恩智浦半导体
应用笔记

文档编号：AN12972
第 0 版，2021 年 2 月

针对性能要求高的应用，优化 S32K1xx eDMA

作者：恩智浦半导体

1. 介绍

直接存储器访问（DMA）是一种功能，用于在主存储器和外围设备之间进行数据传输，而无需通过 CPU。通过与主机处理器并行工作，可以实现更高的传输速度。

eDMA 控制器可以执行复杂的数据传输，但有时其传输速度会因同时访问内部总线而受到限制。

另一个限制传输速度的因素是所连接外设的响应时间（其引擎的工作频率），这是找到外设时钟、eDMA 和 CPU 时钟频率的适当组合的关键。

本应用笔记旨在为读者提供线索和良好实践，以提高其应用的性能。更多详细信息，参见 S32K1xx-RM。

目录

注意

值得注意的是，eDMA 的性能会因应用的相关因素而有所不同，并且在某些用例中这些因素是无法改变的。本文档提供对这些因素的分析，并对如何管理某些情况做出指导。

2. 在 S32K14x 上集成 eDMA 模块

下图显示了 S32K14x 产品集成，后面的列表显示了一些可能影响数据传输性能的功能/组件。

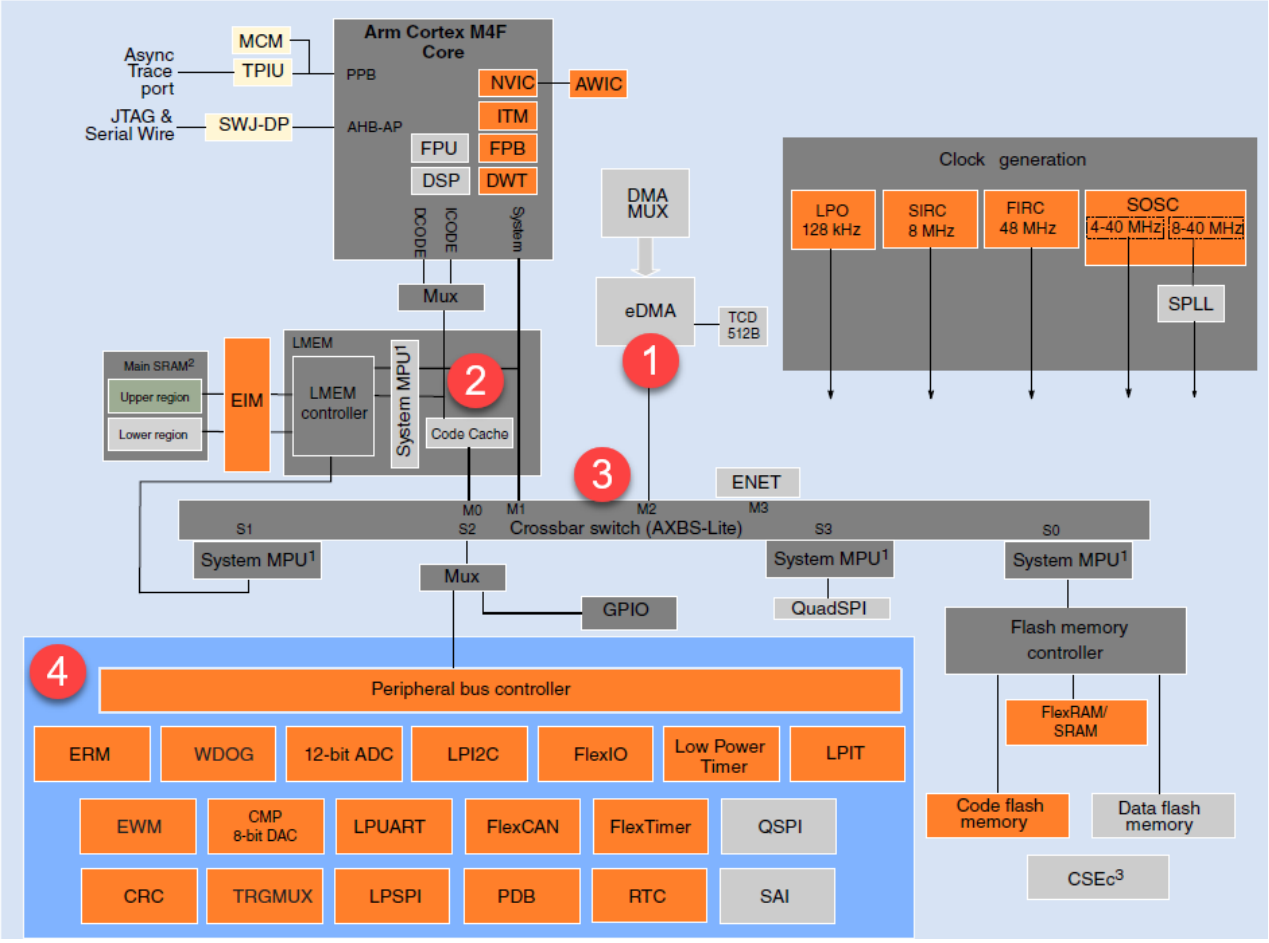

图 1. S32K14x 框图

1. eDMA 选项：通道数量、优先级配置、带宽控制、传输数据宽度、启用的功能等。下一节将讨论这些选项。
2. 代码高速缓存：该芯片包括一个 4KB 代码高速缓存，以最大程度地减少存储器访问延迟对性能的影响。LMEM 控制器为处理器提供紧耦合的处理器本地存储器，以及到所有从存储器空间的总线路径。

3. 交叉开关（Crossbar switch）：当多个总线主节点访问同一从节点时，AXBS 提供总线主节点之间的仲裁。当一个 eDMA 尝试访问 SRAM（或任何从节点），而其他主节点（即 CPU）也尝试访问该从节点时，就会出现一个瓶颈。根据用例情况，可能需要使用 MCM_CPCR[CBRR]寄存器添加带宽控制。MCM_CPCR[CBRR]可以将交叉仲裁设置为轮询或固定优先级。如果应用需要尽可能地控制总线，则需要设置固定优先级（其他主节点有长期无法访问从节点的风险）。
4. 外设配置：根据需要实现的应用，可以对与 eDMA 相关的外设进行一些配置，以提高性能。在下一章中，我们将讨论一些用例。

