Application Note

显示 C2000™ 控制 MCU 优化信号链的实时基准测试

TEXAS INSTRUMENTS

摘要

实时控制的关键是减少感应、处理和驱动的延迟（此三者一并定义为实时信号链）。许多软件基准测试仅专注于处理方面，这方面的数据通常以每秒百万条指令 (MIPS) 表示，而不会充分考虑外设、CPU 和协处理器之间的交互。这样的基准测试无法完全体现系统的实时性能水平。本应用报告介绍了围绕真实控制应用程序创建的实时基准测试，其中突出了实时控制的复杂性以及对于这种更全面的基准测试方法的需求。实时基准测试中的数据可用于深入了解 C2000™ 控制 MCU 的特性，正是这些特性使该器件成为适合实时控制应用程序的出色平台。

内容

1 引言 3
2 ACI 电机控制基准测试应用程序 3
2.1 源代码 3
2.2 TMS320F28004x 的 CCS 项目 4
2.3 TMS320F2837x 的 CCS 项目 5
2.4 验证应用程序行为 6
2.5 基准测试方法 7
2.6 用于分析应用程序的 ERAD 模块 8
3 实时基准测试数据分析 9
3.1 ADC 中断响应延迟 10
3.2 外设访问 11
3.3 TMU（数学增强）影响 12
3.4 闪存性能 13
3.5 控制律加速器 (CLA) 14
4 C2000 价值定位 17
4.1 高效执行信号链，使实时响应比计算速度更高的 MIPS 器件更好 18
4.2 具有低延迟的出色的实时中断响应 19
4.3 外设紧密集成，可扩展具有大量外设访问的应用 19
4.4 最优三角函数引擎 21
4.5 多功能性能提升计算引擎 (CLA) 22
4.6 由于执行差异小而导致确定性执行 23
5 总结 24
6 参考文献 24
7 修订历史记录 25

插图清单

图 1-1. 实时信号链 3
图 2-1. ACI 电机基准测试应用程序方框图 3
图 2-2. α 相电流 (DualTimeA) 和单位电机转速 (DualTimeB) 的图形视图 6
图 2-3. ACI 电机基准测试输出 7
图 2-4. 实时基准测试阶段 8
图 2-5. 具有 ERAD 分析的应用程序统计信息的脚本输出 9
图 3-1. ACI 电机基准测试中断响应分量 10
图 3-2. ACI 电机基准测试中断响应基准测试数据 10
图 3-3. ACI 电机基准测试的外设访问基准测试数据 11
图 3-4. 使用 TMU 和 FastRTS 在 RAM 上的控制环执行时间 12
图 3-5. 针对 TMU 和 FastRTS 编译的应用程序大小 12
图 3-6. TMS320F28004x 从 RAM 和闪存执行 的时间 13

图 3-7. TMS320F2837x 从 RAM 和闪存执行的时间 13
图 3-8. 用于 CLA 执行整个信号链的 ACI 电机基准测试输出 15
图 3-9. 具有 CLA 转移功能的 C28x 的 ACI 电机基准测试输出 17
图 4-1. ACI 电机控制基准测试执行（相对周期和相对时间） 18
图 4-2. ADC 中断响应延迟（相对周期和相对时间） 19
图 4-3. ADC 读取并转换为浮点的效率（相对周期和相对时间） 20
图 4-4. 使用三角函数引擎进行 Park 变换（相对周期和相对时间） 21
图 4-5. ACI 电机控制基准测试 CLA 执行（相对周期和相对时间） 22
图 4-6. 从闪存执行 ACI 电机控制基准测试的差异（相对周期和相对时间） 23

表格清单

表 3-1. F28004x ADC INT 响应持续时间的理论估计值 11
表 3-2. 实际控制应用的外设访问估计周期 12
表 3-3. 闪存执行性能与 RAM 执行性能的比较 14
表 3-4. 用于 F28004x 上 CLA 和 C28x 的 ADC 中断响应延迟 15
表 3-5. F28004x 上 CLA 和 C28x 的外设访问延迟 15
表 3-6. F28004x 的 CLAmath 性能 16

商标

1 引言

通过实时基准测试可全盘了解信号链性能情况。如图 1-1 所示，这包括中断响应时间、上下文保存开销、外设读写以及控制算法，所有这些共同构成了实时控制操作。

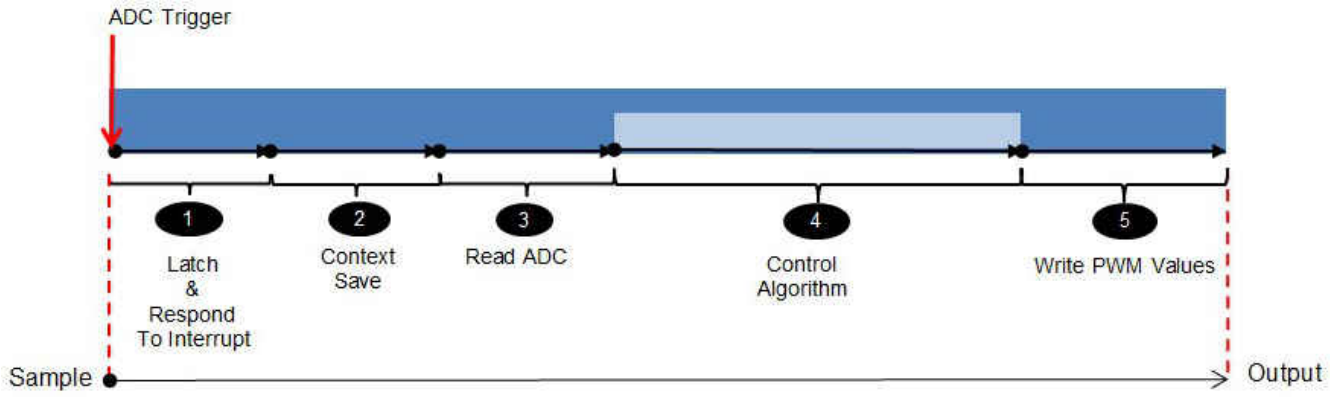


图 1-1. 实时信号链

C2000 架构具有独特的功能，通过最小化采样到输出的时间，使得其成为适用于实时控制的微控制器。本文档中的基准测试演示了这些 C2000 特性。

2 ACI 电机控制基准测试应用程序

ACI 电机控制基准测试模拟无传感器交流感应电机控制应用程序。该应用程序执行所有典型操作：模数转换器 (ADC) 读取操作（以感应相电流）、作用于感测电流的转换模块操作、PWM 写入操作（以控制相电压）。无需特殊的外部硬件即可提供激励，因为该应用程序具有可对感应电机的行为进行建模的代码块。为了模拟闭环行为，电机模型的预期电流通过 DAC 模块馈入 ADC 中。单个 ADC 配置为通过两个通道依次感应 A 相电流和 B 相电流。未感应 C 相电流，因为可以从 A 相电流和 B 相电流得出该值。三个 PWM 写入操作模拟控制三相 A 、B 和 C 电压的占空比。图 2-1 表示基准测试应用程序的控制循环中断例程中的执行块。控制循环中断以 2KHz 的速率触发，并在应用程序终止之前执行 1024 次控制循环中断例程的迭代。

在此框图中，深灰色框表示外设访问，浅灰色框表示控制算法，白色框表示代码块，代码块不是真正的 ACI 电机控制应用程序的一部分，而是在基准测试中用于模拟电机的行为。

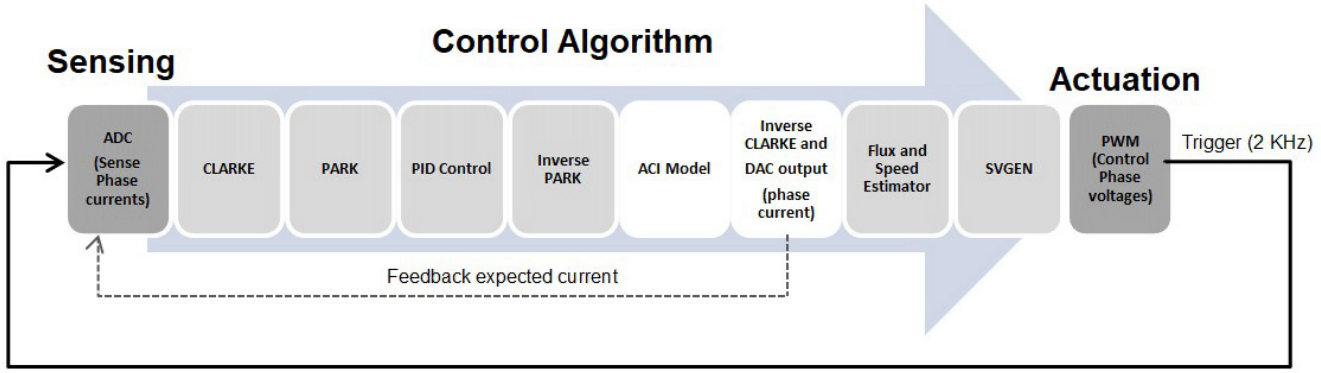


图 2-1. ACI 电机基准测试应用程序方框图

2.1 源代码

C2000Ware 中提供了软件基准测试示例。

- C2000Ware **3.04.00.00** 及更高版本中提供的 **C28x** CPU 基准测试
- C2000Ware **4.00.00.00** 及更高版本中提供的 **CLA** 加速器基准测试。

可在 C2000Ware 安装程序中的以下位置找到该示例。

<C2000Ware>\examples\demos\benchmark\aci_motor_benchmark

顶层文件夹为：

- 常用
- device_support
- f28004x
- f2837x

源代码位于“*common*”和“*device_support*”文件夹中。“常用”文件夹包含器件独立代码，例如变换算法、电机建模代码等。*“device_support”*文件夹在为器件命令的子文件夹中包含专用于特殊器件的代码，例如器件配置、外设读取/写入等等。

该应用程序在在 Code Composer Studio™ (CCS) 中支持 TMS320F28004x and TMS320F2837x 器件 ，并且可以在 TMS320F28004x Launchpad 以及 TMS320F2837x Launchpad 上执行。相应的 CCS 项目位于 *“f28004x”* 和 *“f2837x”* 顶层文件夹内的子文件夹 *“ccs”* 中。

