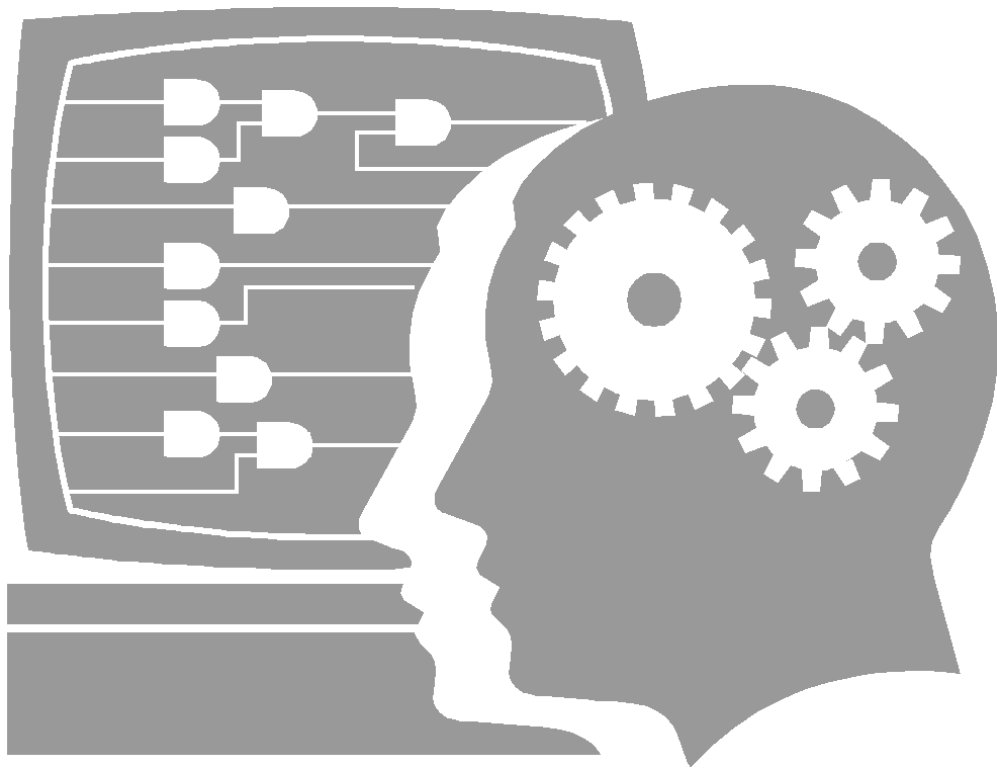


# 现代计算机组成原理实验讲义



杭州康芯电子有限公司

[www.kx-soc.com](http://www.kx-soc.com)

# 目 录

前 言	1
实验一 QUARTUSII EDA 工具与 VHDL 基础实验	4
1-1. 应用 QuartusII 完成基本组合电路设计	4
1-2. 应用 QuartusII 完成基本时序电路的设计	5
1-3. 设计含异步清 0 和同步时钟使能的加法计数器	6
1-4. 7 段数码显示译码器设计	7
1-5. 8 位数码扫描显示电路设计	8
1-6. 数控分频器的设计	9
1-7. 32 位并进/出移位寄存器设计	10
1-8. 在 QuartusII 中用原理图输入法设计 8 位全加器	10
1-9. 在 QuartusII 中用原理图输入法设计较复杂数字系统	11
1-10. 用 QuartusII 设计正弦信号发生器	11
1-11. 8 位 16 进制频率计设计	13
1-12. 序列检测器设计	16
1-13. VHDL 状态机 A/D 采样控制电路实现	17
实验二 运算器组成实验	19
1. 算术逻辑运算实验	19
2. 带进位算术运算实验	22
3. 移位运算器实验	23
实验三 存储器实验	24
1. FPGA 中 LPM_ROM 定制与读出实验	24
2. FPGA 中 LPM_RAM 读写实验	26
3. FIFO 定制与读/写实验	27
4. FPGA 与外部 16 位 RAM 接口实验	28
实验四 微控制器实验	30
1 节拍脉冲发生器时序电路实验	30
2. 程序计数器 PC 与地址寄存器 AR 实验	32
3. 微控制器组成实验	33
实验五 总线控制实验	36
实验六 基本模型机设计与实现	38
实验七 带移位运算的模型机设计与实现	47
实验八 复杂模型机的设计与实现	52
实验九. 较复杂 CPU 设计示例	56
实验十. 16 位精简指令 CPU 设计实验	58
实验十一 32 位 NIOS CPU 嵌入式系统软硬件设计实验	67
1 NIOS 软硬件开发流程	67
3 SOPC 整体系统生成	72
4 Nios 硬件系统生成	72
实验十二 32 位 NIOS CPU 测控系统串口接收程序设计	75
实验十三 GSM 短信模块程序设计	75
实验十四 基于 32 位 NIOS CPU 的秒表程序设计	76
实验十五 NIOS AVALON SLAVE 总线外设 (PWM 模块) 设计	79
实验十六 NIOS AVALON SLAVE 总线外设 (数码管动态扫描显示模块) 设计	79
实验十七 基于 NIOS 的 VGA 显示终端设计	79
实验十八 DMA 应用和俄罗斯方块游戏设计	80
实验十九 为 NIOS 嵌入式系统增加算法加速协处理模块控制指令	80
实验二十. 计算机体系结构实验	80
实验二十一. 89K51 单片机核应用系统软硬件设计实验	80
<b>8051/89C51 CPU 核及片上系统设计实验</b>	

4-1	基本 CPU 软硬件设计	
4-2	8051/89C51 单片机核，等精度频率计与液晶显示实验	
4-3	8051/89C51 单片机核，等精度频率计与数码管显示实验	
附录：	GW48 EDA/SOPC 主系统使用说明 .....	89

## 前 言

随着大规模集成电路技术和计算机技术的不断发展,在涉及通信、国防、工业自动化、计算机应用、仪器仪表等领域的电子系统设计中,现场可编程(FPGA)技术含量正以惊人的速度上升。电子类的新技术项目的开发也更多地依赖于 FPGA 技术的应用,特别是随着 VHDL 等硬件描述语言综合工具功能和性能的提高,计算机中许多重要的元件,包括 CPU 都用硬件描述语言来设计和表达,许多 CPU (如 8051 单片机、8086 等),硬核嵌入式系统(如 ARM、Excalibue 系列 FPGA)、软核嵌入式系统(如 Nios),微机 CPU,乃至整个计算机系统都用 FPGA 来实现,即所谓的单片系统: SOC 和 SOPC (System On a Chip、System On a Programmerble Chip)。计算机和 CPU 的设计技术及其现方法进入了一个全新的时代!不但如此,传统的 CPU 结构模式,纽曼结构和哈佛结构正在受到巨大的挑战。

例如,美国赢通系统公司(Wincom Systems)推出一款令人惊叹的服务器,其核心部分由 FPGA 完成的超强功能 CPU。该系统工作能力超过 50 台戴尔或 IBM 计算机,或 Sun Microsystems 公司的服务器。该服务器的处理速度要比传统服务器快 50 到 300 倍。我们知道,传统的个人电脑及服务器通常采用英特尔的奔腾处理器或 SUN 公司的 SPARC 芯片作为中央处理单元,而赢通的这一产品却没有采用微处理器,而是由 FPGA 芯片驱动。FPGA 芯片的运行速度比奔腾处理器慢,但可并行处理多项任务,而微处理器一次仅能处理一项任务。因此,赢通公司的服务器只需配置几个价格仅为 2000 多美元的 FPGA 芯片,便可击败 SUN 公司的服务器或采用英特尔处理器的电脑。

50 多年前,匈牙利数学家纽曼(John von Neumann)提出了电脑的设计构想:通过中央处理器从存储器中存取数据,并逐一处理各项任务。现在,通过采用 FPGA, SOPC 及顺序与并行方式相结合的软核嵌入式系统取代传统微处理器,导致 Xilinx 的 CEO Willem Roelandts 所说的“由纽曼提出的电脑架构已经走到尽头”,“可编程芯片将掀起下一轮应用高潮。

FPGA 芯片以操作灵活著称,可以重复擦写无限次,而微处理器均采用固定电路,只能进行一次设计。设计人员可通过改变 FPGA 中晶体管的开关状态对电路进行重写,即重配置。从而,尽管 FPGA 芯片的时钟频率要低于奔腾处理器,但是由于 FPGA 芯片可并行处理各种不同的运算,所以可完成许多复杂的任务。例如网页显示,全球天气建模及基因组合核对等,而且处理速度比奔腾处理器或数字信号处理器快得多。

戴尔和 SUN 公司生产的某些标准服务器也采用了 ALTERA 公司的 FPGA 芯片。时代逻辑公司对这些标准服务器加以改进之后,生产了一种用于基因研究的高速处理设备。时代逻辑公司总监 Christopher Hoover 说该设备比原来的产品至少快 1000 倍。

美国的 Annapolis Micro Systems 公司在其电脑芯片电路板中也集成了 FPGA 芯片,以便提高其产品性能。该公司首席执行官 Jane Donaldson 指出,相关产品的销售量与前两年相比翻了一番。

IBM 和 Xilinx 公司正在合力开发一种混合芯片,以整合前者的 PowerPC 微处理器和后者的 FPGA 芯片。这种芯片的好处是:一台网络服务器的 FPGA 部分可以根据不同的标准进行订制,而不用为每个国家开发一种新的芯片。

美国 Star Bridge Systems 公司也声称已在进行一项技术尝试,即采用 FPGA 芯片和该公司自己的 Viva 编程语言开发出“超级电脑(hypercomputer)”,对该超级电脑进行测试的美国国家航空航天局(NASA)科学家表示“其运行速度无与伦比,这一产品的性能令人过目难忘”。

超级电脑是科技世界中的极品:售价奇高、速度飞快、集成了数以千计的微处理器。但这种超级电脑也浪费了非常多的芯片资源,每个处理器只能进行单任务操作,大部分功能难以充分发挥。现在有了另一种更为简洁的设计:设计工程师采用 FPGA 芯片来武装超级电脑,取代了原先大量的英特尔奔腾处理器。经过使用硬件描述语言和相关软件语言的设计, FPGA 芯片可并行处理多项任务,从而使所有电路都能随时发挥作用。又由于 FPGA 芯片可以反复编程培植,而且几乎可瞬时完成。例如可通过利用 FPGA 的重配置功能,在某一时刻它可以用来预报全球天气状况,而下一时刻又可根据某公司做的主要利率对冲情况来评估债券市场的风险,或是转而去作图象信息处理。

其它公司或机构的研究人员,如美国加州大学伯克利分校(University of California, Berkeley)和杨百翰大学(Brigham Young University)的研究员也正在设计基于 FPGA 的电脑,这些电脑可在运行中实现动

态重配置。这对定位危险目标等军事应用和面容识别一类的计算密集型安全应用十分有用。

不言而喻，全新的计算机设计技术和实现技术向传统的计算机组成原理的教学内容和实验方式提出尖锐的挑战。相形之下，传统的计算机组成原理实验方式已显得十分不合时宜。显然，传统的实验方式，就教学实验内容上看，与现代的计算机组成模式、理念、基本理论和构成形式及方案上都无任何吻合之处；就实验方法上看，完全是一种脱离现代计算机实际组成技术和测试技术（现代测试技术包括 JTAG、嵌入式逻辑分析仪等）的，由实验系统设计者一相情愿的构建，难免引出诸多对初学者产生误导的实验方法；就实验模式和模型上看，与真实的现代计算机组成和构建方式相去更远。现代计算机，包括嵌入式系统绝不可能是由一大堆独立的、低速的、传统逻辑器件连接而成，而是由硬件描述语言来表达，由 ASIC 或 FPGA 来实现的计算机系统；就教学实验内容上看，传统的计算机组成原理实验只能模拟普通 CPU 的工作，然而现代计算机应用领域，RSIC 精简指令 CPU 的设计和应用正以前所未有的巨大规模向前发展，当前，无论在开发技术的投入还是应用市场的开拓都远远超过了基于传统构架的 CPU（国内绝大多数学校仍然基于此 CPU 的教学）。

不久前，清华大学对美国一些知名大学计算机实验室（如斯坦福大学）的调研表明，那里计算机方面的硬件实验，包括计算机组成原理实验早已不用那种传统接插式实验，而是全部采用 EDA 技术进行所有的软硬件实验!!!

显然，使用大规模 FPGA、EDA 软件工具和 IEEE 标准硬件描述语言构建的现代计算机组成原理实验系统取代传统的计算机组成原理实验已成为势在必行。

利用 FPGA 技术，在实验中能方便灵活地设计出简单完整的 CPU 模型机。基于查找表硬件结构的商用 FPGA 是当前进行快速系统原型设计最流行的 ASIC 手段。ALTERA 的 ACEX 系列 FPGA 产品具有片上 EAB，可以构成构成各种类型的存储器结构，利用在其内部的 LPM 可以实现微程序控制和管理复杂逻辑电路。

现代计算机组成原理实验为实验者提供了全新的学习平台，彻底克服传统组成原理实验项目与实际 CPU 设计技术完全脱钩，学用脱节，甚至误导的缺陷。让学生有机会接触到最新的计算机组成与设计方面的知识，使学习与工程实际相结合。

传统计算机组成原理实验系统与现代计算机组成实验系统性能特点比较

结构与功能特点	传统计算机组成原理实验系统	现代计算机组成原理实验系统
实验特点	本身只是一验证性模型，与实际的计算机设计模型无关	真实反映了现代计算机设计工程实现原理、测试方法和设计技术
结构特点	由规模不等的离散集成电路块和 CPLD 等器件构成 CPU 模型	整个 CPU，乃至 RAM、ROM 和通信接口可在单片 FPGA 中实现
实验 CPU 总线控制方式	采用三态门控制，仅适用于 74 系列小规模集成电路构成方式	采用总线多路开关，适于 VLSI 和 FPGA 等大规模集成电路工程
CPU 指令与微指令存储与形成方式	通过外部 ROM 或 EEPROM 构成，指令的数量和微指令的宽度受到限制，难以扩展，CPU 模型结构被限制。	既可以采用传统的 ROM 或 EEPROM 存储，又可以采用 FPGA 中的 EAB 嵌入式方式，构成单片系统，更符合现代 CPU 设计理念。
CPU 指令和微指令的实现方式	手工设计、画微指令流程图；手工（烧写或键入）输入方式实现。设计效率低、可靠性低，查错、排错、调试困难，耗时费力。	利用计算机输入，形成专用文件格式，由 EDA 工具自动配置进 FPGA 中设定的 RAM、ROM 中，便捷、高效、实用，规范
可用硬件资源	采用中小规模集成电路，硬件资源非常有限，且结构固定，不便于系统扩展、设计思路受限制，有创意的设想无从得到验证	采用 FPGA 超大规模集成电路，可利用资源丰富，灵活，设计者可根据需要反复调整和改变电路结构，创新设想易得到验证和实现
观察计算机内部指令执行情况，及软硬件排错	通过有限的发光二极管和数码管设置观察点，难以观察指令执行的细节情况，如竞争、毛刺等。硬件电路和软件排错都十分困难	除了能在 PC 上对整个软硬件系统进行时序仿真外，还可通过 JTAG 口使用嵌入式逻辑分析仪对 CPU 内部任意点，跟踪指令与测试。

实验设计、连线方式及可靠性	元件间通过硬件连线，手工完成，费时费力，效率低、可靠性差。外部连线过多、导线与器件反复插拔，导致导线内部折断损伤。	各功能部件间无外部连线，几乎所有接线都在 FPGA 片内通过计算机连接实现，并自动检测排错，现场配置，可靠性高，无寿命限制。
设计可移植性和可保存性	由于需当场连线，故功能模型无可移植性和保存性，且必须有实验系统才能做实验，所以绝难保证每一同学给出自己特色的设计	可保存，可移植，可在自己的 PC 上设计和硬件仿真。最后到实验室在实验系统上作硬件测试即可，每一同学的设计都有自己特点
各功能模块可改进性	基本不能	各模块功能都可改变，如 ALU, 移位器, RAM/ROM 的容量位宽等
工作速度	由于大多采用 74 系列、TTL 器件，工作速度低	采用高速、低压、低功耗 FPGA，速度可达 100MHz→真实 CPU
嵌入式模块的利用	不能利用	Nios、各种 I/O 接口模块，如并行接口、串行通信接口，VGA 等
可扩展性	不能	可扩展形成计算机组成中不同 CPU 结构，总线宽，及实用接口等
多功能性	只能对计算机组成原理作传统方式的验证性实验，功能单一、模式陈旧，国外一流大学，如斯坦福大学计算机系完全不用此类实验方式，而用 EDA 技术完成相关的实验，	可实现现代计算机组成原理实验、EDA 实验、硬件描述语言 VHDL、Verilog 教学实验、电子设计竞赛开发、实用 CPU 或单片机设计或验证等等
RSIC CPU 设计实验	完全不能	实现容易
嵌入式系统硬件实现和软件开发实验	完全不能	利用 QuartusII、SOPC Builder 和嵌入式软核 Nios 可以实现。

现代计算机组成原理实验系统 GW-48 CCP 的 CPU 核心部分由 FPGA 设计实现，通过 FPGA 与单片机的接口，将 CPU 核心部件中的指令寄存器、程序计数器、地址寄存器、暂存寄存器、运算寄存器、缓冲寄存器、存储器、微地址寄存器、输入缓冲寄存器等大量的数据实时地在数码管和 LCD 液晶显示屏显示出来。各类操作指示、数据动态流向显示，直观明了，一目了然，摆脱了与电脑联机的麻烦。该实验仪采用 FPGA 设计 CPU 内部结构，采用模块化设计，单元电路分开，模块间连接通过内部总线和总线选择多路开关连接相，不必进行硬件连线，从而大大提高了实验的成功率。

传统的计算机组成原理实验台体积庞大，使用的芯片种类繁多，实验中需要花许多时间进行大量的连线，系统的可靠性低，由于芯片或连线出现的各种故障排查困难。

GW48 C+计算机组成原理实验台采用模块化的系统结构，学生可通过一系列基本单元实验和模型计算机综合设计实验，对 CPU 的运算功能、控制功能、总线结构、指令系统的设计和微指令的实现，以及 CPU 内部是如何工作的，有直观、深刻的认识。学生在进行各个单元实验和综合实验时，既可以通过系统计算机进行综合设计，系统软件仿真、观察仿真波形，更重要的是可以在 GW48 CP+实验平台上，将自己设计的 CPU 电路下载到 FPGA 中进行硬件仿真。观察 CPU 内部的各种信息：包括数据总线、地址寄存器、程序计数器、指令译码器、指令寄存器、控制信号、内部寄存器、数据寄存器、微指令存储器 LPM\_ROM 中的数据等，都实时、直观地显示在 LCD 屏幕上，使学生实时观察每条指令及微指令的执行情况，从而对计算机的原理、结构，从部件到系统，直到计算机整机有一个形象的、生动的、本质的认识。

通过利用 GW48 CP+现代计算机组成原理实验系统的学习，还能使学生在 VHDL 语言、EDA 软件工具和 FPGA 的应用方面获得大量实用的技术。

本实验的先期课程为不少于 20 学时的 EDA 技术课，包括 FPGA 应用技术、MaxplusII/QuartusII 软件、原理图输入设计方法，以及 VHDL 基础。推荐选择用科学出版社的《EDA 技术实用教程》或清华大学出版社的《EDA 技术与 VHDL》一书。

## 实验一 QuartusII EDA 工具与 VHDL 基础实验

本实验中所配的 13 个实验主要作为计算机组成原理实验的前期练习，以便熟悉 VHDL 语言、原理图输入方法，以及 EDA 工具和 EDA 实验系统的使用方法。主要参考清华大学出版社的《EDA 技术与 VHDL》或《现代计算机组成原理》一书。

实验内容和数量可根据实际需要选择，每一个实验，在所配的光盘中都有对应的实验示例和实验指导课件。

### 1-1. 应用 QuartusII 完成基本组合电路设计

《示例程序和实验指导课件位置》:

\Experiments\Expmt1\chpt4\Ep1c6\_41\_mux21A\ 工程 mux21A

(1) 实验目的: 熟悉 Quartus II 的 VHDL 文本设计流程全过程，学习简单组合电路的设计、多层次电路设计、仿真和硬件测试。

(2) 实验内容 1: 首先利用 Quartus II 完成 2 选 1 多路选择器（例 3-3）的文本编辑输入(mux21a.vhd)和仿真测试等步骤，给出图 3-3 所示的仿真波形。最后在实验系统上进行硬件测试，验证本项设计的功能。

(3) 实验内容 2: 将此多路选择器看成是一个元件 mux21a，利用元件例化语句描述图 3-18，并将此文件放在同一目录中。以下是部分参考程序:

```
...  
    COMPONENT MUX21A  
    PORT ( a, b, s : IN STD_LOGIC;  
           y : OUT STD_LOGIC);  
    END COMPONENT ;  
...  
    u1 : MUX21A PORT MAP(a=>a2, b=>a3, s=>s0, y=>tmp);  
    u2 : MUX21A PORT MAP(a=>a1, b=>tmp, s=>s1, y=>outy);  
    END ARCHITECTURE BHV ;
```

【例 3-3】

```
ENTITY mux21a IS  
    PORT ( a, b, s: IN BIT;  
           y : OUT BIT );  
END ENTITY mux21a;  
ARCHITECTURE one OF mux21a IS  
    BEGIN  
        PROCESS (a,b,s)  
        BEGIN  
            IF s = '0' THEN y <= a ; ELSE y <= b ;  
            END IF;  
        END PROCESS;  
    END ARCHITECTURE one ;
```

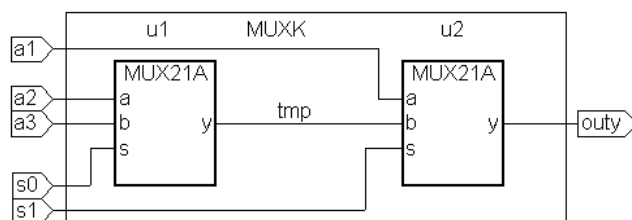
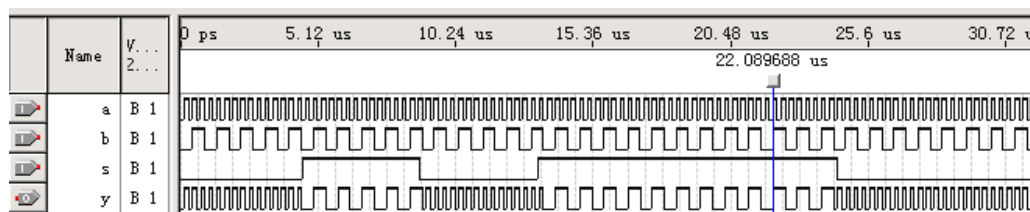


图 3-18 双 2 选 1 多路选择器



3-3 mux21a功能时序波形

按照本章给出的步骤对上例分别进行编译、综合、仿真。并对其仿真波形作出分析说明。

(4) 实验内容 3: 引脚锁定以及硬件下载测试。若选择目标器件是 EP1C3, 建议选实验电路模式 5 (附录图 7), 用键 1(PIO0, 引脚号为 1)控制 s0; 用键 2(PIO1, 引脚号为 2)控制 s1; a3、a2 和 a1 分别接 clock5(引脚号为 16)、clock0(引脚号为 93)和 clock2(引脚号为 17); 输出信号 outy 仍接扬声器 spker(引脚号为 129)。通过短路帽选择 clock0 接 256Hz 信号, clock5 接 1024Hz, clock2 接 8Hz 信号。最后进行编译、下载和硬件测试实验(通过选择键 1、键 2, 控制 s0、s1, 可使扬声器输出不同音调)。

(5) 实验报告: 根据以上的实验内容写出实验报告, 包括程序设计、软件编译、仿真分析、硬件测试及详细实验过程; 给出程序分析报告、仿真波形图及其分析报告。

(6) 附加内容: 根据本实验以上提出的各项实验内容和实验要求, 设计 1 位全加器。首先用 Quartus II 完成 3.3 节给出的全加器的设计, 包括仿真和硬件测试。实验要求分别仿真测试底层硬件或门和半加器, 最后完成顶层文件全加器的设计和测试, 给出设计原程序, 程序分析报告、仿真波形图及其分析报告。

(7) 实验习题: 以 1 位二进制全加器为基本元件, 用例化语句写出 8 位并行二进制全加器的顶层文件, 并讨论此加法器的电路特性。

## 1-2. 应用 QuartusII 完成基本时序电路的设计

(1) 实验目的: 熟悉 Quartus II 的 VHDL 文本设计过程, 学习简单时序电路的设计、仿真和测试。

(2) 实验内容 1: 根据实验 4-1 的步骤和要求, 设计触发器(使用例 3-6), 给出程序设计、软件编译、仿真分析、硬件测试及详细实验过程。

### 【例 3-6】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY DFF1 IS
    PORT (CLK : IN STD_LOGIC ;
          D : IN STD_LOGIC ;
          Q : OUT STD_LOGIC );
END ;
ARCHITECTURE bhv OF DFF1 IS
    SIGNAL Q1 : STD_LOGIC ; --类似于在芯片内部定义一个数据的暂存节点
BEGIN
    PROCESS (CLK,Q1)
    BEGIN
        IF CLK'EVENT AND CLK = '1' THEN Q1 <= D ;
        END IF;
    END PROCESS ;
    Q <= Q1 ; --将内部的暂存数据向端口输出(双横线--是注释符号)
END bhv;
```

(3) 实验内容 2: 设计锁存器(使用例 3-14), 同样给出程序设计、软件编译、仿真分析、硬件测试及详细实验过程。

### 【例 3-14】

```
...
PROCESS (CLK, D) BEGIN
    IF CLK = '1' --电平触发型寄存器
    THEN Q <= D ;
    END IF;
END PROCESS ;
```

(4) 实验内容 3: 只用一个 1 位二进制全加器为基本元件和一些辅助的时序电路, 设计一个 8 位串行二进制全加器, 要求: 1、能在 8-9 个时钟脉冲后完成 8 位二进制数(加数被加数的输入方式为并行)的加法运算, 电路须考虑进位输入 Cin 和进位输出 Cout;

2、给出此电路的时序波形, 讨论其功能, 并就工作速度与并行加法器进行比较;

3、在 FPGA 中进行实测。对于 GW48 EDA 实验系统, 建议选择电路模式 1 (附录图 3), 键 2, 键 1 输入 8 位加数; 键 4, 键 3 输入 8 位被加数; 键 8 作为手动单步时钟输入; 键 7 控制进位输入 Cin; 键 9 控制清 0; 数码 6 和数码 5 显示相加和; 发光管 D1 显示溢出进位 Cout。



4、键 8 作为相加起始控制，同时兼任清 0；工作时钟由 clock0 自动给出，每当键 8 发出一次开始相加命令，电路即自动相加，结束后停止工作，并显示相加结果。就外部端口而言，与纯组合电路 8 位并行加法器相比，此串行加法器仅多出一个加法起始/清 0 控制输入和工作时钟输入端。（提示：此加法器有并/串和串/并移位寄存器各一）。

(5) 实验报告：分析比较实验内容 1 和 2 的仿真和实测结果，说明这两种电路的异同点。详述实验内容 3。

### 1-3. 设计含异步清 0 和同步时钟使能的加法计数器

《示例程序和实验指导课件位置》：\Experiments\ Expmt1\chpt4\Eplc6\_43\_cnt10\ 工程：cnt10

(1) 实验目的：学习计数器的设计、仿真和硬件测试，进一步熟悉 VHDL 设计技术。

(2) 实验原理：实验程序为例 3-22，实验原理参考 3.4 节，设计流程参考本章。

#### 【例 3-22】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT10 IS
    PORT (CLK,RST,EN : IN STD_LOGIC;
          CQ : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
          COUT : OUT STD_LOGIC );
END CNT10;
ARCHITECTURE behav OF CNT10 IS
BEGIN
    PROCESS(CLK, RST, EN)
        VARIABLE CQI : STD_LOGIC_VECTOR(3 DOWNTO 0);
    BEGIN
        IF RST = '1' THEN CQI := (OTHERS =>'0') ; --计数器异步复位
        ELSIF CLK'EVENT AND CLK='1' THEN --检测时钟上升沿
            IF EN = '1' THEN --检测是否允许计数（同步使能）
                IF CQI < 9 THEN CQI := CQI + 1; --允许计数，检测是否小于9
                ELSE CQI := (OTHERS =>'0'); --大于9，计数值清零
            END IF;
        END IF;
        IF CQI = 9 THEN COUT <= '1'; --计数大于9，输出进位信号
        ELSE COUT <= '0';
        END IF;
        CQ <= CQI; --将计数值向端口输出
    END PROCESS;
END behav;
```

(3) 实验内容 1：在 Quartus II 上对例 3-22 进行编辑、编译、综合、适配、仿真。说明例中各语句的作用，详细描述示例的功能特点，给出其所有信号的时序仿真波形。

(4) 实验内容 2：引脚锁定以及硬件下载测试（参考 4.2 节）。引脚锁定后进行编译、下载和硬件测试实验。将实验过程和实验结果写进实验报告。

(5) 实验内容 3：使用 SignalTap II 对此计数器进行实时测试，流程与要求参考 4.3 节。

(6) 实验内容 4：从设计中去除 SignalTap II，要求全程编译后生成用于配置器件 EPCS1 编程的压缩 POF 文件，并使用 ByteBlasterII，通过 AS 模式对实验板上的 EPCS1 进行编程，最后进行验证。

(7) 实验内容 4：为此项设计加入一个可用于 SignalTap II 采样的独立的时钟输入端（采用时钟选择 clock0=12MHz，计数器时钟 CLK 分别选择 256Hz、16384Hz、6MHz），并进行实时测试。

(8) 思考题：在例 3-22 中是否可以不定义信号 CQI，而直接用输出端口信号完成加法运算，即：

CQ <= CQ + 1? 为什么？

(9) 实验报告：将实验原理、设计过程、编译仿真波形和分析结果、硬件测试实验结果写进实验报告。

## 1-4. 7 段数码显示译码器设计

《示例程序和实验指导课件位置》: \Experiments\ Expmt1\chpt4\Ep1c6\_51\_DECL7S\  
工程: DECL7S

(1) 实验目的: 学习 7 段数码显示译码器设计; 学习 VHDL 的 CASE 语句应用及多层次设计方法。

(2) 实验原理: 7 段数码是纯组合电路, 通常的小规模专用 IC, 如 74 或 4000 系列的器件只能作十进制 BCD 码译码, 然而数字系统中的数据处理和运算都是 2 进制的, 所以输出表达都是 16 进制的, 为了满足 16 进制数的译码显示, 最方便的方法就是利用译码程序在 FPGA/CPLD 中来实现。例 5-18 作为 7 段译码器, 输出信号 LED7S 的 7 位分别接如图 5-18 数码管的 7 个段, 高位在左, 低位在右。例如当 LED7S 输出为 “1101101” 时, 数码管的 7 个段: g、f、e、d、c、b、a 分别接 1、1、0、1、1、0、1; 接有高电平的段发亮, 于是数码管显示 “5”。注意, 这里没有考虑表示小数点的发光管, 如果要考虑, 需要增加段 h, 例 5-18 中的 LED7S:OUT STD\_LOGIC\_VECTOR(6 DOWNT0 0)应改为...(7 DOWNT0 0)。

(3) 实验内容 1: 说明例 5-18 中各语句的含义, 以及该例的整体功能。在 QuartusII 上对该例进行编辑、编译、综合、适配、仿真, 给出其所有信号的时序仿真波形。

提示: 用输入总线的方式给出输入信号仿真数据, 仿真波形示例图如图 5-17 所示。

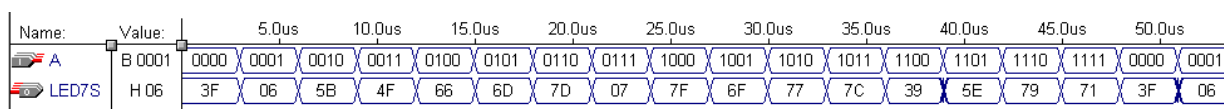


图 5-17 7 段译码器仿真波形

### 【例 5-18】

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
ENTITY DECL7S IS
    PORT ( A : IN STD_LOGIC_VECTOR(3 DOWNT0 0);
          LED7S : OUT STD_LOGIC_VECTOR(6 DOWNT0 0) );
END ;
ARCHITECTURE one OF DECL7S IS
BEGIN
    PROCESS( A )
    BEGIN
        CASE A IS
            WHEN "0000" => LED7S <= "0111111" ;
            WHEN "0001" => LED7S <= "0000110" ;
            WHEN "0010" => LED7S <= "1011011" ;
            WHEN "0011" => LED7S <= "1001111" ;
            WHEN "0100" => LED7S <= "1100110" ;
            WHEN "0101" => LED7S <= "1101101" ;
            WHEN "0110" => LED7S <= "1111101" ;
            WHEN "0111" => LED7S <= "0000111" ;
            WHEN "1000" => LED7S <= "1111111" ;
            WHEN "1001" => LED7S <= "1101111" ;
            WHEN "1010" => LED7S <= "1110111" ;
            WHEN "1011" => LED7S <= "1111100" ;
            WHEN "1100" => LED7S <= "0111001" ;
            WHEN "1101" => LED7S <= "1011110" ;
            WHEN "1110" => LED7S <= "1111001" ;
            WHEN "1111" => LED7S <= "1110001" ;
            WHEN OTHERS => NULL ;
        END CASE ;
    END PROCESS ;
END ;
```

(4) 实验内容 2: 引脚锁定及硬件测试。建议选 GW48 系统的实验电路模式 6 (参考附录图 8), 用数码 8 显示译码输出(PIO46-PIO40), 键 8、键 7、键 6 和键 5 四位控制输入, 硬件验证译码器的工作性能。

(5) 实验内容 3: 用第 3 章介绍的例化语句, 按图 5-19 的方式连接成顶层设计电路 (用 VHDL 表述), 图中的 CNT4B 是一个 4 位二进制加法计数器, 可以由例 3-22 修改获得; 模块 DECL7S 即为例 5-18 实体元件, 重复以上实验过程。注意图 5-19 中的 tmp 是 4 位总线, led 是 7 位总线。对于引脚锁定和实验, 建议

选电路模式 6，用数码 8 显示译码输出，用键 3 作为时钟输入(每按 2 次键为 1 个时钟脉冲)，或直接接时钟信号 clock0。

(6) 实验报告：根据以上的实验内容写出实验报告，包括程序设计、软件编译、仿真分析、硬件测试和实验过程；设计程序、程序分析报告、仿真波形图及其分析报告。

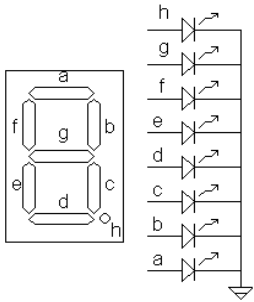


图 5-18 共阴数码管及其电路

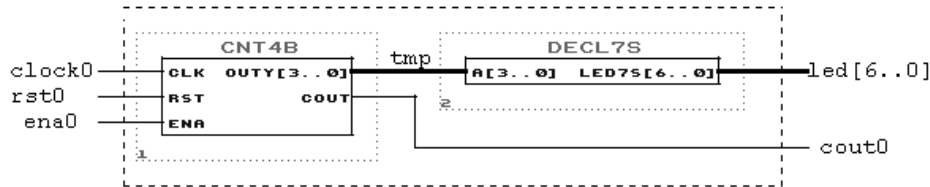


图 5-19 计数器和译码器连接电路的顶层文件原理图

### 1-5. 8 位数码扫描显示电路设计

《示例程序和实验指导课件位置》：\Experiments\Expmt1\chpt5\Ep1c6\_52\_SCAN\

工程：SCAN\_LED

(1) 实验目的：学习硬件扫描显示电路的设计。

(2) 实验原理：图 5-20 所示的是 8 位数码扫描显示电路，其中每个数码管的 8 个段：h、g、f、e、d、c、b、a（h 是小数点）都分别连在一起，8 个数码管分别由 8 个选通信号 k1、k2、...k8 来选择。被选通的数码管显示数据，其余关闭。如在某一时刻，k3 为高电平，其余选通信号为低电平，这时仅 k3 对应的数码管显示来自段信号端的数据，而其它 7 个数码管呈现关闭状态。根据这种电路状况，如果希望在 8 个数码管显示希望的数据，就必须使得 8 个选通信号 k1、k2、...k8 分别被单独选通，并在此同时，在段信号输入口加上希望在该对应数码管上显示的数据，于是随着选通信号的扫变，就能实现扫描显示的目的。

例 5-19 是扫描显示的示例程序，其中 clk 是扫描时钟；SG 为 7 段控制信号，由高位至低位分别接 g、f、e、d、c、b、a 7 个段；BT 是位选控制信号，接图 5-20 中的 8 个选通信号：k1、k2、...k8。程序中 CNT8 是一个 3 位计数器，作扫描计数信号，由进程 P2 生成；进程 P3 是 7 段译码查表输出程序，与例 5-18 相同；进程 P1 是对 8 个数码管选通的扫描程序，例如当 CNT8 等于"001" 时，K2 对应的数码管被选通，同时，A 被赋值 3，再由进程 P3 译码输出"1001111"，显示在数码管上即为“3”；当 CNT8 扫变时，将能在 8 个数码管上显示数据：13579BDF。

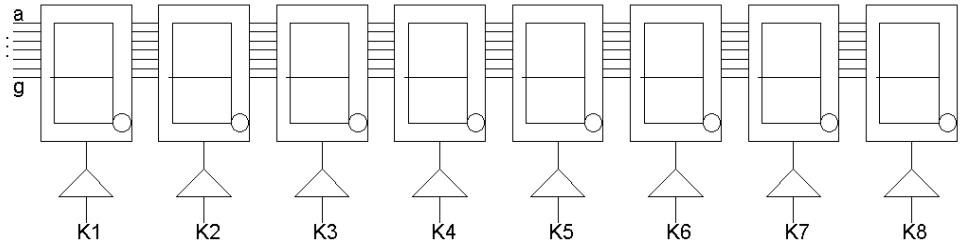


图 5-20 8 位数码扫描显示电路

【例 5-19】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY SCAN_LED IS
    PORT (
        CLK : IN STD_LOGIC;
        SG : OUT STD_LOGIC_VECTOR(6 DOWNTO 0); --段控制信号输出
        BT : OUT STD_LOGIC_VECTOR(7 DOWNTO 0); --位控制信号输出
    );
END;
ARCHITECTURE one OF SCAN_LED IS
    SIGNAL CNT8 : STD_LOGIC_VECTOR(2 DOWNTO 0);
    SIGNAL A : INTEGER RANGE 0 TO 15;
BEGIN
```

```

P1: PROCESS( CNT8 )
BEGIN
    CASE CNT8 IS
        WHEN "000" => BT <= "00000001" ; A <= 1 ;
        WHEN "001" => BT <= "00000010" ; A <= 3 ;
        WHEN "010" => BT <= "00000100" ; A <= 5 ;
        WHEN "011" => BT <= "00001000" ; A <= 7 ;
        WHEN "100" => BT <= "00010000" ; A <= 9 ;
        WHEN "101" => BT <= "00100000" ; A <= 11 ;
        WHEN "110" => BT <= "01000000" ; A <= 13 ;
        WHEN "111" => BT <= "10000000" ; A <= 15 ;
        WHEN OTHERS => NULL ;
    END CASE ;
END PROCESS P1;
P2: PROCESS(CLK)
BEGIN
    IF CLK'EVENT AND CLK = '1' THEN CNT8 <= CNT8 + 1;
    END IF;
END PROCESS P2 ;
P3: PROCESS( A ) --译码电路
BEGIN
    CASE A IS
        WHEN 0 => SG <= "01111111"; WHEN 1 => SG <= "00001110";
        WHEN 2 => SG <= "10110111"; WHEN 3 => SG <= "10011111";
        WHEN 4 => SG <= "11001110"; WHEN 5 => SG <= "11011101";
        WHEN 6 => SG <= "11111101"; WHEN 7 => SG <= "00001111";
        WHEN 8 => SG <= "11111111"; WHEN 9 => SG <= "11011111";
        WHEN 10 => SG <= "11101111"; WHEN 11 => SG <= "11111100";
        WHEN 12 => SG <= "01110011"; WHEN 13 => SG <= "10111110";
        WHEN 14 => SG <= "11110011"; WHEN 15 => SG <= "11100011";
        WHEN OTHERS => NULL ;
    END CASE ;
END PROCESS P3;
END;

```

(3) 实验内容 1: 说明例 5-19 中各语句的含义, 以及该例的整体功能。对该例进行编辑、编译、综合、适配、仿真, 给出仿真波形。实验方式: 若考虑小数点, SG 的 8 个段分别与 PIO49、PIO48、...、PIO42 (高位在左)、BT 的 8 个位分别与 PIO34、PIO35、...、PIO41 (高位在左); 电路模式不限, 引脚图参考附录图 12。将 GW48EDA 系统左下方的拨码开关全部向上拨, 这时实验系统的 8 个数码管构成图 5-20 的电路结构, 时钟 CLK 可选择 clock0, 通过跳线选择 16384Hz 信号。引脚锁定后进行编译、下载和硬件测试实验。将实验过程和实验结果写进实验报告。

(4) 实验内容 2: 修改例 5-19 的进程 P1 中的显示数据直接给出的方式, 增加 8 个 4 位锁存器, 作为显示数据缓冲器, 使得所有 8 个显示数据都必须来自缓冲器。缓冲器中的数据可以通过不同方式锁入, 如来自 A/D 采样的数据、来自分时锁入的数据、来自串行方式输入的数据, 或来自单片机等。

## 1-6. 数控分频器的设计

《示例程序和实验指导课件位置》: \Experiments\ Expmt1\chpt5\Ep1c6\_53\_DVF\

工程: DVF

(1) 实验目的: 学习数控分频器的设计、分析和测试方法。

(2) 实验原理: 数控分频器的功能就是当在输入端给定不同输入数据时, 将对输入的时钟信号有不同的分频比, 数控分频器就是用计数值可并行预置的加法计数器设计完成的, 方法是将计数溢出位与预置数加载输入信号相接即可, 详细设计程序如例 5-20 所示。

(3) 分析: 根据图 5-21 的波形提示, 分析例 5-20 中的各语句功能、设计原理及逻辑功能, 详述进程 P\_REG 和 P\_DIV 的作用, 并画出该程序的 RTL 电路图。

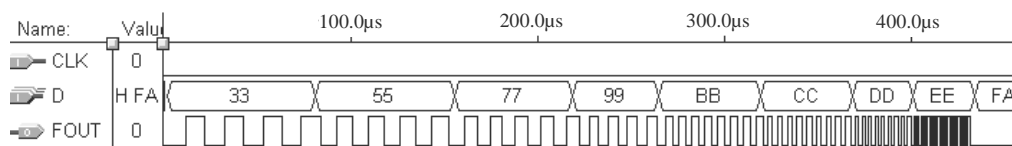


图 5-21 当给出不同输入值 D 时, FOUT 输出不同频率(CLK 周期=50ns)

(4) 仿真：输入不同的 CLK 频率和预置值 D，给出如图 5-21 的时序波形。

(5) 实验内容 1：在实验系统上硬件验证例 5-20 的功能。可选实验电路模式 1（参考附录图 3）；键 2/键 1 负责输入 8 位预置数 D(PIO7-PIO0)；CLK 由 clock0 输入，频率选 65536Hz 或更高(确保分频后落在音频范围)；输出 FOUT 接扬声器(SPKER)。编译下载后进行硬件测试：改变键 2/键 1 的输入值，可听到不同音调的声音。

(6) 实验内容 2：将例 5-20 扩展成 16 位分频器，并提出此项设计的实用示例，如 PWM 的设计等。

(7) 思考题：怎样利用 2 个由例 5-20 给出的模块设计一个电路，使其输出方波的正负脉宽的宽度分别由两个 8 位输入数据控制？

(8) 实验报告：根据以上的要求，将实验项目分析设计，仿真和测试写入实验报告。

#### 【例 5-20】

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY DVF IS
    PORT ( CLK : IN STD_LOGIC;
           D : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
           FOUT : OUT STD_LOGIC );
END;
ARCHITECTURE one OF DVF IS
    SIGNAL FULL : STD_LOGIC;
BEGIN
    P_REG: PROCESS(CLK)
        VARIABLE CNT8 : STD_LOGIC_VECTOR(7 DOWNTO 0);
    BEGIN
        IF CLK'EVENT AND CLK = '1' THEN
            IF CNT8 = "11111111" THEN
                CNT8 := D; --当CNT8计数计满时，输入数据D被同步预置给计数器CNT8
                FULL <= '1'; --同时使溢出标志信号FULL输出为高电平
            ELSE
                CNT8 := CNT8 + 1; --否则继续作加1计数
                FULL <= '0'; --且输出溢出标志信号FULL为低电平
            END IF;
        END IF;
    END PROCESS P_REG ;
    P_DIV: PROCESS(FULL)
        VARIABLE CNT2 : STD_LOGIC;
    BEGIN
        IF FULL'EVENT AND FULL = '1' THEN
            CNT2 := NOT CNT2; --如果溢出标志信号FULL为高电平，D触发器输出取反
            IF CNT2 = '1' THEN FOUT <= '1'; ELSE FOUT <= '0';
        END IF;
    END IF;
    END PROCESS P_DIV ;
END;
```

## 1-7. 32 位并进/并出移位寄存器设计

仅用例 5-8 一个 8 位移位寄存器，再增加一些电路，如 4 个 8 位锁存器等，设计成为一个能为 32 位二进制数进行不同方式移位的移位寄存器。这个电路模型十分容易用到 CPU 的设计中。

## 1-8. 在 QuartusII 中用原理图输入法设计 8 位全加器

(1) 实验目的：熟悉利用 Quartus II 的原理图输入方法设计简单组合电路，掌握层次化设计的方法，并通过一个 8 位全加器的设计把握利用 EDA 软件进行原理图输入方式的电子线路设计的详细流程。

(2) 实验原理：一个 8 位全加器可以由 8 个 1 位全加器构成，加法器间的进位可以串行方式实现，即将低位加法器的进位输出 cout 与相邻的高位加法器的最低进位输入信号 cin 相接。而一个 1 位全加器可以按照 6.1 节的方法来完成。

(3) 实验内容 1：按照 6.1 节介绍的方法与流程，完成半加器和全加器的设计，包括原理图输入、编译、综合、适配、仿真、实验板上的硬件测试，并将此全加器电路设置成一个硬件符号入库。键 1、键 2、键

3(PIO0/1/2)分别接 ain、bin、cin；发光管 D2、D1(PIO9/8)分别接 sum 和 cout。

(4) 实验内容 2：建立一个更高层次的原理图设计，利用以上获得的 1 位全加器构成 8 位全加器，并完成编译、综合、适配、仿真和硬件测试。建议选择电路模式 1（附录图 3）；键 2、键 1 输入 8 位加数；键 4、键 3 输入 8 位被加数；数码 6/5 显示加和；D8 显示进位 cout。

(5) 实验报告：详细叙述 8 位加法器的设计流程；给出各层次的原理图及其对应的仿真波形图；给出加法器的时序分析情况；最后给出硬件测试流程和结果。

## 1-9. 在 QuartusII 中用原理图输入法设计较复杂数字系统

(1) 实验目的：熟悉原理图输入法中 74 系列等宏功能元件的使用方法，掌握更复杂的原理图层次化设计技术和数字系统设计方法。完成 8 位十进制频率计的设计。

(2) 原理说明：利用 6.2 节介绍的 2 位计数器模块，连接它们的计数进位，用 4 个计数模块就能完成一个 8 位有时钟使能的计数器；对于测频控制器的控制信号，在仿真过程中应该注意它们可能的毛刺现象。最后按照 6.2 节中的设计流程和方法即可完成全部设计。

(3) 实验内容：首先完成 2 位频率计的设计，然后进行硬件测试，建议选择电路模式 2；数码 2 和 1 显示输出频率值，待测频率 F\_IN 接 clock0；测频控制时钟 CLK 接 clock2，若选择 clock2 = 8Hz，门控信号 CNT\_EN 的脉宽恰好为 1 秒。然后建立一个新的原理图设计层次，在此基础上将其扩展为 8 位频率计，仿真测试该频率计待测信号的最高频率，并与硬件实测的结果进行比较。

(4) 实验报告：给出各层次的原理图、工作原理、仿真波形图和分析，详述硬件实验过程和实验结果。

## 1-10. 用 QuartusII 设计正弦信号发生器

《示例程序和实验指导课件位置》：\Experiments\ Expmt1\chpt7\Ep1c6\_71\_SINGT\  
工程：SINGT

(1) 实验目的：进一步熟悉 QuartusII 及其 LPM\_ROM 与 FPGA 硬件资源的使用方法。

(2) 实验原理：参考本章相关内容。

(3) 实验内容 1：根据例 7-4，在 Quartus II 上完成正弦信号发生器设计，包括仿真和资源利用情况了解（假设利用 Cyclone 器件）。最后在实验系统上实测，包括 SignalTap II 测试、FPGA 中 ROM 的在系统数据读写测试和利用示波器测试。最后完成 EPCS1 配置器件的编程。

【例 7-4】 正弦信号发生器顶层设计

```
LIBRARY IEEE; --正弦信号发生器源文件
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY SINGT IS
PORT ( CLK : IN STD_LOGIC; --信号源时钟
      DOUT : OUT STD_LOGIC_VECTOR (7 DOWNT0 0) );--8 位波形数据输出
END;
ARCHITECTURE DACC OF SINGT IS
COMPONENT data_rom --调用波形数据存储器 LPM_ROM 文件：data_rom.vhd 声明
PORT(address : IN STD_LOGIC_VECTOR (5 DOWNT0 0));--6 位地址信号
      inclock : IN STD_LOGIC ;--地址锁存时钟
      q : OUT STD_LOGIC_VECTOR (7 DOWNT0 0) );
END COMPONENT;
SIGNAL Q1 : STD_LOGIC_VECTOR (5 DOWNT0 0); --设定内部节点作为地址计数器
BEGIN
PROCESS(CLK ) --LPM_ROM 地址发生器进程
```

