一种电机 FOC 算法 DSP 系统设计及实现

岳梦云,白 冰
(北京宇航系统工程研究所,北京 100076)

摘 要: 本文设计了一种适用于电机矢量控制算法的数字信号处理系统的微架构定义,包括其指令集定义、存储器模型以及与主 CPU 的交互模式. 该设计具有通过固定部分多操作数有效缩减指令编码长度提高代码密度以及后台执行多周期指令提高 ALU 并行效率的显著优点. 文中给出了典型的 FOC 控制算法在 DSP(Digital Signal Processor)指令集上实现的指令周期数,也给出了对应架构的电路实现情况,最终以 ARM CORTEX-M0 及几款主流 DSP 作为比较基线,通过实测实验数据证明了体系结构的高能效比,以较为有限的电路面积代价,极大提高了集成 DSP 的嵌入式系统的运行效率.

关键词: 电机控制; 嵌入式系统; 矢量控制; 数字信号处理器; 微架构; 指令集
中图分类号: TN453 **文献标识码:** A **文章编号:** 0372-2112 (2020)10-2041-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.10.023

A DSP Design and Implementation for Motor FOC

YUE Meng-yun, BAI Bing
(*Beijing Institute of Astronautical Systems Engineering, Beijing* 100076, *China*)

Abstract: A fully customized digital signal processor for motor field-oriented control is proposed. The instruction set, memory model and interaction method with main-CPU is discussed. Fixing some of the operands in multi-op instructions allows a reduced instruction encoding length and therefore an economical code size. Background execution improves arithmetic and logic unit(ALU) computation parallelism. In this article, the instruction cycles of typical field-oriented control(FOC) algorithm is shown. The silicon implementation of the proposed digital signal processor (DSP) is provided. Taken the commonly used ARM Cortex-M0 processor and several main-stream DSPs as the comparison baseline, the experiment results demonstrate that the micro architecture and circuit implementation of this DSP is energy-efficient and competitive. This approach improves the processing performance at the cost of limited circuit resource.

Key words: motor control; embedded system; field-oriented control(FOC); digital signal processor(DSP); microarchitecture; instruction set architecture(ISA)

1 引言

永磁同步电机(PMSM)矢量控制(FOC)是一种使用正弦信号作为电机相电流,从而使得电机平稳高效运转的电机控制算法. 在 PMSM 中,转子磁场需要与定子电流感应磁场同步且存在稳定相差从而获得最大稳定转矩. 因此需要根据转子位置精确控制定子电流. 随着控制系统的运算能力提升以及应用需求对于能效、噪声等要求的提高,FOC 算法已经日渐成为主流[1~3]. 但应用表现的提升需要以控制算法的复杂度为代价. 在高速运转的应用环境中,要求软件算法在一个 PWM 周期内完成特定的运算和控制操作. 因此,如何低成本、高效率地完成 FOC 算法的软硬件实现成为了一个活跃的研究课题和系统工程问题[4,5]. 目前市面上主流的 MCU 厂商都有集成 DSP 的产品推出,如 ST 的 STM32F3 系列,TI 的 C2000 系列,以及 ADI 的 ADSP-21xx 和 ADSP-CM40x 系列等,定点数 DSP 的实现早在 2000 年左右即已出现并普及,如 TI 的 TMS320C54x 以及 ADSP-2181 等等. 本文后续会针对指令效率和实现成本对不同架构进行对比.

本文提出了一种指令集及微架构全定制的电机驱动 FOC 算法的定点数数字信号处理加速器,该处理器

收稿日期:2018-10-28;修回日期:2020-02-24;责任编辑:蓝红杰

最终以 HHGrace 90nm EFlash Low Power 工艺进行流片，主频 96MHz，实测可以在 1.6μs 内完成一次 PWM 周期的全部运算.