下图显示了 eDMA 内部构成，随后列出了可能影响 eDMA 性能的配置。

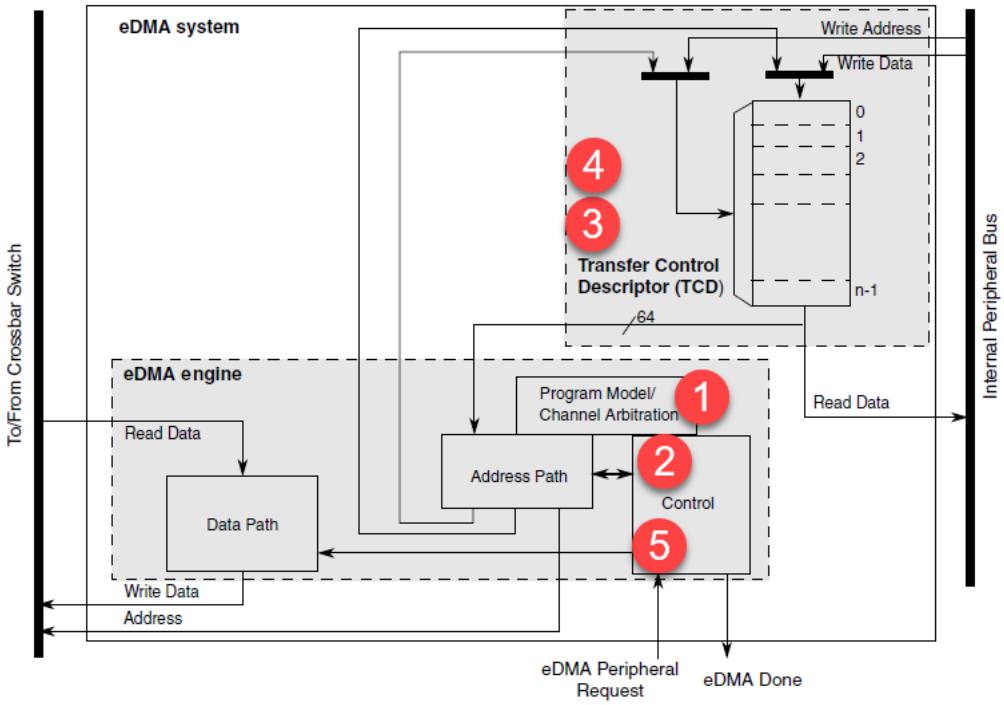

图 2. eDMA 框图

1. 优先级配置：eDMA 能够在通道之间选择固定优先级或轮询，根据应用的不同，可能需要一个通道比其他通道具有更高的优先级。一种较好的方法是设置固定优先级并将较高的优先级分配给关键通道，但在某些情况下不能这样做，而是要根据具体的应用进行配置。
2. 通道数量：如果只有一个 eDMA 引擎，那么同一时间只能服务一个通道。当出现更多的活动通道和持续不断的 eDMA 请求，就会出现这样的情况：当一个通道正在被服务时，对另一个通道的服务很可能会有延迟。
3. 带宽控制：在某些应用（具有大的传输数据量）中，需要避免交叉开关中其他总线主节点长期无法访问从节点，为此，eDMA 可以在每次 R/W 操作中停止自己的引擎。此选项允许其他主控制器（如 CPU）控制从端口并能够与 eDMA 一起工作。此功能不会提高性能，但在某些实现中需要加以考虑。
4. 传输数据宽度：eDMA 支持可编程的源、目标和传输数据宽度，对于源和目标宽度相等的数据传输，eDMA 引擎执行一系列源读取/目标写入的操作。对于宽度不相等的描述符，需要多次访问较小宽度的数据来引用较大宽度的数据。例如，源宽度引用 8 位数据，而目标宽度是 32 位数据，则需要执行 4 次读取操作，然后执行一次 32 位写入操作。