```

BEGIN
    IF CLK'EVENT AND CLK = '1' THEN Q1<=Q1+1; --Q1 作为地址发生器计数器
    END IF;
END PROCESS;

u1 : data_rom PORT MAP(address=>Q1, q => DOUT,inclock=>CLK);--例化
END;

```

信号输出的 D/A 使用实验系统上的 DAC0832，注意其转换速率是  $1\mu s$ ，其引脚功能简述如下：

**ILE**：数据锁存允许信号，高电平有效，系统板上已直接连在 +5V 上；**WR1**、**WR2**：写信号 1、2，低电平有效；**XFER**：数据传送控制信号，低电平有效；**VREF**：基准电压，可正可负， $-10V \sim +10V$ ；**RFB**：反馈电阻端；**IOUT1**/**IOUT2**：电流输出端。D/A 转换量是以电流形式输出的，所以必须将电流信号变为电压信号；**AGND**/**DGND**：模拟地与数字地。在高速情况下，此二地的连接线必须尽可能短，且系统的单点接地点须接在此连线的某一点上。

建议选择 GW48 系统的电路模式 No.5，由附录对应的电路图可见，DAC0832 的 8 位数据口 D[7..0] 分别与 FPGA 的 PIO31、30..、24 相连，如果目标器件是 EP1C3T144，则对应的引脚是：72、71、70、69、68、67、52、51；时钟 CLK 接系统的 clock0，对应的引脚是 93，选择的时钟频率不能太高（转换速率  $1\mu s$ ，）。还应该注意，DAC0832 电路须接有  $\pm 12V$  电压：GW48 系统的  $\pm 12V$  电源开关在系统左侧上方。然后下载 SINGT.sof 到 FPGA 中；波形输出在系统左下角，将示波器的地与 GW48 系统的地（GND）相接，信号端与“AOUT”信号输出端相接。如果希望对输出信号进行滤波，将 GW48 系统左下角的拨码开关的“8”向下拨，则波形滤波输出，向上拨则未滤波输出，这可从输出的波形看出。

基本步骤如下（详细步骤可参考该书第 4 章）：

## 一、顶层文件设计

### 1 创建工程和编辑设计文件

正弦信号发生器的结构由 3 部分组成（图 3-1）：数据计数器或地址发生器、数据 ROM 和 D/A。性能良好的正弦信号发生器的设计要求此 3 部分具有高速性能，且数据 ROM 在高速条件下，占用最少的逻辑资源，设计流程最便捷，波形数据获最方便。图 3-1 所示是此信号发生器结构图，顶层文件 SINGT.VHD 在 FPGA 中实现，包含 2 个部分：ROM 的地址信号发生器由 5 位计数器担任，和正弦数据 ROM，拒此，ROM 由 LPM\_ROM 模块构成能达到最优设计，LPM\_ROM 底层是 FPGA 中的 EAB 或 ESB 等。地址发生器的时钟 CLK 的输入频率  $f_0$  与每周期的波形数据点数（在此选择 64 点），以及 D/A 输出的频率  $f$  的关系是：

$$f = f_0 / 64$$

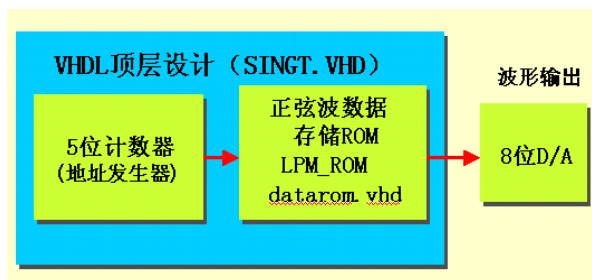


图 3-1 正弦信号发生器结构图

### 2 创建工程

### 3 编译前设置

在对工程进行编译处理前，必须作好必要的设置。具体步骤如下：1、选择目标芯片；2、选择目标器件编程配置方式；3、选择输出配置；

### 4 编译及了解编译结果

### 5、正弦信号数据 ROM 定制（包括设计 ROM 初始化数据文件）

另两种方法要快捷的多，可分别用 C 程序生成同样格式的初始化文件和使用 DSP Builder/MATLAB 来生成。

### 6 仿真

### 7 引脚锁定、下载和硬件测试

### 8 使用嵌入式逻辑分析仪进行实时测试

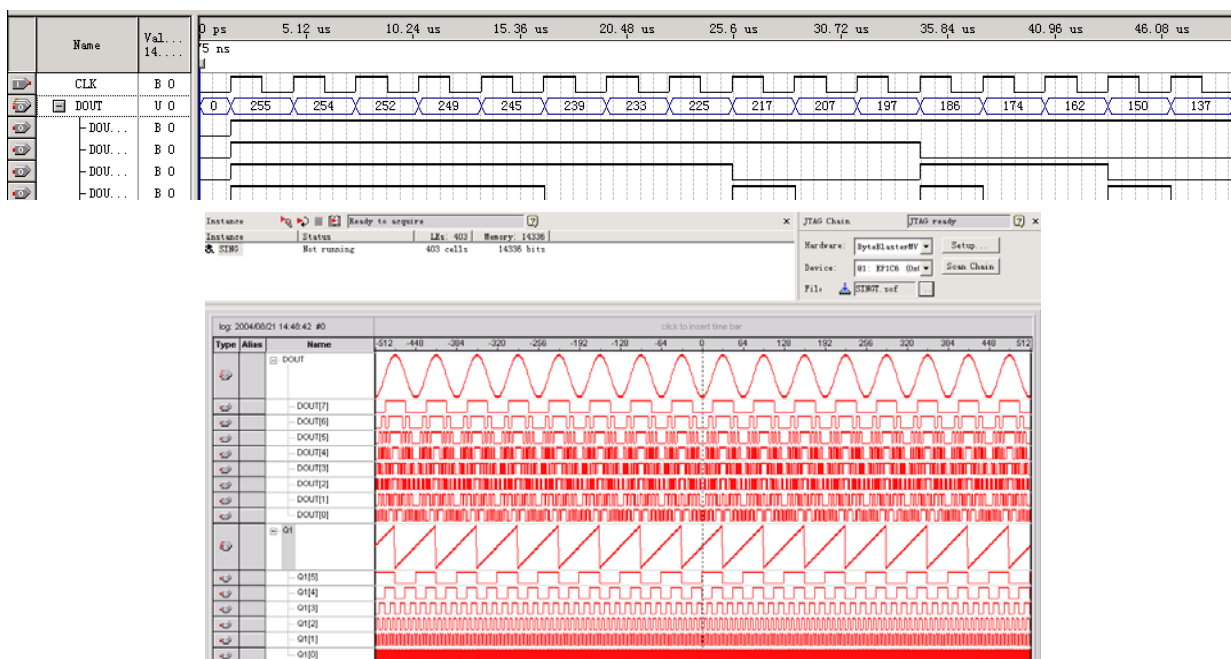


图 3-33 SignalTapII 数据窗的实时信号

9 对配置器件 EPCS4/EPCS1 编程

10 了解此工程的 RTL 电路

图

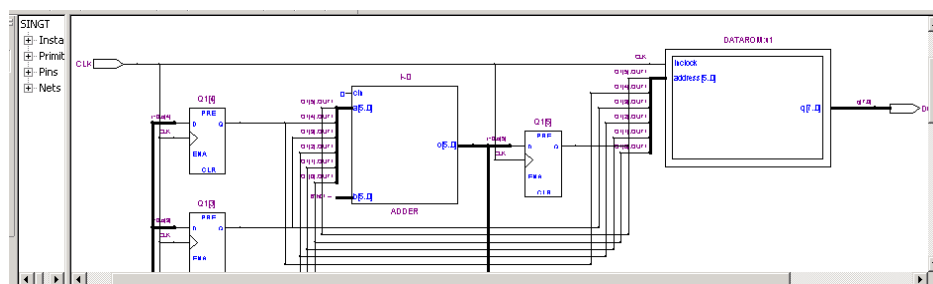


图 3-35 工程 singt 的 RTL 电路图

实验内容 2: 修改例 7-3 的数据 ROM 文件, 设其数据线宽度为 8, 地址线宽度也为 8, 初始化数据文件使用 MIF 格式, 用 C 程序产生正弦信号数据, 最后完成以上相同的实验。

实验内容 3: 设计一任意波形信号发生器, 可以使用 LPM 双口 RAM 担任波形数据存储, 利用单片机产生所需要的波形数据, 然后输向 FPGA 中的 RAM (可以利用 GW48 系统上与 FPGA 接口的单片机完成此实验, D/A 可利用系统上配置的 0832 或 5651 高速器件)。

实验报告: 根据以上的实验内容写出实验报告, 包括设计原理、程序设计、程序分析、仿真分析、硬件测试和详细实验过程。

## 1-11. 8 位 16 进制频率计设计

《示例程序和实验指导课件位置》: \Experiments\ Expm1\chpt7\Ep1c6\_72\_FREQTEST\  
工程: FREQTEST

(1) 实验目的: 设计 8 位 16 进制频率计, 学习较复杂的数字系统设计方法。

(2) 实验原理: 根据频率的定义和频率测量的基本原理, 测定信号的频率必须有一个脉宽为 1 秒的输入信号脉冲计数允许的信号; 1 秒计数结束后, 计数值被锁入锁存器, 计数器清 0, 为下一测频计数周期作好准备。测频控制信号可以由一个独立的发生器来产生, 即图 7-34 中的 FTCTRL。根据测频原理, 测频控制时序可以如图 7-33 所示。

设计任务是: FTCTRL 的计数使能信号 CNT\_EN 能产生一个 1 秒脉宽的周期信号, 并对频率计中的 32 位二进制计数器 COUNTER32B (图 7-34) 的 ENABL 使能端进行同步控制。当 CNT\_EN 高电平时允许计数;



低电平时停止计数，并保持其所计的脉冲数。在停止计数期间，首先需要有一个锁存信号 **LOAD** 的上跳沿将计数器在前 1 秒钟的计数值锁存进锁存器 **REG32B** 中，并由外部的 16 进制 7 段译码器译出，显示计数值。设置锁存器的好处是数据显示稳定，不会由于周期性的清 0 信号而不断闪烁。锁存信号后，必须有一清 0 信号 **RST\_CNT** 对计数器进行清零，为下 1 秒的计数操作作准备。

(3) 实验内容 1：分别仿真测试模块例 7-7、例 7-8 和例 7-9，再结合例 7-10 完成频率计的完整设计和硬件实现，并给出其测频时序波形及其分析。建议选实验电路模式 5；8 个数码管以 16 进制形式显示测频输出；待测频率输入 **FIN** 由 **clock0** 输入，频率可选 4Hz、256Hz、3Hz...50MHz 等；1Hz 测频控制信号 **CLK1HZ** 可由 **clock2** 输入(用跳线选 1Hz)。注意，这时 8 个数码管的测频显示值是 16 进制的。

(4) 实验内容 2：参考例 3-22，将频率计改为 8 位 10 进制频率计，注意此设计电路的计数器必须是 8 个 4 位的 10 进制计数器，而不是 1 个。此外注意在测频速度上给予优化。

(5) 实验内容 3：用 **LPM** 模块取代例 7-8 和例 7-9，再完成同样的设计任务。

(6) 实验内容 4：用嵌入式锁相环 **PLL** 的 **LPM** 模块对实验系统的 50MHz 或 20MHz 时钟源分频，**PLL** 的输出信号作为频率计的待测信号。注意 **PLL** 的输入时钟必须是器件的专用时钟输入脚 **CLK**→**pin16** (**clock5**) 或 **pin17** (**clock2**)，且输入频率不能低于 16MHz。(实验中可以将 50MHz 频率用线引向 **clock2**，但要拔除其上的短路帽)

(7) 实验报告：给出频率计设计的完整实验报告。

#### 【例7-7】

```
LIBRARY IEEE; --测频控制电路
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY FTCTRL IS
    PORT (CLKK : IN STD_LOGIC;           -- 1Hz
          CNT_EN : OUT STD_LOGIC;        -- 计数器时钟使能
          RST_CNT : OUT STD_LOGIC;       -- 计数器清零
          Load : OUT STD_LOGIC );       -- 输出锁存信号
END FTCTRL;
ARCHITECTURE behav OF FTCTRL IS
    SIGNAL Div2CLK : STD_LOGIC;
BEGIN
    PROCESS( CLKK )
    BEGIN
        IF CLKK'EVENT AND CLKK = '1' THEN -- 1Hz时钟2分频
            Div2CLK <= NOT Div2CLK;
        END IF;
    END PROCESS;
    PROCESS (CLKK, Div2CLK)
    BEGIN
        IF CLKK='0' AND Div2CLK='0' THEN RST_CNT<='1';-- 产生计数器清零信号
        ELSE RST_CNT <= '0'; END IF;
    END PROCESS;
    Load <= NOT Div2CLK;    CNT_EN <= Div2CLK;
END behav;
```

#### 【例7-8】

```
LIBRARY IEEE; --32位锁存器
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY REG32B IS
    PORT ( LK : IN STD_LOGIC;
          DIN : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
          DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) );
END REG32B;
ARCHITECTURE behav OF REG32B IS
BEGIN
    PROCESS(LK, DIN)
    BEGIN
        IF LK'EVENT AND LK = '1' THEN DOUT <= DIN;
        END IF;
    END PROCESS;
END behav;
```

```

    END PROCESS;
END behav;

【例7-9】
LIBRARY IEEE;  --32位计数器
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY COUNTER32B IS
    PORT (FIN : IN STD_LOGIC;           -- 时钟信号
          CLR : IN STD_LOGIC;           -- 清零信号
          ENABL : IN STD_LOGIC;         -- 计数使能信号
          DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)); -- 计数结果
END COUNTER32B;
ARCHITECTURE behav OF COUNTER32B IS
    SIGNAL CQI : STD_LOGIC_VECTOR(31 DOWNTO 0);
BEGIN
    PROCESS(FIN, CLR, ENABL)
    BEGIN
        IF CLR = '1' THEN CQI <= (OTHERS=>'0'); -- 清零
        ELSIF FIN'EVENT AND FIN = '1' THEN
            IF ENABL = '1' THEN CQI <= CQI + 1; END IF;
        END IF;
    END PROCESS;
    DOUT <= CQI;
END behav;

```

```

【例7-10】
LIBRARY IEEE;  --频率计顶层文件
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY FREQTEST IS
    PORT ( CLK1HZ : IN STD_LOGIC;
          FSIN : IN STD_LOGIC;
          DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) );
END FREQTEST;
ARCHITECTURE struc OF FREQTEST IS
    COMPONENT FTCTRL
        PORT (CLKK : IN STD_LOGIC;           -- 1Hz
              CNT_EN : OUT STD_LOGIC;        -- 计数器时钟使能
              RST_CNT : OUT STD_LOGIC;       -- 计数器清零
              Load : OUT STD_LOGIC );        -- 输出锁存信号
    END COMPONENT;
    COMPONENT COUNTER32B
        PORT (FIN : IN STD_LOGIC;           -- 时钟信号
              CLR : IN STD_LOGIC;           -- 清零信号
              ENABL : IN STD_LOGIC;         -- 计数使能信号
              DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)); -- 计数结果
    END COMPONENT;
    COMPONENT REG32B
        PORT ( LK : IN STD_LOGIC;
              DIN : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
              DOUT : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) );
    END COMPONENT;
    SIGNAL TSTEN1 : STD_LOGIC;
    SIGNAL CLR_CNT1 : STD_LOGIC;
    SIGNAL Load1 : STD_LOGIC;
    SIGNAL DT01 : STD_LOGIC_VECTOR(31 DOWNTO 0);
    SIGNAL CARRY_OUT1 : STD_LOGIC_VECTOR(6 DOWNTO 0);
BEGIN
    U1 : FTCTRL PORT MAP(CLKK =>CLK1HZ,CNT_EN=>TSTEN1,
                        RST_CNT =>CLR_CNT1,Load =>Load1);
    U2 : REG32B PORT MAP( LK => Load1, DIN=>DT01, DOUT => DOUT);

```

```

U3 : COUNTER32B PORT MAP( FIN => FSIN, CLR => CLR_CNT1,
                          ENABL => TSTEN1, DOUT=>DTO1 );
END struc;

```

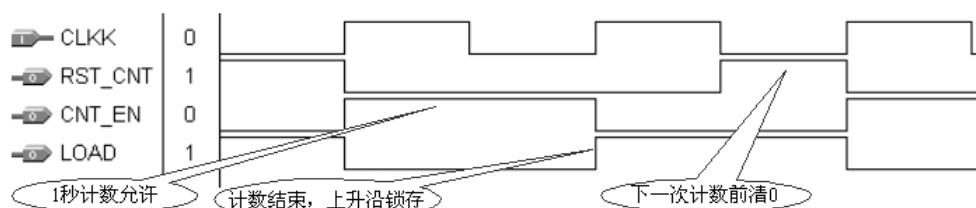


图7-33 频率计测频控制器FTCTRL测控时序图

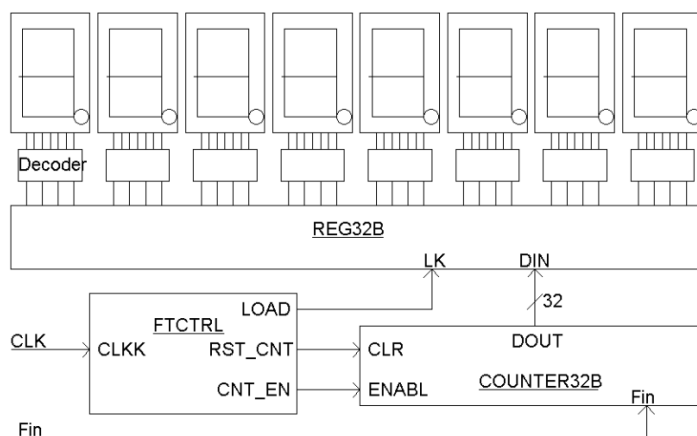


图 7-34 频率计电路框图

## 1-12. 序列检测器设计

**《示例程序和实验指导课件位置》：** \Experiments\ Expmt1\chpt8\Ep1c3\_81\_SCHK\  
**工程：** SCHK

(1) 实验目的：用状态机实现序列检测器的设计，了解一般状态机的设计与应用。

(2) 实验原理：序列检测器可用于检测一组或多组由二进制码组成的脉冲序列信号，当序列检测器连续收到一组串行二进制码后，如果这组码与检测器中预先设置的码相同，则输出 1，否则输出 0。由于这种检测的关键在于正确码的收到必须是连续的，这就要求检测器必须记住前一次的正确码及正确序列，直到在连续的检测中所收到的每一位码都与预置数的对应码相同。在检测过程中，任何一位不相等都将回到初始状态重新开始检测。例 8-11 描述的电路完成对序列数“11100101”的检测，当这一串序列数高位在前(左移)串行进入检测器后，若此数与预置的密码数相同，则输出“A”，否则仍然输出“B”。

(3) 实验内容 1：利用 QuartusII 对例 8-11 进行文本编辑输入、仿真测试并给出仿真波形，了解控制信号的时序，最后进行引脚锁定并完成硬件测试实验。建议选择电路模式 No.8（附录图 10），用键 7(PIO11)控制复位信号 CLR；键 6(PIO9)控制状态机工作时钟 CLK；待检测串行序列数输入 DIN 接 PIO10(左移，最高位在前)；指示输出 AB 接 PIO39~PIO36(显示于数码管 6)。下载后：①按实验板“系统复位”键；②用键 2 和键 1 输入 2 位十六进制待测序列数“11100101”；③按键 7 复位(平时数码 6 指示显“B”)；④按键 6(CLK) 8 次，这时若串行输入的 8 位二进制序列码(显示于数码 2/1 和发光管 D8~D0)与预置码“11100101”相同，则数码 6 应从原来的 B 变成 A，表示序列检测正确，否则仍为 B。

(4) 实验内容 2：根据习题 8-3 中的要求 3 提出的设计方案，重复以上实验内容(将 8 位待检测预置数由键 4/键 3 作为外部输入，从而可随时改变检测密码)。

(5) 实验思考题：如果待检测预置数必须以右移方式进入序列检测器，写出该检测器的 VHDL 代码(两进程符号化有限状态机)，并提出测试该序列检测器的实验方案。

(6) 实验报告：根据以上的实验内容写出实验报告，包括设计原理、程序设计、程序分析、仿真分析、

硬件测试和详细实验过程。

**【例 8-11】**

```
LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY SCHK IS
    PORT(DIN, CLK, CLR : IN STD_LOGIC; --串行输入数据位/工作时钟/复位信号
          AB : OUT STD_LOGIC_VECTOR(3 DOWNTO 0)); --检测结果输出
END SCHK;
ARCHITECTURE behav OF SCHK IS
    SIGNAL Q : INTEGER RANGE 0 TO 8 ;
    SIGNAL D : STD_LOGIC_VECTOR(7 DOWNTO 0); --8位待检测预置数(密码=E5H)
BEGIN
    D <= "11100101 " ; --8位待检测预置数
    PROCESS( CLK, CLR )
    BEGIN
        IF CLR = '1' THEN      Q <= 0 ;
        ELSIF CLK'EVENT AND CLK='1' THEN --时钟到来时, 判断并处理当前输入的位
        CASE Q IS
            WHEN 0=> IF DIN = D(7) THEN Q <= 1 ; ELSE Q <= 0 ; END IF ;
            WHEN 1=> IF DIN = D(6) THEN Q <= 2 ; ELSE Q <= 0 ; END IF ;
            WHEN 2=> IF DIN = D(5) THEN Q <= 3 ; ELSE Q <= 0 ; END IF ;
            WHEN 3=> IF DIN = D(4) THEN Q <= 4 ; ELSE Q <= 0 ; END IF ;
            WHEN 4=> IF DIN = D(3) THEN Q <= 5 ; ELSE Q <= 0 ; END IF ;
            WHEN 5=> IF DIN = D(2) THEN Q <= 6 ; ELSE Q <= 0 ; END IF ;
            WHEN 6=> IF DIN = D(1) THEN Q <= 7 ; ELSE Q <= 0 ; END IF ;
            WHEN 7=> IF DIN = D(0) THEN Q <= 8 ; ELSE Q <= 0 ; END IF ;
            WHEN OTHERS => Q <= 0 ;
        END CASE ;
        END IF ;
    END PROCESS ;
    PROCESS( Q )
    BEGIN
        IF Q = 8 THEN AB <= "1010" ; --序列数检测正确, 输出 “A”
        ELSE          AB <= "1011" ; --序列数检测错误, 输出 “B”
        END IF ;
    END PROCESS ;
END behav ;
```

## 1-13. VHDL 状态机 A/D 采样控制电路实现

《示例程序和实验指导课件位置》: \Experiments\ Expmt1\chpt8\Ep1c3\_82\_ADCINT\

工程: ADCINT

(1) 实验目的: 学习用状态机对 A/D 转换器 ADC0809 的采样控制电路的实现。

(2) 实验原理: ADC0809 的采样控制原理已在 8.2.1 节中作了详细说明(实验程序用例 8-2)。

ADC0809 是 CMOS 的 8 位 A/D 转换器, 片内有 8 路模拟开关, 可控制 8 个模拟量中的一个进入转换器中。转换时间约 100 $\mu$ s, 含锁存控制的 8 路多路开关, 输出有三态缓冲器控制, 单 5V 电源供电。

主要控制信号如图 8-3 所示: START 是转换启动信号, 高电平有效; ALE 是 3 位通道选择地址(ADDC、ADDB、ADDA)信号的锁存信号。当模拟量送至某一输入端(如 IN1 或 IN2 等), 由 3 位地址信号选择, 而地址信号由 ALE 锁存; EOC 是转换情况状态信号, 当启动转换约 100 $\mu$ s 后, EOC 产生一个负脉冲, 以示转换结束; 在 EOC 的上升沿后, 若使输出使能信号 OE 为高电平, 则控制打开三态缓冲器, 把转换好的 8 位数据结果输至数据总线, 至此 ADC0809 的一次转换结束。

**【例 8-2】**

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ADCINT IS
    PORT(D : IN STD_LOGIC_VECTOR(7 DOWNTO 0); --来自0809转换好的8位数据
          CLK : IN STD_LOGIC; --状态机工作时钟
          EOC : IN STD_LOGIC; --转换状态指示, 低电平表示正在转换
          ALE : OUT STD_LOGIC; --8个模拟信号通道地址锁存信号
          START : OUT STD_LOGIC; --转换开始信号
          OE : OUT STD_LOGIC; --数据输出3态控制信号
          ADDA : OUT STD_LOGIC; --信号通道最低位控制信号
          LOCK0 : OUT STD_LOGIC; --观察数据锁存时钟
```

```

        Q : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)); --8位数据输出
END ADCINT;
ARCHITECTURE behav OF ADCINT IS
TYPE states IS (st0, st1, st2, st3,st4) ; --定义各状态子类型
    SIGNAL current_state, next_state: states :=st0 ;
    SIGNAL REGL      : STD_LOGIC_VECTOR(7 DOWNTO 0);
    SIGNAL LOCK      : STD_LOGIC; -- 转换后数据输出锁存时钟信号
BEGIN
    ADDA <= '1';--当ADDA<='0', 模拟信号进入通道IN0; 当ADDA<='1', 则进入通道IN1
        Q <= REGL; LOCK0 <= LOCK ;
    COM: PROCESS(current_state,EOC) BEGIN --规定各状态转换方式
        CASE current_state IS
            WHEN st0=>ALE<='0';START<='0';LOCK<='0';OE<='0'; next_state <= st1; --0809初始化
            WHEN st1=>ALE<='1';START<='1';LOCK<='0';OE<='0'; next_state <= st2; --启动采样
            WHEN st2=> ALE<='0';START<='0';LOCK<='0';OE<='0';
                IF (EOC='1') THEN next_state <= st3; --EOC=1表明转换结束
                ELSE next_state <= st2; END IF ; --转换未结束, 继续等待
            WHEN st3=> ALE<='0';START<='0';LOCK<='0';OE<='1'; next_state <= st4;--开启OE,输出转换好
的数据
            WHEN st4=> ALE<='0';START<='0';LOCK<='1';OE<='1'; next_state <= st0;
            WHEN OTHERS => next_state <= st0;
        END CASE ;
    END PROCESS COM ;
    REG: PROCESS (CLK)
    BEGIN
        IF (CLK'EVENT AND CLK='1') THEN current_state<=next_state; END IF;
    END PROCESS REG ; -- 由信号current_state将当前状态值带出此进程:REG
    LATCH1: PROCESS (LOCK) -- 此进程中, 在LOCK的上升沿, 将转换好的数据锁入
    BEGIN
        IF LOCK='1' AND LOCK'EVENT THEN REGL <= D ; END IF;
    END PROCESS LATCH1 ;
END behav;

```

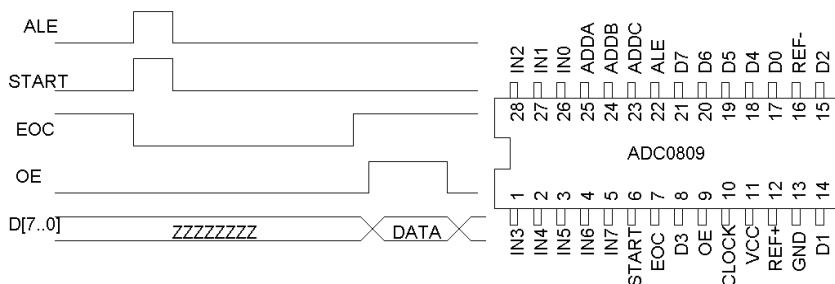


图8-3 ADC0809工作时序

(3) 实验内容：利用 QuartusII 对例 8-2 进行文本编辑输入和仿真测试；给出仿真波形。最后进行引脚锁定并进行测试，硬件验证例 8-2 电路对 ADC0809 的控制功能。

测试步骤：建议选择电路模式 No.5，由对应的电路图可见，ADC0809 的转换时钟 CLK 已经事先接有 750kHz 的频率，引脚锁定为：START 接 PIO34，OE（ENABLE）接 PIO35，EOC 接 PIO8，ALE 接 PIO33，状态机时钟 CLK 接 clock0，ADDA 接 PIO32(ADDB 和 ADDC 都接 GND)，ADC0809 的 8 位输出数据线接 PIO23~PIO16，锁存输出 Q 显示于数码 8/数码 7(PIO47~PIO40)。

实验操作：将 GW48 EDA 系统左下角的拨码开关的 4、6、7 向下拨，其余向上，即使 0809 工作使能，及使 FPGA 能接受来自 0809 转换结束的信号（对于 GW48-CK 系统，左下角选择插针处的“转换结束”和“A/D 使能”用二短路帽短接）。下载 ADC0809 中的 ADCINT.sof 到实验板的 FPGA 中；clock0 的短路帽接可选 12MHz、6MHz、65536Hz 等频率；按动一次右侧的复位键；用螺丝刀旋转 GW48 系统左下角的精密电位器，以便为 ADC0809 提供变化的待测模拟信号（注意，这时必须在例 8-2 中赋值：ADDA <= '1'，这样就能通过实验系统左下的 AIN1 输入端与电位器相接，并将信号输入 0809 的 IN1 端）。这时数码管 8 和 7 将显示 ADC0809 采样的数字值（16 进制），数据来自 FPGA 的输出。数码管 2 和 1 也将显示同样数据，此数据直接来自 0809 的数据口。实验结束后注意将拨码开关拨向默认：仅“4”向下。

(4) 实验思考题：在不改变原代码功能的条件下将例 8-2 表达成用状态码直接输出型的状态机。

(5) 实验报告：根据以上的实验要求、实验内容和实验思考题写出实验报告。

## 实验二 运算器组成实验

### 1. 算术逻辑运算实验

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt2 / 实验 2-1.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt2 / DEMO\_2\_1\_alu

#### 一. 实验目的

1. 了解简单运算器的数据传输通路。
2. 验证运算功能发生器的组合功能。
3. 掌握算术逻辑运算加、减、与的工作原理。
4. 验证实验台运算的 8 位加、减、与、直通功能。
5. 按给定数据，完成几种指定的算术和逻辑运算。

#### 二. 实验内容

##### 1. 实验原理

算术逻辑单元 ALU 的数据通路如图 2-1 所示。其中运算器 ALU181 根据 74LS181 的功能用 VHDL 硬件描述语言编辑而成，构成 8 位字长的 ALU。参加运算的两个 8 位数据分别为 A[7..0]和 B[7..0]，运算模式由 S[3..0]的 16 种组合决定，而 S[3..0]的值由 4 位 2 进制计数器 LPM\_COUNTER 产生，计数时钟是 Sclk（图 2-1）；此外，设 M=0，选择算术运算，M=1 为逻辑运算，C<sub>N</sub> 为低位的进位位；F[7..0]为输出结果，C<sub>O</sub> 为运算后的输出进位位。两个 8 位数据由总线 IN[7..0]分别通过两个电平锁存器 74373 锁入，ALU 功能如表 2-1 所示。

表 2-1 ALU181 的运算功能

选择端 S3 S2 S1 S0	高电平作用数据		
	M=H	M=L 算术操作	
	逻辑功能	Cn=L（无进位）	Cn=H（有进位）
0 0 0 0	$F = A$	$F = A$	$F = A$ 加1
0 0 0 1	$F = A + B$	$F = A + B$	$F = (A + B)$ 加1
0 0 1 0	$F = AB$	$F = A + B$	$F = A + \bar{B} + 1$
0 0 1 1	$F = 0$	$F = \text{减}1$ （2 的补码）	$F = 0$
0 1 0 0	$F = \overline{AB}$	$F = A$ 加 $\bar{B}$	$F = A$ 加 $\bar{B}$ 加1
0 1 0 1	$F = \bar{B}$	$F = (A + B)$ 加 $\bar{B}$	$F = (A + B)$ 加 $\bar{B} + 1$
0 1 1 0	$F = A \oplus B$	$F = A$ 减 $B$	$F = A$ 减 $B$ 减1
0 1 1 1	$F = \bar{A}B$	$F = A + B$	$F = (A + \bar{B})$ 减1
1 0 0 0	$F = \bar{A} + B$	$F = A$ 加 $AB$	$F = A$ 加 $AB$ 加1
1 0 0 1	$F = \overline{A \oplus B}$	$F = A$ 加 $B$	$F = A$ 加 $B$ 加1
1 0 1 0	$F = B$	$F = (A + \bar{B})$ 加 $AB$	$F = (A + \bar{B})$ 加 $AB$ 加1
1 0 1 1	$F = AB$	$F = AB$	$F = AB$ 减1
1 1 0 0	$F = 1$	$F = A$ 加 $A^*$	$F = A$ 加 $A$ 加1
1 1 0 1	$F = A + B$	$F = (A + B)$ 加 $A$	$F = (A + B)$ 加 $A$ 加1
1 1 1 0	$F = A + B$	$F = (A + \bar{B})$ 加 $A$	$F = (A + \bar{B})$ 加 $A$ 加1
1 1 1 1	$F = A$	$F = A$	$F = A$ 减1

注 1、\* 表示每一位都移至下一更高有效位，“+”是逻辑或，“加”是算术加

注 2、在借位减法表达上，表 2-1 与标准的 74181 的真值表略有不同。

#### 三. 实验步骤

##### (1) 设计 ALU 元件

在 Quartus II 环境下，用文本输入编辑器 Text Editor 输入 ALU181.VHD 算术逻辑单元文件，编译 VHDL 文件，并将 ALU181.VHD 文件制作成一个可调用的原理图元件。

##### (2) 以原理图方式建立顶层文件工程

选择图形方式。根据图 2-1 输入实验电路图，从 Quartus II 的基本元件库中将各元件调入图形编辑窗口、连线，添加输入输出引脚。

将所设计的图形文件 ALU.bdf 保存到原先建立的文件夹中，将当前文件设置成工程文件，以后的操作就都是对当前工程文件进行的。

(3) 器件选择

选择 Cyclone 系列，在 Devices 中选择器件 EP1C6QC240C8。编译，引脚锁定，再编译。引脚锁定后需要再次进行编译，才能将锁定信息确定下来，同时生成芯片编程/配置所需要的各种文件。

(4) 芯片编程 Programming (可以直接选择光盘中的示例已完成的设计进行验证实验)

打开编程窗口。将配置文件 ALU.sof 下载进 GW48 系列现代计算机组成原理系统中的 FPGA 中。

(5) 选择实验系统的电路模式是 NO.0，验证 ALU 的运算器的算术运算和逻辑运算功能

根据表 2-1，从键盘输入数据 A[7..0]和 B[7..0]，并设置 S[3..0]、M、Cy，验证 ALU 运算器的算术运算和逻辑运算功能，记录实验数据。

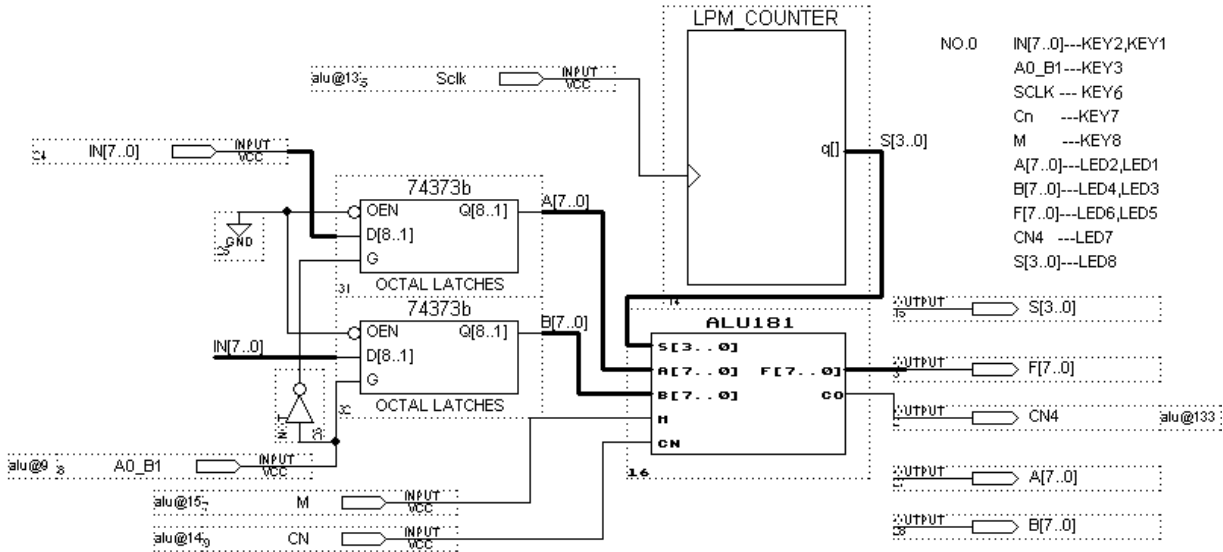


图 2-1 算术逻辑单元 ALU 实验原理图

四. 实验任务

(1) 按图 2-1 所示，在本验证性示例中用数据选择开关（键 3 控制）的高/低电平选择总线通道上的 8 位数据进入对应的 74373 中；即首先将键 3 输入高电平，用键 2、键 1 分别向 A[7..0] 置数 01010101 (55H)，这时在数码管 4/3 上显示输入的数据 (55H)；然后用键 3 输入低电平，再用键 2、键 1 分别向 B[7..0]置数 10101010 (AAH)，这时在数码管 2/1 上显示输入的数据 (AAH)；这时表示在图 2-1 中的两个 74373 锁存器中分别被锁入了加数 55H 和被加数 AAH。可双击图 2-1 的 ALU181 元件，了解其 VHDL 描述。

(2) 设定键 8 为低电平，即 M=0（允许算术操作），键 6 控制时钟 SCLK，可设置表 2-1 的 S[3..0]=0~F。现连续按动键 6，设置操作方式选择 S[3..0]=9（加法操作），使数码管 8 显示 9，以验证 ALU 的算术运算功能：当键 7 设置 cn=0（最低位无进位）时，数码管 7/6/5=0FF (55H+AAH=0FFH)；

当键 7 设置 cn=1（最低位有进位）时，数码管 7/6/5=100 (55H+AAH+1=100H)；

(3) 若设定键 8 为高电平，即 M=1，键 KEY6 控制时钟 SCLK，设置 S[3..0]=0~F，KEY7 设置 cn=0 或 cn=1，验证 ALU 的逻辑运算功能，并记录实验数据。

表 2-2 A[7..0]，B[7..0]设置值检查

F[7..0]	SW_B	寄存器内容		S3 S2 S1 S0	M	BUS
		A[7..0]	B[7..0]			
		01010101	10101010			
		01010101	10101010			

(4) 验证 ALU181 的算术运算和逻辑运算功能，ALU181 模块功能可参照表 2-1。

表 2-3 给定了寄存器 DR1=A[7..0]和 DR2=B[7..0]的数据(十六进制)，要求根据此数据对照逻辑功能表所

得的理论值(要求课前完成)与实验结果值进行比较(均采用正逻辑 0)。

(4)表 2-4 列出了 8 种常用的算术与逻辑运算要求指定的操作内容，正确选择运算器数据通路、控制参数 S3、S2、S1、S0、M，并将实验结果值填入括号内，表中给定原始数据 DR1=A[7..0]和 DR2=B[7..0]，以后的数据取自前面运算的结果。

表 2-3

S3 S2 S1 S0	A[7..0]	B[7..0]	算术运算 M=0		逻辑运算 (M=1)
			cn=0 (无进位)	cn=1 (有进位)	
0 0 0 0	A A	5 5	F= ( )	F= ( )	F= ( )
0 0 0 1	A A	5 5	F= ( )	F= ( )	F= ( )
0 0 1 0	A A	5 5	F= ( )	F= ( )	F= ( )
0 0 1 1	A A	5 5	F= ( )	F= ( )	F= ( )
0 1 0 0	F F	0 1	F= ( )	F= ( )	F= ( )
0 1 0 1	F F	0 1	F= ( )	F= ( )	F= ( )
0 1 1 0	F F	0 1	F= ( )	F= ( )	F= ( )
0 1 1 1	F F	0 1	F= ( )	F= ( )	F= ( )
1 0 0 0	F F	F F	F= ( )	F= ( )	F= ( )
1 0 0 1	F F	F F	F= ( )	F= ( )	F= ( )
1 0 1 0	F F	F F	F= ( )	F= ( )	F= ( )
1 0 1 1	F F	F F	F= ( )	F= ( )	F= ( )
1 1 0 0	5 5	0 1	F= ( )	F= ( )	F= ( )
1 1 0 1	5 5	0 1	F= ( )	F= ( )	F= ( )
1 1 1 0	5 5	0 1	F= ( )	F= ( )	F= ( )
1 1 1 1	5 5	0 1	F= ( )	F= ( )	F= ( )

表 2-4 8 种常用的算术与逻辑运算

操 作	S3 S2S1S0	M	Cn	DR1	DR2	运算关系及结果显示	Cn4
逻辑乘				66	FF	$DR_1 \cdot DR_2 \rightarrow DR_2 ( )$	
传送						$DR_1 \rightarrow DR_2 ( )$	
按位加						$DR_1 \oplus DR_2 \rightarrow DR_2 ( )$	
取反						$\overline{DR_1} \rightarrow DR_2 ( )$	
加 1						$DR_2 + 1 \rightarrow DR_2 ( )$	
求负						$\overline{DR_2} + 1 \rightarrow DR_2 ( )$	
加法						$DR_1 + DR_2 \rightarrow DR_2 ( )$	
减法						$DR_1 - DR_2 \rightarrow DR_2 ( )$	

五. 实验要求

1、做好实验预习，掌握运算器的数据传送通路和 ALU 的功能特性，并熟悉本实验中所用的控制台开关的作用和使用方法。

2、写出实验报告，内容是：

①实验目的； ②按理论分析值填写好表 2-2、表 2-3 和表 2-4，给出对应的仿真波形。

③列表比较实验数据（2）的理论分析值与实验结果值；并对结果进行分析。实验结果与理论分析值比较，有没有不同？为什么？ ④通过本实验，你对运算器 ALU 有何认识，有什么心得体会？

六. 实验题与思考题

1. 用 VHDL 实现输入暂存器 74373B 的功能，及模式选择计数器 LPM\_COUNTER 的功能。
3. 用 VHDL 表达整个 ALU 实验电路的功能，对电路进行仿真、引脚锁定、并在实验台上实现其功能。
4. 用 VHDL 设计一个简化的 8 位 alu，具有基本算术运算（加、减、带进位加、减）功能和逻辑运算（与 AND、或 OR、异或 XOR、非 NOT 等）功能，给出仿真波形，并在实验台上实现。
5. 用 VHDL 设计一个 16 位的 ALU，实现基本的算术逻辑运算，为了节省逻辑资源，建议使用两个 8 位 ALU 模块级联而成。
6. 对 ALU181 进行算术运算和逻辑运算的功能仿真，并记录仿真波形。



2. 带进位算术运算实验

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt2 / 实验 2-2.ppt  
实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt2 / DEMO\_2\_2\_ALUc

一. 实验目的

- 1、验证带进位控制的算术运算功能发生器的功能。      2、按指定数据完成几种指定的算术运算。

二. 实验原理

在实验（1）的基础上增加进位控制电路，将运算器 ALU181 的进位位送入 D 锁存器，由 T<sub>4</sub> 和 CN 控制其写入，在此，T<sub>4</sub>是由键 5 产生的脉冲信号，这时，CN 的功能是电平控制信号（高电平时，CN 有效），控制是否允许将进位信号 co 加入下一加法周期的最低进位位，从而可实现带进位控制运算。

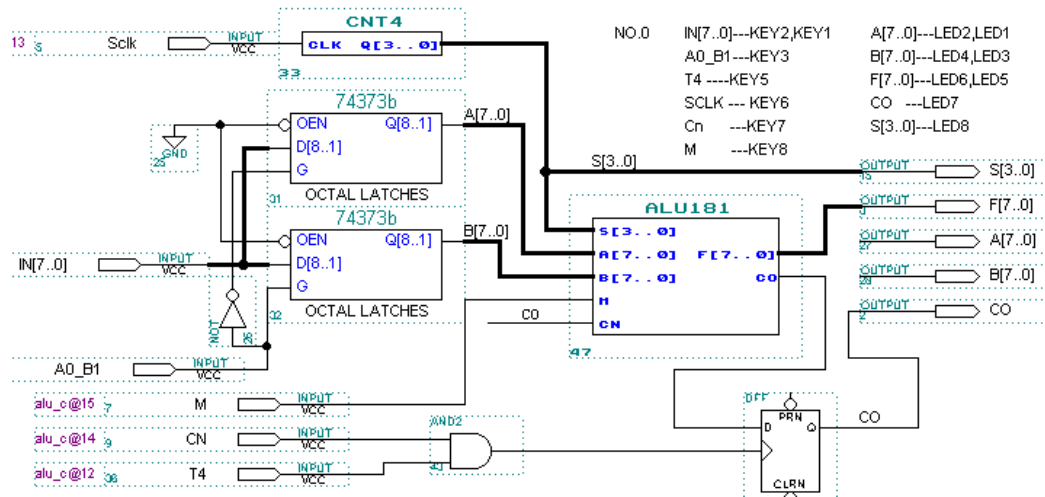


图 2-2A 带进位控制的 ALU

三. 实验步骤

- (1) 根据电路图 2-2A 和波形图 B，首先使键 5（T<sub>4</sub>）和键 7（CN）=0；键 8（M）和键 3（A<sub>0</sub>\_B<sub>1</sub>）=1；连续按键 6，使产生 9 个脉冲，这时数码管 8 显示 9(作加法运算)；再用键 2，键 1 输入加数 9DH（数码管 4/3 显示 9D）；
- (2) 按键 3=0，再用键 2，键 1 输入被加数 E5H（数码管 4、3、2、1 分别显示加数和被加数）；再将键 8（M）置 0，使 ALU 作算术运算，这时可以从数码管 6，5 上看到 9DH+E5H=82H（低 8 位和）；
- (3) 先将键 7（CN）置为 1(允许锁存 ALU 的进位)，再用键 5（T<sub>4</sub>）产生一个正脉冲，就能将进位锁入 D 触发器中：数码管 7 将显示 1，表示加法有进位，并被锁；同时可以看到此进位被累加，使数码管 6，5=83H。
- (4) 置键 8=1，在实验箱上作逻辑运算方面的实验，给出相应的仿真波形图；
- (5) 利用带进位控制，控制 T<sub>4</sub>，分别由低到高输入 3 个 8 位加数和被加数，计算 24 位加法：7AC5E9 H+ BD5AF8H = ? 最后按照下表完成实验，记录实验数据，给出对应仿真波形图。

表 2-5

S3 S2 S1 S0	A[7..0]	B[7..0]	算术运算 M=0		逻辑运算 (M=1)
			cn=0 (无进位)	cn=1 (有进位)	
0 1 0 1	F F	0 1	F= ( )	F= ( )	F= ( )
0 1 1 0	F F	0 1	F= ( )	F= ( )	F= ( )
0 1 1 1	F F	0 1	F= ( )	F= ( )	F= ( )
1 0 0 0	F F	F F	F= ( )	F= ( )	F= ( )
1 0 0 1	F F	F F	F= ( )	F= ( )	F= ( )
1 0 1 0	F F	F F	F= ( )	F= ( )	F= ( )

四. 实验要求

- 1、做好实验预习，掌握带进位控制的算术运算功能发生器的功能特性。
- 2、写出实验报告，内容是：①实验目的；②按理论分析值填写表 2-5。③列表比较实验数据的理论分

析值与实验结果值；并对结果进行分析。④实验结果与理论分析值比较，有没有不同？为什么？

五. 附加实验题和思考题

- 1. 带进位运算与不带进位运算有何区别？
- 2. 如何实现带进位运算，将上一次运算的进位位用于下一次的运算当中，并实现多个 8 位数据的（如两个 24 位数据的加法）运算？在控制电路上应作怎样的改动？给出 24 位加法详细的仿真波形图。

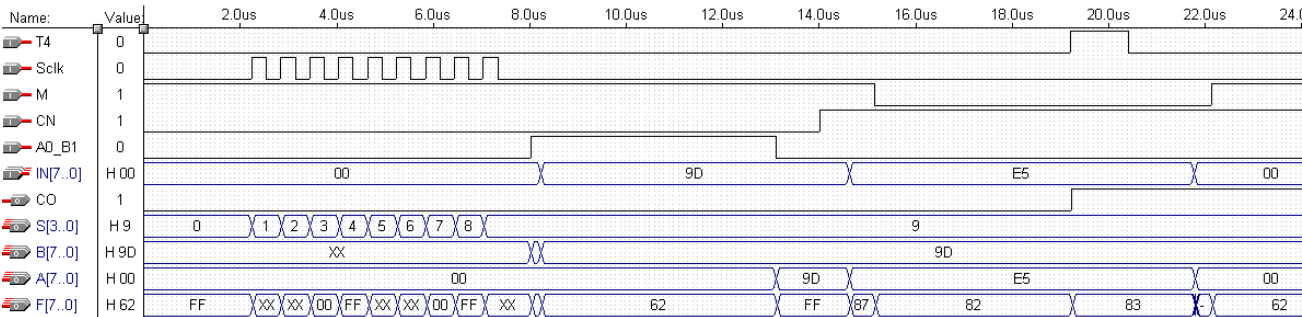


图 2-2B 带进位控制的 ALU 的仿真波形

3. 移位运算器实验

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt2 / 实验 2-3.ppt  
实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt2 / DEMO\_2\_3\_shifter

一. 实验目的

- 1. 验证移位控制的组合功能。

二. 实验原理

1、移位运算实验原理图如图 2-3 所示。移位运算器 SHEFT 使用 VHDL 语言编写，其输入/输出端分别与键盘/显示器 LED 连接。移位运算器是时序电路，在时钟信号到来时状态产生变化，CLK 为其时钟脉冲。由 S<sub>0</sub>、S<sub>1</sub>、M 控制移位运算的功能状态，具有数据装入、数据保持、循环右移、带进位循环右移，循环左移、带进位循环左移等功能。移位运算器的具体功能见表 2-7 所示：

- 2. 电路连接、输入数据的按键、输出显示数码管的定义如图 2-3 右上角所示。

CLK——时钟脉冲，通过键 5 产生 0—1；  
M ——工作模式，M=1 时带进位循环移位，由键 8 控制；  
C0 ——允许带进位移位输入，由 键 7 控制；  
S ——移位模式 0~3，由 键 6 控制，显示在数码管 LED8 上；  
D[7..0] ——移位数据输入，由 键（2 和 1）控制，显示在数码管（2 和 1）上；  
QB[7..0]——移位数据输出，显示在数码管（6 和 5）上；  
CN——移位数据输出进位，显示在数码管（7）上；

三. 实验步骤

- （1）实验台选择模式 0、下载（Configure）到实验台；示例工程文件是 1SHEFT.bdf，
- （2）键入待移位数据。通过键盘键 1、键 2 向 D[7..0]置数 01101011（6BH，显示在数码管 2 和 1）。
- （3）将 D[7..0]装入移位运算器 QB[7..0]。键 6 设置(S1,S0)=3，键 8 设置 M=0，（S&M=6，允许加载待移位数据，显示于数码 8）；此时用键 5 产生 CLK（0-1-0），将数据装入（加载进移位寄存器，显示在数码管 6 和 5）。
- （4）对输入数据进行移位运算。再用键 6 设置为(S1,S0)=2（S&M=4，显示于数码 8，允许循环右移）；连续按键 5，产生 CLK，输出结果 QB[7..0]（显示在数码管 6 和 5）将发生变化：6BH→B5H→DAH...，
- （5）键 8 设置 M=1（允许带进位循环右移），观察带进位移位允许控制 C0 的置位与清零对移位的影响；
- （5）根据表 2-7，通过设置（M、S<sub>1</sub>、S<sub>0</sub>）验证移位运算的带进位和不带进位移位功能。

四. 实验要求

- 1、做好实验预习，掌握带进位控制的算术运算功能发生器的功能特性。
- 2、写出实验报告，内容是：①实验目的，实验原理；②按理论分析值准备并填写好实验数据表；③列表比较实验数据的理论分析值与实验结果值，并对结果进行分析；④实验结果与理论分析值比较，有没有不同？为什么？

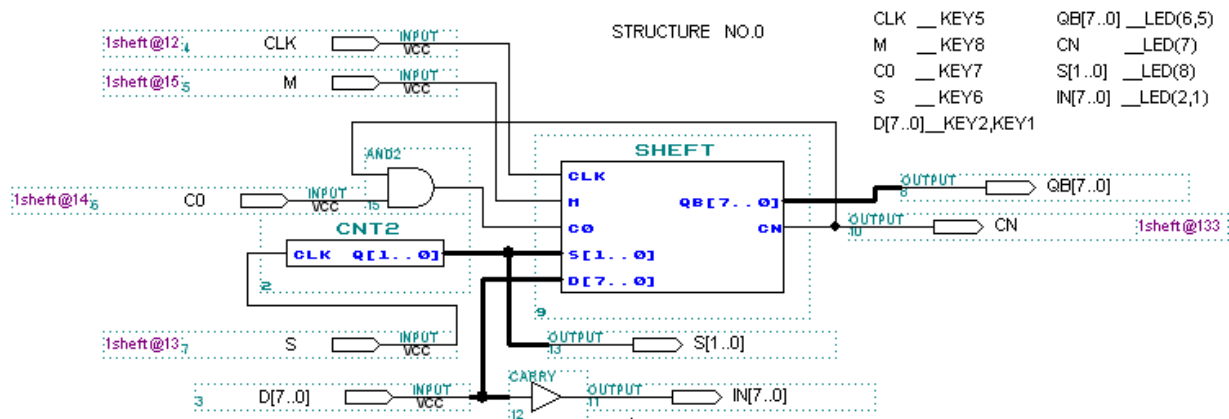


图 2-3 移位运算实验原理图

表 2-7 移位发生器的功能

G	S1	S0	M	功 能
0	0	0	任意	保持
0	1	0	0	循环右移
0	1	0	1	带进位循环右移
0	0	1	0	循环左移
0	0	1	1	带进位循环左移
任意	1	1	任意	加载待移位数据

五. 思考题

如何实现有符号数的算术右移和算术左移？修改用 VHDL 编写的实验参考程序，进行功能仿真，并在实验台上调试实现。 附：实验参考程序请参考文件夹中的 1SHEFT.VHD 和 CNT2.VHD

实验三 存储器实验

1、FPGA 中 LPM\_ROM 定制与读出实验

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt3 / 实验 3-1.ppt  
实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt3 / DEMO\_3\_1\_rom

一. 实验目的

- 1、掌握 FPGA 中 lpm\_rom 的设置，作为只读存储器 ROM 的工作特性和配置方法。
- 2、用文本编辑器编辑 mif 文件配置 ROM，学习将程序代码以 mif 格式文件加载于 lpm\_ROM 中；
- 3、在初始化存储器编辑窗口编辑 mif 文件配置 ROM； 4、验证 FPGA 中 mega\_lpm\_ROM 的功能。

二. 实验原理

ALTERA 的 FPGA 中有许多可调用的 LPM (Library Parameterized Modules)参数化的模块库，可构成如 lpm\_rom、lpm\_ram\_io、lpm\_fifo、lpm\_ram\_dq 的存储器结构。CPU 中的重要部件，如 RAM、ROM 可直接调用他们构成，因此在 FPGA 中利用嵌入式阵列块 EAB 可以构成各种结构的存储器，lpm\_ROM 是其中的一种。lpm\_ROM 有 5 组信号：地址信号 address[ ]、数据信号 q[ ]、时钟信号 inclock、outclock、允许信号 memenable，其参数都是可以设定的。由于 ROM 是只读存储器，所以它的数据口是单向的输出端口，ROM 中的数据是在对 FPGA 现场配置时，通过配置文件一起写入存储单元的。图 3-1-1 中的 lpm\_ROM 有 3 组信号：inclk——输入时钟脉冲；q[23..0]——lpm\_ROM 的 24 位数据输出端；a[5..0]——lpm\_ROM 的 6

位读出地址。

实验中主要应掌握以下三方面的内容：

- (1) lpm\_ROM 的参数设置；
- (2) lpm\_ROM 中数据的写入，即 LPM\_FILE 初始化文件的编写；
- (3) lpm\_ROM 的实际应用，在 GW48\_CP+实验台上的调试方法。

三. 实验步骤

(1) 用图形编辑，进入 mega\_lpm 元件库，调用 lpm\_rom 元件，设置地址总线宽度 address[]和数据总线宽度 q[], 分别为 6 位和 24 位,并添加输入输出引脚，如图 3-1-1 设置和连接。

(2) 设置图 3-1-1 为工程。

(3) 在设置 lpm\_rom 数据参数选择项 lpm\_file 的对应窗口中（图 3-1-2），用键盘输入 lpm\_ROM 配置文件的路径（rom\_a.mif），然后设置在系统 ROM/RAM 读写允许，以便能对 FPGA 中的 ROM 在系统读写。

(4)用初始化存储器编辑窗口编辑 lpm\_ROM 配置文件（文件名.mif）。这里预先给出后面将要用到的微程序文件：rom\_a.mif 。rom\_a.mif 中的数据是微指令码（图 3-1-3）。

(5) 全程编译。

(6) 下载 SOF 文件至 FPGA，改变 lpm\_ROM 的地址 a[5..0]，外加读脉冲，通过实验台上的数码管比较读出的数据是否与初始化数据(rom\_a.mif 中的数据)一致。

注，下载 sof 示例文件至实验台上的 FPGA，选择实验电路模式仍为 NO.0，24 位数据输出由数码 8 至数码 3 显示，6 位地址由键 2、键 1 输入，键 1 负责低 4 位，地址锁存时钟 CLK 由键 8 控制，每一次上升沿，将地址锁入，数码管 8/7/6/5/4/3 将显示 ROM 中输出的数据。发光管 8 至 1 显示输入的 6 位地址值。

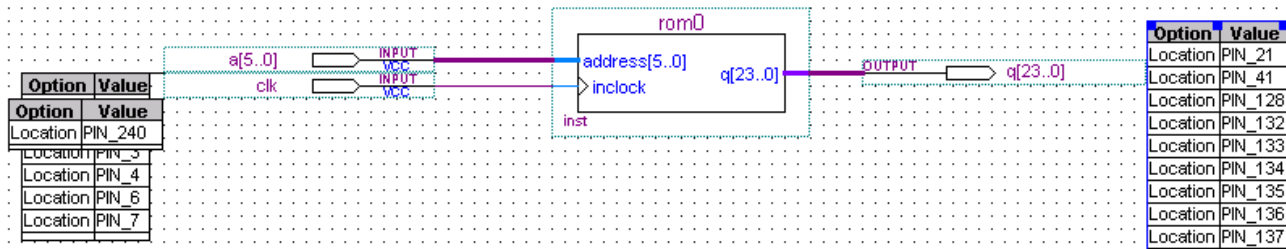


图 3-1-1 lpm\_ROM 的结构图

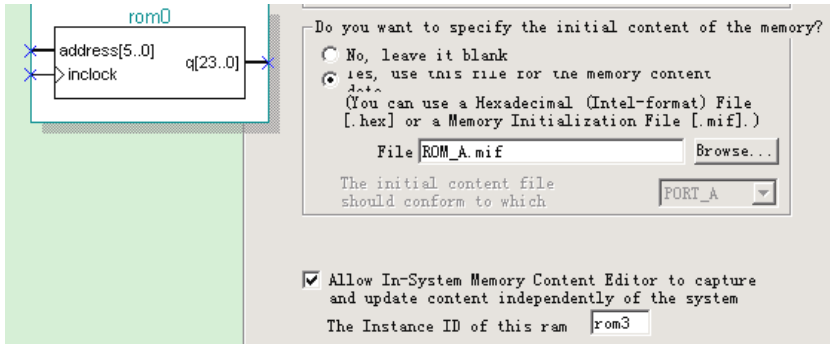


图 3-1-2 设置在系统 ROM/RAM 读写允许

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	018108	00ED82	00C050	00E004	00B005	01A206	959A01	00E00F
08	00ED8A	00ED8C	00A008	008001	062009	062009	070A08	038201
10	001001	00ED83	00ED87	00ED99	00ED9C	31821D	31821F	318221
18	318223	00E01A	00A01B	070A01	00D181	21881E	019801	298820
20	019801	118822	019801	198824	019801	018110	000002	000003
28	000004	000005	000006	000007	000008	000009	00000A	00000B
30	00000C	00000D	00000E	00000F	000010	000011	000012	000013
38	000014	000015	000016	000017	000018	000019	00001A	00001C

图 3-1-3 rom\_a.mif 中的数据

(7) 打开 QuartusII 的在系统存储模块读写工具，了解 FPGA 中 ROM 中的数据，并对其进行在系统读写操作（图 3-1-4）。

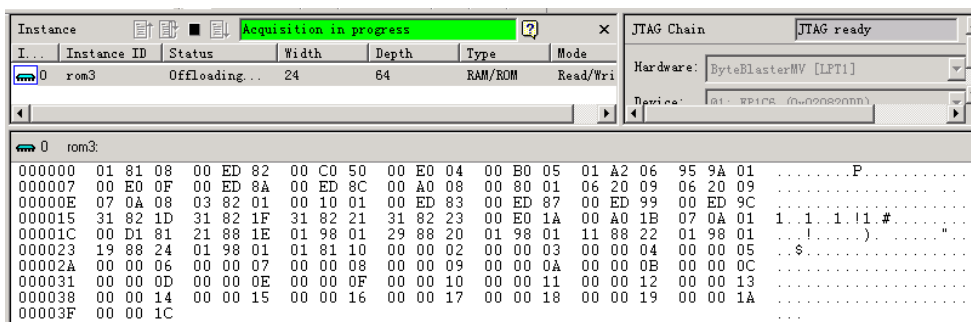


图 3-1-4 在系统存储模块读写

## 四. 实验要求

(1)实验前认真复习 LPM-ROM 存储器部分的有关内容。 (2)记录实验数据, 写出实验报告, 给出仿真波形图。 (3)通过本实验, 对 FPGA 中 EAB 构成的 LPM-ROM 存储器有何认识, 有什么收获?