如图 1 所示，典型的 FOC 算法中一个 PWM 周期需要先将采集的相电流根据角度位置估计，经过 Clarke 变换和 Park 变换到电机的旋转坐标系，旋转去耦可以简化对复杂三相电机的控制. 然后计算测量值与控制量之间的差异，经过 PID(Proportion-Integral-Derivative)模块运算之后得到施加于电机的控制电压，再根据角度值，将准直流电压变换回旋转坐标系，通过 SVPWM(Space Vector Pulse Width Modulation)模块计算 PWM 占空比，从而实现通过 MOS 通断产生电机控制电压.

常规运算主要包括低通滤波，坐标系变换的计算包括采样电流值的常量乘、以及三角函数正弦余弦计算等.

$$\begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} I_d \\ I_q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} I_\alpha \\ I_\beta \end{bmatrix} \tag{2}$$

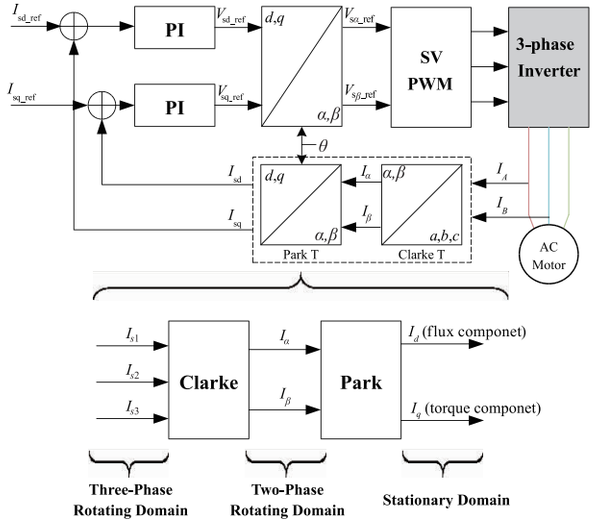


图1 FOC算法框图

PID 主要涉及的运算包括乘累加、累加、跳转分支等，其中为了避免数值溢出需要进行饱和操作. SVPWM 模块包含的分支跳转更多，因此可以考虑使用 DSP 进行处理，也可以使用通用处理器进行处理. 统一使用 DSP 处理的好处是减少了 DSP 和主控 CPU 的数据交互时间.

2 体系架构

作为嵌入式系统的协处理器，本文提出的 DSP 架构不与主处理器共享程序和数据空间，结构框图如图 2 所示. 典型地，以集成了 ARM Cortex-M0 的 MCU 为例，应用程序通常存储于 flash，而运行时数据一般存储在内存，即片上 SRAM 中. 如果采用共享存储的方案，M0 访问 flash/SRAM 会与 DSP 构成总线竞争. 而由于 flash 的最快读取响应时间一般也要在 15ns 以上，对于主频较高的器件，通常无法做到单周期返回取指的指令，SRAM 的读取竞争也无法保证 DSP 每个总线周期取指令一次.

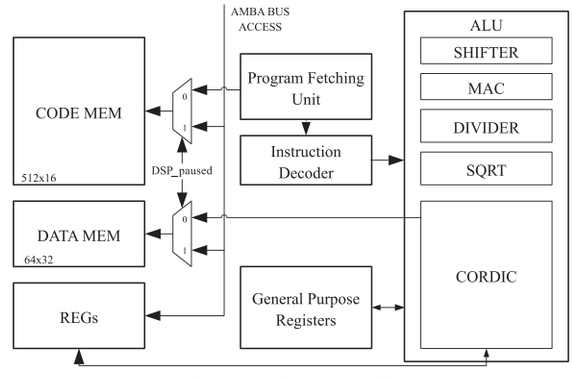


图2 DSP系统结构框图

出于保证 DSP 访存性能的考虑，DSP 采用独立程序和数据存储空间，大小分别为 512×16bit 和 64×32bit. DSP 程序(code mem)和数据(data mem)在 DSP 启动运行前或运行暂停时可以由主控 CPU 进行读写操作，从而实现 DSP 与 CPU 的数据交互，code mem 主要用于存储 DSP 运行时的 DSP 指令，通常只需在 DSP 运行前装载一次，data mem 主要用于存储 DSP 运行时的一些程序变量，以及与主控 CPU 之间的数据交互. 而 DSP 在运行时 code mem/data mem 均不允许 CPU 进行访问操作.

