间段来补偿传播延时、相位偏移（例如，由于振荡器漂移）和定义采样点和每一位的采样次数。

4.2.7 错误管理逻辑（EML）

错误管理逻辑负责限制传输层模块的错误。它接收来自 BSP 的出错报告，然后把有关错误统计告诉 BSP 和 IML。

第五章 系统仿真与 FPGA 验证

在分层的“自顶而下”的正向设计过程中，每个层次的设计结果都需要进行验证，以保证设计中的错误尽早发现，尽早消除。随着 IC 设计规模的不断扩大，设计验证的复杂性也迅速增加。一个超大规模的集成电路很难不借助计算机完成对设计的验证，本项目采用了几种 EDA 工具，完成了对不同阶段设计结果的仿真验证。

验证的目的主要有以下三个方面：验证硬件语言描述语法的正确性；验证设计结果的逻辑功能是否符合原始规定；验证设计结果中是否含有违反设计规则的错误。即使是用自动综合软件生成的设计结果，也需要进行验证。事实上，任何设计软件都不可能是完美无缺的，不可能考虑到所有的因素。而且库中的一些器件延时和线延时很可能影响到设计，使设计结果偏离设计者的预期。另外，在使用硬件描述语言时，有可能没有正确地反映设计者的设计思想和目标，或者设计思想本身就是错误的，而且硬件描述语言的使用出现语法错误也是常有的。所以在设计的各个阶段都需要对设计进行仿真验证[29]。

常用的验证原理主要有三种：仿真验证或称模拟（Simulation）、规则检查（Design Rule Checking）和形式验证（Formal Verification）。仿真验证，是指对已有的设计输入一系列外部激励信号，通过观察设计中的各个信号在外部激励信号作用下的反应判断该设计是否正确。规则检查是分析电路设计结果中各种数据的关系是否符合设计规则。在版图设计中主要检查不同层线条的相互关系、线宽，以及电学性能等是否符合规定，EDA 工具中的 DRC 检查可以完成这方面的验证。形式验证是最近兴起的一种验证方法。它利用理论证明的方法和数学的方法来验证设计结果的正确性。虽然形式验证可以大大简化设计人员的工作量，但目前这种验证方法还没有成熟，没有实用的商业软件问世。本设计采用了仿真验证来验证设计的逻辑功能和时序[30]。

5.1 仿真与 Testbench

仿真是集成电路设计流程中的重要组成部分，是验证项目的一种手段，与项目设计、实际验证同样重要，利用好仿真功能将提高效率、降低风险性。

仿真包括功能仿真和时序仿真。功能仿真又称前端仿真，是在不考虑器件延时的理想情况下进行的逻辑验证。通过功能仿真可以验证一个项目的逻辑功能是否正确。时序仿真又称后仿真，是在考虑了具体适配器件的各种延时的情况下进行的仿真，时序仿真不仅能测试逻辑功能，还能测试目标器件在最差情况下的时间关系。

Testbench 是为逻辑设计仿真而编写的代码，它能直接同逻辑设计接口，向逻辑

设计施加激励，并检测其输出，如图 5-1 所示。

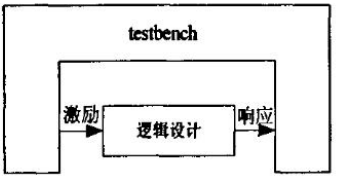


图 5-1 Testbench 作用

Testbench 通常使用 VHDL、Verilog 或者 OpenVera 编写，同时还能调用外部的文件和 C 函数。Testbench 可以使用同逻辑设计不同的描述语言，仿真器通常都提供不同描述语言的混合仿真。

（1）ModelSim 软件的简介

FPGA设计流程包括设计输入，仿真，综合，生成，板级验证等很多阶段。在整个设计流程中，完成设计输入并成功进行编译仅能说明设计符合一定的语法规范，并不能说明设计功能的正确性，这时就需要通过仿真对设计进行验证。在FPGA 设计中，仿真一般分为功能仿真（前仿真）和时序仿真（后仿真）。功能仿真又叫逻辑仿真，是指在不考虑器件延时和布线延时的理想情况下对源代码进行逻辑功能的验证；而时序仿真是在布局布线后进行，它与特定的器件有关，又包含了器件和布线的延时信息，主要验证程序在目标器件中的时序关系。在有些开发环境中，如Xilinx ISE中，除了上述的两种基本仿真外，还包括综合后仿真，转换后（post-translate）仿真，映射后（post. map）仿真等，这样做完每一步都可进行仿真验证，从而保证设计的正确性[34]。

ModelSim是Mentor Graphics子公司MentorTechnology的产品，是当今最通用的FPGA 仿真器之一。ModelSim 功能强大，它支持FPGA设计的各个阶段的仿真，不仅支持VHDL仿真，Verilog仿真，而且支持VHDL和Verilog混合仿真。它不仅能做仿真，还能够对程序进行调试，测试代码覆盖率，对波形进行比较等[35]。

