基于 FPGA 的自定义 CPU 架构设计*

李 俊[1],任连新[2],廖振雄[3]
(1.深圳市合信自动化技术有限公司,广东 深圳 518055;
2.华南理工大学 自动化科学与工程学院,广东 广州 510640;3.深圳市科创思科技有限公司,广东 深圳 518055)

摘 要:为满足当前工业应用下越来越多的分布式计算的需求,提出了一种在 FPGA 芯片中构建自定义指令集的 CPU 的方式,以此来使 FPGA 具有类似于单片机的处理指令的能力。并且,这种能力的前提是复用计算单元,因此资源消耗有限,不会随着计算量的增加而增大。在自定义指令集 CPU 的改进型架构中,使用了并行计算的结构,使得运算速度大幅提升。最后,结合实际应用案例,移植电流环计算中的 FOC 算法到自定义 CPU 中运算。并用 ModelSim 软件进行仿真,测试其计算时间仅需 7.48 μs。

关键词:自定义指令集;CPU 架构;FPGA;并行计算结构;FOC
中图分类号:TN402 **文献标识码**:A **DOI**:10.16157/j.issn.0258-7998.200011

中文引用格式:李俊,任连新,廖振雄. 基于 FPGA 的自定义 CPU 架构设计[J].电子技术应用,2020,46(5):40-43,49.
英文引用格式:Li Jun,Ren Lianxin,Liao Zhenxiong. Design of custom CPU architecture based on FPGA[J]. Application of Electronic Technique,2020,46(5):40-43,49.

Design of custom CPU architecture based on FPGA

Li Jun[1],Ren Lianxin[2],Liao Zhenxiong[3]
(1.Shenzhen Co-trust Technology Limited Company,Shenzhen 518055,China;
2.College of Automation Science and Engineering,South China University of Technology,Guangzhou 510640,China;
3.Shenzhen Kechuangsi Technology Limited Company,Shenzhen 518055,China)

Abstract: In order to meet the needs of more and more distributed computing in current industrial applications, this article proposes a way to build a custom instruction set CPU in an FPGA chip. In this way, the FPGA has the ability to process instructions similar to a microcontroller. Moreover, the premise of this capability is the reuse of computing units, so resource consumption is limited and will not increase as the amount of calculation increases. In the improved architecture of the custom instruction set CPU, a parallel computing structure is used, which greatly improves the operation speed. Finally, combining the actual application case, the FOC algorithm in the current loop calculation is transplanted to the operation in the custom CPU. And ModelSim software is used to simulate, its calculation time is only 7.48 μs.

Key words: custom instruction set; CPU architecture; FPGA; parallel computing structure; FOC

0 引言

在目前的工业应用环境下,许多的工业设备控制器中都包含有相当复杂程度的算法。例如:状态观测器、卡尔曼滤波器、模糊控制算法、甚至是神经网络算法,不一而足。其中有些算法计算步骤复杂,同时又对控制带宽有一定的要求,所以对设备的处理器芯片的运算能力要求很高。因此,有些设备中可能会同时存在 2 个甚至多个处理器,分别完成不同的功能算法。这样多个处理器分布式地处理不同的算法,完成不同的控制功能,无疑是一个解决思路。但是这又会带来 IC 数量增加,成本上升的问题。同时,现在的工业设备控制器的主控 PCB 上的集成度越来越高,在上面增加 IC 对硬件设计也是一种挑战。

现在许多设备控制主板上都会使用 FPGA 芯片来对外部信号做预处理。本文旨在提供一种在 FPGA 内构建精简架构的 CPU, 使其可以具备算法处理能力的方法。这样,在不额外增加处理器的前提下,可以让 FPGA 也承担一部分的计算任务[1]。

