Application Note

使用外部 RAM 扩展 AM13x AI 功能：软件和硬件设计说明

摘要

德州仪器 (TI) 支持边缘 AI 应用，具有各种工具链和 MCU 微控制器，适用于各种终端设备。在快速发展的边缘 AI 领域，当用户尝试在 MCU 中部署相对“复杂”的边缘 AI 模型时，内存不应成为制约瓶颈。本应用手册利用 AM13x 外部外设接口 (EPI) 在外部 SDRAM 中减轻 AI 模型的负载，帮助用户在系统级别实现性能、成本和功耗之间的平衡。本应用手册介绍了相关方法和分步指南。该演示项目可在 AM13E230x MCU_SDK 中获得，网址为“*am13e230x_sdk_26_01_00_00/examples/driverlib/epi/epi_sdram_read_write*”，受 SDK 版本 26.01.00 支持，并使用 SDRAM AS4C32M16S [14] 进行了测试。

内容

1 简介.....2
1.1 TI AM13E230x 简介.....2
1.2 TI MCU 的 AI 工具链简介.....3
2 详细说明.....5
2.1 EPI 的 Sysconfig 设置.....5
2.2 用于外部 SRAM 执行的 AI 模型库部署.....6
2.3 硬件设计最佳实践.....9
3 总结.....11
4 参考资料.....12

商标

1 简介

TI 支持全面的工具链，可帮助用户在 AM13x、C2000、AM26x、MSPM0 等不同 TI MCU 系列中部署 AI 应用。在系统级别，AI 模型必须产生低延迟且精度良好的结果，或者延迟可能造成命中但保持更高的精度。这是系统架构师通常会遇到的权衡问题。

在速度至关重要的场景中，AI 模型是“轻量”模型，这些模型只需要数十 KB 的闪存和 SRAM。NPU 引擎可以加速 AI 模型过程，以实时输出结果；在模型大小至关重要的情况下，模型部署受到 MCU 片上内存的限制，甚至 CPU 足以用于模型处理。由于这个限制，用户必须选择处理器级器件，但由于成本和功耗较高，这使得实际应用不可行。

TI AM13x MCU 通过使用外部外设接口 (EPI) 解决了这个问题，EPI 接口可连接外部 SDRAM、ASRAM、SRAM 或者闪存内存，从而减轻外部 SDRAM 中的 AI 模型负载。此功能为 AI 模型执行提供了最大 64MB 的外部 SDRAM；采用此结构时，系统成本和功耗也显著降低。

除了优点外，这种设计也有两个缺点。第一个缺点是尺寸。EPI 连接需要 32 个引脚。支持 32 引脚 EPI 信号的 AM13E230x 的最低封装是 LQFP100。该封装的外形为 15.5mm× 15.5mm，因此还必须考虑 SRAM 尺寸。因此，对于需要小尺寸的应用，这种设计不切实际。

第二个缺点是速度。AM13E230x 片上 NPU 不支持访问外部 SRAM，因此转移到外部 SDRAM 上的模型必须由在 200MHz 上运行的 ARM® Cortex®-M33 执行。指令访问速度也受 EPI 总线的限制，这使得此设计对于高速应用（例如需要实时 AI 输出且延迟极低的应用）不切实际。

本手册中讨论的技术不限于 EdgeAI 应用，但可以扩展类似的方法，以在 AM13x MCU 上存储和执行各种应用。

1.1 TI AM13E230x 简介

AM13E230x 微控制器 (MCU) 属于 AM13x 高度集成的低功耗 32 位 MCU 系列，该 MCU 系列基于 Arm Cortex-M33 32 位 CPU，工作频率最高可达 200MHz。这些实时控制优化型 MCU 提供高性能模拟，控制和数字外设集成。MCU 提供具有内置纠错码 (ECC) 且高达 512KB 的嵌入式闪存程序内存（2 个 256KB 存储体）以及具有硬件奇偶校验且高达 128KB 的 SRAM。也提供更小的内存配置型号。

通过一个 MCAN 和多达 6 个 UNICOMM 外设支持增强型通信接口，从而支持 UART/LIN，I2C/SMBUS 和 SPI 的组合。为了连接外部器件或内存，高速 EPI 可以连接到 SDRAM 或异步 RAM 器件，例如 FPGA 或 ASIC。

