TDA4 片内外系统中的 Memory 管理

王力，陆思恺 Central FAE

摘要

Jacinto™ 7 TDA4 系列处理器是 TI 公司基于 Keystone 架构推出的最新一代汽车处理器，主要致力于辅助驾驶系统（ADAS）和自动驾驶（AD）领域芯片解决方案。TDA4 系列汽车处理器基于片上多核异构芯片架构，16nm 系统工艺不仅为芯片带来了高系统集成度，而且大幅降低了多功能高级汽车平台设计所需的外设复杂度以及成本。性能上，在提供高达 ASIL-D 的系统功能安全等级支持的同时，以领先于市场的性能/功耗比为深度学习/视觉加速提供了卓越的处理能力。

在 TDA4 整机系统中存在多种片内外的 Memory 内存。例如其中片外有支持系统启动的 OSPI/QSPI（Octal/Quad Serial Peripheral Interface）NOR Flash 以及数据存储使用的 MMC（Multi Media Card），亦或程序运行所需的主存 DDR（Double Data Rate SDRAM）等。在 TDA4 芯片内部，由于多核异构的设计框架，不仅为了保证各个核心自身运行的效率，也为了提高各个核心之间的数据交换，TDA4 也配置有例如 MSMC（Multicore Shared Memory Controller），OCMC（On Chip Memory Controller），TCM（Tightly-Coupled Memory）以及 Cache 等多种内存介质以及控制器。

无论 TDA4 片内外，每个 Memory 对于系统都有着特殊作用，本文通过对典型 TDA4 整机系统中片内外的 Memory 进行逐一剖析，分析了其功能，配置流程以及优化方向，并在最后分别对其在系统中的功能定位进行示例阐述，为 TDA4 系统中的 Memory 管理以及优化提供了方向。

目录

图

表

1. TDA4 片内外存储系统介绍

以 TDA4VM 为例，在典型的系统应用中，在 TDA4 芯片外部一般挂载 LPDDR4 作为系统运行内存，OSPI 作为固件启动介质，MMC/UFS 作为文件系统存储设备以及使用 EEPROM 作为系统 Device ID 识别等；在 TDA4 芯片内部，配置有 OCMC 可作为 TDA4 ROM code 以和 MCU 域程序的运行内存，以及 MSMC 作为优化 TIDL 运行效率的 SRAM 或者 A72 核心的 L3 Cache，除此之外，还有众多的 TCM 以及 L1/L2 Cache 等内存为多核协同工作提供优化。以 TDA4VM 为例，其配置如表 1 所示。

表 1 TDA4VM 片内外 Memory 配置

	片外 Memory					片内 Memory			
	DDR	**OSPI/QSPI**	**MMC/SD**	**UFS**	**EEPROM**	**MSMC**	**OCMC**	**TCM**	**Cache**
TDA4VM	EMIFx1	OSPI x1 QSPI x1	MMCSD x3	UFS x1	Based on I2C	MSMC x1 With 8MB	1MB in MCU Domain	64KB in R5F Core	80KB L1 per A72… 32KB L1 per R5F…

TDA4 家族中有众多的 PN 系列 SoC 芯片，各不相同的 PN 有着不相同的内存大小/控制器数量的配置等，表 1 仅以 TDA4VM 为例。

1.1 TDA4 片外存储系统及接口

以 TDA4VM 官方 EVM 为例，其典型的外部 Memory 框图如图 1 所示。其中

- TDA4 能够支持功耗更低的 LPDDR4 作为运行内存，其通过标准的 EMIF 接口与 TDA4 连接，总线宽度为 32bit，速率上支持常用的 3733MHz/4266MHz，从而理论 DDR 带宽能够最高达到 17Gbps.
- OSPI/QSPI 是基于常规 SPI 协议拓展的串行外设接口，作为八进制/四进制读取的 NOR Flash 存储介质，其数据吞吐量更高。并且由于 OSPI 效率更优于 QSPI，一般使用 OSPI 作为 TDA4 前期引导加载程序 SBL 以及 R5F 核心固件加载的存储介质。
- EMMC 作为 NAND Flash，其内存空间更大且成本更低，通常用来加载复杂的文件系统文件。在 TDA4 中提供了三个 MMCSD 控制器 MMCSD0/MMCSD1/MMCSD2，支持 EMMC 5.1 协议且其数据总线分别为 8bit/4bit/4bit，用户可选择进行 EMMC/SD 卡的挂载及驱动。
- UFS Flash 是基于串行通信的传输机制，与并行传输且半双工模式的 EMMC 相比，由于 UFS 是基于串行通信的全双工模式，其读写速度也远大于 EMMC。在 TDA4 中集成了一个 UFS 控制器，能够支持 UFS2.1 的协议。
- EEPROM 由于其断电数据不丢失的属性，可以用来在板载系统中存储固定的版本信息，例如芯片/软件型号以及特定配置等。其基于 I2C 总线挂载，理论上个数不受限制，用户可根据实际需求进行配置。

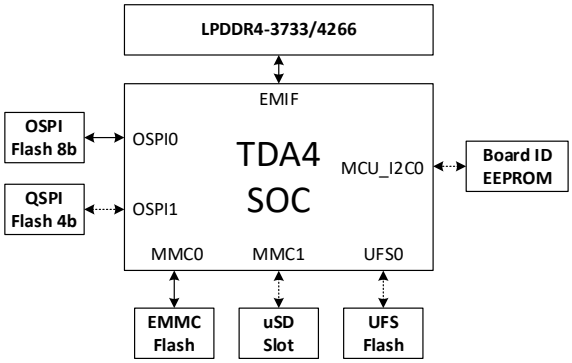

图 1 TDA4 典型外部存储设备框图

各种不同的 Memory 存储介质，由于其自身性质的原因，在整机系统启动以及工作时承担着各不相同的角色，用户需要根据实际应用中对于启动时间、运行效率、成本控制等各个方面的需求对 Memory 的种类，大小等进行最优化选择。

1.2 TDA4 片内内存系统及控制器

在 TDA4 系列 SoC 中，出于安全功能考虑，芯片内部划分为 MAIN Domain 以及 MCU Domain，两个域之间电源时钟相互隔离，可通过内部总线进行数据交换，其内部的内存框图如图 2 所示。其中以 TDA4VM 为例：