1 基于 FPGA 的 CPU 架构设计

1.1 (精简)特殊指令集 CPU 结构的引出

一套算法在 C 语言中一般是以一系列的算术计算和逻辑运算结合完成的。C 语言的编译器(一般是 GCC 编译器)会把相应的 C 语句编译成可供 CPU 执行的汇编指令。这样,一个通用指令集架构的 CPU 可以完成许多的指令运算。它的运算单元(ALU)单一时刻运算某一个指令。

*基金项目:广东省省级科技计划项目(2016B090911003)

用 FPGA 完成一整套的算法计算时，如果为每一条计算公式单独地实例化一个运算单元(可能是乘法器，加法器，或者是其他任意的计算逻辑)，那么消耗的资源将会随着计算量的增加而增长。一片 FPGA 芯片的资源有限，总会在计算量增长到某个边界时全部用完。而且，每一个计算逻辑只计算某一个单一的算式，十分浪费资源。如何做到复用计算资源呢？可以仿照 CPU 的架构，定义一组可以执行各种计算指令的单元，再通过一个流程调度器，依次把数据放入计算单元，取得计算结果。这样，不论何种计算都可以通过复用一个计算单元来完成。也即：自定义一套(精简)特殊指令集，再按照指令集设计一个 CPU 架构。所有需要计算的公式和数据都以指令的形式输入进这个 CPU，CPU 根据指令进行一系列的操作，输出计算结果[2-3]。

1.2 CPU 的结构

CPU 中包含至少一个逻辑运算单元(Arithmetic Logic Unit，ALU)，用来处理各种指令，返回结果。CPU 中应该有一个指令调度器，负责调度需要执行的指令。指令本身包含操作码、操作数的地址和结果的地址。CPU 根据指令中的操作数地址取数送入 ALU 中，ALU 根据相应的操作码运算，再根据结果地址把结果存放起来。指令保存在一块片上内存里，这块内存需要在上电的时候初始化，把需要执行的指令在初始化的时候加载上来。这类似于 CPU 上电工作的时候，需要从 Flash 或者磁盘中把要执行的指令先读到内存里待执行。对数据的操作，要用另一块片上内存。包括过程数据在内的数据都在这片内存里被存、取。CPU 根据指令来调度这个内存中的数。显然数据和指令不在同一个内存里，因此数据和指令的寻址由各自的总线管理[4]。所以，定义的这个 CPU 是哈佛结构的，如图 1 所示。

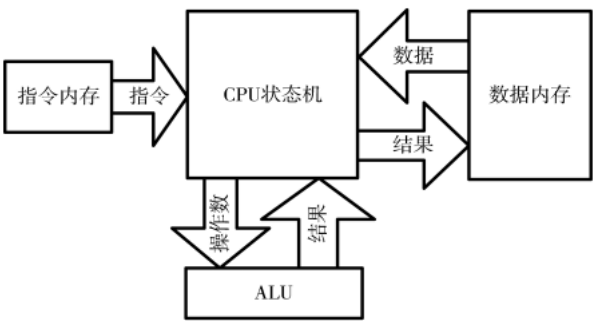

图 1 基于 FPGA 的 CPU 架构

1.3 CPU 的子功能模块

前文指出，CPU 需要实现几个功能：指令调度、指令执行、数据存取[5]。于是设计以下模块来完成相应的工作。

1.3.1 CPU 状态机和指令内存

“CPU 状态机”模块实现了一个状态机，管理 CPU 运行的流程，规定了指令的执行逻辑。状态机的执行过程是这样的：先从指令内存中取一条指令，对其解析，然后用指令中的操作数地址从数据内存取数，再把操作数和操作码一同送入 ALU，等待 ALU 的结果后，按照指令中的结果地址，把结果写入数据内存。

