Application Note

Sitara™ AM64x/AM243x 基准测试

摘要

本应用手册包含有关 AM64x 和 AM243x 系列器件的基准测试。

内容

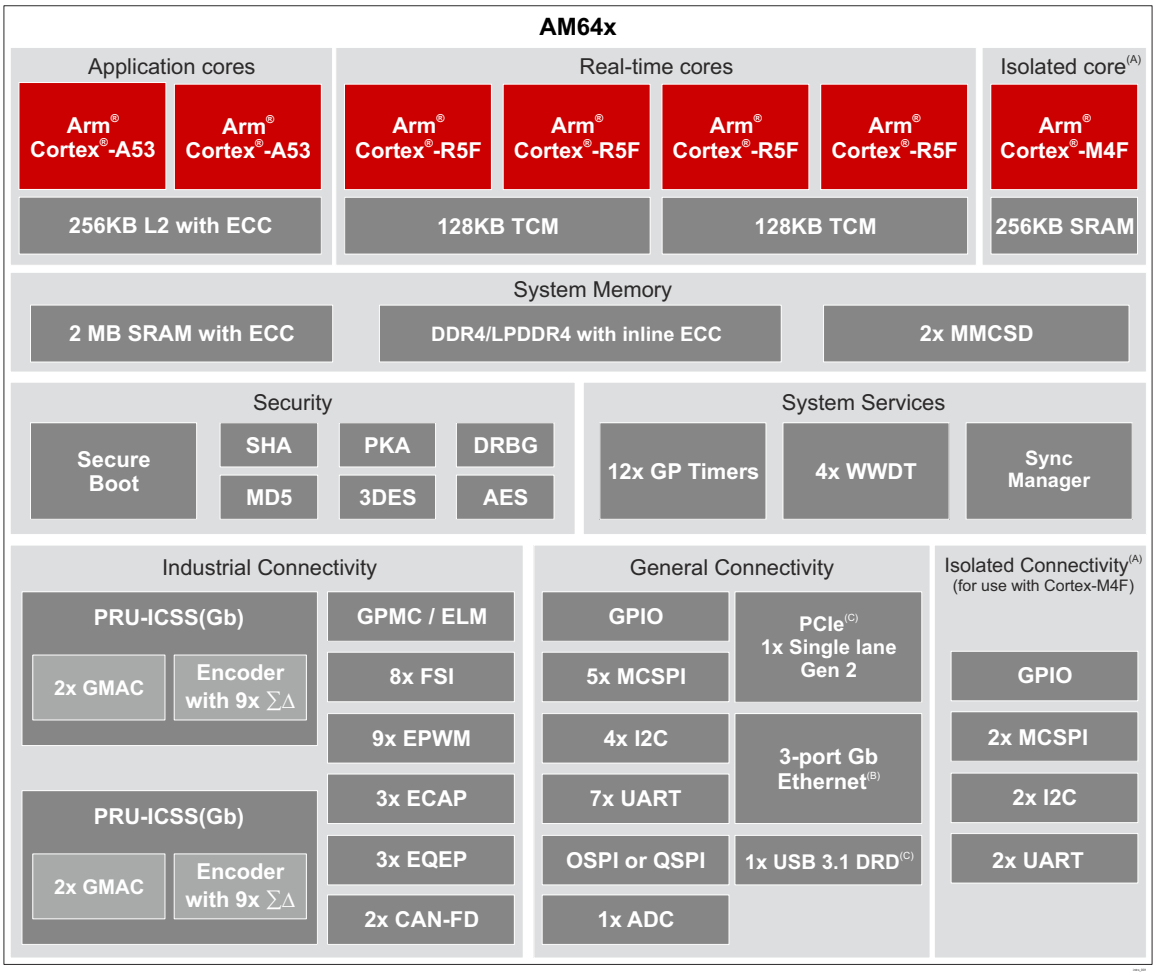
插图清单

表格清单

商标

1 简介

基准测试是在 Cortex®-A53 和 Cortex-R5F 内核上进行的。有关最新结果，请参阅性能指南和 AM64x Processor SDK 中的基准测试演示应用程序。基准测试通过以下方法生成：TMDS64GPEVM、SK-AM64X 和 LP-AM243。评估板关键板参数包括：Cortex-A53 内核时钟速度为 1GHz，Cortex-R5F 内核时钟速度为 800MHz，16 位宽 DDR4 或 LPDDR4 速度为 1600MT/s。AM64x 添加了一个包括 256kB L2 高速缓存的双核 Cortex-A53，但除此以外，器件相同。有关参考，请参阅图 1-1 和图 1-2。