该应用程序有多个实现变体：一个变体使用数学引擎三角函数加速器 (TMU) 来执行 Park、逆向 Park 和通量估算器控制算法所需的三角计算，另一个变体使用称为 FastRTS 的软件库来执行三角计算。FastRTS 库包含在 C2000Ware 中，该库和文档可在 <C2000Ware>\libraries\math\FPUfastRTS\c28\ 中找到。与基于软件库的实现方案相比，这两个变体的目标是展示 TMU 可以提供的数学引擎性能提升程度。

另一组实现变体涉及 CLA。一种变体是 C28x CPU 将部分计算转移载到 CLA，另一种变体是基准测试控制代码完全仅从 CLA 执行。这两个变体的目标是展示如何使用 CLA 来帮助实现实时目标。

2.2 TMS320F28004x 的 CCS 项目

可以使用“导入项目”选项将项目导入 CCS，并选择位于*“f28004x\ccs”*中的项目。该项目带有几个预定义的构建配置。项目导入后，可以用于构建不同的构建配置同，如下所述：

- ***SignalChain_RAM_TMU：***该应用程序使用 TMU 指令对在 RAM 外执行的完整信号链进行编译，用于控制算法 (例如使用三角函数的 Park 和逆向 Park。构建设置由 SIGNAL_CHAIN=1 和 USE_FAST_TRIG_LIB=0 定义控制。
- ***SignalChain_FLASH_TMU：***该应用程序针对与 SignalChain_RAM_TMU 类似的完整信号链进行编译，不同之处在于它是在闪存外执行的。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=0 和 _FLASH 定义控制。
- ***SignalChain_RAM_FastRTS：***该应用程序使用 FastRTS 而不是 TMU 对在 RAM 外执行的完整信号链进行编译，用于控制算法 (例如使用三角函数的 Park 和逆向 Park。构建设置由 SIGNAL_CHAIN=1 和 USE_FAST_TRIG_LIB=1 定义控制。
- ***SignalChain_FLASH_FastRTS：***该应用程序针对与 SignalChain_RAM_FastRTS 类似的完整信号链进行编译，不同之处在于它是在闪存外执行的。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=1 和 _FLASH 定义控制。
- ***ControlAlgo_RAM_TMU：***该应用程序仅针对控制算法进行编译，而不针对完整信号链进行编译，也就是说，在此构建配置中不会配置外设，并且不会产生中断。这可以作为将该应用程序导入到其他器件的起点。构建设置由 SIGNAL_CHAIN=0 和 USE_FAST_TRIG_LIB=0 定义控制。

- ***SignalChain_RAM_TMU_CLA_OFFLOAD：***该应用程序使用 TMU 指令在 RAM 外执行的完整信号链进行编译，用于为控制算法（例如在 C28x CPU 端使用三角函数的 Park 和逆向 Park。C28x CPU 将一些控制代码计算转移到 CLA，从而实现并行执行。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=0 和 CLA_OFFLOAD 定义控制。
- ***SignalChain_FLASH_TMU_CLA_OFFLOAD：***该应用程序对与 SignalChain_RAM_TMU 类似的完整信号链进行编译，不同之处在于 C28x CPU 代码是在闪存外执行的。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=0、CLA_OFFLOAD 和 _FLASH 定义控制。
- ***SignalChain_RAM_CLAmath_CLA：***该应用程序使用 CLAmath 库从 CLA 在 RAM 外执行的完整信号链进行编译，用于为控制算法（例如使用三角函数的 Park 和逆向 Park），与 C28x CPU 不同，CLA 加速器没有 TMU。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=0 和 CLA_CPU 定义控制。

构建应用程序后，将应用程序加载到目标并在 CCS 中选择 *运行*选项以执行该应用程序。

本应用报告中使用了七个 信号链构建配置中的数据来比较 TMU 与 FastRTS 执行情况、RAM 与闪存执行情况以及 CLA 执行情况。

2.3 TMS320F2837x 的 CCS 项目

通过使用“导入项目”选项并选择位于“*f287x\ccs*”中的项目将项目导入 CCS 中。该项目带有几个预定义的构建配置。项目导入后，可以为不同的构建配置来构建项目，描述如下：

- ***SignalChain_RAM_TMU：***该应用程序使用 TMU 指令对在 RAM 外执行的完整信号链进行编译，用于控制算法（如使用三角函数的 Park 和逆向 Park）。构建设置由 SIGNAL_CHAIN=1 和 USE_FAST_TRIG_LIB=0 定义控制。
- ***SignalChain_FLASH_TMU：***该应用程序对与 SignalChain_RAM_TMU 类似的完整信号链进行编译，不同之处在于该信号链是在闪存外执行的。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=0 和 _FLASH 定义控制。
- ***SignalChain_RAM_TMU_CLA_OFFLOAD：***该应用程序使用 TMU 指令对在 RAM 外中执行的完整信号链进行编译，用于控制算法（例如在 C28x CPU 端使用三角函数的 Park 和逆向 Park）。C28x CPU 将一些控制代码计算转移载到 CLA，从而实现并行执行。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=0 和 CLA_OFFLOAD 定义控制。
- ***SignalChain_FLASH_TMU_CLA_OFFLOAD：***该应用程序对与 SignalChain_RAM_TMU 类似的完整信号链进行编译，不同之处在于 C28x CPU 代码是在闪存外执行的。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=0、CLA_OFFLOAD 和 _FLASH 定义控制。
- ***SignalChain_RAM_CLAmath_CLA：***该应用程序使用 CLAmath 库从 CLA 编译在 RAM 外执行的完整信号链，用于控制算法（例如使用三角函数的 Park 和逆向 Park），与 C28x CPU 不同，CLA 加速器没有 TMU。构建设置由 SIGNAL_CHAIN=1、USE_FAST_TRIG_LIB=0 和 CLA_CPU 定义控制。

构建应用程序后，进入 CCS 菜单选项“*Tools->Debugger Options->Auto Run and Launch Options*”，并取消选中“*On program load or restart*”选项，以自动运行到主符号。之所以需要进行此设置，是因为此器件上的断点有限，并且应用程序中的 printf 消息会使用需要使用断点资源的 CIO。因此，需要禁用“Auto run to main”选项才能让应用程序正常运行。加载应用程序并选择运行选项以执行该应用程序。

本应用报告中使用了五种构建配置中的数据来比较 RAM 与闪存执行情况（因为 F2837x 具有与 F28004x 不同且性能更高的闪存），并了解 CLA 执行情况。

2.4 验证应用程序行为

该应用程序在终止之前执行控制循环 1024 次迭代。应用程序将每次执行迭代获得的 α 相电流以及单位电机转速保存在缓冲区 DLogCh1 和 DLogCh2 中。这些保存的值可以绘制在 CCS 视图中。这些图形的与在实际的无传感器交流感应电机应用中生成的曲线相似。要检查该应用程序是否已正确执行，请在测试完成后暂停目标，然后打开“*Tools->Graph->Dual Time*”来绘制保存的值，点击“*Import*”，并选择位于顶层文件夹“*f28004x*”中的图形属性文件“*aci.graphProp*”。相同的图形属性文件也可以用于在 TMS320F28004x 或 TMS32F2337x 目标上执行的任何构建配置应用程序。α 相电流在 DualTimeA 图中绘制，单位电机转速在 DualTimeB 图中绘制。X 轴具有 1024 个样本中每个样本的数据点，Y 轴具有每个数据点相应的值。图形曲线应类似于图 2-2，其中 α 电流呈正弦曲线状，而电机速度收敛至 0.5 个单位的期望值。