同时模块里使用了一个 FPGA 的片上 RAM(当作 ROM)作为这个 CPU 的指令内存，其中储存被执行的指令。指令内存需要初始化，FPGA 在上电配置时就对 RAM 加载初始化数据。指令内存是当作 ROM 来使用的，在上电过程中加载一次数据，之后都不会对指令内存中的数据作修改。因此 RAM 在实例化的时候没有分配写操作接口，只有读操作接口。指令内存的读取地址由状态机管理，上电初始化和复位情况下，地址为“0”。在处理算数指令(乘法、加法、位移除法等数学计算的指令)时，每处理完一条指令，指令内存的寻址地址自加“1”，指向下一条；在处理逻辑判断指令时，如果 ALU 返回的结果为真，指令内存的寻址地址就跳转到结果地址。CPU 借此来实现指令的跳转。

状态机的执行顺序如下：状态机取出一条指令后，把指令截取为 4 个部分并锁存：操作码、操作数 0 的地址、操作数 1 的地址、结果地址。每个部分占一个字节，一条指令有 4 个字节。再按照操作数 0 的地址和操作数 1 的地址从数据内存的相应位置取出操作数。待操作数取出后，连同操作码一起送入 ALU。ALU 完成运算后，再根据结果地址把运算结果存入数据内存。

1.3.2 CPU 数据内存

数据内存模块以真双口 RAM 的方式实例化了一个片上 RAM，作为 CPU 的数据内存。数据内存用来存放包括过程数据在内的数据，供 CPU 存取。CPU 从这里取出操作数去运算，计算完成后再存入运算结果。

在一套算法公式中，计算通常会用到一些常数。用初始化 RAM 的方法，把这些常数在上电配置的时候就预存进 RAM 里。

1.3.3 ALU

逻辑运算器模块，简称 ALU 模块，是 CPU 的运算单元。这个 ALU 是根据指令集的架构定义的，可以处理加法、减法、乘法、除法、位移、三角函数(Sin/Cos)、逻辑判断(大于、大于等于、等于)这些运算或判断，并输出结果。ALU 会根据操作码来执行不同的运算。

这样，ALU 就是一个可以复用的计算单元，所有的计算资源都集中在这里。目前定义的操作码见表 1。

1.3.4 外部数据的访问

根据上述关于 CPU 结构的讨论，整个 CPU 已经可以运行起来了。但是离实际应用还有一点距离。在上述结构中，数据的流转都是在这个 CPU 结构的内部。那么，当 CPU 需要获取结构外部的数据时(比如，需要读入一个 ADC 的输入值时)，该怎么办呢？显然这时候就需要 CPU 结构还能够访问到除了数据内存之外的地方。

方法也很简单：把一部分数据内存的寻址映射到寄存器上去。当 CPU 状态机寻址数据的地址为某个特定

表 1 操作码

		Bit4	Bit3	Bit2	Bit1	Bit0	[4:0](Hex)
Null		0	0	0	0	0	0x00
Mult		0	0	0	0	1	0x01
Add-Sub	Add	0	0	0	1	1	0x03
	Sub	0	0	0	1	0	0x02
Div	Shift-Div	0	0	1	0	0	0x04
	Div	0	0	1	0	1	0x05
Compare	G	0	1	1	0	0	0x0C
	GE	0	1	0	1	0	0x0A
	E	0	1	0	0	1	0x09
Trigone	Sin()	1	x	x	x	1	0x11
	Cos()	1	x	x	x	0	0x10

的地址段时，寻址目标不再是数据内存，而是一片寄存器。外部数据可以通过这些寄存器被读入 CPU。

于是，CPU 对数据的寻址分为了两部分：一部分地址映射到数据内存中，一部分地址映射到外部寄存器中。映射到外部寄存器的数据可以用来读取外部输入值，输出计算结果。

1.4 改进型 CPU 结构

1.4.1 并行计算结构的提出

FPGA 因为其原理的特殊性，以并行执行能力而著称。根据这一特点，考虑可否让基于 FPGA 的 CPU 具备并行计算的能力。

1.4.2 并行结构(Parallel Structure)