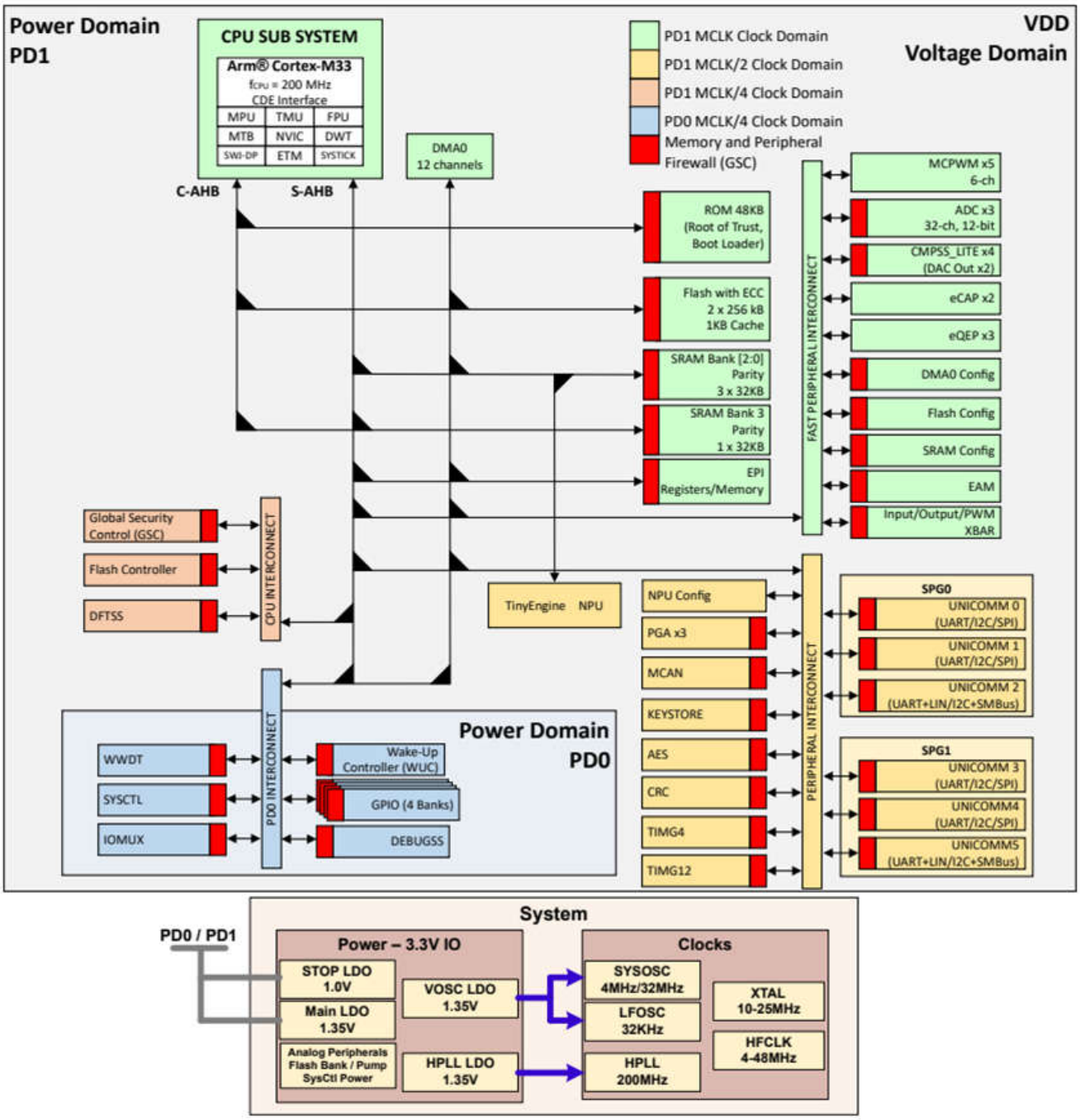

图 1-1. AM13E230x 功能框图

1.2 TI MCU 的 AI 工具链简介

TI 为适用于 MCU 的 EdgeAI 提供强大的工具链支持。具有不同经验水平的客户可以利用 TI 工具链在 MCU 上构建和部署 AI 模型。TI 工具链负责管理整个流程，从数据采集到数据标记，再到训练 AI 模型以及编译和部署，不一而足。