## 五. 思考题

(1) 如何在图形编辑窗口中设计 LPM-ROM 存储器? 怎样设计地址宽度和数据线的宽度? 怎样导入 LPM-ROM 的设计参数文件和存储 LPM-ROM 的设计参数文件?

(2)怎样对 LPM-ROM 的设计参数文件进行软件仿真测试? (3)怎样在 GW48 实验台上对 LPM-ROM 进行测试? (4)学习 LPM-ROM 用 VHDL 语言的文本设计方法(顶层文件用 VHDL 表达)。

(5)了解 LPM-ROM 存储器占用 FPGA 中 EAB 资源的情况。

## 2. FPGA 中 LPM\_RAM 读写实验

实验课件参考: /CMPUT\_EXPMT/Experiments/Expmt3 / 实验 3-2.ppt

实验示例参考: /CMPUT\_EXPMT/Experiments/Expmt3 / DEMO\_3\_2\_RAM

### 一. 实验目的

- 1、了解 FPGA 中 RAMlpm\_ram\_dq 的功能,
- 2、掌握 lpm\_ram\_dq 的参数设置和使用方法,
- 3、掌握 lpm\_ram\_dq 作为随机存储器 RAM 的工作特性和读写方法。

### 二. 实验原理

在 FPGA 中利用嵌入式阵列块 EAB 可以构成存储器, lpm\_ram\_dq 的结构如图 3-2-1。数据从 ram\_dp0 的左边 D[7..0]输入, 从右边 Q[7..0]输出, R/W——为读/写控制信号端。数据的写入: 当输入数据和地址准备好以后, 在 inclock 是地址锁存时钟, 当信号上升沿到来时, 地址被锁存, 数据写入存储单元。

数据的读出: 从 A[7..0]输入存储单元地址, 在 CLK 信号上升沿到来时, 该单元数据从 Q[7..0]输出。

R/W——读/写控制端, 低电平时进行读操作, 高电平时进行写操作;

CLK——读/写时钟脉冲;

DATA[7..0]——RAM\_dq0 的 8 位数据输入端;

A[7..0]——RAM 的读出和写入地址;

Q[7..0]——RAM\_dq0 的 8 位数据输出端。

### 三. 实验步骤

(1)按图 3-2-1 输入电路图。并进行编译、引脚锁定、FPGA 配置。

(2)通过键 1、键 2 输入 RAM 的 8 位数据(选择实验电路模式 1), 键 3、键 4 输入存储器的 8 位地址。键 8 控制读/写允许, 低电平时读允许, 高电平时写允许; 键 7 (CLK0)产生读/写时钟脉冲, 即生成写地址锁存脉冲, 对 lpm\_ram\_dq 进行写/读操作。

(3)注意, lpm\_ram\_dq 也能加入初始化文件(这里是 5\_ram.mif,是后面将要用到的模型 CPU 执行程序文件), 注意此文件加入的路径表达和文件表达(3-2-2): ./5\_ram.mif , (后缀 mif 要小写);

同时择在系统读写 RAM 功能, RAM 的 ID 名取为: ram1。

注, 验证程序文件在 DEMO5\_lpm\_ram 目录, 工程名是 ram\_dp1.bdf, 下载 ram\_dp1.sof 至实验台上的 FPGA, 选择实验电路模式为 NO.1, 按以上方式首先进行验证实验。首先控制读出初始化数据, 与载入的初始化文件 ram\_dp1.mif 中的数据进行比较, 然后控制写入一些数据, 再读出比较。使用在系统读写 RAM 的工具对其中的数据进行读写操作(图 3-2-3), 设置成连续读模式, 将在系统读写工具窗口的数据与实验



箱上数码管上显示的数据对照起来看。

四. 实验要求

(1)实验前认真复习运算器和存储器部分的有关内容；(2)写出实验报告。

五. 思考题与实验题

(1) 如何在图形编辑窗口中设计 lpm\_ram\_dq 存储器？怎样设定地址宽度和数据线的宽度？设计一数据宽度为 6，地址线宽度为 7 的 RAM，仿真检验其功能，并在 FPGA 上进行硬件测试。

(2) 如何建立 lpm\_ram\_dq 的数据初始化，如何导入和存储 lpm\_ram\_dq 参数文件？生成一个 mif 文件，并导入以上的 RAM 中。(3) 怎样对 lpm\_ram\_dq 设计参数文件进行软件仿真测试？(4) 使用 VHDL 文件作为顶层文件，学习 lpm\_ram\_dq 的 VHDL 语言的文本设计方法。(5) 了解 lpm\_ram\_dq 存储器占用 FPGA 中 EAB 资源的情况。(6) 使用系统读写 RAM 的工具对其中的数据进行读写操作。

(6) lpm\_ram\_dq 存储器在 CPU 中有何作用？

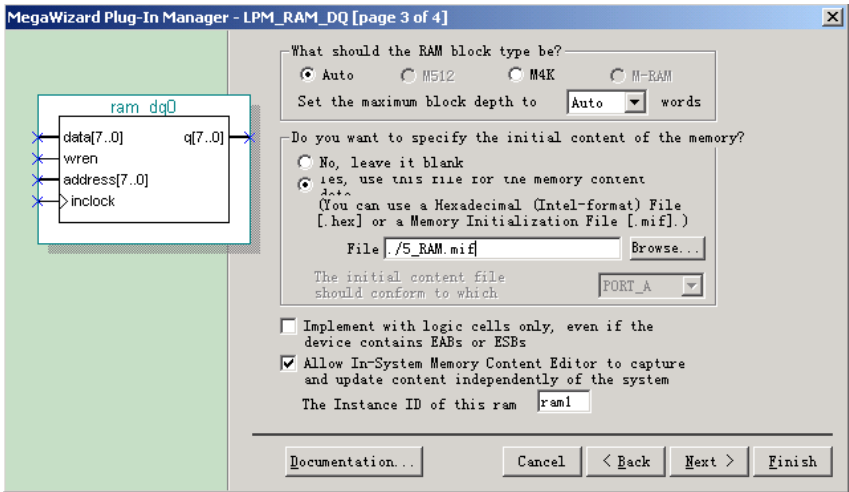


图 3-2-2 lpm\_ram\_dq 加入初始化文件和选择在系统读写 RAM 功能

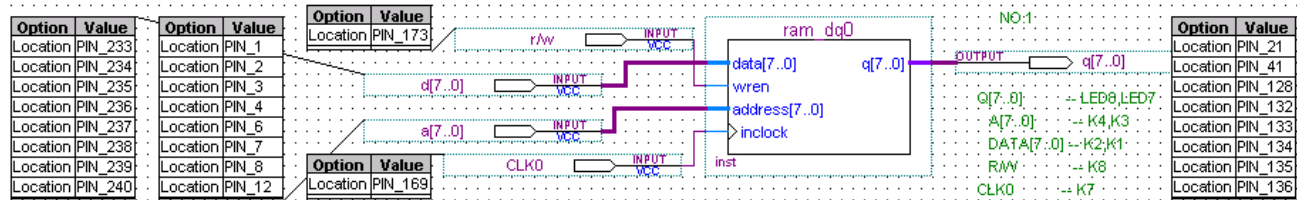


图 3-2-1 lpm\_ram\_dp 实验电路图

3. FIFO 定制与读/写实验

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt3 / 实验 3-3.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt3 / DEMO\_3\_3\_FIFO

一. 实验目的

- 1、掌握 FPGA 中先进先出存储器 lpm\_fifo 的功能，工作特性和读写方法。
- 2、了解 FPGA 中 lpm\_fifo 的功能，掌握 lpm\_fifo 的参数设置和使用方法，
- 3、掌握 lpm\_fifo 作为先进先出存储器 FIFO 的工作特性和读写方法。

二. 实验原理

FIFO (First In First Out) 是一种存储电路，用来存储、缓冲在两个异步时钟之间的数据传输。使用异步 FIFO 可以在两个不同时钟系统之间快速而方便地实时传输数据。在网络接口、图像处理、CPU 设计等方面，FIFO 具有广泛的应用。在 FPGA 中利用嵌入式阵列块 EAB 可以构成存储器，lpm\_fifo 的结构如图 3-3-1 所示。

- WR — 写控制端，高电平时进行写操作；
- RD — 读控制端，高电平时进行读操作；
- CLK — 读/写时钟脉冲；
- CLR — FIFO 中数据异步清零信号；
- D[7..0] — lpm\_fifo 的 8 位数据输入端；
- Q[7..0] — lpm\_fifo 的 8 位数据输出端；

U[7..0] — 表示 lpm\_fifo 已经使用的地址空间。

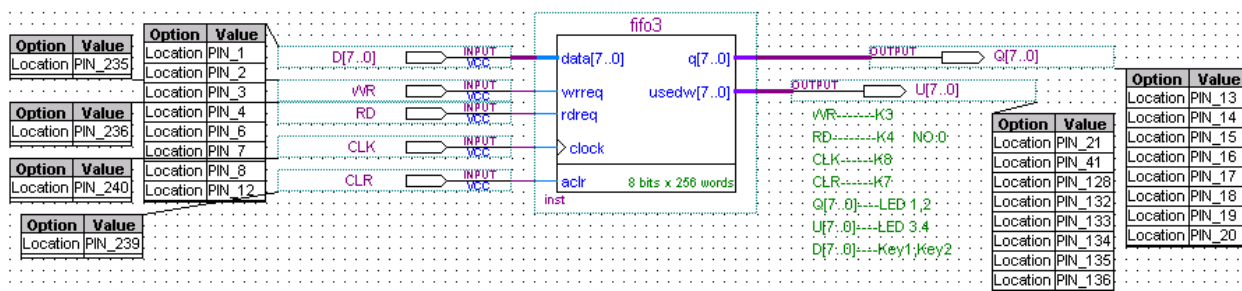


图 3-3-1 lpm\_fifo 的实验结构图

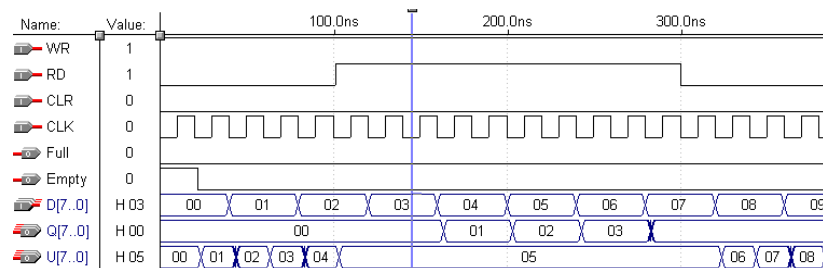


图 3-3-2 lpm\_fifo 的仿真波形图

### 三. 实验步骤

1. 编辑输入 lpm\_fifo 实验电路（双击原理图 3-3-1 的 FIFO 元件，可进入该元件的编辑窗）。
2. 将编译通过的文件下载到 GW-48 实验台，实验台选择工作模式 NO.0；
3. 通过实验台上的键 1、键 2 输入数据，键 3 控制读/写允许 WR（高电平写有效，低电平读有效），键 7 控制数据清 0（高电平清 0 有效）、键 8 输入 CLK 信号，数码管 4/3 显示已占用地址，数码管 2/1 显示 FIFO 输出的数据：

（1）将数据写入 LPM-FIFO：键 3 置高电平（写允许）；键 7 清 0 一次；键 1、键 2 每输入一个新数据（数据显示于发光管 D8-D1），键 8 就给出一个脉冲（按键 0-1-0），将数据压入 FIFO 中；

（2）将数据读出 LPM-FIFO：键 3 置低电平（读允许）；随着键 8 给出脉冲，观察数码管 2/1 显示的 FIFO 中输出的数据，与刚才写入的数据进行比较，同时注意数码 4/3 显示的地址数变化的顺序。

注，验证程序文件工程名是 fifo2.bdf，下载 fifo2.sof 至实验台上的 FPGA，选择实验电路模式为 NO.0，按以上方式首先进行验证实验。

### 四. 实验要求

1. 实验前认真复习 LPM-FIFO 存储器部分的有关内容。
2. 完成 FIFO 设计和验证，给出仿真波形图，增加“空”、“未满”、“满”的标志信号，写出实验报告。

### 五. 思考题与实验题

1. 通过本实验，对 FPGA 中 EAB 构成的 LPM-FIFO 存储器有何认识，有什么收获？
2. 如何了解 lpm\_fifo 存储器占用 FPGA 中 EAB 资源的情况？
3. lpm\_fifo 存储器在 CPU 设计中有何作用？当 lpm\_fifo“空”、“未满”、“满”时，full、empty 和 usedw[7..0]d 的输出信号如何变化？
4. 怎样通过波形仿真了解 LPM-FIFO 存储器的功能？
5. 如何设置 LPM-FIFO 存储器各项参数？

## 4. FPGA 与外部 16 位 RAM 接口实验

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt3 / 实验 3-4.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt3 / DEMO\_3\_4\_SRAM16B

## 一. 实验目的

1. 掌握 FPGA 与外部 RAM 的硬件接口技术；
2. 通过 FPGA 控制，向外部 RAM 写入数据；
3. 通过 FPGA 控制，从外部 RAM 读出数据，并且用数码管显示读出的数据。

## 二. 实验原理

用 FPGA 与外部 RAM 接口,实现对外部 RAM 的读写控制.FPGA 需要产生地址信号和读写控制信号,并且需要采用具有双向 I/O 功能的电路结构,实现对 SRAM 数据端口输入/输出操作。接口电路主要由可增减地址计数器 LPM\_COUNTER、三态总线控制器 LPM\_BUSTRI、读写控制电路组成。实验电路结构如图 3-4-1 所示,

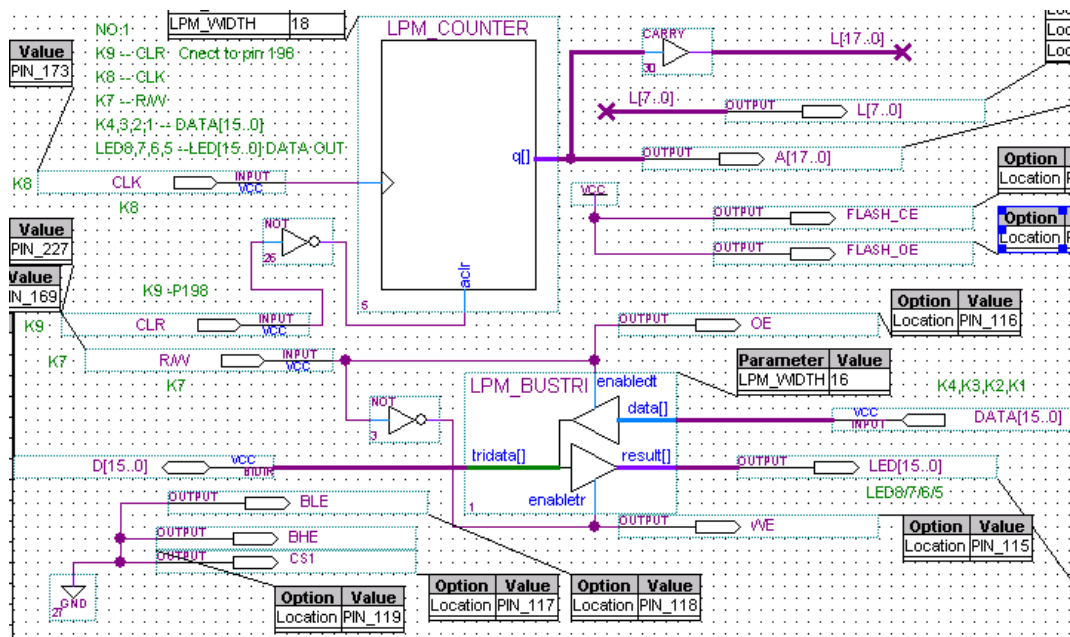


图 3-4-1 FPGA 与外部 16 位数据, 18 位地址线宽 SRAM 接口电路结构

## 六. 16 位 SRAM 读写逻辑设计

如果要设计 16 位数据总线的 CPU, 又要用到容量的 RAM, 就必须外接具有 16 位数据口的 RAM。在实验板上与 FPGA 相接有 2 片 256K 字节/每片的 16 位 RAM: IDT71V416, 电路连接如图 3-4-2 所示, FPGA 读写控制电路原理图如图 3-4-1 所示。实验验证步骤如下 (对其中 1 片 RAM 读写):

- 1、验证程序 sram.bdf, 下载 sram.sof 至实验台上的 FPGA, 选择实验电路模式为 NO. 1; 用一接插线将适配板上方的 P196 针与实验板主系统上的键 9 的插针相接, 键 9 作为地址计数器清 0 控制端。
- 2、利用键 4、键 3、键 2、键 1 输入数据, 放在 RAM 数据口, 如 ABCDH (显示于数码管 4/3/2/1);
- 3、按动键 9, 对地址信号发生计数器清 0, 键 7 置 1 (写 RAM 允许, 高电平为读 RAM 写允许);
- 4、写 RAM。用键 4/3/2/1 每更新一次 16 位输入数据, 就按动 1 次键 8 (0→1→0), 即使地址值自动加 1 (地址值显示于发光管 D8—D1, 左为高位, 了解图 3-4-1 的地址计数器功能)。
- 5、读出已被写入的数据。按动键 9, 对地址信号发生计数器清 0, 键 7 置 0 (读 RAM 允许), 之后连续按键 8, 递增地址值 (地址值显示于发光管 D8—D1), 将能依次顺序 (显示于数码 8/7/6/5 上) 读出外部 16 位 RAM 中已写入的数据, 与输入数据进行比较。

## 五. 思考题

1. FPGA 如何与外部存储器双向数据总线接口? FPGA 采用怎样的电路结构、如何控制双向数据口的数据输入/输出?
2. 若要对任意指定存储单元进行读写, 图 3-4-1 电路应如何修改? 请在实验台上验证所设计的功能。
3. 根据图 3-4-1 和 3-4-2, 从新锁定引脚, 对另一块 RAM 进行读写。
4. 根据图 3-4-1 和 3-4-2, 对 Flash SST39VF080/160 进行读写。
5. 通过本实验, 对 CPU 与外部存储器接口电路设计有何认识, 有什么收获?
6. 在计算机外部存储器的读写时序是怎样的? 怎样使 FPGA 满足对外部 RAM 的读/写时序要求?



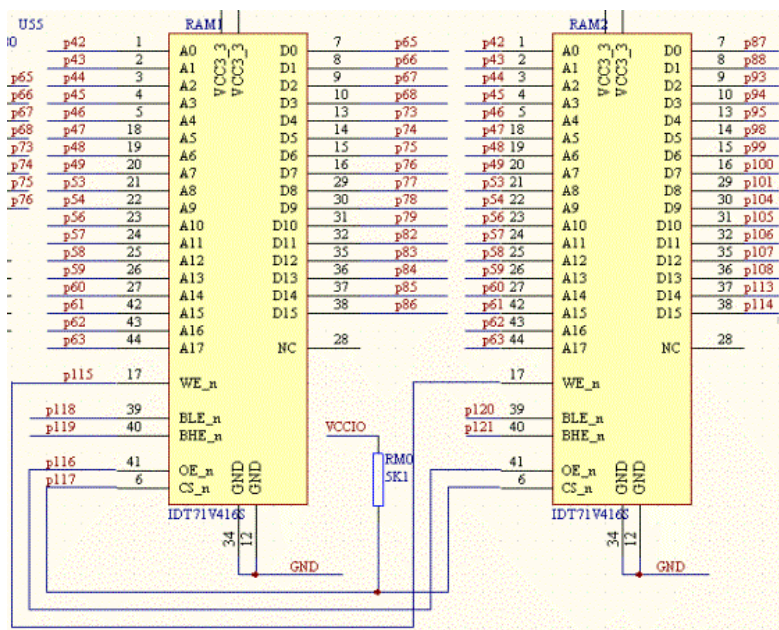


图 3-4-2 16 位 SRAM 和 6264 电路原理图

## 实验四 微控制器实验

### 1 节拍脉冲发生器时序电路实验

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt4 / 实验 4-1.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt4 / DEMO\_4\_1\_S\_T

#### 一. 实验目的

- (1) 掌握节拍脉冲发生器的设计方法和工作原理。
- (2) 理解节拍脉冲发生器的工作原理。

#### 二. 实验原理

计算机之所以能够按照人们事先规定的顺序进行一系列的操作或运算，就是因为它的控制部分能够按一定的先后顺序正确地发出一系列相应的控制信号。这就要求计算机必须有时序电路。控制信号就是根据时序信号产生的。本实验说明时序电路中节拍脉冲发生器的工作原理。

##### 1、连续节拍发生电路设计（图 4-1-1）：

可由 4 个 D 触发器组成，可产生 4 个等间隔的时序信号  $T_1 \sim T_4$ ，其中 CLK1 为时钟信号，由实验台右边的方波信号源 clock0 提供，可产生 1Hz~12MHz 的方波信号频率。实验者可根据实验自行选择信号频率。当 RST1 为低电平时，T1 输出为“1”，而 T2、T3、T4 输出为“0”；当 RST1 由低电平变为高电平后，T1~T4 将在 CLK1 的输入脉冲作用下，周期性地轮流输出正脉冲，机器进入连续运行状态（EXEC）。

T1~T4 以及 CLK1、RST1 的工作波形如图 4-1-2 所示。示例工程文件是 T4.bdf。硬件实验验证方法如图 4-1-1 所示，下载 T4.SOF 文件，选择实验模式 1，Clock0 接 4Hz，键 8 控制 RST1，高电平时可以看到，发光管 1、2、3、4 分别显示 T1、T2、T3、T4 的输出电平（实验结果与仿真波形图 4-1-2 比较！）。

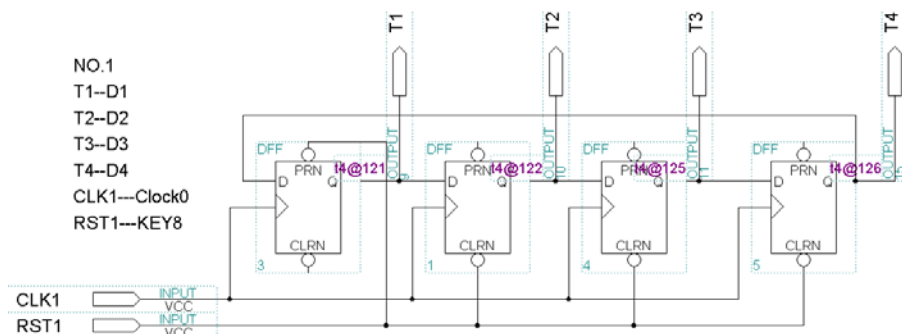


图 4-1-1 节拍脉冲发生器的工作原理

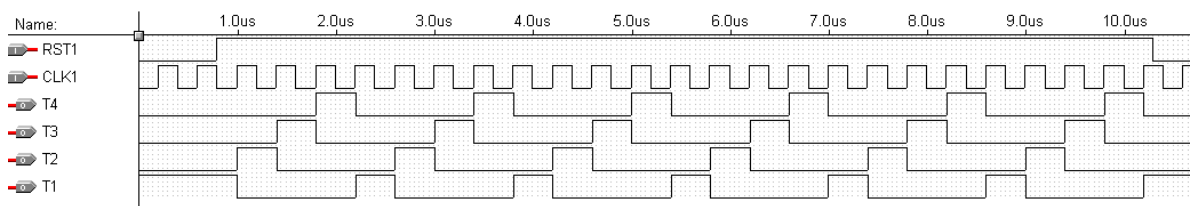


图 4-1-2 节拍脉冲发生器工作波形

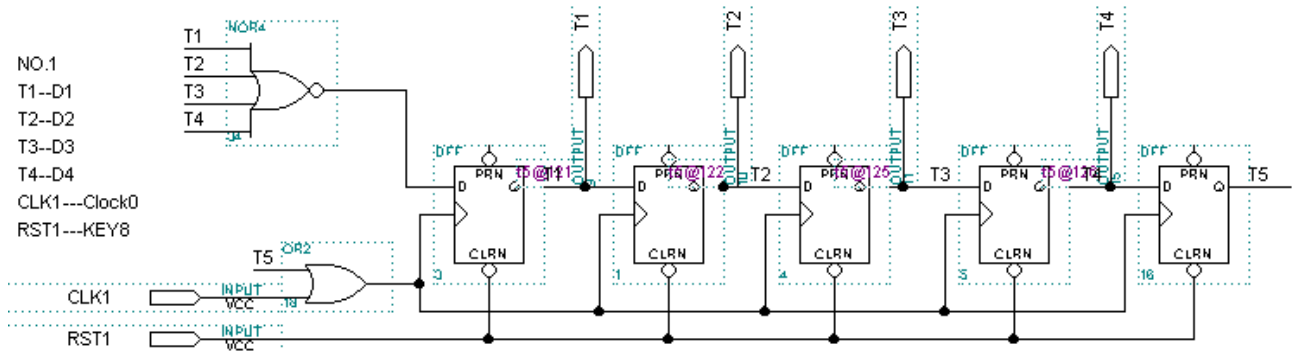


图 4-1-3 单步运行电路工作原理

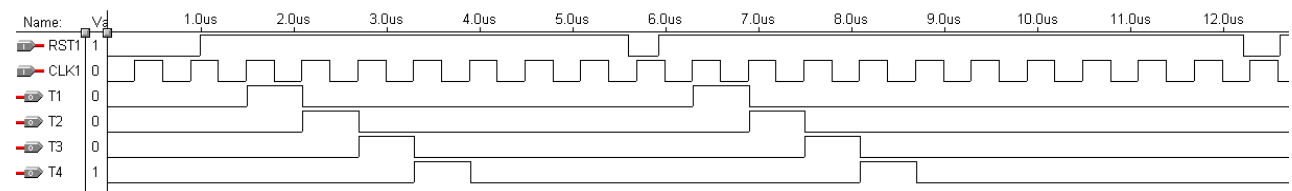


图 4-1-4 单步运行电路工作波形

## 2、单步节拍发生电路（图 4-1-3）：

将图 4-1-1 电路稍加改变即可得到图 4-1-3 所示的单步运行电路。该电路每当 RST1 出现一个负脉冲后，仅输出一组 T1、T2、T3、T4 节拍信号，直到 RST1 出现下一个负脉冲，波形如图 4-1-4 所示。

示例工程文件是 T5.bdf。硬件实验验证方法如图 4-1-3 所示，下载 T5.SOF 文件，选择实验模式 1，Clock0 接 4Hz（选择范围是 1Hz-50MH），键 8 控制 RST1。每出现一个负脉冲，发光管 1、2、3、4 分别显示 T1、T2、T3、T4 的输出电平一次（实验结果与仿真波形图 4-1-4 比较！）。

## 3、单步/连续节拍发生电路（图 4-1-5）：

增加两个 2-1 多路选择器，可将图 4-1-3 电路改变为图 4-1-5 所示电路。S0 是单步或连续节拍发生控制信号，当 S0=0，选择单步运行方式；当 S0=1，选择连续运行方式。图 4-1-6 为此电路的仿真波形。

示例工程文件是 TS5.bdf。硬件实验验证：下载 TS5.SOF 文件，选择实验模式 1，Clock0 接 4Hz，键 8 控制 RST1，键 7 控制 S0，发光管 1、2、3、4 分别显示 T1、T2、T3、T4 的输出电平（实验结果与仿真波形图 4-1-6 比较！）。

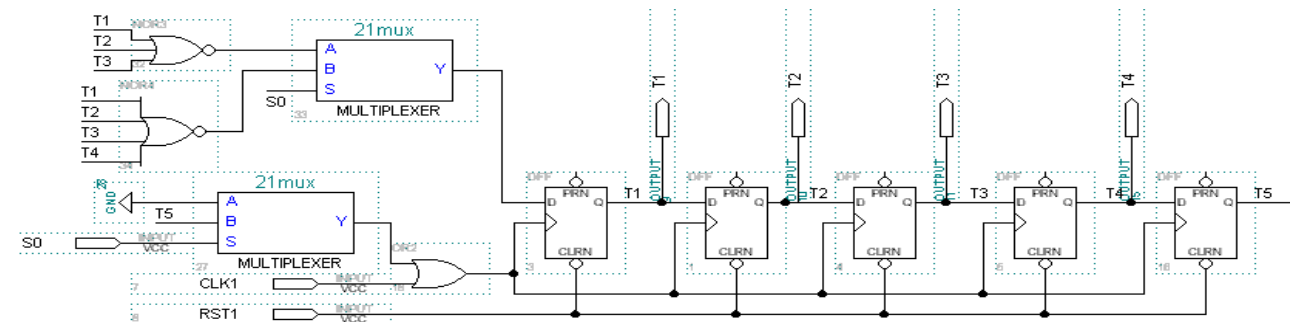


图 4-1-5 单步/连续运行电路工作原理

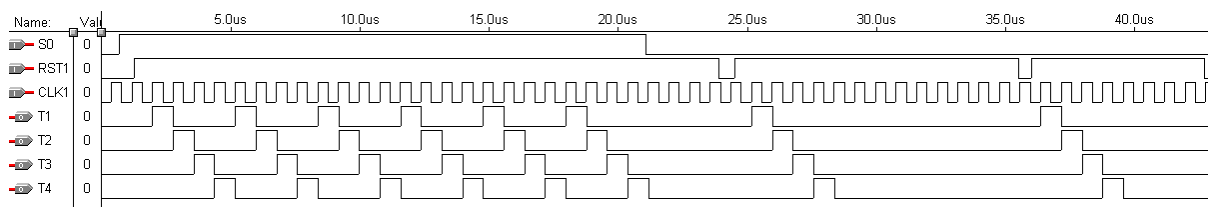


图 4-1-6 单步运行电路工作波形

### 三. 实验步骤

- (1) 硬件验证测试连续节拍发生电路（图 4-1-1），实验结果与仿真波形图 4-1-2 比较！
- (2) 硬件验证测试单步节拍发生电路（图 4-1-3），实验结果与仿真波形图 4-1-4 比较！
- (3) 硬件验证测试单步/连续节拍发生电路（图 4-1-5），实验结果与仿真波形图 4-1-6 比较！
- (4) 绘出相应的时序波形图。

### 四. 实验报告

- (1)实验原理。
- (2)绘制相应的时序波形图。
- (3)实验结果分析、讨论。

### 五. 思考题

1. 单步运行与连续运行有何区别，它们各自的使用环境怎样？
2. 如何实现单步/连续运行工作方式的切换？
3. 用 VHDL 设计实现节拍控制电路，并通过实验台验证其（单步/连续）功能。

## 2. 程序计数器 PC 与地址寄存器 AR 实验

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt4 / 实验 4-2.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt4 / DEMO\_4\_2\_PC\_AR

### 一. 实验目的

1. 掌握地址单元的工作原理。
2. 掌握的两种工作方式，加 1 计数和重装计数器初值的实现方法；
3. 掌握地址寄存其从程序计数器获得数据和从内部总线获得数据的实现方法。

### 二. 实验原理

地址单元主要由三部分组成：地址寄存器和多路开关。

程序计数器 PC 用以指出下一条指令在主存中的存放地址，CPU 正是根据 PC 的内容去存取指令的。因程序中指令是顺序执行的，所以 PC 有自增功能。程序计数器提供下一条程序指令的地址，如电路图 4-2-1 所示，在 T4 时钟脉冲的作用下具有自动加 1 的功能；在 LDPC 信号的作用下可以预置计数器的初值（如子程序调用或中断相应等）。当 LDPC 为高电平时，计数器装入 data[ ]端输入的数据。aclr 是计数器的清 0 端，高电平有效（高电平清零）；aclr 为低电平时，允许计数器正常计数。

地址寄存器 AR（74273）锁存访问内存 SRAM 的地址。273 中的地址来自两个渠道。一是程序计数器 PC 的输出，通常是下一条指令的地址；二是来自于内部数据总线的的数据，通常是被访问操作数的地址。

为了实现对两路输入数据的切换，在 FPGA 的内部通过总线多路开关 BUSMUX 进行选择。LDAR 与多路选择器的 sel 相连，当 LDAR 为低电平，选择程序计数器的输出；当 LDAR 为高电平时，选择内部数据总线的的数据。

### 三. 实验步骤

1. 按照 图 4-2-1 程序计数器原理图编辑、输入电路，实验台选择 NO.0 工作模式。对输入原理图进行编译、引脚锁定、并下载到实验台。示例工程文件是 PC\_unit.bdf。硬件实验验证（与仿真波形图 4-2-2 比较!）。实验说明：

(1) 下载 pc\_unit.sof ； (2) 用模式键选模式“0”，再按一次右侧的复位键；

(3) 键 2 和键 1 可输入 8 位总线数据 B[7..0]（此值显示于发光管 D1~D8 和数码管 2/1）；CLR（键 5）按 2 次(0→1→0)，产生一正脉冲，高电平清零；LDAR（键 6）=0 时，BUSMUX 输出程序计数器 PC 的值；LDAR=1 时，BUSMUX 输出 B[7..0]总线数据。LDPC（键 7）：程序计数器 PC 预置控制端，当 LDPC=1 时，将 B[7..0]总线数据装入程序计数器 PC；当 LDPC=0 时，程序计数器 PC 处于计数自动工作状态，对 T4 进行计数；T4（键 8）：程序计数器 PC 的计数时钟 CLK，键 8 按动两次产生一个计数脉冲。

2. 通过 B[7..0]设置程序计数器的预加载数据。当 LDPC=0 时，观察程序计数器自动加 1 的功能；当 LDPC=1 时，观察程序计数器加载输出情况，示例操作：

- 1、所有键置 0，键 2/1 输入 A5；按键 5→PC 计数器清 0(0→1→0)；
- 2、连续按动键 8，可以从数码 8/7 上看到 AR 的输出，即 PC 值；
- 3、按键 6→‘1’，选通直接输出总线上的数据 A5 作为 PC 值，按键 8，产生一个脉冲上升沿，即可看到 AR（显示在数码 8/7）的输出为 A5；
- 4、使键 6=0，仍选通 PC 计数器输出，这时键 2/1 输入 86，按键 7 产生一个上升脉冲(0→1→0)，即用 LDPC 将 86 加载进 PC 计数器；
- 5、连续按动键 8，可以发现 AR 的输出在 86 上累加输出：86、87、88 等。

#### 四. 实验报告

(1)实验原理。 (2)绘制相应的时序波形图。 (3)实验结果分析、讨论。

#### 五. 思考题

分支和转移程序与顺序程序有何区别？要实现程序的分支和转移，需要对程序计数器 PC 和地址寄存器 AR 作怎样的操作？应改变哪些控制信号，请在实验台上实现程序转移的功能。

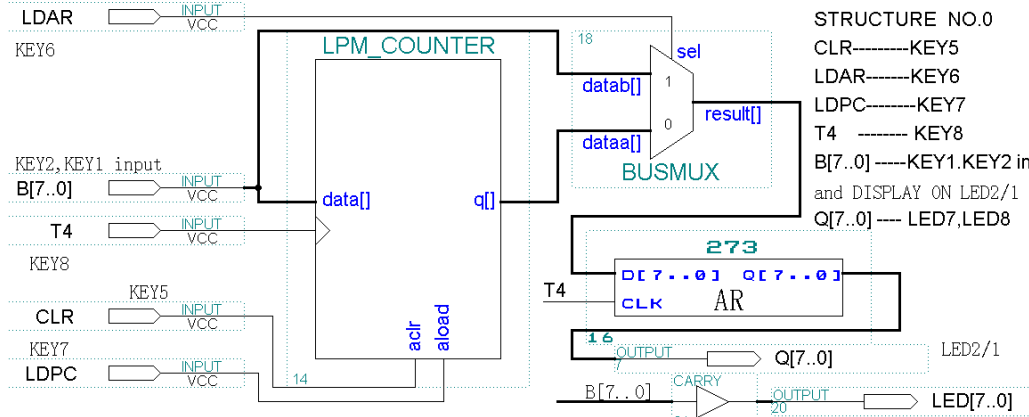


图 4-2-1 程序计数器原理图

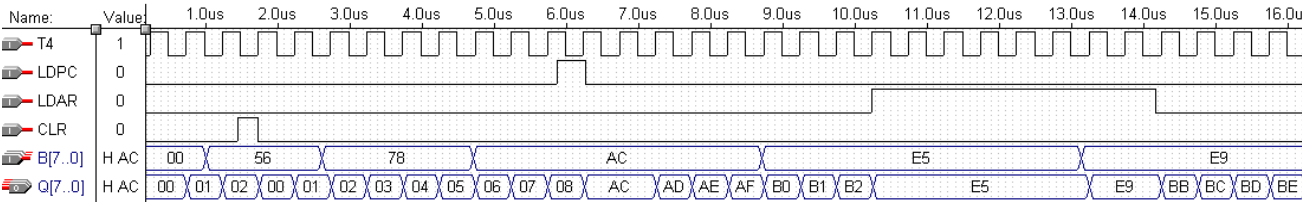


图 4-2-2 程序计数器工作波形

### 3. 微控制器组成实验

#### 一. 实验目的

1. 掌握微程序控制器的工作原理和构成原理
2. 掌握微程序的编写、输入，观察微程序的运行。

#### 二. 实验原理

##### 1. 微程序控制电路

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt4 / 实验 4-3.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt4 / DEMO\_4\_3\_uC

微程序控制器的组成如图 4-3-1。其中控制存储器由 FPGA 中的 LPM\_ROM 构成，输出 24 位控制信号。在 24 位控制信号中，微命令信号 18 位，微地址信号 6 位。在不判别测试的情况下，在 T2 时刻将打入微地址寄存器 uA 的内容，即为下一条微指令地址。当 T4 时刻进行测试判别时，转移逻辑满足条件后输出的负脉冲通，过强制端将某一触发器置为“1”状态，完成地址修改。

微程序控制器中的微控制代码可以通过对 FPGA 中 LPM\_ROM 的配置进行输入，通过编辑 LPM\_ROM.mif

文件修改微控制代码。详细情况可参考实验三中 FPGA 中 LPM\_ROM 的配置方法。微指令控制电路内部结构如图 4-3-2 所示。

三. 实验步骤

1. 微指令控制电路实验。

下载 se5\_1.sof 到实验台，或输入图 4-3-2 微指令控制电路，并按照图中说明锁定引脚。编译、下载到实验系统中，选择实验台工作模式 No.1。键盘/显示定义如下：

- 1) 键 1、键 2 输入 6 位微指令数据 I[7..2]，键 2 中的高两位还作为标志位 FC、FZ；
- 2) 键 3 输入分支控制信号 P[4..1]；
- 3) 键 4 输入控制台的控制信号 SWA、SWB；
- 4) 键 8 输入节拍信号 T4；
- 4) 数码 5、数码 6 显示微地址控制信号 SE[6..1]。

根据微程序控制器的内部结构，记录当 FC、FZ 变化时，微指令 I[7..2]的变化，对输出微地址控制信号 SE[6..1]的影响；

- 观察、记录当微指令 I[7..2]的值变化时，SE[6..1]的变化情况；
- 观察、记录分支信号 P[4..1]有效时，微指令 I[7..2]的变化对输出微地址控制信号 SE[6..1]的影响；
- 观察、记录 SWA、SWB 对输出微地址控制信号 SE[6..1]的影响。

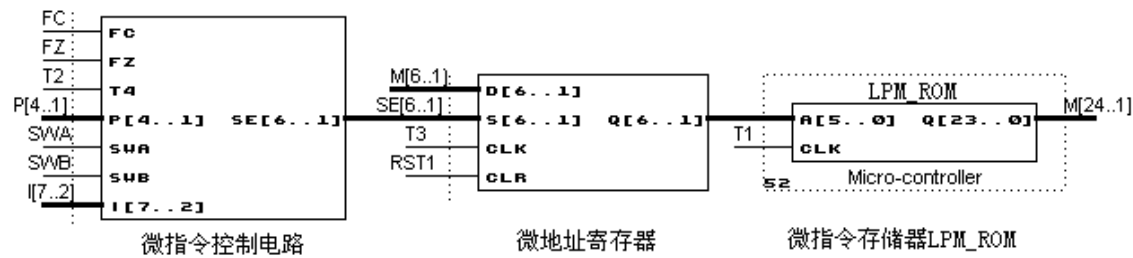


图 4-3-1 微程序控制电路

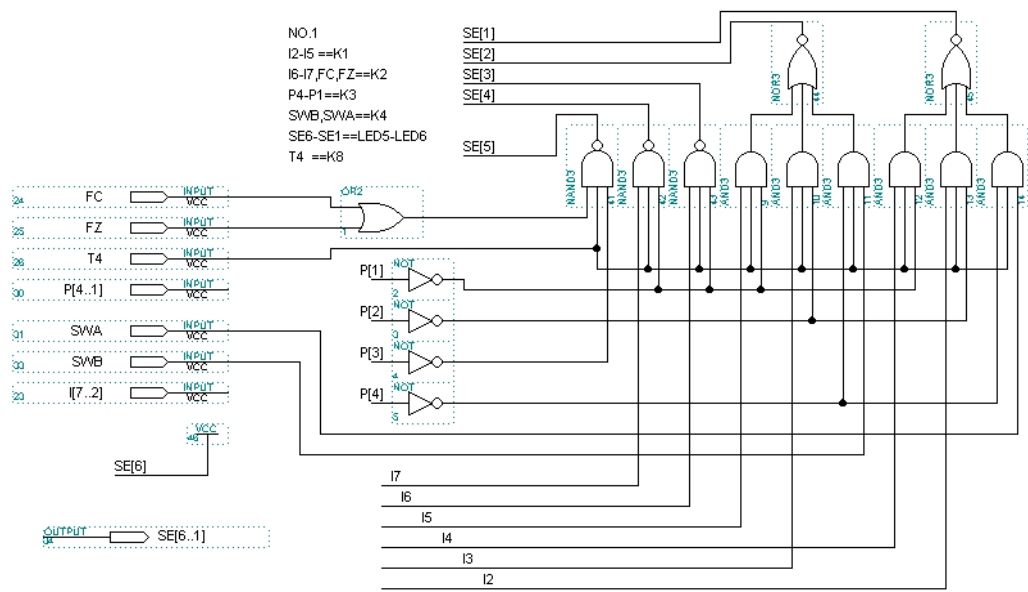


图 4-3-2 微指令控制电路内部结构

2. 微地址寄存器电路实验。

下载 se6\_1.sof 到实验台。或按照图 3-3-3 中说明锁定引脚，编译、下载到实验系统中，选择实验台工作模式 No.1。键盘/显示定义如下：

- 1) 键 1、键 2 输入 D 触发器数据 d[6..1]；
- 2) 键 4、键 3 输入 D 触发器置“1”控制信号 S[6..1]，低电平有效；
- 3) 键 7 输入 D 触发器复位（清零）控制信号，低电平有效；
- 4) 键 8 输入时钟信号 CLK；
- 5) 数码 7、8 显示 D 触发器输出信号 q[6..1]。



观察记录微地址寄存器在正常工作情况下，由 d[6..1]输入、q[6..1]输出的微地址实验数据，以及在发生控制/转移情况下，当 s[6..1]信号有效时，q[6..1] 输出的微地址发生变化的情况。

2. 微地址寄存器电路

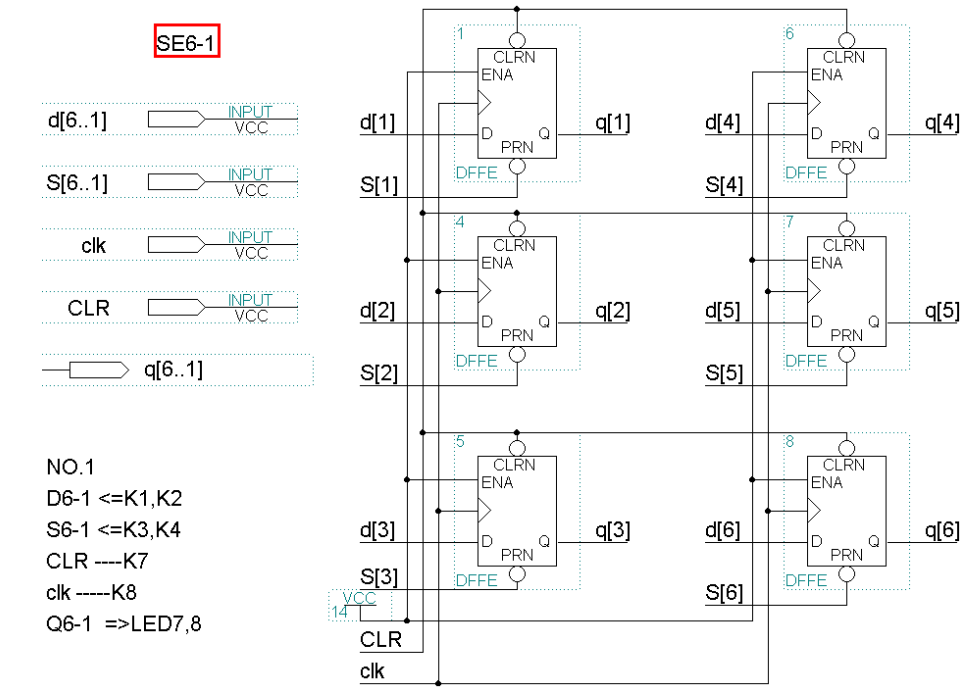


图 4-3-3 微地址寄存器电路

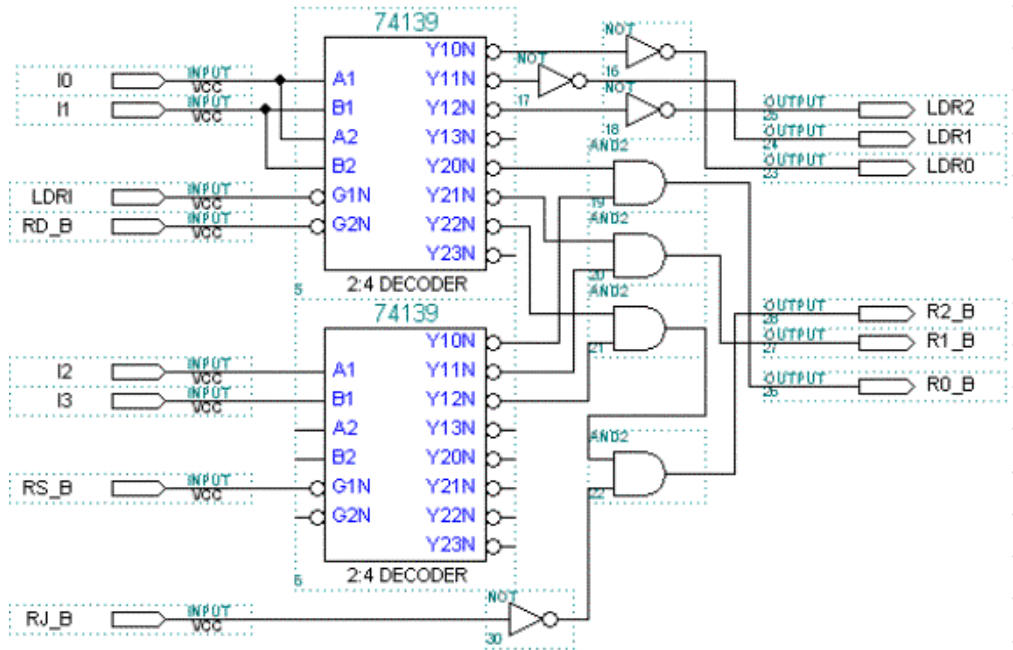


图 4-3-4 微地址译码器

3. 微地址译码器实验

下载 ldr0\_2.sof 到实验台（打开相应的工程文件 ldr0\_2.gdf，观察其原理图和引脚锁定情况）。或按照图 4-3-4 中说明锁定引脚，编译、下载到实验系统中，选择实验台工作模式 No.5。

观察、记录微指令信号中 I[3..0] 的变化、控制信号 LDRI、RD\_B、RS\_B、RJ\_B 的变化，对输出选通信号 LDR0~LDR2、R0\_B~R2\_B 的影响。

四. 实验报告

- (1)实验原理。
- (2)绘制相应的时序波形图。
- (3)实验结果分析、讨论。

## 五. 思考题

1. 在微指令控制电路中, 当 FC 或 FZ 有效时, 对其输出 S[6..1]有何影响? 对微地址寄存器的输出会有何影响? 如何实现对程序的控制/转移功能?
2. 说明 P[4..1]信号分别有效时, 对微指令控制电路中输出 S[6..1]有何影响?
3. 当控制信号 SWA、SWB 取不同的值时, 对微指令控制电路中输出 S[6..1]有何影响?