（2）ModelSim软件的仿真流程

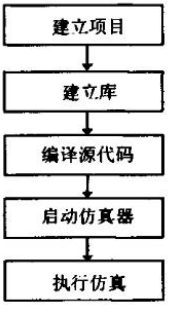


图 5-2 ModelSim 的仿真流程

5.1.1 加减法单元的仿真

加减法单元是 ALU 的重要组成部分，可实现对输入数据的带进位的加、减法运算。如图 5-3 所示，由于加减符号位和上位进位标志的不同而产生不同的结果输出、向下位的进位和溢出。分别实现了带进位的加减法运算，并产生相应的标志位。

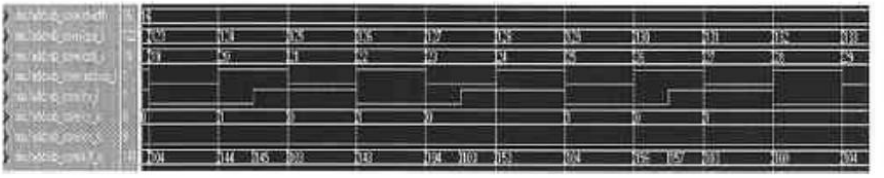


图 5-3 加减法单元仿真结果

5.1.2 十进制调整的仿真

单片机只能对于二进制数进行运算，因为算术运算功能的硬件实现都是基于二进制数的；当对于十进制数的运算，只使用算术运算指令是不够的，必须配合使用一套运算调整指令，才能得到正确的运算结果。

过程为：若相加后累加器 A 低 4 位大于 9 或半进位标志 AC＝1，则加 06H 修正；若 A 的高 4 位大于 9 或进位标志 CY＝1，则对高 4 位也加 06H 修正；若 AC＝1，CY＝1 同时发生，或者高 4 位虽等于 9，当低 4 位修正后有进位，则 A 应加 66H 修正，从而产生相应的结果。

图 5-4 十进制调整单元仿真结果

5.1.3 ALU 的仿真

ALU 是 MCU 的核心部分，它是完成对数据的算术运算、逻辑运算的功能单元，是一种功能较强的组合逻辑电路单元。ALU 模块是由算术运算和逻辑运算电路组成，几乎所有的指令都要在 ALU 中执行。ALL 包含了各种算术运算和逻辑运算功能，包括加、减、乘、除、逻辑运算、移位运算和位操作等。

逻辑指令包括：0011(LAND)、0101(LOR)、0110(LXOR)、0111(RL)、1000(RLC) 1001(RR)、1010(RRC)、1011(COMP)和 1100(INV)八种逻辑关系，产生的仿真结果如图 5-5 所示。

图 5-5 ALU 单元仿真结果

5.1.4 定时/计数器的仿真

定时计数器分别有 4 种工作方式，依次是方式 0：13 位定时计数；方式 1：16 位定时计数；方式 2：8 位自动重装方式和方式 3：2 个 8 位计数器。图 5-6 和图 5-7 分别是方式 1 和方式 2，作为定时、计数器时的仿真图。由于仿真图过多，这里只选择具有代表性的两种来说明。