- MAIN 域中配置有 8MB 大小的 MSMC 内存，其向上与 A72，R5F 处理器核心以及 C7x DSP 核心等连接，向下与 DDR 以及通过 NAVSS 与内部总线连接。数据总线位宽 512bit，地址寻址总线位宽 40bit。
- MCU 域中配置有 1MB 大小的 OCMC，直接挂载在 MCU 域的内部总线上，能够作为 TDA4 ROM code 以及启动引导程序 SBL/SPL，或 MCU 域 OS 的运行内存。
- TCM 是集成在 R5F 核心中的低延迟的 Memory，根据功能的不同又可以划分为 ATCM 和 BTCM，这部分 Memory 离 CPU 最近所以效率最高，不需要 Cache 的操作。
- 为了保证每个核心的高效运行，TDA4 内部为每个处理器核心都会配置有内部的 L1/L2 Cache，能够大大提高处理器核心对数据的读取以及处理速度。

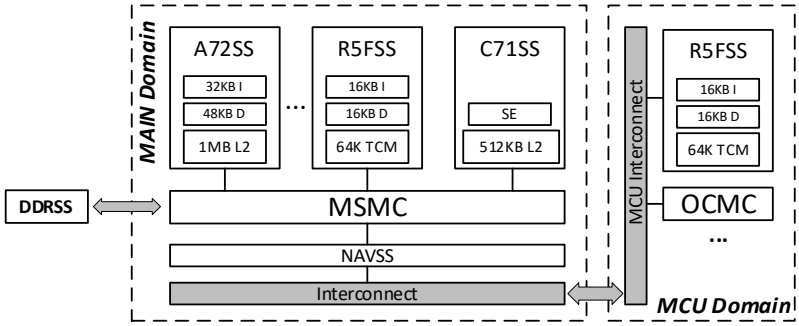

图 2 TDA4 典型内部内存框图

TDA 内部的内存 Memory 会影响着内部的数据交互速率以及处理器的处理能力等，例如 MSMC，OCMC 可对内存空间进行功能划分进而实现不同方向的优化，TCM 和 Cache 可配置预加载进而提升处理器处理能力等等。用户需要根据数据在内部的传输流程，进而对这些内存进行分配以及管理。

1.3 地址映射模块 RAT

在 TDA4 系统中，R5F，C66 等核心为 32bit 处理器，而 A72 以及 C7x 等为 64bit 处理器，同时其它一些例如 VPAC_SRAM 内存或者 PCIe 控制器等模块也在 40bit 地址段上。这样会导致 R5F 等处理器最多只能寻址访问 32bit 也就是 4G 大小的地址空间，导致 32bit 处理器无法访问高于 32bit 的内存以及控制器的局限，所以在 TDA4 中提供了地址映射模块 RAT（Region Address Translation），能够将输入的 32bit 地址段映射成 64bit 地址段（实际应用中 48bit 位即可），从而实现片上系统的相互访问。如图 3 所示为 RAT 功能框架示意图。

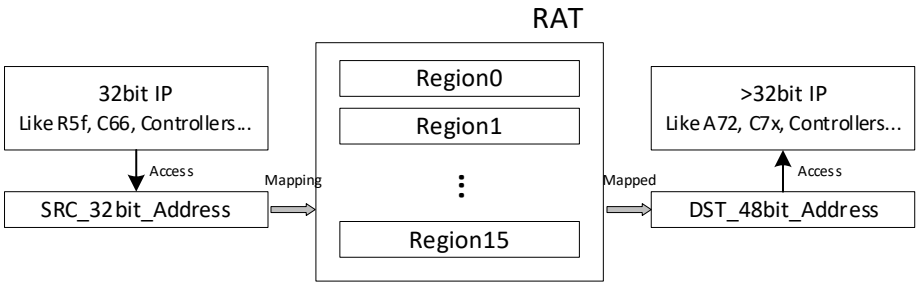

图 3 RAT 功能示意图

如图 3 所示，每个 RAT 映射模块含有 16 个 region，用户可选择任意一个 region 进行映射，每个 region 最大可映射 4GB 内存访问空间。用户需要将指定的 32bit 输入地址作为 RAT 的初始映射地址 RAT_BASE_k；并在 RAT 模块中设置所需要映射的高 48bit 地址空间，其由高 16 位 RAT_TRANS_U_k 以及低 32bit 位 RAT_TRANS_L_k 构成；然后在 RAT 控制寄存器 RAT_CTRL_k 中对 region 大小进行设置并使能，其中 k 取值大小为 0~F。其示例代码如下所示：

```
#define RAT_BASE 0x****_****
#define REGION_ID *
*(unsigned int *)(RAT_BASE + 0x20 + (REGION_ID*0x10)) = 0x80000013;
*(unsigned int *)(RAT_BASE + 0x24 + (REGION_ID*0x10)) = 0x03000000; //IN ADDRESS
*(unsigned int *)(RAT_BASE + 0x28 + (REGION_ID*0x10)) = 0x02000000; //Lower 32 bits of the real physical address.
*(unsigned int *)(RAT_BASE + 0x2C + (REGION_ID*0x10)) = 0x0000004F; //Upper 16 bits of the real physical address.
```

其中 RAT_BASE 为所选择用来指定映射的 RAT 模块的地址，例如 MCU 域 R5F 的 RAT 控制器地址为 MCU_ARMSS_RAT_CFG（0x40F90000），详情可查询 TDA4 TRM 手册；REGION_ID 代表使用 RAT 中的哪一个 region，大小为 0~F。可以看到，上述代码作用为将 32bit 地址 0x3000_0000 映射到 48bit 地址 0x0000_004F_0200_0000，映射空间大小为 8Byte 并使能 RAT。设置完成后，处理器或控制器对地址 0x3000_0000 的访问将映射到 0x0000_004F_0200_0000 地址段。

通常在对内部内存资源进行分配管理，或者对外部 Memory 进行寻址访问时，需要考虑 RAT 的使能以及配置。

2. 片外存储系统

TDA4VM 的片外存储系统主要为 SoC 的系统启动以及功能实现提供了保证。DDR 作为主存用以支持系统以及应用运行，但数据会断电丢失；而 OSPI/QSPI Flash，EMMC/SD，UFS 以及 EEPROM 等则有着断电数据不丢失的特性，分别可用于存储系统启动镜像，文件系统，以及板级配置文件等。