## 实验五 总线控制实验

实验课件参考: /CMPUT\_EXPMT/Experiments/Expmt5 / 实验 5-1.ppt

实验示例参考: /CMPUT\_EXPMT/Experiments/Expmt5 / DEMO\_5\_1\_BUS

### 一. 实验目的

1. 理解总线的概念及特性。
2. 掌握总线传输控制特性。

### 二. 实验原理

#### 1. 总线的基本概念

总线是多个系统部件之间进行数据传输的公共通路, 是构成计算机系统的骨架。借助总线连接, 计算机在系统各部件之间实现传送地址、数据和控制信息的操作。所谓总线就是指能为多个功能部件服务的一组公用信息线。

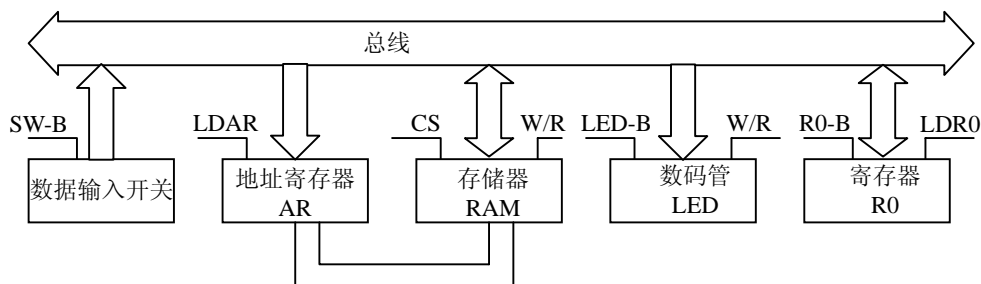


图 5-1 总线实验传输框图

#### 2. 实验原理

实验所用总线实验传输框图如图 4-1 所示。它将几种不同的设备挂在总线上, 有存储器、输入设备、输出设备、寄存器。这些设备在传统的系统中需要有三态输出控制, 然而在 FPGA 的内部没有三态输出控制结构, 因此必须采用总线输出多路开关结构加以控制。按照传输要求恰当有序地控制它们, 就可以实现总线信息传输。

### 三. 实验内容

#### 1. 实验要求

根据挂在总线上的几个基本部件, 设计一个简单的流程。

- (1) 输入设备将数据打入寄存器 R0。
- (2) 输入设备将另一个数据打入地址寄存器 AR。
- (3) 将寄存器 R0 中的数据写到当前地址的存储器中。
- (4) 将当前地址的存储器中的数用数码管显示。

图 5-4 是总线控制的时序仿真波形图。

#### 2. 实验步骤

(1) 实验电路如图 5-2 所示。写使能 WE=1 允许写, =0 禁止写, 允许读; inclock 为数据 DATA 锁存时钟。具体操作可参考图 5-3。

(4)、工程文件是 BUS-4.bdf, 下载 BUS-4.sof 到实验台的 FPGA 中;

(5)、实验内容 1, 根据图 5-2 完成实验操作: 选择实验模式 “0”; 再按一次右侧的复位键(用一接线将实验板上键 9 的输入端插针与适配板上 FPGA 的第 P196 针相连, 以便能用键 9 控制 OUT 锁存器的时钟;): 初始状态: 1、键 4、键 3 控制设备选择端: sel[1..0]=00 (键 4/键 3=00,); 2、此时由键 2/键 1 输入的数据 (26H, 显示于数码管 2/1) 直接进入 BUS (数码管 8/7 显示), 键 5、6、7 为低电平; 3、键 8=1 (允许 RAM 写入) 完成图 5-2 所示的操作: 4、键 5 发正脉冲 (0-1-0), 将数据打入寄存器 R0; 5、键 2/键 1 再输入数据 (如 37H); 6、键 6 发正脉冲 (0-1-0), 将数据打入地址寄存器 AR; 7、键 2/键 1 再输入数据 (如 48H); 8、

键 7 发正脉冲 (0-1-0)，将数据写入 RAM (此时必须键 8 输出 ‘1’，注意此时进入 RAM 的数据 48H 是放在地址 37H 单元的)；9、键 2/键 1 再输入数据(如 59H)；10、键 9 发正脉冲 (0-1-0)，将数据写入寄存器 OUT (数码管 6/5 将显示此数)；11、键 4、键 3 分别选择 sel[1..0]=00、01、10、11，从数码管 8/7 上观察被写入的各寄存器中的数据。

(6)、实验内容 2：先将数据 28H 写入 RAM 的地址 (4AH)，再将数据 1BH 送进 R0，最后将刚才写入 RAM 中地址 (4AH) 的数据读出送到 OUT 口。依据总线电路图 5-3，操作如下：

1、用一接线将实验板上键 9 的输入端插针与适配板上 FPGA 的第 P196 针相连，以便能用键 9 控制 OUT 锁存器的时钟；键 3、4、5、6、7、8 都为低电平，使键 4/键 3=00，即总线多路选择器 sel[1..0]=00，选择由键 2/键 1 输入的数据 4AH (地址)，直接进入 BUS；

2、按键 6 两次 (0-1-0)，产生一个正脉冲，将地址数据 4AH (地址) 锁入地址寄存器 AR，如图 5-3 所示，此数据直接进入 RAM 的 address 端；

3、按键 2/键 1，输入数据 28H (数据)，此时直接进入总线 BUS，并进入 RAM 的 data 数据端；按键 8=1 (RAM 写允许)；按键 7 两次，将数 28H 写入 RAM (地址为 4AH)，最后按键 8=0，写禁止，读允许。

4、由键 2/键 1 输入的数据 1BH，按键 5 两次 (0-1-0)，产生一个正脉冲，即此数写入 R0 寄存器。

5、读 RAM 送到 OUT：由键 2/键 1 输入的数据 4AH，按键 5 两次，使 4AH 进入 AR；

6、按键 7 两次，RAM 中 4AH 单元中的数据 28H 输出，再使键 4/键 3=10，即总线多路选择器 sel[1..0]=10，此时 RAM 数据口的 28H 进入总线 BUS (可从数码管 8/7 上看到)；

7、按键 9 一次 (此键是单脉冲)，RAM 口的 28H 即被锁如输出口 OUT 寄存器，由数码管 6/5 显示。

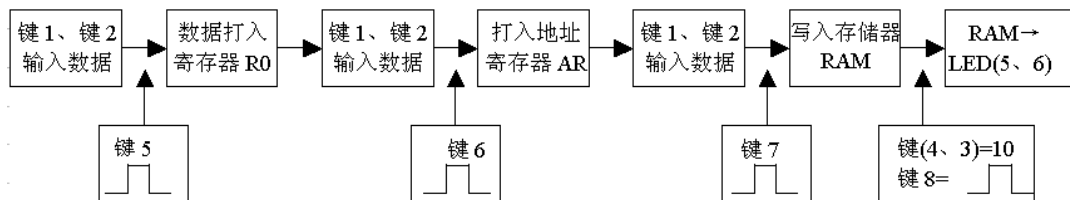


图 5-2 总线数据传输练习操作步骤

3、键盘/显示定义详细说明：

1) 键 2、键 1 输入 D[7..0]，输入的数据同时显示在数码 2和数码 3上。

2) 键 4、键 3 输入控制设备选择端 sel[1..0]，如图 5-2 所示，键 4/键 3 控制总线多路选择器，选择不同设备的数据进入总线：sel[1..0]= 00：

3) 输入设备 INPUT 数据进入总线 BUS；= 01：寄存器 R0 中的数据进入总线 BUS；= 11：地址寄存器 AR 的数据进入总线 BUS；= 10：存储器 RAM 的数据进入总线 BUS；

4) 总线 BUS 上的输出数据显示在数码 8和数码 7上；

5) 键 5 控制寄存器 R0 的输入选通锁存端；

6) 键 6 控制地址寄存器 AR 输入选通锁存端；

7) 键 7 控制 LPM\_RAM 数据 DATA 输入锁存端；

8) 键 8 控制 LPM\_RAM 写入允许 WE 端，=1 有效；

9) 键 9 控制输出设备 OUTPUT 的输入选通端，输出数据显示在数码 6和数码 5上，要求首先用一接线将实验板上键 9 的输入端插针与适配板上 FPGA 的第 P196 针相连。

对于 GW48-CP++型实验系统，附加键的可以用实验系统右下角的键 9 至 14，不用插连线，将其上的 6 个短路帽全部靠上插“FPGA”即可。每一键对应的 PIO 口已标在键上方。

#### 四. 实验报告

(1) 实验原理。(2) 在思考题中选作 2~3 题，给出实现方案和具体的操作步骤。

(3) 绘制相应的时序波形图。

(4) 实验结果分析、讨论。

#### 五. 思考题实验题

1. 如何向 RAM 中输入多个数据，并在输出设备 OUTPUT 上显示这些数据？(将 3 个数据写入 RAM 的不同地址中，再将它们分别读出，在 OUT 上显示)



2. 传输过程中是否会在总线上发生数据冲突？若发生冲突应怎样避免？

3. 用 VHDL 设计数据寄存器。

1. 要进行以下数据传输，应如何控制输入/输出选通信号，通过数据总线实现多个部件之间的数据传输？请给出实现方案和具体的操作步骤。完成以下 7 道总线传输实验练习题，写出实验的详细过程，并配以仿真波形图加以说明：

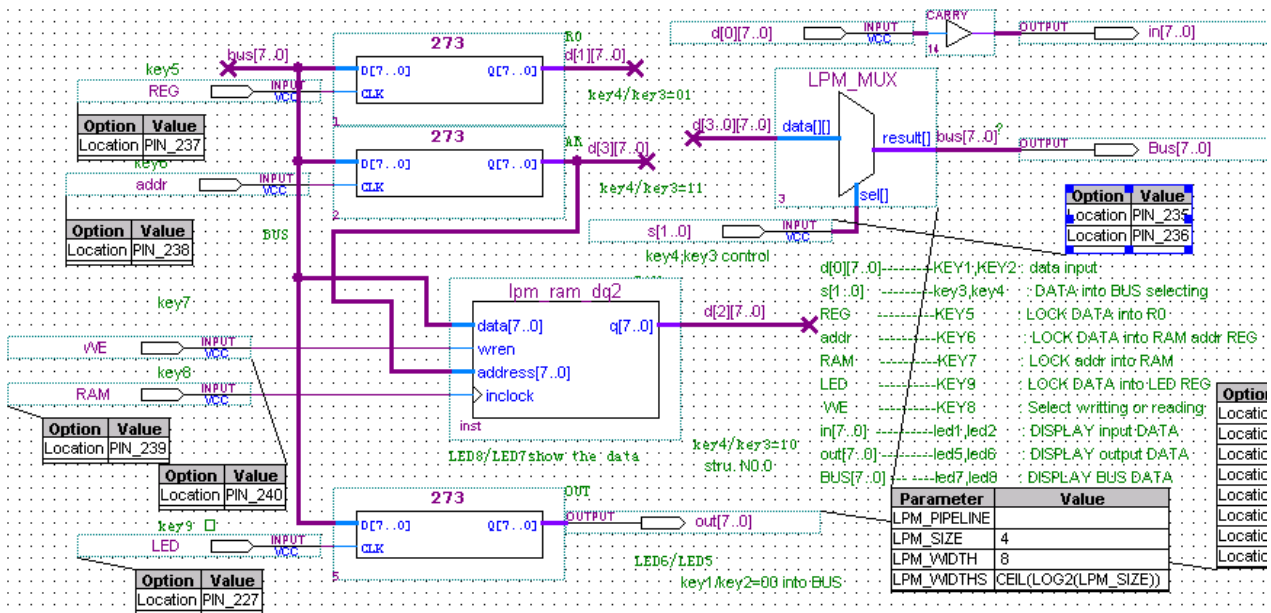


图 5-3 总线控制实验线路图

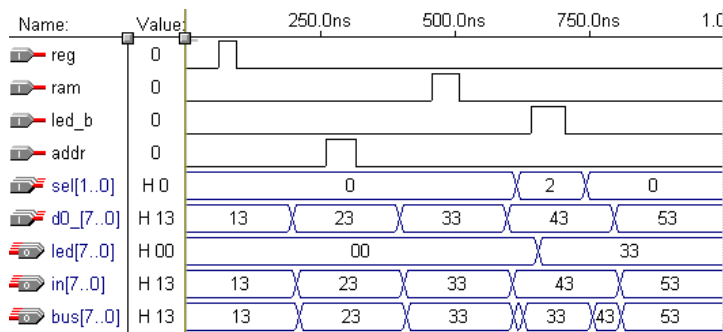


图 5-4 总线控制的时序仿真波形图

练习题编号	功 能	助 记 符
1	把 INPUT 设置的数据写入 R0	IN R0, KEY
2	把 INPUT 设置的数据写入 RAM 某单元	IN RAM, KEY
3	把 RAM 某单元内容读入 R0	LD R0, RAM
4	把 R0 内容读入 RAM 某单元	ST RAM, R0
5	把 R0 内容输出显示	OUT LED, R0
6	把 RAM 某单元内容输出显示	OUT LED, RAM
7	把 INPUT 设置的数据直接输出显示	OUT LED, KEY

5. 将用图形设计的实验电路改写成用 VHDL 语言设计的实验电路。

## 实验六 基本模型机设计与实现

### 一. 实验目的

1. 深入理解基本模型计算机的功能、组成知识；
2. 深入学习计算机各类典型指令的执行流程；
3. 学习微程序控制器的设计过程和相关技术，掌握 LPM\_ROM 的配置方法。
4. 在掌握部件单元电路实验的基础上，进一步将单元电路组成系统，构造一台基本模型计算机。

5. 定义五条机器指令，并编写相应的微程序，上机调试，掌握计算机整机概念。掌握微程序的设计方法，学会编写二进制微指令代码表。

6. 通过熟悉较完整的计算机的设计，全面了解并掌握微程序控制方式计算机的设计方法。

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt6 / 实验 6-1.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt6 / CPU5\_GW48CP+ ; 显示屏为 LCD128X64

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt6 / CPU5\_GW48CP++ ;显示屏为 LCD240X128

二. 实验原理

1. 在部件实验过程中，各部件单元的控制信号是人为模拟产生的，而本实验将能在微过程控制下自动产生各部件单元控制信号，实现特定的功能。实验中，计算机数据通路的控制将由微过程控制器来完成，CPU 从内存中取出一条机器指令到指令执行结束的一个指令周期，全部由微指令组成的序列来完成，即一条机器指令对应一个微程序。

2. 指令格式

(1) 指令格式

采用寄存器直接寻址方式，其格式如下：

位	7	6	5	4	3	2	1	0
功能	OP-CODE				rs		rd	

其中，OP-CODE 为操作码，rs 为源寄存器，rd 为目的寄存器，并规定：

Rs 或 rd	选定的寄存器
00	R0
01	R1
10	R2

本实验采用五条机器指令：IN（输入）、ADD（二进制加法）、STA（存数）、OUT（输出）、JMP（无条件转移），其指令格式如下（最高 4 位二进制数为操作码）：

指令表

助记符	机器指令码	Addr 地址码	功能说明
IN	0 0H		“INPUT” 中的数据→R0
ADD addr	1 0H	XX H	R0+[addr] →R0
STA addr	2 0H	XX H	R0 → [addr]
OUT addr	3 0H	XX H	[addr] → BUS
JMP addr	4 0H	XX H	addr →PC

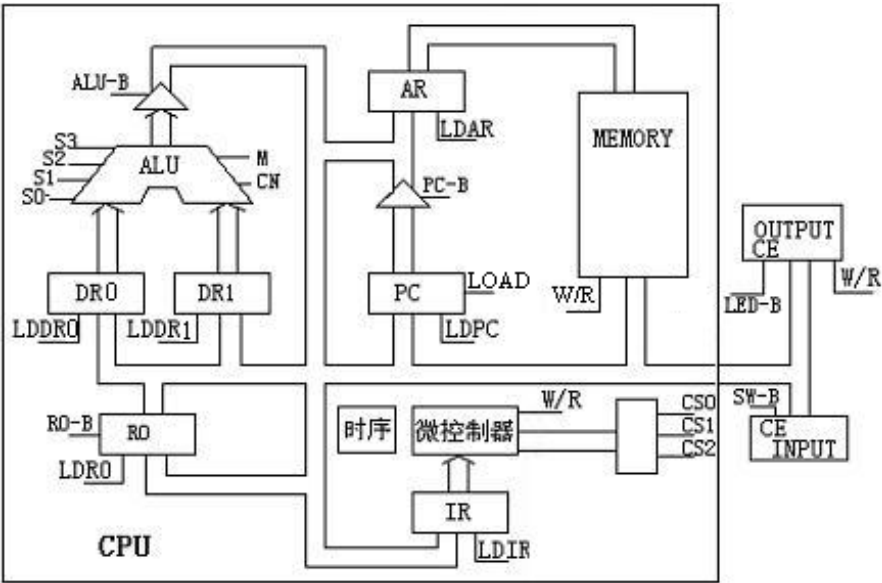


图 6-1 数据通路框图

其中 IN 为单字长（8 位二进制），其余为双字长指令，XX H 为 addr 对应的十六进制地址码。为了向 RAM 中装入程序和数据，检查写入是否正确，并能启动程序执行，还必须设计三个控制台操作微程序。

1、存储器读操作（KRD）：下载实验程序后按总清除按键（CLR）后，控制台 SWA、SWB 为“0 0”时，可对 RAM 连续手动读出操作。

2、存储器写操作（KWE）：下载实验程序后按总清除按键（CLR）后，控制台 SWA、SWB 为“0 1”时，可对 RAM 连续手动写操作。

3、启动程序（RP）：下载实验程序后按总清除按键（CLR）后，控制台 SWA、SWB 为“1 1”时，即可转入到微地址“01”号“取指令”微指令，启动程序运行。

根据以上要求设计数据通路框图，如图 6-1 所示。

SWB	SWA	控制台指令
0	0	读内存（KRD）
0	1	写内存（KWE）
1	1	启动程序（RP）

表 6-1 24 位微代码定义：

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
S3	S2	S1	S0	M	Cn	WE	A9	A8	A			B			C			uA5	uA4	uA3	uA2	uA1	uA0

表 6-2 A、B、C 各字段功能说明：

A 字段				B 字段				C 字段			
15	14	13	选择	12	11	10	选择	9	8	7	选择
0	0	0		0	0	0		0	0	0	
0	0	1	LDRi	0	0	1	RS-B	0	0	1	P（1）
0	1	0	LDDR1	0	1	0		0	1	0	
0	1	1	LDDR2	0	1	1		0	1	1	
1	0	0	LDIR	1	0	0		1	0	0	P（4）
1	0	1	LOAD	1	0	1	ALU-B	1	0	1	LDAR
1	1	0	LDAR	1	1	0	PC-B	1	1	0	LDPC

24 位微代码中各信号的功能

- (1) uA5—uA0：微程序控制器的微地址输出信号，是下一条要执行的微指令的微地址。
- (2) S3、S2、S1、S0：由微程序控制器输出的 ALU 操作选择信号，以控制执行 16 种算术操作或 16 种逻辑操作中的某一种操作。
- (3) M：微程序控制输出的 ALU 操作方式选择信号端。M=0 执行算术操作；M=1 执行逻辑操作。
- (4) Cn：微程序控制器输出的进位标志信号，Cn=0 表示 ALU 运算时最低位有进位，Cn=1 则表示无进位。
- (5) WE：微程序控制器输出的 RAM 控制信号。当/CE=0 时，如 WE=0 为存储器读；如 WE=1 为存储器写。
- (6) A9、A8——译码后产生 CS0、CS1、CS2 信号，分别作为 SW\_B、RAM、LED 的选通控制信号。
- (7) A 字段（15、14、13）——译码后产生与总线相连接的各单元的输入选通信号（见表 6-1）。
- (8) B 字段（12、11、10）——译码后产生与总线相连接的各单元的输出选通信号。
- (9) C 字段（9、8、7）——译码后产生分支判断测试信号 P(1)~P(4)和 LDPC 信号。

系统涉及到的微程序流程见图 6-2。当执行“取指令”微指令时，该微指令的判断测试字段为 P(1)测试。由于“取指令”微指令是所有微程序都使用的公用微指令，因此 P(1)的测试结果出现多路分支（见图 6-2 左图）。用指令寄存器的高 4 位（IR7-IR4）作为测试条件，出现 5 路分支，占用 5 个固定地址单元。

控制台操作为 P(4)测试（见图 6-2 右图），它以控制台信号 SWB、SWA 作为测试条件，出现了 3 路分支，占用 3 个固定微地址单元。当分支微地址单元固定后，剩下的其它地方就可以一条微指令占用控制存储器的一个微地址单元，随意填写。注意：微程序流程图上的微地址为 8 进制！

当全部微程序设计完毕后，应将每条微指令代码化，表 6-2 即为图 6-2 的微程序流程图按微指令格式转化而成的“二进制微代码表”。

表 6-2 二进制微代码表

微地址	微指令	S3	S2	S1	S0	M	CN	WE	A9	A8	A	B	C	UA5—UA0
0 0	0 1 8 1 1 0	0	0	0	0	0	0	0	1	1	0 0 0	0 0 0	1 0 0	0 1 0 0 0 0
0 1	0 1 E D 8 2	0	0	0	0	0	0	0	1	1	1 1 0	1 1 0	1 1 0	0 0 0 0 1 0
0 2	0 0 C 0 4 8	0	0	0	0	0	0	0	0	1	1 0 0	0 0 0	0 0 1	0 0 1 0 0 0
0 3	0 0 E 0 0 4	0	0	0	0	0	0	0	0	1	1 1 0	0 0 0	0 0 0	0 0 0 1 0 0
0 4	0 0 B 0 0 5	0	0	0	0	0	0	0	0	1	0 1 1	0 0 0	0 0 0	0 0 0 1 0 1
0 5	0 1 A 2 0 6	0	0	0	0	0	0	0	1	1	0 1 0	0 0 1	0 0 0	0 0 0 1 1 0
0 6	9 1 9 A 0 1	1	0	0	1	0	0	0	1	1	0 0 1	1 0 1	0 0 0	0 0 0 0 0 1
0 7	0 0 E 0 0 D	0	0	0	0	0	0	0	0	1	1 1 0	0 0 0	0 0 0	0 0 1 1 0 1
1 0	0 0 1 0 0 1	0	0	0	0	0	0	0	0	0	0 0 1	0 0 0	0 0 0	0 0 0 0 0 1
1 1	0 1 E D 8 3	0	0	0	0	0	0	0	1	1	1 1 0	1 1 0	1 1 0	0 0 0 0 1 1
1 2	0 1 E D 8 7	0	0	0	0	0	0	0	1	1	1 1 0	1 1 0	1 1 0	0 0 0 1 1 1
1 3	0 1 E D 8 E	0	0	0	0	0	0	0	1	1	1 1 0	1 1 0	1 1 0	0 0 1 1 1 0
1 4	0 1 E D 9 6	0	0	0	0	0	0	0	1	1	1 1 0	1 1 0	1 1 0	0 1 0 1 1 0
1 5	0 3 8 2 0 1	0	0	0	0	0	0	1	1	1	0 0 0	0 0 1	0 0 0	0 0 0 0 0 1
1 6	0 0 E 0 0 F	0	0	0	0	0	0	0	0	1	1 1 0	0 0 0	0 0 0	0 0 1 1 1 1
1 7	0 0 A 0 1 5	0	0	0	0	0	0	0	0	1	0 1 0	0 0 0	0 0 0	0 1 0 1 0 1
2 0	0 1 E D 9 2	0	0	0	0	0	0	0	1	1	1 1 0	1 1 0	1 1 0	0 1 0 0 1 0
2 1	0 1 E D 9 4	0	0	0	0	0	0	0	1	1	1 1 0	1 1 0	1 1 0	0 1 0 1 0 0
2 2	0 1 A 0 1 0	0	0	0	0	0	0	0	1	1	0 1 0	0 0 0	0 0 0	0 1 0 0 0 0
2 3	0 1 8 0 0 1	0	0	0	0	0	0	0	1	1	0 0 0	0 0 0	0 0 0	0 0 0 0 0 1
2 4	0 6 2 0 1 1	0	0	0	0	1	1	0	0	0	0 1 0	0 0 0	0 0 0	0 1 0 0 0 1
2 5	0 1 0 A 0 1	0	0	0	0	0	0	0	1	0	0 0 0	1 0 1	0 0 0	0 0 0 0 0 1
2 6	0 0 D 1 8 1	0	0	0	0	0	0	0	0	1	1 0 1	0 0 0	1 1 0	0 0 0 0 0 1

指令寄存器（IR）：指令寄存器用来保存当前正在执行的一条指令。当执行一条指令时，先把它从内存取到缓冲寄存器中，然后再传送至指令寄存器。指令划分为操作码和地址码段，由二进制数构成，为了执行任何给定的指令，必须对操作码进行测试“P(1)”，通过节拍脉冲 T4 的控制，以便识别所要求的操作。

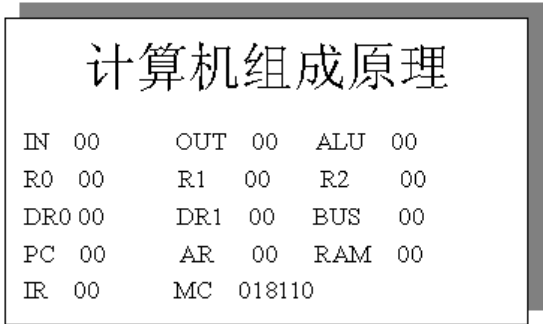


图 6-2 LCD 液晶显示屏

指令译码器：根据指令中的操作码强置微控制器单元的微地址，使下一条微指令指向相应的微程序首地址。

实验中 LCD 液晶显示屏可以用来显示模型机 CPU 中各组成单元的内容。将 CPU\_5.sof 文件下载到实验台后，按系统复位键，LCD 液晶显示屏即显示 CPU 中各组成单元的内容。其功能说明如表 6-3 所示：

表 6-3 LCD 液晶显示屏功能说明

名称	作用	名称	作用
IN	输入单元 INPUT	DR1	暂存器 DR1
OUT	输出单元 OUTPUT	DR2	暂存器 DR2
ALU	算术逻辑单元	PC	程序计数器
BUS	内部数据总线	AR	地址寄存器
R0	寄存器 R0	RAM	程序/数据存储器
R1	寄存器 R1	IR	指令寄存器
R2	寄存器 R2	MC	微程序控制器

地址(16进制)	内容(16进制)	助记符	说明
00	00	IN	“INPUT DEVICE” → R0
01	10	ADD [0AH]	R0+[0AH] → R0
02	0A		
03	20	STA [0BH]	R0 → [0BH]
04	0B		
05	30	OUT [0BH]	[0BH] → OUT 输出口
06	0B		
07	40	JMP [12H]	12H → PC
08	12		
09	00		
0A	34		自定
0B			求和结果

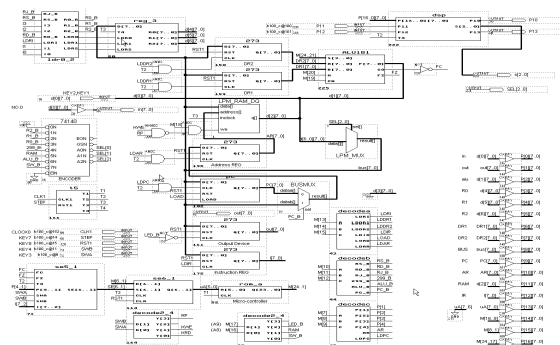


图 6-3 模型计算机电路原理图

实验程序 1:

说明: 1、指令 IN 为单字节指令, 指令码: 00, 其功能是将输入口 IN 的数据送到寄存器 R0;

2、指令 ADD [0AH]为双字节指令, 指令码: 100A, 其功能是将 R0 中的数据加上 RAM 地址 0AH 中的数据 (34H);

3、指令 STA [0BH]为双字节指令, 指令码: 200B, 其功能是将 R0 中的数据送到 RAM 的 0BH 地址单元中;

4、指令 OUT[0BH]为双字节指令, 指令码: 300B, 其功能是将 RAM 的 0BH 地址单元中的数据送到 OUT 输出口上;

5、指令 JMP [12H] 为双字节指令, 指令码: 4012, 其功能是将其操作码下一地址单元 (08H) 的数据作为转跳地址。

### 三. 实验步骤

1. 微程序的输入: 根据表 6-2 所对应的二进制微代码, 编辑 LPM\_ROM 配置文件 ROM\_11.mif(参考 demoD\_cpu5 文件夹中的同名文件), 并将其保存在与实验电路 CPU\_5.bdf 工程所在的文件夹中, 与实验电路 CPU\_5.bdf 一同编译后, 得到下载文件 CPU\_5.sof。下载配置文件 CPU\_5.sof 下载到实验系统。实验板上的时钟 clock0 选择输入频率为 1.5MHz。图 6-3 是示例原理图, 详见 CPU\_5.bdf。

2. 输入模型机的程序(示例工程文件是 CPU\_5.bdf)

#### (一) 手动写入

(1) 使用控制台 KWE 和 KRD 微程序将机器指令程序 (“实验程序 1”: 按地址输入指令代码, 如地址 00、01、02、03、04... 分别对应指令码 00、10、0A、20、0B...) 装入模型机 CPU 的程序 RAM(LPM\_RAM\_DQ) 中, 并进行检查。根据图 6-2 控制台微程序流程图, 在微指令的控制下, 依次输入机器指令代码:

① 以下将数据 35、C4 依次装入 00、01 地址为例: 选择实验模式 N0.0, 输入数据显示于数码 2、1 上;

② 将控制开关 SWB、SWA (键 4、键 3) 设置为: 0、1; 模型机的复位控制信号 RST (键 8) =1;

③ 机器指令代码的数据输入由键 2、键 1 输入, 先键入 35, 再按两次键 7, 即 0->1->0, 产生一个写入正脉冲, 这时观察右上液晶屏上的输入端口 IN=35; PC=00 (当前将要输入的地址); MC=018110 微指令。

再按两次键 7 (地址寄存器加 1), 根据图 5-2 控制台微程序流程图, 进入到 KWE (01) 分支, 进入并执行了微地址 “21” 中的操作, 这时控制此操作的微指令码 MC=01ED94, PC 自动加 1, PC=1。

④ 按键 7, 再产生一个脉冲, 进入并执行了微地址 “24” 中的操作; 观察液晶, 数据 35 进入总线 BUS=35, 35 进入 RAM=35, 此时微指令码 MC=062011, 此时将机器指令代码数据写入了 LPM\_RAM 中;

⑤ 此后每当出现 MC=062011 时, 即可利用键 2, 键 1 输入待写入 RAM 的数据, 此时如 C4, 连续按键 7, 再产生 2 个脉冲, 即将 C4 写入 RAM, PC 加 1, 微指令码变成 MC=062011;

⑥ 重复③—⑤的步骤, 将 “实验程序 1” 的全部机器指令代码输入 RAM。

(2) 以下是检查 RAM 中的内容。当全部机器指令代码输入模型机后, 在微指令的控制下, 依次检查 LPM\_RAM 中已输入的机器指令代码。步骤如下:

① 按复位键 8=1, 使模型机中的 PC 复位;

② 将控制开关 SWB、SWA (键 4、键 3) 设置为: 0、0;

③ 复位信号 RST (键 8) =0;

④ 按键 7, 每两个 2 次单步运行 (产生 2 个正脉冲), 可读出 LPM\_RAM 中以写入的数据; 根据图 6-1 的 CPU 部件和信息流程, 对于读出的每一数据, 仔细观察液晶上显示的 MC、PC、AR、IN、BUS、RAM、DR0、DR1 的数据变化。重复以上步骤, 依次检查 LPM\_RAM 中已输入的机器指令代码。

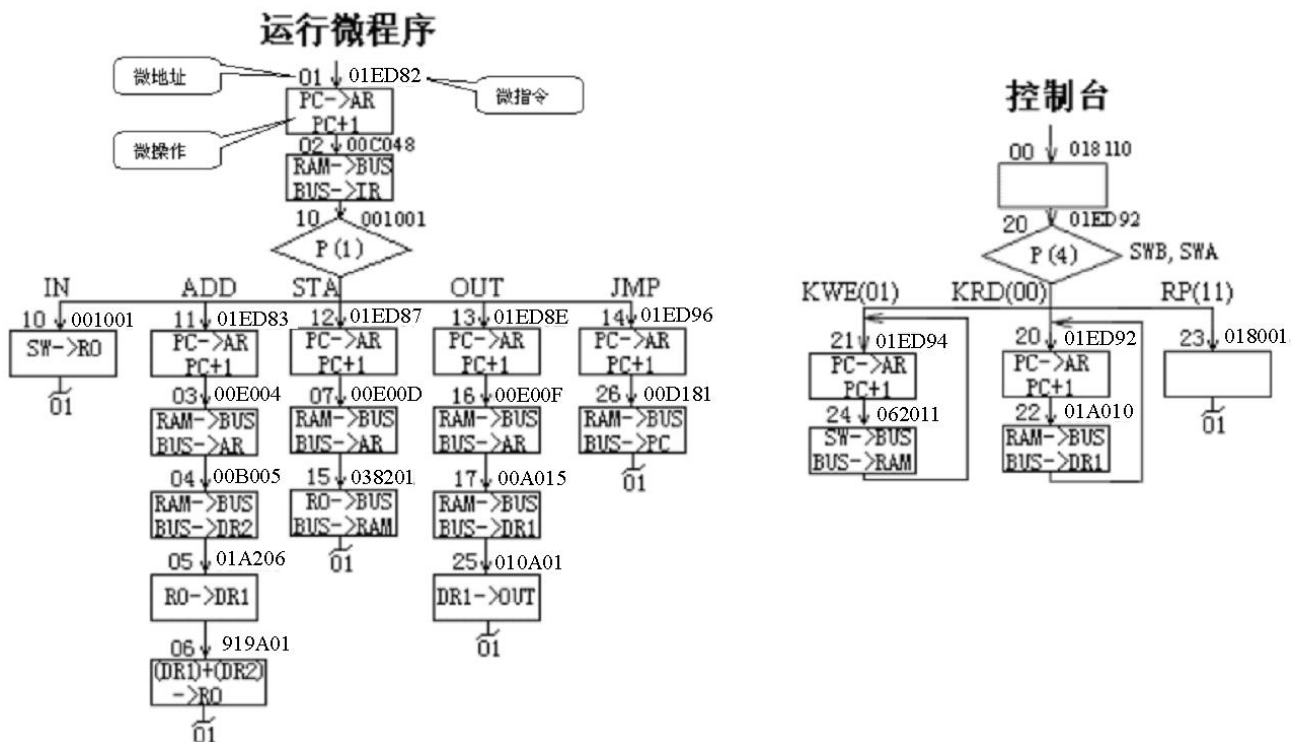


图 6-2 微程序流程图(注,图中的 DR1 应改为 DR0, DR2 应改为 DR1)

Instance	Instance ID	Status	Width	Depth	Type	Mode	JTAG Chain
0	rom6	Not run...	24	64	RAM/ROM	Read/Write	Hardware: ByteBlaster
1	ramc	Not run...	8	256	RAM/ROM	Read/Write	

000000	01	81	10	00	ED	82	00	C0	48	00	E0	04	00	E0	05	01	A2	06	91	9A	01	00	E0	0D
000008	00	10	01	00	ED	83	00	ED	87	00	ED	8E	00	ED	96	03	82	01	00	E0	0F	00	A0	15
000010	01	ED	92	01	ED	94	00	A0	10	00	80	01	06	20	11	07	0A	01	00	D1	81	00	00	00
000018	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000028	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000038	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000000	00	10	0A	20	0B	30	0B	40	12	00	34	34	00	00	00	00	00	00	00	00	00	00	00	00
000019	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000032	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00004B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000064	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00007D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000096	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000AF	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000C8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000E1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000FA	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

图 6-3 用在系统 EAB 读写工具对 FPGA 中的 ROM 和 RAM 进行观察和改写

## (二) 自动配置 LPM\_RAM

如果程序量大, 手动输入效率太低, 可以在计算机上编译好代码文件, 并随同模型 CPU 设计文件一同编译进 SOF 下载文件中, 直接下载进入 FPGA。

(1) 在 QuartusII 环境下, 打开工程文件 CPU\_5.bdf, 修改 CPU\_5.bdf 中 LPM\_RAM\_DQ 的参数, 将初始化文件 LPM\_FILE 设置为: “./5\_ram.mif”; 打开 “5\_ram.mif” (在示例中已有此文件), 根据 “实验程序 1”, 在 5\_ram.mif 中输入全部机器指令代码 (示例中已经输入)。

(2) 将工程文件重新编译后, 下载到实验台中, 即完成 LPM\_RAM 的配置。

(3) 根据以上的方法, 复位信号 RST (键 8) =1; 将控制开关 SWB、SWA (键 4、键 3) 设置为: 0、0, 按键 7, 每两个 2 次单步运行 (产生 2 个正脉冲), 检验配置进入 FPGA 中的程序代码。

## (三) 执行程序

(1) 按 1 次系统复位键 8, 并置键 8 为高电平, 使 CPU 允许正常工作;

(2) 控制开关 (键 4、键 3) 设置为 SWB、SWA=1,1, 处于程序执行方式, 观察图 6-1 控制台: RP (11);

(3) 通过键 2、键 1 输入运算数据, 如 56H,

注意, 1、实验箱上数码管 7、6 显示的是下一节拍将要执行的微指令的微地址码;

## 2、数码管 3 显示的是进位情况，有进位，LED3 显示 1，无进位 LED3 显示 0。

下面以列表 6-4 的方式说明微程序的执行过程，每按 2 次键 7 产生一个 STEP 单步脉冲，执行一个微操作。微地址 uA 以八进制表示，微指令 MC 以十六进制表示。

表 6-4 微指令执行情况

STEP	后续 uA 微地址	MC 微指令	PC	IR 指令	完成功能	执行结果	
1	00	018110	00	00	控制台（读/写/运行）功能判断	控制台操作	
2	23	018101			SWB、WSA=（11）转 RP，分支转移		
3	01	008001			转程序执行方式		
4	02	01ED82	01	00	执行第 1 条指令。（输入 IN）	PC→AR=00H, PC+1=01H, AR 指向指令地址	
5	10	00C048	02		取指令, 将 RAM 中的指令送指令寄存器	RAM(00H)=00→BUS→IR=00H	
6	01	001001			接收键 1、2 输入的数据, 送 R0 寄存器	R0=56H, 键 1、键 2 输入数据 56H	
7	02	01ED82			执行第 2 条指令,（加法 ADD）	PC→AR=01H, PC+1=02H, AR 指向指令地址	
8	09	00C048		10	取指令	RAM(01H)=10H→BUS→IR=10H	
9	03	01ED83	03		间接寻址, AR 指向取数的间接地址	PC→AR=02H, PC+1=03H, RAM=10H	
10	04	00E004			取数地址送 AR	RAM(10)=0AH→BUS→AR=0AH	
11	05	00B005			从 RAM 中取数送 DR2	RAM(0AH)=34H→BUS→DR2=34H	
12	06	01A206		将 R0 的数据送 DR1	(R0)=56H→BUS→DR1=56H		
13	01	919A01	04	20	加法运算: (DR1)+(DR2)→R0	56H+34H=8AH→R0=8AH	
14	02	01ED82			执行第 3 条指令（存储 STA）	PC→AR=03H, PC+1=04H	
15	12	00C048			05	取指令	RAM(03H)=20H→BUS→IR=20H
16	07	01ED87				间接寻址, AR 指向存数的间接地址	PC→AR=04H, PC+1=05H
17	15	00E00D	存数的地址送 AR	RAM(04)=0BH→BUS→AR=0BH			
18	01	038201	R0 的内容存入 RAM(0BH)单元	(R0)=8AH→BUS→RAM(0BH)=8AH			
19	02	01ED82	06	30	执行第 4 条指令（输出 OUT）	PC→AR=05H, PC+1=06H	
20	13	00C048	07		取指令	RAM(05H)=30H→BUS→IR=30H	
21	16	01ED8E			间接寻址, AR 指向取数的间接地址	PC→AR=06H, PC+1=07H	
22	17	00E00F			取数地址送 AR	RAM(06)=0BH→BUS→AR=0BH	
23	25	00A015			从 RAM 中取数送 DR1	RAM(0BH)=8AH→BUS→DR1=8AH	
24	01	010A01			DR1 的内容送输出单元 OUT	DR1=8AHH→BUS→OUT=8AH	
25	02	01ED82			08	执行第 5 条指令（转移 JMP）	PC→AR=07H, PC+1=08H
26	14	00C048		09	40	取指令	RAM(07H)=40H→BUS→IR=40H
27	26	01ED96	40		间接寻址, AR 指向转移的间接地址	PC→AR=08H, PC+1=09H	
28	01	00D181			转移地址送 PC, 转到 00H。	RAM(08H)=00→BUS→PC=00H	
29	02	01ED82	01		执行第 1 条指令——程序循环	PC→AR=00H, PC+1=01H	
30	10	00C048	01	00	取指令		
...							

### （四）EAB 在系统读写

使用在系统 EAB 读写工具对模型 CPU 中的存放微程序的 ROM 和存放程序与数据的 RAM 进行观察和改写（图 6-3）。

### （五）用嵌入式逻辑分析仪了解 CPU 运行情况

可以利用实验系统上的液晶屏上的数据显示和嵌入式逻辑分析仪同时了解 CPU 的每一单步运行情况（图 6-4）。注意，图 6-4 的嵌入式逻辑分析仪设置情况：采用时钟使用 CPU 的工作时钟（CLK1=1.5MHz），采样深度 64 位，触发位置：Pre..，触发信号用单步控制信号：STEP；触发方式：上升沿。

左侧的观察信号，data[1]是 ALU；data[2]是 RAM；P[10]是 AR；P[12]是 IR；

CPU 运行的逻辑分析仪波形数据如图 6-5 所示。

### （六）实验要求

1. 实验之前应认真准备，写出实验步骤和具体设计内容；
2. 实验前应掌握所有控制信号的作用；
3. 掌握在 QuartusII 环境下，采用图形编辑方法的设计技术；
4. 掌握在微程序控制下机器指令的写入、读出、和程序执行方法；
5. 掌握 LPM\_RAM 的配置方法，实现对机器指令输入；
6. 掌握微程序的设计方法，学会编写二进制微指令代码表。
7. 掌握对 LPM\_ROM 的配置方法，实现微指令代码表的输入。



7、通过液晶屏，观察各相关寄存器、ALU、DR1、PC、IR、AR、BUS、MC 等内容的变化情况，根据表 6-2 微程序控制流程，单步跟踪微程序的执行情况。通过 INPUT（键 2、键 1）输入运算数据，跟踪程序的执行情况，并详细记录每条微指令执行后，相关单元输出数据的变化情况，依次执行机器指令，从而验证所设计的正确性。在完成基本验证实验后，根据这 5 条指令，自行设计程序、输入和调试，记录实验数据。

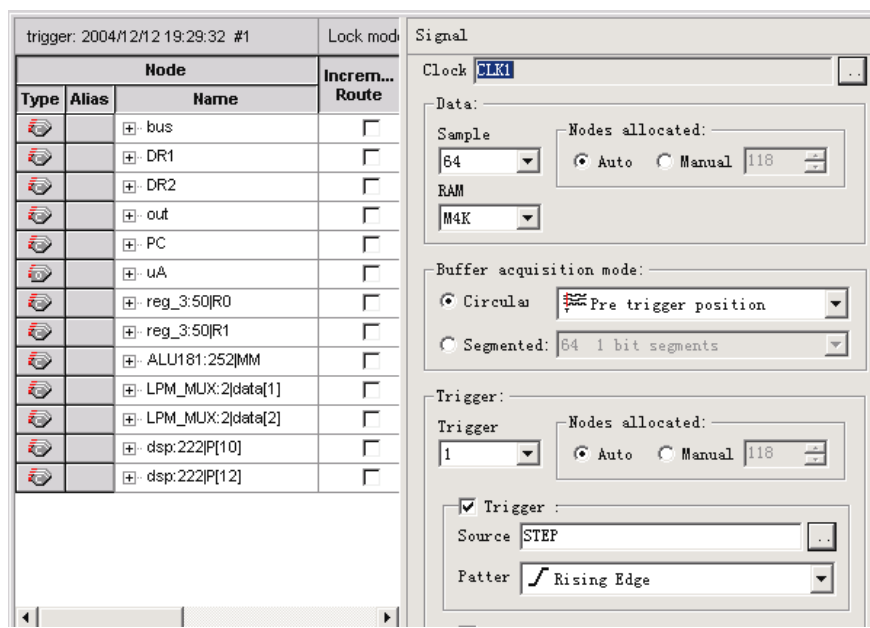


图 6-4 嵌入式逻辑分析仪设置情况

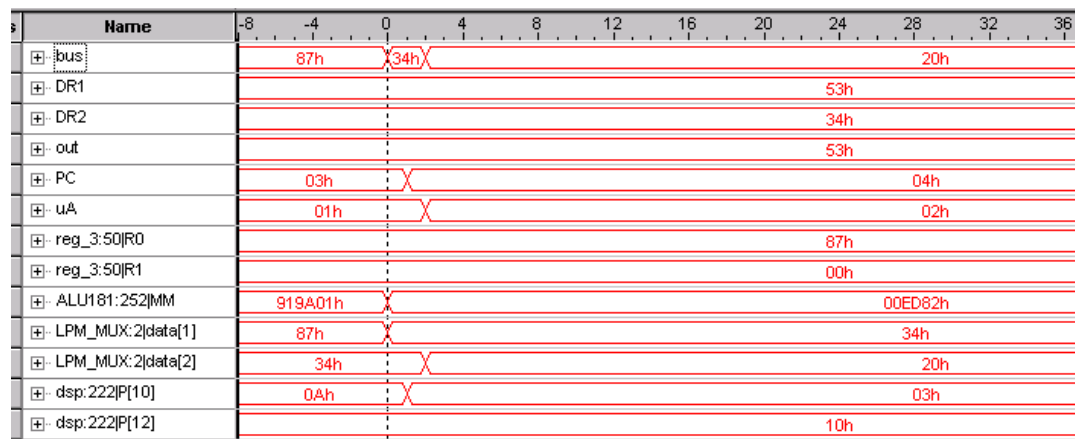


图 6-5 嵌入式逻辑分析仪采样波形数据

(七) 实验报告

- 1. 实验的原理，实验步骤和具体实验结果；
- 2. 实验中遇到的主要问题和分析解决问题的思路；
- 3. 通过实验，自己的学习经验和切身体会，以及对教学实验的意见和建议。

(八) 思考题实验题

- 1. 若要修改算术逻辑单元 ALU 的功能，应如何用 VHDL 语言来实现？如何编程、如何用计算机进行功能仿真、时序仿真？如何通过 GW48 实验系统进行硬件功能验证？通过具体实验来回答。
- 2. 用 VHDL 语言设计程序计数器单元 PC。
- 3. 用 VHDL 语言设计地址寄存器单元 AR。
- 4. 用 VHDL 语言设计指令译码器单元 IR。
- 5. 将整个模型 CPU 都用用 VHDL 进行表达。
- 6. 除了已有的 IN、ADD、STA、OUT、JMP 指令外，再设计减法指令 SUB、带进位加 ADDC、逻辑与 AND、逻辑或 OR 和异或 XOR 共 10 条指令，编写相应微程序流程图，写出微程序代码表和相应的 mif 文件，并配置进 LPM\_ROM 中；编写由宏程序代码组成的 3 个程序，并用此 CPU 完成相应的程序功能。
- 7. 适当修改 ALU181.VHD，使之对不同操作产生的进位都能保留（锁存）。

实验操作说明： 下载 B100\_C.SOF；选模式 0，按一次复位键；8 位数据 in[7..0]由键 2、键 1 输入（此值显示于键对应



的数码管上); SWB, SWA 由键 4, 键 3 控制工作模式: 模式 00→KRD (读出); 模式 01→KWR (写入); 模式 11→RP (程序执行)。RST1 →模型 CPU 复位, 键 8 控制, 低电平有效。工作时应置高电平。STEP →键 7 控制, 单步执行键; Clock0 一选择 1.5MHz, 是 CPU 工作时钟。通过 LCD 液晶显示屏观察“基本模型机 CPU”中各基本工作单元的内容。

8. 将图 6-6 结合硬件实验详细说明此 CPU 对“实验程序 1”的执行过程和图中各信号的含义及不同节拍下变化的原因和结果, 如 d0\_[7..0]、PC、AR、DR1、DR2、R0、CO、T4..T1、uA、bus[7..0]等等。

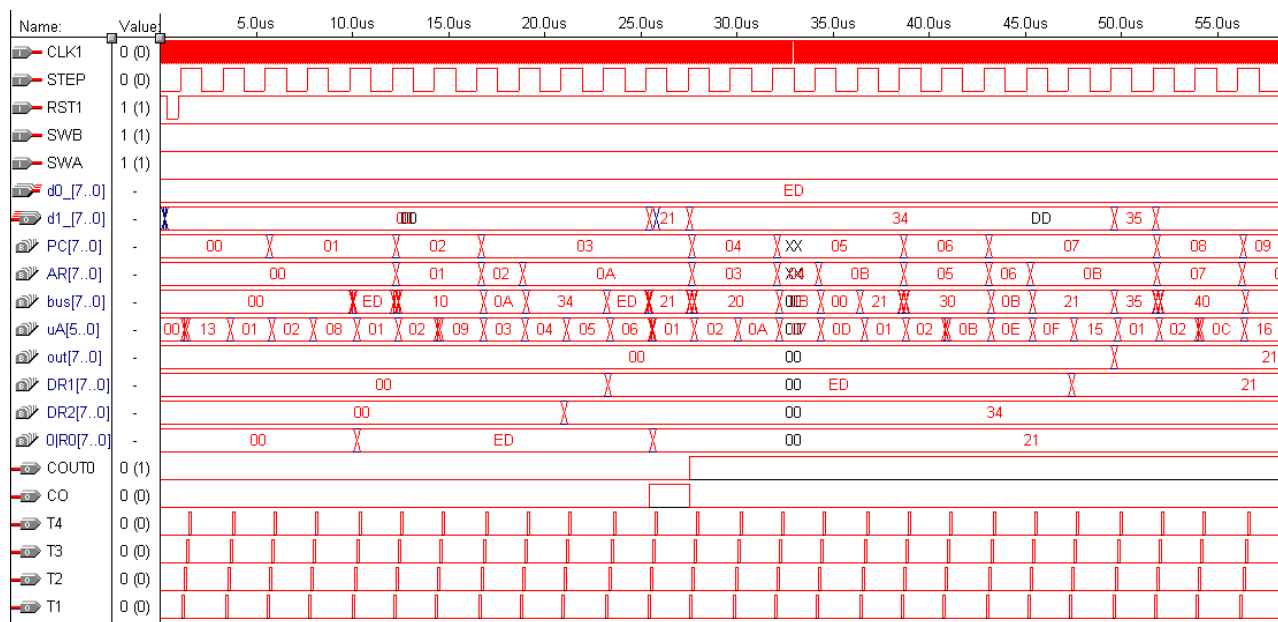


图 6-6 模型 CPU 执行“实验程序 1”的仿真波形图 (注。其中的 DR1, DR2 分别表示液晶屏上的 DR0/DR1)

### (九) 图 6-1 中各信号的功能说明

(1) uA5—uA0 微程序控制器的微地址输出信号。(2) IR7—IR5 指令寄存器 IR7、IR6、IR5 输出信号, 输入至微程序控制器作为修改微地址的控制信号。(3) CLK1——时钟信号源, 输入频率为 6~12MHz。

(4) T1~T4——时序信号发生器提供的四个标准输出信号, 可以采用单拍或连续两种方式输出。

(5) s3、s2、s1、s0、——由微程序控制器输出的 ALU 操作选择信号, 以控制执行 16 种算术操作或 16 种逻辑操作中的某一种操作。

(6) M—微程序控制输出的 ALU 操作方式选择信号端。M=0 执行算术操作; M=1 执行逻辑操作。

(7)  $\overline{cn}$ —微程序控制器的输出的进位标志信号,  $\overline{cn}=0$  表示 ALU 运算时最低位有进位  $\overline{cn}=1$ , 则表示无进位。(8)SWE—微程序控制器的微地址修改信号。(9)SRD—微程序控制器的微地址修改信号。

(10) RST1—清“0”信号输入端。(11)LDAR——微程序控制器的输入信号, 将程序计数器的内容打入到存储器地址寄存器 AR 中, 产生访问 RAM 的地址。

(12) /CE——微程序控制器输出的 RAM 选片信号, /CE=0 时, LPM\_RAM 单元被选中。

(13) WE——微程序控制器输出的 RAM 控制信号。当/CE=0 时, 如 WE=0 为存储器读; 如 WE=1 为存储器写。(14)BUS(7..0)——微程序控制器的内部数据总线。

(15) LDPC——微程序控制器输出的 PC 打入信号。(16) LOAD——微程序控制器的输出信号。LOAD=0 时, PC(程序计数器)处于并行置数状态; LOAD=1 时, PC 处于计数状态。

(17) ALU\_B—微程序控制器输出信号, 控制运算器的运算结果是否送到总线 BUS。低电平有效。

(18) PC\_B—微程序控制器输出信号, 控制程序计数器的内容是否送到总线 BUS, 低电平有效。

(19) R0\_B—微程序控制器输出信号, 控制寄存器 R4 的内容是否达到总线 BUS, 低电平有效。

(20) SW\_B—微程序控制器输出信号, 控制(键 2、键 1)的八位数据是否送到总线, 低电平有效。

(21) LDR0—微程序控制器的输出信号。控制把总线上的数据打入寄存器 DR0。

(22) LDR1—微程序控制器输出信号, 控制把总线上的数据打入寄存器 DR1

(23) LDIR—微程序控制器输出信号, 控制把总线上的数据(指令)输入到指令寄存器 IR 中。

(25) P(1)—微程序控制器输出的修改微地址 P(1), 标志信号。用于机器指令的微程序分支测试。

(26) uA—微程序控制器的微地址寄存器输出控制信号，uA=0，微地址信号输出。

(27) STEP—时序发生器启动控制信号。按 2 次 STEP 键，时序发生器可输出一组(单步)或连续的时序信号 T1、T2、T3、T4。

## 实验七 带移位运算的模型机设计与实现

### 一. 实验目的

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt7 / 实验 7-1.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt7 / CPU6\_GW48CP+；显示屏为 LCD128X64

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt7 / CPU6\_GW48CP++；显示屏为 LCD240X128

1. 在基本模型 CPU 基础上，增加移位运算单元 SHEFT，构建一台具有移位运算功能的模型 CPU。
2. 在实验六的 5 条基本机器指令基础上，增加 4 条移位运算指令，并编写相应的微程序，上机调试，掌握 CPU 整机概念。
3. 进一步熟悉较完整的 CPU 设计，全面了解掌握微程序控制方式 CPU 的设计方法。

### 二. 实验原理

在基本模型计算机实验的基础上，增加移位运算单元 SHEFT。

1. 数据格式：模型机采用定点补码表示法表示数据，字长为 8 位，其格式如下：

7	6	5	4	3	2	1	0
符号	尾数						

其中第 7 位为符号位，数值表示范围是： $-1 < X \leq 1$ 。

2. 指令格式：

(1) 算术逻辑指令：设计 9 条算术逻辑指令并用单字节表示，采用寄存器直接寻址方式，格式如下：

7	6	5	4	3	2	1	0
OP-CODE				rs		rd	

其中，OP-CODE 为操作码，rs 为源寄存器，rd 为目的寄存器，并规定：

Rs 或 rd	选定的寄存器
00	R0
01	R1
10	R2

(2) 移位运算器的功能（实验中所用到的带移位运算功能的模型计算机的数据通路框图如图 7-1 所示。）

G	S1	S0	M	功 能
0	0	0	任意	保持
0	1	0	0	循环右移
0	1	0	1	带进位循环右移
0	0	1	0	循环左移
0	0	1	1	带进位循环左移
任意	1	1	任意	装数

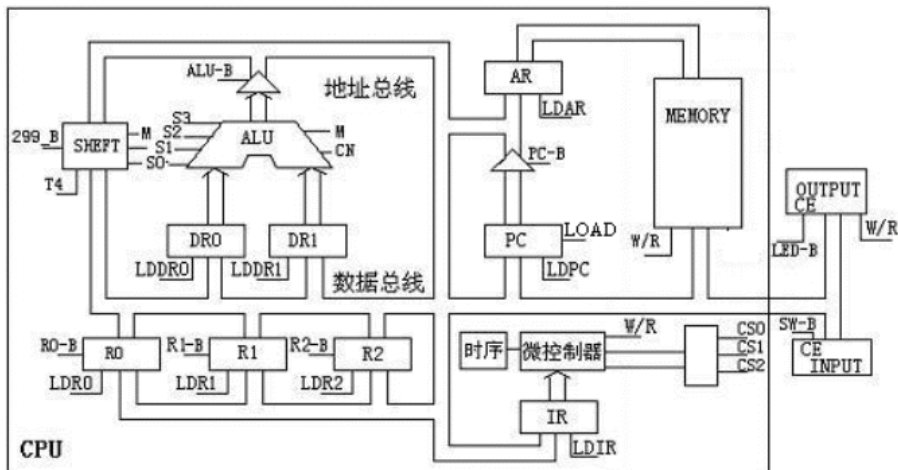


图 7-1 复杂模型机数据通路框图

微代码定义如表 7-1 所示。

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
S3	S2	S1	S0	M	Cn	WE	A9	A8	A	B	C	uA5	uA4	uA3	uA2	uA1	uA0						

表 7-2 带移位运算模型机微程序

微地址	微指令	S3	S2	S1	S0	M	CN	WE	A9	A8	A	B	C	uA5...uA0							
00	018108	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	
01	01ED82	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0	0	0	0	1	0
02	00C050	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	0	0
...略, 详见本实验工程文件 .\demoe_cpu6\b100_c.bdf 中元件 rom_a 中的文件 rom_11.mif																					
22	019801	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1
23	198824	0	0	0	1	1	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0
24	019801	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	1

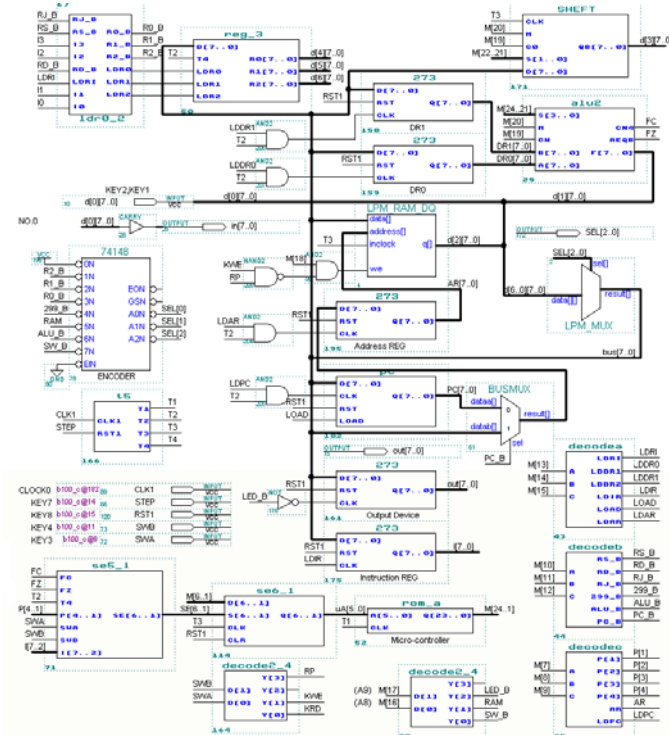


图 7-2 模型 CPU 内部结构图 (详见./demoE\_cpu6\b100\_c.bdf)

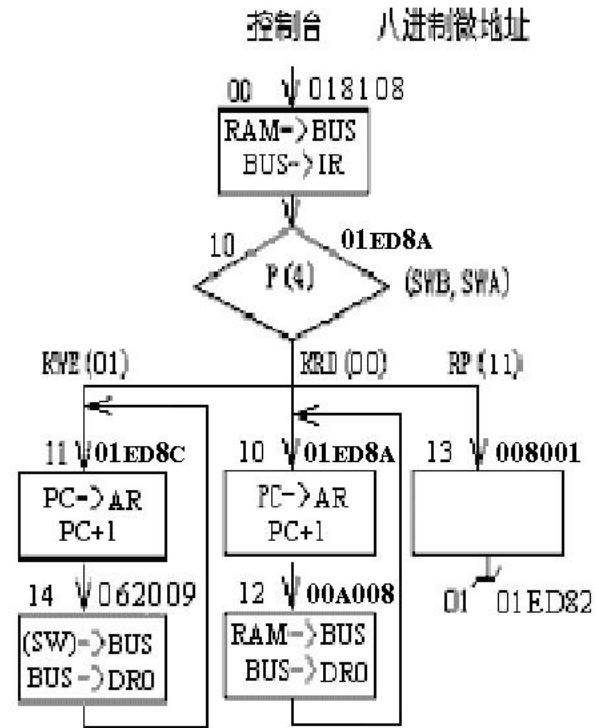


图 7-3 控制台操作流程

三. 实验步骤

(一)操作流程

1. 微程序的输入：表 7-2 所对应的二进制微代码可通过对 FPGA 中存放微代码的 LPM-ROM 配置文件 rom\_11.mif 的编辑、输入，并与实验电路一同编译后，得到下载文件 CPU\_6.sof。  
若采用 FPGA 外部 EEPROM 存放微程序，将专用的 EEPROM 配置文件下载到实验系统中。
2. 输入、编译实验电路文件：通过微机将编译通过的配置文件 CPU\_6.sof 下载到实验系统。
3. 输入的机器指令程序(按照上一章实验六介绍的方法)
  - (1) 使用，在微程序控制下，通过键盘将机器指令程序写入 LPM\_RAM 程序存储器中，并进行检查。
  - (2) 对照执行程序的机器指令，将实验程序 2 中的代码正确地写入到 LPM\_RAM 的中初始化配置文件 RAM\_6.mif 中（在 CPU\_6.bdf 中元件 LPM\_RAM\_DQ 中）。通过对 LPM-RAM 配置文件的编辑、输入，并与实验电路一同编译后，得到新的下载文件 CPU\_6.sof。

