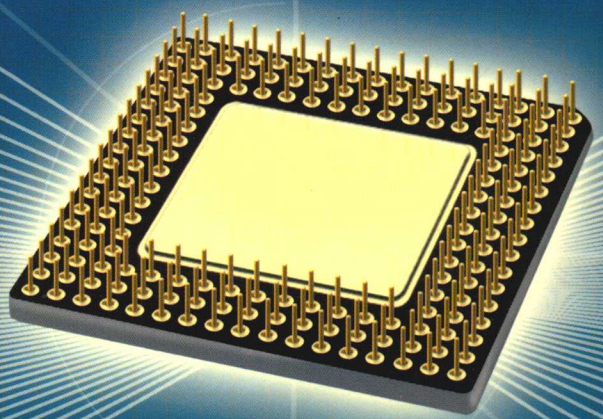
CPU/SOC及外围电路应用设计

——基于FPGA/CPLD

林容益 编著



北京航空航天大学出版社

内附光盘

CPU/SOC 及外围电路应用设计
——基于 FPGA/CPLD

林容益　编著

北京航空航天大学出版社

内 容 简 介

本书详尽介绍了简易8位CPU(含RISC)架构体系开发、设计及模拟测试的方法,各种CPU的外围设计模拟测试和CPU连接架构成单片机及SOC的方法。同时也对现代高速16/32位CPU架构体系开发测试和实例以及现代SOPC发展平台作了详尽的介绍分析,并配有例题程序光盘一张,方便读者学习使用。

本书可作为电子、电机、计算机、控制等专业的学生和从事VLSI、CPU、SOC芯片设计应用的科研人员的参考用书。

图书在版编目(CIP)数据

CPU/SOC及外围电路应用设计:基于FPGA/CPLD/林容益编著. —北京:北京航空航天大学出版社,2004.7

ISBN 7-81077-432-8

Ⅰ.C… Ⅱ.林… Ⅲ.微处理器—电路设计 Ⅳ.TP332

中国版本图书馆CIP数据核字(2004)第047168号

北京市版权局著作权合同登记号 图字:01-2003-7744

CPU/SOC及外围电路应用设计——基于FPGA/CPLD

林容益 编著

责任编辑 朱伟锋

*

北京航空航天大学出版社出版发行

北京市海淀区学院路37号(100083) 发行部电话:010-82317024 传真:010-82328026

http://www.buaapress.com.cn E-mail:bhpress@263.net

涿州市新华印刷有限公司印装 各地书店经销

*

开本:787×1 092 1/16 印张:32.5 字数:832千字

2004年7月第1版 2004年7月第1次印刷 印数:5 000册

ISBN 7-81077-432-8 定价:55.00元(含光盘1张)

序　言

现代数字电路的发展速度很快，尤其是高速度大容量的可编程逻辑元件装置不断的推陈出新，对应延时为200万门纳秒，采用0.13 μm技术的PLD元件不断的开发推出，使得数字电路规划设计及实现相当简洁方便。各种高速外围如VGA显示器、PCI Bus、MAC网络、USB、I^2C、SCI等等外围接口的规划设计或芯片化的合成都显得一蹴可及。尤其对简易8位或16位单片机结构设计与发展，进而对于32位高功能的CPU开发设计，采用这种高集成度的CPLD/FPGA搭配各种高功能开发纠错软件的方法，是开始电子科技扎根的必要开发技术，因而成为主要的发展目标。

数字通信、医疗影像、数字电视、高传真多声道数字音响AC3、MPEG 3/4、语音压缩解压缩、数字影像处理、辨识系统、雷达声纳、地震检测分析、各种矿产探勘等，以及工业控制、电力控制等，这些领域都可以采用数字信号处理技术。举一个很明显的例子，若要将类似的影音加以压缩是难以实现的或许会相当麻烦，但是转成数字信号后，就可根据数码格式加以分析运算来压缩数据进行保存或交流。这样不但节省空间时间，且可采用数字滤波器使得干扰性较少。这些应用领域都采取了数字信号控制处理的技术，其特有的数学演算技巧，都是DSP处理器的重要提升领域，但在哪个频率范围开始跨入数字化处理呢？若是以一般MCU来串行处理最多也仅可达到1 MHz的信号，高于此频率则需要大量的硬件来平行运算处理。因此使用一个高速运算的MCU搭配硬件构成的各种元件合成在一个芯片内，这就是现代SOC(System On a Chip)所追求的主要技术发展方向，并包括外围的驱动操作系统OS(Operation System)以及应用驱动软件构成，实为现代数字化高科技化的不可或缺的应用装置。