2.1. LPDDR4

LPDDR (Low-Power Double Date Rate) 是 DDR SDRAM 的一种。与传统 DDR 相比 LPDDR 体积更小，功耗更低，因而被广泛地应用于移动式电子产品中，如智能电话，平板电脑，汽车电子等。LPDDR 的运行电压（工作电压）相比 DDR 的标准电压要低，从第一代 LPDDR 到 LPDDR4，每一代 LPDDR 都使内部读取大小和外部传输速度加倍。其中 LPDDR4 可提供 32Gbps 的带宽，输入/输出接口数据传输速度最高可达 3200Mbps，电压降到了 1.1V。TDA4VM 与 LPDDR4 的硬件设计请参考设计手册 LPDDR4 Board Design and Layout Guidelines。

硬件设计完毕后需要在 SBL 启动流程中对 DDR 进行初始化配置等，而后所挂载的 DDR 才能够作为系统主存进行使用。推荐使用 TI 提供的 DDRSS Register Configuration Tool 在 Excel 表格中对所使用的 DDR 芯片的参数信息进行填写，并生成对应的配置文件，其中 Excel 表格会根据输入的 DDR 参数信息输出四种格式的 DDR 配置文件，用户可根据设计需求使用相对应的文件对 DDR 进行初始化以及验证。图 4 为使用 DDRSS Configuration Tools 对 DDR 进行配置以及验证的大致流程。

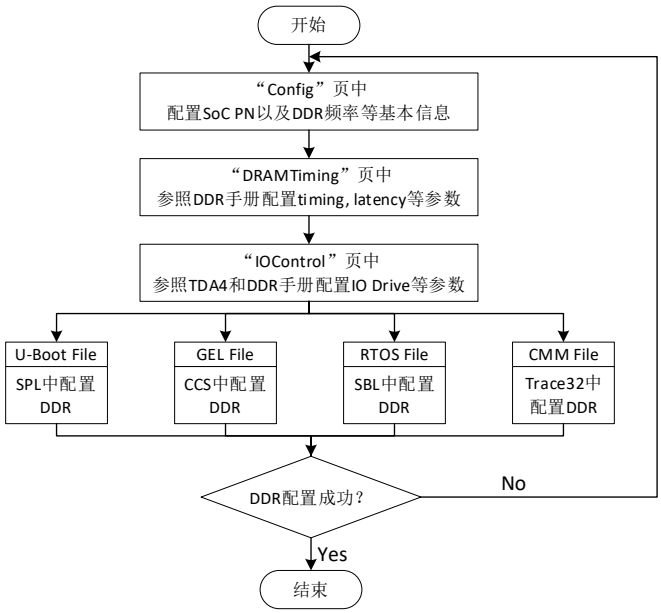

图 4 DDRSS Register Configuration Tool 配置 DDR 流程

以 TDA4 中典型的 SBL 启动为示例，DDR 的初始化以及配置等都会在 SBL 代码逻辑中完成，其在 SBL 中的大致配置流程如图 5 所示。值得注意的是，DDR 作为系统运行的主存，TDA4 内部的各个核心系统运行内存，应用运行内存以及核间共享内存等都需要在 DDR 内存地址中进行映射，并保证不产生冲突，否则会存在系统或应用奔溃的风险；DDR 上的内存映射请参照 DDR Memory Map User Guide。

图 5 SBL 中配置 DDR 流程图

值得一提的是，系统在运行过程中，DDR 的实时读写速度以及带宽等都会影响系统的高效运作。在 TDA4 系统中有多种测量 DDR 实时读写以及带宽方法，用户可根据这些方法来对系统状态进行监测：

a. 在代码中可通过 *appPerfStatsDdrStatsExport()*对 DDR 实时带宽占用进行获取。
b. 可以使用 lmbench 工具，解压后修改 scripts/os 中：os=arm-linux 同时修改 scripts/compiler 中 cc=所使用的编译链的路径，该工具的使用方法可参考相关资料。

2.2. OSPI

OSPI/QSPI 是基于常规 SPI 协议拓展的串行外设接口，分别以八进制/四进制对 NOR Flash 存储介质进行读取。以 TDA4VM 为例，其在 MCU 域中提供了两个 OSPI 的控制器模块，都支持对 OSPI 以及 QSPI Flash 的读写。一般由于 OSPI 效率更优于 QSPI，所以在使用单一外部 Flash 时，常使用 OSPI 作为 TDA4 前期引导加载程序 SBL 以及 R5F 核心固件加载的存储介质。图 6 所示为 TDA4 内部 OSPI 控制器的大致结构示意图。

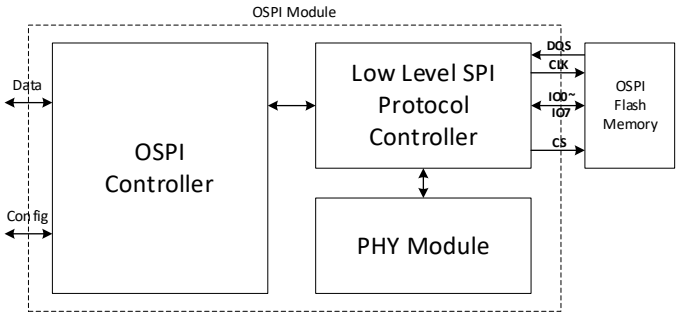

图 6 OSPI Module 示意图

OSPI 作为 TDA4 外挂的 Flash Memory，其和 DDR 一样也在 SBL 阶段进行初始化。例如在 SBL 阶段，TDA4 会通过 *SBL_ReadSysfwImage()*函数对 DMSC 的固件 Sysfw 进行读取，以及 *SBL_BootImage()* 对 MCU 域 R5F 固件进行加载等等，此处不再赘述。

值得一提的是，TDA4 为 OSPI 提供了 133MHz 以及 166MHz 两种不同的工作时钟选择，其可在下面代码段中进行配置：

```
pdk/packages/ti/boot/sbl/src/ospi/sbl_ospi.c 文件中
配置宏 ospi_cfg.funcClk = OSPI_MODULE_CLK_133M;
```