5. 启用的 eDMA 功能：eDMA 的一些功能（如分散聚集、链接通道或次循环偏移）可以简化特定应用的实现。其中一些功能具有增加传输时间的缺点，在这些情况下，软件设计人员必须分析以牺牲 eDMA 性能为代价获得的潜在好处。

2.1. 分散聚集功能

分散聚集（SGA）功能允许 eDMA 通道在主循环完成后载入不同的传输控制描述符（TCD）。这个功能的基本理念是，当一个通道完成其主循环后，该通道将重新载入保存在本地存储器中的新 TCD，所有这些都无需 CPU 干预。

此功能允许用户为一个通道定义不同的 TCD，但它会导致 eDMA 传输延迟，因为要在主循环结束后载入新的 TCD，eDMA 引擎需要从保存 TCD 的存储器（闪存或 RAM）中取消引用 TCD 的指针地址，因此，eDMA 引擎需要经过一些总线才能到达它，这意味着添加到该过程中的时钟周期可能会根据总线可用性而增加。

下图显示了 eDMA 引擎必须经过的路径，以便重新载入 TCD 配置（黄色），以及由于总线访问（红色）而可能出现的瓶颈（假设 TCD 保存在 RAM 中）。

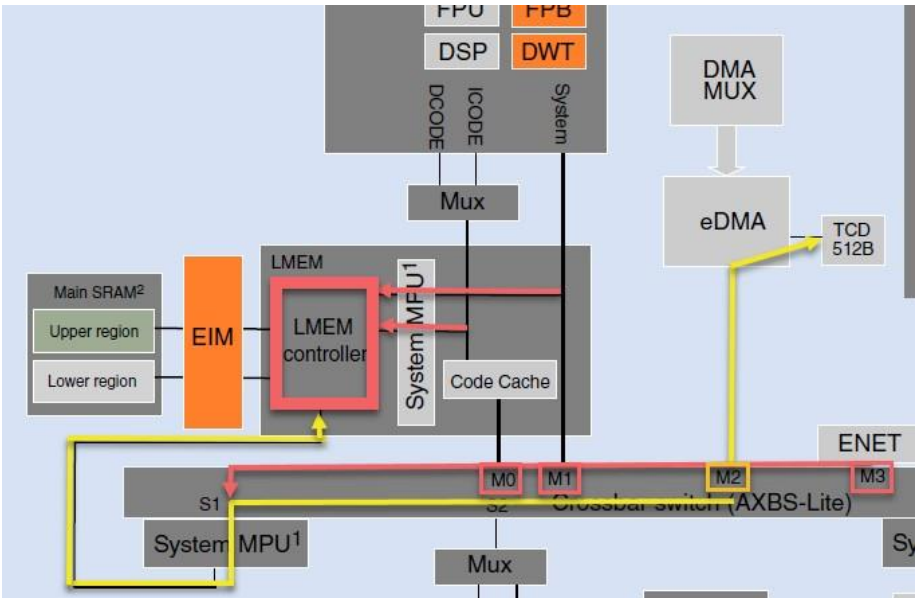

图 3. 重新载入 TCD 时，eDMA 到 RAM 存储器的路径

2.2. 通道与通道之间的链接

链接通道功能允许在一个 eDMA 通道完成一次传输（主循环或次循环）后，触发另一个 eDMA 通道。该功能允许“连接”或“链接”通道，因此当一次传输完成后，可以从另一个通道开始其他传输配置。该功能通常用于当一次传输取决于另一次传输的作业时，或当需要特定的传输顺序时。