DSP 采用访存取指、译码、执行写回 3 级流水.

根据 FOC 算法所需的运算，DSP 数据通路包括移位、乘累加、除法、开方、三角函数等运算模块. 其中乘累加运算模块在加数为 0 可作为普通乘法器使用，三角函数模块有两种模式，分别为正余弦模式和反正切模式，正余弦模式以角度值 θ 输入，计算并输出 $\sin\theta/\cos\theta$；反正切模式以坐标(x,y)为输入，计算并输出角度值 $\theta=\arctan(y/x)$以及向量模值 $\sqrt{x^2+y^2}$.

为了进一步复用运算单元，DSP 在暂停时，允许 CPU 直接通过总线访问 DSP 内部的除法器、开方器和 CORIDC 模块的寄存器接口，从而实现了 CPU 直接调用 DSP 的运算模块.

2.1 寄存器堆

本文的 DSP 架构中共包含 8 个核心寄存器，其中除 R0 为常 0 寄存器外，都可作通用寄存器使用，用途如表 1 所示. 其中由于受指令编码长度限制，某些多操作数指令

需要固定结果寄存器编码,因此 R6 和 R7 在某些指令中固定用作结果寄存器. 所有寄存器均为 32bit 位宽.

表 1　DSP 架构核心寄存器

寄存器	用途
R0	常 0 寄存器
R1 ~ R5	通用寄存器
R6	通用寄存器/除法商结果寄存器
R7	通用寄存器/MAC 结果寄存器/arctan 角度值结果寄存器

2.2　指令集

针对软件算法需求,经过精简设计,DSP 仅包含 4 种共 16 条指令,占用 4bit 指令编码长度,考虑升级需要,1bit 作为指令编码保留位. 指令集及伪代码如表 2 所示. 其中大部分指令为单周期指令,条件分支指令因为依赖于上一条指令的计算结果,因此需要插入一个指令气泡. 由于某些运算在实际 FOC 算法中出现次数较少,没有必要使用大量电路面积进行单周期实现,处于电路成本的折中考虑,除法指令、开方指令、CORDIC 指令均为多周期指令. 而多周期指令在占用相应运算单元时并不会暂停 DSP 流水线的取指、执行、访存操作. 只是在汇编程序流中不允许对多周期运算单元进行重入调用,但支持其他类型运算指令的并发. 这种方式可以有效消除多周期指令阻塞流水线对运算效率的影响.

DSP 杂项指令 IRQ 用于向 CPU 提起中断,允许在 DSP 程序中暂停运行并将控制权交给 CPU. CPU 在 DSP 暂停时可以访问其数据区域,从而实现二者之间的数据交互. 通常在 DSP 开始运行前,CPU 需要将 DSP 运行时所需数据按照一定的数据结构填入 DSP 的 data mem.