针对各种数字化所需要的大量运算处理，例如快速傅立叶转换运算(FFT)、有限脉冲(FIR)、无限脉冲(IIR)以及数字前向误差校正(FEC)、维特比(Viterbi)调制解调，以及PID控制器、编码器运算、功率检测运算处理等等，都需要极高速的乘加运算能力，并具有存储器数据指针移位运算的能力。各种数字信号演算方法的相继提出使得数字运算更加简洁快速。像这样需要高速、高精确度的运算器，必须以CPLD/FPGA硬件直接执行平行运算处理，但成本相当高，或是采用多个的运算专用(如FFT、FIR芯片)硬件模块来执行。

由以上得到一个结论，就是极高速必须采用硬件直接予以运算处理，或用芯片组来构成，中高速则采用多个DSP MCU作平行运算处理，一般的语音处理则采用较新的单个DSP MCU就绰绰有余了。假如将高效率运算的DSP MCU再搭配高速可平行运算的高灵活性硬件CPLD/FPGA则能发挥更高的效率，就如同PC中的CPU搭配高速外围的芯片组一样。对应较高速的信号可由CPLD/FPGA作前导取指或初步处理运算后，再交给DSP CPU来处理，或者是DSP CPU处理前后的信号可交给此CPLD/FGPA对外处理以便分摊CPU的工作，提高信号处理的效率，这也是近代数字化电路架构的主流。作者所设计规划的就是高集成度FPGA/CPLD微控器及其外围设计开发控制系统，这也是本系统设置及本书阐述的最主要目的。

要达到良好的数字信号处理功能，首先必须对高速，且灵活性极强的现代可编程硬件结构

电路CPLD/FPGA有相当的认识，本书利用FPGA/CPLD进行各种简易8/16位CPU的设计开发模拟和测试实验，接着介绍各种接口电路的设计规划和模拟测试，然后再详细说明了CPU作存储器映射或独立的I/O作连接规划控制的方法，紧跟着介绍现代各种单片机的架构及其规划应用，令读者了解现代各种单片机MCU的设计开发方法及其将来的发展方向。使用这些微控器来编写应用及控制程序，若能以其架构为着眼点来进入当然是驾轻就熟，实在是开发或应用单片机不可或缺的技术入门基础。

本书对现代多功能的16/32位CPU，如MIPS、ARM、TI的TMS320C5000或TMS320C600系列等具有平行运算能力的CPU等架构都进行了详细的分析说明。尤其对ARM的架构分析、SOC系统芯片的开发硬件及软件开发平台等都进行了详细的介绍，更列举了现代数字广播系统DAB对应于TI的专用SOC芯片DRE200的应用，并介绍了TI的ARM9和TMS320C55及各种外围元器件所构成的SOC芯片OMAP5910/19等的功能。而针对要自行发展SOC硬件架构的设计开发平台，则介绍ALTERA的Excalbur芯片组及高速AD/DA接口的SOPC开发硬件平台和QUTAS的SOPC硬件(SOPCP Builder)开发软件工具等，是组成一个开发现代高速数字化SOPC的最佳途径。

本书对32位的MIPS CPU由简化架构的规划设计及模拟测试，到5层管线节架构下其对应所产生的数据冒险(Hazard)、程序转移的指令停止(Stall)等技术性处理及模拟测试都进行了深入的分析说明，并配合作者所设计的SN-CPLDE3实验测试平台和EP1K100QC208-3的FPGA主机板，以及显示监控模块，进行实验检测验证，读者将可快速地进入CPU微控器及外围和SOC芯片设计开发及应用控制领域。事半功倍是作者秉持的一贯原则。

本书内容共分成五大部分，说明如下：

1. 简易8位单片机的架构(含管线节)开发设计和测试实验。
2. 各种外围如GPIO、CTC、PWM、UART、SCI、I^2C、QEP、DB、VGA及ADC/DAC等简易外围的设计开发和测试模拟。
3. 现代各种档次单片机微控器的架构功能及其发展应用。
4. 16/32位CPU(MIPS)及流水线等架构设计开发和模拟测试及应用。
5. 现代高性能平行处理CPU及ARM和SOC等芯片的架构分析及其开发硬件和软件平台介绍。