在一些场景下，例如快速启动，快速唤醒场景下，用户希望得到更快的 OSPI 读写速率，TDA4 的 OSPI controller 提供了 PHY mode，通过对 tuning 获得最佳的时序配置，这样能够获得不经过分频的更快的时钟信号，同时也弥补了单个硬件的差异，提高了稳定性。并且在开启 OSPI PHY Mode 之后，需要打开 OSPI 的 DMA 开关，才能在 PHY mode 下使用 DMA 数据搬运，而不是 memory copy 的搬运形式，这样能够显著的提升 OSPI Flash 的读取速度。

```
使能 OSPI 的 PHY mode：
Pdk/packages/ti/boot/sbl/src/ospi/Sbl_ospi.c 文件中
配置宏 ospi_cfg.phyEnable = true;

使能 OSPI 的 DMA 搬运：
pdk/packages/ti/boot/sbl/sbl_component.mk 文件中
配置宏 SBL_USE_DMA = yes
```

在 TDA4 对 OSPI Flash 读取之前，需要先将所有要用到的镜像烧录至 OSPI Flash 中，对于 TDA4-GP 芯片，可以参考 GP 烧录应用手册进行烧录；对于 TDA4-HS 芯片，可以参考 HS 烧录应用手册进行烧录。

2.3. MMC

以 TDA4VM 为例，其在内部 MAIN 域集成了三个 MMCSD 模块，每个都对应一个 MMCSD 的控制器，用户可根据实际需求对其进行硬件连接以及配置，例如以 TDA4VM 的官方 EVM 板为例，其在 MMCSD0 外接的 EMMC Memory，并通过 MMCSD1 外接的 SD 卡，可分别或同时使用 EMMC 以及 SD 卡做文件存储读写。如图 7 所示为 TDA4VM 内部的 MMCSD 模块示意图。

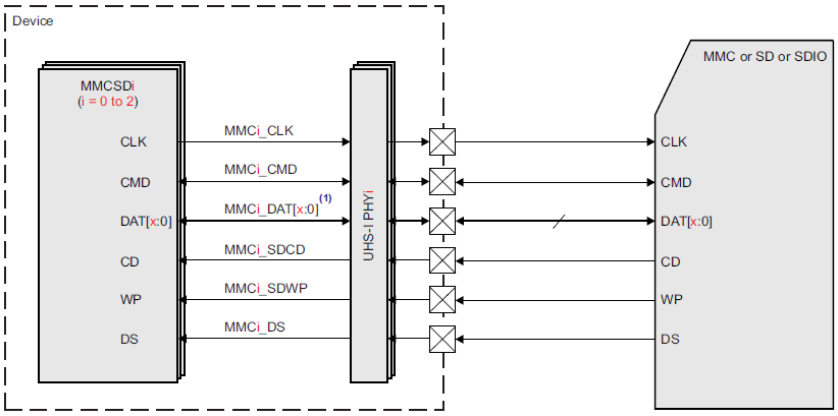

图 7 OSPI 模块示意图

虽然 MMC 以及 SD 卡速度略慢于 OSPI Flash，但其空间大且成本相应的也更低，所以常用于存储系统运行时的文件系统，应用场景非常普及，此处不再赘述。

2.4. UFS

UFS (Universal Flash Storage) 是一种高性能接口的通用闪存存储器，其是基于串行通信的传输机制，与并行传输且半双工模式的 EMMC 相比，由于 UFS 是基于串行通信的全双工模式，其读写速度也远大于 EMMC。

TDA4VM 在 MAIN 域提供了一个 UFS2.1 的接口，其支持 GEAR1~ GEAR3 的执行标准，2 lane 的读写速度分别能达到 1.46 Gbps，2.91 Gbps 以及 5.83 Gbps。图 8 为 TDA4VM 中 UFS 模块示意图，详细配置请参照 TDA4 TRM 手册。

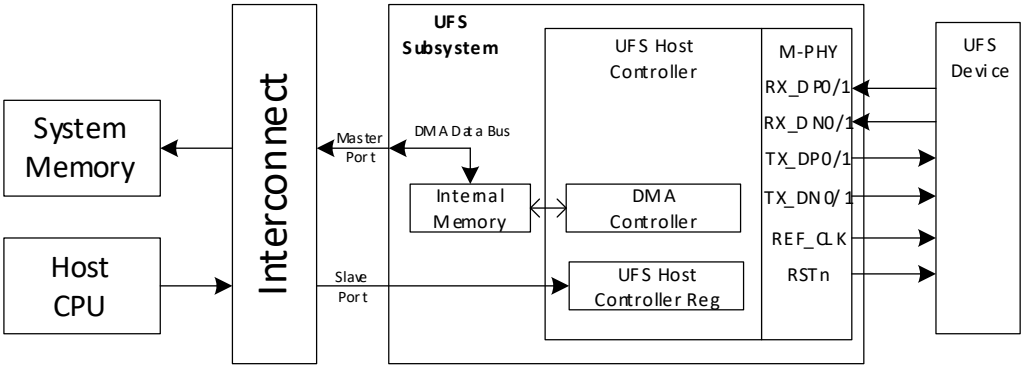

图 8 UFS Module 示意图

UFS 通常也用作外部文件系统的存储，其相比较于 EMMC 虽然速度更快，但同时成本也更高，而文件系统的读取通常不需要更快的读写速度，所以用户需要根据实际需求来进行选择。

2.5. EEPROM

EEPROM（Electrically Erasable Programmable Read Only Memory），电可擦除可编程只读存储器，是一种掉电之后数据不会丢失的存储芯片。EEPROM 的擦除不需要借助于其它设备，它是以电子信号来修改其内容的，而且是以 Byte 为最小修改单位，不必将数据全部擦除才能写入，彻底摆脱了 EPROM Eraser 和编程器的束缚。有并行接口和串行接口的 EEPROM，但并行接口基本上被串行接口的 EEPROM 取代，常见的串行接口有 I2C，SPI，Microwire, 1-wire 等。

在 TDA4 芯片内部集成了众多 I2C，SPI 控制器，例如 TDA4VM 在 MAIN 域中集成了 7 组 I2C 控制器，MCU 域中集成了 2 组 I2C 控制器，在 WKUP 域集成了一组 I2C 控制器。理论上 EEPROM 作为 I2C 挂载的 slave device，其个数不受限制，用户可根据系统设计需求对 EEPROM 进行配置。例如在 TDA4VM 的官方 EVM 板中，有挂载在 MCU_I2C0 上用于 BOOT 参数配置的 EEPROM，存储板级 Board ID 信息的 EEPROM 以及网口参数配置的 EEPROM 等。