表 2　DSP 指令集

Arithmetic Instruction							
INST	**Rd1**	**Rd2**	**Rs1**	**Rs2**	**Rs3**	**Description**	**Pseudo code**
ADD	Rd1		Rs1	Rs2		addition	Rd1 = Rs1 + Rs2
SUB	Rd1		Rs1	Rs2		subtraction	Rd1 = Rs1-Rs2
MAC	R7	R6	Rs1	Rs2	Rs3	multiply-accumulate	R7 = Rs1 * Rs2 + Rs3
DIV	R6		Rs1	Rs2	Rs3	division	R6 = (Rs1 << 16 + Rs2)/Rs3
ASR	Rd1		Rs1	Rs2		arithmetic right shift	Rd1 = Rs1 >> Rs2
ASL	Rd1		Rs1	Rs2		arithmetic left shift	Rd1 = Rs1 << Rs2
SAT	Rd1		Rs1	Rs2		saturation	If(Rd1 < Rs1) Rd1 = Rs1 ; else if(Rd1 > Rs2) Rd1 = Rs2
SIN_COS	Rd1	Rd2	Rs1			sin and cos	Rd1 = cos(Rs) ; Rd2 = sin(Rs)
ARCTAN	R7	Rd2	Rs1	Rs2		arctan and module	R7 = arctan(Rs2/Rs1) ; Rd2 = sqrt(Rs1^2 + Rs2^2)
SQRT	Rd1		Rs1			square root	Rd1 = sqrt(Rs1)
Memory Instruction							
INST	**Rd1**	**Rd2**	**Rs1**	**Rs2**	**Rs3**	**Description**	**Pseudo code**
LDR	Rd1		Imm			Load data	Rd1 = SRAM[imm]
STR	Rd1		Imm			Store data	SRAM[imm] = Rd1
Branch Instruction							
INST	**Rd1**	**Rd2**	**Rs1**	**Rs2**	**Rs3**	**Description**	**Pseudo code**
JUMP			Imm			Unconditional jump	PC = PC + 4 + Imm
JLE			Rs1	Rs2	Imm	Conditional jump	PC = PC + 4 + Imm if (Rs1 ≤ Rs2)
Miscellaneous Instruction							
INST	**Rd1**	**Rd2**	**Rs1**	**Rs2**	**Rs3**	**Description**	**Pseudo code**
IRQ						Generate IRQ and halt DSP	

3　Python 模拟器

本项目使用 PyCharm 作为 IDE 进行了 DSP Python 模拟器设计. 编写 Python 模拟器的目的是为了衔接 Keil C 工程与 RTL 仿真的调试,模拟器同时也完成了自主指令集 DSP 指令汇编程序到机器码的翻译,汇编程序的调试,运行过程寄存器数值的跟踪输出. Python 模拟器主体,包含 inst_parser(inst) 和 inst_translater(inst) 两个函数,分别完成一条指令的运行模拟和机器码翻译,需要注意的是 inst_parser 会在指令运行中维护修改 GPR(R1, R2,…,R7)以及 PC,由于存在跳转,因此并不是在汇编指令流中顺序执行;而机器码的翻译是对汇编指令流的顺序处理,翻译完成的机器码会输出到文件. Testbench 主体,完成模拟器中指令模拟和指令翻译的调用,完整的汇编程序仅仅对应一次 FOC PWM 周期,因此 testbench 会循环调用 inst_parser 以模拟多个 PWM 周期的运行情

况. 而 inst_translater 仅需被 testbench 调用一次来完成指令翻译,翻译好的机器码用于后续 RTL 仿真时 code mem 的初始化,在实际应用环境中通过 MCU 由用户将 code/data 数据分别写入 code mem/data mem.

通过 DSP 模拟器主体程序完成汇编指令到机器码的翻译并产生二进制/十六进制仿真文件的产生,此文件需要由 CPU 装载进 DSP code mem,但只需装载一次,可以 const 数组写入 Keil 工程. 同时,模拟器还可以提供全部 DSP 核心寄存器在每个指令周期的数值,从而便于与 RTL 设计进行核对. 模拟器还支持断点和寄存器改写等调试指令.

4 实验及结论

4.1 性能评估

通过 RTL 仿真可以获得 DSP 运行的精确周期数,如表 3 所示. 其中 SVPWM 不是计算密集型任务,而是控制密集型,本文最终选择使用主控 CPU 执行 SVPWM 函数. 但每次 CPU 接管 DSP 数据,都会因上下文切换而引入 0.4μs 左右的切换时间.