A. 外设与 M4F 内的核隔离是一项可选特性。在非隔离配置中，MCU 域资源可跨 SoC 共享。

图 1-1. 功能方框图：AM64x

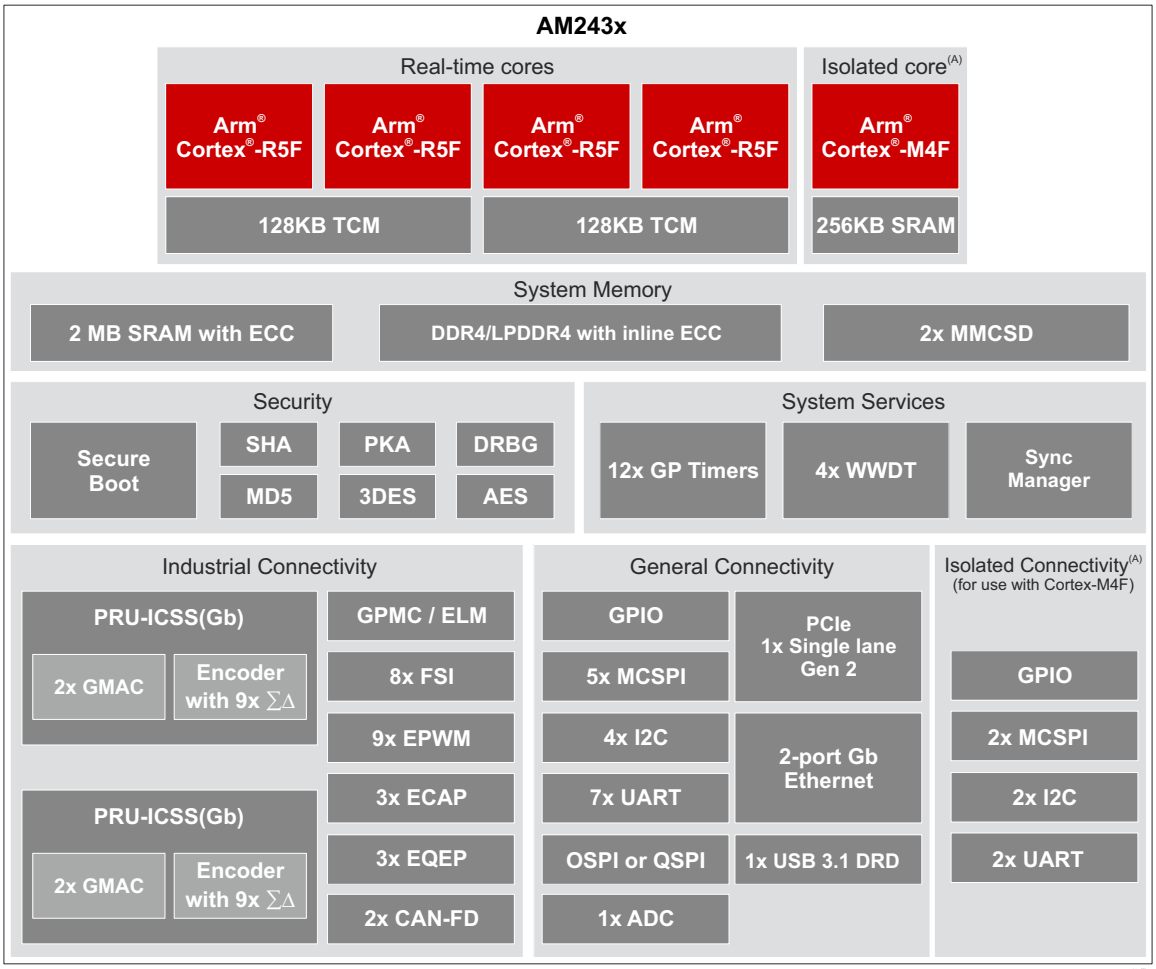


图 1-2. 功能方框图：AM243x

2 处理器内核基准测试

本节包含以下由 Arm® Cortex 处理器内核生成的基准测试：1) Dryhstone（一项综合基准测试）和 2) math.h 中所定义函数的三角基准。

2.1 Dhrystone

Dhrystone 基准测试仅适用于内核，并从所有现代处理器的 L1 高速缓存（热缓存）运行。该测试程序随时钟速度的增加而线性增加。计算分数时，通过参考 1MIP 机器的分数 (1757)，对运行基准循环的时间进行标准化。虽然该基准测试于 1984 年由 Reinhold P. Weicker 引入，但 Dhrystone 至今仍用于嵌入式处理。由于分数随时钟速度的增加而线性增加，通常进一步标准化为 DMIPS/MHz/内核。对于标准 Arm 内核，在相同的编译器和标志中，DMIPS/MHz 将是相同的。Dhrystone 是一个单核基准测试，有时会使用多个简单内核并行运行此基准测试。对于具有两个 1GHz A53 内核 (6000DMIPS) 和四个 800MHz R5F 内核 (6400DMIPS) 的 AM6442，总分为 12400DMIPS。

	Cortex-A53 (1GHz)	Cortex-R5F (800MHz)
Dhrystones	5263158	2,962,962
标准化 Dhrystones（除以 1 MIP 机器的 1757 参考值）	2996	1686
每个核心的 DMIPS/MHz	3	2.1
编译器和标志	GCC 9.2 -march=ARMv8 -O3	Arm Compiler 6.14 -mcpu=cortex-r5 -O3
操作系统	Linux 5.10 (2021 LTS)	裸机

2.2 三角函数

三角函数是用于电机控制的计算函数。最常用的三角函数为 sine、cosine、arctan 和 arctan2。标准 C 库 *math.h* 可提供高度精确的三角函数实现，但最坏情况下的执行时间可能是不可接受的。对于 Arm Cortex-R5F，Arm CMSIS DSP 库也可为部分三角函数提供经过优化的实现。Sitara™ 软件开发套件 MCU-PLUS-SDK_AM243X 和 PROCESSOR-SDK-AM64X 包含 CMSIS 库（位于文件夹 <mcu-plus-install-directoty>/source/cmsis），并进一步优化了 ti_r5fmath_trig 库（位于文件夹 <mcu-plus-install-directoty>/examples/motor_control/benchmark_demo/common）中的部分函数。表 2-1 和 表 2-2 显示了使用 C 运行时库 (math.h)、CMSIS 和 ti_r5fmath_trig 库提供的实现的最常用三角函数的性能。这些实现会影响准确度（如最大误差一栏所示），以及计算时间，有时也会计算额外存储器消耗（有存储器中常量的查找表），如表大小一栏所示。此表需要在 TCM 或 L1D 高速缓存（热缓存）中，以实现记录的性能。