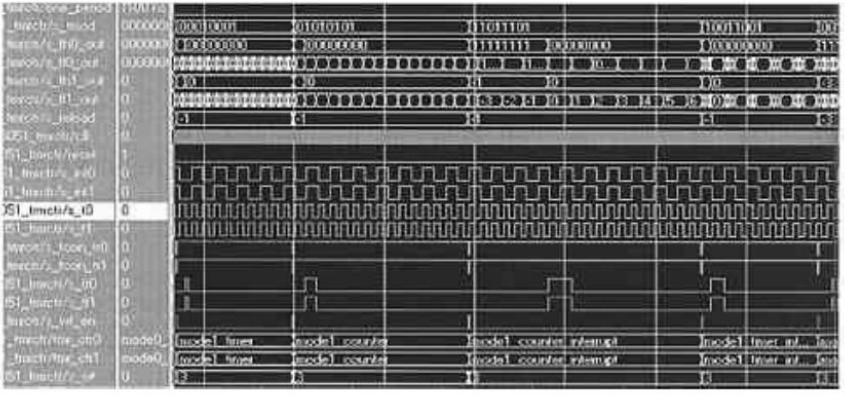


图 5-6 定时/计数器方式 1 仿真

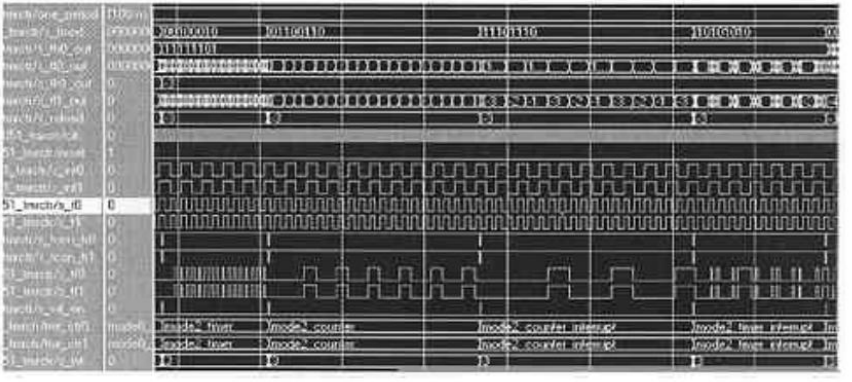


图 5-7 定时/计数器方式 2 仿真

5.1.5 串行口的仿真

串口有 4 种工作方式，本文只针对方式 0 仿真，其他方法相似。

方式 0 为寄存器输入/输出方式，rxdwr_o 为数据传输标志位，数据开始传输时为 1，传输完毕为 0。txd_o 传输同步脉冲，传输 8 个脉冲后 ti 置 1，表示一帧传输完毕，可以申请中断。从图 5-8 中波形可看出实现了数据串行传送的功能，达到了设计要求。

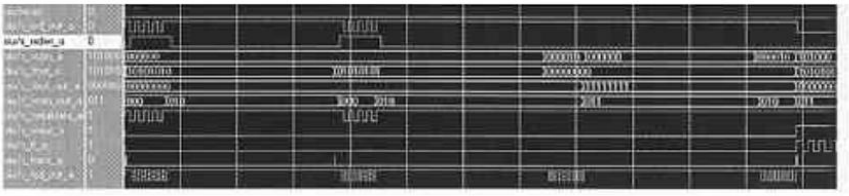


图 5-8 串行口方式 0 仿真

5.2 验证系统的组成

为了验证系统功能的正确性、灵活性，本设计采用了引导程序的概念，实现了用户程序的自动引导。在 ROM 中初始装入引导程序，起始地址 0000H，引导程序实现下位机与上位机的通信，把用户程序到如 PRAM 中，从而实现了，程序运行、验证。

5.2.1 程序 RAM（PRAM）

如前面提到的 PRAM 是为了保存用户的应用程序而设立的一个存储器。具有 64K 的存储空间，足适合一般程序存储。本设计采用了 Altera 公司提供的参数化模块（LPM）器件。具体过程同前面 ROM 的建立过程相似，在此不再赘述了。

5.2.2 数据多路选择器（DATAMUX）

数据多路选择器设计是为了实现引导 ROM 和用户 PRAM 程序之间的切换。其结构如图 5-9 所示，其中 select_i 是存储器选择器发出的一个选择信号，确定输出是哪一路进来信号。

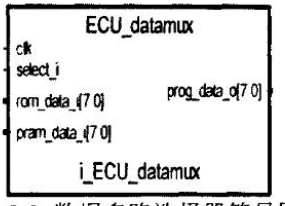


图 5-9 数据多路选择器符号图

```
if (s_select = '1') then
  s_prog_data <= s_rom_data;    --把 ROM 输出
else
  s_prog_data <= s_pram_data;   --把 PRAM 输出
end if;
```

5.2.3 存储器选择器（CHIPSEL）

因为本设计采用了引导程序的概念，增加了存储器，所以选择存储器，使其使能是又一个重要的设计部分，存储器选择器就是用来实现存储器的选择。如图 5-10，输入由 ROM、RAMX 地址组成，依据地址分配而确定 ROM、PRAM 和 RAMX 的输出使能。其中输出：

```
signal s_rom_en   : -- ROM   0000 to 1FFFh 代码空间
signal s_pram_en  : -- PRAM 2000 to 3FFFh 代码和数据空间
signal s_ramx_en  : -- XRAM 0000 to 1FFFh 数据空间
```

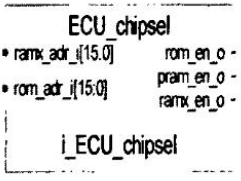


图 5-10 存储器选择器符号图

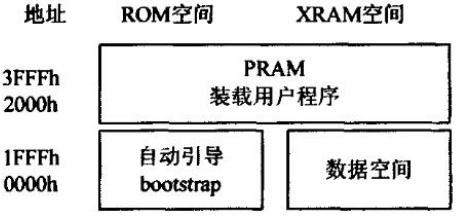


图 5-11 存储器空间分配图

5.2.4 锁项环（STRATIXPLL）

由于本设计采用的 FPGA 芯片是 Altera 公司的 Stratix 系列芯片，芯片上只提供最小 50MHz 的频率，而本设计 ECU 频率只有 25MHz，所以采用了锁项环实现分频