不同工具的用户指南附在参考资料 部分中：

- Git Hub [1] 中的 tinyml-lensorlab 版本和 TI.com 官方网站 [2] 中的边缘 AI Studio GUI 版本包含 TI 验证的 AI 模型和整个工具链，用于 AI 模型编译。用户可以直接将这些模型集成到应用中。在 Git Hub 版本中，用户可以直接读取 .py 文件，以了解工具链的工作原理。这些模型均针对速度而开发，无需外部 SRAM，用户可以使用 TI 的 launchpad 直接训练和验证模型。这些模型的 MCU 部署示例包含在各个 MCU 器件 SDK 中。

- 如果用户想与 PyTorch 等常用的 AI 模型开发人员一起执行一些 DIY AI 模型，适用于 MCU 的 TI 神经网络编译器 [3] 可以支持将 .onnx 文件编译到 MCU .a 库文件中。在这种情况下，如果必须将 MCU 片上 NPU 用作加速器，则必须使用 NPU 支持的操作 [4] 来构建该模型，并且必须将 TI-NPU QAT [5] 应用于 .onnx 文件，以使模型适合 NPU。如果需要在 CPU 内核中执行该模型，则 DQD 足以满足量化需求。用户必须自行仔细验证模型。编译器还输出 lib.c 文件，供用户调试每层的输入和输出。这些具有不同 AI 训练结构的高度定制模型可以轻松占用大量内存。一种情况是使用变压器结构的 AI 模型，在这种情况下，用户需要使用外部 SDRAM 进行内存扩展。

在这两种方法中，需要嵌入到 MCU 工程中的最终生成的 AI 模型是一个“.a”文件（代码和权重库文件），和一个“.h”头文件来提供库的接口。

2 详细说明

本节指导用户逐步在 MCU 片上闪存中加载 AI 模型文件，并将模型执行负载转移到外部 SDRAM。

2.1 EPI 的 Sysconfig 设置

本节旨在使用 sysconfig 工具生成 EPI 外设的初始化代码。有关 EPI IP 的详细规格，请参阅 AM13E230 TRM [6] EPI 部分。生成的代码可在文件 <ti_sdk_dl_config.c> 中找到，并在 main 函数的 SYSCFG_DL_init 函数中调用。要详细了解 EPI，请参阅参考资部分料中随附的 AM13x EPI 学院章节。

需要配置 2 个关键规格：刷新计数和 EPI 时钟分频器。

2.1.1 刷新计数

刷新计数基于外部时钟速度、每个组的行数及刷新周期。RFSH 字段表示需要自动刷新之前还需要多少个外部时钟周期。正常公式为：

$$\text{RFSH} \leq (\text{t\_Refresh\_us} \ / \ \text{number\_rows}) \ / \ \text{ext\_clock\_period\_us} \quad (1)$$

刷新周期通常为 64ms，即 64000us。行数通常为 4096 或 8192。ext_clock_period 是以 us 表示的值，可通过将 1000 除以时钟速度 (MHz) 来推导得出。因此，50MHz 为 1000 / 50 = 20ns，或 0.02us。如果系统时钟在 50MHz 下运行且 EPIBAUD 寄存器值为 0 的情况下，则典型的 SDRAM 为每个存储体 4096 行：

$$\text{RFSH} = (64000 \ / \ 4096) \ / \ 0.02 = \ 15.625\text{us} \ / \ 0.02\text{us} = 781.25 \quad (2)$$

RFSH 字段中的默认值为 750 十进制，即 0x2EE，以提供安全裕度并提供每次刷新 15us。请注意，此数字必须始终小于或等于方程式 2 要求的值。如果使用大于所允许的数字，SDRAM 刷新频率不足，数据将丢失。

2.1.2 EPI 时钟分频器

EPI 控制器 SDRAM 接口可在高达 50MHz 的频率下运行。EPIBAUD 寄存器中的 COUNT0 字段用于配置 EPI 时钟的速度：当主时钟 (MCLK) 速度高达 50MHz 时，COUNT0 字段可以为 0x0000，SDRAM 接口能够以与 MCLK 相同的速度运行。但是，如果 MCLK 以较高的速度运行，则总线接口只能以半速运行，并且 COUNT0 字段必须至少配置为 0x0001，配置请参见图 2-1。