本书是台湾第一本针对CPU的架构及外围等设计开发测试技术的专业科技书。以引导方式深入浅出予以说明和模拟检测，相信是从事CPU/SOC芯片设计、VLSI开发等的专业科技人员不可或缺的经典参考书。

编纂匆匆！虽一再地编校，缪误难免，祈请不吝指正！不胜感谢！

谨志于

正弦电子工业有限公司　移动电话(0939)-285809
台北市双园街十八巷二号一楼(02)2302-2574
集电公司　台北县三重市自强路四段八号五楼(02)2286772

2003/09/28　林容益　敬上

目　录

第 1 章

简易单片机的结构及其开发模拟测试

16 位 PD 程序存储器 16 个精简指令 RISC 单片机简介

以 16 个双读取—写入的 DARAM 作为寄存器的操作单元(Operator),因此需要以 4 位来寻址,在 16 位的程序存储器 PD 中,其操作指令设计设置下列 3 种模态。

- 采用 3 个操作单元,即 RS,RT 和 RD 等会占用 12 位,而指令则仅有 4 位,最多的操作指令则有 16 种,为简化 CPU 结构以便说明,故此种参运算元指令大都是 ALU 的运算,如 ADD,SUB,AND ,OR,XOR 等 5 个指令,其格式为

  OP RT,RS,RD

- 寄存器与直接数据的操作模式,寄存器若为 8 位数据,则可直接与 8 位数据操作,因此单一操作单元 4 位与 8 位数据可直接作运算,如 ADD,SUB,AND,OR 等共 4 个,其格式为

  OP Rd, Imm

- 分支程序:没有运算单元的指令,故除了 4 位的指令外分支程序寻址就有 12 位,分成条件的转移,如 JZ,JNZ,JC,JNC 等 4 种,以及无条件的转移,如 JMP 和子程序调用 CALL 和 RET 回主程序等共 7 种程序转移指令。

以上 16 个精简指令操作功能及格式如表 1-1 所列。

指令储存于程序存储器内,要达到上述的操作功能,则程序存储器必须有一个可更改,或可预设或递增的程序计数器来对应程序存储器作寻址,以便读取指令。寄存器的寻址以及立即数据或是转移的程序地址,程序数据以及寄存器的寻址等皆由程序存储器予以取指,因此 MCU 的第一个动作需要由程序存储器—程序计数器的寻址予以取指,称之为取指(Fetch)动作。

由程序存储器取指到的数据或寄存器的寻址,则必须一次由寄存器库内读取数据,以便于 MCU 作运算、数据读写,或条件判断,因此这个步骤称为读(Read)动作。

读取到寄存器或存储器的数据后便可进行 ALU 的运算,或不作运算,仅单纯的读写动作,故这个步骤称为执行(Execute)动作。

当执行 MCU 操作后,操作结果可能需要将其写入寄存器或存储器,或结果作为条件的执行程序转移,这个步骤称为写(Write)动作。

以上 4 个步骤,将依次分别详细分析,对应 HDL 的描述设计及模拟测试如下:

表 1-1　16 个 RISC 精简指令格式及对应说明

序	指令及操作单元格式				指令符号	指令功能说明	标　志
0	0000	xxxx	yyyy	zzzz	XORX,Y,Z	X 与 Y 相异或放入 Z 寄存器	
1	0001	----	----	----	RET	回主程序	
2	0010	aaaa	aaaa	aaaa	JMP AAAH	无条件跳到 AAH 相对地址	
3	0011	cccc	cccc	cccc	CALL CCCH	无条件调用 CCCH 相对地址	
4	0100	zzzz	zzzz	zzzz	JZ　ZZZH	当 Zf=1 跳到 ZZZH 相对地址	
5	0101	zzzz	zzzz	zzzz	JNZ ZZZH	当 Zf=0 跳到 ZZZH 相对地址	
6	0110	cccc	cccc	cccc	JC　CCCH	当 Cf=1 跳到 CCCH 相对地址	
7	0111	cccc	cccc	cccc	JNC ZZZH	当 Cf=0 跳到 CCCH 相对地址	
8	1000	xxxx	yyyy	zzzz	ADD X,Y,Z	X 与 Y 内容相加放入 Z 寄存器	Zf,Cf
9	1001	xxxx	yyyy	zzzz	SUB X,Y,Z	X 与 Y 内容相减放入 Z 寄存器	Zf,Cf
10	1010	xxxx	yyyy	zzzz	AND X,Y,Z	X 与 Y 相和放入 Z 寄存器	Zf
11	1011	xxxx	yyyy	zzzz	OR X,Y,Z	X 与 Y 相或放入 Z 寄存器	Zff
12	1100	xxxx	dddd	dddd	ADD X,#DD	X 与数据 DD 直接相加放入 X	Zf,Cf
13	1101	xxxx	dddd	dddd	SUB X,#DD	X 与数据 DD 直接相减放入 X	Zf,Cf
14	1110	xxxx	dddd	dddd	AND X,#DD	X 与数据 DD 直接 AND 放入 X	Zf
15	1111	xxxx	dddd	dddd	OR X,#DD	X 与数据 DD 直接 OR 放入 X	Zf