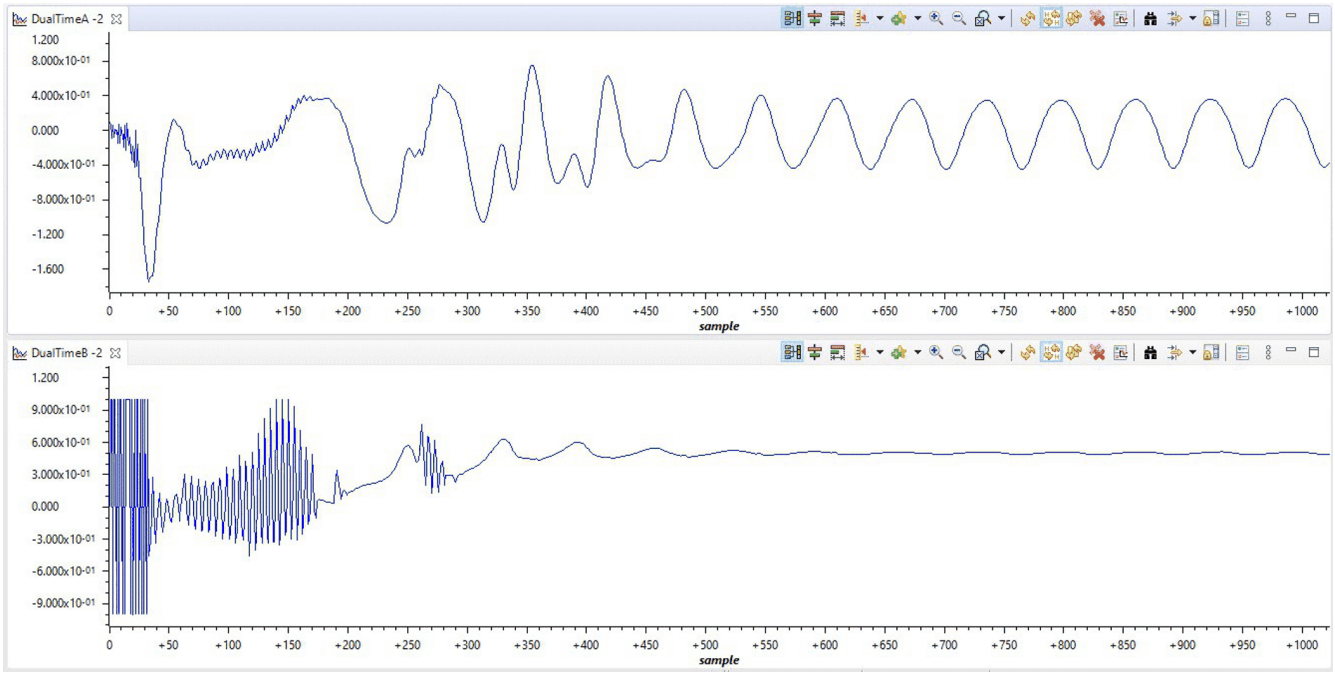


图 2-2. α 相电流 (DualTimeA) 和单位电机转速 (DualTimeB) 的图形视图

2.5 基准测试方法

应用程序使用定时计数器和 PWM 计数器在周期计数中对应用程序的各个部分进行基准测试。因此，不需要像示波器之类的外部硬件来测量基准测试结果。为 1024 个控制循环执行迭代中的每个迭代收集基准测试数据，并通过应用程序中的 printf 消息在控制台窗口上输出这些数据。器件类型、CPU 时钟频率和 RAM 或闪存（具有等待状态）执行信息与每个应用程序执行块的周期计数（平均值、最大值、最小值）一起输出。图 2-3 显示了使用 TMU 时从 RAM 在 F28004x 上执行的示例输出。

```
ACI Benchmark Test:
F28004x CPUCLK = 100000000 Hz, from RAM, using TMU


Execution in cycle counts (avg, max, min) over 1024 iterations

------------------------------------------------------------------------
                                        AVG     MAX     MIN
 INT Response (trigger to ISR entry) :  49      53      47
 Read 2 ADC, convert float           :  3       3       3
 Clarke Transform                    :  11      11      11
 3 PID Controller Transforms         :  107     107     107
 Inverse Park Transform              :  23      23      23
 Flux Estimator                      :  167     168     166
 Speed Estimator                     :  66      67      66
 Park Transform                      :  18      18      18
 SVGen Transform                     :  75      82      66
 Write 3 PWM                         :  10      10      10
------------------------------------------------------------------------
 Total                               :  529     542     517
------------------------------------------------------------------------

Motor Speed at end of test (expected 0.5) : 0.500923

Test program end.
```

图 2-3. ACI 电机基准测试输出

2.5.1 使用计数器进行基准测试的详细信息

实时基准测试需要测量信号链不同阶段的性能。如图 2-4 所示，此过程大致可以分为三部分：硬件响应、编译器生成的上下文保存以及中断服务例程 (ISR) 内的用户代码。

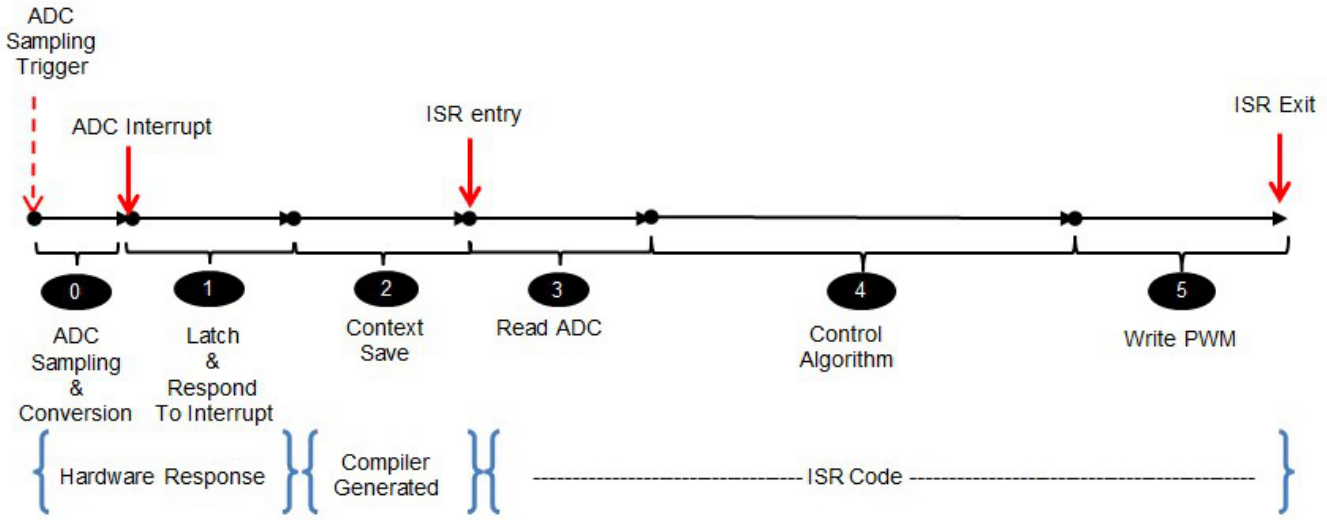


图 2-4. 实时基准测试阶段

衡量 ISR 内部的执行情况是一项简单的任务。通过在代码块前后插入用于读取定时计数器的代码段并计算差值，可使定时器外设对 ISR 内部的代码进行基准测试。应用程序使用这种方法测量控制代码以及 ADC 读取和 PWM 写入。然而，这种方法无法保存编译器生成的上下文和测量硬件响应。

在此特定应用程序中，当 PWM 计数器达到某个值（时基周期）时使用 PWM 来触发 ADC 采样，并在采样完成时， ADC 反过来产生中断。虽然无法单独测量硬件响应和保存编译器生成的上下文，但可以通过在 ISR 代码首次运行时读取 PWM 计数器并计算读取的 PWM 值与触发 ADC 采样时的 PWM 值之间的差值，来一起测量这两者。这就是应用程序测量 INT 响应（硬件响应 + 编译器上下文保存）的方法。

2.6 用于分析应用程序的 ERAD 模块

F28004x 器件具有一个称为 ERAD（嵌入式实时分析和诊断）的硬件模块，该模块具有可用于分析应用程序的比较器和计数器。“*f28004x*”目录包含一个名为“*aci_stats_using_erad.js*”的脚本，该脚本演示了如何在不修改应用程序的情况下以非侵入方式使用 ERAD 来测量和分析应用程序的执行情况。该脚本可用于验证发生中断的次数（应为 1024），还可以作为示例来说明如何使用 ERAD 测量从 ADC 中断产生到 ISR 进入以及 ISR 用户代码执行的执行周期。该脚本仅用于 ERAD 演示和应用程序验证。此脚本生成的测量数据不用于本文档的实时基准测试分析中。

备注

该脚本仅适用于 F28004x SignalChain_RAM_TMU 配置。由于在该脚本中分析的 ISR 地址采用了硬编码来进行“SignalChain_RAM_TMU”配置的构建，因此需要修改脚本才能对为另一配置构建的应用程序进行基准测试。

按如下步骤使用该脚本：

1. 将为“*SignalChain_RAM_TMU*”配置编译的应用程序加载到目标。
2. 在 CCS 中打开脚本控制台视图。
3. 输入以下命令来启动脚本：

   loadJSFile "C:\TI\c2000\C2000Ware_3_04_00_00\examples\demos\benchmark\aci_motor_benchmark\f28004x\aci_stats_using_erad.js"
4. 该脚本将打印出全部为 0 的初始测量值。
5. 运行目标，并在测试完成后停止目标。
6. 目标停止后，脚本将输出捕获的测量值。

图 2-5 显示了脚本输出。

```
js:> loadJSFile "C:\TI\c2000\C2000Ware_3_04_00_00\examples\demos\benchmark\aci_motor_benchmark\f28004x\aci_stats_using_erad.js"

Stats at start of test:
INT occurrence count = 0     INT to ISR cycle count (max) = 0    ISR cycle count(max) = 0

Stats at end of test:
INT occurrence count = 1024      INT to ISR cycle count (max) = 39   ISR cycle count(max) = 2335

js:>
```

图 2-5. 具有 ERAD 分析的应用程序统计信息的脚本输出

备注

ERAD 模块从 ADC 中断产生到 ISR 进入测量“INT 到 ISR 周期计数”。这不包含 ADC 采样时间和 ADC 触发时间，这些时间包含在应用程序代码测得的“INT 响应（触发到 ISR）周期计数”中。因此，ERAD 脚本测得的“INT 到 ISR 周期计数”比应用程序代码测得的“INT 响应（触发到 ISR）周期计数”少约 10 个周期，这个周期等于 ADC 采样和触发持续时间。

3 实时基准测试数据分析

如前几节所述，该应用程序的代码分析信号链的不同部分，并将测得的执行情况以周期计数的形式输出到控制台。对这些数字的分析将展现 C2000 架构的特性，以便对其进行优化用于实时控制。这一分析还将揭示实时基准测试方法对于真实评估系统实时性能的重要性。

备注

本章介绍的 TI C2000 器件基准测试数据是使用 2021 年在 C2000Ware v3.04.00.00 中发布并在 TMS320F280049C Launchpad 和 TMS320F28379D Launchpad 上执行的应用程序收集的。该应用程序使用 Code Composer Studio v10.1.0.00010 执行，并使用 TI v20.2.1.LTS 编译器编译，其中的优化设置为推荐级别 - 2（全局优化），并进行优化以用于速度 - 5 。该应用程序以 COFF 格式进行编译。

3.1 ADC 中断响应延迟

中断延迟是了解实时系统响应时间的重要因素。评估系统中断延迟的典型方法是计算硬件响应中断并跳转到中断矢量（硬件锁存和响应）所需的周期数。但是，在实时应用中，此时间只是响应延迟的一部分。

考虑到无传感器 ACI 电机基准测试应用程序，PWM 促使 ADC 开始采样。ADC 需要一定数量的周期才能完成采样或转换，具体数量取决于系统配置以及 ADC 功能。ADC 准备就绪后，将产生中断。在 ACI 电机基准测试中，ADC 配置为产生早期中断，即在采样完成之后以及转换开始时（而不是转换结束时）产生中断。ADC 产生中断后，硬件对中断做出响应，并跳转到中断矢量。然而，中断服务例程不会立即开始执行用户代码，而是在此之前必须执行一些编译器生成的上下文保存，然后再执行用户代码。