功能。采用锁项环也是为了以后当设计优化时容易改变频率，也实现了稳定频率。锁项环是采用 LPM 模块实现的，采用了 2 分频的方法实现了需要的频率。

5.3 自动引导程序原理（BOOTSTRAP）

自动引导程序是为了实现程序的自动执行，而在引导区里装载一个小小的程序，每次系统重起、复位，系统会装载入引导程序，引导程序再去调用其他应用程序从而实现系统功能。

5.3.1 BOOTSTRAP 意义及作用

Bootstrap 是皮鞋后部的一条小带子或一个小环，它可以使你方便地把鞋子穿起来。在计算机中，是指使用一个很小的程序将某个特定的程序（通常是指操作系统）载入计算机中。在本设计中引用是为了实现用户程序的引导，而不必每次都要固化到 ROM 中，这样既可以实现自动转载也可以脱离 Quartus 应用程序而改变用户程序。

5.3.2 下位机（MCU）引导程序组成

下位机引导程序是依据引导协议编写的汇编语言程序，经过编译后形成 HEX 形式文件固化到 ROM 中的，用来引导用户程序进去 PRAM 中。其执行过程如下：

（1）配置串口为 8-N-1 形式，波特率为 4800，没有硬件握手信号

（2）当下位机程序（BOOTSTRAP）开始执行时，其将会通过串口向上位机发送一个“=”符号，表示可以连接。

（3）此时上位机将会通过串口向下位机传送用户程序，一个 HEX 格式的文件。

（4）文件传输完毕，下位机会传送一个“:”符号。但如果传输错误，将会传送一个标志值，标志有什么错误产生。

（5）如果有错误产生，引导程序将不会接受下载程序而返回一个“?”表示询问。

（6）有效的文件下载之后，引导程序将会向上位机传回一个信息，包括下载文件的大小和程序执行的地址/2000<CR>。

（7）如果地址正确，在执行下载程序之前，引导程序会返回一个“@”符号表示通信结束。

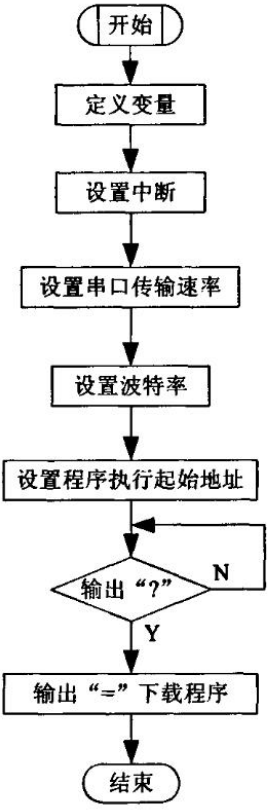


图 5-12 下位机引导程序流程图

5.3.3 上位机（PC）引导程序组成

上位机程序可以实现与下位机通信并下载用户程序，是按照通信协议而编写 C 程序，通过编译形成可执行文件，运行于 PC 机上。但需要下载应用程序是只要运行此程序，指定下载文件（HEX）文件的地址，就可以通过协议正确下载入 FPGA 中。程序流程图如下图所示：

主程序包含多个子程序分别实现不同的功能：介绍如下：

```
init()  //搜索所有串口程序，初始化
initBaud(PORT, 211)      //设置搜索到的串口 4800 Baud, 8-N-1
int readReady(int COM)      //检测串口可读
int writeReady(int COM)      //检测串口可写
unsigned char serWriteRead(int COM)      //接收数据字符
unsigned char serRead(unsigned char b，int COM)      //发送数据字符
```

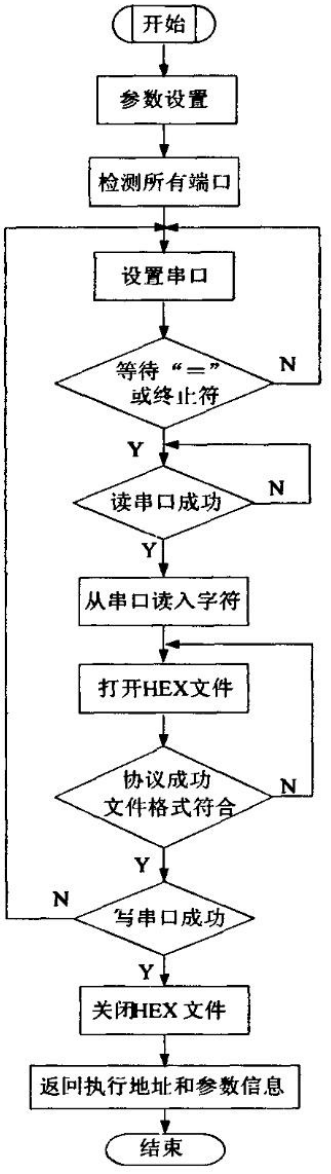


图 5-13 上位机引导程序流程图

5.4 FPGA 验证过程及结果

用于芯片设计的诊断是 FPGA 芯片的一个重要的应用。在以往的芯片设计中，从系统设计，逻辑设计，各层次仿真到综合，直至流出样片才能在硬件层次验证设

计的正确性。如果最后才发现有错，不仅要浪费流片的巨大成本，而且要耗费大量的时间。有什么措施能增强设计在硬件层次的正确性呢？FPGA 正是因应了这个要求。在流样片之前，可以把设计写入 FPGA 芯片中，并把此芯片置于测试系统中，或置于应用系统中，通过测试可以验证芯片设计的正确性，诊断出错误的地方。由于 FPGA 相对于流片的时间和成本要优越得多，所以把 FPGA 用于设计能够很好的验证设计的各项功能、时序，大大的提高了芯片设计的可靠性。