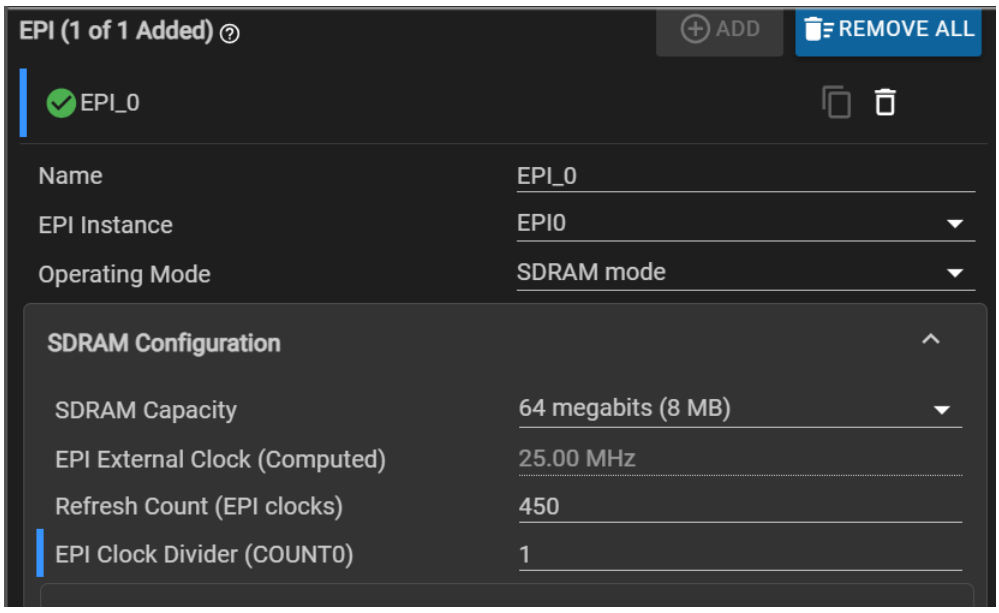

图 2-1. EPI 配置示例 1

EPI 可以为外部 RAM 数据和指令分配地址，由于未连接外设，因此无需基地址配置。有关详细信息，请参阅图 2-2。在 SDRAM 模式下无需配置中断，H/W 引脚多路复用器由 sysconfig 工具自动分配。