实际上，如图 3-1 所示，实际中断延迟包含四个部分：触发延迟 + ADC 采样持续时间 + 硬件锁存和响应 + 编译器生成的上下文保存。

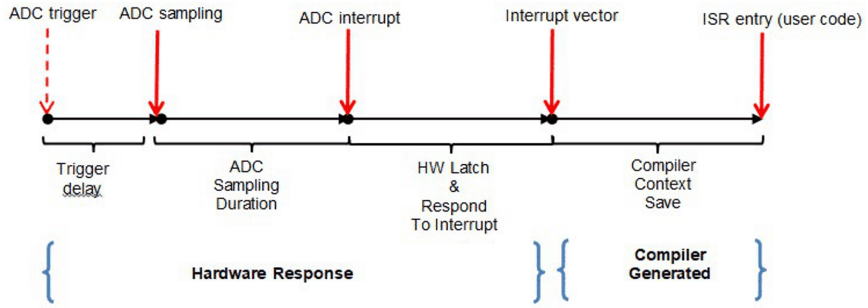


图 3-1. ACI 电机基准测试中断响应分量

ACI 实时基准测试将此值输出为“INT Response (trigger to ISR entry)”。从图 3-2 中可以看出，C28x INT 响应时间平均为 49 个周期，表示在无传感器 ACI 电机示例中，应用程序开始响应感测事件的实际延迟。传统基准测试不会以此方式测量系统。INT 响应周期在最小值和最大值之间的差异是可以预期的，因为在 CPU 接收异步中断事件之前，可能还需要额外的周期来完成正在进行的后台指令。

```
ACI Benchmark Test:
F28004x CPUCLK = 100000000 Hz, from RAM, using TMU

Execution in cycle counts (avg, max, min) over 1024 iterations

------------------------------------------------------------------------
                                    AVG     MAX     MIN
 INT Response (trigger to ISR entry) :  49      53      47
 Read 2 ADC, convert float          :  3       3       3
 Clarke Transform                   :  11      11      11
 3 PID Controller Transforms        :  107     107     107
 Inverse Park Transform             :  23      23      23
 Flux Estimator                     :  167     168     166
 Speed Estimator                    :  66      67      66
 Park Transform                     :  18      18      18
 SVGen Transform                    :  75      82      66
 Write 3 PWM                        :  10      10      10
------------------------------------------------------------------------
 Total                              :  529     542     517
------------------------------------------------------------------------

Motor Speed at end of test (expected 0.5) : 0.500923

Test program end.
```

图 3-2. ACI 电机基准测试中断响应基准测试数据

表 3-1 列出了每个单独元素的理论估计影响值，并且可以看出总周期数与应用程序基准测试获得的最小周期数一致。

表 3-1. F28004x ADC INT 响应持续时间的理论估计值

触发延迟 (1)	ADC 采样持续时间 (2)	硬件锁存和响应(3)	编译器上下文保存(4)	总计（周期数）
2	8	14	23	**47**

(1) 来自器件专用数据表中的 ADC 时序图。
(2) 由应用程序根据器件专用数据表中的建议进行配置。
(3) 来自器件专用 TRM 中的外设中断说明。
(4) 通过检查生成的代码和计算周期获得。

查看影响总周期数的各个组件，可以看出 PWM 和 ADC 紧密集成，并且触发延迟也带来了几个额外的周期。ADC 早期中断产生特性允许系统响应中断时并行进行 ADC 转换。这样允许转换周期与中断产生和编译器上下文保存周期重叠，从而减少响应 ADC 采样事件所需的总时间。相比之下，对于任何其他不具有 ADC 早期中断产生特性的系统，中断响应将包括转换周期，因此会增加总响应时间。此特性的另一个优点是，ADC 结果可在时间更接近采样时间时应用于系统，从而实现更精确的系统控制。

3.2 外设访问

ACI 实时基准测试有两个外设访问响应：ADC 读取并转换为浮点以及 PWM 写入。有关每种操作的响应时间，请参阅图 3-3。

```
ACI Benchmark Test:
F28004x CPUCLK = 100000000 Hz, from RAM, using TMU

Execution in cycle counts (avg, max, min) over 1024 iterations

-------------------------------------------------------------------------
                                        AVG     MAX     MIN
 INT Response (trigger to ISR entry) :  49      53      47
 Read 2 ADC, convert float           :  3       3       3
 Clarke Transform                    :  11      11      11
 3 PID Controller Transforms         :  107     107     107
 Inverse Park Transform              :  23      23      23
 Flux Estimator                      :  167     168     166
 Speed Estimator                     :  66      67      66
 Park Transform                      :  18      18      18
 SVGen Transform                     :  75      82      66
 Write 3 PWM                         :  10      10      10
-------------------------------------------------------------------------
 Total                               :  529     542     517
-------------------------------------------------------------------------

Motor Speed at end of test (expected 0.5) : 0.500923

Test program end.
```

图 3-3. ACI 电机基准测试的外设访问基准测试数据

在此特定实现方案中，仅读取两个 ADC 输入，且写入三个 PWM 输出。从表 3-2 中可以看出，在涉及电机控制或数字电源的其他实际应用中，这些操作的数量可能更大，并且可能对控制应用程序的执行持续时间产生更大的影响。

在 C2000 架构中，CPU 与外设紧密集成，并且对指令集架构进行优化以用于浮点运算。从表 3-3 中的基准测试数据可以看出，读取一个 ADC 输入并将其转换为浮点仅需约 2 个周期，而写入一个 PWM 输出却需要约 3 个周期。因此，C2000 架构可以有效地扩展到需要大量访问外设的应用中。

表 3-2 根据基准数据获取一些实际控制应用的估计周期。

表 3-2. 实际控制应用的外设访问估计周期

应用	ADC 读取次数	PWM 写入次数	估计总周期数
三相交流 电机控制应用	7	3	23 (1)
图腾柱 PFC 数字电源应用	10	9	47(2)

(1) 7 次 ADC 读取（每次 2 个周期）+ 3 次 PWM 写入（每次 3 个周期）= (7 x 2) + (3 x 3) = 23
(2) 10 次 ADC 读取（每次 2 个周期）+ 9 次 PWM 写入（每次 3 个周期）= (10 X 2) + (9 X 3) = 47

3.3 TMU（数学增强）影响

许多 C2000 目标都具有称为三角函数加速器 (TMU) 的数学扩展，这是一种浮点单元 (FPU) 扩展，通过高效执行控制系统应用中常用的三角和算术运算来增强 C28x+FPU 的指令集。

ACI 电机控制应用具有以下控制算法：涉及三角函数的 Park、逆向 Park 和通量估算器。为了演示 TMU 可能产生的影响，F28004x 具有两个构建配置 ***SignalChain_RAM_TMU*** 和 ***SignalChain_RAM_FastRTS***。

在下图中，针对两种配置中的每种配置，图 3-4 显示了从基准周期数据得出的执行时间，图 3-5 显示了编译器工具生成的链接器映射文件中指示的控制代码（不包括驱动程序库和运行时）的应用程序大小（代码和数据）。

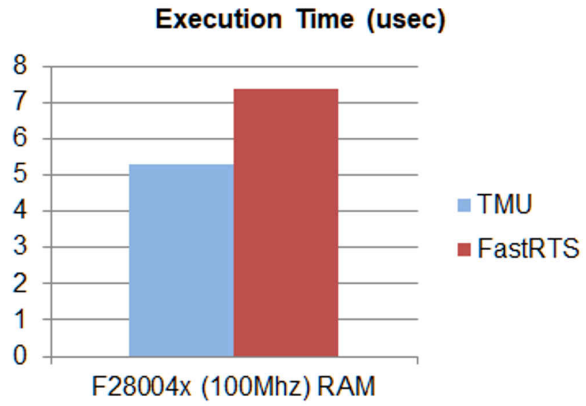


图 3-4. 使用 TMU 和 FastRTS 在 RAM 上的控制环执行时间

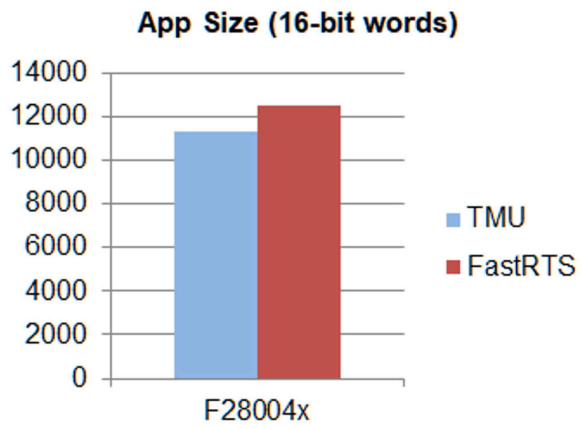


图 3-5. 针对 TMU 和 FastRTS 编译的应用程序大小

图中的数据表明，TMU 具有两个方面的影响：

- **缩短执行时间：**应用程序的执行速度提高了 28%。
- **减小应用程序大小：**典型的三角实现包括查找表和浮点计算（用于计算常见的三角运算）。TMU 为这些三角运算提供了指令，并消除了查找表和复杂的浮点代码。这意味着应用程序大小将减小，从而允许存储器更小但带有 TMU 的器件能够支持该应用程序。在此特定示例中，应用程序大小减小了 14%。

3.4 闪存性能

许多应用程序都配置为在闪存外执行。因此，闪存性能是衡量实时系统功能的关键标准。典型的基准测试通过执行合成代码来计算闪存效率，但在执行包含非线性代码流、数据表和外设访问的应用程序时，合成代码可能无法显示出性能。

F28004x 和 F2837x 具有两种构建配置：“***SignalChain_RAM_TMU***”和“***SignalChain_FLASH_TMU***”。这些配置的执行时间如图 3-6 和图 3-7 所示。