设计虽然经过仿真，仿真是在理想情况下得到得结果，但在实际应用是可能会遇到某些特殊的情况，本设计采用了 FPGA 方式来验证设计的正确性。只是验证了 MCU 单元的功能，具体通过下载进去一个 LED 灯循环点亮程序，完成了串口通信、定时器延时、中断等几个 MCU 基本功能[36]。

5.4.1 试验开发板介绍

Stratix Professional Edition Nios 开发板为基于 Altera 公司的 Stratix 器件进行嵌入式系统的开发提供了硬件的开发平台。开发板上的 FPGA 器件的型号是 EP1S40780C5，它具有 41250 个逻辑单元（LE，Logic Elements）和 3423744 比特的片上存储器单元。如图 5-15 所示，其中包括：

（1）存储器

开发板上的 U5 是一个 8Mbyte AMD AM29LV065D Flash 存储器，它连接到 FPGA，并且具有两个作用。

（2）Stratix 器件中运行的 NiosII 嵌入式处理器可以将该 Flash 存储器用作通用可读存储器和非易失性存储器件。

该 Flash 存储器可以存储 Stratix 器件的配置数据，在系统上电时配置控制器（EPM7128AE）从该 Flash 存储器中读出配置数据并将之下载到 FPGA 中。

（3）SDRAM 器件

开发板上的 U57 是 Micron MT48LC4M32B2 型号的 SDRAM 芯片，它具有 PC100 功能并且具有自刷新模式。该 SDRAM 可以与在系统时钟的正边沿寄存的所有信号全同步。

（4）扩展引脚

PROTO1 扩展口包括 J11、J12、J13 扩展口，可以用于实现 FPGA 与具有特殊功能的子卡之间的连接。它与 CompactFlash 连接器共用 Stratix I/O 引脚，在实际应用中同一时刻只能选择两者中的一个应用。

（5）配置控制器件

Stratix Professional Edition Nios 开发板的控制配置器件是 MAX 7000 EPM7128AE CPLD 器件。

（6）串行接口

J19 和 J27 时标准的 DB-9 串行接口，它们主要用于与主机通信。由于 Stratix FGPA 端口不能直接与 RS-232 电平相连，所以在 J19、J27 与 Stratix FGPA 之间需要连接左移缓冲器（U52 和 U58）[37]。

5.4.2 验证及结果

为了进一步验证设计的正确性，将一个 LED 灯的程序下载到 Altera 开发板，通过结果验证 MCU 微处理器功能的正确性。

首先需要在 Quartus 中绑定 FPGA 管脚，具体如下图所示。设置 P0 口作为为 LED 输出，串口和时钟等管脚。

	To	Location	I/O Bank	I/O Standard	General Function	Special Function	Res
1	clk	PIN_E17	1	LVTTL			
2	p0_o[0]	PIN_H27	7	LVTTL			
3	p0_o[1]	PIN_H28	7	LVTTL			
4	p0_o[2]	PIN_J23	7	LVTTL			
5	p0_o[3]	PIN_L24	7	LVTTL			
6	p0_o[4]	PIN_J25	7	LVTTL			
7	p0_o[5]	PIN_J26	7	LVTTL			
8	p0_o[6]	PIN_L20	7	LVTTL			
9	p0_o[7]	PIN_L19	7	LVTTL			
10	reset_n	PIN_W5	4	LVTTL			
11	al_txd_o[0]	PIN_Y23	7	LVTTL			
12	al_txd_o[0]	PIN_U21	7	LVTTL			

图 5-14 管脚绑定

图 5-15 中是用来下载的 FPGA 开发板，其中白色的盒是一个 USB 下载线，它的作用是在计算机和开发板之间进行数据的传输。串口是用来和上位机通信，下载数据等。