整个 CPU 的结构在最初的设计中，只包含一组 CPU 状态机、数据内存、ALU，这也符合上文中的讨论。这样的设计可以让指令在 CPU 中逐条地被执行，当指令数量较多时，执行时间自然也变长。为了提高 CPU 的计算性能，在同样的时间内执行尽可能多的指令，提出了平行计算结构。

有些算法会具备对称的特点，于是考虑在实现这种算法的时候利用其对称性，同一时间操作两条对称的运算，提高效率。因此提出消耗资源翻倍，但是能换来提升效率约 65% 的方式——用一组平行的 CPU 状态机和 ALU 同时运算不同的指令。这些资源消耗会加倍：CPU 状态机、指令内存、ALU。两者共用一个数据内存。两个状态机同时启动，运行对称的指令，从同一个数据内存中取数，在两个 ALU 中分别计算出结果，存入数据内存，如图 2 所示。

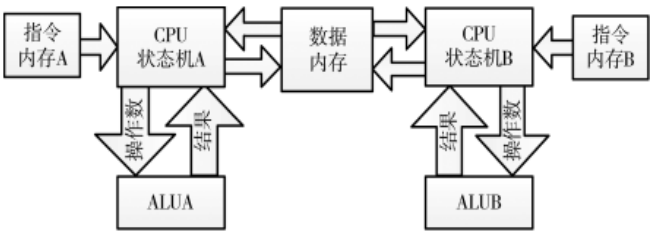

图 2 并行 CPU 的结构

这种结构在使用上是有条件的。平行结构类似于双核 CPU 的工作方式。此时，为了保证两组计算流程中用到的数据的依赖关系，要求指令的执行情况完全地对称——A 组在计算加法时 B 组也必须计算加法，A 组计算除法时 B 组也要计算除法。即使有一个 ALU 没有执行有效操作，去操作一些无效数据，也要保证和另一个 ALU 消耗相同的运算时间。这样，两个 ALU 可以在没有互锁或是数据保护的情况下正确地运行，同时启动，同时结束。

2 实验结果

2.1 CPU 状态机的仿真

图 3 显示了 CPU 状态机的工作流程。图中展示了 CPU 状态机操作一条指令的过程：首先从指令中取出一条指令，再取得两个操作数进行运算，运算完成后一条指令就执行完了。