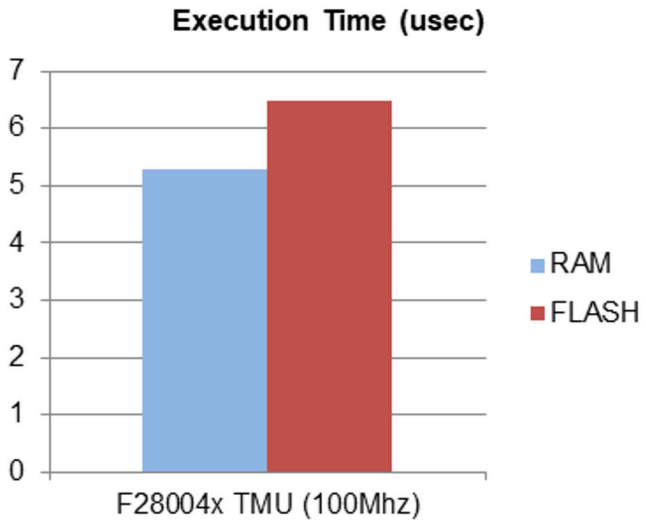


图 3-6. TMS320F28004x 从 RAM 和闪存执行 的时间

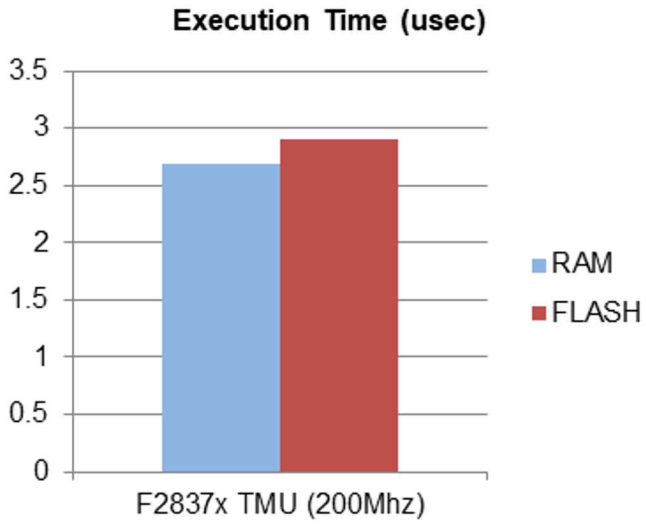


图 3-7. TMS320F2837x 从 RAM 和闪存执行的时间

表 3-3 列出了根据总平均执行周期基准测试数据得出的闪存性能。

表 3-3. 闪存执行性能与 RAM 执行性能的比较

	RAM 执行（周期数）	闪存执行（周期数）	闪存相对性能 (RAM_cycles * 100 / Flash_cycles)	闪存技术
TMS320F28004x	529	648	82%	• 100MHz CPU 时钟 • 20MHz 闪存速度 • 4 个等待状态
TMS320F2837x	537	582	92%	• 200MHz CPU 时钟 • 50MHz 闪存速度 • 3 个等待状态

从表中的数据可以看出，与 RAM 性能相比，C2000 器件在 ACI 电机控制等实际应用中的闪存性能非常好（大约为 80-90%）。

3.5 控制律加速器 (CLA)

CLA 是一个任务驱动型完全可编程的独立 32 位浮点硬件加速器，专用于数学密集型计算。CLA 还可访问 ADC 和 PWM 等外设。CLA 支持八个硬件任务。硬件任务可以由 C28x CPU 或控制外设发起。有关 CLA 的更多详细信息，请参阅“参考文献”部分中链接的文档。这些功能使 CLA 成为一个强大工具，通过两种方式即可实现实时系统性能目标：

1. CLA 可由外设触发，并可以执行整个控制循环，从而释放 C28x CPU 以用于其他活动（如运行通信协议栈），从而有效地提高可用计算能力。
2. C28x CPU 可以将部分计算转移到 CLA，这可以减少执行时间，从而提高性能。

ACI 实时基准测试已导入，以展示 CLA 的这两种用途。

3.5.1 CLA 上执行的完整信号链

CLA 是一种任务驱动型架构，与控制外设紧密集成。器件专用 TRM 和数据表包含有关 CLA 可访问的外设的信息。

ADC 可以触发 CLA 上的任务，就像触发 C28x CPU 上的中断一样。CLA 还可以写入 PWM。这些功能允许在 CLA 上执行整个 ACI 信号链，即 ADC 触发 CLA 任务，CLA 任务执行整个控制循环。当 CLA 执行信号链时，C28x CPU 处于空闲状态。这表明，在一个更全面的示例中，可以对应用程序进行分区以从 CLA 中运行控制循环，并让 C28x CPU 运行其他操作（如通信协议栈）。

有关实现方案，请参阅“*SignalChain_RAM_CLAmath_CLA*”构建配置。

```
ACI Benchmark Test:
F28004x CPUCLK = 100000000 Hz, from RAM, CLA using CLAmath


Execution in cycle counts (avg, max, min) over 800 iterations

-------------------------------------------------------------------------
                                      AVG     MAX     MIN
 INT Response (trigger to ISR entry) :  17      17      17
 Read 2 ADC, convert float           :  4       4       4
 Clarke Transform                    :  15      16      15
 3 PID Controller Transforms         :  112     113     112
 Inverse Park Transform              :  85      86      84
 Flux Estimator                      :  342     365     321
 Speed Estimator                     :  87      91      87
 Park Transform                      :  83      84      82
 SVGen Transform                     :  87      99      72
 Write 3 PWM                         :  6       6       6
-------------------------------------------------------------------------
 Total                               :  838     881     800
-------------------------------------------------------------------------

Motor Speed at end of test (expected 0.5) : 0.498148
```

图 3-8. 用于 CLA 执行整个信号链的 ACI 电机基准测试输出

对基准测试的见解详见如下。

3.5.1.1 CLA ADC 中断响应延迟

CLA 是一种任务驱动型架构，因此没有传统中断架构上下文保存和恢复开销。因此，CLA 具有较低的中断延迟，允许其能“及时”读取 ADC 样本。更多有关此主题的信息，请参阅器件专用 TRM 的 *CLA* 部分。

在此应用中，用于触发 CLA 任务的 ADC 设置与 ADC 触发 C28x 中断的配置相同。表 3-4 显示了 CLA 与 C28x 相比较的情况，并展示了在没有任何上下文保存开销情况下的低延迟。

表 3-4. 用于 F28004x 上 CLA 和 C28x 的 ADC 中断响应延迟

计算内核	ADC 中断响应（周期数）
CLA	17
C28x	47

3.5.1.2 CLA 外设访问

CLA 与 ADC 和 PWM 外设紧密集成，就像 C28x CPU 一样。CLA 的外设访问延迟与 C28x CPU 类似，因此 CLA 也同样高效。这种效率对于控制循环的执行至关重要。

表 3-5. F28004x 上 CLA 和 C28x 的外设访问延迟

计算内核	2 次 ADC 读取（周期数）	3 次 PWM 写入（周期数）
CLA	4	6
C28x	4	9

3.5.1.3 CLA 三角函数计算

CLA 在浮点计算方面效率很高。然而，与具有 TMU 硬件加速器的 C28x CPU 不同，CLA 没有三角硬件加速器，需要通过软件实现。C2000Ware 软件包提供了 CLAmath 库，该库可以高效地实现三角运算。在 ACI 应用程序中，CLAmath 库 API 用于包含三角运算的 Park、逆向 Park 和通量估算器计算块。

表 3-6. F28004x 的 CLAmath 性能

计算内核	Park/逆向 Park（周期数）	通量估算器（周期数）
CLA	83/85	342
C28x	18/24	167

利用 TMU 运算可以轻松地将为 C28x CPU 创建的代码导入到 CLA。C28x TMU 编译器内在函数在对 CLA 进行编译时映射到 CLAmath 库函数。因此，在带有 TMU 的 C28x CPU 上运行的 ACI 基准测试控制代码无需修改即可从 CLA 运行，只需链接 CLAmath 库并包含 CLAmath.h 头文件。

3.5.2 将计算转移到 CLA

C28x CPU 可以使用 IACK 指令通过软件触发 CLA 任务。C28x 可以使用这一功能，在 C28x CPU 上运行的代码中的特定点将计算转移到 CLA。ACI 基准测试示例是相当线性的，因为一个控制算法块需要来自前一个控制算法块的输入。然而，两种情况可以引入并行性：

1. PID 控制块：

   PID 控制算法有三个实例，其中一个（PID Id 实例）不依赖于其他实例，可以并行处理。在实现中，C28x 将 PID Id 控制执行转移到 CLA 任务 2，同时 C28x 并行执行速度和 Iq 的 PID。
2. SVGen 控制块：

   SVGen 控制块取决于 Park 逆变换的输出。由于这是一种无传感器 ACI 实现方式，因此还存在额外的控制块，如通量估算器和速度估算器。SVGen 不依赖于这些控制块，可以与这些估算器并行处理。在此实现方式中，C28x 将 SVGen 计算转移到 CLA 任务 1。

有关实现方式，请参阅“*SignalChain_RAM_TMU_CLA_OFFLOAD*”构建配置。该实现方式演示了转换代码如何轻松地将计算转移到 CLA 。相同的 PID 内联 C 函数头文件包含在 C28x 源文件以及 CLA 源文件中，并在 C28x 端编译成 C28x 代码，而在 CLA 端编译成 CLA 代码。

```
ACI Benchmark Test:
F28004x CPUCLK = 100000000 Hz, from RAM, C28 using TMU CLA offloading


Execution in cycle counts (avg, max, min) over 1024 iterations

-------------------------------------------------------------------------
                                      AVG     MAX     MIN
 INT Response (trigger to ISR entry) :  48     53      47
 Read 2 ADC, convert float           :  3      3       3
 Clarke Transform                    :  11     11      11
 3 PID Controller Transforms         :  86     86      86
 Inverse Park Transform              :  25     25      25
 Flux Estimator                      :  167    168     167
 Speed Estimator                     :  66     67      66
 Park Transform                      :  18     18      18
 SVGen Transform                     :  32     32      32
 Write 3 PWM                         :  9      9       9
-------------------------------------------------------------------------
 Total                               :  465    472     464
-------------------------------------------------------------------------

Motor Speed at end of test (expected 0.5) : 0.501991

Test program end.
```