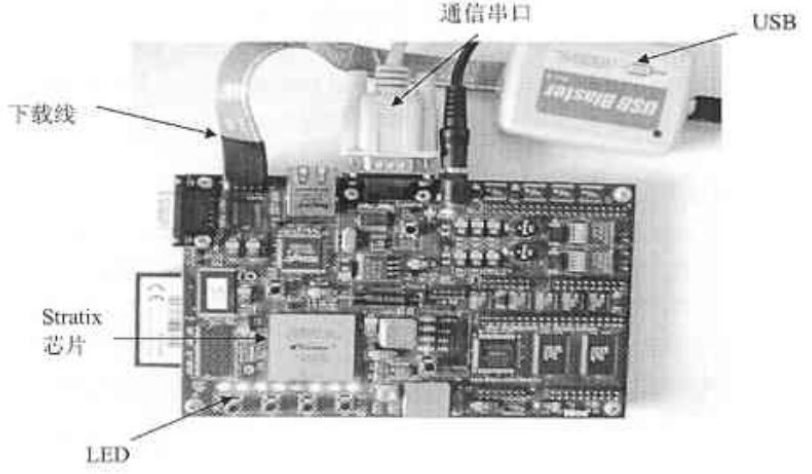


图 5-15 Altera 开发板

当完成引导程序的下载后，打开上位机 PC 应用程序，界面 5-16 所示。寻找通信串口。当设置完串口后显示“=”表示下位机与上位机初始通信完毕，开始下载用户程序。等程序下载完毕会返回信号，并设置程序初始执行地址为 2000H，MCU 开始从 2000H 执行用户程序。LED 循环点亮程序的执行结果如图 5-16 所示。

程序执行过程涉及了 MCU 的各种功能：需要通过串口通信、定时/计数器做波特率发生器、按键产生中断信号及程序正确的执行结果，这些表明 MCU 设计的正确性，程序运行的稳定性。

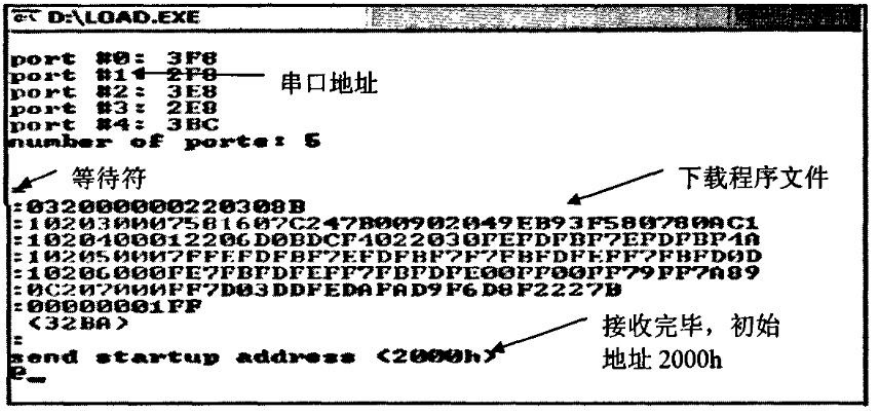


图 5-16 上位机应用程序界面及运行结果

第六章 结束语

本文研究了国内外电子控制单元的应用和发展情况，总结出目前国内发展的不足，提出设计要求。利用电子信息技术、现场总线技术和集成电路设计技术，研制了基于 CAN 总线的电子控制单元；依据设计原则，提出设计方案；并按照自顶向下的集成电路设计原则，设计并裁减 8051 兼容的 MCU 核。分别对 MCU 核的各个组成部分设计了仿真文件，并编写了自动引导程序用于 FPGA 验证，实现了验证过程的简洁化。

本设计可以实现对车身、底盘等低速设备的控制。采用 8051 兼容的 IP 核，实现了应用方便；采用流水线技术，实现了处理速度的提高；采用各个部分集成的方式，实现了集成化、减少面积；集成了 CAN 总线，实现了远程控制、通信和级联。因此本设计将具有及其广泛的应用前景。

参考文献

[1] 袁大宏.发展我国的汽车电子产业[J].电子世界.2002[3]:2-4

[2] 张云龙，袁大宏等.电控汽车故障诊断技术的现状与发展趋势[J].汽车技术.2000[7]:30-32

[3] 何宇漾.电控汽车中电子技术的应用[J].渤海大学学报(自然科学版)2005[1]:28-33

[4] 徐建平.现代汽车自诊断系统效率的分析[J].汽车技术.2004[5]:1-4

[5] 徐建平.现代汽车微机自振断系统及其特性[J].贵州大学学报.2004[5]:200-204

[6] 唐颖.EDA 技术与单片机系统[J].现代电子技术，2002，11(3): 31-32

[7] 李广军，孟宪儿.可编程 ASIC 设计及应用[M] .成都：电子科技大学出版社，2004

[8] 任晓东，文博.CPLD/FPGA 高级应用开发指南[M].电子工业出版社，2003：7，67

[9] 徐志军，徐光辉.CPLD/FPGA 的开发与应用[M].电子工业出版社，2002： 15-16

[10]陆重阳，卢东华，文爱军.IP 技术在 SOC 中的地位及应用[J] .微电子技术，2002，(8):20-23

[11]潘松，王国栋.基于 EDA 技术 CPLD/FPGA 应用前景[J].电子与机自动化，1999，(3):3-6

[12]慈艳柯，陈秀英，吴孙桃等.片上系统的设计技术及其研究进展[J].半导体技术.2001, 7(26):12-16

[13]张振荣，晋明武，王毅平.MCS-51 单片机原理及实用技术[M].北京：人民邮电出版社，2000

[14]梁合庆.增强闪存 80C51 教程[M].北京：电子工业出版社[M]，2003

[15]胡汉才.单片机原理及其接口技术[M].北京：清华大学出版社，2002

[16]尹林子，李广军.基于 VHDL 的 8051 IP 核设计.中国通信学会通信令用集成电路委员会,中国电子学会通信学分会.CCIC2004 中国通信集成电路技术与应用研讨会论文集.杭州，2004.62~65