表 2-1. Arm Cortex-R5F（32 位浮点）Sine 和 Cosine

函数	最大周期	平均周期	最大误差 (abs rad)	表大小	库
ti_r5fmath_sin()	34	34	7.20E-07	polynomial	TI_R5FMATHLIB
ti_r5fmath_cos()	38	38	2.90E-07	polynomial	TI_R5FMATHLIB
ti_r5fmath _sincos()	51	51	7.20E-07	polynomial	TI_R5FMATHLIB
ti_r5fmath_fast_sincosB()	57	57	1.90E-07	polynomial	TI_R5FMATHLIB
arm_cos_f32()	66	66	1.80E-05	表 2KB	CMSIS
sinf()	71	71	8.40E-08	polynomial	math.h
cosf()	81	81	9.70E-08	polynomial	math.h
arm_sin_cos_f32()	114	114	6.10E-07	表 2KB	CMSIS
sin() (double)	340	296	3.00E-08	polynomial	math.h

表 2-2. Arm Cortex-R5F（32 位浮点）Arctan 和 Arctan2

函数	最大周期	平均周期	最大误差 (abs rad)	表大小	库
ti_r5fmath_atanFast()	45	39	3.76E-03	函数	TI_R5FMATHLIB
ti_r5fmath_atan2Fast()	54	54	3.76E-03	函数	TI_R5FMATHLIB
ti_r5fmath_atan()	80	68	6.00E-07	poly	TI_R5FMATHLIB
atanf()	111	90	6.80E-08	poly	math.h
ti_r5fmath_atan2()	97	90	6.00E-07	poly	TI_R5FMATHLIB
atan2f()	204	171	2.00E-07	poly	math.h

3 计算和存储系统基准测试

本节包含涉及 Arm Cortex 处理器内核和 SoC 存储系统的基准测试。包括综合基准测试，例如 LMbench 和 CoreMark-Pro。数学函数基准测试包括线性代数和快速傅里叶变换 (FFT)等函数。

3.1 存储器带宽和延迟

STREAM 和 LMBench 的子集是用于测量实现的存储器带宽和软件延迟的基准测试程序。

3.1.1 LMBench

LMBench 是一套适用于处理器内核和操作系统基元的微基准测试工具。存储器带宽和延迟相关测试非常适用于现代嵌入式处理器。每次运行的结果略有不同 (<10%)。

LMBench 基准测试 *bw_mem* 测量实现的存储器复制性能。其使用参数 *cp* 复制数组，*bcopy* 参数使用运行时 glibc 版本的 *memcpy()* 标准函数。我利用 SIMD 等实现更高性能，在实施高度优化的基础上进行 glibc 实践。等于或小于给定级别高速缓存大小的 size 参数可测量进行典型的 for 循环或 *memcpy()* type 操作的软件可实现的存储器带宽。通常用于计算外部存储器带宽。带宽根据字节读写（每读写 1 字节计为 1）计算，结果应约为 STREAM 复制结果的一半。下表显示了相对于理论线速测得的带宽和效率。使用的线速计算方式为：DDR MT/s x 宽度 ÷ 2（构成复制的读取和写入均会消耗总线）。此基准测试还允许利用 -P 参数创建并行线程。为了获得较大多核存储器带宽，需要创建与操作系统可用内核相同数量的线程，即 2 个 (AM64x Linux (-P 2))。

	Arm Cortex-A53，DDR4-1600MT/s-16 位	DDR4 效率	Arm Cortex-A53，LPDDR4-1600MT/s-16 位	LPDDR4 效率
bw_mem -P 2 8M bcopy（双核，glibc memcpy）	1226MB/s	77%	1100MB/s	69%
bw_mem 8M bcopy（单核，glibc memcpy）	1016MB/s	64%	883MB/s	55%
bw_mem -P 2 8M cp（双核，内联复制循环）	628MB/s	39%	756MB/s	47%
bw_mem 8M cp（单核，内联复制循环）	528MB/s	33%	599MB/s	37%

LMBench 基准测试 *lat_mem_rd* 用于测量外部存储器 (AM64x DDR4/LPDDR4) 观察到的存储器存取延迟和高速缓存命中率。有两个参数，分别是事务大小（64，如以下截图所示）和读取跨度 (512)。选择这两个数值是为了测量高速缓存和外部存储器的延迟，而不是处理器数据预取器或其他推测执行的延迟。有些存取模式可实现预取，但此基准测试特别适用于无法实现预取的存取模式下的相关测量。左列是数据存取模式的大小（单位为兆字节），右侧是往返读取延迟（单位为纳秒）。Arm Cortex-A53 读取延迟概述如下：

- L1D 为 3ns
- L2 延时为 14ns
- 对于 DDR4-1600 存取，延时为 196ns
- 对于 LPDDR4-1600，延时为 217ns

以下为 DDR4 运行示例。对于 LPDDR4，在 L1D 和 L2 大小下，结果相同，但在最大大小下，结果会略高 (217ns)。

```
root@am6x-evm:~# lat_mem_rd 64 512
"stride=512
0.00049 3.006
0.00098 3.006
0.00195 3.006
0.00293 3.006
0.00391 3.006
0.00586 3.006
0.00781 3.006
0.01172 3.006
0.01562 3.006
0.02344 3.009
0.03125 3.120
0.04688 9.212
0.06250 10.677
0.09375 12.269
0.12500 12.984
0.18750 13.651
0.25000 14.066
0.37500 115.226
0.50000 168.747
0.75000 189.919
1.00000 192.138
1.50000 193.431
2.00000 194.175
3.00000 194.870
4.00000 195.202
6.00000 195.463
8.00000 195.622
12.00000 195.700
16.00000 195.761
24.00000 195.876
32.00000 195.938
48.00000 196.001
64.00000 196.006
```