图 3-9. 具有 CLA 转移功能的 C28x 的 ACI 电机基准测试输出

将计算转移到 CLA 的机会减少了执行周期计数，从而将性能提高 12%。

4 C2000 价值定位

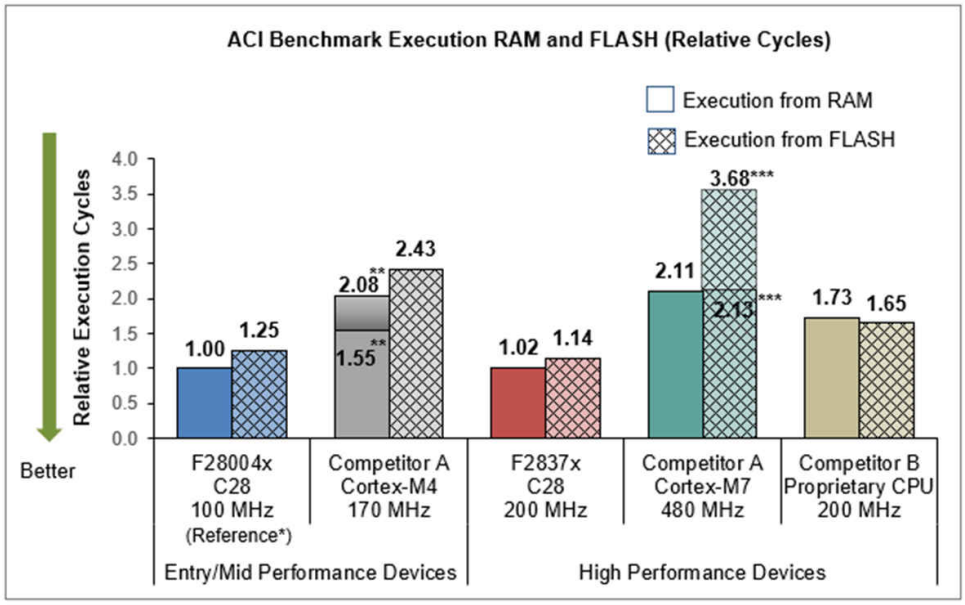
ACI 电机控制基准测试应用程序是独立的应用程序，不需要任何特殊的外部硬件即可执行。因此，该应用程序可以轻松地导入到任何器件，以便评估实时控制性能。多器件分析表明，C2000 平台提供了具有优化架构的丰富功能集，可在整个实时信号链中实现出色的性能。

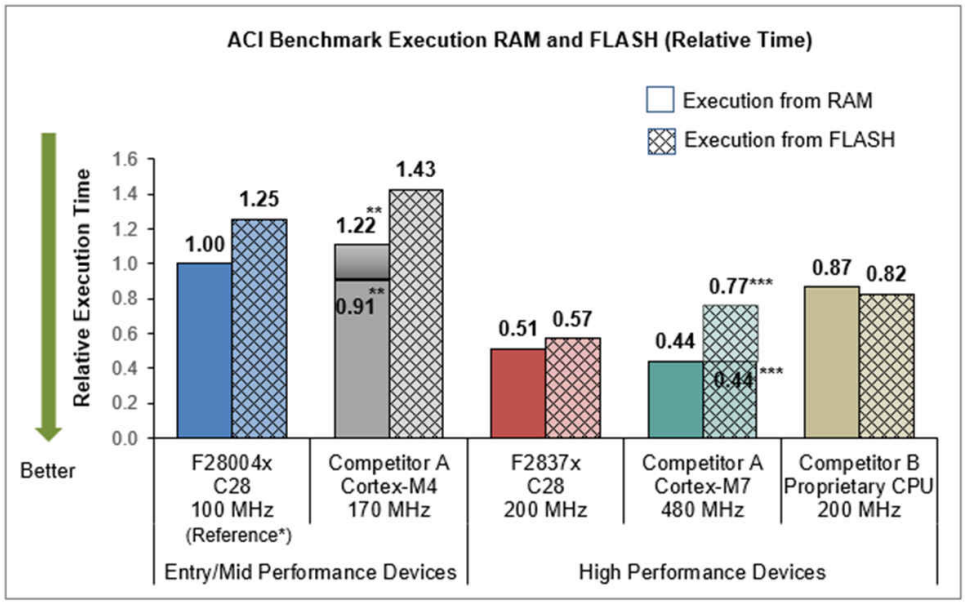
备注

1. 本章介绍的针对 TI C2000 器件和竞争器件 A 和 B 的多器件分析结果基于 TI 在 2021 年内部编写的实际测量数据，并将 C2000Ware 3.04.00.00 中的 ACI 电机基准测试示例导入到了这些器件。
2. 通过将工具链设置为编译代码以实现最大性能/速度，并以最大 CPU 频率运行器件来完成所有的测量。
3. 测量每个器件的周期/时间，并与从 RAM 执行的 F28004x 进行数据比较，以确定相对性能。每张图表都显示了相对性能，这就是为什么 F28004x 被标记为“参考”且始终为 1.0 的原因。
4. 在图中，较低的相对数字表示较少的周期或较少的时间，因此意味着性能较好。
5. 根据 TI 命名法，图中将类别分类为“入门级/中级性能”和“高性能”。TI 器件和相应的竞争器件属于同一类别。

4.1 高效执行信号链，使实时响应比计算速度更高的 MIPS 器件更好

ACI 基准测试数据表明，C2000 器件能够快速响应采样事件并有效执行响应，从而实现***快速的采样-输出响应***。因此允许将更多 MIPS 用于其他操作。 ***F28004x（入门级/中级性能器件）和 F2837x（高性能器件）分别具有 82% 和 92%的良好闪存执行效率***，在闪存外执行也是满足实时性能要求的可行选择。





* 如注释中所述，从 RAM 执行 F28004x 仅供参考。

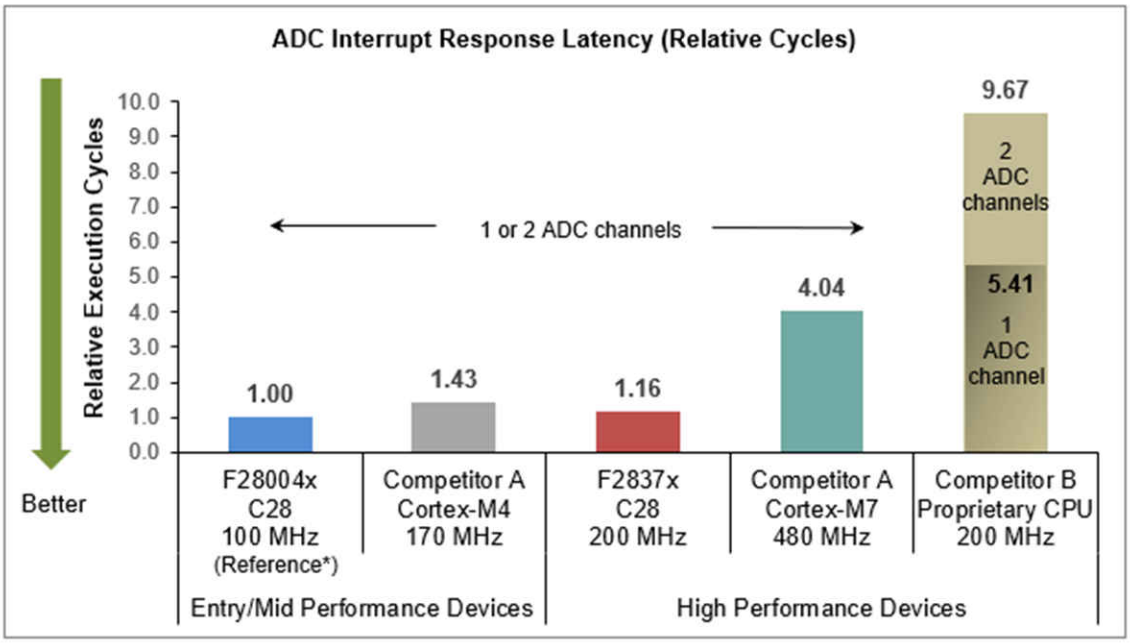
** 使用更快速的 RAM 或标准 RAM 组在最好和最坏情况下的性能数字。

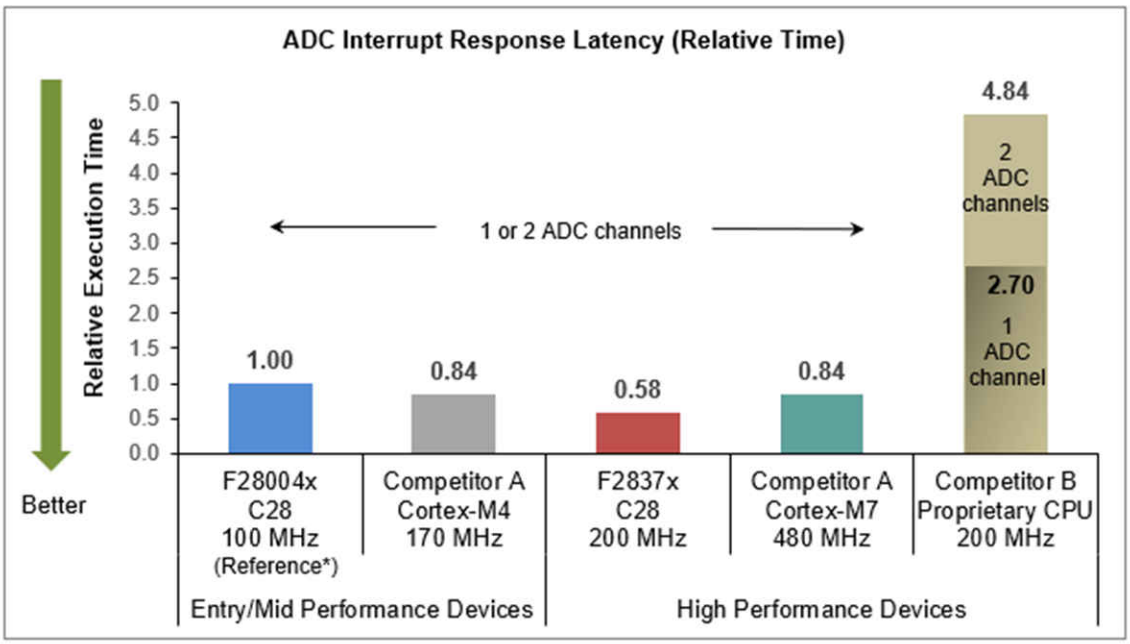
*** 使用热缓存和冷缓存在最好和最坏情况下的性能

图 4-1. ACI 电机控制基准测试执行（相对周期和相对时间）

4.2 具有低延迟的出色的实时中断响应

C2000 器件具有出色的模拟外设。***ADC 具有快速采样率*** 和***早期中断产生*** 特性，其中可以在序列中的第一个 ADC 通道值仍在转换时触发中断。再加上***高效的中断架构***，在响应对 ADC 采样事件的触发时，周期数将减少。