表 3 DSP 执行各任务运行周期数

模块	指令数	调用次数	总周期数
LowPass filter	9	4	36
Clarke	5	1	5
Park	9	1	9
PID	23	2	46
Rev Park	10	1	10
SVPWM	47	1	47
Over all	153		

目前,DSP 运行在 96MHz 时钟频率,为了便于进行性能评估,本文使用通用定时器进行计时,定时器工作于 96MHz 时钟,开启 DSP 前进行一次定时器计数值读取,结束再进行一次定时器计数值读取,一次 MCPWM 周期的运算约需 1.6μs,与 RTL 仿真数据吻合. 而相应地,将 FOC 控制代码由 Cortex-M0 执行,同时在程序起止进行时间戳对比,得到 ARM Cortex-M0 运行同样程序需要 22μs,与 RTL 仿真结果吻合. 由于 Cortex-M0 不具备硬件除法指令,如果通过软件来模拟除法指令,根据操作数不同,消耗的指令周期数从 100 到几百不等.

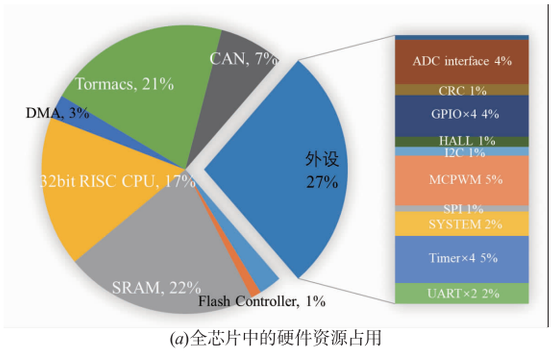
如表 4 所示,除了与 Coretex-M0 的软件实现进行性能对比之外,本文选取了 3 款市面上的广泛使用的集成 DSP 的 MCU 进行对比. 其中 TI 的 C2000 TMS320F28004x Piccolo 微控制器为入门级 32 位 DSP 控制器; TMS320C54x 为 TI 早期的 16 位定点数 DSP; ADSP-2181 为 ADI 早期的 16 位定点数 DSP,对照数据均来源于 TI 官方网站(www. ti. com)和 ADI 官方网站(www. analog. com). 需要注意的是 TMS320F28004x 的三角函数等运算是基于浮点数的,共享 FPU 寄存器. 在架构设计上,TMS320C54x 和 ADSP-2181 为多条并行总线设计,架构为 memory based addressing mode. 而 TMS320F28004x 与本文的架构较为类似,多个主设备分别配备独立总线,在外设接口处设置总线仲裁. 算术逻辑路径的延时或流水线级数往往是由制造工艺决定的. 从主频上看,本文的实现达到市面入门级主流 DSP 水平,同时由于可以完成单周期取指,因此 MIPS 指标与主流 DSP 处理器持平. 在早期的定点数 DSP 不支持三角函数、除法、开方等操作的,在表格中以“-”表示. 其余算子或计算任务通过执行周期数进行比较. TMS320F28004x 具备 TMU,并可以进行三角函数、1/x、开方等算术运算,指令周期数要优于本文提出的 DSP 架构. 但具体到类似 Park 变换的计算任务,本文的 DSP 架构甚至表现更优. 考虑到本文提出的 DSP 架构与 M0 硬件规模相当,作为 MCU 协处理器集成实现后成本应该与国产主流 M0 内核 MCU 相当,成本可以控制在 ¥3.00 以内,而 TI 和 ADI 的 DSP 器件,即使按照毛利率 60% 计算,成本仍在 $2.00 以上.