3.1.2 STREAM

STREAM 是测量数据存储系统性能的微基准测试，无需任何数据重复。它旨在不命中高速缓存，执行数据预取和推测存取。STREAM 使用双精度浮点（64 位），但对于大多数现代处理器而言，存储器存取成为瓶颈。四个单项分数包括 copy（复制）、scale（乘常数）、add（数字相加）及 triad（乘法累加）。对于带宽，每读取一个字节计数为 1，每写入一个字节计数为 1，得到的分数是 LMBench 带宽的两倍。下表显示了相对于理论线速测得的带宽和效率。使用的线速是 DDR MT/s 与宽度的乘积。为了获得总体最大吞吐量，使用命令 *stream -M 16M -P 2 -N 10*，这意味着两个并行线程和 10 次迭代。

	DDR4-1600MT/s-16 位带宽	DDR4-1600MT/s-16 位效率	LPDDR4-1600MT/s-16 位带宽	LPDDR4-1600MT/s-16 位效率
复制	2482MB/s	78%	2221MB/s	69%
乘常数	2516MB/s	79%	2268MB/s	71%
add	2350MB/s	73%	2130MB/s	67%
triad	2355MB/s	74%	2139MB/s	67%

3.1.3 Cortex-R5 存储器存取延迟

表 3-1 记录了在 R5 运行速度为 800MHz 时测得的 Cortex-R5F 内核上的裸机软件在各个目标地址的读取延迟。

表 3-1. 存储器读取延迟

目标	延迟（单位为纳秒）
本地 TCM	1.25
另外一个 Cortex-R5F 的 TCM	77.5
共享片上存储器 (MSRAM)	63.75
ICSS DMEM	127.5
外部 DDR4	280
外部 OSPI	496.25
GPMC（并行接口边界）	271.25

3.2 CoreMark®-Pro

CoreMark-Pro 测试了整个处理器，增加了对多核技术，整数和浮点工作负载组合以及用于利用更大存储子系统的数据集的全面支持。CoreMark-Pro 的组件利用各种级别的高速缓存，数据存储容量高达 3MB。许多但并非所有测试也会使用 pthreads，以允许使用多个内核。分数随内核数量的增加而增加，但总是低于线性增加（双核分数小于单核分数的 2 倍）。

请勿将 CoreMark-Pro 与更小巧的 CoreMark 混淆，后者和 Dhrystone 一样，都是包含在现代处理器 L1 缓存中的微基准测试。

	Arm Cortex-A53
单核	604
双核	1063（单核的 1.66 倍）

3.3 快速傅里叶变换

快速傅里叶变换 (FFT) 是许多应用中的一个多重累加的重构模块。下表显示了 1024 点单精度浮点复杂 FFT 的执行时间。Arm Cortex-A53 基准测试使用了 Ne10 库，该库利用了 Cortex A53 的高级 SIMD 或 NEON 加速。

	1024pt 浮点 CFFT 执行时间（单线程/内核）
Arm Cortex-A53 (1GHz)	27 微秒
Arm Cortex-R5 (800MHz)	134 微秒

3.4 加密基准测试

AM64x Linux SDK 包含可供应用程序使用的 openssl 加密库，例如用于一些 HTTPS、ssh 和 netconf 实现，以获取加密功能的优化实现。为了获得最高性能，应使用 EVP 库提供更高级别的接口。表 3-2 显示了在 AM64x 上运行的一组选定的软件观察到的性能的部分基准测试。运行的命令是 *openssl speed -elapsed -evp <cryptographic mode> -multi 2*。这利用了两个 A53 内核，每个内核使用两个线程。

表 3-2. 对称加密和安全哈希 (单位为 Mbit/s)

	帧大小 (字节)					
	16	64	256	1024	8192	16384
aes-128-gcm	855	2438	4671	6004	6656	6637
aes-256-gcm	811	2256	4057	5241	5624	5658
aes-128-ctr	87	190	725	2446	7513	8836
sha256	559	1672	3865	5812	6860	6866
sha512	153	614	976	1390	1584	1617
chacha20-poly1305	494	1067	2091	2380	2541	2540

公钥加密的进一步基准测试如表 3-3 中所示。使用命令 *openssl speed -elapsed <algorithm> -multi 2* 可运行测试。

表 3-3. 公钥加密基准测试

RSA	大小	512	1024	2048	3072	4096
	签名/秒	5254	1174	181	59	21
	验证/秒	67996	23579	6777	3138	1523
ECDSA	曲线	nistp224	nistp256	nistp521	nistk233	nistb233
	签名/秒	310	1074	71	241	237
	验证/秒	501	2717	103	130	128

4 应用基准测试

本节包含有关以下应用级基准测试的信息和示例：1) 机器学习推理，2) 用于电机驱动控制的场定向控制 (FOC) 环路，3) 使用 BCDMA 的快速外设组件互连 (PCIE) 到 DDR 的传输带宽，以及 4) 使用 BCDMA 的 DDR 到 DDR 传输带宽。

4.1 机器学习推理

AM64x SDK 已经集成了开源 TensorFlow Lite，用于边缘的深度学习推理。AM64x 并非专门针对实时图像处理的，但可以用于对一些边缘应用执行机器学习推理。以下示例是基于 ImageNet 数据库和 1000 种物体类别运行 TensorFlow Lite 模型进行图像分类 (224 x 224 像素，3 字节颜色)。海军少将葛丽丝·霍普 (Grace Hopper) 的示例图像安装在文件系统中 (可点击访问)。在调用 TensorFlow Lite 之前，示例 *label_image* 程序将裁剪并调整 bmp 图像大小为 224 x 224 像素。经过量化感知训练的 Mobilenetv1 网络 (mobilenet_v1_1.0_224_quant.tflite) 的推理时间基准是 280 毫秒。示例运行如下所示，也可参阅 AM64x Linux SDK (位于文件夹 /usr/share/tensorflow-lite/examples) ：

```
root@am6x-evm:/usr/share/tensorflow-lite-1.15/examples# ./label_image -i grace_hopper.bmp -l
labels.txt -m mobilenet_v1_1.0_224_quant.tflite
Loaded model mobilenet_v1_1.0_224_quant.tflite
resolved reporter
invoked
average time: 280.587 ms
0.780392: 653 military uniform
0.105882: 907 Windsor tie
0.0156863: 458 bow tie
0.0117647: 466 bulletproof vest
0.00784314: 835 suit
```