1-1　CPLD 设置 CPU 的 HDL 描述硬件结构说明

1-1-1　CPLD 设置 CPU 的 HDL 使用 LPM 模块简介

本系统使用到 LPM 模块的“LPM_MUX”，以大小 LPM_SIZE＝SS 设置多路组数，及 LPM_WIDTH＝WW 多路位宽和输入选择控制码位宽 LPM_WIDTHS＝SW 等参变数设置。输入为 MUX. DATA[ ]，输出为 MUX. RESULT[ ]及控制选择输入的 MUX. SEL[ ]等 3 个输入输出矩阵接口端。多路器共有 muxa，muxb 及 muxstk 等 3 组，另外使用 LPM 模块的“LPM_ADD_SUB”作 ALU 的 ADD 及 SUB 等加减运算和 PC 寻址、跳转及递增等运算控制。pcadd 其参数是 LPM_WIDTH＝12 位宽及 LPM_DIRECTION ＝“ADD”为加法运算方向设置，输入为 pcadd. dataa[]及 pcadd. datab[]2 组 12 位输入，输出 pcadd. result[]为运算结果。另外一个模块是 FLEX 10KXX 系列 EAB 模块仅有的“LPM_ROM”程序存储器模块。以参数 LPM_WIDTH＝16 设置其 ROM 是 16 位，而地址宽则以 LPM_WIDTHAD＝8 设置，地址为 A7～A0 共 8 位。其 ROM 的内容由 LPM_FILE＝“prog2. mif”予以写入。另外需设置寻址输入控制模式是否需以寄存器予以锁存或不为寄存器模式的 LPM_ADDRESS_CONTROL＝“UNREGISTERED”，同样输出数据设置为寄存器式的“REGISTERED”或非寄存器 LPM_OUTDATA＝“UNREGISTERED”模式，寻址的输入 prom. addr[7..0]＝pc[7..0]作输入设置，而输出数据设置为 pd[15..0]＝prom. q[15..0]。详细的 LPM 模块的参数设置及输入输

出的编写设置可参照 ALTERA 公司的 HELP 文档内的 Megafunction /LPM。

1-1-2 CPLD 设置 CPU 的 HDL 使用寄存器简介

CPU 内部所需使用的运算的寄存器共有 16 组 8 位寄存器，故需以 D 型反向器担任，即 rgg[15..0][7..0]宣告其为 DFF，共有 16×8=128 个寄存器，程序计数器共 12 位，故也需 12 个 DFF，即 pc[11..0]。另外由 rgg[][]所取出的数据在加入运算前必须先予以寄存，故需 rqab[7..0]及 rqbb[7..0]等 2 组 8 位 DFF 寄存器，共需 16 个 DFF。ALU 运算结果需将其寄存，也要一个 8 位 DFF 寄存器 rdi[7..0]作数据寄存。另外程序内容即 prom.q[15..0]必需在第一时序将其寄存于 pdx[]内，接着第 2 时序来临将 pdx[]内容转存于 pdy[]，当然 pdx[]内容又会被新的 prom.q[]写入，因此，pdx[]及 pdy[]共需使用 2 个 16 位寄存器作数据锁存。程序计数器的寻址共 12 位也需要加以锁存寄存，以便推入堆栈或递增或进行相对地址计算跳转等控制。以 pcx[11..0]随时对 PC 值加以锁存备用，16 个 0～F 寻址的通用寄存器需要 4 位寄存器 ws[3..0]作运算单元 X，Y，Z 等寄存器的寻址设置。堆栈共有 4 组，各为 12 位寄存程序计数器地址，故 stk[3..0][11..0]共需 4×12=48 个 DFF 反向器寄存取 PC 的 CALL 及 RET 使用。堆栈寻址指针寄存器 sp[1..0]需 2 个 DFF 反向器。故以上总计需 128+12+16+32+12+4+48+2=264 个 DFF 反向器。另外寄存器写入使能标志 wen，wenx，weny 等需 3 个 DFF 反向器，PC 寻址控制设置标志 pcmx，pcmxx，pcadmx，pcadmxx 等 4 个 DFF 寄存器及堆栈状态 pushst 和其寄存标志 pushstx 等 2 个 DFF 反向器，及 ALU 的运算方向 aldb 寄存器等共 9 个 DFF，组合成 264+9=273 个 DFF 寄存器。当然 FLEX 10KXX 内还有 128×8 位 RAM 可用。