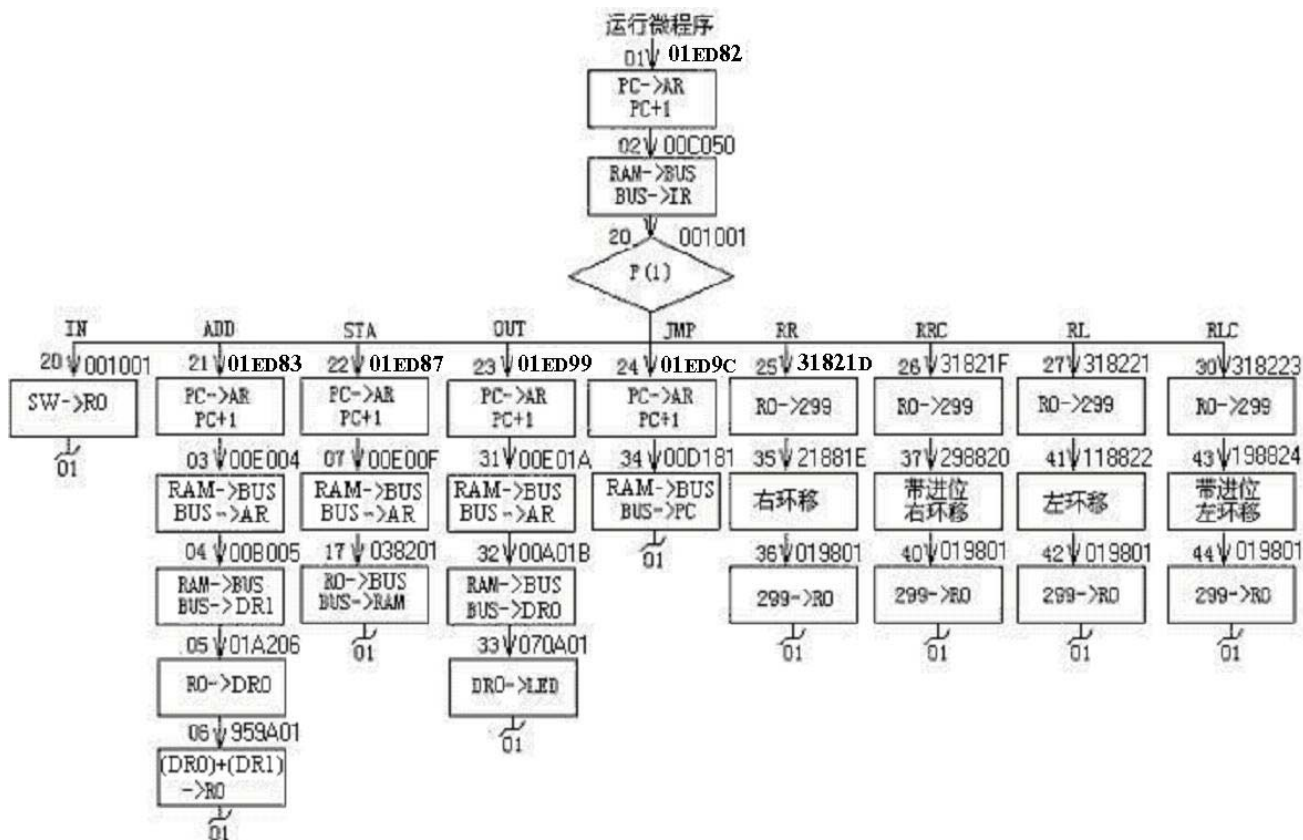
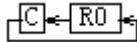




图 7-4 微程序流程图

#### 实验程序 2:

RAM 地址 (16 进制)	内容 (16 进制)	助记符	说 明
00	00	IN	"INPUT DEVICE" → RO
01	10	ADD [0DH]	RO+[0DH] → RO
02	0D		
03	80	RLC	
04	00	IN	"INPUT DEVICE" → RO
05	60	RRC	
06	70	RL	
07	20	STA [0EH]	RO → [0EH]
08	0E		
09	30	OUT [0EH]	[0EH] → OUTPUT
0A	0E		
0B	40	JMP [addr]	00 → PC
0C	00		自定
0D	45		存数单元
0E			

## (二)、执行程序

- (1) 按 1 次系统复位键 8，并置键 8 为低电平，使 CPU 允许正常工作；
- (2) 控制开关（键 4、键 3）设置为 SWB、SWA=1,1，处于程序执行方式，观察图 7-3 控制台：RP (11)；
- (3) 通过键 2、键 1 输入运算数据，如 56H，

下面以列表的方式说明微程序的执行过程，每按 2 次键 7 产生一个 STEP 单步脉冲，执行一个微操作。  
微地址 uA 以八进制表示，微指令 MC 以十六进制表示：

表 7-3 微指令执行情况

STEP	后续 uA 微地址	MC 微指令	PC	IR 指令	完成功能	执行结果
1	00	018108	00	00	控制台（读/写/运行）功能判断	控制台操作，微指令从 00H 开始执行
2	13	018101			SWB、WSA=（11）转 RP，分支转移	
3	01	008001			转程序执行方式	
4	02	01ED82	01	00	<b>执行第 1 条指令。</b> （输入 IN）	PC→AR=00H，PC+1=01H
5	20	00C050			取指令，将 RAM 中的指令送指令寄存器	RAM(00H)=00→BUS→IR=00H
6	01	001001			接收 IN 输入端口的数据，送 R0 寄存器	R0=56H，键 1、键 2 输入数据 56H
7	02	01ED82	02	10	<b>执行第 2 条指令，</b> （加法 ADD）	PC→AR=01H，PC+1=02H，指向指令地址
8	21	00C050			取指令，将 RAM 中的指令送指令寄存器	AR=01H，RAM=10H→BUS→IR=10H
9	03	01ED83			指向操作数地址	PC→AR=02H，PC+1=03H
10	04	00E004	03	10	间接寻址，以 RAM 内容做操作数地址	RAM(02H)=0DH→BUS→AR=0DH
11	05	00B005			将操作数送 DR2	RAM(0DH)=45H→BUS→DR2=45H
12	06	01A206			将 R0 的内容送 DR1	(R0)=56H→BUS→DR1=56H
13	01	959A01	04	80	完成加法运算：(DR1)+(DR2)→R0	56H+45H=9BH，ALU=9BH，R0=9BH
14	02	01ED82			<b>执行第 3 条指令，</b> 指向指令地址	PC→AR=03H，PC+1=04H
15	30	00C050			取指令（带 C 左循环 RLC）	IR=80H
16	43	318223	05	80	R0 的内容送移位寄存器 SFT	(R0)=9BH→BUS→SFT=9BH
17	44	198824			带进位 C 左循环	SFT=36H，BUS=36H
18	01	019801			结果送 R0	SFT=36H→BUS→R0=36H
19	02	01ED82	06	60	<b>执行第 4 条指令，</b> 指向指令地址	PC→AR=04H，PC+1=05H。键入数据 B7H
20	20	00C050			取指令（输入 IN）	IN=B7H，指令 IR=00H，
21	01	001001			接收键 1、2 输入的数据，送 R0 寄存器	R0=B7H
22	02	01ED82	07	70	<b>执行第 5 条指令，</b> 指向指令地址	PC→AR=05H，PC+1=06H
23	26	00C050			取指令（带 C 右循环 RRC）	指令 IR=60H
24	37	31821F			R0 的内容送移位寄存器 SFT	(R0)=B7H→BUS→SFT=B7H
25	40	298820	08	20	SFT 带进位 C 右循环	SFT=DBH
26	01	019801			结果送 R0	SFT=DBH→BUS→R0=DBH
27	02	01ED82			<b>执行第 6 条指令，</b> 指向指令地址	PC→AR=06H，PC+1=07H
28	27	00C050	09	30	取指令（左循环 RL）	指令 IR=70H
29	41	318221			R0 的内容送移位寄存器 SFT	(R0)=DBH→BUS→SFT=DBH
30	42	118822			SFT 不带进位左循环	SFT=B7H
31	01	019801	0A	40	结果送 R0	SFT=B7H→BUS→R0=B7H
32	02	01ED82			<b>执行第 7 条指令，</b> 指向指令地址	PC→AR=07H，PC+1=08H
33	22	00C050			取指令（存储 STA）	指令 IR=20H
34	07	01ED87	0B	40	间接寻址，以 RAM 内容做存数地址	PC→AR=08H，PC+1=09H，RAM=20H
35	17	00E00F			RAM 内容送地址寄存器 AR	RAM(08H)=0EH→BUS→AR=0EH
36	01	038201			将 R0 的内容存入 RAM(0EH)地址单元	R0=B7H→BUS→RAM(0EH)=B7H
37	02	01ED82	0C	40	<b>执行第 8 条指令，</b> 指向指令地址	PC→AR=09H，PC+1=0AH
38	23	00C050			取指令（输出 OUT）	RAM(09)=30H→BUS→IR=30H(指令)
39	31	01ED99			间接寻址，以 RAM 内容作取数地址	PC→AR=0AH，PC+1=0BH
40	32	00E01A	0D	40	RAM 内容送地址寄存器 AR	RAM(0AH)=0EH→BUS→AR=0EH
41	33	00A01B			从 RAM(0EH)取数，送 DR1	RAM(0EH)=B7H→BUS→DR1=B7H
42	01	070A01			DR1 的内容送 OUT 输出端口	(DR1)=B7H→BUS→OUT=B7H
43	02	01ED82	0E	40	<b>执行第 9 条指令，</b> 指向指令地址	PC→AR=0BH，PC+1=0CH
44	24	00C050			取指令（转移 JMP）	指令 IR=40H
45	34	01ED9C			间接寻址，以 RAM 内容作转移地址	PC→AR=0CH，PC+1=0DH
46	01	00D181	0F	40	将 RAM 内容送 PC，实现程序转移	RAM(0CH)=00.→BUS→PC=00H
47	02	01ED82			<b>执行第 1 条指令，程序循环</b>	PC→AR=00H，PC+1=01H
48	20	00C050			取指令	指令 IR=00H
...	01	001001		00	输入数据……	



## 四. 实验要求

1. 实验之前应认真准备，写出实验步骤和具体设计内容；
2. 实验前应了解所有控制信号的作用；
3. 在 Quartus II 环境下，用图形编辑方法设计带进位模型机的数据通路；
4. 在微程序控制下将机器指令的写入程序存储器中，并读出检查；和程序单步调试执行方法；
5. LPM\_RAM 的配置方法，实现对机器指令输入；
6. 微程序的设计方法，学会编写二进制微指令代码表。
7. 掌握对 LPM\_ROM 的配置方法，实现微指令代码表的输入。
8. 首先对已给的实例进行验证性实验，本实验工程文件在 CPU\_6.bdf 中，是原理图文件，参考实验 6，下载 SOF 文件 CPU\_6.sof。按照实验 6 中“(三) 执行程序”给出的描述方法，依据图 7-1、图 7-3、图 7-4 和实验程序 2 详细描述实验程序 2 的微程序执行过程，并写入实验报告中。
9. 按照实验 5 的要求和方法，首先利用在系统 EAB 读写工具实时了解此 CPU 内的 RAM 和 ROM 中的数据，然后用嵌入式逻辑分析仪和实验箱上的液晶同时观察 CPU 的运行过程，给出报告。
10. 自行编制一个新的程序，根据此程序写出 MIF 微指令的 ROM 配置文件并更新 rom\_11.mif 文件，画出微程序流程图，更新 LPM\_RAM\_DQ 中的程序代码文件 RAM\_6.mif，最后详细给出执行过程。

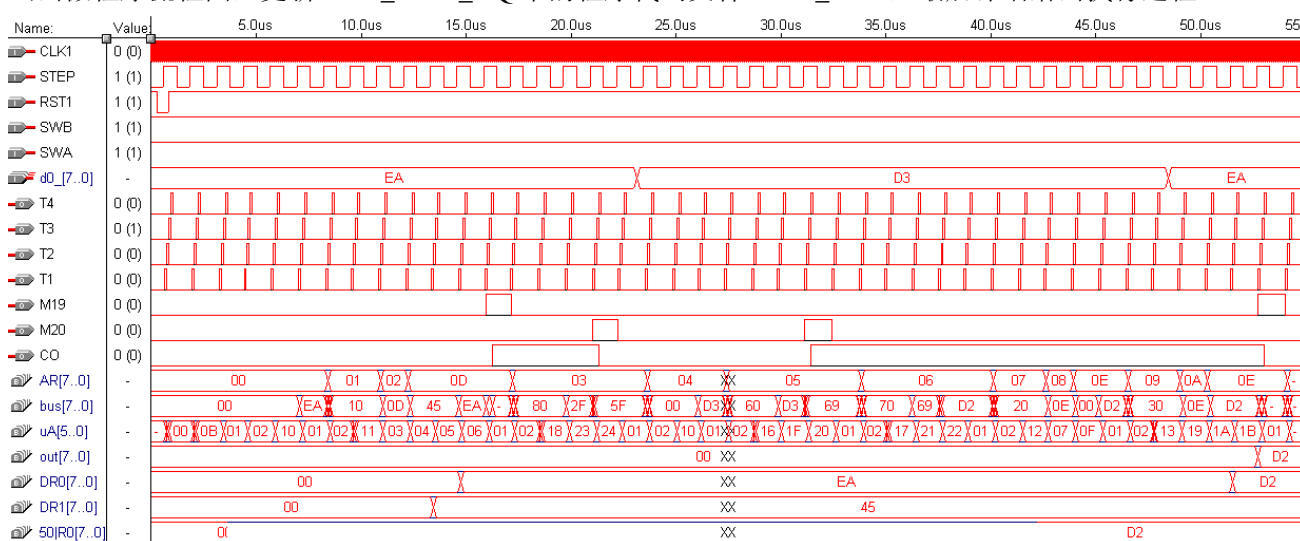


图 7-5 模型 CPU 执行“实验程序 2”的仿真波形图

## 五. 实验报告

1. 实验的原理，实验步骤和具体实验结果；
2. 实验中遇到的主要问题和分析解决问题的思路；
3. 通过实验，自己的学习经验和切身体会，以及对教学实验的意见和建议。

## 六. 思考题和实验题

1. 参考直接转移指令“JMP [直接地址]”，编写出实现其他寻址方式的控制转移指令。

- (1) 立即寻址：转移地址为用立即数表示的绝对地址。“JMP 绝对地址”
- (2) 相对寻址：转移地址为在当前 PC 基础上加上相对转移偏移量“JMP 偏移地址”
- (3) 条件转移：根据运算标志 FC、FZ 的状态进行转移：

JC 相对地址

JZ 相对地址

请设计相应微程序流程图，确定微程序代码，在实验台上验证所设计的功能。

2. 设计一条比较指令“CMP rs, rd”，比较源操作数 rs 和目的操作数 rd 的大小，运算结果会影响标志位 FC 和 FZ，将标志位保存在标志寄存器中。请编写相应微程序流程图，确定相应的微程序代码。

3. 如果执行程序的机器指令存放在外部存储器 EEPROM 中，模型机的相应电路应如何修改？如何通过控制台 SWB 和 SWA 键设定为输入方式，输入执行程序的机器指令？

4. 加法指令采用双地址格式，一个操作数采用隐含寻址方式，存放在寄存器 R0 中，另一个操作数在存储器中：

- (1) 若另一个操作数采用寄存器间接寻址，设计出指令的格式，并画出其微程序流程图。
- (2) 若另一个操作数采用存储器间接寻址，设计出指令的格式，并画出其微程序流程图。
- (3) 若另一个操作数采用变址寻址，设计出指令的格式，并画出其微程序流程图。

(4) 若另一个操作数采用基址加变址寻址, 设计出指令的格式, 并画出其微程序流程图。

5. 根据图 7-5, 在实验箱上完成相应实验, 将实验数据与波形数据进行比较, 并根据图 7-5 和电路图 CPU\_6.bdf, 详细说明 CPU 对实验程序 2 的执行过程和图中所有信号在不同节拍下变化的原因和结果。

## 实验八 复杂模型机的设计与实现

### 一. 实验目的

实验示例参考: /CMPUT\_EXPMT/Experiments/Expmt8 / CPU7\_GW48CP+ ; 显示屏为 LCD128X64

实验示例参考: /CMPUT\_EXPMT/Experiments/Expmt8 / CPU7\_GW48CP++ ; 显示屏为 LCD240X128

1. 综合运用所学计算机原理知识, 设计并实现较为完整的计算机。
2. 设计指令系统。
3. 编写简单程序, 在所设计的复杂模型计算机上调试运行。

### 二. 实验原理

1. 数据格式: 模型机采用定点补码表示法表示数据, 字长为 8 位, 其格式如下:

7	6	5	4	3	2	1	0
符号	尾数						

其中第 7 位为符号位, 数值表示范围是:  $-1 \leq X < 1$ 。

2. 指令格式: 所设计的指令分为四大类共十六条, 其中包括算术逻辑指令、I/O 指令、访问、转移指令和停机指令。(1) 算术逻辑指令。

设计 9 条算术逻辑指令并用单字节表示, 采用寄存器直接寻址方式, 其格式如下:

7	6	5	4	3	2	1	0
OP-CODE				rs		Rd	

其中, OP-CODE 为操作码, rs 为源寄存器, rd 为目的寄存器, 并规定:

Rs 或 rd	选定的寄存器
00	R0
01	R1
10	R2

(2) 访问指令及转移指令: 访问指令有 2 条, 即存数 (STA)、取数 (LDA); 2 条转移指令, 即无条件转移 (JMP)、结果为零或有进位转移指令 (BZC), 指令格式为:

7	6	5	4	3	2	1	0
00		M		OP-CODE		Rd	
D							

其中, OP-CODE 为操作码, rd 为目的寄存器地址 (用于 LDA、STA 指令)。D 为位移量 (正负均可), M 为寻址模式, 其定义如下:

寻址模式 M	有效地址 E	说明
00	$E=D$	直接寻址
01	$E=(D)$	间接寻址
10	$E=(RI)+D$	RI 变址寻址
11	$E=(PC)+D$	相对寻址

在本模型机中规定变址寄存器 RI 为寄存器 R2。

(3) I/O 指令: 输入 (IN) 和输出 (OUT) 指令采用单字节指令, 其格式如下:

7	6	5	4	3	2	1	0
OP-CODE				addr		Rd	


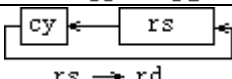
其中, addr=01 时选中 “INPUT DEVICE” 中的键盘输入设备, addr=10 时, 选中 “OUTPUT DEVICE” 中的 LCD 点阵液晶屏作为输出设备。

(4) 停机指令: 指令格式如下:

7	6	5	4	3	2	1	0
OP-CODE				00		00	

3. 指令系统: 共有 16 条基本指令, 其中算术逻辑指令 7 条, 访问内存指令和程序控制指令 4 条, 输入/输出指令 2 条, 其他指令 1 条。各条指令的格式、汇编符号、功能如表 8-1 所示。



助记符号	指令格式	功能
CLR rd	0111 00 rd	$0 \rightarrow rd$
MOV rs, rd	1000 rs rd	$rs \rightarrow rd$
ADC rs, rd	1001 rs rd	$rs + rd + cy \rightarrow rd$
SBC rs, rd	1010 rs rd	$rs - rd - cy \rightarrow rd$
INC rd	1011 rd	$rd + 1 \rightarrow rd$
AND rs, rd	1100 rd	$rs \wedge rd \rightarrow rd$
COM rd	1101 rd	$\overline{rd} \rightarrow rd$
RRC rs, rd	1110 rd	
RLC rs, rd	1111 rd	
LDA M, D, rd	00 M 00 rd D	$E \rightarrow rs$
STA M, D, rd	00 M 01 rd D	$rd \rightarrow E$
JMP M, D	00 M 10 rd D	$E \rightarrow PC$
BZC M, D	00 M 11 rd D	当 CY=1 或 Z=1 时, $E \rightarrow PC$
IN addr, rd	0100 01 rd	$addr \rightarrow rd$
OUT addr, rd	0101 10 rd	$rd \rightarrow addr$
HALT	0110 00 00	停机

本模型机的数据通路框图如图 7-1。根据机器指令系统要求，设计微程序流程图及确定微地址，如图 8-3。

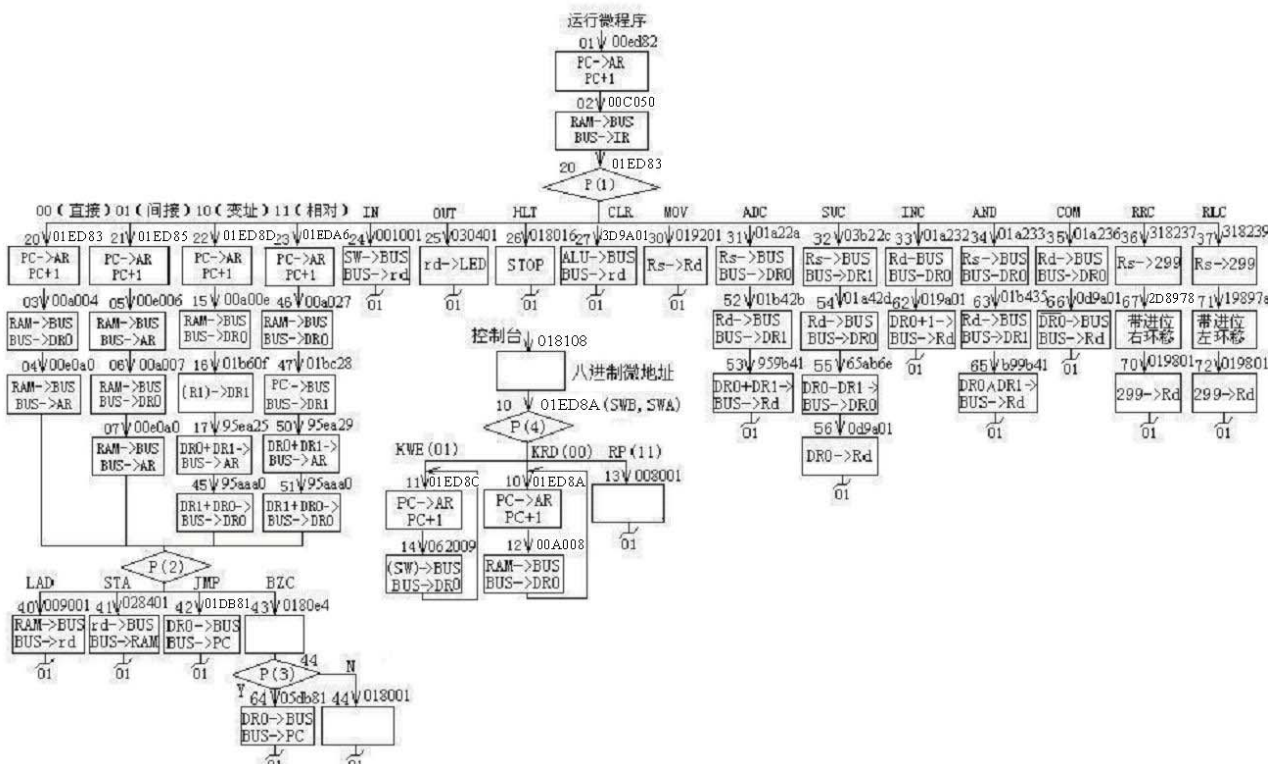


图 8-3 微程序流程图

三. 实验内容

按照系统建议的微指令格式，参照微指令流程图，将每条微指令代码化，译成二进制代码表，并将二进制代码表转换为联机操作时的十六进制格式文件。

微代码定义如表 7-1 所示。

24	23	22	21	20	19	18	17	16	15 14 13	12 11 10	9 8 7	6	5	4	3	2	1
S3	S2	S1	S0	M	Cn	WE	A9	A8	A	B	C	uA5	uA4	uA3	uA2	uA1	uA0

A 字段				B 字段				C 字段			
15	14	13	选择	12	11	10	选择	9	8	7	选择
0	0	0		0	0	0		0	0	0	
0	0	1	LDRi	0	0	1	RS-B	0	0	1	P (1)
0	1	0	LDDR0	0	1	0	RD-B	0	1	0	P (2)
0	1	1	LDDR1	0	1	1	RI-B	0	1	1	P (3)
1	0	0	LDIR	1	0	0	299-B	1	0	0	P (4)
1	0	1	LOAD	1	0	1	ALU-B	1	0	1	AR
1	1	0	LDAR	1	1	0	PC-B	1	1	0	LDPC

实验程序 3:

程序	助记符	功能
\$P00 44	IN 01, R0	Input =>R0
\$P01 46	IN 01, R2	Input =>R2
\$P02 98	ADC R2, R0	(R0)+(R2) =>R0
\$P03 81	MOV R0, R1	(R0) =>R1
\$P04 F5	RLC R1, R1	R1 带进位左移 =>R1
\$P05 0C	BZC 00, 00	Cy、Z 为 1 时，循环
\$P06 00		

表 7-2A 微程序代码

微地址	微指令	S3	S2	S1	S0	M	CN	WE	A9	A8	A	B	C	UA5—UA0
0 0	0 1 8 1 0 8	0	0	0	0	0	0	0	1	1	0 0 0	0 0 0	1 0 0	0 0 1 0 0 0
0 1	0 1 E D 8 2	0	0	0	0	0	0	0	1	1	1 1 0	1 1 0	1 1 0	0 0 0 0 1 0
0 2	0 0 C 0 5 0	0	0	0	0	0	0	0	0	1	1 0 0	0 0 0	0 0 1	0 1 0 0 0 0
...略，详见本实验工程文件 .\demoe_cpu78\b100_7.bdf 中元件 rom_7 中的文件 rom_ex7.mif														
7 1	1 9 8 9 7 A	0	0	0	1	1	0	0	1	1	0 0 0	1 0 0	1 0 1	1 1 1 0 1 0
7 3	0 1 9 8 0 1	0	0	0	0	0	0	0	1	1	0 0 1	1 0 0	0 0 0	0 0 0 0 0 1
7 4	0 7 0 A 0 8	0	0	0	0	0	1	1	1	0	0 0 0	1 0 1	0 0 0	0 0 1 0 0 0
7 5	0 6 2 0 0 9	0	0	0	0	0	1	1	0	0	0 1 0	0 0 0	0 0 0	0 0 1 0 0 1

表 7-2B 微程序代码（详见本实验工程文件 .\demoe\_cpu78\b100\_7.bdf 中元件 rom\_7 中的文件 rom\_ex7.mif）

微地址	代码	微地址	代码	微地址	代码	微地址	代码
\$M00	018108	\$M10	01ed83	\$M20	009001	...略	
\$M01	01ed82	\$M11	01ed85	\$M21	028401	\$M3A 019801	
\$M02	00c050	...略		\$M22	01db81	\$M3B 070a08	
...略		\$M1D	01a236	...略		\$M3C 062009	
\$M0E	01b60f	\$M1E	318237	\$M2E	0d9a01	\$M3D 000000	
\$M0F	95ea25	\$M1F	318239	\$M2F	01aa30	\$M3E 000000	

四. 实验步骤

1、打开 GW48 系统的电源；编辑微指令的配置文件 ROM\_EX7.mif（内容参见表 8-2 所示），并与模型机的工程文件 CPU\_7.bdf 一起重新编译，生成下载文件 CPU\_7.SOF。

2、对照执行程序的机器指令，将实验程序 3 中的代码正确地写入到 LPM\_RAM 的中初始化配置文件 RAM\_7.mif 中（在 CPU\_7.bdf 中元件 LPM\_RAM\_DQ 中）。通过对 LPM-RAM 配置文件的编辑、输入，并与实验电路一同编译后，得到新的下载文件 CPU\_7.sof，下载此文件到 FPGA 中；

4、用实验台的模式选择键选择模式“0”，按一次右侧的复位键；

5、in[7..0]——键 2 和键 1,数据输入 8 位（此值显示于键对应的数码管上）；SWB、SWA — 键 4、键

3 进行工作模式选择：00—KRD（读出）模式；01—KWR（写入）模式；11—RP（程序执行）模式。

6、选择 SWB、SWA=01（写入）模式。在微程序的控制下，通过实验台（键 2 和键 1）输入程序的机器指令代码，按单步执行键（键 7），将机器指令代码写入程序存储器，直到输入全部指令代码。

8、单步执行程序，通过 LCD 液晶显示屏观察“复杂模型机 CPU”中各基本工作单元内容。根据自己所设计的实验程序，按照微程序流程图，跟踪微指令和指令的执行情况。

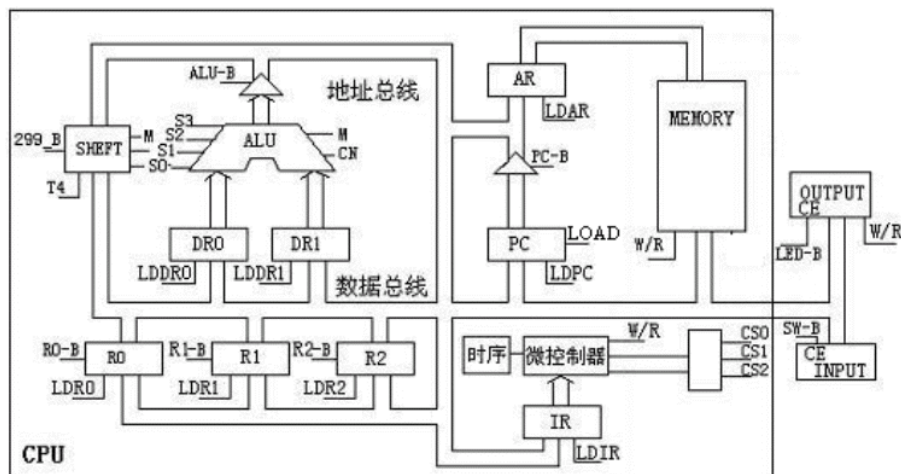


图 8-1 复杂模型机数据通路框图

## 五. 实验方法:

2、手动读出程序代码。将 SWA=0、SWB=0 和 RST=1，系统处于读出 RAM 数据方式，即键 3=0、键 4=0、键 8=1；通过键 7 读出脉冲，在微程序控制下，根据地址寄存器 AR 的变化，通过 LCD 观察读出 RAM 中程序的机器指令代码数据，进行比较。

3、执行程序（由于有预先加载于 RAM 中的程序代码文件 RAM\_7.mif，可直接运行程序）

(1) 按“系统复位”键使实验台复位，程序计数器清零  $PC=00$ ，微地址寄存器清 0。

(2) 将 SWB、SWA=11 和 RST 置高电平, 系统处于执行程序方式, 即键 3=1、键 4=1、键 8=1:

(3) 键 7 产生脉冲, 单步执行微指令。若要从“INPUT DEVICE”输入数据, 可通过实验台键 2、键 1 输入相应数据。根据微程序流程图, 跟踪每一条微指令的执行情况, 观察 LCD 显示的模型机中各单元的内容、分析程序执行情况、验证程序的功能。

(4) 参考实验五，下载 SOF 文件 CPU\_7.sof。按照实验五中“(三) 执行程序”给出的描述方法，依据图 8-1、图 8-3 和实验程序 3，详细描述实验程序 3 的微程序执行过程，并写入实验报告中。

分别输入加操作后无进位和有进位两对数分别进行实验，设 1) IN=82, IN=A9; 2) IN=45, IN=3C。通过数码 3 观察进位情况（注意，此程序要通过 IN 分别键入不同的数据！）

4. 利用本章的 16 条指令，自行编制一个新的程序，根据此程序写出 MIF 微指令的 ROM 配置文件并更新 ROM\_EX7.mif 文件，画出微程序流程图，更新 LPM\_RAM\_DQ 中的程序代码文件 RAM\_7.mif，最后详细给出执行过程。

## 六. 实验报告

七. 设计实验题目

- 1. 输入任意几个整数，求其和并存储、输出显示。
- 2. 求 1 到任意一个整数之间的所有奇数之和并输出显示。求 1 到任意一个整数之间的所有偶数之和并输出显示。
- 3. 求 1 到任意一个整数之间的所有能被 3 整除的数之和并输出显示。
- 4. A、让实验台上的 8 个发光二极管 D1~D8 从左向右依次轮流循环显示。  
B、让实验台上的 8 个发光二极管中的两个从右向左依次轮流循环显示。
- 5. A、让输出设备 OUTPUT 显示数据加 1 计数。 B、让输出设备 OUTPUT 显示数据减 1 计数。
- 6. 对存储器 RAM 中 40H~4FH 单元的数据求和，结果存放到 50H~51H 中，计算其平均值存放到 52H 单元并输出显示。
- 7. 存储器中存放在从 40H 和 50H 开始的两个多字节数相加，将结果存放在 60H 开始的存储单元中。
- 8. 若要增加微程序可用的存储空间，对现有的模型机结构需作哪些改动？哪些控制部件需要改、怎样改？
- 9. 对照图 7-5，给出本实验中的 CPU 执行“实验程序 3”的仿真波形图。

实验九. 较复杂 CPU 设计示例

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt9 / CPU8\_GW48CP+；显示屏为 LCD128X64  
实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt9 / CPU8\_GW48CP++ ;显示屏为 LCD240X128  
以实验八设计题目的第 2 题为例说明设计过程。求 1 到任意一个整数之间的所有奇数之和并输出显示。

1. 编写满足题目要求的汇编语言源程序

汇编语言源程序：		功 能
LP0:	IN R0	从开关输入任意一个整数 n→R0
	MOV R1, 1	将立即数 1→R1 (R1 存放参与运算的奇数)
	MOV R2, 0	将立即数 0→R2 (R2 存放累加和)
LP1:	CMP R0, R1	将 R0 中的整数 n 与 R1 中的奇数进行比较
	JB LP2	若 R1<R0, 则转到 LP2 处执行
	ADD R1, R2	否则, 累加求和
	INC R1	R1 的内容加 2, 形成下一个奇数
	INC R1	
	JMP LP1	跳转到 LP1 继续执行
LP2:	OUT R2	输出累加和
	JMP LP0	重新开始

2. 确定指令格式。 为了完成求和功能，需要使用 8 条指令，其中包括算术指令、I/O 指令、访问、转移指令和加 1 指令。

(1) I/O 指令。 输入（IN）和输出（OUT）指令采用单字节指令，其格式如下：

7	6	5	4	3	2	1	0
操作码				Addr		目的寄存器	

其中，addr=01 时选中“INPUT DEVICE”中的键盘输入设备，addr=10 时，选中“OUTPUT DEVICE”中的 LCD 点阵液晶屏作为输出设备。

(2) 比较和相加指令。 比较指令（CMP）和相加指令（ADD）用单字节表示，采用寄存器直接寻址方式，其格式如下：

7	6	5	4	3	2	1	0
操作码				Rs		rd	

其中，OP-CODE 为操作码，rs 为源寄存器，rd 为目的寄存器，并规定：

Rs 或 rd	选 定 的 寄 存 器
00	R0
01	R1
10	R2

(3) 转移指令。 无条件转移（JMP）和结果为零或有进位转移指令（JB），指令格式为：

7	6	5	4	3	2	1	0
操作码				X	X	X	X
地				址			

(4) MOV 指令。 指令格式如下：

7	6	5	4	3	2	1	0
操作码				X	X	R <sub>d</sub>	
立即数							

(5) 加 1 指令 INC。指令格式如下:

7	6	5	4	3	2	1	0
操作码				X	X	R <sub>d</sub>	

(5) 数据格式:

7	6	5	4	3	2	1	0
符号位	尾数						

3. 指令系统。本模型机有 8 条基本指令。每条指令的格式、汇编符号、功能如表 9-1 所示。

表 9-1 指令系统

助记符号	指令格式	功能												
IN rd	<table><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>X</td><td>X</td><td colspan="2">rd</td></tr></table>	1	0	0	0	X	X	rd		input → rd 寄存器				
1	0	0	0											
X	X	rd												
OUT rd	<table><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>X</td><td>X</td><td colspan="2">rd</td></tr></table>	1	1	1	1	X	X	rd		rd → output				
1	1	1	1											
X	X	rd												
ADD rs, rd	<table><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td colspan="2">rs</td><td colspan="2">rd</td></tr></table>	1	1	0	0	rs		rd		rs + rd → rd				
1	1	0	0											
rs		rd												
CMP rs, rd	<table><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td colspan="2">rs</td><td colspan="2">rd</td></tr></table>	1	0	1	0	rs		rd		rs - rd → rd				
1	0	1	0											
rs		rd												
INC rd	<table><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>X</td><td>X</td><td colspan="2">rd</td></tr></table>	1	1	0	1	X	X	rd		Rd +1→ rd				
1	1	0	1											
X	X	rd												
MOV data, rd	<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>X</td><td>X</td><td colspan="2">rd</td></tr><tr><td colspan="4">data</td></tr></table>	1	0	0	1	X	X	rd		data				data rd
1	0	0	1											
X	X	rd												
data														
JMP addr	<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr><tr><td colspan="4">addr</td></tr></table>	1	1	1	0	X	X	X	X	addr				Addr →PC
1	1	1	0											
X	X	X	X											
addr														
JB addr	<table><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr><tr><td colspan="4">addr</td></tr></table>	1	0	1	1	X	X	X	X	addr				若小于, 则 addr → PC
1	0	1	1											
X	X	X	X											
addr														

4.按照指令格式将汇编语言源程序汇编成机器代码。与汇编语言源程序对应的机器语言源程序:

	助记符	地址 (十六进制)	机器代码 (十六进制)	功能
LP0:	IN R0	00	80	Input → R0
	MOV R1, 1	01	91	1→R1
		02	01	
	MOV R2, 0	03	92	0→R2
		04	00	
LP1:	CMP R0, R1	05	A1	R0-R1→R1
	JB L2	06	B0	(LP2) →PC
		07	0D	
	ADD R1, R2	08	C6	R1+R2→R2
	INC R1	09	D1	R1+1→R1
	INC R1	0A	D1	R1+1→R1
	JMP L1	0B	E0	(LP1)→PC
		0C	05	
LP2:	OUT R2	0D	F2	R2→output
	JMP LP0	0E	E0	(LP0)→PC
		0F	00	

5. 确定微地址和微指令

表 9-2 微地址和微指令表

微地址	微指令	S3	S2	S1	S0	M	CN	WE	A9	A8	A	B	C	UA5—UA0
0 0	0 1 8 1 1 0	0	0	0	0	0	0	0	1	1	0 0 0	0 0 0	1 0 0	0 1 0 0 0 0
0 1	0 1 E D 8 2	0	0	0	0	0	0	0	0	1	1 1 0	1 1 0	1 1 0	0 0 0 0 1 0
0 2	0 0 C 0 4 8	0	0	0	0	0	0	0	0	1	1 0 0	0 0 0	0 0 1	0 0 1 0 0 0
...略, 详见本实验工程文件 .\demoe_cpu78\b100_8.bdf 中元件 rom_a 中的文件 rom_ex8.mif														
2 7	0 0 D 1 8 1	0	0	0	0	0	0	0	0	1	1 0 1	0 0 0	1 1 0	0 0 0 0 0 1
3 0	0 0 D 1 8 1	0	0	0	0	0	0	0	0	1	1 0 1	0 0 0	1 1 0	0 0 0 0 0 1
3 1	9 1 9 A 0 1	0	0	0	0	0	0	0	1	1	0 0 0	0 0 0	0 0 0	0 0 0 0 0 1
3 2	9 1 9 B 4 1	1	0	0	1	0	0	0	1	1	0 0 1	1 0 1	1 0 1	0 0 0 0 0 1

6. 根据微指令表编写微指令 LPM\_rom 的初始化文件(示例文件在 CPU\_8.bdf 中元件 rom\_a 中的文件 rom\_ex8.mif)

7. 设计微程序流程图

8. 在 MUX-PLUS II 环境下设计模型机电路的工程文件(示例文件是.\demoe\_cpu8\CPU\_8.bdf)。将模型机电路图文件与微指令 LPM\_rom 的初始化文件一起重新编译，并下载到实验台目标系统中(程序示例文件在.\demoe\_cpu8\CPU\_8.bdf 中元件 LPM\_RAM\_DQ 中的 ram\_8.mif)。

9. 在微程序的控制下通过实验台键盘输入程序的机器指令代码。

10. 单步运行调试程序。记录程序执行过程中的实验数据、通过 LCD 显示屏跟踪程序的执行情况，观察、分析所设计的微指令是否正确，发现问题及时调整、修改，重新进行编译、下载和单步调试。

11. 连续运行。若单步调试正确，可在键盘 (input) 输入程序所需数据后将实验台从单步切换到连续运行方式。

12. 实验报告：(1) 设计过程。指令系统、微程序流程图、汇编语言源程序和对应的机器语言源程序、模型机原理图，工作原理。(2) 实验数据。实验数据须经过实验教师签字验收。调试过程、问题排查、数据处理、结论。(3) 对设计过程的分析和总结

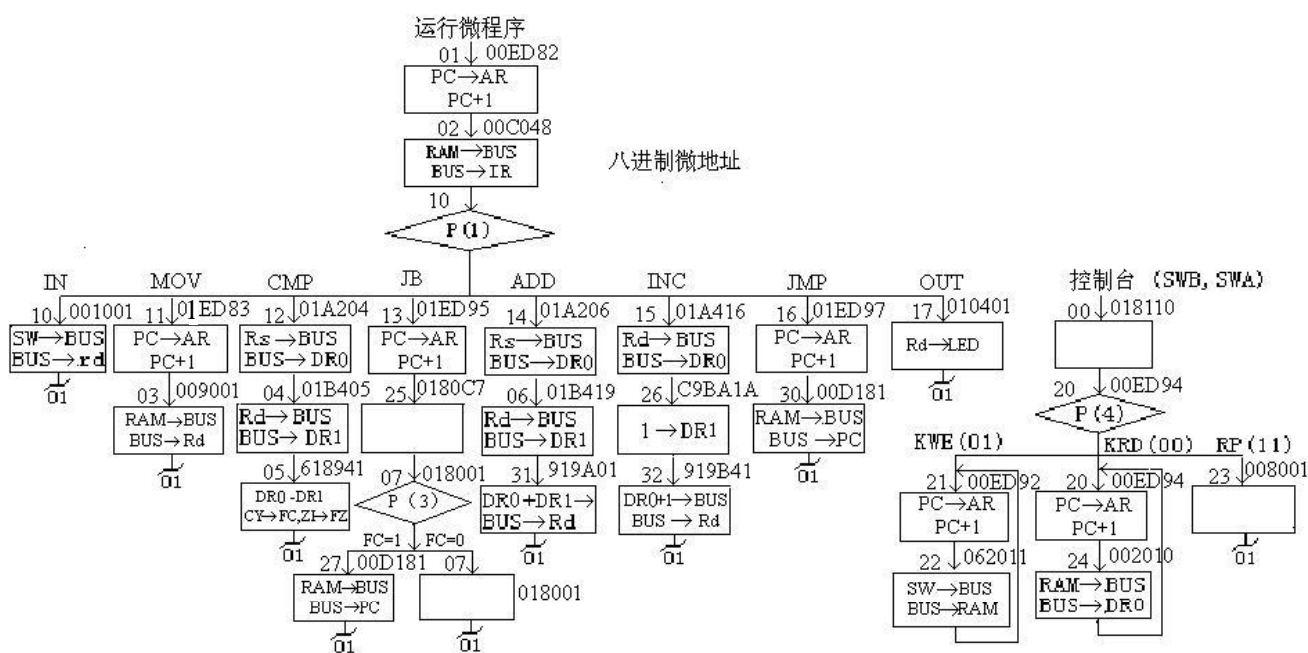


图 9-1 微程序流程图

## 实验十. 16 位精简指令 CPU 设计实验

实验示例参考：/CMPTUT\_EXPMT/Experiments/Expmt16 / CPU16bit\_GW48CP+；显示屏为 LCD128X64

实验示例参考：/CMPTUT\_EXPMT/Experiments/Expmt16 / CPU16bit\_GW48CP++；显示屏为 LCD240X128

本实验介绍一个用 VHDL 设计的 16 位精简指令微处理器设计实验。与此 CPU 相比，实验六至实验九中的 CPU 属于复杂指令 CPU，特点是占用的逻辑资源比较少，运行速度较慢，结构简单，特别是指令的增减比较容易，且不影响整体逻辑资源的占用。本实验中的 CPU 属于精简指令 CPU，指令运行周期短，占用时钟节拍少，运行速度快，但占用资源多，但对于目前高度发展的大规模集成电路技术，拥有足够的硬件资源来实现不同结构和不同用途的精简指令 CPU 并非难事。

### 一. CPU 结构

CPU 的结构图见图 10-1。

处理器中包含了大量基本器件单元、8 个 16 位的寄存器、一个 ALU 运算器、一个移位寄存器、一个程序计数器、一个指令寄存器、一个比较器、一个地址寄存器和一个控制单元。这些都共用一组 16 位的三态数据总线。

系统采用自顶向下的方法进行设计，顶层设计由微处理器和存储器通过一个双向数据总线连接，一根地址总线和一些控制线组成。处理器从外存储器中读取指令，并执行这些指令去运行一个程序，程序指令储存在指令寄存器中并由控制单元译码。这些控制单元使得相应的信号相互作用并使处理单元执行这些指

令。如果指令是两个寄存器相加，控制单元会将第一个寄存器的值写到 **OpReG** 作为暂时的存储量，第二个寄存器的值将被传到数据线，运算是相加模式，结果将被存在输出寄存器。输出寄存器最终将运算结果值复制到目标文件。

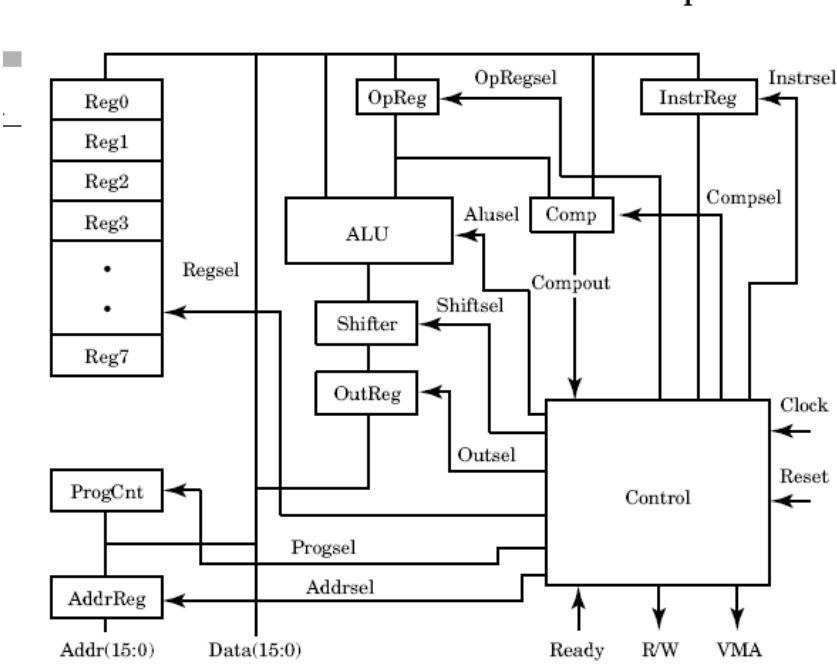


图 10-1 16 位 CPU 结构框图

当执行一条指令时，CPU 要分多个步骤进行。首先保存当前指令的地址，当一条指令执行完后，程序存储器跳到下一条指令的地址，如果是执行顺序指令，则是下一条指令；要是分支，则直接跳到下一条指令的地址，控制单元复制程序寄存器。地址寄存器在地址线上输出新的地址，同时，控制单元将 **R/W** 置 0，执行读操作，将 **VMA** 置 1，告诉存储器地址是有效的。

存储器将地址译码并将数据传给数据线，当数据在数据线上时，存储器将 **READY** 信号置 1，表明存储器的数据可以用了。控制单元将存储器中数据写到指令寄存器控制单元，接着将访问指令和译码指令，执行译码指令，进程循环进行。

二. 指令结构

在设计处理器时首先要确定 CPU 具有哪些功能、采用哪些指令，确定指令的格式。为了使所设计的 CPU 具有基本的运算功能，指令可按照以下形式设计。

指令分为如下几类：

- 1、装载指令：将立即数、其他寄存器的数据，或存储器中的数据装入某一寄存器中；
- 2、存储指令：将寄存器的值送到存储器
- 3、分支指令：分支指令使处理器转到其它地址，其中一些分支指令为条件转移，另外一些为无条件转移。
- 4、移位指令：这些指令用移位寄存器执行移位操作，实现数据传递。

指令格式。指令有一些共有属性，但也有各自的特点。本 CPU 所有的指令都包含 5 位操作码，单字节指令在低 6 位指令中包含两个 3 位寄存器。一些指令，比如 **INC**（累加）只用到其中一部分，但是另外一些指令，如 **MOVE**（转移），即将一个寄存器中的数据转移到另外一个寄存器中。双字节指令中第一个字节中包含目标寄存器的地址。第二个字节中包含了指令地址或者操作数。常用指令格式如下。

单字节指令：

操作码					源操作数			目的操作数		
Opcode					SRC			DST		
15	14	13	12	11	5	4	3	2	1	0



双字节指令：第一个字节中包含和目标寄存器的地址。第二个字节中包含了指令地址或者操作数。

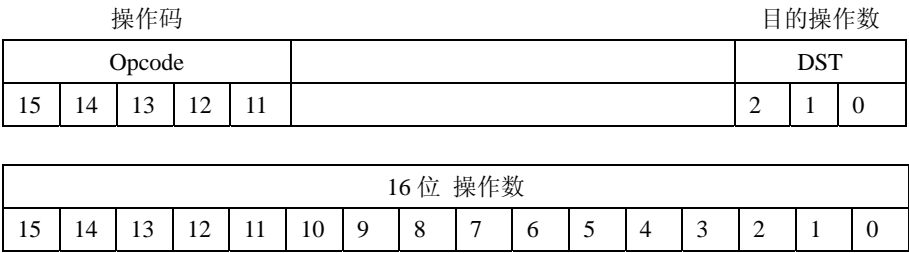


图 10-2 指令格式

例如 **LOADI**（立即装载）如下：**LOADI 1, 16#15**

这条指令装载 16 进制数 15 到寄存器 1，指令码如图 10-3 所示。

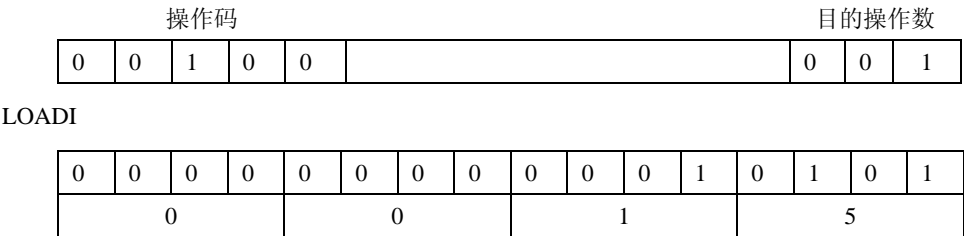


图 10-3 装载指令 **LOADI** 格式

控制单元译码时第一个字节的操作数，决定了字节两个字节的长度，装载第二个字节，使指令完整。

这两个字用 16 进制表示为：2001H，0015H。在处理器中设计了下列操作码，列于图 10-4 中。

操作码	指令	功 能
00000	NOP	NO operation
00001	LOAD	Load register
00010	STORE	Store register
00011	MOVE	Move value to register
00100	LOADI	Load register with immediate value
00101	BRANCHI	Branch to immediate address
00110	BRABCHGTI	Branch great than to immediate address
00111	INC	Increment
01000	DEC	Decrement
01001	AND	And two registers
01010	OR	Or two registers
01011	XOR	Xor two registers
01100	NOT	Not a registers
01101	ADD	Add two registers
01110	SUB	Subtract two registers
01111	ZERO	Zero a registers
10000	BRANCHLTI	Branch less than to immediate address
10001	BRANCHLT	Branch less than
10010	BRANCHNEQ	Branch not equal
10011	BRANCHEQI	Branch to immediate address
10100	BRANCHGT	Branch great than
10101	BRANCH	Branch all the time
10110	BRANCHEQ	Branch if equal

10111	BRANCHEQI	Branch to immediate address
11000	BRANCHLTEI	Branch to immediate address
11001	BRANCHLTE	Branch less or equal
11010	SHL	Shift left
11011	SHR	Shift right
11100	ROTR	Rotate right
11101	ROTL	Rotate left

图 10—4 操作码功能表

### 三. CPU 顶层设计

CPU 元件的 VHDL 描述。首先，顶层程序包 CPU-LIB-VHDL 用来说明连接各个元件之间的信号的类型。此程序包描述了多个用于规定运算器功能、移位寄存操作和用于 CPU 控制的状态的类型。为了验证微处理器的功能，在此设计了一个 CPU 与存储器接口的模型。