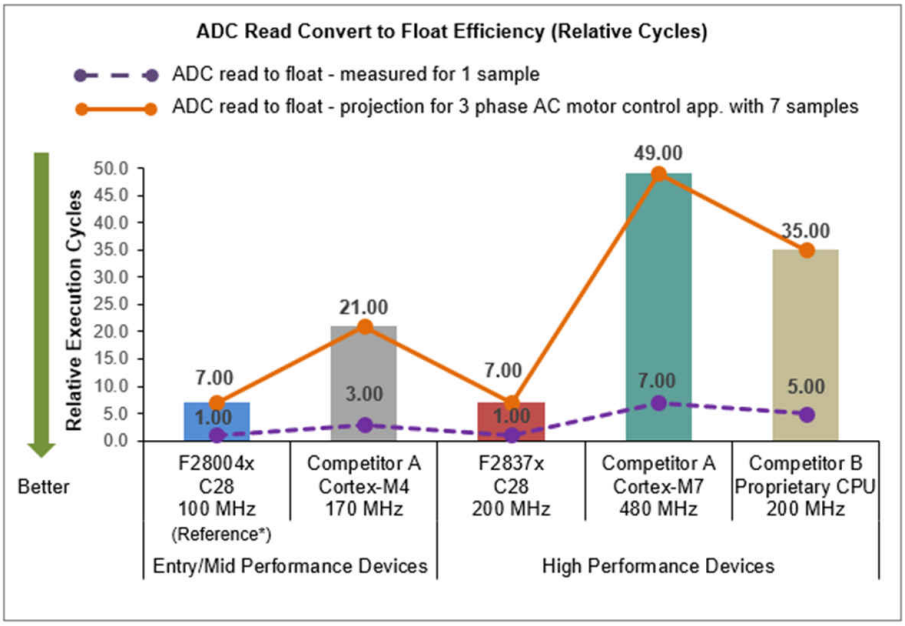


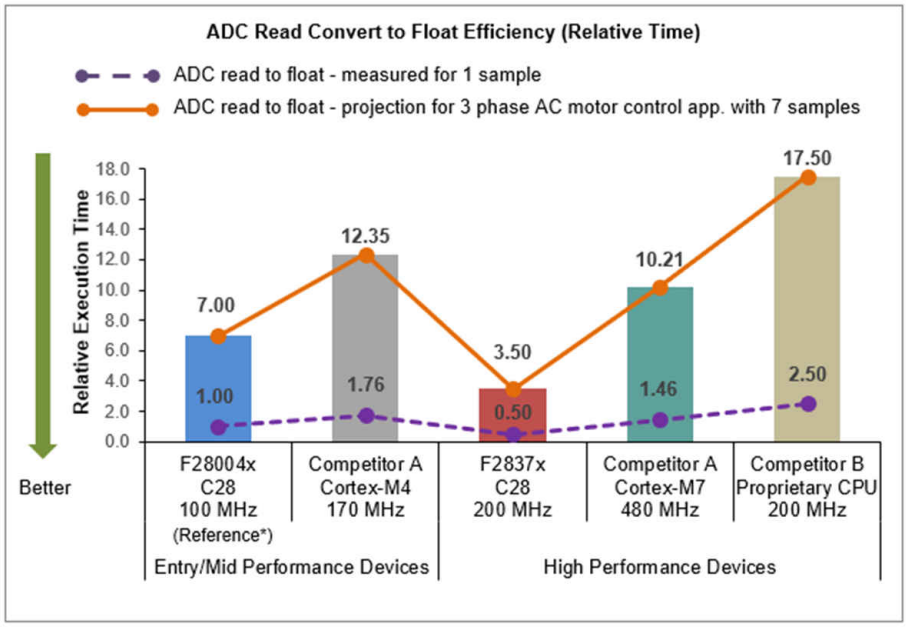
* 如注释中所述，从 RAM 执行 F28004x 仅供参考。

图 4-2. ADC 中断响应延迟（相对周期和相对时间）

4.3 外设紧密集成，可扩展具有大量外设访问的应用

C2000 指令集和外设紧密集成力使读取 ***ADC 结果并转换为浮点*** 仅需 ***2 个周期***。因此，ADC 访问开销非常小，所以可以高度扩展到具有大量 ADC 访问的应用程序。



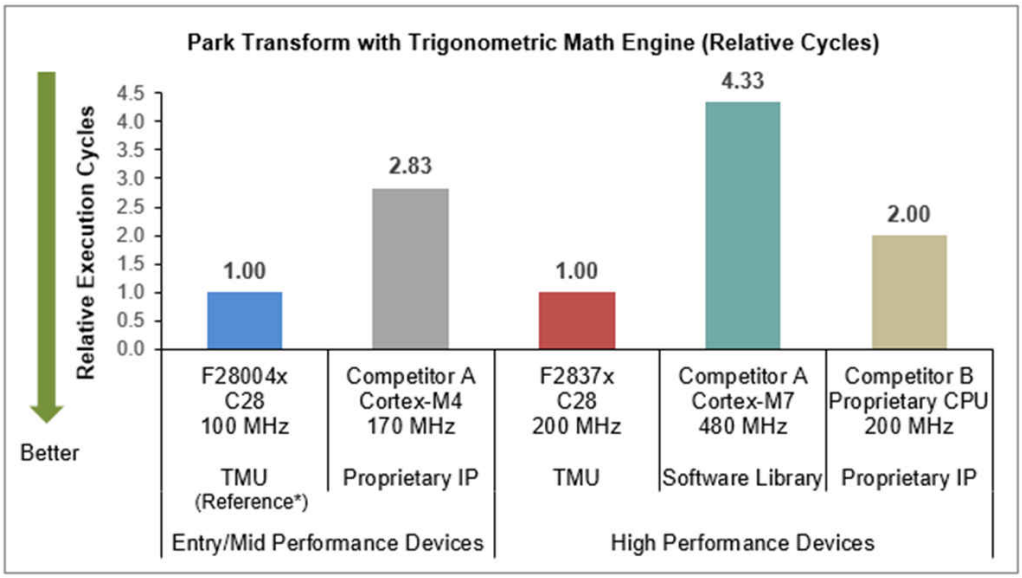


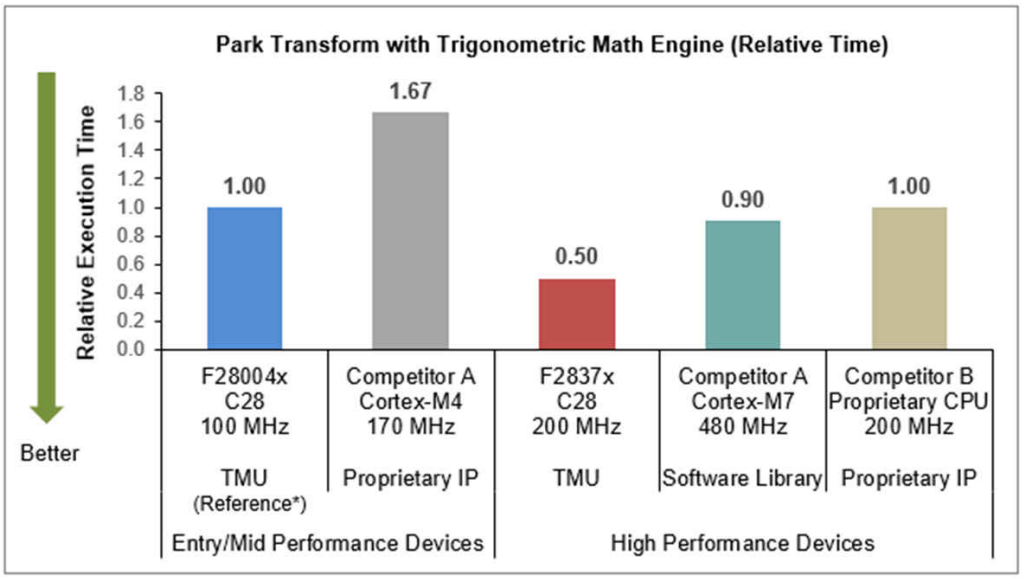
* 如注释中所述，从 RAM 执行 F28004x 仅供参考。

图 4-3. ADC 读取并转换为浮点的效率（相对周期和相对时间）

4.4 最优三角函数引擎

三角运算在许多控制算法中都很常见，Park 变换就是这样的例子。有效地执行三角运算对于最小化计算持续时间至关重要。***各种性能级别的所有第 3 代 C2000 器件***，包括入门级/中级性能的器件（如 F28004x）到高性能的器件（如 F2837x），均具有称为***三角函数加速器 (TMU)*** 的三角函数引擎。TMU 扩展了用于三角运算的指令集。借助编译器内在函数的支持、***易于使用*** 的编程模型，可以轻松地将 TMU 的性能提升应用于 C2000 器件上执行的实时控制算法。