EEPROM 的配置非必须，视设计需求而定，软件可通过 I2C Slave ID 进行访问呢，并且对硬件要求低，此处不赘述。

3. 片内内存系统

从 BSP 以及应用层来看，TDA4 片内可供配置并影响运行效率的内存主要有 MSMC，OCMC，R5F TCM 以及 Cache 等，其主要原因是因为芯片内部的 Cache 类内存通常比 SRAM 更快，而 SRAM 又比 DDR 等外部 Memory 运行效率更高，所以在某些特定应用下，可根据功能要求对上述内存进行灵活配置，从而达到性能最优。

3.1. MSMC

TDA4 内部都配置有一块共享内存及控制器 MSMC，例如 TDA4VM 内部就配置有大小为 8MB 的 MSMC，MSMC 通过 512bit 的高带宽数据总线与其它 IP 连接，其系统框图如图 9 所示。

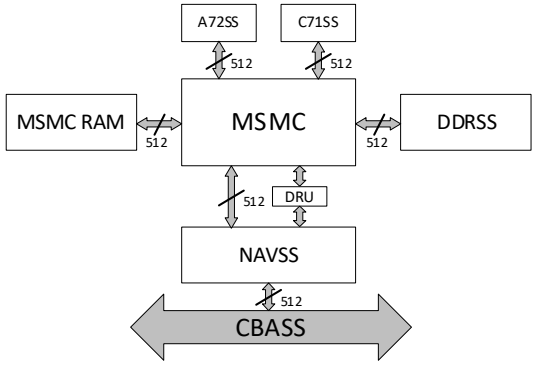

图 9 MSMC 系统框图

在 TDA4 中，MSMC 可以配置为 SRAM 给 C7x+MMA 使用，这样能够使得深度学习 TIDL 在做卷积计算时，其中间层结果不用频繁的吞吐 DDR，从而能够大幅的提升 TIDL 的卷积计算速度并大大的降低对 DDR 带宽的占用。同样的，MSMC 也直接与 A72 处理器核心连接，能够配置为 A72 核心的 L3 Cache，从而使得 A72 程序在运行时对速度的读写以及处理速度更快，从而提高 A72 核心的处理能力。

以 PSDKRA 7.3 为例，下述步骤将描述如何将 MSMC 内存大小在 C7x SRAM 以及 A72 L3 Cache 之间进行划分：

- 由于 TIDL Import 工具在对模型文件进行转换时也需要对 MSMC 大小进行配置，所以首先需要需要在 TIDL Import 配置文件中对 DMSC 大小进行指定。在如下文件中：

```
/tidl_j7_02_00_00_07/ti_dl/test/testvecs/config/import/device_config.cfg 文件中：
# Size of L3 (MSMC) SRAM Memory in KB which can be used by TIDL
MSMCSIZE_KB      =  7968
```

其中 7968 代表所有 8MB 的 MSMC 都分配给 C7x 作为 SRAM 使用，例如若想将 MSMC 中的 5MB 作为 SRAM 使用，此处需将 7968 改为 4896。TIDL Import 配置更新后，所使用的 TIDL 模型需要重新导入。

- 然后需要在驱动层的配置文件中将 MSMC 的大小对应进行同步更新，如下文件所示：

/pdk_jacinto_07_03_00_29/packages/ti/drv/sciclient/soc/v1/sciclient_defaultBoardcfg.c 文件中：

```
…
/* boardcfg_msmc */
.msmc = {
    .subhdr = {
        .magic = BOARDCFG_MSMC_MAGIC_NUM,
        .size = sizeof(struct boardcfg_msmc),
    },
    .msmc_cache_size = 0xC,
},
```

此处 msmc_cache_size 代表将要分配给 MSMC 作为 SRAM 使用的内存大小，例如此处 0xC 表示将 MSMC 中的 5MB 作为 SRAM 使用，3MB 作为 A72 Cache 使用。其取值与内存分配大小的关系如表 2 所示：

表 2 驱动配置文件中 MSMC SRAM 与 Cache 分配示意表

msmc_cache_size	内存分配
0x0	MSMC 所有的 8MB 内存大小都配置为 SRAM
0x6	MSMC 的 1.5MB 大小配置为 Cache, 其余 6.5MB 配置为 SRAM
0xC	MSMC 的 3MB 大小配置为 Cache, 其余 5MB 配置为 SRAM
0xF	MSMC 所有的 8MB 内存大小都配置 Cache

- 接着由于 TDA4 多核异构的架构，每个核心以及资源的分配需要提前对内存进行划分，避免冲突，所以需要在如下文件中相应的对内存分配进行修改：

/vision_apps/apps/basic_demos/app_tirtos/tirtos_linux/gen_linker_mem_map.py 文件中：

```
…
dmsc_msmc_size   = 64*KB;
mpu1_msmc_addr   = msmc_mem_addr;
mpu1_msmc_size   = 128*KB;
c7x_1_msmc_addr  = mpu1_msmc_addr + mpu1_msmc_size;
misc_msmc_stack_size = 32*KB;
c7x_1_msmc_size  = 5*MB - mpu1_msmc_size - dmsc_msmc_size - misc_msmc_stack_size;
dmsc_msmc_addr   = c7x_1_msmc_addr + c7x_1_msmc_size + misc_msmc_stack_size;
```

这里需要将 c7x_1_msmc_size 计算式中的总大小改为 5MB 大小，修改完成后，需要手动运行此 Python 文件重新生成内存分配文件，如下所示：

```
cd vision_apps/apps/basic_demos/app_tirtos/tirtos_linux
./gen_linker_mem_map.py
```

- 最后需要将所有固件文件重新编译并更新，以板级配置文件为例，如下编译指令进行更新：

```
cd vision_apps/apps/basic_demos/app_tirtos/tirtos_linux
./gen_linker_mem_map.py
```

值得注意的是，此处对于 MSMC 大小的划分作为 SRAM 以及 Cache 使用时，需要以 1MB 大小对齐，即以 1MB 作为最小单位进行划分。

同样的，此处 MSMC 支持 ECC 校验，其每 256bit 数据将产生 10bit 的汉明码，具体使能方式此处不赘述，请参照 TRM 手册以及 safety Manual。