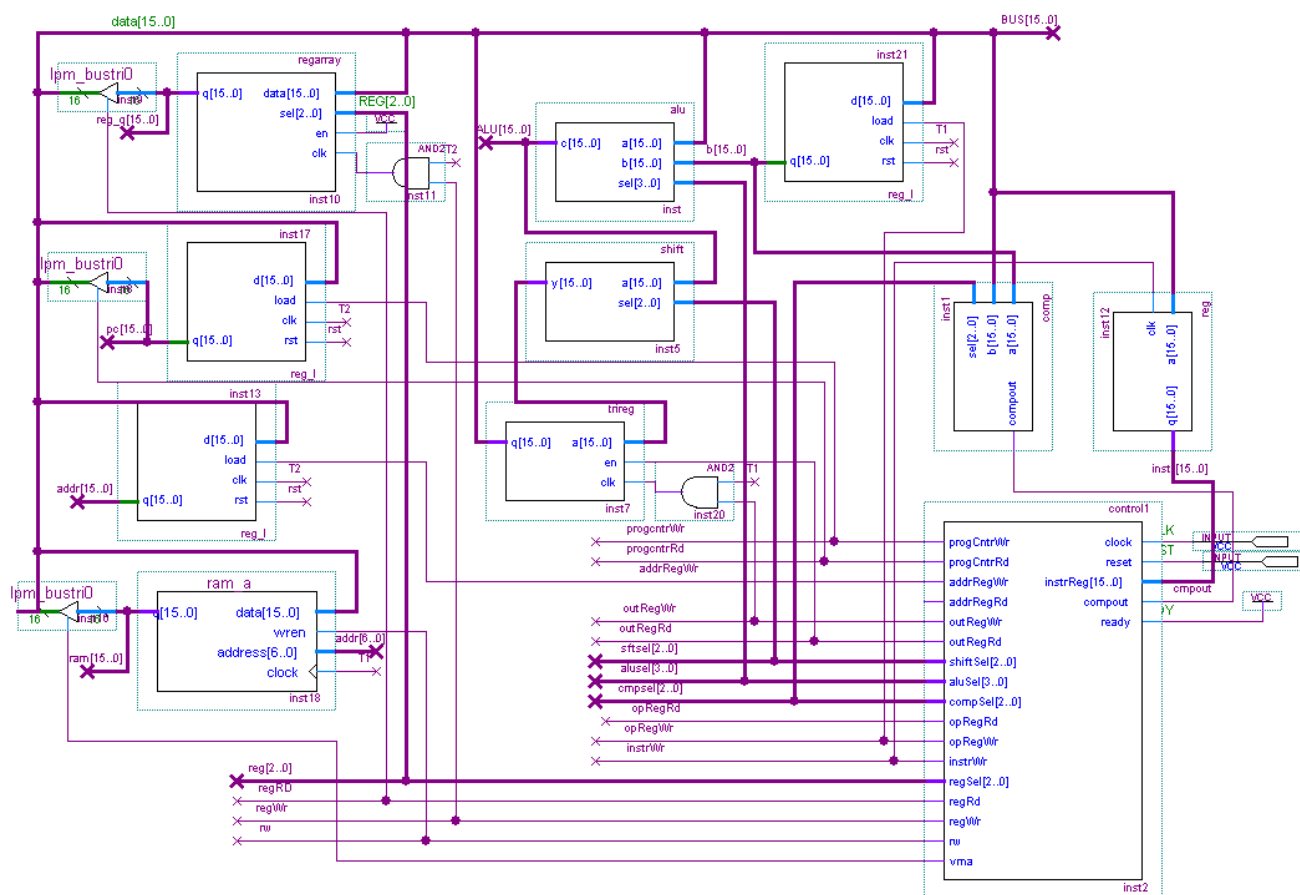


图 10—5 CPU 顶层结构图

最顶层的设计文件 toop\_16.bdf,用 Quartus 的图形文件描述,其结构如图 10—5 所示。元件 mem\_a 是一个存储元件,包含了 CPU 执行的指令和数据。由 FPGA 中的嵌入式阵列块 EAB 构成,有简单的总线接口,CPU 能对存储器进行读/写操作。

从存储器 MEM 中读取数据的过程如下:由信号 ADDR 选择一个相应的地址作为 READ 的存储单元,使 R/W 为 0,信号 SEL 置 1,信号 DATA 为存储单元的值,若信号 READY 为 1,说明存储器是可用的。写一个数据到存储器单元,首先将地址信号放在 ADDR 上,信号 R/W 和 SEL 置 1,数据从数据输入端写入存储器单元中。

存储器分成两个部分:第一部分为指令区,第二部分数据区。指令部分包含了将被执行的指令,这是一段数据块复制的应用程序;数据部分包含了在指令中将被执行的数据。CPU 指令从 00 开始到 0D 结束,数据区从 10 开始到 3F 结束,到存储器的末地址。存在存储器中的指令是简单的运算规则,将数据从一个

单元转移到另一个单元。

存储器 RAM 的初始化程序和数据保存在 RAM\_16.MIF 文件中。内容如表 10-1 所示。

当 CPU 得到一个 RESET 信号，复制操作开始。RESET 信号使得 CPU 内部的工作状态复位，从存储单元 00 开始处理指令。程序一开始用一些指令设置适当的寄存器，使得块复制操作能够进行。寄存器 1 包含了起始地址或者被复制存储器块第一个元件的地址。寄存器 2 包含了目标存储器块的地址。寄存器 6 包含了被复制存储器块的末地址。程序中，从指针 REG1 指向的单元取数，送往指针 REG2 指向的单元，然后将取数指针 REG1 与结束标志 REG6 的值进行比较，判断数据块复制是否结束。若不相等，则修改源指针和目的指针，继续循环传输复制数据块；若相等，则程序结束，停止运行。

表 10-1

WIDTH = 16;	
DEPTH = 256;	
ADDRESS_RADIX = HEX;	
DATA_RADIX = HEX;	
CONTENT BEGIN	
0 : 2001;--源操作数（10）送 REG1	
1 : 0010;--	
2 : 2002;--目的操作数（30）送 REG2	
3 : 0030;--	
4 : 2006;--结束地址（2f）送 REG6	
5 : 002f;--	
6 : 080b; --取数	
7 : 101a;-- 存数	
8 : 300e;--比较 R1>R6 ?	
9 : 0000;-- 若 yes,则停止	
a : 3801;--修改源指针 R1<=R1+1	
b : 3802;--修改目的指针 R2<=R2+1	
c : 280d;-- goto [06]单元地址，循环	
d : 0006;--	
e : 0000;	
	; -----10H—2fH 单元为数据块
	f : 0001;
	10 : 0002;
	11 : 0003;
	12 : 0004;
	13 : 0005;
	14 : 0006;
	15 : 0007;
	16 : 0008;
	17 : 0009;
	18 : 000a;
	19 : 000b;
	1a : 000c;
	1b : 000d;
	1c : 000e;
	1d : 000f;
	1e : 0010;
	1f : 0011;
	20 : 0000;
	END;

四. CPU 设计

CPU 顶层设计由下层元件例化而成。主要有以下一些基本部件：

1、运算器 ALU

算术逻辑单元 ALU，通过一条或多条输入总线完成算数运算或者逻辑运算。运算器 ALU 的结构如图 10-7 所示。

Sel Input	Operation
0000	C = A
0001	C = A AND B
0010	C = A OR B
0011	C = NOT A
0100	C = A XOR B
0101	C = A + B
0110	C = A - B
0111	C = A + 1
1000	C = A - 1
1001	C = 0

图 10-6 运算器的功能

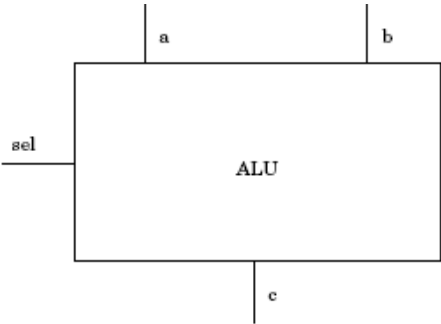


图 10-7 运算器 ALU 结构图

A 和 B 为运算器输入端口，C 为运算器输出端口输出运算结果。输入 SEL 决定了运算的算法，如图 10-7 所示。运算器可完成算术操作，比如加、减运算。还可以进行逻辑运算，如与、或、异或。

2、比较器

比较器的实体名为 COMP。实体 COMP 比较两个值，结果为 1 或 0，取决于比较对象的类型和值。比

较器示意图如 10-8。

比较器的类型决定于输入端口 SEL 的值。例如：比较输入端口 A 和 B 的值是否相等，将 EQ 传到端口 SEL，如果 A 和 B 的值相等，COMPOUT 的值为 1，如果不相等，则为 0。比较类型 T-COMP 定义在先前描述过的文件 CPULIB.VHD 的程序包 CPU-LIB 中。两个输入值的比较操作得到一个位结果，这个位是执行指令时用来控制进程中的操作流程。比较器有一个 CASE 说明语句，CASE 语句的分块由 IF 语句组成。如果条件为真，赋值 1，否则，赋值 0。

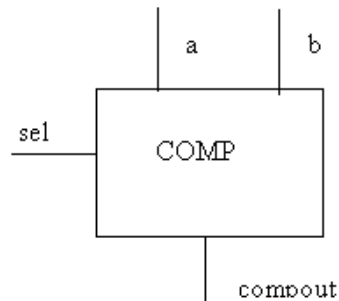


图 10-8 比较器结构图

### 3、控制器

实体 CONTROL 提供必要的信号连线，使得数据流完全地通过 CPU，达到预期的功能。结构体 RTL 包含一个状态机，状态机根据当前状态和输入信号值更新为合适的值，状态机进入下一个状态，控制块只有几个输入端，但有很多输出端口，控制块的所有控制信号都去控制。下面是 CPU 的 VHDL 描述。

结构体 RTL 有两个进程，第一个为组合进程，它根据当前状态和输入产生输出控制信号值和下一个状态输出。第二个为时序进程（有一个时钟信号），用来存储当前状态，当输入时钟上升沿到来时将次态内容送入现态。如果控制块是比较大的状态机，它的每一条指令都包含多个状态，执行指令中的所有状态，完成必要的步骤，使指令执行完。

### 4、寄存器 REG

实体 REG 用作地址寄存器和指令寄存器，这些寄存器在时钟上升沿到来时获得输入数据，使输出 Q 得到一个数据，当上升沿到来时，输入值 A 传给输出 Q。仿真时，REG 有三个端口，端口 A 是数据输入端口，Q 是数据输出端口 CLK 控制实体 REG 中数据的保存。当上升沿到来时，进程 REGPROC 被启动，输入的 A 传给输出 Q。

### 5、REGARRAY

实体 REGARRAY 用来在 CPU 中模拟寄存器组，寄存器存储指令处理的立即数在执行指令时，寄存器读或写操作。寄存器组被作为一个 RAM 这个 RAM 为 8 个 16 位。向 REGARRY 的一个单元地址写数据，输入 SEL 作为单元地址，CLK 上升沿到来时，输入数据被写入单元地址。从 REGARRY 的一个地址单元中读出数据，输入 SEL 作为读的单元地址，使输入 EN 为 1，数据在端口 Q 输出，寄存器阵列有两个独立进程。第一个进程模拟 RAM 存储数据，进程包含一个局部变量 RAMDATA，将存储的数据写到实体 REGRRAY，当 CLK 上升沿到来时，输入 SEL 选定的单元地址输入新的值，这个进程还将单元地址传给信号 TEMP-DATA，使数据传到下一个进程。第二个进程从 REGARRY 中读出数据，当输入 SEL 改变时，第一个进程改变了 TEMP-DATA 的值，信号 TEM-DATA 的值传给第二个进程，如果信号 EN 为 1，则第二个进程输出值为 TEMP-DATA。否则，输出 Z，Z 说明实体 REGARRY 禁止输出。

### 6、SHIFT

实体 SHIFT 在 CPU 中实现移位和循环操作，SHIFT 有一条 16 位输入总线和一条 16 位输出总线，输入信号 SEL 决定执行哪一种转移。移位的类型有：通过、左移还是右移、左循环还是右循环。方式 SHIFT PASS 使移位器输入数据直接传给输出，不执行任何移位操作。SHL 和 SHR 决定了移位器是左移还是右移，ROTL 和 ROTR 决定是左循环还是右循环。

### 7、TRIREG

三态寄存器元件 **TRIREG**。三态寄存器和主要的数据总线相连，可以存储数据，也可以将数据传到数据总线，实体 **TRIREG** 有 4 个端口。A 为寄存器数据输入端，Q 为寄存器数据输出端，输入信号 **CLK** 将新的值存到寄存器。当输入信号上升沿到来时，将数据存到寄存器中。当 **CLK** 上升沿到来时，端口 A 上的数据被储存在寄存器中。输入 **EN** 字用来控制输出 q。当 **EN** 值为 1 的时候，寄存器输出 q。当 **EN** 为 0 的时候，输出 q 为高阻，禁止输出。

五. 实验步骤

- 1. 按照汇编语言的格式，编写数据块复制的汇编语言程序。
- 2. 按照指令格式将汇编语言源程序汇编成机器代码。即汇编语言源程序对应的机器语言源程序，格式如下：

助记符	地址（十六进制）	机器代码（十六进制）	功能
LOADI R1,#0010	00	2001	源操作数地址 10→R1
	01	0010	
LOADI R2,#0030	02	2002	目的操作数地址 30→R2
	03	0030	
LOADI R6,#2F	04	2006	结束地址 2F→R6
	05	002F	
.....	06	.....	

- 3. 根据机器代码表编写程序 **LPM\_ram** 的初始化文件(示例文件在 **CPU\_16.bdf** 中元件 **mem\_16** 中的文件 **ram\_16.mif**)
- 4. 在 **QUARTUS II** 环境下设计 **CPU\_16** 的工程文件(示例文件是 **.\demoe\_cpu\CPU\_16.bdf**)。将 **CPU\_16** 的电路图文件与 **LPM\_ram** 的初始化文件 **ram\_16.mif** 一起重新编译。
- 5. 对编译后的文件进行仿真。建立仿真文件，根据仿真波形分析结果，找出错误并改正，重新编译。仿真波形如图 10-9 所示。

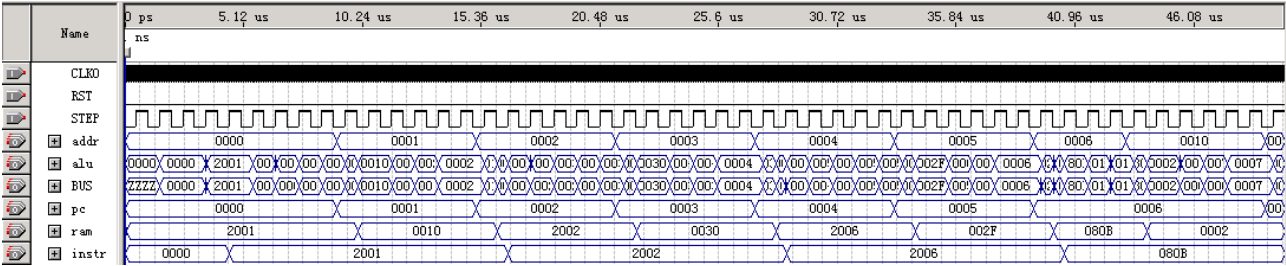


图 10-9 CPU\_16 仿真波形

仿真波形图中给出了部分信号的波形，**RST**—复位信号，高电平有效；**STEP**—单步运行节拍信号；**addr**—存储器地址；**alu**—算术逻辑单元；**BUS**—CPU 内部总线（16 位）；**pc**—程序计数器；**ram**—存储器；**instr**—指令寄存器输出的指令。

**CPU** 复位经历 **RST** 的 6 个 **STEP**。**RST1**：将 **aluSel**、**shiftSel** 设置为 0，使 **ALU** 和 **SHIFT** 处于直通状态；**RST2**：保持 **RST1** 的状态，同时将 **outRegWr** 置 1，允许向输出寄存器写数据；**RST3**：允许输出寄存器输出数据；**RST4**：将输出寄存器输出的数据写入 **PC** 程序计数器和 **AddrREG** 地址寄存器；**RST5**：**vma=1**、**rw=0**，向 **mem** 存储器发“读”命令；**RST6**：检测 **Ready** 状态，若 **Ready=1** 则准备进入运行(**execute**)状态。否则继续检测 **Ready** 标志。

运行(**execute**)状态。在 **execute** 状态每一条指令都需要经过：取指令——指令译码——指令执行，重复循环。

第 1 条指令是一条将立即数 **0010H** 送 **REG1**。从存储器 **mem** 中取出指令 **2001H**，存入指令寄存器 **instrREG**，**PC** 指针指向下一个地址单元，经过译码确定这是一条取数指令。然后，根据地址寄存器 **addr** 的值 **0001H**，从存储器的 **0001H** 单元取出立即数 **0010H**，送寄存器 **REG1**。接着修改 **PC** 指针和 **addrREG** 指针。同学们可以依此方法，通过仿真波形和 **CPU** 控制器的硬件描述语言 **CONTROL.VHD**，分析其它程序语句的运行情况，以及各个功能部件的工作时序和控制方法。

- 6. 将仿真通过、重新编译后的 **CPU\_16.sof** 文件下载到实验台目标系统中，实际运行调试程序。
- 实验台选择模式 0，**CLK0** 选择 750KHz，键 8 为复位键、高电平复位、低电平程序运行。按键 7 单步

运行程序（键 7 每按 2 次执行一步），通过实验系统上的 LCD 液晶屏幕，观察记录运行结果。LCD 显示屏结构如图 12-10 所示。

7. 按照实验 5 的要求和方法，首先利用在系统 EAB 读写工具实时了解此 CPU 内的 RAM 中的数据，然后用嵌入式逻辑分析仪和实验箱上的液晶同时观察 CPU 的运行过程，给出报告。

表 10-2 LCD 液晶显示屏功能说明

名称	作用	名称	作用
IN	输入单元 INPUT	OUT	输出单元 OUTPUT
ALU	算术逻辑单元	BUS	数据总线
DR	数据寄存器 R	REG	寄存器阵列
AR	地址寄存器	PC	程序计数器
RAM	程序/数据存储单元	IR	指令寄存器



图 10-10 LCD 液晶显示屏

8. 利用实验系统上的液晶屏上的数据显示和嵌入式逻辑分析仪同时了解 CPU 的每一单步运行情况(图 10-11)。注意，图 10-11 的嵌入式逻辑分析仪设置情况：用 CLK0 作为采样时钟，采用时钟使用 CPU 的工作时钟（CLK1=1.5MHz），采样深度 64 位，触发位置：Pre...，触发信号用单步控制信号：STEP；触发方式：上升沿。

通过逻辑分析仪观察以下信号：addr、ALU、alusel、compout、instr、PC、RAM、REG、STEP 等信号。设置波形时，如图 10-12 所示。

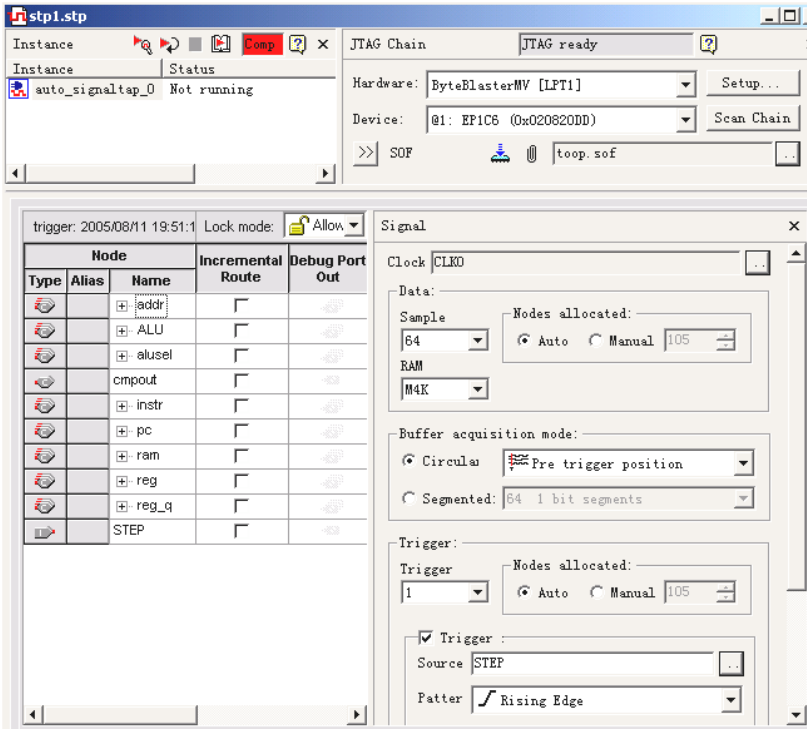


图 10-11 嵌入式逻辑分析仪设置情况

CPU 运行的逻辑分析仪波形数据如图 10-12 和图 10-13 所示。图 10-12 显示了用在系统读写工具读出的 FPGA 中 LPM\_RAM 的内容，即程序的机器代码和数据块。程序的机器指令和 RAM 中的数据可以现场进行修改，在图 10-14 中选择要观察的元件 RAM，通过点击快捷键就可以对 LPM\_RAM 数据进行上载/下载或跟踪观察。图 10-13 显示了当触发信号 STEP 上升沿到来时，嵌入式逻辑分析仪采集到 FPGA 中各观察信号的波形。通过重复按键 7 产生 STEP 信号，可以一步一步地观察 CPU 执行过程中各信号的变化过程。若需要观察其他信号，可以重新设置（添加或删除）信号，并重新编译、下载、启动逻辑分析仪进行观察。

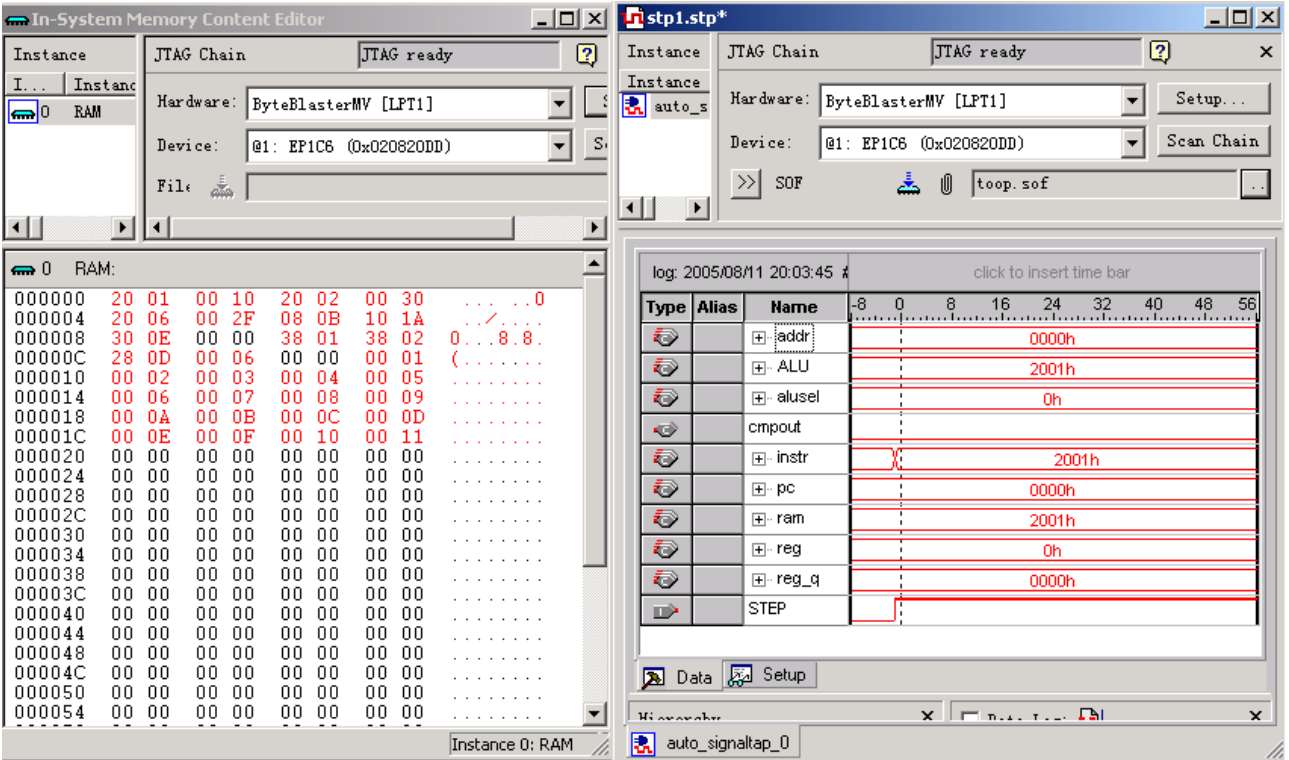


图 10-12 用在系统 EAB 读写工具对 LPM\_RAM 进行观察和改写

图 10-13 CPU\_16 运行时逻辑分析仪显示波形



图 10-14 用在系统 EAB 读写工具对 LPM\_RAM 操作

## 六、实验要求

1. 实验之前应认真准备，写出实验步骤和具体设计内容；
2. 实验前应掌握所有控制信号的作用；
3. 掌握在 QuartusII 环境下，用 VHDL 进行基本单元的设计方法；
4. 对所设计的基本模块进行波形仿真，进行功能验证；
5. 采用图形编辑方法的设计技术，进行顶层设计；
6. 掌握在微程序控制下机器指令的写入、读出、和程序执行方法；
7. 掌握 LPM\_RAM 的配置方法，实现对机器指令输入。
8. 通过 LCD 液晶屏，观察各相关寄存器、ALU、DR、PC、IR、AR、BUS、Instr 等内容的变化情况。
9. 通过 STEP（键 7）单步跟踪程序的执行情况，并详细记录每条微指令执行后，相关单元输出数据的变化情况，依次执行机器指令，从而验证所设计的正确性。
10. 在完成基本验证实验后，自行设计程序、输入和调试，记录实验数据。

## 七、实验报告

1. 实验的原理，实验步骤和具体实验结果；
2. 实验中遇到的主要问题和分析解决问题的思路；



3. 通过实验, 自己的学习经验和切身体会, 以及对教学实验的意见和建议。

## 八. 思考题与附加实验题

1、说明 LOAD 指令的工作过程, 分析该指令的工作时序。

2. 说明 STORE、MOVE、LOADI 等其他指令的工作过程, 分析该指令的工作时序。

3. 嵌入式逻辑分析仪的设置, 如何选择采样时钟和触发信号, 如何设置被观察信号?

4. 编写程序实现两个数据块对应数据字相加, 并存储在新的地址区间。

5、在上题中, 启动逻辑分析仪观察程序的执行情况, 用在系统 EAB 读写工具对 LPM\_RAM 中数据变化情况进行观察。

6、在顶层文件中采用了寄存器和锁存器元件, 这两种元件的功能有何不同?

7、在本 CPU 设计中, 若程序计数器 PC 不采用寄存器的结构, 而是用通用计数器结构来实现, 应如何修改电路结构? 相关的 VHDL 文件应如何修改?

8、添加新的指令应如何实现? 试为本 CPU 增加 1~2 条新的指令, 并通过实际验证。

9、为 CPU 增加 IN 和 OUT 的功能, 通过实验台的键盘和数码管, 输入数据和显示输出。

## 实验十一 32 位 Nios CPU 嵌入式系统软硬件设计实验

### 1 Nios 软硬件开发流程

设计流程的第一步, 是设计规划。需要根据系统设计要求, 划分好各个软硬件模块。完整的基于 Nios 的 SOPC 系统是一个软硬件复合的系统, 在开发时可以分为硬件、软件两个部分。对于通常的嵌入式系统开发, 往往 CPU 是不可更改的, 因而外围设备的变动也受到 CPU 的限制, 因而通常的嵌入式开发更多的是 PCB 设计及软件开发。但 Nios 是一个可灵活定制的 CPU, 它的外设是可选的 IP 核或自定义逻辑, 可以根据系统设计要求, 通过 SOPC Builder 向导式的界面定制裁剪得当的 SOPC 系统。Nios 的开发流程分为两个大部分: 硬件开发与软件开发。

Nios 的硬件设计流程就是为了定制合适的 CPU 和外设, 在 SOPC Builder 和 QuartusII 中完成。在这里可以灵活定制 Nios CPU 的各个特性甚至指令, 可以使用 Altera 提供的大量的 IP Core 来加快开发者开发 Nios 外设的速度, 提高外设的性能, 也可以使用第三方的 IP Core, 或者使用 VHDL、Verilog 来自己定制外设。完成 Nios 的硬件开发后, SOPC Builder 可以帮助开发者生成相应的 SDK (软件开发包)。这是由于在硬件开发中的 Nios CPU 及其外设构成的系统是自定义的, 存储器、外设地址的映射等都各不相同, 需要的 SDK 也应是专有的, 不过不用开发者操心, SOPC Builder 自动生成 SDK。

NIOS 嵌入式系统是自己定制的、裁剪过的, 可能受到硬件的局限会小一些。开发者可以使用汇编或 C 语言, 甚至 C++, 来进行嵌入式程序设计。当软硬件开发都完成了, 接下来就需要在 Nios 开发板上实现一个要求设计的系统的原型。在这个软硬件原型上, 运行整个系统, 测试是否达到要求。

### 2 Nios 软硬件开发流程

在这一节, 将以一个简单的基于 Nios 的 SOPC 系统的开发过程来详细讲述 Nios 硬件开发的具体流程。

#### 1、新建 SOPC 设计项目

首先, 需要在 QuartusII 中建立一个设计项目 (Project)。请参考第 3 章, 使用 “New Project Wizard 新建一个工程。

注意, 由于一开始, 文件价中并没有设计文件, 所以先建立一个空的工程。首先建立一个文件夹, 例如:d:\myprj。设定一个空文件夹为项目的工作目录, 如命名项目名称为:nios\_dvp。选择器件系列为 Cyclone, 器件为 EP1C6Q240C8。需要指出的是, 在 QuartusII 中进行 SOPC 设计, 必须在有项目被打开时, 才能进行, 否则 SOPC 的设计工具 SOPC Builder 是不能开启的。

选择 QuartusII 菜单 “Tools” → “SOPC Builder...”, 打开与 QuartusII 集成的 SOPC 开发工具: SOPC Builder (图 5-2) 就是 SOPC Builder 的启动画面。当在一个新的项目中首次打开 SOPC Builder, 会弹出 “Creat New System(建立新系统)” 对话框 (见图 5-2), 在对话框中输入需要建立的 SOPC 系统的名称, 选择 SOPC Builder 生成的 HDL 代码的类型。在此设定 SOPC 系统名称为: nios32; 语言类型为 VHDL。按 “OK” 按钮后, 进入 SOPC Builder 的设计界面 (图 5-3)。

从图上看，整个 SOPC Builder 界面可以分成三个部分：左边是一个组件（Nios 嵌入式系统元件）选择栏，用树型结构列出了 SOPC Builder 的组件；右边空白处可以列出加入的组件；下方是提示栏，提示一些 SOPC Builder 的提示信息 and 警告错误信息。在右上方，可以选择器件系列和系统工作频率。在这个示例中选择器件系列是 Cyclone，和系统工作频率是 50MHz。

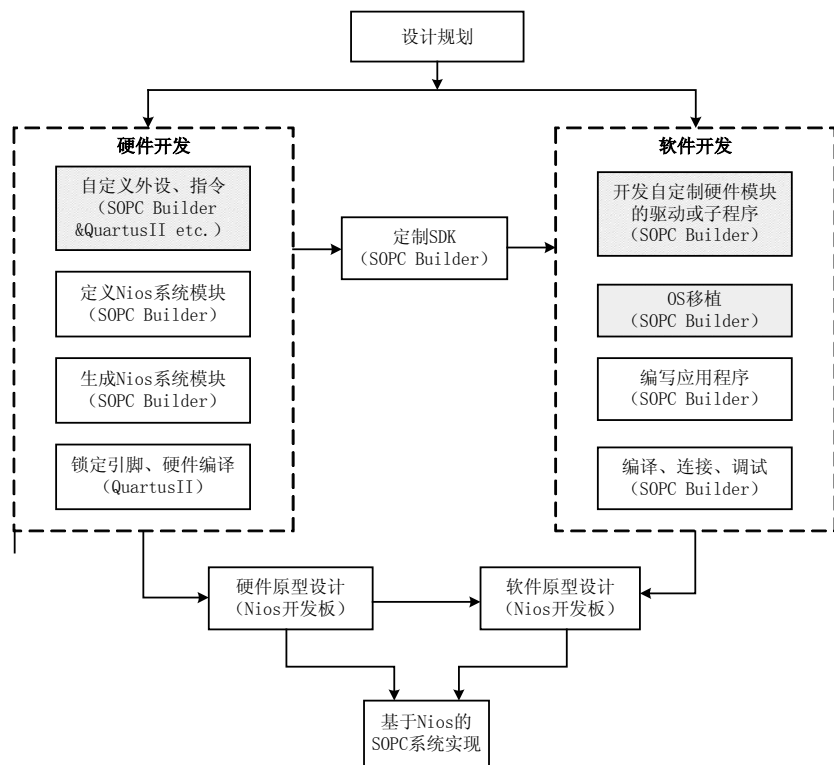


图 5-1 Nios 软硬件开发流程

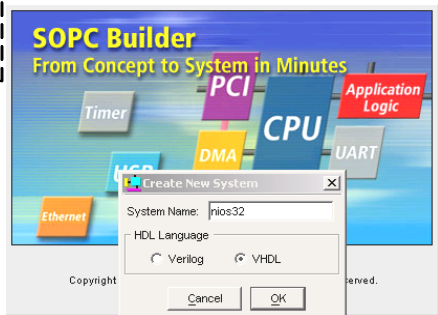


图 5-2 建立一个 SOPC 系统模块

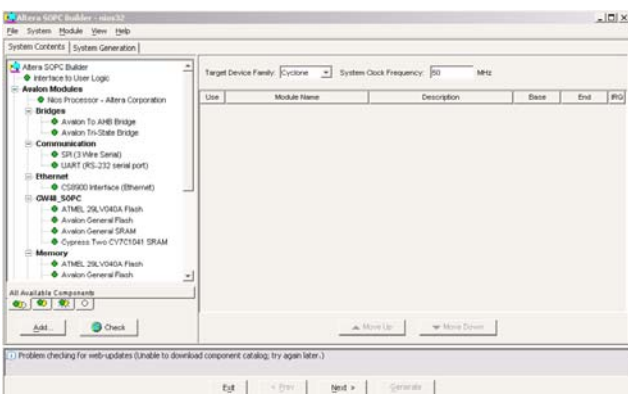


图 5-3 SOPC Builder 设计界面

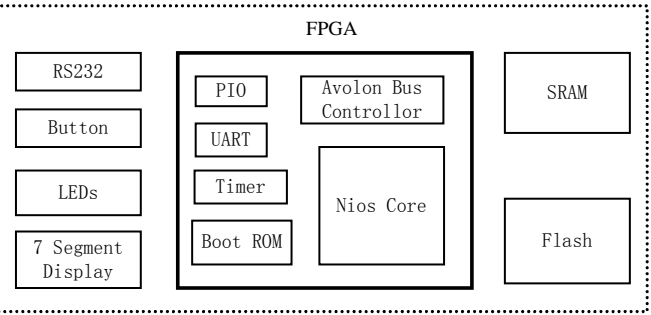


图 5-4 一个基本的 SOPC 系统结构

2、基本 SOPC 系统

这个基本的 SOPC 系统大致可以分为三个部分：FPGA 部分、存储器部分和外围元件部分（图 5-4）。

FPGA 部分是建立在 FPGA 上的，核心是 Nios CPU Core，我们需要在 SOPC Builder 中需要设计的就是 FPGA 部分。在进行具体的设计之前，从图 5-4 了解需要建立的基本 SOPC 系统。即以下要建的 Nios 系统包含的元件模块有，FPGA 内部：一个 Nios CPU 核；连接于 Nios 核的是，一个存放启动和调试程序的内部存储器，Boot ROM、一个 UART 串行通信电路模块（RS232 核）、一个内部定时器、一个 Avolon 总线控制器和一些 PIO 外围接口模块。为使 Nios 系统正常工作，在 FPGA 外围必须接有一个 RS232 通信口、一些控制键、几个发光管和数码管以及 SRAM 和 Flash ROM。

3、加入 Nios CPU Core

首先是加入 CPU 核，选择 SOPC Builder 的组件选择栏中的“Avalon Modules”→“Nios Processor”，点击鼠标右键，选择“Add New Nios Processor...”，打开添加 Nios 对话框(图 5-5)。Nios CPU 核有两种结

构: Nios-16 (16 位 CPU) 和 Nios-32 (32 位 CPU), 区别主要在于 CPU 内部数据带宽。这里选择默认的 Nios-32。在“Configuration Option”中选择“Standard Debug/Average LE usage”, 在“Configuration Options”框中选中“Enable Advanced Configuration Controls”。以便此 CPU 同时还能用来调试程序。然后点击“Next”几次, 进入“Debug”选页, 选中“Enable Nios OCI Debug module”。

最后点击“Finish”按钮完成 Nios CPU Core 的添加过程, 随后 SOPC Builder 的界面就会改变(图 5-6)。在图 5-6 中可以看到, “nios\_0” 作为一个 CPU Core 组件已经加入 SOPC 系统。而在下方提示栏中显示了加入 Nios 后的相关信息, 和下一步的操作提示。提示中指出, 已经有 Avalon 总线的 Master (主控制器) 存在。这里的 CPU 核的名称为“nios\_0”, 需要更改其名字, 选中“nios\_0”, 右键选择“Rename”, 更改组件名称为“cpu”。

#### 4、加入 boot\_monitor\_rom

一个 CPU 系统往往需要一个 Boot ROM 用于系统引导, 在 SOPC 系统中一般也需要这种 Boot ROM, 在 Boot ROM 中可以放入 GERMS Monitor 调试程序, 以便从外界控制 Nios CPU。在组件选择栏中选择“Memory” → “On-Chip Memory”, 打开加入界面, 如图 5-7 所示。

选择“Memory Type”为“ROM (read-only)”, 数据位宽 (Data Width) 为 32 位 (32 bits), 容量 (Size) 为 2KB。按“Next”按钮, 进入 Boot ROM 的内容设置(图 5-8)。选择“GERMS Monitor”。GERMS Monitor 是一个监控程序, 用于 Nios 的仿真调试。然后更改组件名称为“boot\_rom”。

5.2.4 加入 UART UART 是“通用异步收发器”的英文缩写, 就是常用的串口, 基本的 SOPC 系统可以通过串口与上位机通信, 也用于 Nios 系统的仿真调试。在组件选择栏中选择“Communication” → “UART (RS232 series port)”加入。选择波特率为 115200, 其余按照默认配置, 点击“Finish”按钮完成加入。更改组件名称为“uart1”。

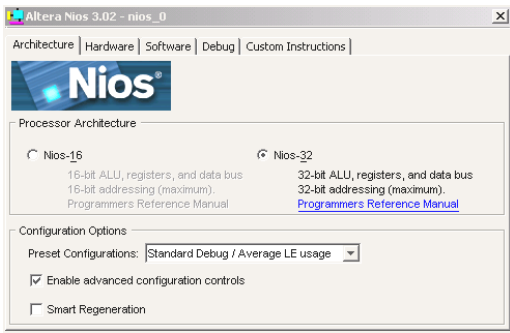


图 5-5 加入 32 位 Nios 核

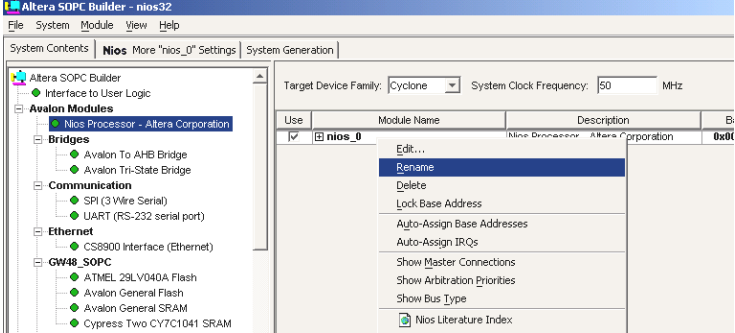


图 5-6 更改组件名称

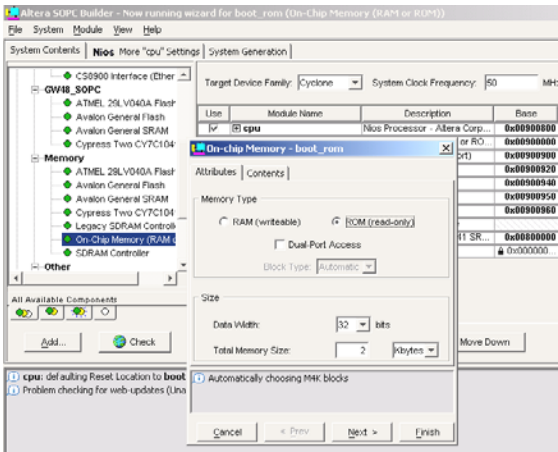


图 5-7 加入 Boot ROM

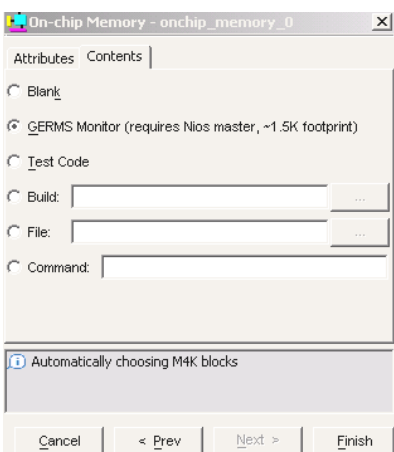


图 5-8 Boot ROM 内容设置

5、加入 Timer 在组件选择栏中选择“Others” → “Interval Timer”加入 SOPC 系统的内部定时器。一切都按照默认配置, 点击 Finish 完成加入。更改组件名称为“Timer1”(图 5-9)。

6、加入 Button PIO 在这, 一般需要加入用于按键的 PIO, PIO 就是通用 I/O 口。在组件选择栏中选择“Others” → “PIO”加入。选择为 4 位, 以对应 4 个按键, 并改成“Input”输入模式。点击“Next”进行输入选项设置(图 5-10)。

选择产生中断请求 (IRQ)，为双边沿触发，边沿模式 (图 5-11)，点击“Finish”完成加入。更改组件名称为“button\_pio”。

7、加入 Led PIO。加入发光二极管 LED PIO，在组件选择栏中选择“Others”→“PIO”加入，选择为 8 位，以对应 8 个 LED，并改成“Output”输出模式。点击“Finish”完成加入。更改组件名称为“led\_pio”。

8、加入数码管 PIO。加入两个 7 段码数码管 PIO 在组件选择栏中选择“Others”→“PIO”加入。选择为 16 位 (图 5-12)，以对应 2 个 7 段码，并改成“Output”输出模式。点击“Finish”完成加入。更改组件名称为“seven\_seg\_pio”。

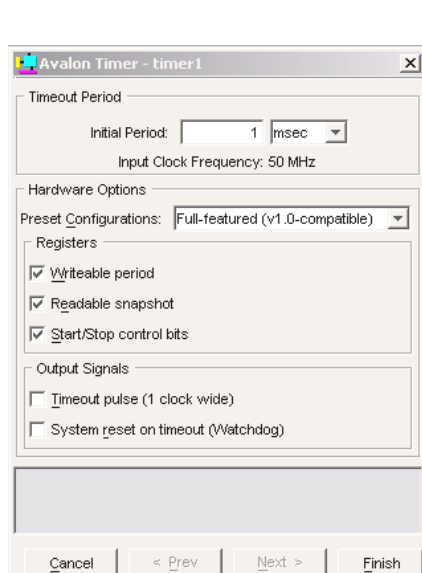


图 5-9 加入 Timer

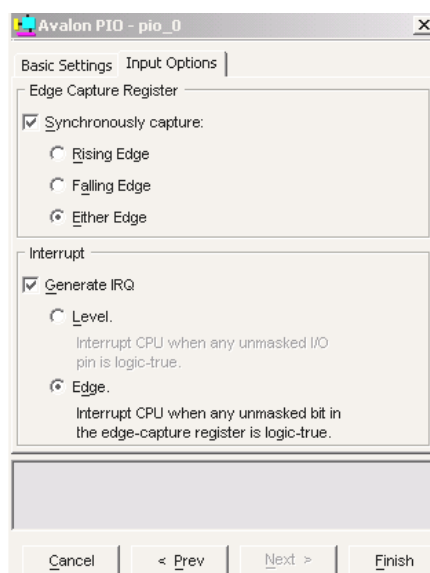


图 5-10 加入 Button PIO

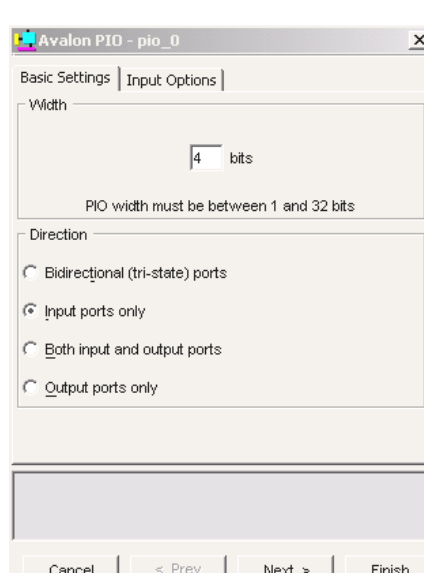


图 5-11 Button PIO 的输入设置

9、加入 Avalon 三态总线桥。Nios CPU 与 SRAM、Flash 相接需要 Avalon 三态总线桥。在组件选择栏中选择“Bridge”→“Avalon Tri-State Bridge”，加入 (图 5-13)。点击“Finish”完成加入。更改组件名称为“tri\_state”。

10、加入 SRAM。有了总线控制器就能在 FPGA 外围接其他总线元件了，首先接 SRAM。在组件选择栏中选择“GW48-SOPC”→“Cypress Two Cy7c1041 SRAM”，加入，选择默认设置。点击“Finish”完成加入。改名称为“sram1”。

11、加入 Flash。在组件选择栏中选择“GW48-SOPC”→“Avalon General Flash”，加入 Flash ROM，在“Attributes”栏选择地址线和数据线宽；在“Timing”栏，设置 nios 对 Flash 的读写时序；此后选择默认设置。点击“Finish”完成加入。更改组件名称为“general\_flash1”。图 5-16 就是本系统的 Nios 基本元件安装设置完成的界面。

12、Flash ROM 锁定地址。Flash 锁定地址就是首先确定 Flash 中放程序的首地址。这里设定 0x00000000。编辑图 5-16 的窗，使“general\_flash1”元件的 BASE 地址为 0x00000000。然后右键点击“general\_flash1”元件，选择“Lock Base Address”，锁定设置好的“general\_flash1”的基地址 (图 5-17)。

13、调整所有存储器的地址。如图 5-18 所示，选择菜单“System”，在下拉栏中选择“Auto-Assign Base Addresses”的自动基地址分配项。完成后如图 5-19 所示。

14、调整所有存储器的地址。点击“Next”后，进入图 5-20 界面，选择设置各种程序放置的位置，选择启动程序“Reset Location”放在 boot rom 中，中断矢量表“Vector Table”、程序、数据，都放在 sram1 中。最后完成的设置如图 5-21 所示。然后在“System Boot ID”内加入系统启动 ID 码。点击“Next”后，进入图 5-22 界面，选择产生 SDK 开发程序、VHDL 系统表述和仿真文件。点击“Generate”按钮后，启动系统生成，让 SOPC Builder 帮助 Nios 开发者生成 SOPC 系统。在这个过程中会生成一个“cpu\_sdk”的目录，建立了 SOPC 的软件开发环境，也同时生成了用于 QuartusII 编译的 HDL 文件。如果没有错误，将出现图 5-23 所示的界面。

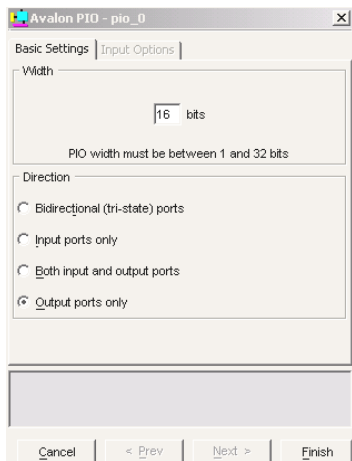


图 5-12 加入两个数码管 seven\_seg\_pio

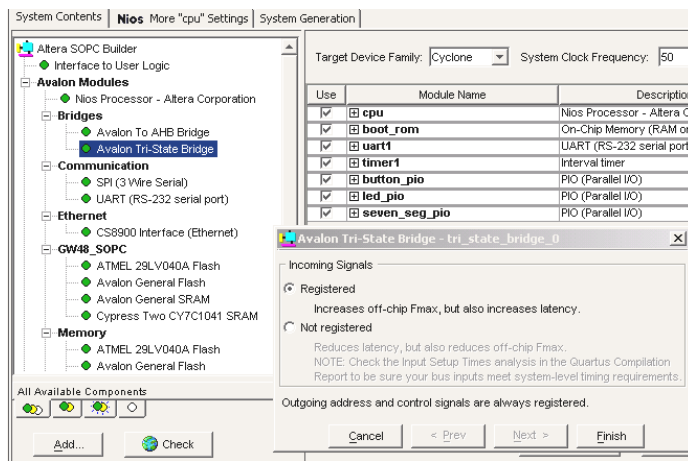


图 5-13 加入 Avalone tri bus 三态总线控制器

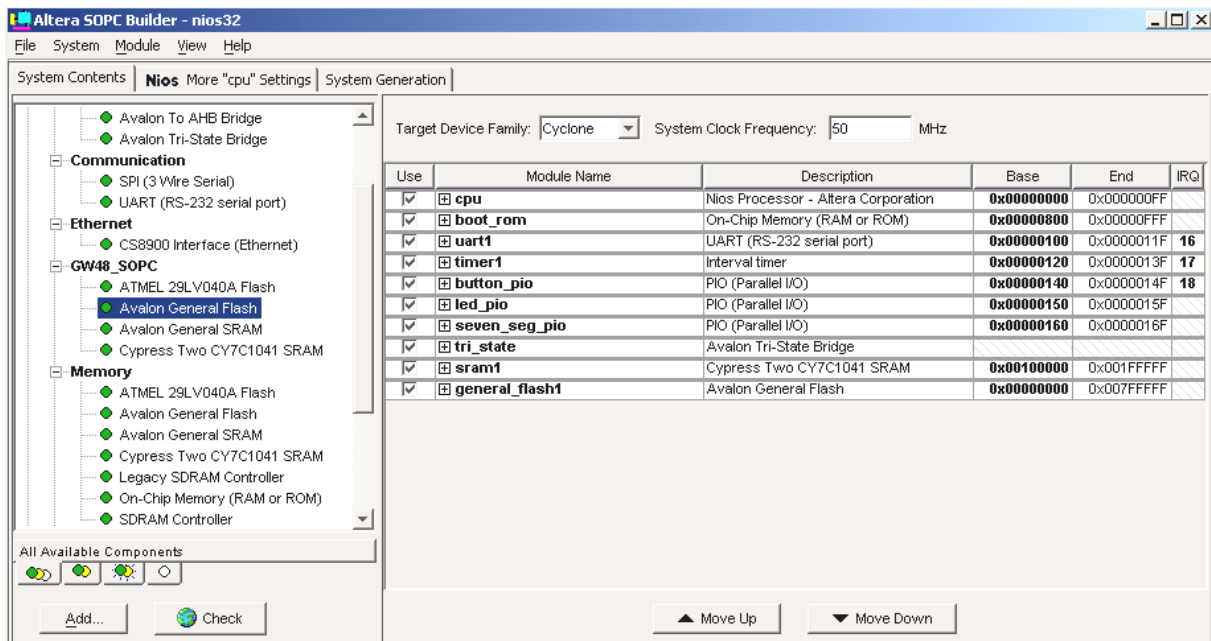


图 5-16 Nios 系统基本元件安装设置完成

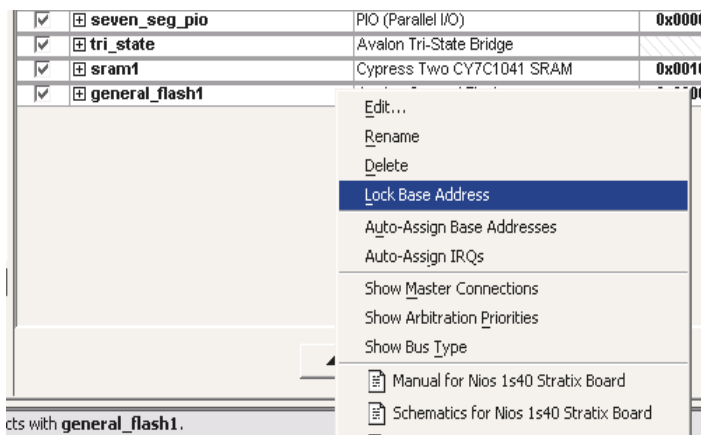


图 5-17 锁定 Flash ROM 的起始地址

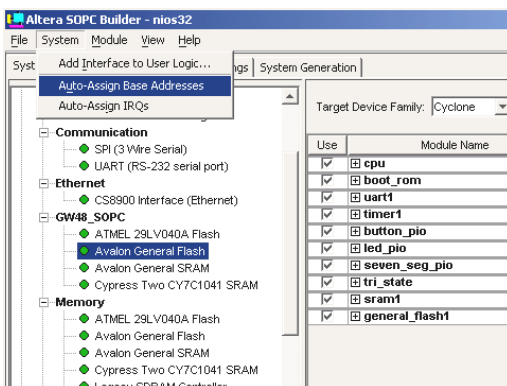


图 5-18 选择对所有存储器自动分配地址



Use	Module Name	Description	Base	End	IRQ
<input checked="" type="checkbox"/>	<b>cpu</b>	Nios Processor - Altera Corporation	<b>0x00900800</b>	0x009008FF	
<input checked="" type="checkbox"/>	<b>boot_rom</b>	On-Chip Memory (RAM or ROM)	<b>0x00900000</b>	0x009007FF	
<input checked="" type="checkbox"/>	<b>uart1</b>	UART (RS-232 serial port)	<b>0x00900900</b>	0x0090091F	<b>16</b>
<input checked="" type="checkbox"/>	<b>timer1</b>	Interval timer	<b>0x00900920</b>	0x0090093F	<b>17</b>
<input checked="" type="checkbox"/>	<b>button_pio</b>	PIO (Parallel I/O)	<b>0x00900940</b>	0x0090094F	<b>18</b>
<input checked="" type="checkbox"/>	<b>led_pio</b>	PIO (Parallel I/O)	<b>0x00900950</b>	0x0090095F	
<input checked="" type="checkbox"/>	<b>seven_seg_pio</b>	PIO (Parallel I/O)	<b>0x00900960</b>	0x0090096F	
<input checked="" type="checkbox"/>	<b>tri_state</b>	Avalon Tri-State Bridge			
<input checked="" type="checkbox"/>	<b>sram1</b>	Cypress Two CY7C1041 SRAM	<b>0x00800000</b>	0x008FFFFF	
<input checked="" type="checkbox"/>	<b>general_flash1</b>	Avalon General Flash	<b>0x00000000</b>	0x007FFFFF	