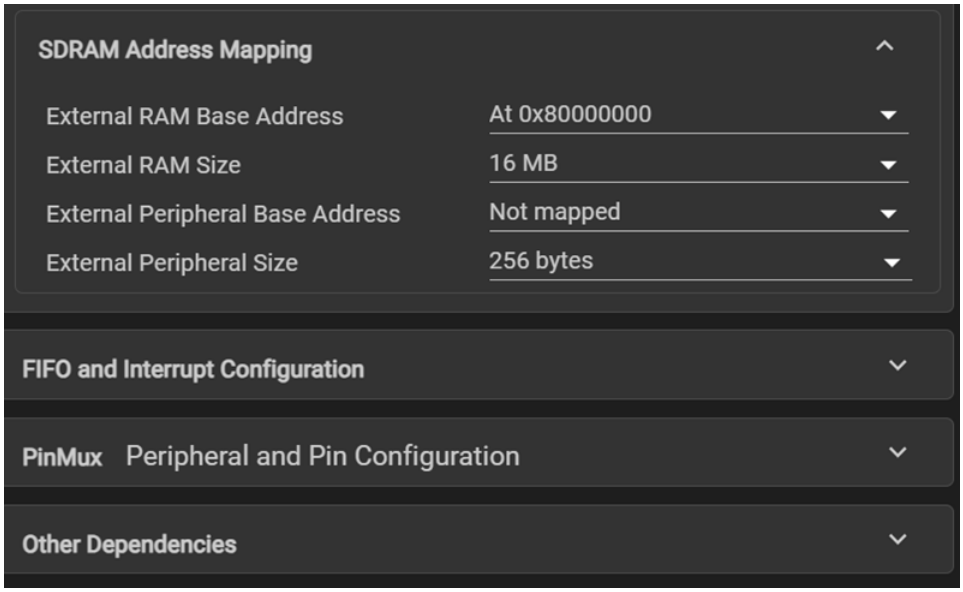

图 2-2. EPI 配置示例 2

2.2 用于外部 SRAM 执行的 AI 模型库部署

AI 应用中的注意力矩阵、权重矩阵等占据了 MCU 中的大部分 SRAM。矩阵只有在 AI 模型开始运行时才会运行，应用结束后才会停止运行。借助此功能，AI 模型的闪存需求远小于 RAM 需求。通过将 AI 模型存储在 MCU 片上闪存中并使用外部 SDRAM 执行 AI 模型，可以满足 AI 模型内存要求。本节将指导用户使用编译器关键字来实现此内存分配。

如简介 中所述，AI 模型工程中需要集成的文件是一个“a”库，其中包含两个段：一个 .text 段用于 AI 函数，一个 .rodata 段用于表示权重和表。用户可以使用链接器段将这些段作为加载映像存储在闪存中，并在启动时复制到 SDRAM。所有功能地址均从 SDRAM 地址空间分配 (`0x80000000+`)。表 2-1 展示了内存分配：

表 2-1. 在闪存中加载 AI 模型并在外部 SDRAM 中运行时的内存分配

闪存（加载映像已存储）		SDRAM（代码从此处执行）
0x000xxxxx	-> copyCode() ->	0x80000000（由 syscfg 设置）
.text（AI 函数）		.text（PC 从此处获取）
.rodata（权重和表）		.rodata

以下各节介绍了实现目标内存分配的分步设置。该示例使用“customer_ai_model.a”作为“.a”文件名，用户可以相应地更改该文件名。基本的 AI 嵌入式工程是指“..\am13e230x_sdk_<version>\examples\ai\generic_timeseries_classification”[11]。

有关编译器符号详细信息和 cmd 文件详细信息，请参阅参考资料 部分中的 <TI ARM CLANG 编译器工具用户指南> [8] 和 <TI 连接器命令文件入门> [9]。该演示工程可在 AM13E230x MCU_SDK 中找到，网址为“*am13e230x_sdk_26_01_00_00/examples/driverlib/epi/epi_sdram_read_write*”。

2.2.1 第 1 步：将 AI “.a”文件添加到 CCS 工程中

将 `customer_ai_model.a` 复制到工程文件夹中，以替换“libCMSISDSP_m33_ti_arm_clang.a”文件。

在 CCS 中： **Project Properties -> Build -> Arm Linker -> File Search Path**

- 添加库文件名： 'customer_ai_model.a'
- 将包含“.a”文件的文件夹路径添加到库搜索路径

2.2.2 第 2 步：链接器脚本更改 (`linker_m33_ti_arm_clang.cmd`)

将内存段“EPI_CS2_EXEC_1”添加到工程中。调整“长度”以适应客户的 AI 模型大小。下面提供了 CMD 文件的更改：

```
MEMORY
{
    RAM_S        : ORIGIN = 0x20000000, LENGTH = 0x18000   // 96KB internal SRAM
    RAM_C        : ORIGIN = 0x00C18000, LENGTH = 0x8000    // 32KB code SRAM
    FLASH_BANK0  : ORIGIN = 0x00000000, LENGTH = 0x40000   // 256KB Flash
    FLASH_BANK1  : ORIGIN = 0x00040000, LENGTH = 0x40000   // 256KB Flash

    // External SDRAM through EPI – AI model code + weights execute from here
    // Customer with 300KB AI model: increase to 0x0080000 (512KB) or 0x0100000 (1MB)
    EPI_CS2_EXEC_1(RWX) : origin = 0x80000000, length = 0x0010000  // 64KB (demo size)
}
```

添加 .CS1_EXECUTE 段并将 `libCMSISDSP_m33_ti_arm_clang.a` 替换为“customer_ai_model.a”文件名。下面提供了 CMD 文件的更改：

```
.CS1_EXECUTE : LOAD = FLASH_BANK0, palign(8),
               RUN  = EPI_CS2_EXEC_1,
               LOAD_START(emif1_cs2_sec1_loadstart),
               LOAD_SIZE(emif1_cs2_sec1_loadsize),
               RUN_START(emif1_cs2_sec1_runstart),
               RUN_SIZE(emif1_cs2_sec1_runsize)
{
    // Replace with the customer's AI .a filename.
    // The <*> wildcard grabs ALL objects in the archive automatically.
    customer_ai_model.a<*>(.text .text.* .rodata .rodata*)
}
```