基于浮点 Mobilenetv2 模型的且分辨率 (224 x 224 x 3) 完全相同的图像的推理时间是 362 毫秒。控制台命令和打印输出如下所示：

```
root@am6x-evm:/usr/share/tensorflow-lite-1.15/examples# ./label_image -i grace_hopper.bmp -l
labels.txt -mtest/mobilenet_v2_1.0_224.tflite
Loaded model test/mobilenet_v2_1.0_224.tflite
resolved reporter
invoked
average time: 362.22 ms
0.911345: 653 military uniform
0.014466: 835 suit
0.0062473: 440 bearskin
0.00296661: 907 Windsor tie
0.00269019: 753 racket
root@am6x-evm:/usr/share/tensorflow-lite-1.15/examples#
```

控制台中打印的位于推理时间下面的数字是前五个分类结果，即 imagenet labels.txt 中 0 至 1 之间的数字和 1000 种类别之一。准确度结果是模型和输入图像的基准，而不是运行推理的器件的基准。

所有 .tflite 模型将在 AM64x 上运行，并选择量化 Mobilenetv1 和浮点 Mobilenetv2 作为共同基准，可以用于内插计算推理应用的性能。量化 Mobilenetv1 在文件系统中，Mobilenetv2 从托管模型（托管模型见 tensorflow.org）中下载。

4.2 场定向控制 (FOC) 环路

MCU+ 和 Linux SDK 包含基准测试演示应用程序，该程序包括实施图 4-1 中所示算法并显示环路性能的示例 FOC 环路。其中的函数包括 Clark、Park、逆向 Park、Sin、Cos、PI 控制器及空间矢量生成。

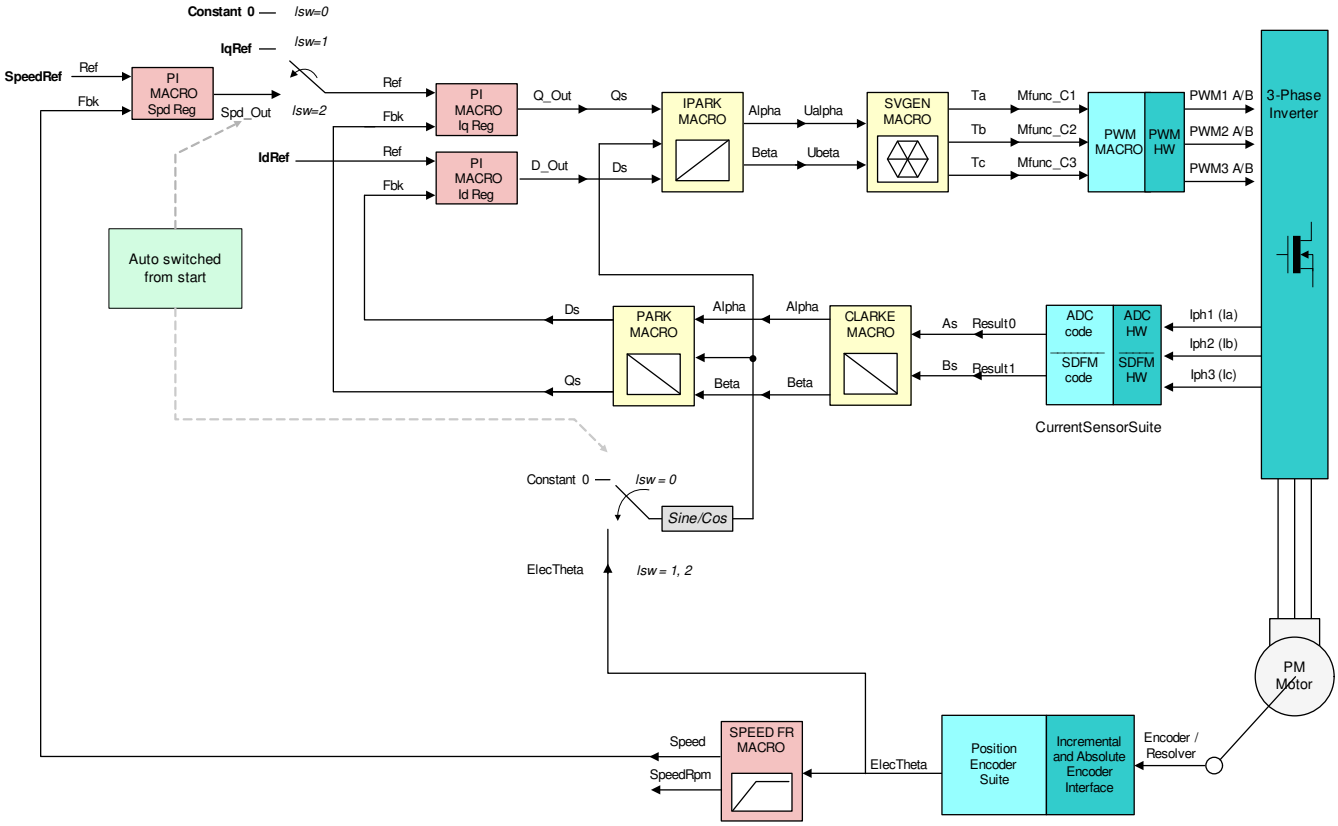


图 4-1. 速度闭环场定向控制环路

FOC 环路在 AM64x 单核 Arm Cortex-R5F 的性能如下表所示。Arm Cortex-R5F 内核多达四个，每个内核均可通过其所控制的电机独立实现该性能。