图 5-19 地址分配完成

### 3 SOPC 整体系统生成

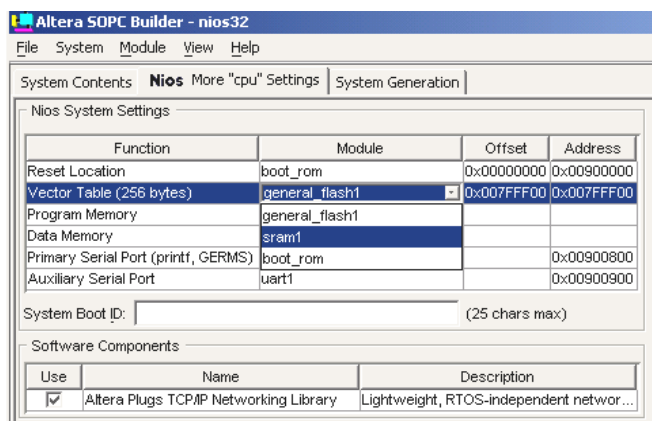


图 5-20 选择设置各种程序放置的位置

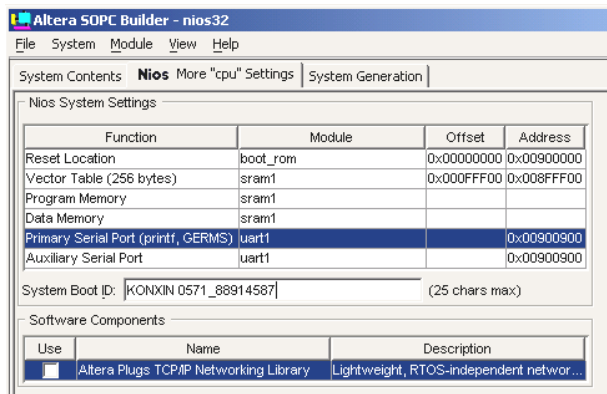


图 5-21 选择结束并加入系统启动 ID

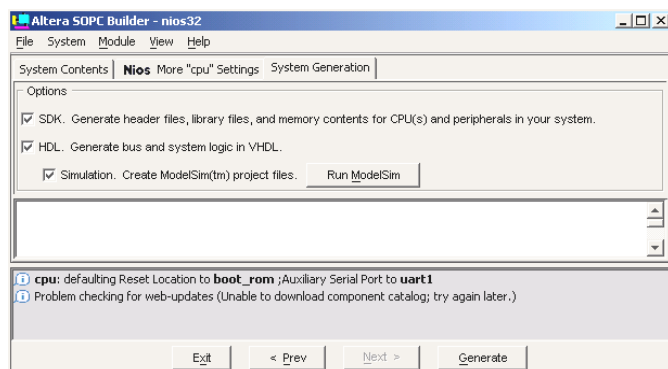


图 5-22 选择生成文件并启动生成 Nios 系统

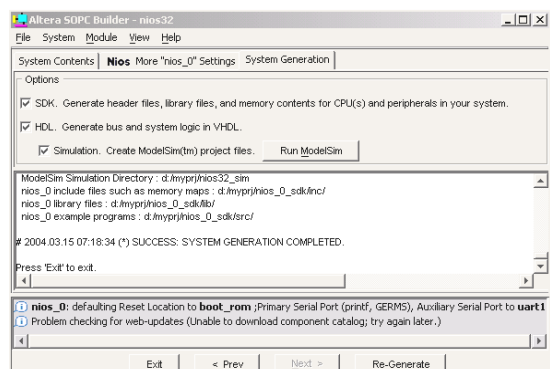


图 5-23 Nios 系统生成成功

### 4 Nios 硬件系统生成

点击“Exit”，退出 Nios 编辑界面。在主控制窗口，选择“File”→“New”，在出现的如图 5-24 的窗中选择“Block Diagram/Schematic File”，建立原理图编辑窗（图 5-25）。在此窗中右键选择“Insert”→“Symbol”，调出刚才设计好的 Nios 系统。在如图 5-26 左侧的“Project”下选择“nios32”，点击“OK”，即可调入 Nios 系统模块（图 5-27）。之后，将此原理图文件另存为工程名“nios\_dvp”，然后设定该文件为项目当前文件，方法是选择“Project”菜单项下的“Focus to Current Entity”即可。下面的工作是加上引脚，所定引脚，进行全程编译/综合/适配。但考虑到要所定的引脚有 100 多，十分烦琐。可以用下面的方法来解决：

关闭刚才设计工程，打开目录为“nios\_practice\_model”中的工程“m0a”。这是一个带有所有必须引脚锁定信息的空壳工程。在此工程中从新如本文介绍的那样从头开始设计 Nios 系统（图 5-28）。直至完成所有设计，到出现图 5-26 的模块。双击左侧的工程文件名“M0A”，将弹出带有引脚的原理图（图 5-29，30），然后在带有引脚的空壳原理图编辑窗内调入已生成的 Nios 系统模块，拼接好好模块，完成所有设计，准备编译。

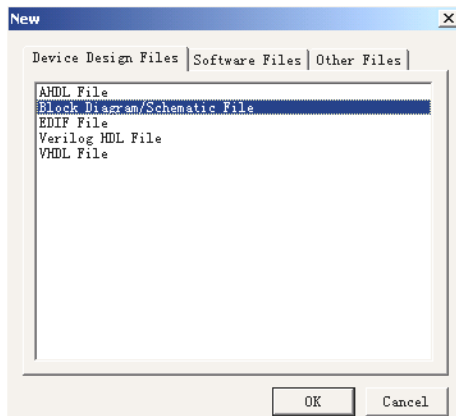


图 5-24 建立原理图文件

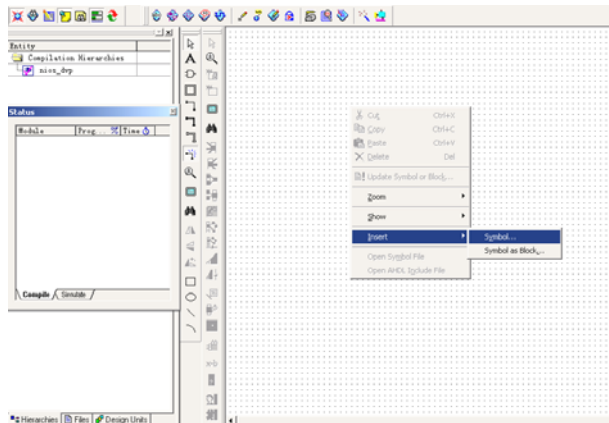


图 5-25 调入已生成的 Nios 系统模块

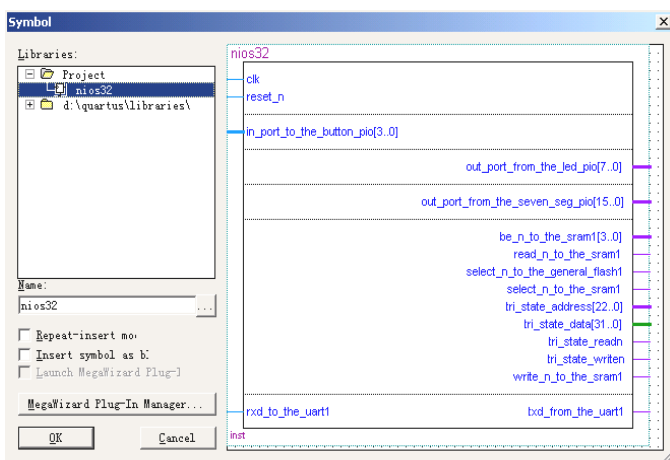


图 5-26 从库中选择已生成的 Nios 系统模块

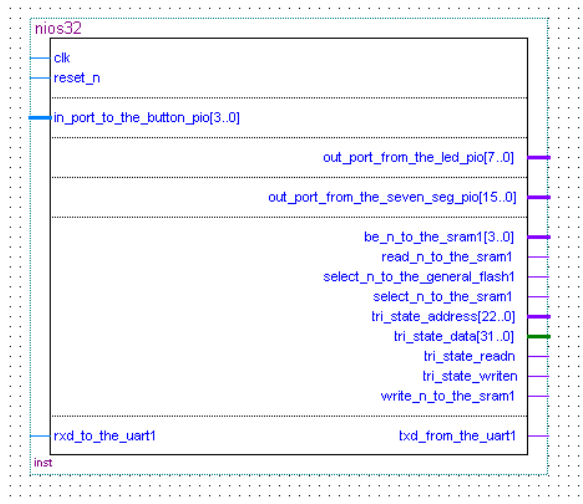


图 5-27 调入了已生成的 Nios 系统模块

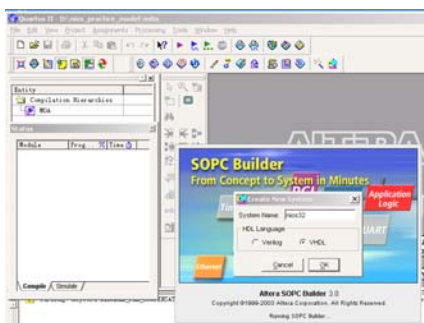


图 5-28 调入已生成的 Nios 系统模块

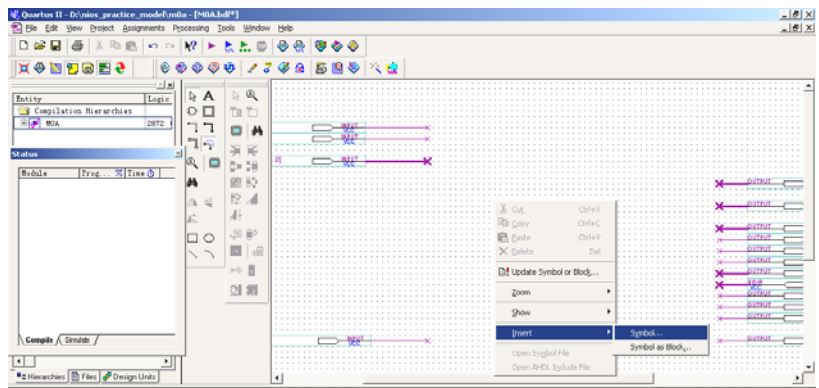


图 5-29 在带有引脚的空壳原理图编辑窗内调入已生成的 Nios 系统模块

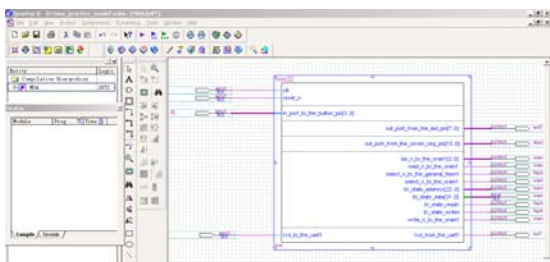


图 5-30 拼接好模块，完成所有设计，准备编译

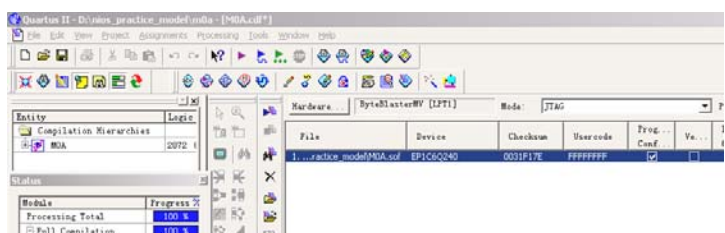


图 5-31 向 FPGA 下载

1、设置编译 SOPC 系统 然后设定该文件为项目当前文件，用 QuartusII 进行编译。基于 Nios 的 SOPC



系统硬件开发就完成了。最后将设计好的 SOF 文件通过 JTAG 口下载到实验系统上的 FPGA 中（图 5-31），准备软件调试。

2、下载完成后，将 GW48-SOPC 系统设定为模式 5，打开调试窗 Nios SDK shell（图 5-32）。方法是点击 Windows 左下角的“开始”，选择“Altera”→“Nios3.02”→“Nios SDK Shell”，打开如图 5-33 的调试窗，键入命令：

`nr -t -r` 再按计算机回车键。即向实验系统上 FPGA 内的 Nios 系统发出命令，要求将 RS232 口强制设置为显示终端口。然后按实验系统的键 8（这是 Nios 的复位键），使之从低点平到高点平

（高电平时进入正常工作），这时在图 5-33 所示的窗口将出现信息：nios 系统的版本码和设计系统的用户 ID 码。这表示 FPGA 中的 Nios 工作正常。此窗就留作程序调试用的终端显示窗。然后再打开另一个 Nios SDK shell 窗，在此窗中将路径设置在本设计工程目录的 src 子目录上（图 5-34）：

`d/nios_practice_model/cpu_sdk/src`

然后输入命令。并按回车键：

`nb -d hello_world.c`

即对此目录中一个已有的 C 程序 `hello_world.c` 进行编译，并产生调试信息。然后再输入命令，并按回车键：

`nd hello_world.srec`

即向 Nios 系统下载，并执行，此时将弹出单步调试界面（图 5-35）。当按动左上角第 2 个键，即可以看到单步执行 C 程序，同时从第一个 SDK shell 窗中看到执行的结果。



图 5-32 打开调试窗 Nios SDK shell

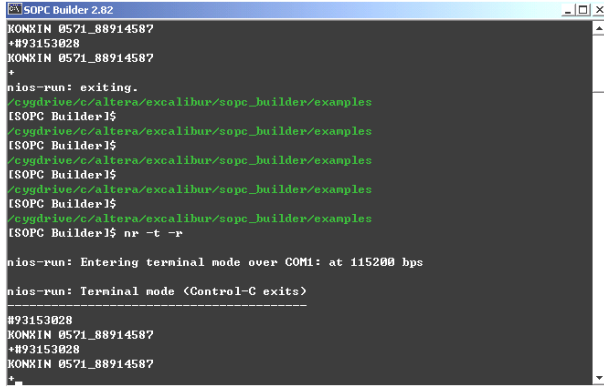


图 5-33 打开调试窗 Nios SDK shell

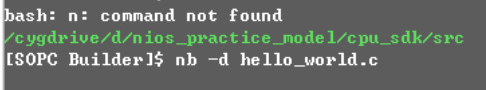


图 5-34 在此输入命令： `nb -d hello_world.c`

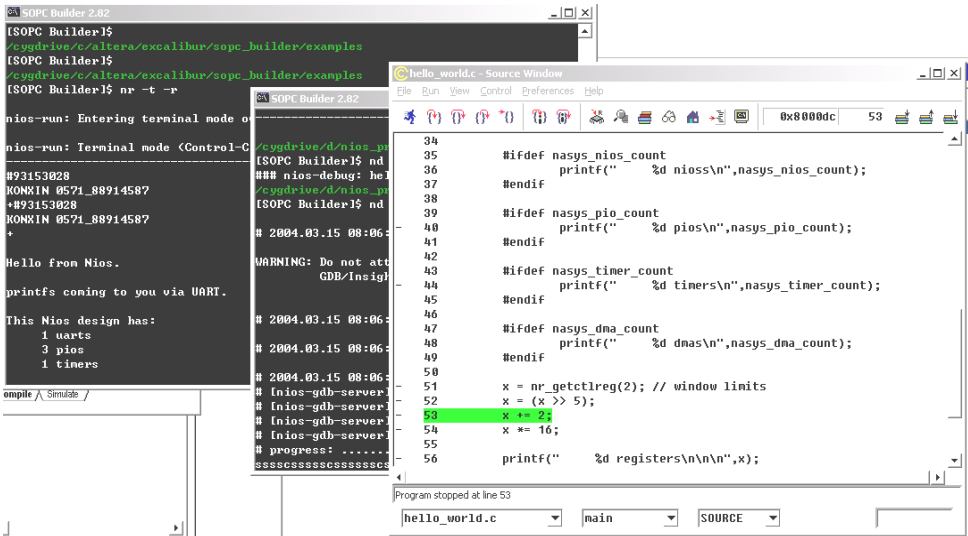


图 5-35 单步运行

## 实验十二 32 位 Nios CPU 测控系统串口接收程序设计

(1) 实验目的：用中断编程方式处理串口数据收发，以实现简单测控系统的解析和发送。

(2) 实验原理：串口中断编程的原理已在 8.2 节中详细阐述，8.2.3 节中的 `uart_isr_test` 程序也演示了串口数据帧的解析和发送，如图 8-14 所示。

(3) 实验内容：练习完成 `uart_isr_test` 程序设计示例，仿造该示例设计一个接收和发送简单测控系统的串口程序。串口数据收发的帧结构如表 8-20 所示：

表 8-20 串口收发数据帧结构

帧类型	格 式	说 明
接收端数据帧	ATT1T2T3T4<回车符>	AT 后接四个 ASCII 码表示的温度数据，每个温度数据占 4 字节，如 AT+012-005-021+111
接收端命令帧	AT?<回车符>	AT 后接一个问号，查询四个温度数据，如 AT?
接收端回复帧 1	ATOK<回车符>	接收完全一个数据帧后发送回复信息，如 ATOK
接收端回复帧 2	ATT1T2T3T4<回车符>	AT 后接四个 ASCII 码表示的温度数据，每个温度数据占 4 字节，如 AT+012-005-021+111

(4) 实验思考题：如果接收数据帧不仅有温度数据，还可能有电流、电压、压力等数据参数，那么串口收发数据帧结构应如何定义？如果接收端和发送端并不仅仅是一对一，是单点对多点或多点对多点时又如何处理？串口程序中的串口接收状态应如何声明才能使程序结构更为简洁、有效？请编程实现一个最简单的应用示例。

(5) 实验报告：完成实验报告。

## 实验十三 GSM 短信模块程序设计

(1) 实验目的：了解串口编程及 MC35i 短信模块的使用，实现简单 GSM SMS 收发功能。

(2) 实验原理：通过 SOPC 系统上的串口连接 GSM 短信模块 MC35i 实现一个 GSM SMS 系统，如图 8-14 所示。通过 AT Command Set 完成短信的收发操作。

(3) 实验内容：连接 GSM 短信模块。使用下列 AT 命令实现短信的收发。

方向	AT Command	含义及说明 ( )
Nios→MC35i	AT<CR>	连接 MC35i，<CR>表示回车，即'\r'、0x0d
MC35i→Nios	OK<CR><LF><CR><LF>	MC35i 应答
Nios→MC35i	ATE0<CR>	回音消除
MC35i→Nios	OK<CR><LF><CR><LF>	
Nios→MC35i	AT+CMGF=1	设置为 Text 模式
MC35i→Nios	OK<CR><LF><CR><LF>	
Nios→MC35i	AT+CMGW="8613112345678"<CR>	
MC35i→Nios	<CR><LF>>	MC35i 回“》”，提示输入文本
Nios→MC35i	Abcd1234<Ctrl+Z>	输入短信内容（文本），加“Ctrl+Z”结束输入
MC35i→Nios	OK<CR><LF><CR><LF>	发送完毕
Nios→MC35i	AT+CMGL="REC UNREAD"<CR>	查询 SIM 卡中已接收未读短信
MC35i→Nios	.....	列表已接收未读短信
Nios→MC35i	AT+CMGR=<INDEX><CR>	读出指定的索引号 (<INDEX>) 对应的短信

MC35i→Nios	+CMGR:<stat>,<oa>,<[alpha]>,<scts> [,<tooa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length> ]<CR><LF><data> OK<CR><LF><CR><LF>	<data>就是短信内容
------------	---	--------------

(4) 实验思考题：结合上面的短信收发程序，能不能编写通过短信实现对 Nios 系统上的 PIO 进行控制，或者把 Nios 系统上的按键信息发送到远端。

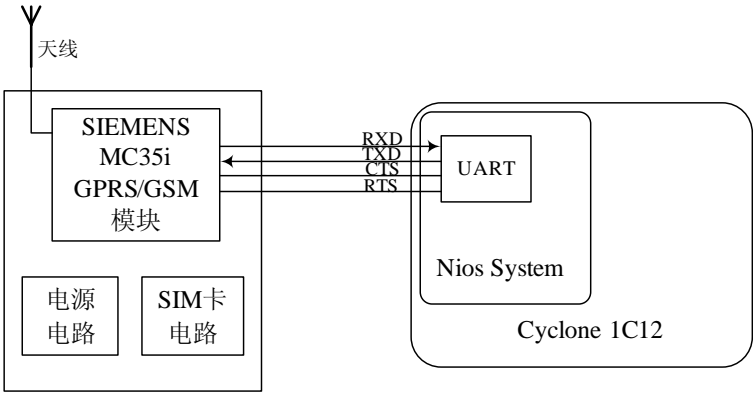


图 8-14 SMS 系统

### 实验十四 基于 32 位 Nios CPU 的秒表程序设计

- (1) 实验目的：学习 Nios 系统软件程序设计的综合应用。
- (2) 实验原理：应用 Nios 系统定时器、按键、中断处理、LED 和 LCD 外设编程原理都已在第 8、9 章中做过相应介绍。
- (3) 实验内容 1：设计一个具有以下功能的秒表程序（例 9-7）。程序结构如图 9-6 所示。其功能要求为：
  - 具有百分秒计时功能，计时时间显示在 16×2 标准的 LCD 上。
  - 使用两个按键 s0 和 s1，按键 s0 提供复位功能，s1 提供启动、暂停计时功能。

注意： 实验内容中的 LCD 显示功能在 Altera 提供的 Nios 开发板上可以实现。GW\_SOPC 实验箱提供的 LCD 是 64X128 的，因此在 GW\_SOPC 实验箱上完成本实验时可以将 LCD 初始化和 LCD 显示功能忽略，在示例代码中的 LCD 相关代码也要作相应忽略。

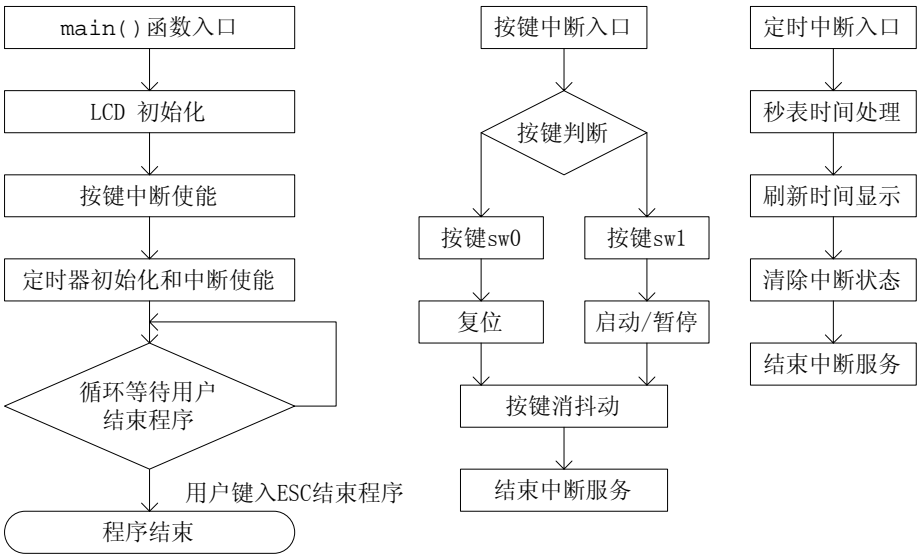


图 9-6 实验内容 1 程序结构

下面列出了实验内容 1 的示例代码。

【例 9-7】简易秒表编程示例

```

/* 功    能：秒表启动、暂停、复位，LCD显示秒表时间，最大23时59分59秒。 */
/* 源 文 件：ex_09_03_1.c */
#include <excalibur.h>
// 全局变量
static np_pio *pio = na_button_pio;      // 按键指针
static np_timer *timer = na_timer1;      // 定时器指针
const long nTimerPeriod = nasys_clock_freq / 100; // 定时周期10毫秒
static long nTimerCount = 0;              // 0.01秒计时数值，每次定时中断变化
const long MAX_COUNT = 3600 * 24 * 100;   // 0.01秒计时最大数值
static int nButtons = 0x000F;             // 按键检测数值
static char strTime[16];                  // 时间显示字符串
// 用户子程序
void button_isr(int context);              // 按键中断服务子程序
void timer_isr(int context);              // 定时中断服务子程序
void start_timer(void);                   // 启动定时器
void pause_timer(void);                   // 暂停定时器
void show_time(void);                     // 时间显示子程序
// main主程序
int main(void)
{
    // LCD初始化
    nr_pio_lcdinit(na_lcd_pio);           // GW_SOPC开发板上可忽略

    // 安装按键中断服务子程序、按键中断使能
    nr_installuser_isr(na_button_pio_irq, button_isr, (long)pio);
    pio->np_piointerruptmask |= 0x000F;
    // 设置定时周期0.01秒
    timer->np_timerperiodh = nTimerPeriod >> 16;
    timer->np_timerperiodl = nTimerPeriod & 0xffff;
    // 安装定时器中断服务子程序
    nr_installuser_isr(na_timer1_irq, timer_isr, (long)timer);
    // 设置定时器循环工作方式
    timer->np_timercontrol |= np_timercontrol_cont_mask;
    // 定时器中断使能
    timer->np_timercontrol |= np_timercontrol_ito_mask;
    // 开启定时器
    start_timer();
    // 等待用户输入ESC键值结束程序
    while (nr_rxchar() != 27)
    {
    };
    // 定时器停止工作
    pause_timer();
    // 提示退出程序，系统复位
    printf("\nExit ex09_03_01.\004\n");
    nr_jumptoreset();
}
// start_timer子程序，启动定时器
void start_timer(void)
{
    // 定时器控制寄存器stop位清零，start位写1
    timer->np_timercontrol &= ~np_timercontrol_stop_mask;
    timer->np_timercontrol |= np_timercontrol_start_mask;
}
// pause_timer子程序，暂停定时器
void pause_timer(void)
{
    // 定时器控制寄存器start位清零，stop位写1
    timer->np_timercontrol &= ~np_timercontrol_start_mask;
    timer->np_timercontrol |= np_timercontrol_stop_mask;
}
// show_time子程序，在LCD模块和串口终端上显示时间
void show_time(void)
{
    int i;
    sprintf(strTime, "%02d:%02d:%02d.%02d",

```

```

        (nTimerCount / 360000) % 24,
        (nTimerCount / 6000) % 60,
        (nTimerCount / 100) % 60,
        nTimerCount % 100);
// 在LCD屏上显示时间
nr_pio_lcdwritescreen(strTime);    // GW_SOPC开发板上可忽略
// 在串口终端上显示计时时间
for (i = 0; i < 16; ++i)
    printf("\b");
printf("%s", strTime);
}
// button_isr, 按键中断服务子程序
void button_isr(int context)
{
    // 判断是否有按键按下
    if (nButtons != pio->np_piodata & 0x000F)
    {
        // 如果是按键SW0, 执行复位操作
        if (~(pio->np_piodata) & 0x0001)
        {
            pause_timer();
            nTimerCount = 0;
            // 刷新时间显示
            show_time();
        }
        // 如果是按键SW1, 执行启动、暂停操作
        if (~(pio->np_piodata) & 0x0002)
        {
            // 如果定时器已停止, 启动定时器; 反之则暂停定时器
            if (timer->np_timercontrol & np_timercontrol_stop_mask)
                start_timer();
            else
                pause_timer();
        }
        // 保存当前按键键值到nButtons
        nButtons = pio->np_piodata & 0x000F;
        // 按键消除抖动
        nr_delay(10);
    }
}
// timer_isr, 定时器中断服务子程序
void timer_isr(int context)
{
    np_timer *timerT = (np_timer *)context;    // 得到定时器指针
    // 0.01秒定时中断计数加1
    nTimerCount++;
    // 判断计数值是否超出最大表示范围
    if ( nTimerCount >= MAX_COUNT )
    {
        // 计数超出范围, 停止定时器, 计数值清零
        nTimerCount = 0;
        pause_timer();
    }
    // 刷新时间显示
    show_time();
    // 清除定时器中断状态
    timerT->np_timerstatus = 0;
}

```

(4) 实验内容 2: 在实验内容 1 的基础上添加两个按键 s2 和 s3, s2 用于依次记录秒表时间, s3 用于依次读取记录时间。读取记录时间也显示在 LCD 上。两个 LED 数码管则显示记录个数, 最多 99 个。

(5) 实验思考题: 实验中使用的定时器是 Nios 系统内部的定时器单元, 请使用 VHDL 硬件描述语言设计一个硬件秒表模块, 以替代 Nios 系统中的定时器单元。

在 standard\_32 硬件设计项目中添加已设计好的秒表模块, 综合、适配、仿真成功之后, 下载到 Nios 开发板上。修改实验内容 2 的软件程序, 实现相同的功能。试比较该程序与实验内容 2 程序的不同。

(6) 实验报告: 根据以上要求完成实验报告。

## 实验十五 Nios Avalon Slave 总线外设（PWM 模块）设计

(1) 实验目的：按照本章内容设计一个 Avalon 从 (Slave) 外设，完成 PWM 功能。

(2) 实验原理：按照 Avalon 总线规范设计 Avalon Slave，添加到 SOPC 组件库中。

(3) 实验内容：

步骤一：按照本章相关内容进行 Avalon PWM 从外设的设计，并加入到 SOPC Builder 组件库。

步骤二：在 Nios 系统中加入步骤一制作的 PWM 组件，驱动直流电机运转，并控制其速度。

(4) 实验思考题：在本章示例的 PWM 外设模块，是遵循 Avalon Slave 同步传输时序的，PWM 模块的时钟也是系统的时钟，如果 PWM 模块的工作时钟与系统的时钟不一致，如何设计该 Avalon 从外设。

## 实验十六 Nios Avalon Slave 总线外设（数码管动态扫描显示模块）设计

(1) 实验目的：参照本章内容设计一个 Avalon 从外设，完成数码管动态扫描显示功能。

(2) 实验原理：按照 Avalon 总线规范，设计 Avalon Slave，添加到 SOPC Builder 组件库。该外设可以完成 8 位数码管的动态扫描显示。

(3) 实验内容 1：

步骤一：参照实验 10-1 进行 Avalon 数码管动态扫描显示从外设的设计，并加入到 SOPC Builder 组件库。

步骤二：在 Nios 系统中加入步骤一制作的动态扫描显示组件，驱动数码管显示。

(4) 实验内容 2：用同样方法设计液体显示驱动模块。

(5) 实验思考题：可不可以控制数码管显示的亮度，如何实现。

(6) 实验报告：完成实验报告。

## 实验十七 基于 Nios 的 VGA 显示终端设计

(1) 实验目的：了解 VGA 控制器设计原理及其应用，实现一个 VGA 显示终端。

(2) 实验原理：实验系统结构参见本章图 11-9，所不同的是，实验数据不是由按键而是由实验箱串口 1 输入。VGA 控制器设计原理和 DMA 软件编程参见俄罗斯方块游戏设计部分。

(3) 实验内容：完成本章俄罗斯方块游戏设计部分内容，在该设计的硬件系统上编写一个终端程序，该程序能将用户串口输入显示在一个 VGA 显示器上。汉字和字符的显示可参考俄罗斯方块游戏设计中的图形显示部分代码如 draw\_pixel、draw\_asc、draw\_hz 和 draw\_str 等子程序。要注意显示满屏时的滚屏操作部分的程序设计，该操作的刷新速度不应太慢。

(4) 实验思考题：

a. 在前述实验基础上再添加一个串口命令解析功能，实现表 11-2 所示串口命令：

串口命令解析可查考第 9 章串口中断 uart\_isr\_test 程序代码，-equ 命令解析功能可要求简单些，主要是完成两个 32 位正整数 (0~65535) 的四则运算，如 -equ12+34 这个命令的解析结果为 12+34=56

b. 参考键盘 PS/2 标准，用 VHDL 硬件描述语言描述一个 PS/2 键盘控制模块，然后将其添加到 NIOS 硬件系统中，最后将思考题 a 中的串口部分代码改写为键盘模块代码。

表 11-2 VGA 显示终端解析命令帧

帧 类 型	格 式	说 明
-clr 命令帧	-clr<回车符>	VGA 显示器清屏
-equ 命令帧	-equ 数学表达式<回车符>	计算一个简单的数学表达式，如 -equ12+34
-h 命令帧	-h<回车符>	显示帮助信息
-equ 命令的回复帧	结果字符串<回车符>	显示 -equ 命令的运算结果，如 -equ12+34 结果为 46
-h 命令的回复帧	字符串<回车符>	在 VGA 上同时显示帮助信息。

## 实验十八 DMA 应用和俄罗斯方块游戏设计

全部内容参考《SOPC 技术实用教程》第 11 章第 2 节

## 实验十九 为 Nios 嵌入式系统增加算法加速协处理模块控制指令

以上三项实验主要参考清华大学出版社的《SOPC 技术实用教程》一书的第 7 章、第 8 章、第 9 章、第 10 章、第 11 章和第 12 章。

或所配光盘的路径：/必读文档 / Nios\_32 位嵌入式系统软硬件设计 / \*.pdf 中的 6 个 PDF 文件。

实验课件参考：/CMPUT\_EXPMT/Experiments/Expmt9\_soc / 实验 9-1.ppt

实验示例参考：/CMPUT\_EXPMT/Experiments/Expmt9\_soc / DEMO9\_CPU32

## 实验二十. 计算机体系结构实验

详细实验原理和实验内容参考《现代计算机组成原理》一书



## 实验二十一. 89K51 单片机应用系统软硬件设计实验

**实验示例 1**（硬件系统构成和简单汇编语言设计与下载），假设对于 GW48-CP++系统：

一、用 **QuartusII** 打开工程： \CMPUT\_EXPMT\Experiments\GWCP\_PK3\MCU8951

对于 GW48-CP+系统（GW48-PK2），要拔去右上方的单片机： \GWCP\_PK2\MCU8951

其顶层设计是用原理图表示的，如图 21-1 所示，它包含的部件有：

- 1、“KX8951”是 89K51 的内核，它包含 8031 的所有功能，它的主要信号线功能说明如表 21-1 所示。
- 2、“ram256”是为单片机核配置的数据 RAM，相当于 89C51 的 256 字节内部 RAM，是由 EAB 构成的。
- 3、“rom4KB”是为单片机核配置的程序 ROM，相当于 89C51 内部的 4K 字节 ROM，由 EAB 构成。通过设置，可将容量扩展为 64KB（FPGA 中的 EAB 不够，可以外接 ROM）。ROM 中的程序已被加载其中，是 HEX 格式。首先可以用编辑器编写单片机汇编程序：\*.asm，再用编译器将其编译成 HEX 格式文件。
- 4、“REG8B”/“REG4B”5 个锁存器，可以将 P1 口输出的数据分别锁住，在实验系统上的数码管上显示。锁存信号由 P2 口的不同位提供。

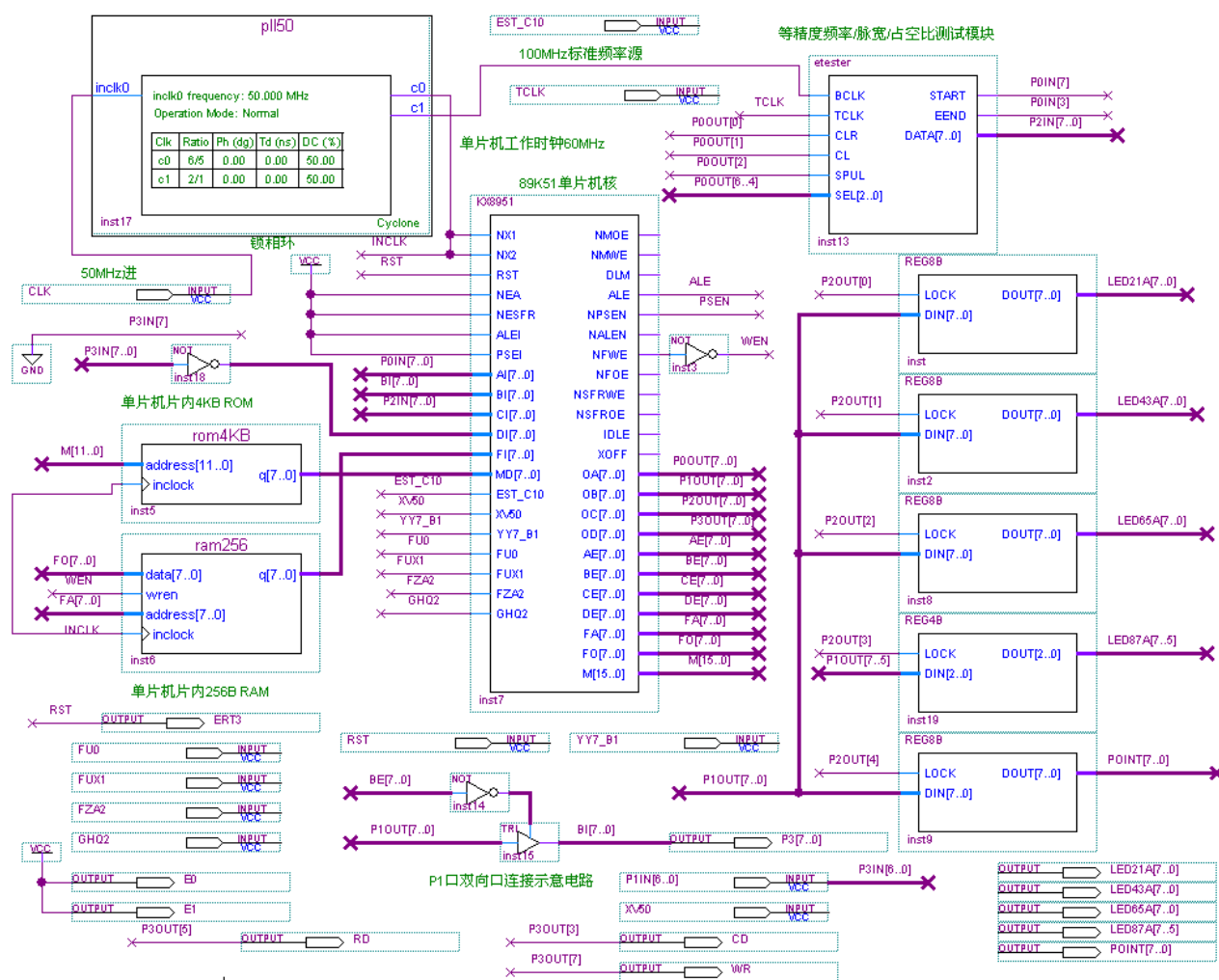


图 21-1 含 89C51 单片机核的单片机应用系统电路图

5、“PLL50”是嵌入式锁相环，输入频率为 50MHz，输出频率有 2 个，一个是 c0 = 60MHz，给单片机作工作时钟；另一个是 c1 = 100MHz，作为等精度频率计的标准时钟（图 21-2）。

6、“ETESTER”是等精度计测试模块，工作原理可参考《EDA 技术与 VHDL》一书。如图 21-2 所示，其 8 位数据与单片机的 P2 口输入口相连，P0 口作为控制口，被测频率输入端被锁定在第 228 脚，在适配板的中上方的一个插针。

7、如图 21-3 所示，图中给出了 P1 口双向口连接的方法：需要一个 3 态端口，3 态控制端由 BE[7..0] 担任，3 态口的输入端接 OB[7..0]（即 P1OUT[7..0]），输出端的内部接 BI[7..0]，外部端口接 P1[7..0]。

表 21-1 单片机核信号端口功能明

单片机信号	端口类型	功能说明
M[15..0]	输 出	程序存储器地址总线
MD[7..0]	输 入	程序存储器数据总线
NMOE	输 出	程序存储器输出使能，低电平有效
FA[7..0]	输 出	片内 RAM 地址总线
FO[7..0]	输 出	片内 RAM 数据输入总线（由单片机核输出）
FI[7..0]	输 入	片内 RAM 数据输出总线
NFOE	输 出	片内 RAM 数据输出使能，低电平有效
NFWE	输 出	片内 RAM 数据写入使能，低电平有效
NSFROE	输 出	外部特殊寄存器输出使能，低电平有效
NSFRWE	输 出	外部特殊寄存器写入使能，低电平有效
NESFR	输 入	如果没有外部特殊寄存器，拉高此电平
OA[7..0]	输 出	P0 口数据输出端，7 位
OB[7..0]	输 出	P1 口数据输出端，7 位
OC[7..0]	输 出	P2 口数据输出端，7 位
OD[7..0]	输 出	P3 口数据输出端，7 位
AI[7..0]	输 入	P0 口数据输入端，7 位
BI[7..0]	输 入	P1 口数据输入端，7 位
CI[7..0]	输 入	P2 口数据输入端，7 位
DI[7..0]	输 入	P3 口数据输入端，7 位
AE[7..0]	输 出	P0 口作为双向口的控制信号，执行输出指令时，为低电平
BE[7..0]	输 出	P1 口作为双向口的控制信号，执行输出指令时，为低电平
CE[7..0]	输 出	P2 口作为双向口的控制信号，执行输出指令时，为低电平
DE[7..0]	输 出	P3 口作为双向口的控制信号，执行输出指令时，为低电平
NEA	输 入	使能程序计数器的值进入 P0 和 P2 口
NX1	输 入	单片机工作时钟输入端
NX2	输 入	单片机工作时钟输入端，但在进入休闲状态时可控制停止
RST	输 入	复位信号线
ALE	输 出	地址锁存信号
NPSEN	输 出	外部程序存储器使能，低电平有效
NALEN	输 出	对 ALE 和 PSEN 信号的双向控制信号，低电平允许输出
XOFF	输 出	振荡器禁止信号，用于省电模式
IDLE	输 出	在休闲模式中，可通过外部控制 NX2 的时钟输入

## 二、实验系统设置和下载

首先向 FPGA 下载 SOF 文件：\CMPUT\_EXPMT\Experiments\GWCP\_PK3\MCU8951，这时在单片机中默认配置了一个程序：\CMPUT\_EXPMT\Experiments\GWCP\_PK3\ASM\KX\_240A.hex，其对应的程序是 KX\_240A.ASM，这是一个对 128X240 液晶显示控制程序。对于此液晶的数据口是 P1 口，控制信号 CD、WR、RD 由 P3 口控制。时钟用短路帽接 50MHz（如果适配板上是 20MHz 晶振，要从新设置图 21-2 中的锁相环）。对于 GW48-CP+系统（GW48-PK2），对应的程序是：GWPK2LCD.hex

实验系统选择模式 3，此时键 8 控制单片机复位，按键一次，即复位一次；键 1 至键 7 分别控制单片机 P3 口的 P3.0、P3.1、P3.2、P3.3、P3.4、P3.5、P3.6，按下为高电平，松开为低电平。

将 14 针短连线连接 128X240 液晶屏下方的“J\_FPGA”和“J\_LCD”两个接口，按动一次复位键（键 8），就可以看见液晶屏上显示“杭州康芯电子有限公司”等字样。

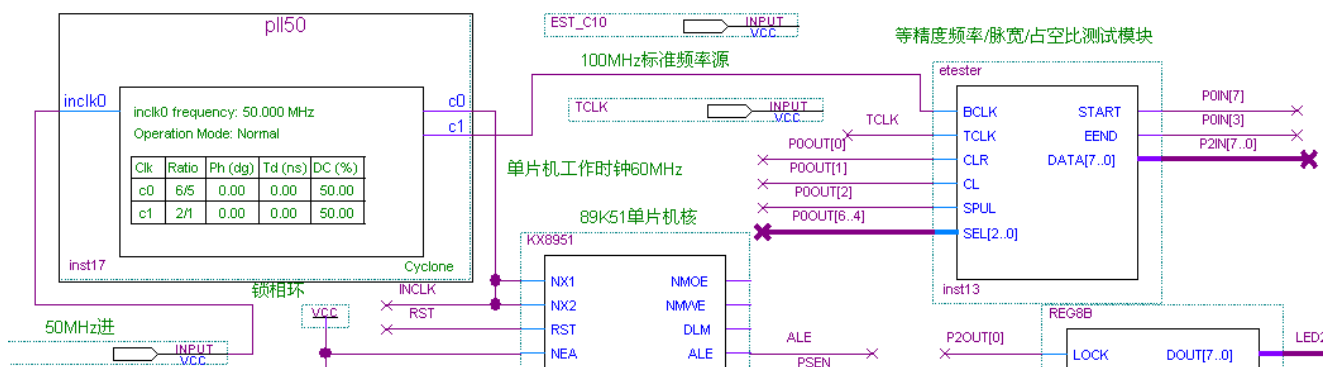


图 21-2 嵌入式锁相环和等精度频率计部分电路图

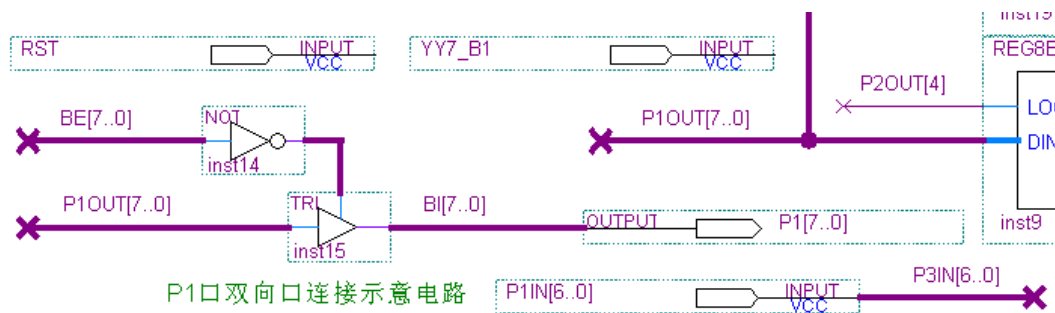
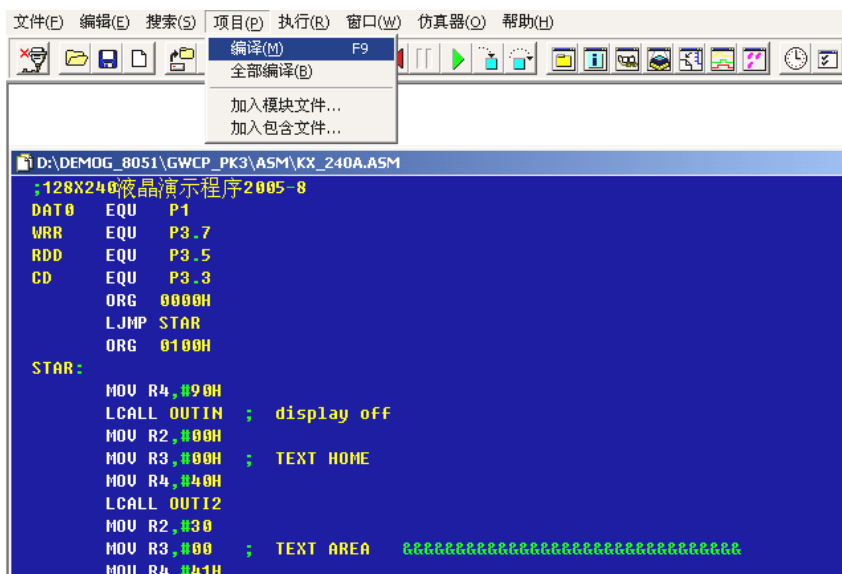


图 21-3 将 P1 口连成双向口方式的电路图

### 三、单片机软件修改、编译和下载

如果要修改其中单片机软件，可按照如下步骤进行：

- 1、打开汇编语言编辑编译器 \CMPUT\_EXPMT\Experiments\GWCP\_PK3\ASM\WAVE\BIN\WAVE. 调出对应程序（图 21-4），修改后进行编译产生 hex 文件。



- 2、首先向 FPGA 下载 SOP 文件，再在 QuartusII 上打开 RAM/ROM 在系统编辑器（图 21-5，详细方法可参考《EDA 技术与 VHDL》一书）。

打开如图 21-5 的窗口后，首先在右上角设置“Setup”，再点击 ROM 元件名“ROM1”，再点击上方的上载按钮，将 FPGA 的中单片机中 ROM 的程序码读出显示（图 21-5）；然后用右键点击元件名“ROM1”，即弹出如图 21-5 所示下拉窗口，在此窗口中选择 Import Data from file 项，准备向 FPGA 加载新的程序。点击将出现文件浏览窗（图 21-6），选择刚才编译好的文件 KX\_240A.hex，双击，即将此文件调入计算机缓冲区；最后选中元件名“ROM1”，再点击上方的下载按钮，即刻将此文件代码通过 JTAG 口在系统载入 FPGA 中的 ROM 中，按一次复位键，即启动 FPGA 中的单片机运行此程序。

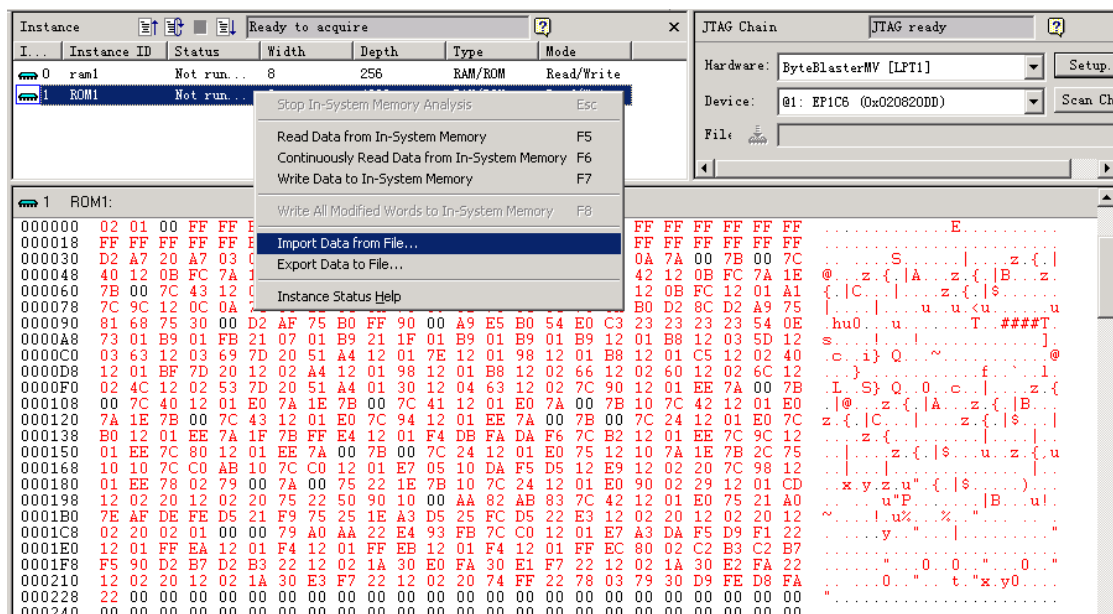


图 21-5 加入编译好的单片机程序，选择 Import Data from file

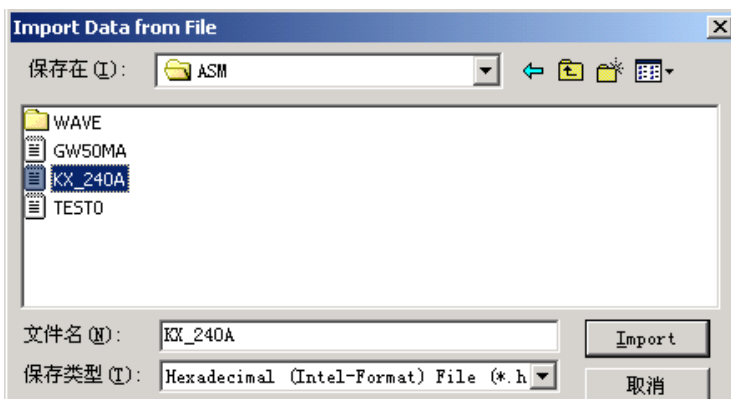


图 21-6 选择刚才编译好的文件 KX\_240A.hex

### 实验示例 2（运行一个简单的汇编程序）

使用以上介绍的方法下载一个简单程序，测试 P1 口向外锁存并在数码管上显示数据的功能；

此程序的路径是 \CMPUT\_EXPMT\Experiments\GWCP\_PK3\ASM\TEST0.hex。

仍然选择模式 3，复位仍为键 8，下载后要复位一次。此程序运行的结果是在数码管 1、2、3、4、5、6 上分别显示：A、B、C、D、E、F，并与实验系统的上排 8 和发光管显示 01010101（间隔发光）。

### 实验示例 3（启动等精度频率计功能）

这是一个软硬结合的设计和运行项目；使用以上介绍的方法下载等精度频率计程序；

此程序的路径是 \CMPUT\_EXPMT\Experiments\GWCP\_PK3\ASM\GW50MA.hex。

对于 GW48-CP+ 系统（GW48-PK2），\Experiments\GWCP\_PK2\ASM\GW50M.hex。

仍然选择模式 3，复位仍为键 8，下载后要复位一次。测试方法是：

用一短线接于适配板的中上方的一个插针（第 228 脚），作为被测频率输入口，将此线的另一端接在实验系统左下的某一频率信号端，如“4Hz”处，按键 1（测试频率选择键），就能看到数码管 6、5、4、3、2、1 上显示此测试频率，在某一数码管对应下方的发亮的发光管表示小数点在此数码管上，如 3.99999Hz；

频率显示的单位规律是：当小于 1MHz 频率，小数点处的值的单位是“Hz”；大于 1MHz 小于 10MHz，第 7 数码管下的发光管亮，这时，数码管 6 上的值的单位是“MHz”；大于 10MHz，有两个发光管发亮，左边发光管对应的数码管上显示的值的单位是“10MHz”。（测频率时，数码管 8/7 显示“E7”）

按复位键后，再按键 2，将在数码管 3、2、1 上显示占空比，如 49.9；数码管 8/7 显示“C7”

按复位键后，再按键 3，将在数码管 6、5、4、3、2、1 上显示脉宽，小数点处单位是微秒 us。数码管 8/7 显示“A7”。

## 6-1 基本 CPU 软硬件设计

本项实验是用单片 FPGA 实现一个基本完整的 8051 单片机：1、拥有标准 8051 完全兼容的指令系统的 CPU（图 6-1 中的 CPU\_Core）；2、256 字节内部 RAM；3、4K 字节程序 ROM；4、每一此编译下载后都能根据需要更新 ROM 中的程序，所以该单片机的实现和使用如同 89C51 一样方便。

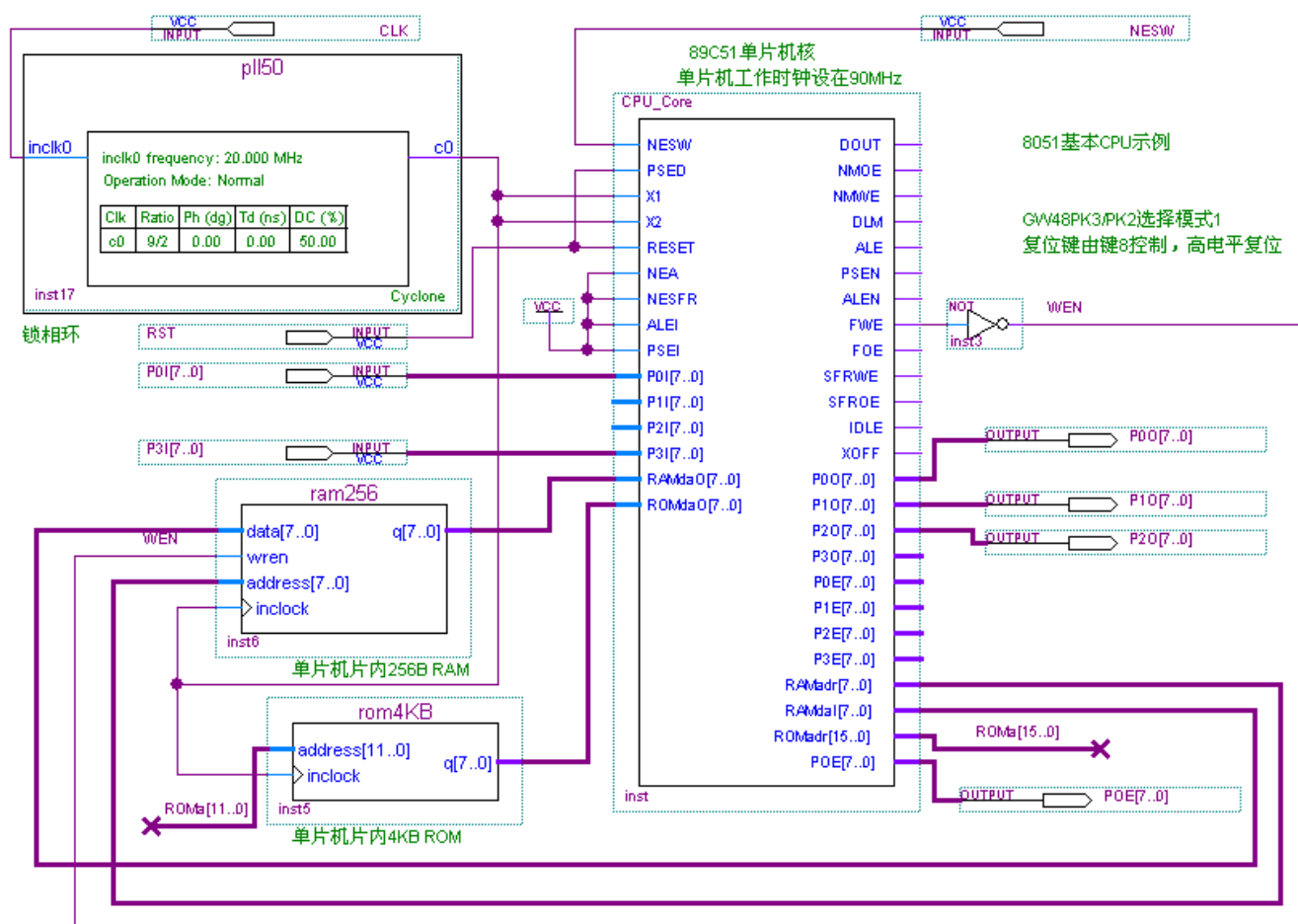


图 6-1 基本 8051CPU 核实验电路

此实验工程路径：/GW48\_PK/KX\_BASIC\_CPU/MCU8951，用 QuartusII 打开工程 MCU8951，其顶层原理图如图 6-1 所示。其中 CPU\_Core 是 8051 单片机核，由 VQM 网表文件源码构成，单片机时钟由锁相环提供，设在 90MHz 上，元件 RAM 和 ROM 都是 LPM 模块，前者设为 256 字节，后者设为 4KB，放置程序代码；P0I[7..0]、P1I[7..0]、P2I[7..0]、P3I[7..0]分别为 P0、P1、P2、P3 口的输入口；P0O[7..0]、P1O[7..0]、P2O[7..0]、P3O[7..0]分别为 P0、P1、P2、P3 口的输出口；双向口控制由 P0E[7..0]、P1E[7..0]、P2E[7..0]、P3E[7..0]负责。