每个关键字的含义在表 2-2 中进行了说明：

表 2-2. 连接命令文件修改中的关键字含义

关键字	含义
LOAD = FLASH_BANK0`	链接时存储在闪存中的 AI 模型二进制映像
RUN = EPI_CS2_EXEC_1	从 SDRAM 分配的所有 AI 功能地址 (`0x80000000+`)
LOAD_START (emif1_cs2_sec1_loadstart)	链接器符号 = 映像开始的闪存地址
LOAD_SIZE (emif1_cs2_sec1_loadsize)	链接器符号 = 要复制的字节数
RUN_START (emif1_cs2_sec1_runstart)	链接器符号 = SDRAM 目标 (`0x80000000`)
<*> (.text .text.* .rodata .rodata*)	通配符：捕获存档中的所有代码和权重

2.2.3 'main.c' 变更

本节介绍了用于部署工程的 main.c 文件变更。

2.2.3.1 包含头文件

必须在工程中包含以下头文件：

```
#include "device.h"
#include "log.h"
#include "stdint.h"
#include "dl_epi.h"
#include "dl_epi.c"           /* EPI driver – included directly              */
#include "basic_math_functions.h"  /* CMSIS DSP – defines float32_t           */
#include "ti_sdk_dl_config.h"
```

2.2.3.2 声明由链接器生成的符号

这四个符号由链接器根据段属性自动生成。在 C 语言中，'&symbol' 给出了链接器分配的值。

```
extern unsigned int emif1_cs2_sec1_loadstart;  // Flash address of load image start
extern unsigned int emif1_cs2_sec1_loadsize;   // Size of load image in bytes
extern unsigned int emif1_cs2_sec1_runstart;   // SDRAM destination (0x80000000)
extern unsigned int emif1_cs2_sec1_runsize;    // Size of section at run time
```

2.2.3.3 memcpy 函数

该函数使用 TI 链接器符号约定将段从闪存复制到 SDRAM。'&sdram_code_loadstart' 提供链接器分配的值。

```
memcpy(&sdram_code_runstart,&sdram_code_loadstart,(size_t)&sdram_code_loadsize);
```

2.2.3.4 将条目函数放在 SDRAM 部分中

在演示项目中，“ExecuteFromSDRAM()”是从 SDRAM 执行并调用 "arm_dot_prod_f32()" 的函数。该函数通过“__attribute__”放置在“.CS1_EXECUTE”中。

```
/* This function executes from SDRAM. memcpy() must be called first. */
__attribute__((section(".CS1_EXECUTE"), __noinline__))
void ExecuteFromSDRAM(void)
{
    LOG("Executing from SDRAM!\r\n");

    /* Call arm_dot_prod_f32 – this function is also in .sdram_code (SDRAM).
     * Input data lives in RAM_S and is accessed through the normal data bus.  */
    arm_dot_prod_f32(App_DotProd_SrcA,
                     App_DotProd_SrcB,
                     APP_DOT_PROD_SIZE,
                     &App_DotProd_Result);
}
```

2.2.3.5 Main() 引导顺序

这是正确引导 MCU 并开始使用外部 SDRAM 运行 AI 模型的主要功能序列。

```
int main(void)
{
    /* Step 1: Hardware init – Device_Init() sets up clocks and cache.
     * SYSCFG_DL_init() configures the EPI peripheral, making SDRAM
     * accessible at 0x80000000. Both must run before any SDRAM access. */
    Device_Init();
    SYSCFG_DL_init();

    /* Step 2: Copy .CS1_EXECUTE section from Flash to SDRAM.
     * Transfers all AI model functions and weights from their Flash
     * load addresses to their SDRAM run addresses (0x80000000+).
     * After this call, all AI functions execute from SDRAM. */
    memcpy(&sdram_code_runstart,&sdram_code_loadstart,(size_t)&sdram_code_loadsize);


    /* Step 3: Call AI inference – PC moves to 0x80000000+ region.
     * In the demo project this is executeFromCs0().
     * For customer: replace with the AI model inference entry point. */
    ExecuteFromSDRAM();/* customer replaces this with their AI inference call */


    /*Verify dot product result. */
    LOG_ASSERT((App_DotProd_Result == APP_DOT_PROD_EXPECTED),
               "arm_dot_prod_f32 result mismatch: expected 20.0\r\n");
    LOG("PASS: arm_dot_prod_f32 = %.1f – executed from SDRAM\r\n",
        App_DotProd_Result);

    LOG("Example PASS\r\n");

    return 0;
}
```