[17]侯伯亨，顾新.VHDL 硬件描述语言与数字电路设计[M].西安电子科技大学出版社，2003

[18]朱明程.FPGA 最新技术综述 [J].电子技术应用.1999，11(2)：57-60

[19]曹琳琳，曹巧媛.单片机原理及接口技术[M].国防科技大学出版社 2000.7：11-111

[20]慈艳柯.MCS-51 单片机芯片反向解剖及正向设计的研究[J] .厦门大学，2002

[21]饶运涛，邹继军，郑勇芸.现场总线 CAN 原理与应用技术[M].北京，空航天大学出版社，2003.6：1-85

[22]邬宽明. CAN 总线原理和应用系统设计[M].北京，航空航天大学出版社，1996.11

[23]确定 SJA1000 CAN 控制器的位定时参数.http://www.zlgmcu.com

[24]SJA1000 独立的 CAN 控制器应用指南.http://www.zlgmcu.com

[25]罗雪梅.基于 SJA1000 的 CAN 总线接口电路的设计与实现[J]. 贵州工业大学学报（自然科学版），2003.32(4)：42-44,54

[26]宋磊，郑荣良.汽车局域网 CAN 总线在客车上的应用研究[J].江苏大学学报(自然科学版)，2002.23(2)：55-58

[27]王建，左启耀，高峰.基于 CAN 总线的客车控制系统[J].吉林大学学报（工学版），2004.34(11)：282-285

[28]李光升，谢永成，吕强等.基于 CAN 总线技术的车辆通讯系统[J].装甲兵工程学院学报，2000.14(3)：48-52

[29]禹定臣.MCS－51 单片机应用系统的设计及仿真调试[J].天中学刊，2001，16(2)：85-86

[30]赵学军.单片机实时嵌入式操作系统微内核的设计[J].桂林电子工业学院学报，2002，22(3)：76-79

[31]Steven J.E. Wilton and Resve Saleh .Programmable Logic IP Cores in SoC Design; Opportunities Challenges. Proceedings of the Custom Integrated Circuits Confierence. 2001 p63-66

[32]Mark Zwolinski. Digital System Design with VHDL[M]. Prentice Hall，2000.10

[33]Michael Keating, Reuse. Methodology Manual for System-on-a-chip Design[J], Kluwer Academic PUBLISHERS，2002.5