3.2. OCMC

在设计之初，出于功能安全以及提高系统整体运行效率考虑，在 TDA4 在芯片内部集成了一块挂载在总线上大小为 1MB 的 SRAM。因为 TDA4 中 MAIN 域以及 MCU 域可以做到相互隔离，且功能安全分别最高支持 ASIL-B 以及 ASIL-D。所以为了确保在双边能够正常交互的前提下，在极端情况发生时其依然能够独立运行且不相互干扰，就在 MCU 域设计了 OCMC 的存在。其在内存中的地址映射以及分配如表 3 所示：

表 3 OCMC 大小及内存分配

Memory	Address	Usage
OCMC	0x41C0_0000	768KB ~ Free for boot image
	0x41CB_FFFF	
	0x41CC_0000	255KB ~ Reserved by ROM Code
	0x41CF_FFFF	

由表 3 可知，OCMC 在整个 TDA4 启动过程中有着重要地位，如图 10 所示为 OCMC 在 TDA4 系统启动过程中不同阶段时的作用范围。

- 在系统上电初期，TDA4 ROM code 就在 OCMC 中执行，ROM code 将会对芯片的启动方式，芯片类型等做基础校验，并引导后续系统启动。
- ROM Code 执行完毕后，MCU1_0 作为系统中最先启动的核心，其由 SBL（Second Boot Loader）引导启动，其中 SBL 也通过外部存储设备（例如 OSPI）加载到 OCMC 中运行。
- SBL 运行过程中，将会拉起其余核心，例如 DMSC，MCU2_x 等以及 MCU1_0 的系统镜像。在 SBL 生命周期运行完毕后，OCMC 中将会运行 MCU1_0 的 OS，服务进程以及应用程序等。例如 AUTOSAR、Resource Mgmt Service，Power Mgmt Service 以及 application 等。

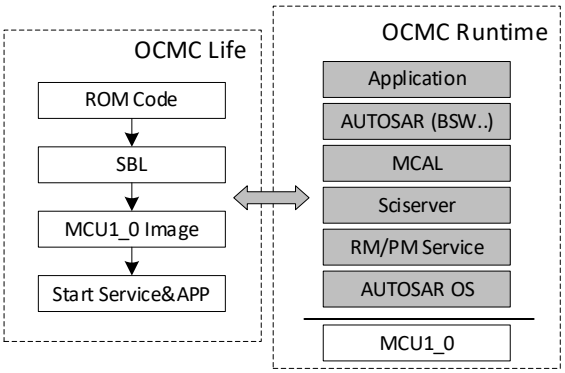

图 10 OCMC 运行流程

可以预见的，由于 OCMC 的特性，一旦图 10 中 OCMC Runtime 的 image 过大，将会导致 OCMC 内存大小不够而无法运行，所以用户需要根据优先级设计，仅将必须的应用设计到 MCU1_0 的 OCMC 中运行，其它的可通过 MAP 文件 link 到 DDR 中运行等。如下所示，可通过 PDK 的编译连接文件对代码的代码段，数据段等进行手动划分以及布置，此处不在赘述。

```
pdk_jacinto_08_01_00_36/packages/ti/build/j721e/linker_r5.lds 文件中：
/* Memory Map */
MEMORY
{
   …
   MCU0_R5F1_ATCM (RWIX)   : origin=0x41400000 length=0x8000
   OCMC_RAM (RWIX)          : origin=0x41CE3100 length=0x1CA00
   DDR0   (RWIX)        : origin=0x80000000 length=0x80000000     /* 2GB */
   ….
}
```

值得注意的是，由于 MCU 域电源时钟都独立存在，即使功能复杂的 MAIN 域发生极端意外，MCU 依然能够基于 OCMC 独立运作。也正是因为 OCMC 的这个特性，所以 MCU 域常用来运行功能安全较高的系统及应用，或者用来监控 MAIN 域系统的运行状态等。

3.3. TCM

在 R5F 处理器核心中，配置有两个轻量但高效的内部 Memory TCM（Tightly-Coupled Memories）。TCM 在处理器内部的运行效率非常高，等同于内部的 L1 Cache，但是相比较于 L1 Cache，其又能够通过 VBUSM 接口被外部资源操作，例如填充预加载指令以及数据等，从而能够降低 Cache Miss 的次数，使得代码运行效率更高。

其中 R5F 中的 TCM 分为两块，ATCM 以及 BTCM，每个大小均为 32KB；其中 BTCM 又分为两个 bank，B0TCM 和 B1TCM，每个大小为 16KB。其大致架构图如图 11 所示。

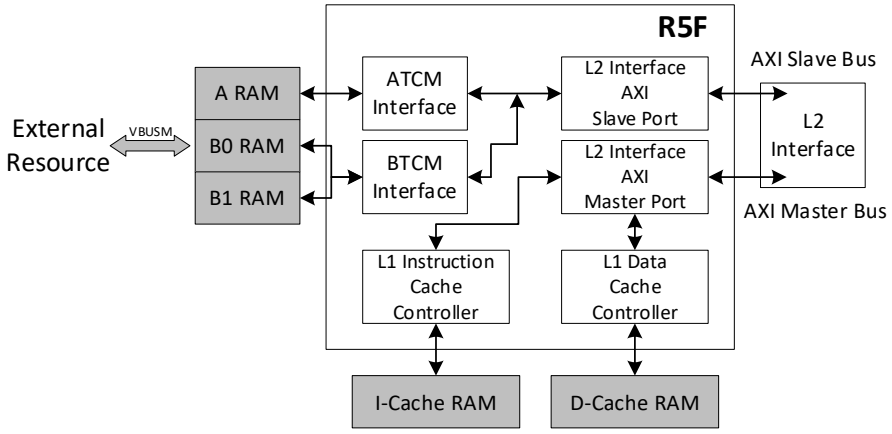

图 11 R5F 中的 TCM 及 Cache 示意图

在 R5F 中 ATCM 和 BTCM 都是可以按照用户需求进行 Enable/Disable 的，当 TCM 被使能作为系统运行 Memory 使用时，其一般可以用来存储中断向量表（ISRs），或者高频访问的数据以及变量等，这样由于其作为 L1 Cache 的存在，使得其访问速度远快于外部 Memory，从而提高运行效率。同样的，R5F 处理完成的数据，也可以暂存在 TCM RAM 中，从而被外部资源获取。但值得注意的是，外部通过 VBUSM 访问 TCM RAM 的优先级是低于 R5F 访问 TCM RAM 的，且在系统运行时，要避免两者同时对 TCM 进行读写，这样会导致访问出错。