2.2 并行结构 ALU 的仿真

图 4 显示了双 ALU 构造的并行计算结构的计算形式。图中，通过编辑指令使得两个 ALU 在工作时同时执

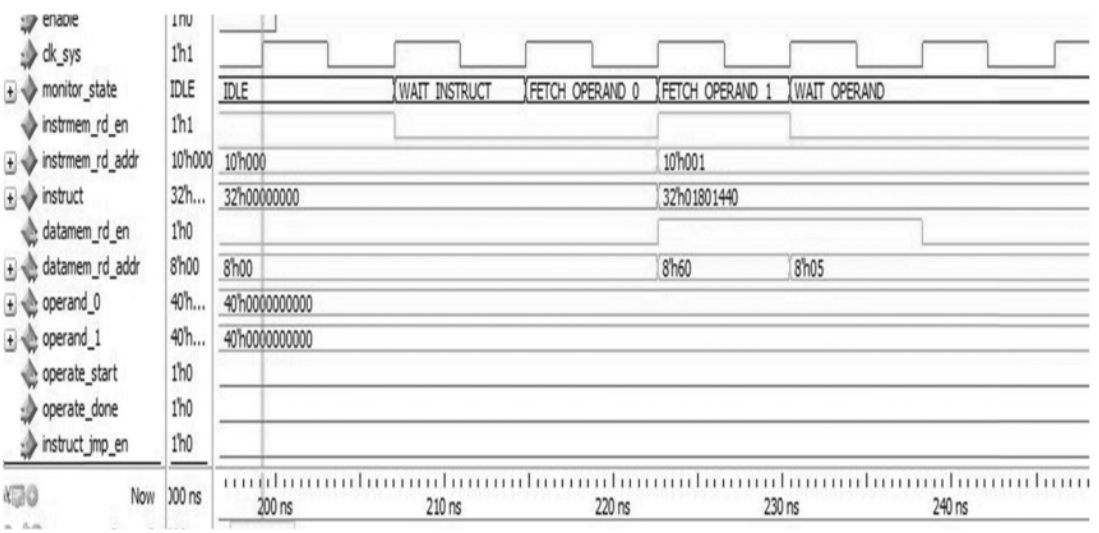

图 3 CPU 状态机仿真

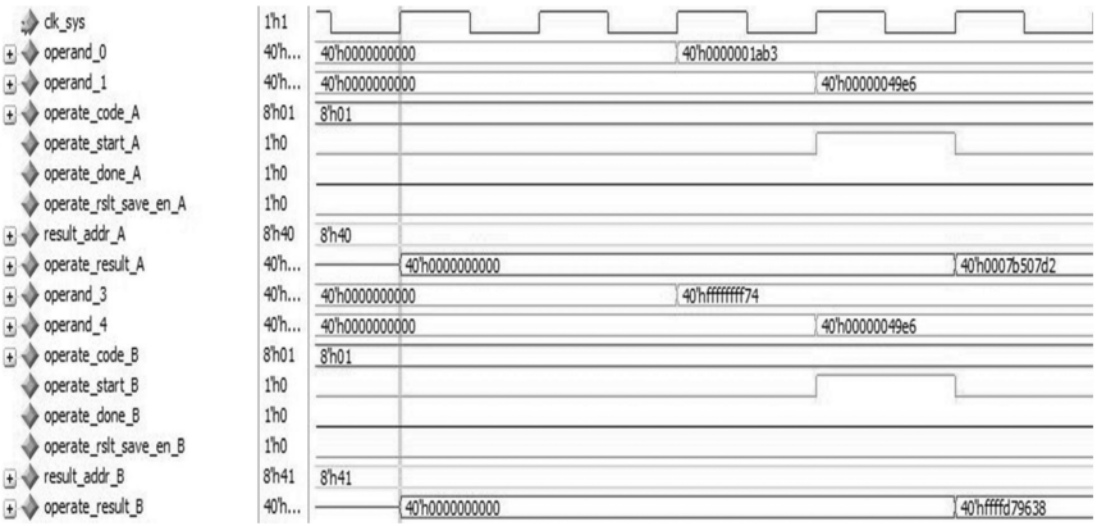

图 4 并行 ALU 计算过程仿真

行相同的运算，同时输出计算结果。这可以充分利用 FPGA 并行性的特点，提高计算效率。

2.3 CPU 的实际应用仿真

为了模拟实际使用的情况，实验选择在构建的 CPU 中计算一个常用的算法——伺服控制中的电流环磁场定向控制(FOC)算法。FOC 是根据电流的采样值和电角度的采样值通过计算得到下个周期 IGBT 开关管导通时间的算法，其计算流程如图 5 所示。

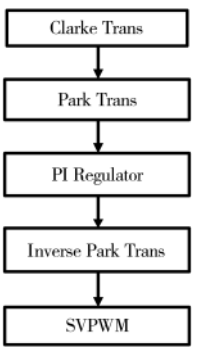

图 5 FOC 计算流程

在上文构建的基于 FPGA 的 CPU 的基础上，编辑指令来实现电流环的 FOC 计算。并基于仿真工具 ModelSim，对建立的 CPU 系统的运行情况进行仿真。得到仿真结果如图 6 所示。

整个 CPU 模块的系统时钟为 125 MHz。指令实现的内容包含了 FOC 计算的全部过程：Clarke 变换、Park 变换、电流 PI 调节器、反 Park 变换和 SVPWM 调制。