2.2.4 第 4 步：CSS 项目验证

为了在应用于客户的 AI `.a` 文件之前验证该方法，CMSIS DSP 库 (`libCMSISDSP_m33_ti_arm_clang.a`) 作为替代。此库包含 `arm_dot_prod_f32()`，这是一个乘法累加函数，是存档中 `BasicMathFunctions.c.obj` 的一部分。

2.3 硬件设计最佳实践

本节介绍了 EPI H/W 设计的通用指南，并讲解如何使用 Altium 更好地满足要求。用户可以参阅 TM4C EPI 参考设计文档 [12] 以获取工程文件。

AM13E230 硬件设计指南 [13] 中列出了有关 MCU H/W 设计的完整指南。应遵循以下 EPI 指南：

- 将目标器件（内存 IC 或 FPGA）靠近 MCU 放置，以保持迹线较短
- 使用 50Ω 单端迹线控制阻抗
- 通过在同一层路由信号组来更大限度地减少反射和串扰，并更大限度减少过孔/层切换
- 匹配以下迹线长度以减少时序偏移：
  - 数据总线：匹配范围在 ±25mil 以内
  - 地址和命令信号：匹配度在 ±25mil 以内
  - 控制信号：匹配范围为 ±25mil 至 ±50mil
  - 时钟信号：与最长数据迹线的匹配范围为 ±5mil 至 ±10mil
- 并行路由地址和命令信号
- 避免残桩并使布线尽可能笔直
- 考虑将数据线路和时钟线路的串联端接电阻器（22Ω 至 33Ω）靠近发送器放置