	FOC 环路执行时间（单位为微秒）
平均	0.4

4.3 使用 BCDMA 时的 PCIE 到 DDR 性能

AM64x MCU+ SDK 包含可启用 PCIE 的演示应用程序。它包括用于 AM64x 的 PCI 端点 (EP) 和 PCIE 根复合体 (RC) 配对的 PCIE 基准测试的代码示例

- PCIE 缓冲区传输 EP
- PCIE 缓冲区传输 RC

类似地，我们在 MCU+ SDK 中提供了 DMA 示例。UDMA 是 DMA 接口的通用名称，对于 AM64x 和 AM243x，它在底层使用 BCDMA。

- UDMA Memcpy 轮询
- UDMA Memcpy 中断

我们将其用作基线测量和合适的设置，以使用 PCIE 上的 BCDMA 多通道对从 EP-DDR 位置到 RC-DDR 位置的最小读取延时和带宽进行基准测试。PktDMA（例如由以太网使用）的行为方式应该非常相似。此外，借助 AM243x/AM64x，RC 或 EP 都可以启动事务。

4.3.1 测试设置

AM64x-PCIE 基准测试设置

下面是我们对 PCIE-BCDMA-DDR 读取性能（就带宽而言）进行基准测试所用的设置。

硬件详细信息：

- TMDS64EVM
- PCIE 电缆可从 Adex Electronics 获得。

有关详细信息，请参阅：

- PCIE 缓冲区传输 RC

软件详细信息：

- 测试代码可从 GitHub 链接获取，该代码以 MCU+ SDK 8.6 为基准。

> **备注**
> 默认情况下，在 MCU+ SDK 中，最多可以将 4 个 UDMA 通道分配给 R5F_0_0 内核。

如果要运行测试的通道数超过默认配置，您需要首先使用资源管理 (RM) 工具来管理内核的 UDMA 通道分配。

图 4-2. AM64x-PCIE 基准测试设置

以下是 PCIE 基准测试期间 2-AM64x 电路板之间数据流的详细信息。

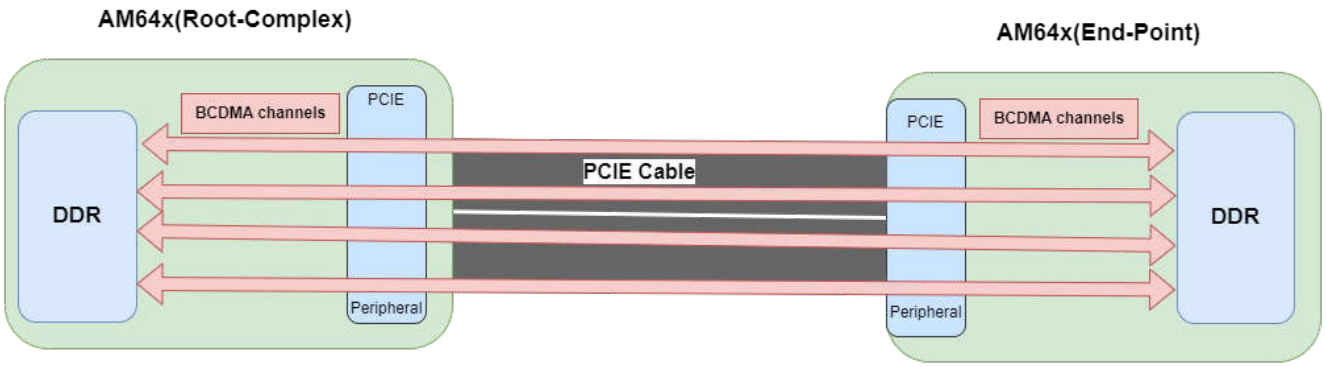


图 4-3. AM64x（PCIE 基准测试）数据流

EP 在 PCIE 中设置缓冲区，RC 将读取它并将其复制到 DDR 中

例如，在 RC 侧：

PCIE_loc_1 = 0x68000000UL + 0x00000000U

PCIE_loc_2 = 0x68000000UL + 0x01000000U

PCIE_loc_3 = 0x68000000UL + 0x02000000U

PCIE_loc_4 = 0x68000000UL + 0x03000000U

和

DDR_loc_1 = 0xA0000000 + 0x00000000U

DDR_loc_2 = 0xA0000000 + 0x01000000U

DDR_loc_3 = 0xA0000000 + 0x02000000U

DDR_loc_4 = 0xA0000000 + 0x03000000U

以此类推。

4.3.2 结果和观察

本节总结了所执行测试的结果和观察。

	PCIE 至 DDR 读取带宽（兆位/秒）				
(B)	1 个 BCDMA 通道	2 个 BCDMA 通道	4 个 BCDMA 通道	8 个 BCDMA 通道	16 个 BCDMA 通道
1	0.89	1.6	2.29	2.67	2.61
2	3.2	4.57	7.11	6.4	6.4
4	6.4	9.14	14.2	12.8	13.13
8	12.8	21.33	23.27	26.95	26.95
16	32	42.67	56.89	51.2	53.89
32	51.2	85.33	102.4	102.4	105.02
64	102.4	17.67	204.8	204.8	210.05
128	204.8	341.33	455.11	409.6	420.10
256	292.57	512	682.67	963.76	862.31
512	455.11	744.72	1170.28	1489.45	1560.38
1024	585.14	1092.26	1489.45	1771.24	1820.44
2048	712.35	1310.72	1872.46	2048	2131.25
4096	780.19	1456.35	2048	2148.72	2166.48
8192	840.20	1618.17	2202.89	2240.55	2202.89
16384	856.68	1638.4	2259.86	2289.47	2205.21
32768	868.03	1664.40	2309.64	2314.73	2208.69