在 TDA4VM 中，ATCM/BTCM 的使能以及分别对应的地址设置等是通过内部信号线控制的，软件上可通过寄存器 *CTRLMMR_MCUSEC_CLSTR0/1_CORE0_CFG* 进行配置，其介绍如表 4 所示：

表 4 CTRLMMR_MCUSEC_CLSTR0_CORE0_CFG 寄存器

Bit	Field	Type	Reset	Description
31-16	RESERVED	R	0h	Reserved
15	NMFI_EN	R/W	0h	Enable Core0 Non-Maskable Fast Interrupts
14-12	RESERVED	R	0h	Reserved
11	TCM_RSTBASE	R/W	1h	Core0 A/BTCM Reset Base Address Indicator 0h - BTCM located at address 0x0 1h - ATCM located at address 0x0
10-8	RESERVED	R	0h	Reserved
7	BTCM_EN	R/W	1h	Enable Core0 BTCM RAM at reset
6-4	RESERVED	R	0h	Reserved
3	ATCM_EN	R/W	0h	Enable Core0 ATCM RAM at reset
2-0	RESERVED	R	0h	Reserved

值得注意的是，当第 3bit 以及第 7bit 使能时，TCM 内部信号线 *CPUn_INITRAMA* 被置高，意味着 ATCM/BTCM 都被使能；同时当第 11bit 为 1 时，TCM 内部信号线 *CPUn_LOCZRAMA* 为高，此时 ATCM RAM 的起始地址为 0x0000_0000，BTCM RAM 起始地址为 0x4101_0000，大小分别都为 32KB。当 ATCM/BTCM 都使能且第 11bit 为 0 时候，TCM 内部信号线 *CPUn_LOCZRAMA* 为低，此时 ATCM RAM 的起始地址为 0x4101_0000，BTCM RAM 起始地址为 0x0000_0000，大小分别都为 32KB。

一般来说，R5F 可以通过两种方式启动系统，第一种是通过外部 Memory 来记载中断向量表。而当 ATCM/BTCM 被使能且作为 R5F 内部 Memory 使用时，其 0 起始地址段的 TCM RAM 可以用来存放中断向量表，另一块 TCM RAM 可用来存放代码的 Entry Point，其配置方法与第 3.2 小节相同，可以在 linker 文件中指定。以 ATCM/BTCM 都使能且 *CPUn_LOCZRAMA* 为高时为例，其大致配置流程如图 12 所示。

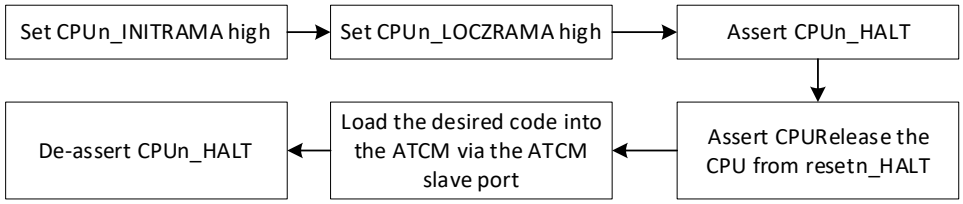

图 12 R5F Boot from TCM Process

当 ATCM/BTCM 都 Disable 时，则这两块 RAM 的内存片段不会出现在 R5F 的 Memory Map 中，即不可被 R5F 访问，但是它是可以被外部资源通过总线访问的。

3.4. Cache

上述介绍的 MSMC，TCM 等都属于特殊类型的 Cache，而在处理器内部，一般都会集成有内部 L1/L2 Cache，能够大大的提高 CPU 以及主存之间的数据交换效率。以 TDA4VM 为例，其内部的 A72，R5F，C66，C7x 都具有内部的 L1 以及 L2 Cache，如图 13 所示。

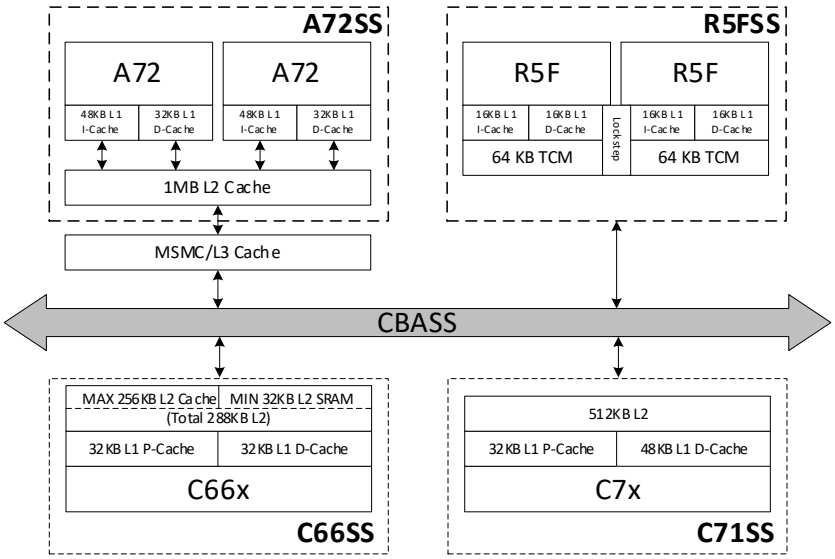

图 13 TDA4 内部处理器 Cache 分配示意图

其中 C66 以及 C7x 中的 L1 P-Cache（Program Cache）以及 L1 D-Cache（Data Cache）与 ARM Core A72 以及 R5F 中的 I-Cache（Instruction Cache）以及 D-Cache（Data Cache）类似，分别常用于缓存指令以及数据。因为在 CPU 运行过程中，指令信息一般只做读出操作，而数据信息会涉及到读写操作且变化较大；同时 P-Cache 一般采用硬件更加直接简单的直接映射缓存，而 D-Cache 一般采用更加复杂的两路组相连缓存。所以如果对外部 Cache 数据与指令的读写进行隔离处理，在物理实现上更加实际且读写效率都会更高。