另外程序内容由 prom.q[]所取得的数据 d15～d0 可暂放于 pd[15..0]节点 NODE。ALU 运算结果的输出也需由 alo[7..0]节点取得，再输入 rdi[7..0]作锁存输入。多路器 MUXB 由寄存器 rgg[][]取得后需，以 rqb[7..0]作节点再输入 rqbb[7..0]寄存器锁存寄存。跳转改变 PC 地址的条件，设置为 tcnd 节点控制和零值 ZF 标志节点 rzr 等，都是内部临时运算节点而非直接的输入输出接口端。

预留有 4×8=32 个 I/O 接口 iopt[3..0][7..0]，由寄存器 BH～FH 加以设置控制，并作双向的输入输出，故称其为 BIDIR 模式。系统主脉冲由 sysclk 端予以输入，故此 CPLD 组成的单片机仅有输入 sysclk 的脉冲输入引脚，及 4×8=32 个可双向设置的 BIDIR 作输入输出端引脚，因此使用 84 引脚的 FLEX 10K10LC84-4 的 CPLD 是足足有余了。

上述的 LPM 声明或参数输入设置及各 DFF 反向器和节点 NODE 的声明是以 VARIABLE 指令声明的，整个声明如下：

```
INCLUDE "lpm_mux";
INCLUDE "lpm_add_sub";
INCLUDE "lpm_rom";
INCLUDE "lpm_ram_dq";
SUBDESIGN riscpu3
(     sysclk,RST          :  INPUT;
      iopt[3..0][7..0]          :  BIDIR;
```

```
    AT[7..0];output;)
VARIABLE
rgg[15..1][7..0]        :  DFF;
muxa,muxb        :  lpm_mux
WITH(LPM_SIZE=16,LPM_WIDTH=8,LPM_WIDTHS=4);
muxstk        :  lpm_mux
WITH(LPM_SIZE=4,LPM_WIDTH=12,LPM_WIDTHS=2);
pc[11..0],cntp[1..0]        :  DFF;
pcadd        :  lpm_add_sub
WITH(LPM_WIDTH=12,LPM_DIRECTION="ADD");
pd[15..0],rqb[7..0],alo[8..0],tcnd,rzr,t[3..0],aldb        : NODE;
rqab[7..0],rqbb[7..0],rdi[7..0],alox[7..0],alox8        : DFF;
wen,swen:dff;
pcadmx,pcmx,pushst        : NODE;
pdx[15..0],ws[3..0]        : DFF;
stk[3..0][11..0],sp[1..0]        : DFF;
prom        : lpm_rom WITH(LPM_WIDTH=16,LPM_WIDTHAD=8,LPM_FILE="prog5p1.mif"
        ,LPM_ADDRESS_CONTROL="UNREGISTERED",LPM_OUTDATA="UNREGISTERED");
sram        : lpm_ram_dq WITH(LPM_WIDTH=8,LPM_WIDTHAD=8,LPM_INDATA="REGISTERED"
        ,LPM_ADDRESS_CONTROL="UNREGISTERED",LPM_OUTDATA
    ="UNREGISTERED");
```

CPLD 设置 CPU 的 HDL 描述设计说明

如前面所介绍的 CPLD 设置 CPU 的 HDL 描述由 3 个部分组成，即寄存器数据运算控制部分、程序计数器运算控制部分、程序内容数据运算控制部分，再加上指令解码控制、条件判断、标志设置 I/O 数据及方向设置程序内容 PROM 的设置等 7 大部分，现将分别依次介绍如下。