图 4-4 显示了使用 BCDMA 通道时 PCIE 到 DDR 的读取性能曲线。这里，1 个 BCDAM 通道被称为 1 TRPD。此处显示的数据范围从 40B 到 1600B 不等，可实现清晰的可视化效果。

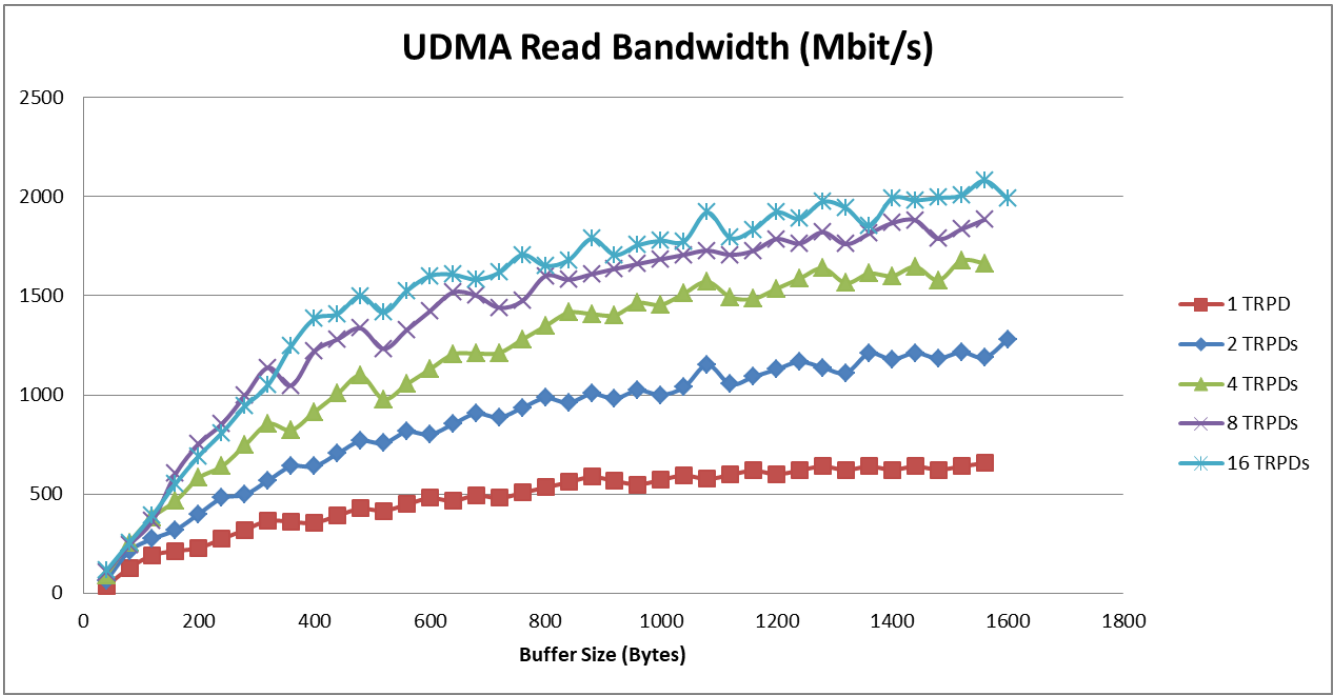


图 4-4. PCIE 性能图

观察结果：

4 个并行 DMA 传输可实现最大吞吐量，进一步添加通道将会有效果，但不会增加吞吐量。

4.4 使用 BCDMA 时的 DDR 到 DDR 性能

AM64x MCU+ SDK 包含基于 UDMA 的示例。

- UDMA Memcpy 轮询
- UDMA Memcpy 中断

在这里，我们使用这些函数来测量使用 BCDMA 和多个并行通道时 DDR 到 DDR 的传输性能（就带宽而言）。

硬件详细信息：

- TMDS64EVM

软件详细信息：

- 用于此基准测试的测试代码可在 GitHub 链接（轮询和中断）上获得，该代码以 MCU+ SDK 8.6 为基准。

备注

默认情况下，在 MCU+ SDK 中，最多可以将 4 个 UDMA 通道分配给 R5F_0_0 内核。

若要测试比默认配置更多的通道：

- 使用资源管理 (RM) 工具管理内核的 UDMA 通道分配。
- 使用所导入的 CCS 工程中的“example.syscfg”文件添加 UDMA 通道数

4.4.1 测试设置

本节介绍了使用 UDMA 为 DDR 到 DDR 复制执行基准测试所用的测试设置。

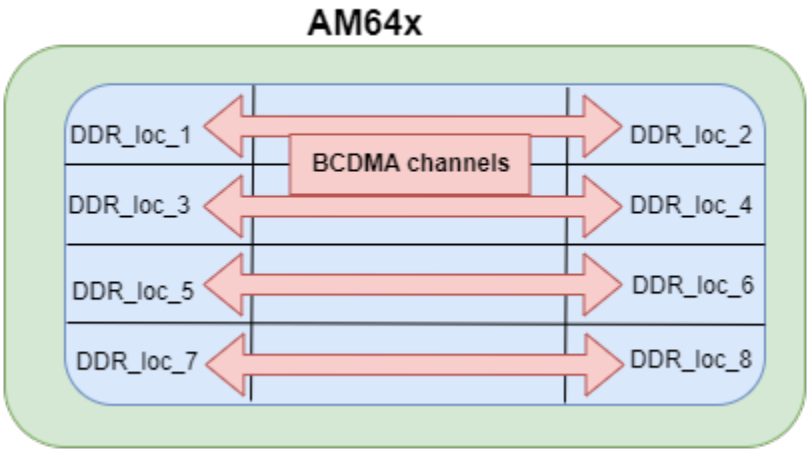


图 4-5. AM64x DDR 至 DDR 数据流

例如：

DDR_loc_1 = 0xA0000000 + 0x00000000U

DDR_loc_2 = 0xA0000000 + 0x01000000U

DDR_loc_3 = 0xA0000000 + 0x02000000U

DDR_loc_4 = 0xA0000000 + 0x03000000U

以此类推。

4.4.2 结果和观察

本节提供使用 UDMA 通道进行 DDR 到 DDR 数据复制的测试结果和观察。

缓冲区大小 (字节)	DDR 到 DDR (UDMA) 传输带宽 (兆位/秒)				
	1 个 BCDMA 通道	2 个 BCDMA 通道	4 个 BCDMA 通道	8 个 BCDMA 通道	16 个 BCDMA 通道
1	1.143	1.6	2.13	2.06	1.80
2	5.33	6.4	5.33	4.74	3.88