值得注意的是，在 C66 处理器核心中，其 L1 以及 L2 的 Cache 是可以通过 L1/L2 Mode Bits 来进行配置的。例如 L1P/L1D/L2 Cache 可通过 L1P/L1D/L2 Mode Bits 来将总大小的内存划分为指定的 SRAM 以及 Cache 大小，详情请参照 TDA4 TRM 手册。

4. 片内外存储系统配置应用实例

对于片外 Memory 的配置管理，主要是为了支撑整个系统的启动以及运行，因为在系统启动时，其系统文件都存储在外部 Memory 存储介质中，且在系统运行时，其大部分程序都运行在主存 DDR 中。以典型的 TDA4 启动流程为例，常见外部介质在 TDA4 启动流程中对应的作用如图 14 所示。

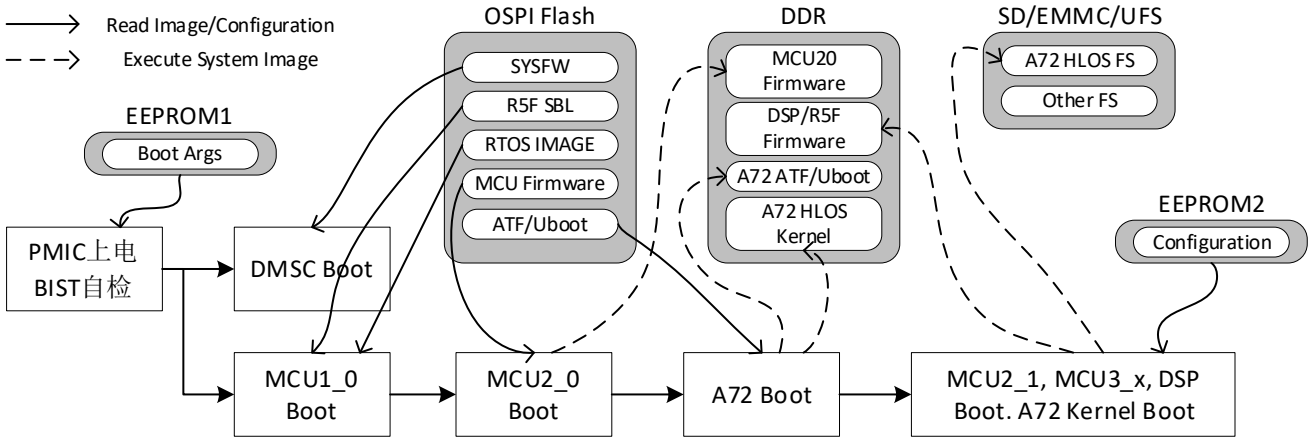

图 14 外部 Memory 在 TDA4 启动流程中的作用

可以看到在 TDA4 内部各核心的启动流程中，OSPI Flash 主要作为各处理器镜像的存储介质，TDA4 从 OSPI 中对系统镜像进行读取；DDR 作为系统运行主存，大多处理器的系统以及应用都在 DDR 中执行；而 SD/EMMC/UFS 这类 Flash 主要用于存储 A72 处理器的上层文件系统；EEPROM 则可以作用于前期 BOOT 阶段用来加载启动参数，或者系统起来之后用于加载例如以太网 IP 配置参数等等。众多外部 Memory 的协同工作才能保证 TDA4 系统的正常启动，用户也可根据自身系统大小需求以及成本控制等方面灵活配置。

对于片内 Memory 的配置管理，主要是为了提高内部处理器核心对数据的读写效率，进而提高处理器的整体处理速度，但不同应用场景下 Memory 可根据实际需求划分或配置为 SRAM/Cache，用户可参照图 15 流程并根据自身需求进行配置。

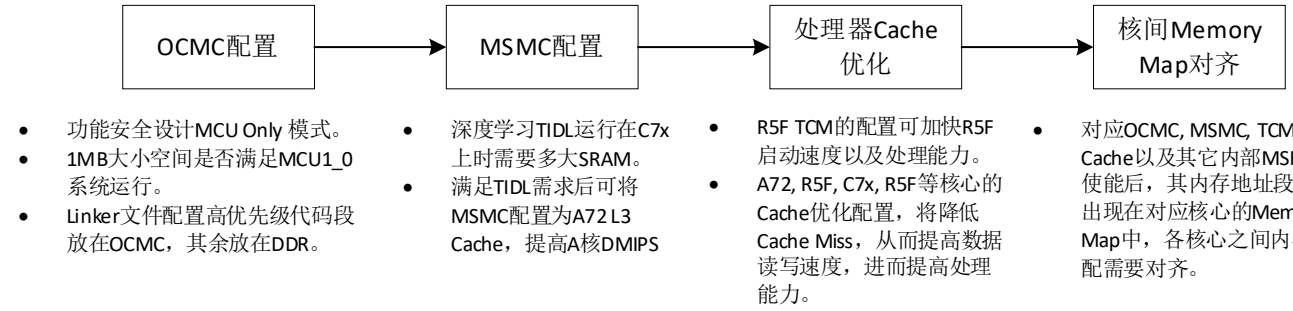

图 15 片内内存配置流程

与外部 Memory 所不同的是，同一 PN 的 TDA4 芯片其内部 Memory 大小是确定的，用户仅需要对其进行使能控制以及资源分配配置等。

5. 总结

TDA4 作为一个高效且强大的多核异构处理器，其内外具有多种 Memory 存储介质，外部存储介质可主要用来支撑系统的正常启动，以及优化启动或运行时对外部镜像或数据的读写速度；而 TDA4 内部集成的各种 Memory 可主要用来优化在系统功能运行时，其各处理器核心的处理能力以及核间的数据交换速率。两者分工不同，互相补充，能够更好的支撑一个高效系统的运行。

6. 参考文献

1. TDA4VM Jacinto™ Processors for ADAS and Autonomous Vehicles Silicon Revisions 1.0 and 1.1 datasheet (Rev. J)
2. DRA829/TDA4VM/AM752x Technical Reference Manual (Rev. B)
3. TDA4 PSDKRA User Guide.
4. TDA4 PSDKLA User Guide.

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