1-1-3　程序存储器的设置设计及取指(Fetch)

对应程序计数器 PC 的递增加一，或 JMP 指令的跳转改到任意 A11～A0 地址，或条件跳转指令“JP CC,ADR”当 CC 条件的 JZ,JNZ,JC,JNC 等成立时就需将程序计数器改到 PC+ADR 相对地址端。也就是说将 PC 值与 ADR 地址相加产生程序分支功能，而对应 CALL 及 RET 指令则在 CALL 指令将下一个 PC 存于堆栈 STK 内，再跳转到相对值 PC+ADR 地址，当执行 RET 指令时需由 STK 的寄存器内取回原 CALL 时存入的下一个 PC 地址值而跳回主程序。程序计数器 PC 运算控制流程如图 1-1 所示。

图 1-1 示意图中可见当执行 JMP,JP CC,ADR 等指令时，以 pcadmx 寄存器的锁存状态判别及确认 PC 要转移地址。也就是说必须确认状态 pcadmx 处于条件解码(本指令第 3 时序)CC 都成立时，令 tcnd=1，同时在此指令解码后的第 3 个时序作相对地址 ADR 的运算跳转。因此改变 PC 指令是在第 3 个时序后才转变了 PC 值出现。若是 CALL 指令时必须令 pushst=1 堆栈标志在转移 pcadmx=1 状态下，由 pushst=1 标志对应执行下一个 pcx[11..0]=ADR 即主程序的下一个地址值，锁存入 SP 堆栈指针所指的 sp[i][11..0]存储器内，并令

堆栈指针 sp[i]减 1(或加 1)以便下一个 CALL 指令再推入 sp－1 地址内。当执行到 RET 指令时除了必须令 pcadmx＝1 作程序转移标志外，同样必须令 pcmx＝1，以便在第 3 个时序时，除了执行由堆栈指针所指的存储器内取回主程序地址作为转移(pcadmx＝1)寻址地址外，可依据 pcmx＝1 标志将堆栈指针寻址加 1，即令 sp[i]＋1 以便下一个 CALL 可正确的使用此堆栈指针。