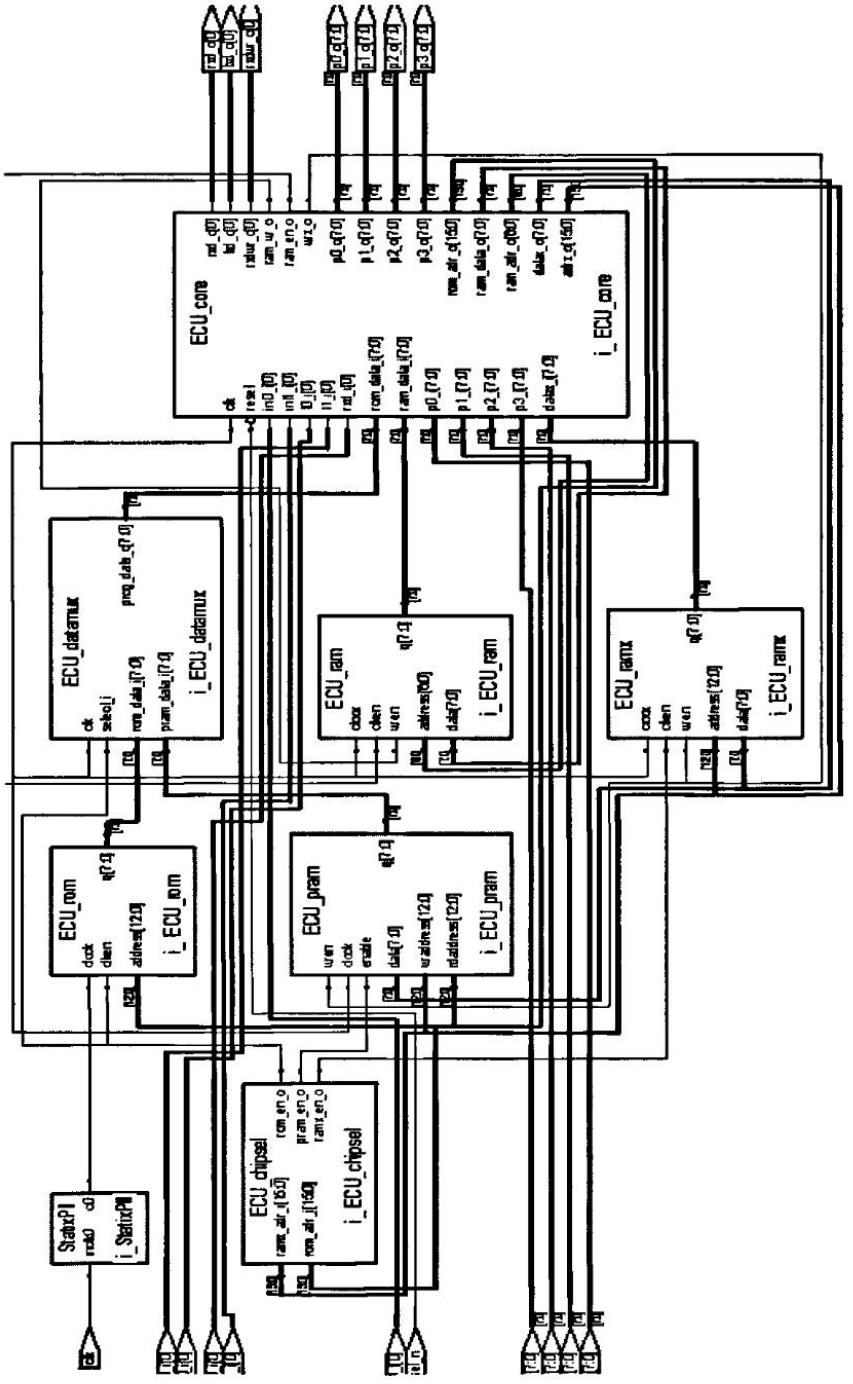
[34]Model Technology. Modelsim SE User's Manual, 2003. 9

[35]Janick Bergeron. Writing Testbenches-Founctionl Verification of HDL Models[M], Kluwer Academic Publishers, 2001.7

[36]Prakash Rashinkar, System-on-a-chip Verification Methodology and Techniques[M], Kluwer Academic Publishers, 2002

[37]Nios Development Board Reference Manual, Stratix Professional Edition. Altera Inc, September, 2004

附录一 FPGA 自动验证系统组成



附录二 发表的学术论文及科研成果

1、杜向军，苗长云. 一种基于并口的虚拟仪器设. 微计算机信息，2005. 11.

2、杜向军，苗长云，李鸿强. 温室智能控制系统. 计算机应用研究，2006. 10~12.

3、HONG-QIANG LI, CHANG-YUN MIAO, XIANG-JUN DU, XIAO-JUN LIU. FPGA Implementation of Forward Error Correction for a Software Defined Radio. Proceedings of The International Conference on Informatics and Control Technologies (ICT2006)

4、Hongqiang Li, Changyun Miao, Xiangjun Du. Radio Frequence IDentification Design and Simulation using the VHDL-AMS language. Proceedings of The IET International Conference on Wireless, Mobile and Multimedia NetworksLondon:the Institution of Engineering and Technology. 2006.274-277

5、于玲，王熙，杜向军. 一种基于 XML 的集中会议控制协议. 微计算机信息，2005. 12.

6、天津市“十五”05 科技创新项目，汽车电子中 ECU 的集成电路设计.

致谢

光阴荏苒，转眼二年多的研究生生涯即将结束。回首往事，心中不禁感慨万千。本课题正是在师长、家人和朋友们热心的鼓励和支持下，才得以顺利完成，此时此刻，我要向他们表达最衷心的感谢！

首先向辛勤培育我的导师苗长云教授致以最诚挚的谢意和由衷的感谢！

二年多的学习、生活、课题的研究及学位论文的构思、撰写到最后的定稿，都灌注了导师苗长云教授的亲切关怀和悉心指导。苗长云教授高尚的道德情操、严谨的治学态度、渊博的学识、敏锐的科学思维、诲人不倦的学者风范、求实创新的开拓精神，时时刻刻激励着我，这些宝贵的精神财富将使我终身受益。

同时，我要感谢李鸿强老师。感谢他在学习上、工作上对我无私的帮助与支持，并在课题上给予了我悉心指导。

在课题进行和论文撰写中，以及在校的学习期间，实验室的很多老师、同学和朋友，给予了我很大的启发和帮助，在此表示深深的谢意。

研究生生涯至此结束了，大家曾经一起努力、互相帮助、共同勉励，这么多年建立起的真挚的师生情、同学情将成为我最宝贵的记忆。

最后，祝愿关心、帮助过我的老师、同学、朋友们在未来的工作、生活中都能开心的生活每一天！你们的帮助、关怀我会铭刻在心！