表 4 不同 DSP 架构比较

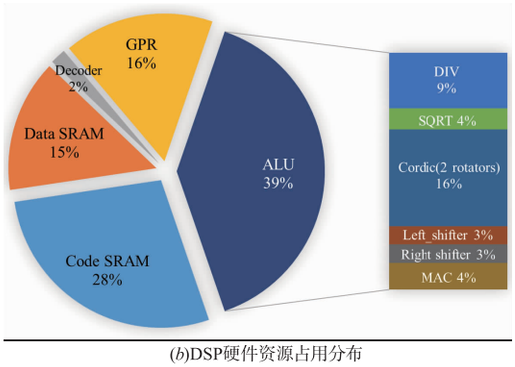
Device	TMS320F-28004x	TMS320-C54x	ADSP-2181	Cortex-M0	This thesis
Freq/MHz	100	40 ~ 120	40	–	96
#Regs	4 ×32bit 2 ×16bit 8 ×32bit fp	Mem based	Mem based	16 ×32 bit	8 ×32 bit
mac	1	1	1	2	1
shift	1	1	1	1	1
sin/cos/arctan	4	–	–	–	8
div	5	–	–	100 ~ 500	10
sqrt	5	–	–	–	8
Park	13	–	–	110	9
Price	$5	$30	$40	–	

4.2 实现情况

本文提出的 DSP 架构最终集成于一款电机驱动专用 MCU 芯片中,原型系统首先使用基于 Xilinx Kintex7 的 FPGA 进行验证,确保 CPU + DSP 的设计方案功能正确. 芯片使用 HHGrace 90nm EFlash Low Power 工艺设计制造,模拟部分使用 Cadence 工具链进行设计验证,数字部分使用 Synopsys 工具链进行设计验证及 APR,部分 IP 在 foundry 进行 merge. 全芯片和 DSP 的硬件资源占用情况如图 3 所示. 管芯尺寸约为 2mm ×2.2mm,实测运行功耗约 27mW,休眠功耗达 μA 级别,通过 -40℃到 125℃的汽车级高低温环境实验测试及 30kV 静电测试和 ±5kV 群脉冲测试.



(*a*)全芯片中的硬件资源占用



(*b*)DSP硬件资源占用分布

图3　硬件资源占用情况

通过表5对比可见,本文提出的DSP架构与中低端32bit嵌入式CPU核心的硬件规模相当. 其中由于需要进行通用CPU不具备的算术操作,DSP庞大的ALU占掉了40%的DSP模块面积,其次是程序存储器(code mem)和数据存储器(data mem),虽然从实现角度看,程序存储器和数据存储器位宽和深度并不大,经过专门的优化,指令和数据密度也已经达到较为满意的程度. 但分立的存储器由于灵敏放大电路等与存储器本身容量无关的电路组成的存在,仍在DSP中占据较大的比重. 因此访存共享是未来的一个可探索方向. 近期比较活跃的RISC-V,由于支持指令集扩展,可以将特定的

表5　芯片实现逻辑单元面积占比

模块	逻辑单元面积/μm^2	百分比
BIST	17361	2. 50%
Flash Controller	7770	1. 12%
SRAM	145674	21. 02%
32bit RISC CPU	113761	16. 42%
DMA	18000	2. 60%
Analog Front End	3234	0. 47%
ADC interface	30155	4. 35%
CAN	48962	7. 07%
CRC	7602	1. 10%
This DSP	138649	20. 01%
GPIO ×4	28519	4. 12%
HALL	6869	0. 99%
I2C	6119	0. 88%
MCPWM	33976	4. 90%
SPI	4127	0. 60%
SYSTEM	16789	2. 42%
Timer ×4	32196	4. 65%
UART ×2	14294	2. 06%
Others		2. 62%
总计	692921	100%

ALU数据通路加入到通用CPU之中,节省掉CPU和外部DSP之间的数据交互上下文切换时间,同时可以做到程序和数据空间的共享,可以比较有效地提升性能面积效率. 因此,也是电机控制DSP微架构未来值得探索的方向之一.

芯片最终实现版图利用率达80%以上. 芯片采用TQFP48封装,为验证其有效性,本文以该芯片为控制核心进行了工业风机控制器的设计开发,并针对顺风启动、逆风启动等不同启动模式进行了测试优化,同时优化负载变化时的共振现象,以稳定机身,减少噪声等. 风机启动电流采样曲线如图4所示. 实验室环境如图5所示. 充分利用DSP模块,可以极大减轻CPU计算任务负担,同时压缩PWM周期,测算可支持最大转速达10krpm以上.