在使能信号“enable”置位“1”以后，CPU 通过外部寄存器读取电流采样值“Ia”、“Ib”和电角度值“Theta1”的值作为外部传感器输入值；读取转矩指令“T_SV”、励磁指令“F_SV”、PI 调节器比例系数“Kp”和积分系数“Ki”的值作为外部指令。然后逐步计算，按顺序输出：Clarke 变换的结果“Ialpha”和“Ibeta”；Park 变换的结果“Iq”和“Id”；PI 调节的结果“Vq”和“Vd”；反 Park 变换的结果“Valpha”和“Vbeta”。运行完整个指令流程后，输出 SVPWM 的三相占空比“TimeU”、“TimeV”、“TimeW”。整个电流环 FOC 计算过程一共耗时 7.48 μs。

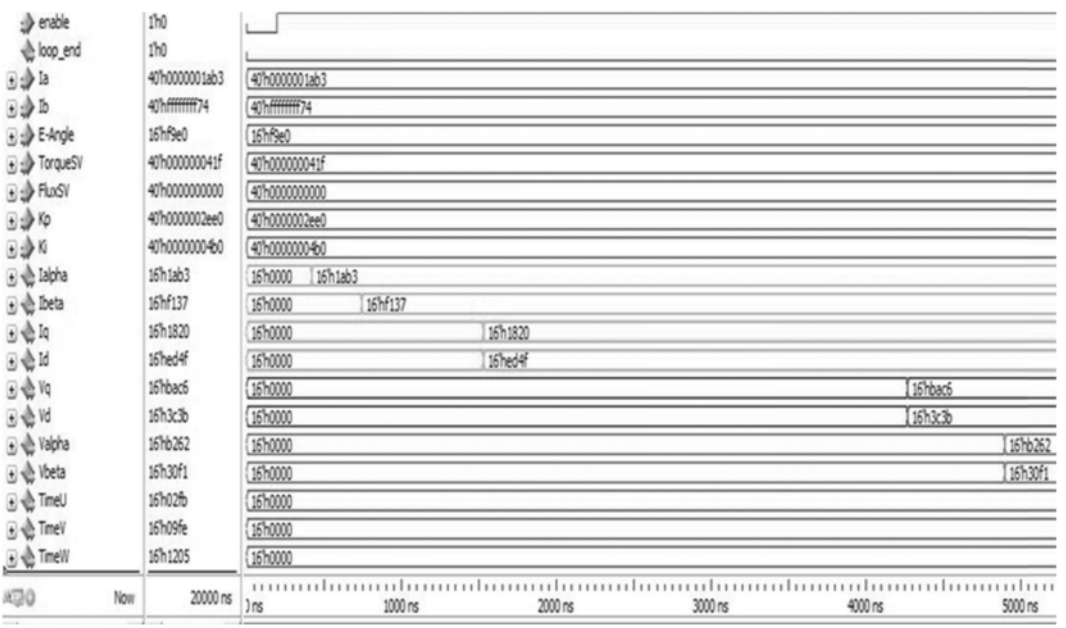

图 6 用基于 FPGA 的 CPU 执行 FOC 计算

(下转第 49 页)