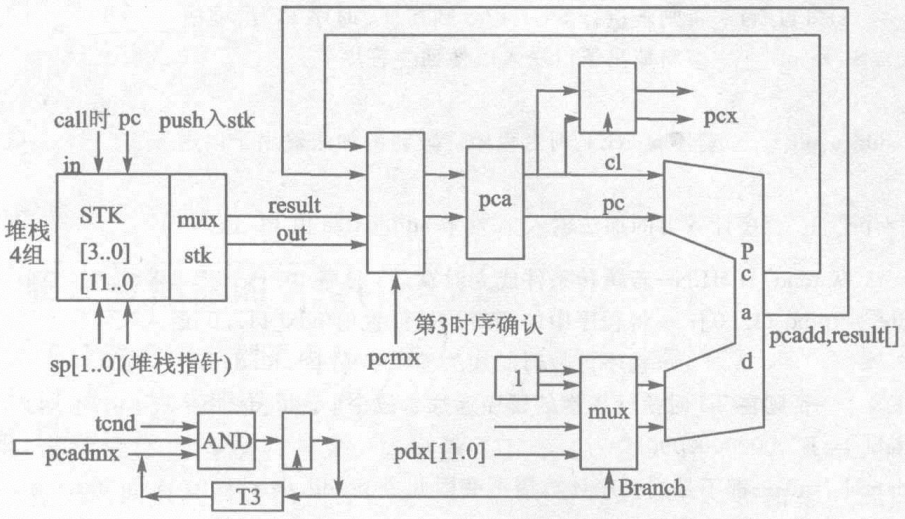


图 1－1　程序计数器运算控制结构框图

程序存储器的地址计数器 PC(Programmer Counter)在不执行转移指令时，每一个执行指令后自动增加 1，依次取出程序数据予以解码执行。若是条件 CC 程序跳转，必须等指令完成操作后，才可得知条件是否成立(tcnd＝1?)，再依据条件状态作程序跳转的动作。而无条件的跳转如 JMP 指令则为了系统解码简洁起见，只要单纯令条件标志 tcnd＝1 就可以了。对应 CALL 子程序的调用执行也是无条件的令 tcnd＝1，除了无条件的程序跳转外，为了将来在执行 RET 指令回主程序时，可正确地回到原主程序执行 CALL 指令的下一个程序地址。因此必须设置一个堆栈指针，寻址一堆存储器用来存放主程序回归的程序存储器地址。在此堆栈的设置是固定一堆存储器，有 4 个存储器，因此可连续作 CALL 调用。子程序随着堆栈指针的递减 1，而将回归的主程序地址存放于对应堆栈存储器内，故 CALL 指令仅可连续执行 4 个层次。而 RET 转移指令以 PCMX 标志作标志，当执行 RET 指令时，则根据 PCMX 依堆栈指针所指的对应堆栈存储器内容送入程序存储器 PC 内，则程序便回到主程序了，但也必须令堆栈指针加 1 回到上一个 CALL 指令的回归主程序存放地址。

程序存储器由 LPM_ROM 取的 CPLD 内部的 EAB 存储器，内含数据由程序 1－1 所示的 PROG5P1. MIF 予以读取送入。此存储器数据为 16 位，地址宽则因为要使用 EPF10K10 较小 CPLD 的限制，故仅使用 8 位原有 256 个程序存储器供应编辑程序。其程序地址由 PC 予以寻址，而程序数据则读取到 PD[]寄存器内，程序计数器的转移控制或递加设置如图 1－1 所示，对应 AHDL 的描述则如程序 1－1 所列。

程序 1－1　对应 AHDL 的描述

```
prom      : lpm_rom
WITH(LPM_WIDTH＝16,LPM_WIDTHAD＝8,LPM_FILE＝"prog5p1. mif"
```

```
,LPM_ADDRESS_CONTROL="UNREGISTERED",LPM_OUTDATA="UNREGISTERED");
prom.address[]=pc[7..0];--程序计数器寻址 PROM 的地址线
pd[]=prom.q[]; --PROM 的数据读出置放于 PD[]寄存器内
pc[].clk=sysclk; --程序计数器由系统 SYSCLK 予以锁存
IF RST THEN PC[]=0 ;--若系统受 RST 输入 1 重设则 PC=00
ELSIF (pcmx & t3) THEN --否则若执行 RET 指令则在 T3 时序 PC 由堆栈
    pc[]=muxstk.result[];--多路输出予以送入以便回主程序
ELSE
    pc[]=pcadd.result[]; --若不是 RET 则由程序计数器的加法输出予以送入
END IF;
pcadd.dataa[]=pc[];--程序计数器的加法输入 A 端 pcadd.dataa 由 PC 送入
IF (pcadmx & t3 & tcnd) THEN--若跳转条件成立时在 T3 时序
   pcadd.datab[]=pdx[11..0]; --将程序中的相对地址设置值 pdx[11..0]送入
                             --程序计数器的加法输入 B 端 pcadd.datab
ELSIF t3 THEN   --否则在 T3 时执行正常的程序递加 1 故令 pcadd.datab=1
    pcadd.datab[]=B"000000000001";
ELSE pcadd.datab[]=0;--都不是则程序计数器不变因此令 pcadd.datab=0
END IF;
```

对应程序存储器由堆栈多路选择器 muxstk.result[]予以读取，或是程序跳转的相对程序地址加法器输出端的 pcadd.result[]予以送入，或者是正常的递增执行的 AHDL 电路。描述如下列所述。

若是 RET 指令则由标志 PCMX 以多路选择辨识，从堆栈指针内容值读回主程序，电路描述如下：

```
pc[].clk=sysclk;
IF RST THEN PC[]=0 ;
ELSIF (pcmx & t3) THEN
        pc[]=muxstk.result[];
ELSE
        pc[]=pcadd.result[];--非 RET 指令则由程序计数器的加法输出端送入
END IF;
pcadd.dataa[]=pc[];
IF (pcadmx & t3 & tcnd) THEN
--程序跳转条件成立时，由直接寻址 PD[11..0]送入 PC 作为 PC+PD[]跳转地址
--计算到程序所设置地址
        pcadd.datab[]=pdx[11..0];
ELSIF t3 THEN
        pcadd.datab[]=B"000000000001";
--不是相对地址差值跳转则以 01 作递加运算
else pcadd.datab[]=0;
END IF;
```