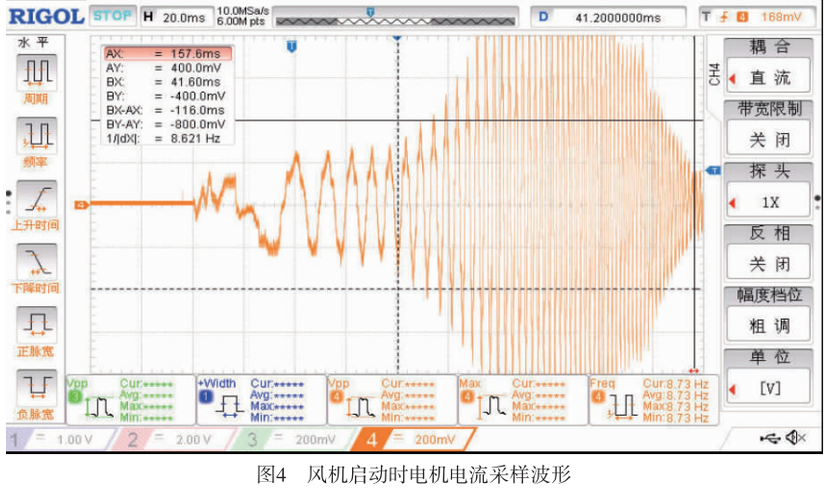


图4　风机启动时电机电流采样波形

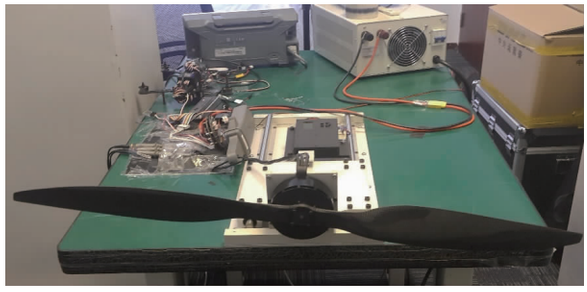


图5 风机实验台

5 结论

本文提出了一种 FOC 算法的电机控制系统 DSP，从指令集体系架构入手进行设计，与典型嵌入式 MCU 相比取得了 10 倍以上的性能提高，以及优秀的能量效率和面积效率，并通过实际实验验证了系统的有效性.

参考文献

[1] Tajima H，Hori Y. Speed sensorless field-orientation control of the induction machine[J]. IEEE Transactions on Industry Applications，2002，29(1)：175 – 180.

[2] Gaolin Wang，et al. A robust speed controller for speed sensorless field-oriented controlled induction motor drives [A]. IEEE Vehicle Power & Propulsion Conference[C]. Harbin：IEEE，2008. 1 – 4.

[3] Nihat Inanc. A robust sliding mode flux and speed observer for motor drives[J]. Electric Power Systems Research，2007，77(12)：1681 – 1688.

[4] Prasad Shrawane. Indirect field-oriented control of induction motor[A]. IEEE International Power Electronics Congress[C]. Sapporo，Japan：IEEE，2010. 102 – 105.

[5] Li Yu，et al. Simulation of PMSM field-oriented control based on SVPWM[A]. 29th Chinese Control and Decision Conference[C]. Chongqing，China：IEEE，2017. 7407 – 7411.

作者简介

岳梦云 女，1988 年 11 月出生于安徽合肥. 2013 年毕业于清华大学微电子所. 现为北京宇航系统工程研究所工程师，主要研究方向为控制集成电路设计.
E-mail：moonfish94@126.com

白 冰 女，1981 年 8 月出生于黑龙江大庆. 2006 年毕业于哈尔滨工业大学，现为北京宇航系统工程研究所高级工程师，研究方向为地面测发控设计.