4	8	12.8	10.67	9.85	7.88
8	16	21.33	23.27	18.96	15.51
16	42.67	42.67	46.54	39.38	31.51
32	85.33	85.33	93.09	75.85	63.01
64	170.67	204.8	170.67	157.54	124.12
128	256	341.33	341.33	341.33	282.48
256	512	585.14	630.15	655.36	555.39
512	1024	1170.28	1489.45	1424.69	1285.02
1024	1638.4	2048	2730.67	2520.61	2520.61
2048	2340.57	2978.91	3640.89	4369.07	3236.34
4096	3276.8	4369.07	4681.14	5041.23	3615.78
8192	4096	5698.79	4946.11	4946.11	4017.53
16384	4681.14	6898.52	5140.08	5190.97	4136.39
32768	4946.11	7710.12	5322.72	5165.40	4181.76

图 4-6 显示了使用 BCDMA 通道时 DDR 到 DDR 的读取性能曲线。

1 个 BCDMA 通道被称为 1 TRPD。

此处显示的数据范围从 40B 到 1600B 不等，可实现清晰的可视化效果。

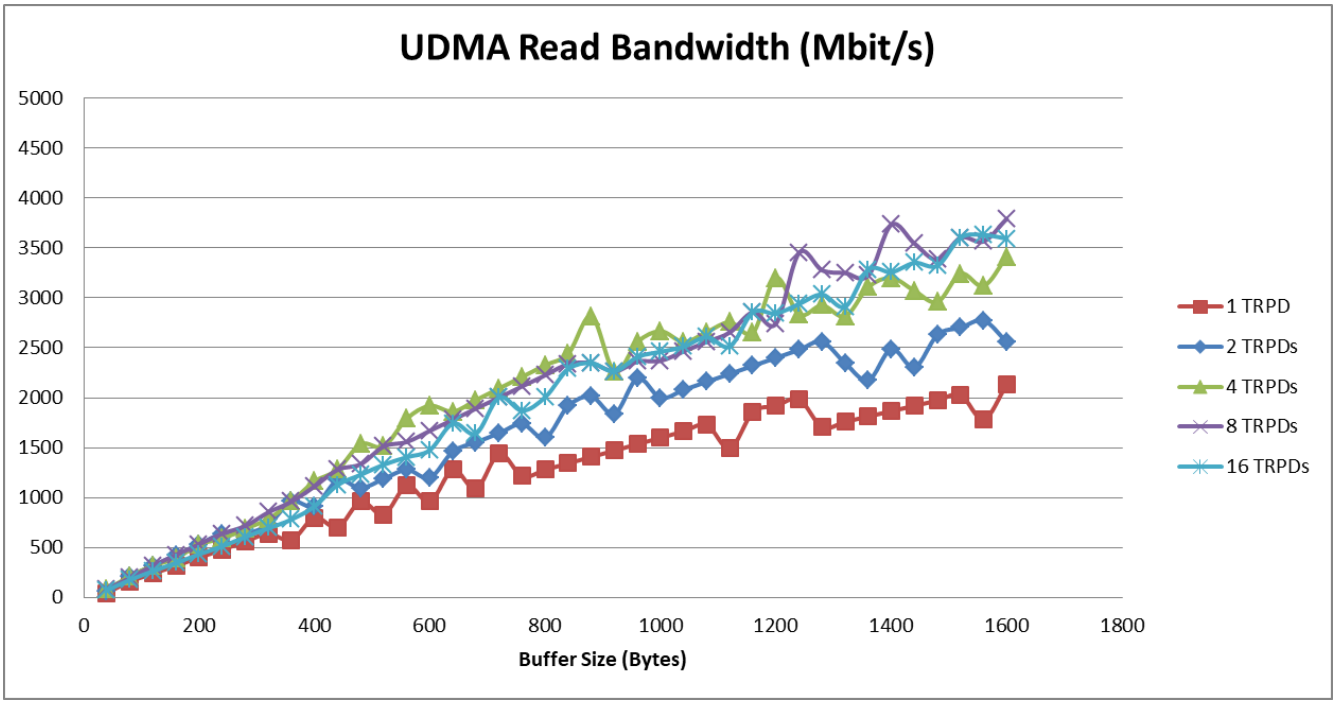


图 4-6. DDR 性能图

观察结果：

4 个并行 DMA 传输可实现最大吞吐量，进一步添加通道将会有效果，但不会增加吞吐量。

5 参考文献

- CoreMark-Pro
- STREAM McCalpin, John D.：“STREAM：高性能计算机中的可持续存储器带宽”，持续更新技术报告 (1991-2007)，可从下述网址访问：http://www.cs.virginia.edu/stream/
- Ne10 数学库
- CMSIS 库
- 托管模型见 tensorflow.org
- OpenSSL

6 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 * (March 2023) to Revision B (January 2024)	Page
• 添加了节 4.3	10
• 添加了节 4.3.1	10
• 添加了节 4.3.2	12

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