请注意，建议从 EPI 引脚到地址/数据之间采用单个布线，然后再到其他引脚，以尽量减少信号反射，如图 2-3 所示。

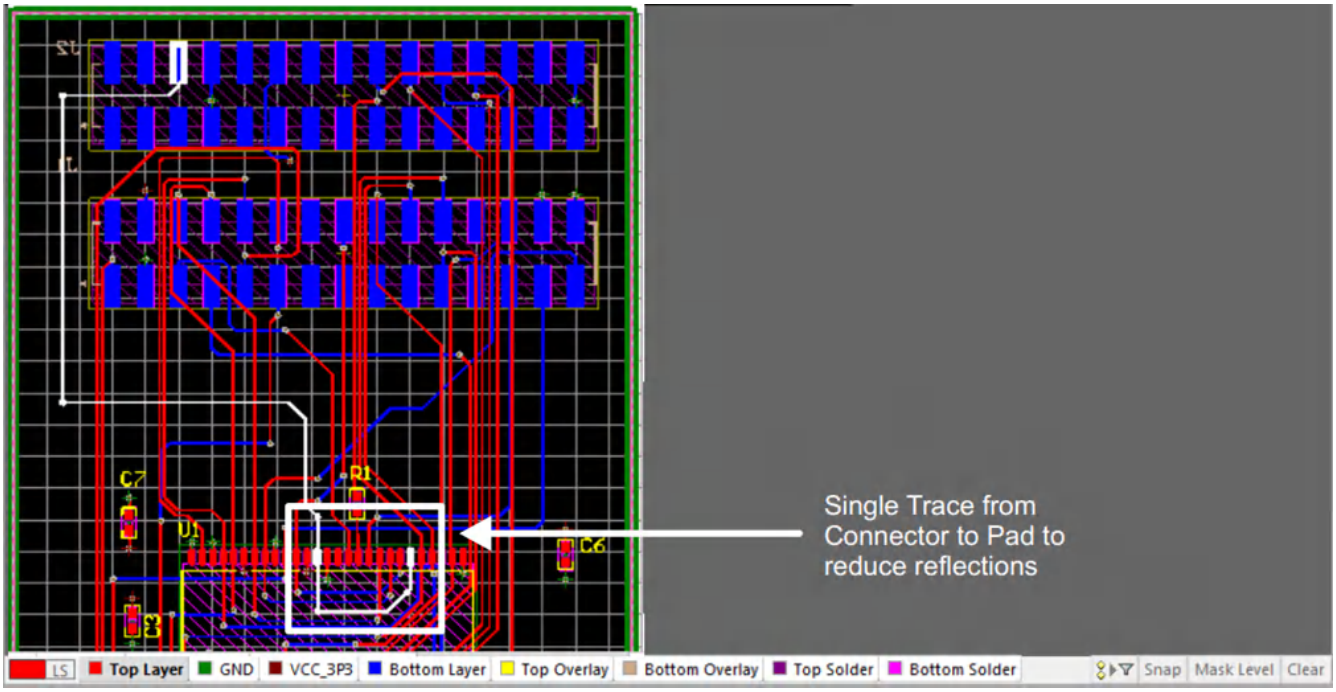

图 2-3. 从 EPI 引脚到地址/数据的单个跟踪

使用 Altium 更轻松地实现 H/W 设计：

- 地址和数据布线必须使用两个不同的网名称，以便用户能够利用长度调优来匹配布线长度。如果两者之间的网络名称当前相同（在电阻器之后），则很难在 Altium 中设置。建议在两者之间使用不同的迹线名称，并且当迹线在 AM13E23019 的 EPI 引脚处相遇时，用户可以使用电阻器将其短接在一起，也可以使用网带元件在引脚处将网络短接在一起。

用户可以在 Altium 中按照以下步骤使用布线匹配：

- 从顶部菜单转到“设计”>“规则”以打开 *PCB 规则和约束*编辑器。
- 展开*高速*类别并选择*匹配的长度*。
- 右键单击并选择*新建规则*以命名和配置规则。
- 在*对象匹配范围*下，选择要匹配的特定网、xSignal 或网类别。
- 在*约束*部分下，设置容差限制。这定义了组中最短和最长迹线之间允许的最大差值。
- 然后，用户可以在此容差组中将所有网组合在一起，并将容差限设置为 25mil。

此外，建议将电阻器放置在相应的 SDRAM 引脚附近，并使所有引脚与相应电阻器之间的距离保持一致，因此用户不需要对该部分布线进行布线匹配。

3 总结

本应用手册介绍了一种将 AI 模型分流到具有 AM13E230 EPI 的外部 SRAM 的方法。这种方法解决了在 MCU 中部署 AI 模型时内存短缺的问题，帮助用户在系统级别实现性能、成本和功耗之间的平衡。本应用手册介绍了相关方法和分步指导，而 AM13E230x MCU_SDK 中提供了演示项目，网址为“*am13e230x_sdk_26_01_00_00/examples/driverlib/epi/epi_sdram_read_write*”。

4 参考资料

1. 德州仪器 (TI)，Tiny ML Tensorlab 用户指南，用户指南。
2. 德州仪器 (TI)，EDGE-AI-STUDIO。
3. 德州仪器 (TI)，TI MCU 神经网络编译器用户指南 - 2.1.2 LTS，用户指南。
4. 德州仪器 (TI)，NPU 支持的层配置。
5. 德州仪器 (TI)，TI-NPU QAT。
6. 德州仪器 (TI)，AM13E230x 微控制器，技术参考手册。
7. Varahmihir，Arpit。外设接口 (EPI)。
8. 德州仪器 (TI)，SECTIONS 指令。
9. 德州仪器 (TI)，TI 连接器命令文件入门。
10. 德州仪器 (TI)，AM13E23019：EPI 代码执行演示。
11. 德州仪器 (TI)，AM13E2X-SDK。
12. 德州仪器 (TI)，高性能微控制器上的 SDRAM 接口设计指南，设计指南。
13. 德州仪器 (TI)，AM13E230x 硬件设计指南，应用手册。
14. Alliance Memory，AS4C32M16SB 数据表，数据表。

重要通知和免责声明

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、与某特定用途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安全、安保法规或其他要求。

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 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 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。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月