本实验选择模式 1，示例程序路径和程序名是：./GW48\_PK/KX\_BASIC\_CPU/ASM/TEST1.ASM

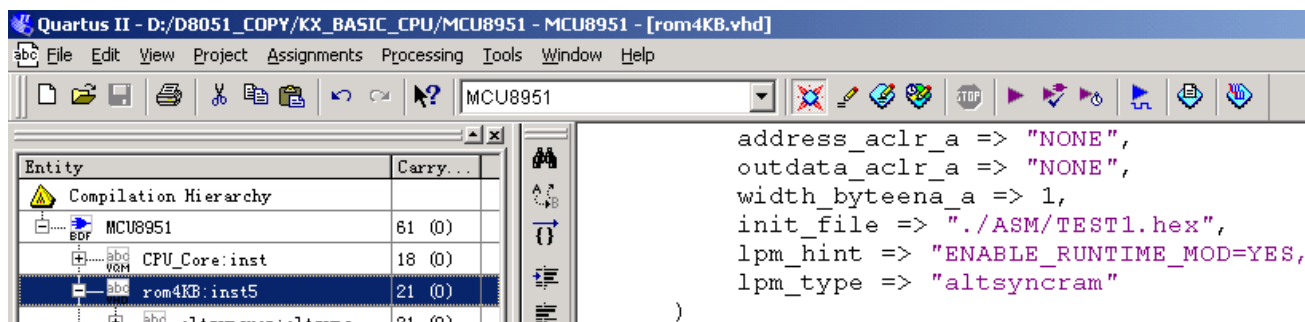


图 6-2 ROM 初始化文件路径

双击图 6-2 左侧 rom4KB，在右侧出现该元件文件，其初始化文件路径指示在 TEST1.HEX 上。

测试程序：键 2/键 1 输入 8 位 2 进制数，加上键 4/键 3 输入的 8 位 2 进制数，其和显示于数码管 8/7，P0/P3 定义为输入口，模式 1 情况下，P0 口的数据由键 2/键 1 输入；P3 口的数据由键 4/键 3 输入；P2/P1 定义为输出口，模式 1 情况下，P2 口的数据由数码 8/7 显示；P1 口的数据由数码 6/5 显示；P1 输出口锁定于数码 6/数码 5 上，演示加 1 指令。

注意，各端口作为输入/输出口是独立的，此例中，P0 口作为输入口，与 FPGA 的 PIO7-PIO0 相接，但同时作为输出口时，与 FPGA 的 PIO39-PIO33 相接，对应模式 1 的发光管 D8-D1，用于显示移位指令。

CLK 时钟输入口，频率选择 20MHz，RST 是复位信号输入端，RST=1 复位，RST=0 允许工作，由键 8 控制。

示例实验步骤如下：

1、设定实验系统为模式 1。用 QuartusII 打开工程文件：/ GW48\_PK /KX\_BASIC\_CPU/MCU8951 ,下载，

按复位键（键 8 0->1->0），键 2/1，键 4/3 输入 2 个 8 位 2 进制数相加，其和显示于数码 8/7。

对照汇编程序 test1.asm，观察单片机的功能。

2、修改汇编程序 test1.asm，编译，并用“Tools”菜单中的工具：In-System Memory Content Editor（图 6-4）下载编译代码：test1.hex。观察软硬件的工作情况。

3、在此基本电路平台上增加一些硬件模块，及与单片机的接口电路，再编辑对应的汇编软件，完成进一步的实验。

4、选择不同模式，和引脚锁定情况，协调软件与硬件设计，完成更大的设计实验。

5、编辑一段用于测试的汇编程序，利用时序仿真和逻辑分析仪，了解 8051 单片机的数据总线、指令总线、不同指令执行、地址总线、ALE、PSEN、个 IO 端口、不同指令对应下的端口方向控制信号 P0E、P1E、P2E、P3E 等信号间的时序情况，给出分析报告。

```
ORG 0000H
MAIN : MOV SP,#60H
      MOV 24H,#00H
      MOV 30H,#01H
ROUND: LCALL DELAY1
      MOV A,24H
      INC A
      MOV 24H,A
      MOV P1,A
      MOV A,30H
      RR A
      MOV P0,A
      MOV 30H,A
      NOP
      NOP
      MOV A,P0
      MOV B,P3
      ADD A,B
      MOV P2,A
      LCALL DELAY1
      SJMP ROUND
DELAY: MOV 20H,#0FFH
W1:    MOV 21H,#0FFH
W2:    DJNZ 21H,W2
      DJNZ 20H,W1
      RET
DELAY1: MOV 22H,#08H
W3:    LCALL DELAY
      DJNZ 22H,W3
      RET
END
```

图 6-3 TEST1.asm 汇编程序

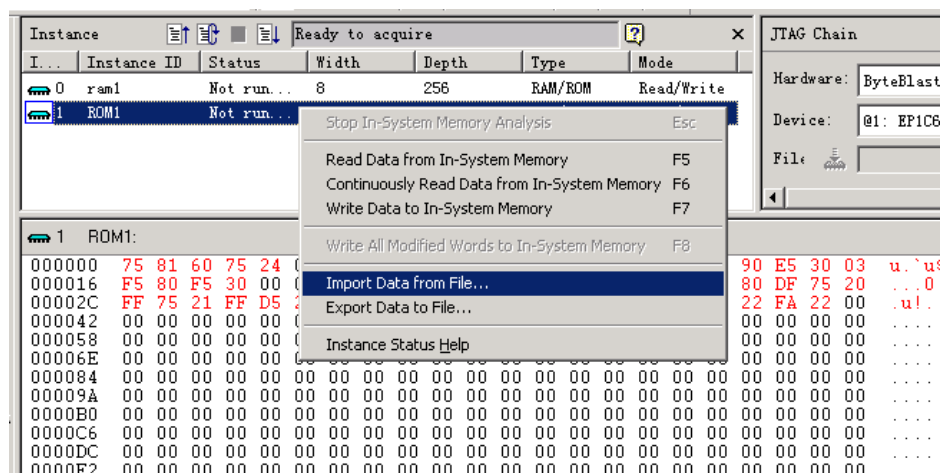


图 6-4 下载汇编程序

HEX 代码

## 6-2 8051/89C51 单片机核，等精度频率计与液晶显示实验

图 6-5 是一个含有等精度频率计测试模块的 8051 单片机系统。该模块由 VHDL 描述，单片机核由 VQM 描述。

此实验工程路径：/ GW48\_PK /GW48PK\_LCD/MCU8951 ,用 QuartusII 打开工程 MCU8951，其顶层原



理图如图 6-5 所示。单片机时钟设在 40MHz 上。

本实验选择模式 3，示例程序路径和程序名是：/ GW48\_PK /GW48PK\_LCD/ASM/PK\_LCD8K.ASM

图 6-5 图中，P1 和 P3 口的输出口控制 LCD，该液晶的用法可参考 5-2 节。图中的 P2 输出口锁定于数码 8/7；P3 口的输入口进入键控信号，由键 8 至键 1 给出。注意，将信号反相后进入单片机。

CLK 频率仍选择 20MHz，RST 复位，RST=1 复位，RST=0 允许工作，由键 9（锁于 IO27）控制。

示例实验步骤如下：

1、设定实验系统为模式 3。用 QuartusII 打开工程文件：/ GW48\_PK /GW48PK\_LCD/MCU8951 ,下载，按复位键（键 9 0->1->0），用一短线将适配板上方 P208 与 clock0 相接，作为等精度频率计信号输入口。

A、按复位键 9 后，按键 1，液晶将显示所测频率。

B、按复位键 9 后，按键 2，液晶将显示所测脉宽。

C、按复位键 9 后，按键 3，液晶将显示所测占空比。

D、按复位键 9 后，按键 8（若插含中文库液晶 12864JDLYY-GB），则显示中文演示内容。

E、按复位键 9 后，分别按键 8 至 4，数码管 8/7 将显示不同的数据，具体数据，可参考汇编程序：PK\_LCD8K.ASM。

2、设定实验系统为模式 3。用 QuartusII 打开工程文件：/ GW48\_PK /GWDVP\_M1/MCU8951 ,下载，可用 4X4 16 键盘控制，此时单片机汇编程序是 GWDVP\_M1.ASM。用一短线将适配板上方 P208 与 clock0 相接，作为等精度频率计信号输入口，接 2X16 字符液晶。插上 4X4 键盘后：

A、按复位键 9 后，按键 K34，液晶将显示所测频率；

B、按复位键 9 后，按键 K35，液晶将显示所测脉宽。

C、按复位键 9 后，按键 K36，液晶将显示所测占空比。

D、按复位键 9 后，按其他键，液晶将显示不同字符。

E、按复位键 9 后，按键 K37（若插含中文库液晶 12864JDLYY-GB），则显示中文演示内容。

3、测试基于 DDS 的移相信号发生器

接上高速 A/D，D/A 板，打开+/-12V 电源，接双踪示波器，

A、按复位键后，按键 K05/04，改变频率控制字，控制输出频率。

B、按复位键后，按键 K07/06，改变相位控制字，控制输出相位差。

修改基于 DDS 的移相信号发生器的功能，提高工作频率等。

4、修改 PK\_LCD8K.ASM，增加一些测试功能，编译、下载，调试软件。

5、基于图 6-5，修改硬件平台，比如 ROM 大小、模块 etester 的功能，增加其他接口模块，完成进一步软硬件调试实验。



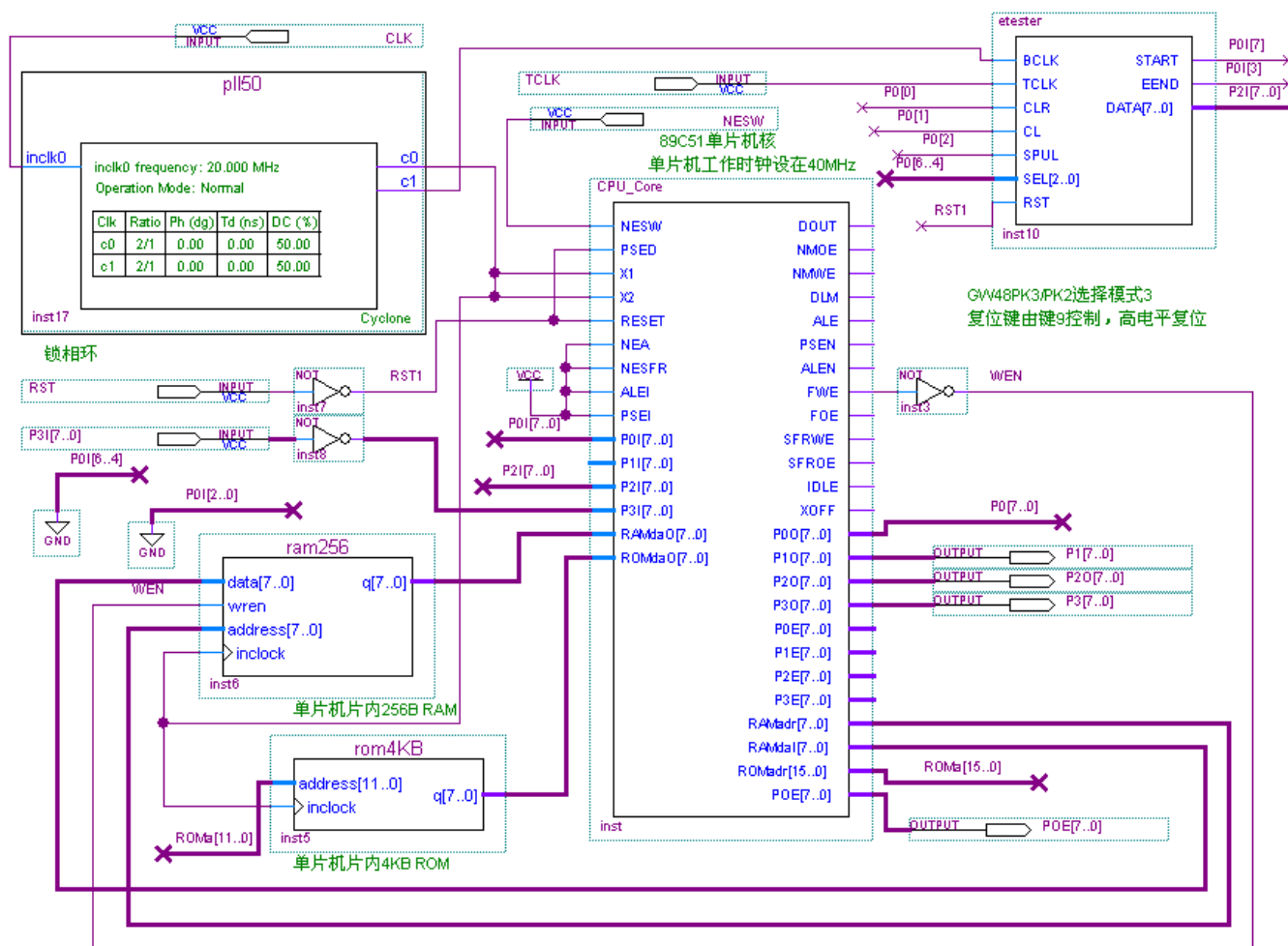


图 6-5 含有等精度频率计测试模块的 8051 单片机系统

### 6-3 8051/89C51 单片机核，等精度频率计与数码管显示实验

用 QuartusII 打开工程 MCU8951，图 6-6 是/D8051\_COPY/GW48PK\_LED/MCU8951 顶层原理图。也含有等精度频率测试模块，但用数码管显示测试结果。注意图中有一个 P3 口的双向口连接电路。

单片机时钟设在 60MHz 上，等精度频率计测试模块的标准频率设在 100MHz 上。

单片机示例程序路径和程序名是：/D8051\_COPY/GW48PK\_LED/ASM/GW50M2.hex，

本实验选择模式 3，用一短线将 P206 与 clock0 相接，作为等精度频率计信号输入口。

键 8 为复位键，键 1 控制频率测试、键 2 控制占空比测试、键 3 控制脉宽测试；

在测频率时，发光管 D8-D1 作为小数点，小于 10MHz 的频率，在对应的位置上一个发光管亮；小于 100MHz，大于 10MHz 的频率，在对应的位置上 2 个发光管亮；大于 100MHz 在对应的位置上 3 个发光管亮；

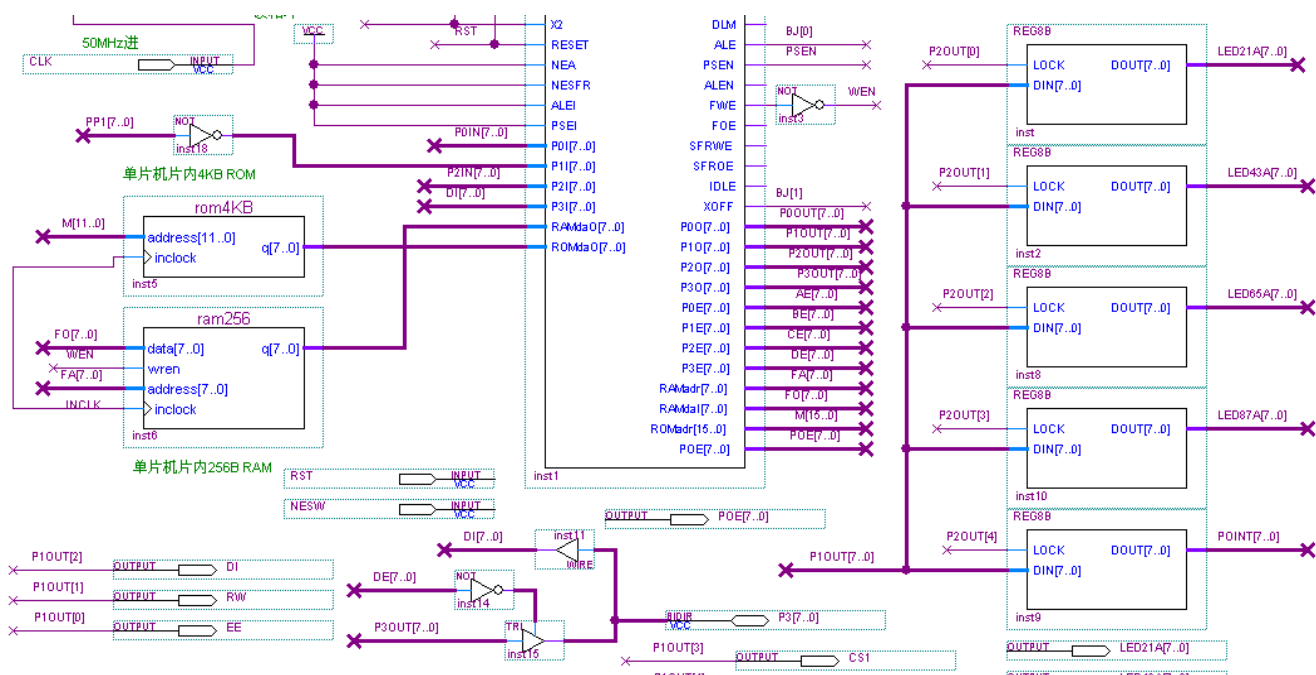


图 6-6

## 附录：GW48CP+主系统使用说明

### 第一节 GW48 教学实验系统原理与使用介绍

#### 一、GW48 系统使用注意事项（用户必读!!!）

a: 闲置不用 GW48 系统时，必须关闭电源!!!

b: 在实验中，当选中某种模式后，要按一下右侧的**复位键**，以使系统进入该结构模式工作。注意此复位键仅对实验系统的监控模块复位，而对目标器件 FPGA 没有影响，FPGA 本身没有复位的概念，上电后即工作，在没有配置前，FPGA 的 I/O 口是随机的，故可以从数码管上看到随机闪动，配置后的 I/O 口才会有确定的输出电平。

c: 换目标芯片时要特别注意，不要插反或插错，也不要带电插拔，确信插对后才能开电源。其它接口都可带电插拔。请特别注意，尽可能不要随意插拔适配板，及实验系统上的其他芯片。

#### 二、GW48 系统主板结构与使用方法

以下将详述 GW48 系列 SOPC/EDA 实验开发系统（GW48-PK2/CK）结构与使用方法，对于这 2 种型号的不同之处将给予单独指出。该系统的实验电路结构是可控的。即可通过控制接口键，使之改变连接方式以适应不同的实验需要。因而，从物理结构上看，实验板的电路结构是固定的，但其内部的信息流在主控器的控制下，电路结构将发生变化——重配置。这种“多任务重配置”设计方案的目的是有 3 个：1、适应更多的实验与开发项目；2、适应更多的 PLD 公司的器件；3、适应更多的不同封装的 FPGA 和 CPLD 器件。系统板面主要部件及其使用方法说明如下。以下是对 GW48 系统主板功能块的注释。

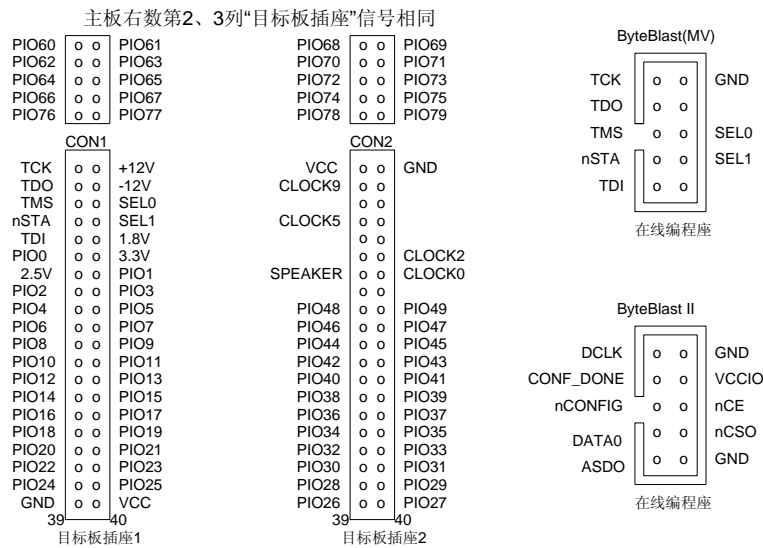
（1）“模式选择键”：按动该键能使实验板产生 12 种不同的实验电路结构。这些结构如第二节的 13 张实验电路结构图所示。例如选择了“NO. 3”图，须按动系统板上此键，直至数码管“模式指示”数码管显示“3”，于是系统即进入了 NO. 3 图所示的实验电路结构。

（2）适配板：这是一块插于主系统板上的目标芯片适配座。对于不同的目标芯片可配不同的适配座。可用的目标芯片包括目前世界上最大的六家 FPGA/CPLD 厂商几乎所有 CPLD、FPGA 和所有 ispPAC 等模拟 EDA 器件。第七节的表中已列出多种芯片对系统板引脚的对应关系，以利在实验时经常查用。

（3）ByteBlasterMV 编程配置口：如果要进行独立电子系统开发、应用系统开发、电子设计竞赛等开发实践活动，首先应该将系统板上的目标芯片适配座拔下（对于 Cyclone 器件不用拔），用配置的 10 芯编程

线将“ByteBlasterMV”口和独立系统上适配板上的 10 芯口相接，进行在系统编程（如 GWDVP-B 板），进行调试测试。“ByteBlasterMV”口能对不同公司，不同封装的 CPLD/FPGA 进行编程下载，也能对 isp 单片机 89S51 等进行编程。编程的目标芯片和引脚连线可参考附图 1，从而进行二次开发。

（4）ByteBlasterII 编程配置口：该口主要用于对 Cyclone 系列 AS 模式专用配置器件 EPCS4 和 EPCS1 等编程。



附表 1-1 在线编程各引脚与不同 PLD 公司器件编程下载接口说明

PLD 公司	LATTICE	ALTERA/ATMEL	XILINX	VANTIS
编程座	IsP LSI	CPLD	FPGA	CPLD
引脚				
TCK (1)	CLK	TCK	DCLK	TCK
TDO (3)	MODE	TDO	CONF_DONE	TDO
TMS (5)	ISPEN	TMS	nCONFIG	/PROGRAM
nSTA (7)	SEL0		nSTATUS	
TDI (9)	SDI	TDI	DATA0	TDI
SEL0	GND	VCC	VCC	GND
SEL1	GND	VCC	VCC	GND

附图 1 GW48 系统电子设计二次开发信号图

（5）混合工作电压源：系统不必通过切换即可为 CPLD/FPGA 目标器件提供 5V、3.3V、2.5V、1.8V 和 1.5V 工作电源，此电源位置可参考附图 1。

（6）JP5 编程模式选择跳线：（仅 GW48-PK2 型含此）。如果要对 Cyclone 的配置芯片进行编程，应该将跳线接于“ByBtII”端，在将标有“ByteBlasterII”编程配置口同适配板上 EPCS4/1 的 AS 模式下载口用 10 芯线连接起来，通过 QuartusII 进行编程。当短路“Others”端时，可对其它所有器件编程，端口信号参考附图 1。

（7）JP6/JVCC/vs2 编程电压选择跳线：跳线 JVCC（GW48-PK2 型标为“JP6”）是对编程下载口的选择跳线。对 5V 器件，如 10K10、10K20、7128S、1032、95108、89S51 单片机等，必须选“5.0V”。而对低于或等于 3.3V 的低压器件，如 1K30、1K100、10K30E、20K300、Cyclone、7128B 等一律选择“3.3V”一端。

（8）并行下载口：此接口通过下载线与微机的打印机口相连。来自 PC 机的下载控制信号和 CPLD/FPGA 的目标码将通过此口，完成对目标芯片的编程下载。计算机的并行口通信模式最好设置成“EPP”模式。

（9）键 1~键 8：为实验信号控制键，此 8 个键受“多任务重配置”电路控制，它在每一张电路图的功能及其与主系统的连接方式随模式选择键的选定的模式而变，使用中需参照第二节中的电路图。

（10）键 9~键 14：（GW48-PK2 型含此键）此 6 个键不受“多任务重配置”电路控制，由于键信号速度慢，所以其键信号输入口是全开放的，各端口定义在插座“JP8”处，可通过手动节插线的方式来实用，键输出默认高电平。

注意，键 1 至键 8 是由“多任务重配置”电路结构控制的，所以键的输出信号没有抖动问题，不需要在目标芯片的电路设计中加入消抖动电路，这样，能简化设计，迅速入门。但设计者如果希望完成键的消抖动电路设计练习，必须使用键 9 至键 14 来实现。

(11) **数码管 1~8/发光管 D1~D16**：受“多任务重配置”电路控制，它们的连线形式也需参照第二节的电路图。

(12) **“时钟频率选择”**：位于主系统的右小侧，通过短路帽的不同接插方式，使目标芯片获得不同的时钟频率信号。

对于“CLOCK0”，同时只能插一个短路帽，以便选择输向“CLOCK0”的一种频率：信号频率范围：0.5Hz~50MHz。由于 CLOCK0 可选的频率比较多，所以比较适合于目标芯片对信号频率或周期测量等设计项目的信号输入端。右侧座分三个频率源组，它们分别对应三组时钟输入端：**CLOCK2、CLOCK5、CLOCK9**。例如，将三个短路帽分别插于对应座的 2Hz、1024Hz 和 12MHz，则 CLOCK2、CLOCK5、CLOCK9 分别获得上述三个信号频率。需要特别注意的是，每一组频率源及其对应时钟输入端，分别只能插一个短路帽。也就是说最多只能提供 4 个时钟频率输入 FPGA：CLOCK0、CLOCK2、CLOCK5、CLOCK9。

(13) **扬声器**：与目标芯片的“SPEAKER”端相接，通过此口可以进行奏乐或了解信号的频率，它与目标器件的具体引脚号，应该查阅附录第 3 节的表格。

(14) **PS/2 接口**：通过此接口，可以将 PC 机的键盘和/或鼠标与 GW48 系统的目标芯片相连，从而完成 PS/2 通信与控制方面的接口实验，GW48-GK/PK2 含另一 PS/2 接口，引脚连接情况参见实验电路结构 NO. 5（附图 7）。

(15) **VGA 视频接口**：通过它可完成目标芯片对 VGA 显示器的控制。详细连接方式参考附图 7（对 GW48-PK2 主系统），或附图 13（GW48-CK 主系统）。

(16) **单片机接口器件**：它与目标板的连接方式也已标于主系统板上：连接方式可参见附图 11。

注 1、对于 GW48-PK2 系统，实验板右侧有一开关，若向“TO\_FPGA”拨，将 RS232 通信口直接与 FPGA 相接；若向“TO\_MCU”拨，则与 89S51 单片机的 P30 和 P31 端口相接。于是通过此开关可以进行不同的通信实验，详细连接方式可参见附图 11。平时此开关应该向“TO\_MCU”拨，这样可不影响 FPGA 的工作！

注 2、GW48-EK 系统上的用户单片机 89C51 的各引脚是独立的（时钟已接 12MHz），没有和其他任何电路相连，实验时必须使用连接线连接，例如，若希望 89C51 通过实验板右侧的 RS232 口与 PC 机进行串行通信，必须将此单片机旁的 40 针座（此座上每一脚恰好与 89C51 的对应脚相接）上的 P30、P31 分别与右侧的 TX30、RX30 相接。

(17) **RS-232 串行通讯接口**：此接口电路是为 FPGA 与 PC 通讯和 SOPC 调试准备的。或使 PC 机、单片机、FPGA/CPLD 三者实现双向通信。对于 GW48-EK 系统，其通信端口是与中间的双排插座上的 TX30、RX31 相连的。详细连接方式参考附图 11。

(18) **“AOUT” D/A 转换**：利用此电路模块（实验板左下侧），可以完成 FPGA/CPLD 目标芯片与 D/A 转换器的接口实验或相应的开发。它们之间的连接方式可参阅附图 7（实验电路结构 NO. 5）：D/A 的模拟信号的输出接口是“AOUT”，示波器可挂接左下角的两个连接端。当使能拨码开关 8：“滤波 1”时，D/A 的模拟输出将获得不同程度的滤波效果。

注意，进行 D/A 接口实验时，需打开系统上侧的+/-12V 电源开关（实验结束后关上此电源！）。

(19) **“AIN0” / “AIN1”**：外界模拟信号可以分别通过系统板左下侧的两个输入端“AIN0”和“AIN1”进入 A/D 转换器 ADC0809 的输入通道 IN0 和 IN1，ADC0809 与目标芯片直接相连。通过适当设计，目标芯片可以完成对 ADC0809 的工作方式确定、输入端口选择、数据采集与处理等所有控制工作，并可通过系统板提供的译码显示电路，将测得的结果显示出来。此项实验首先需参阅第二节的“实验电路结构 NO. 5”有关 0809 与目标芯片的接口方式，同时了解系统板上的接插方法以及有关 0809 工作时序和引脚信号功能方面的资料。

**注意**：不用 0809 时，需将左下角的拨码开关的“A/D 使能”和“转换结束”打为禁止：向上拨，以避免与其他电路冲突。ADC0809 A/D 转换实验接插方法（如，附图 7，实验电路结构 NO. 5 图所示）：

1. 左下角拨码开关的“A/D 使能”和“转换结束”拨为使能：向下拨，即将 ENABLE(9)与 PI035 相接；若向上拨则禁止，即则使 ENABLE(9) ← 0，表示禁止 0809 工作，使它的所有输出端为高阻态。

2. 左下角拨码开关的“转换结束”使能，则使 EOC(7) ← PI036，由此可使 FPGA 对 ADC0809 的转换状态

进行测控。

(20) **VR1/“AIN1”**: VR1 电位器, 通过它可以产生 0V~+5V 幅度可调的电压。其输入口是 0809 的 IN1 (与外接口 AIN1 相连, 但当 AIN1 插入外输入插头时, VR1 将与 IN1 自动断开)。若利用 VR1 产生被测电压, 则需使 0809 的第 25 脚置高电平, 即选择 IN1 通道, 参考“实验电路结构 NO.5”。

(21) **AIN0 的特殊用法**: 系统板上设置了一个比较器电路, 主要以 LM311 组成。若与 D/A 电路相结合, 可以将目标器件设计成逐次比较型 A/D 变换器的控制器件参考“实验电路结构 NO.5”。

(22) **系统复位键**: 此键是系统板上负责监控的微处理器的复位控制键, 同时也与接口单片机和 LCD 控制单片机的复位端相连。因此兼作单片机的复位键。

(23) **下载控制开关**: (仅 GW48-GK/PK 型含此开关) 在系统板的左侧的开关。当需要对实验板上的目标芯片下载时必须将开关向上打 (即“DLOAD”); 而当向下打 (LOCK) 时, 将关闭下载口, 这时可以将下载并行线拔下而作它用 (这时已经下载进 FPGA 的文件不会由于下载口线的电平变动而丢失); 例如拔下的 25 芯下载线可以与其他适配板上的并行接口相接, 以完成类似逻辑分析仪方面的并行通信实验。

(24) **跳线座 SPS**: 短接“T\_F”可以使用“在系统频率计”。频率输入端在主板右侧标有“频率计”处。模式选择为“A”。短接“PI048”时, 信号 PI048 可用, 如实验电路结构图 NO.1 中的 PI048。平时应该短路“PI048”。

(25) **目标芯片万能适配座 CON1/2**: 在目标板的下方有两条 80 个插针插座 (GW48-CK 系统), 其连接信号如附图 1 所示, 此图为用户对此实验开发系统作二次开发提供了条件。

对于 GW48-GK/PK2/EK 系统, 此适配座在原来的基础上增加了 20 个插针, 功能大为增强。增加的 20 个插针信号与目标芯片的连接方式可参考“实验电路结构 NO.5”、附图 11 和第 3 节表格。GW48-EK 系统中此 20 个插针信号全开放。

(26) **左下拨码开关**: (仅 GK/PK2/EK 型含此开关) 拨码开关的详细用法可参考实验电路结构 NO.5 图 (附图 7)。

(30) **+/-12V 电源开关**: 在实验板左上角。有指示灯。电源提供对象: 1) 与 082、311 及 DAC0832 等相关的实验; 2) 模拟信号发生源; 3) GW48-DSP/DSP+适配板上的 D/A 及参考电源; 此电源输出口可参见附图 1。平时, 此电源必须关闭!

(31) **智能逻辑笔**: (仅 GK/PK2 型含此) 逻辑信号由实验板左侧的“LOGIC PEN INPUT”输入。测试结果:

A) **“高电平”**: 判定为大于 3V 的电压; 亮第 1 个发光管; B) **“低电平”**: 判定为小于 1V 的电压; 亮第 2 个发光管。

C) **“高阻态”**: 判定为输入阻抗大于 100K 欧姆的输出信号; 亮第 3 个发光管。注意, 此功能具有智能化;

D) **“中电平”**: 判定为小于 3V, 大于 1V 的电压; 亮第 4 个发光管。E) **“脉冲信号”**: 判定为存在脉冲信号时; 亮所有的发光管。(注意, 使用逻辑笔时, clock0/clock9 上不要接 50MHz, 以免干扰)。

(30) **模拟信号发生源**: (GK/PK2 型含此) 信号源主要用于 DSP/SOPC 实验及 A/D 高速采样用信号源。使用方法如下:

1) 打开 +/-12V 电源; 2) 用一插线将右下角的某一频率信号 (如 65536Hz) 连向单片机上方插座“JP18”的 INPUT 端; 3) 这时在“JP17”的 OUTPUT 端及信号挂钩“WAVE OUT”端同时输出模拟信号, 可用示波器显示输出模拟信号 (这时输出的频率也是 65536Hz); 4) 实验系统右侧的电位器上方的 3 针座控制输出是否加入滤波: 向左端短路加滤波电容; 向右短路断开滤波电容; 5) 此电位器是调谐输出幅度的, 应该将输出幅度控制在 0-5V 内。

(32) **JP13 选择 VGA 输出**: (仅 GW48-GK/PK2 含此)。将“ENBL”短路, 使 VGA 输出显示使能; 将“HIBT”短路, 使 VGA 输出显示禁止, 这时可以将来自外部的 VGA 显示信号通过 JP12 座由 VGA 口输出。此功能留给 SOPC 开发。

(33) **FPGA 与 LCD 连接方式**: (仅 PK2 型含此)。由附图 11 的实验电路结构图 COM 可知, 默认情况下, FPGA 是通过 89C51 单片机控制 LCD 液晶显示的, 但若 FPGA 中有 Nios 嵌入式系统, 则能使 FPGA 直接控制 LCD 显示。方法是拔去此单片机 (在右下侧), 用连线将座 JP22/JP21 (LCD 显示器引脚信号) 各信号分别与座 JP19/JP20 (FPGA 引脚信号) 相连接即可。针对目标器件的型号, 查表锁定引脚后, 参考.\gwdvpb\H128X64 液晶显示使用说明.doc 即可。

(34) **JP23 使用说明:** (仅 GW48-GK/PK2 型含此)。单排座 JP23 有 3 个信号端, 分别来自此单片机的 I/O 口。

(35) **使用举例:** 若模式键选中了“实验电路结构图 NO.1”, 这时的 GW48 系统板所具有的接口方式变为: FPGA/CPLD 端口 PI/031~28 (即 PI/031、PI/030、PI/029、PI/028)、PI/027~24、PI/023~20 和 PI/019~16, 共 4 组 4 位二进制 I/O 端口分别通过一个全译码型 7 段译码器输向系统板的 7 段数码管。这样, 如果有数据从上述任一组四位输出, 就能在数码管上显示出相应的数值, 其数值对应范围为:

FPGA/CPLD 输出	0000	0001	0010	...	1100	1101	1110	1111
数 码 管 显 示	0	1	2	...	C	D	E	F

端口 I/032~39 分别与 8 个发光二极管 D8~D1 相连, 可作输出显示, 高电平亮。还可分别通过键 8 和键 7, 发出高低电平输出信号进入端口 I/049 和 48; 键控输出的高低电平由键前方的发光二极管 D16 和 D15 显示, 高电平输出为亮。此外, 可通过按动键 4 至键 1, 分别向 FPGA/CPLD 的 PI00~PI015 输入 4 位 16 进制码。每按一次键将递增 1, 其序列为 1, 2, ...9, A, ...F。注意, 对于不同的目标芯片, 其引脚的 I/O 标号数一般是同 GW48 系统接口电路的“PI0”标号是一致的 (这就是引脚标准化), 但具体引脚号是不同的, 而在逻辑设计中引脚的锁定数必须是该芯片的具体的引脚号。具体对应情况需要参考第 3 节的引脚对照表。

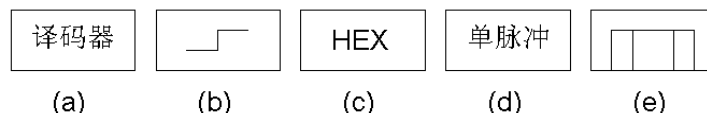
## 第二节 实验电路结构图

### 1. 实验电路信号资源符号图说明

结合附图 2-1, 以下对实验电路结构图中出现的信号资源符号功能作出一些说明:

(1) 附图 2-1a 是 16 进制 7 段全译码器, 它有 7 位输出, 分别接 7 段数码管的 7 个显示输入端: a、b、c、d、e、f 和 g; 它的输入端为 D、C、B、A, D 为最高位, A 为最低位。例如, 若所标输入的口线为 PI019~16, 表示 PI019 接 D、18 接 C、17 接 B、16 接 A。

(2) 附图 2-1b 是高低电平发生器, 每按键一次, 输出电平由高到低、或由低到高变化一次, 且输出为高电平时, 所按键对应的发光管变亮, 反之不亮。



附图 2A 实验电路信号资源符号图

(3) 附图 2A-1c 是 16 进制码 (8421 码) 发生器, 由对应的键控制输出 4 位 2 进制构成的 1 位 16 进制码, 数的范围是 0000~1111, 即<sup>^</sup>H0 至<sup>^</sup>HF。每按键一次, 输出递增 1, 输出进入目标芯片的 4 位 2 进制数将显示在该键对应的数码管上。

(4) 直接与 7 段数码管相连的连接方式的设置是为了便于对 7 段显示译码器的设计学习。以图 NO.2 为例, 如图所标“PI046-PI040 接 g、f、e、d、c、b、a”表示 PI046、PI045..PI040 分别与数码管的 7 段输入 g、f、e、d、c、b、a 相接。

(5) 附图 2-1d 是单次脉冲发生器。每按一次键, 输出一个脉冲, 与此键对应的发光管也会闪亮一次, 时间 20ms。

(6) 附图 2-1e 是琴键式信号发生器, 当按下键时, 输出为高电平, 对应的发光管发亮; 当松开键时, 输出为低电平, 此键的功能可用于手动控制脉冲的宽度。具有琴键式信号发生器的实验结构图是 NO.3。

### 2. 各实验电路结构图特点与适用范围简述

(1) **结构图 NO.0:** 目标芯片的 PI019 至 PI044 共 8 组 4 位 2 进制码输出, 经外部的 7 段译码器可显示于实验系统上的 8 个数码管。键 1 和键 2 可分别输出 2 个四位 2 进制码。一方面这四位码输入目标芯片的 PI011~PI08 和 PI015~PI012, 另一方面, 可以观察发光管 D1 至 D8 来了解输入的数值。例如, 当键 1 控制输入 PI011~PI08 的数为<sup>^</sup>HA 时, 则发光管 D4 和 D2 亮, D3 和 D1 灭。电路的键 8 至键 3 分别控制一个高低电平信号发生器向目标芯片的 PI07 至 PI02 输入高电平或低电平, 扬声器接在“SPEAKER”上, 具体接在哪一引脚要看目标芯片的类型, 这需要查第 3 节的引脚对照表。如目标芯片为 FLEX10K10, 则扬声器接在“3”引脚上。目标芯片的时钟输入未在图上标出, 也需查阅第 3 节的引脚对照表。例如, 目标芯片为 XC95108, 则输入此芯片的时钟信号有 CLOCK0 至 CLOCK9, 共 4 个可选的输入端, 对应的引脚为 65 至 80。具体的输入频率, 可参考主板频率选择模块。此电路可用于设计频率计, 周期计, 计数器等等。

(2) **结构图 NO.1:** 适用于作加法器、减法器、比较器或乘法器等。例如, 加法器设计, 可利用键 4 和键 3 输入 8 位加数; 键 2 和键 1 输入 8 位被加数, 输入的加数和被加数将显示于键对应的数码管 4-1, 相加的



和显示于数码管 6 和 5；可令键 8 控制此加法器的最低位进位。

(3) **结构图 NO. 2:** 可用于作 VGA 视频接口逻辑设计，或使用数码管 8 至数码管 5 共 4 个数码管作 7 段显示译码方面的实验；而数码管 4 至数码管 1，4 个数码管可作译码后显示，键 1 和键 2 可输入高低电平。

(4) **结构图 NO. 3:** 特点是有 8 个琴键式键控发生器，可用于设计八音琴等电路系统。也可以产生时间长度可控的单次脉冲。该电路结构同结构图 NO. 0 一样，有 8 个译码输出显示的数码管，以显示目标芯片的 32 位输出信号，且 8 个发光管也能显示目标器件的 8 位输出信号。

(5) **结构图 NO. 4:** 适合于设计移位寄存器、环形计数器等。电路特点是，当在所设计的逻辑中有串行 2 进制数从 PIO10 输出时，若利用键 7 作为串行输出时钟信号，则 PIO10 的串行输出数码可以在发光管 D8 至 D1 上逐位显示出来，这能很直观地看到串出的数值。

(6) **结构图 NO. 5:** 此电路结构有较强的功能，主要用于目标器件与外界电路的接口设计实验。主要含以下 9 大模块：

1. **普通内部逻辑设计模块。**在图的左下角。此模块与以上几个电路使用方法相同，例如同结构图 NO. 3 的唯一区别是 8 个键控信号不再是琴键式电平输出，而是高低电平方式向目标芯片输入。此电路结构可完成许多常规的实验项目。

2. **RAM/ROM 接口。**在图左上角，此接口对应于主板上，有 1 个 32 脚的 DIP 座，在上面可以插 RAM，也可插 ROM（仅 GW48-GK/PK 系统包含此接口）例如：RAM：628128；ROM：27C020、27C040、29C040 等。此 32 脚座的各引脚与目标器件的连接方式示于图上，是用标准引脚名标注的，如 PIO48（第 1 脚）、PIO10（第 2 脚）、OE 控制为 PIO62 等等。注意，RAM/ROM 的使能 CS1 由主系统左边的拨码开关“1”控制。对于不同的 RAM 或 ROM，其各引脚的功能定义不尽一致，即，不一定兼容，因此在使用前应该查阅相关的资料，但在结构图的上方也列出了部分引脚情况，以资参考。

3. **VGA 视频接口。** 4. **两个 PS/2 键盘接口。**注意，对于 GW48-CK 系统，只有 1 个，连接方式是下方的 PS/2 口。

5. **A/D 转换接口。** 6. **D/A 转换接口。** 7. **LM311 接口。** 8. **单片机接口。** 9. **RS232 通信接口。**注意，**结构图 NO. 5** 中并不是所有电路模块都可以同时使用，这是因为各模块与目标器件的 IO 接口有重合：

1. 当使用 RAM/ROM 时，数码管 3、4、5、6、7、8 共 6 各数码管不能同时使用，这时，如果有必要使用更多的显示，必须使用以下介绍的扫描显示电路。但 RAM/ROM 可以与 D/A 转换同时使用，尽管他们的数据口（PIO24、25、26、27、28、29、30、31）是重合的。这时如果希望将 RAM/ROM 中的数据输入 D/A 中，可设定目标器件的 PIO24、25、26、27、28、29、30、31 端口为高阻态；而如果希望用目标器件 FPGA 直接控制 D/A 器件，可通过拨码开关禁止 RAM/ROM 数据口。

RAM/ROM 能与 VGA 同时使用，但不能与 PS/2 同时使用，这时可以使用以下介绍的 PS/2 接口。

2. A/D 不能与 RAM/ROM 同时使用，由于他们有部分端口重合，若使用 RAM/ROM，必须禁止 ADC0809，而当使用 ADC0809 时，应该禁止 RAM/ROM，如果希望 A/D 和 RAM/ROM 同时使用以实现诸如高速采样方面的功能，必须使用含有高速 A/D 器件的适配板，如 GWAK30+等型号的适配板。RAM/ROM 不能与 311 同时使用，因为在端口 PIO37 上，两者重合。

(7) **结构图 NO. 6:** 此电路与 NO. 2 相似，但增加了两个 4 位 2 进制数发生器，数值分别输入目标芯片的 PIO7~PIO4 和 PIO3~PIO0。例如，当按键 2 时，输入 PIO7~PIO4 的数值将显示于对应的数码管 2，以便了解输入的数值。

(8) **结构图 NO. 7:** 此电路适合于设计时钟、定时器、秒表等。因为可利用键 8 和键 5 分别控制时钟的清零和设置时间的使能；利用键 7、5 和 1 进行时、分、秒的设置。

(9) **结构图 NO. 8:** 此电路适用于作并进/串出或串进/并出等工作方式的寄存器、序列检测器、密码锁等逻辑设计。它的特点是利用键 2、键 1 能序置 8 位 2 进制数，而键 6 能发出串行输入脉冲，每按键一次，即发一个单脉冲，则此 8 位序置数的高位在前，向 PIO10 串行输入一位，同时能从 D8 至 D1 的发光管上看到串行左移的数据，十分形象直观。

(10) **结构图 NO. 9:** 若欲验证交通灯控制等类似的逻辑电路，可选此电路结构。

(11) 当系统上的“模式指示”数码管显示“A”时，系统将变成一台频率计，数码管 8 将显示“F”，“数码 6”至“数码 1”显示频率值，最低位单位是 Hz。测频输入端为系统板右下侧的插座。



(13) **实验电路结构图 COM:** 附图 11 电路仅 GW48-GK/PK2 拥有, 即以上所述的所有电路结构, 包括“实验电路结构 NO.0”至“实验电路结构 NO.B”共 11 套电路结构模式为 GW48-GK/PK2 两种系统共同拥有(兼容), 把他们称为通用电路结构。即在原来的 11 套电路结构模式中的每一套结构图中增加附图 11 所示的“实验电路结构图 COM”。例如, 在 GW48-PK2 系统中, 当“模式键”选择“5”时, 电路结构将进入附图 7 所示的实验电路结构图 NO.5 外, 还应该加入“实验电路结构图 COM”。这样, 在每一电路模式中就能比原来实现更多的实验项目。

实验电路结构图 COM”中各标准信号(PIOX)对应的器件的引脚名, 必须查第七节的表。

附图 11 实验电路结构图 COM (GW48-PK2 上液晶与单片机以及 FPGA 的 I/O 口的连接方式, Cyclone 和 20K 系列器件通用。)

2、GWDVPB 板与 GW48 系统上的目标芯片板相互间完全兼容, 因此可以使用 GW48 系统所有可配的目标芯片, 所以在利用 GWDVPB 板开发时, 就没有了在竞赛中发生逻辑资源不够用的担心, 也没有对使用 FPGA/CPLD 型号和生产厂家的限制!

## (2)、单片机子程序使用方法:

配附的光盘中的“GWDVPB”目录中有与 GWDVPB 系统相配的单片机子程序(GWDVPB.ASM, GWDVPA.ASM), 此文件中有加法、减法、乘法、除法、开方、数码显示、键盘控制、BCD 码到 2 进制码转换、2 进制码到 BCD 码转换等子程序以及设计示例等。注意, 程序中单片机的堆栈设在“60H”, 方法可参考“GWDVPB.ASM”。方法如下:

1、显示子程序: 程序名“DIRR0”, 使用方法见(5、)。 2、数码管熄灭子程序: 程序名“NL0”。

3、键盘子程序: 程序名“KKEYI”, 每调用一次, 则对 P1 口上的 8 个键扫描检测一次, 如果无按键信号, 将不跳出此子程序, 直到测到有按键信号为止, 返回的数据在 ACC 中, ACC 中的数与 P1 口的某一端口序号一样, 如 ACC=3, 即表示, P1.3 上有按键信号。

4、无符号加法子程序: 程序名“ADDMB”。

5、单片机测频率子程序: 程序名“PROSD”, 每调用一次, 即对单片机 P3.5 口上的信号频率测试一次(测频范围: 1Hz—500KHz), 测出的频率显示在数码管上, 最低一位(左第 7 个)的单位是 Hz。

6、除法子程序: 程序名“DIVD1”, 即 2N 字节(2 进制)的数除以 1N 字节的数, 被除数字节数放在 30H 单元中; 除数字节数放在 31H 单元中。例如 N=3, 则 6 个字节的被除数分别放在(4AH, 4BH, 4CH, 4DH, 4EH, 4FH)单元中, 3 字节除数放在(5DH, 5EH, 5FH)单元中, 计算后的商放在(4DH, 4EH, 4FH)单元中, 高位都放在左面。

7、N 字节乘 M 字节乘法子程序: 程序名“MULNM”。 8、2 进制码至 BCD 码转换子程序: 程序名“HEXBCD2”。

9、快速乘法子程序: 程序名“MULT3”。 10、N 字节压缩 BCD 码至 M 字节 2 进制码转换子程序: 程序名“BCDHEX1”。

11、带符号原码加法子程序: 程序名“ADDS1”。 12、开方子程序: 程序名“SQR1”

13、串行 EEPROM93C46 读数子程序: 程序名“SEPRD”。16 位读数方式, 先将数据地址(00H-3FH)放在 22H 单元中, 再调用子程序“LCALL SEPRD”, 读出的数放在 21H 和 20H 单元中, 高位在前。注意, 使用 93C46 要根据板上的信号标注进行连线。

14、串行 EEPROM93C46 写数子程序: 写数顺序, 先调用写允许子程序“LCALL EABLE”, 再将 8 位地址数 XXH、待写入的高 8 位字节和低 8 位字节分别放在 22H、21H 和 20H 单元中, 然后调用写子程序“LCALL SEPWR”, 最后调用写禁止子程序“LCALL DISLE”。15、如图 9-1 所示, 时钟 50MHz 接 CLOCK2、时钟 12MHz 接 CLOCK0。

16、**显示:** 显示系统由 7 个 74LS164 构成串行静态显示电路, 此显示系统的数据口接单片机的 P3.0、CLOCK 接 P3.1; 显示子程序名为“DIRR0”, 7 个数码管的显示缓冲寄存器依次为: 16H、15H、14H、13H、12H、11H、10H, 小数点的缓冲寄存器是 0AH, 其中某一位为 1 时, 对应的数码管的小数点点亮。

(11) 板的下放有一 3 针跳线, 往左短路, 禁止使用 ROM/RAM, 反之允许使用。

## 二、GWDVP-B 板使用注意:

(1)、**兼容性:** 开发中, GWDVPB 板须与 GW48 系统或 GW6C++编程器配合使用, 这表现在:

1、须利用 GW48 提供的 10 芯在系统下载接口和通信线进行编程下载;

2、GWDVPB 板与 GW48 系统上的目标芯片板相互间完全兼容，因此可以使用 GW48 系统所有可配的目标芯片，所以在利用 GWDVPB 板开发时，就没有了在竞赛中发生逻辑资源不够用的担心，也没有对使用 FPGA/CPLD 型号和生产厂家的限制！

### (2)、单片机子程序使用方法：

配附的光盘中的“GWDVPB”目录中有与 GWDVPB 系统相配的单片机子程序（GWDVPB.ASM，GWDVPA.ASM），此文件中有加法、减法、乘法、除法、开方、数码显示、键盘控制、BCD 码到 2 进制码转换、2 进制码到 BCD 码转换等子程序以及设计示例等。注意，程序中单片机的堆栈设在“60H”，方法可参考“GWDVPB.ASM”。方法如下：

1、显示子程序：程序名“DIRR0”，使用方法见（5、）。 2、数码管熄灭子程序：程序名“NL0”。

3、键盘子程序：程序名“KKEY1”，每调用一次，则对 P1 口上的 8 个键扫描检测一次，如果无按键信号，将不跳出此子程序，直到测到有按键信号为止，返回的数据在 ACC 中，ACC 中的数与 P1 口的某端口序号一样，如 ACC=3，即表示，P1.3 上有按键信号。

4、无符号加法子程序：程序名“ADDMB”。

5、单片机测频率子程序：程序名“PROSD”，每调用一次，即对单片机 P3.5 口上的信号频率测试一次（测频范围：1Hz—500KHz），测出的频率显示在数码管上，最低一位（左第 7 个）的单位是 Hz。

6、除法子程序：程序名“DIVD1”，即 2N 字节（2 进制）的数除以 1N 字节的数，被除数字节数放在 30H 单元中；除数字节数放在 31H 单元中。例如 N=3，则 6 个字节的被除数分别放在（4AH，4BH，4CH，4DH，4EH，4FH）单元中，3 字节除数放在（5DH，5EH，5FH）单元中，计算后的商放在（4DH，4EH，4FH）单元中，高位都放在左面。

7、N 字节乘 M 字节乘法子程序：程序名“MULNM”。 8、2 进制码至 BCD 码转换子程序：程序名“HEXBCD2”。

9、快速乘法子程序：程序名“MULT3”。 10、N 字节压缩 BCD 码至 M 字节 2 进制码转换子程序：程序名“BCDHEX1”。

11、带符号原码加法子程序：程序名“ADDS1”。 12、开方子程序：程序名“SQR1”

13、串行 EEPROM93C46 读数子程序：程序名“SEPRD”。16 位读数方式，先将数据地址（00H-3FH）放在 22H 单元中，再调用子程序“LCALL SEPRD”，读出的数放在 21H 和 20H 单元中，高位在前。注意，使用 93C46 要根据板上的信号标注进行连线。

14、串行 EEPROM93C46 写数子程序：写数顺序，先调用写允许子程序“LCALL EABLE”，再将 8 位地址数 XXH、待写入的高 8 位字节和低 8 位字节分别放在 22H、21H 和 20H 单元中，然后调用写子程序“LCALL SEPWR”，最后调用写禁止子程序“LCALL DISLE”。15、如图 9-1 所示，时钟 50MHz 接 CLOCK2、时钟 12MHz 接 CLOCK0。

16、显示：显示系统由 7 个 74LS164 构成串行静态显示电路，此显示系统的数据口接单片机的 P3.0、CLOCK 接 P3.1；显示子程序名为“DIRR0”，7 个数码管的显示缓冲寄存器依次为：16H、15H、14H、13H、12H、11H、10H，小数点的缓冲寄存器是 0AH，其中某一位为 1 时，对应的数码管的小数点发亮。