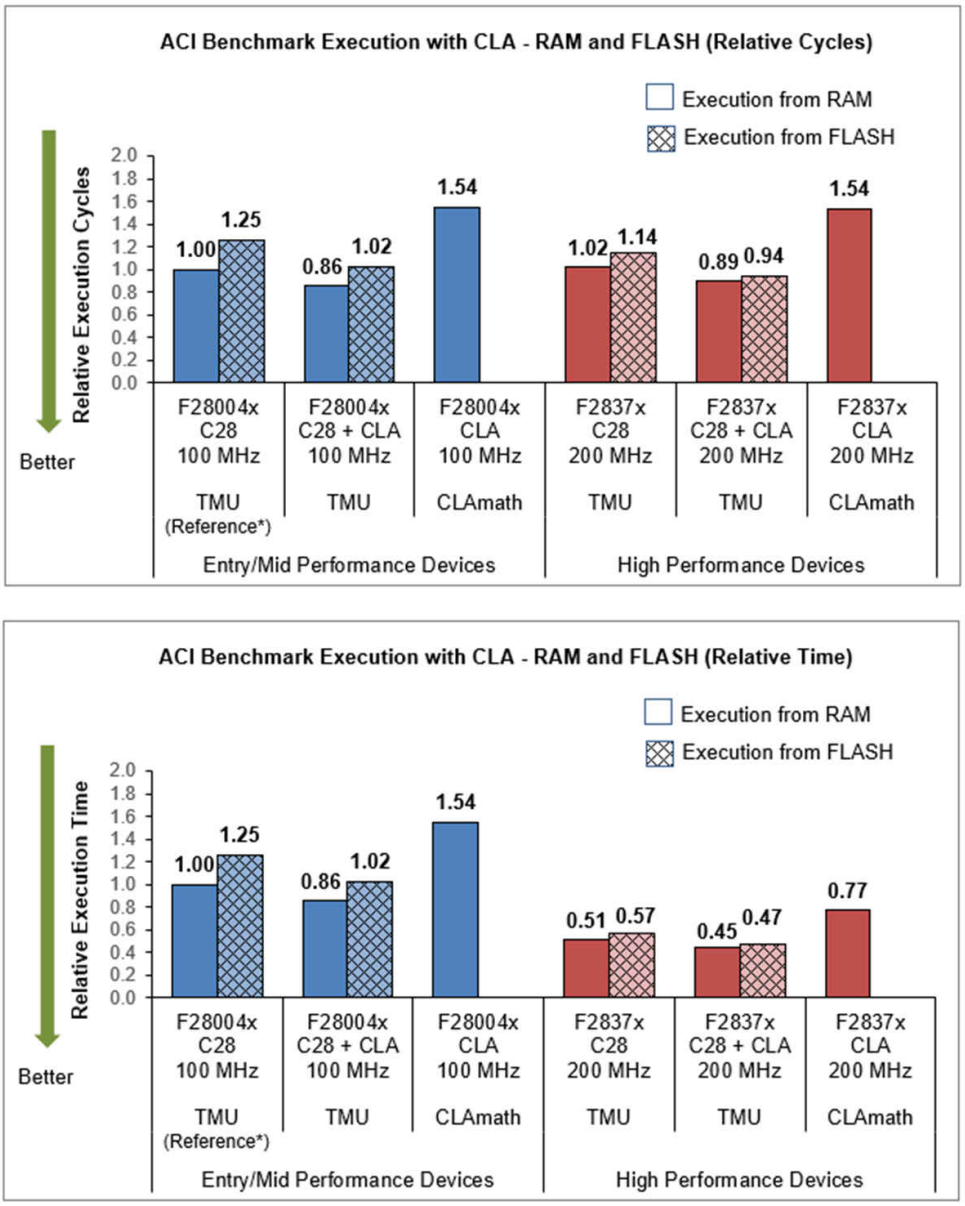
* 如注释中所述，从 RAM 执行 F28004x 仅供参考。

图 4-4. 使用三角函数引擎进行 Park 变换（相对周期和相对时间）

4.5 多功能性能提升计算引擎 (CLA)

许多 C2000 器件都有一个 CLA（控制律加速器），其运行频率与 C28x CPU 相同，可以通过两种不同方式来实现实时应用目标。

- CLA 执行完整信号链 - CLA 独立于 C28x CPU 执行，释放 C28x CPU 以用于其他活动，并有效地将可用 MIP 加倍。
- C28x CPU 将计算转移到 CLA - 并行性通过减少采样到输出的响应时间以实现更好的执行性能。



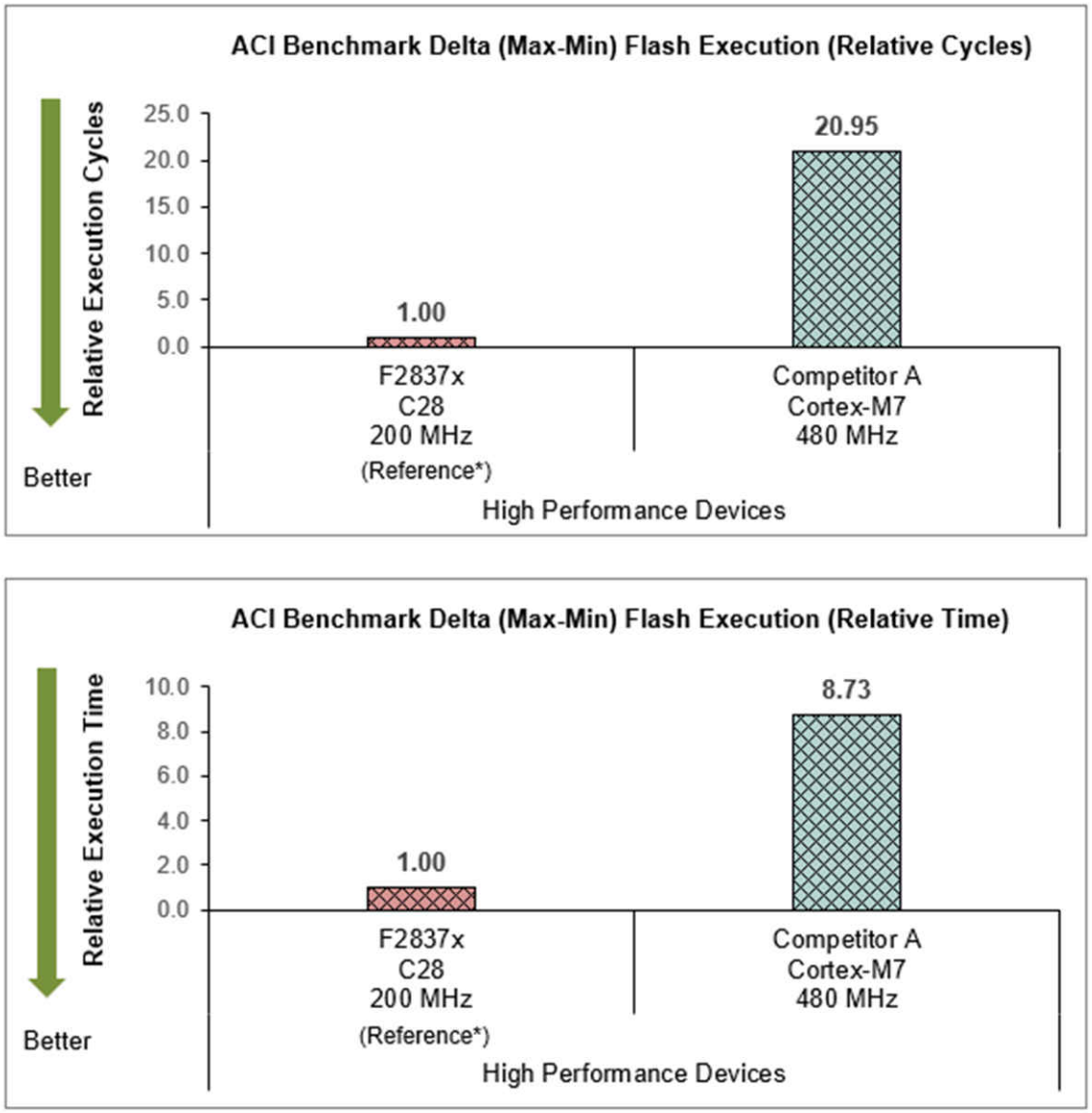
* 如注释中所述，从 RAM 执行 F28004x 仅供参考。

CLA 仅从 RAM 运行，因此没有闪存执行数据点。

图 4-5. ACI 电机控制基准测试 CLA 执行（相对周期和相对时间）

4.6 由于执行差异小而导致确定性执行

实时系统非常需要确定性执行。C2000 系列器件具有用于提高闪存执行且没有任何缓存的微小预取缓冲区。虽然缓存可以提高性能，但由于缓存缺失和一致性操作导致的缓存操作，缓存也会导致执行持续时间产生巨大差异。C2000 器件在没有缓存的情况下也能很好地执行，反过来，也允许采用较小的执行差异进行确定性执行。



* 从闪存执行 F2837x 仅供参考。

图 4-6. 从闪存执行 ACI 电机控制基准测试的差异（相对周期和相对时间）

5 总结

本应用手册注释表里仅关注处理时间的典型行业软件基准测试在执行实时控制应用程序时无法有效地衡量系统的真实性能。实时基准测试方法可以更全面、更真实地反映系统的实时功能。例如，从 RAM 执行带有 TMU 的 F28004x 器件的 ACI 电机基准测试数据表明，非控制算法分量（INT 响应 + 读取 ADC + 写入 PWM）约占总信号链执行持续时间的 12%，这一数据在典型的软件基准测试中是无法计算的。这一百分比会因不同器件的架构和模拟能力而有所不同，对于外设访问或 ADC 采样花费较长时间的器件以及其他具有大量外设访问的实时控制应用程序，此百分比可能更高。

本应用注释中的实时基准测试数据展示了 C2000 架构的不同化特性，例如外设紧密集成、ADC 早期中断特性、具有 TMU 等数学增强功能的优化指令集以及高效的闪存技术，所有这些特性有助于减少采样到输出的响应时间，从而高度优化 C2000 MCU 以用于实时控制应用程序。

6 参考文献

- 德州仪器 (TI)：*TMS320F28004x 微控制器技术参考手册*
- 德州仪器 (TI)：*TMS320F28004x 微控制器数据表*
- 德州仪器 (TI)：*TMS320F2837xD 双核微控制器技术参考手册*
- 德州仪器 (TI)：*TMS320F2837xD 双核微控制器数据表*
- 德州仪器 (TI)：软件开发套件 - 用于 C2000 MCU 的 C2000Ware
- 德州仪器 (TI)：TMU 指令集 -*TMS320C28x 扩展指令集技术参考手册*
- 德州仪器 (TI)：TMU 补充细节 -提高 *C2000™ 微控制器系列的计算性能*
- 德州仪器 (TI)：CLA 详细信息 - 提高 *C2000™ 微控制器系列的计算性能*
- 德州仪器 (TI)：使用 C2000™ 实时微控制器进行开发的基本指南
- 德州仪器 (TI)：CLA 应用报告 - 展示 TI C2000™ CLA 独特功能的软件示例

7 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (April 2021) to Revision A (December 2021) Page

- 更新了本文档的“摘要” 1
- 更新了整个文档中的表格、图和交叉引用的编号格式 3
- 对节 2.1 进行了更新 3
- 对节 2.2 进行了更新 4
- 对节 2.3 进行了更新 5
- 对节 2.5.1 进行了更新。 8
- 对节 2.6 进行了更新。 8
- 对节 3 进行了更新。 9
- 对节 3.1 进行了更新。 10
- 对节 3.2 进行了更新。 11
- 新增了新的 节 3.5。 14
- 对节 4.1 进行了更新。 18
- 对节 4.2 进行了更新。 19
- 对节 4.3 进行了更新 19
- 对节 4.4 进行了更新。 21
- 新增了新的 节 4.5。 22
- 新增了新的 节 4.6。 23
- 对节 5 进行了更新 24
- 对节 6 进行了更新 24

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 TI 的销售条款或 ti.com 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265