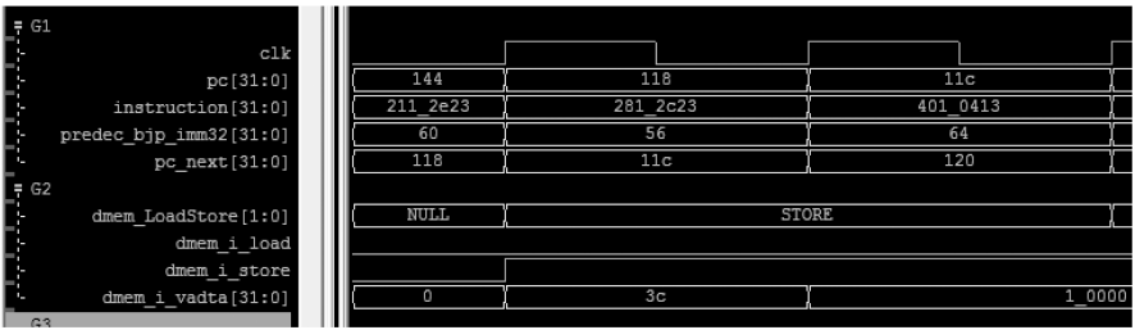

图 11 STORE 指令验证波形

了一种三级流水线的 RISC-V 处理器。使用 RV32I 整数运算指令集对处理器进行了仿真测试，结果表明，所设计的处理器功能正确，达到预定目标。

下一步将通过 SystemVerilog 语言结合 UVM 平台模块的工业级验证方法学，将各个子模块充分测试，实现完备的处理器设计与验证，并在 FPGA 硬件板卡中进行实现应用。

参考文献

[1] 蔡文伟.集成电路设计产业发展现状及对策分析[J].科技经济导刊，2019，27(30)：200-201.

[2] 张晔嘉.国内信息系统自主可控生态环境分析[J].电子质量，2019(7)：58-61.

[3] WATERMAN A，LEE Y，PATTERSON D A，et al.The RISC-V instruction set manual，volume I：User-level ISA[R].CS Division，EECE Department，University of California，Berkeley，2014.

[4] WATERMAN A S.Design of the RISC-V instruction set architecture[R].UC Berkeley，2016.

[5] 詹剑锋.论中国如何发展自主可控和开放的科技产业[J].中国科学院院刊，2019，34(6)：657-666.

[6] 李东泽，曹凯宁，曲明，等.五级流水线 RISC-V 处理器软硬件协同仿真验证[J].吉林大学学报(信息科学版)，2017，35(6)：612-616.

[7] HENNESSY J L，PATTERSON D A.Computer architecture：a quantitative approach[M].Elsevier，2011.

(收稿日期：2020-01-11)

作者简介：

折如义(1965-)，男，本科，副教授，主要研究方向：计算机系统。

李炳辉(1999-)，男，本科，主要研究方向：嵌入式系统。

姜佩贺(1988-)，通信作者，男，博士，讲师，主要研究方向：集成电路设计、芯片安全。

(上接第 43 页)

3 结论

基于 FPGA 的自定义指令集 CPU 可以处理较为复杂的算法，并且计算速度较快。在 FPGA 中构建这种指令集计算部分算法，可以分摊整个系统的算法计算工作量。在一颗 FPGA 芯片中可以构建多个这样的自定义 CPU，达到并行计算、分布式计算的目的。这种 CPU 可以灵活地配置和修改，满足一些特殊指令集的处理。

在实验中，伺服控制算法中的 FOC 算法其计算流程耗时只需要 7.48 μs。这已经比部分 Cortex-M 架构的 ARM 芯片的计算速度要快。

参考文献

[1] 钟瑜，吴明钦.一种高性能并行计算架构的 FPGA 实现[J].电讯技术，2019，59(7)：829-835.

[2] Webb，Warren.Customizable CPU platform combines DSP，FPGA[J].EDN，2008，53(6).

[3] 刘景文.基于 CPLD/FPGA 的 CPU 设计[J].数字技术与应用，2014(2)：162-164.

[4] 尼萨，周维译.计算机系统要素：从零开始构建现代计算机[M].北京：电子工业出版社，2007.

[5] 王培麟.FPGA 架构的 8 位 CISC CPU 设计[J].煤炭技术，2010，29(10)：212-214.

[6] 夏宇闻.Verilog 数字系统设计教程[M].北京：北京航空大学出版社，2008.

[7] 潘松，黄继业.EDA 技术与 VHDL[M].北京：清华大学出版社，2002.

(收稿日期：2020-01-06)

作者简介：

李俊(1975-)，男，硕士，高级工程师，主要研究方向：电源、变频器、伺服系统等开发研究工作。

任连新(1997-)，男，硕士研究生，主要研究方向：电力电子及运动控制技术。

廖振雄(1987-)，男，硕士，工程师，主要研究方向：伺服系统开发研究工作。

版权声明