就存储器和时间而言，此功能是比分散聚集（SGA）功能更好的选择。与分散聚集功能相反，链接通道不使用额外的存储器空间（用于 TCD 配置），因为可以直接在 eDMA 通道的寄存器中配置通道。现在，由于 eDMA 引擎不必通过任何外部总线（如交叉总线）来重新载入 TCD，因此链接通道不会增加该过程的时间（仅增加 eDMA 通道的正常启动时间）。

该功能受可用通道数量的限制，这代表了 TCD 配置的数量限制。此外，次要链接功能与 NBYTES 字段共享寄存器空间存储器，因此，如果启用次要链接通道功能，则传输或次循环计数将限制为 512 次。

在下图中，可以看到控制子模块如何执行链接过程，而无需转到交叉总线和内部总线。

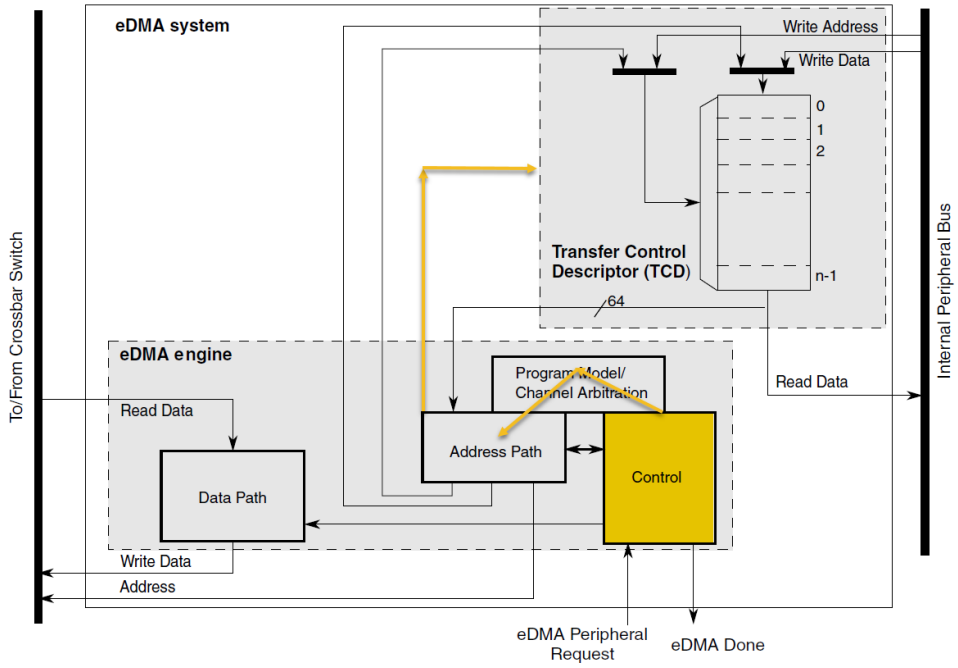

图 4. eDMA 模块内部的链接过程

2.3. eDMA 数据传输过程

任何 eDMA 数据传输过程都可以分为 3 个阶段。

表 1. eDMA 传输过程

激活通道	eDMA 请求（外设或软件）在内部登记，然后通过 eDMA 引擎路由。
	执行通道仲裁（固定优先级或轮询）。
	激活的通道号将转换为所需的访问地址。
	访问 TCD 存储器，从本地存储器读取所需的描述符并将其载入 eDMA 引擎中。
读取/写入	源读取被启动，获取的数据临时存储在数据路径块中。数据将一直存在，直到在目标写入期间该数据被加载到内部总线上。
	这个过程会一直持续到次要字节计数传输完成。
TCD 重新载入	地址路径逻辑对 TCD 中的某些字段执行所需的更新。
	如果主迭代计数已尽，则执行其他操作，例如最终地址调整和 BITER 重新载入。
	此时还会发生可选的 IRQ 置位，或从存储器中获取新的 TCD（使用分散/聚集）。

下表总结了前面的过程，并显示了假设交叉总线空闲时的理论传输时间。请注意，这些数字代表理想的情况。

表 2. eDMA 理想传输时序

通道启动	传输	eDMA 分散/聚集干预
启动用 7 个周期	读取用 3 个周期 写入用 4 个周期	用 11 个周期

3. 用例

正如本应用笔记开头所述，传输时间和性能因应用而异，本章提出了一些用例，以强调我们到目前为止所讲述的要点。

3.1. eDMA 用于串行通信

3.1.1. LPUART 和 FLEXIO

考虑这样一个场景，我们启用两个串行通信，一个实例使用 LPUART，另一个实例使用 FlexIO 模拟 UART，两者均为全双工模式。MCU 的内核运行频率为 80 MHz，总线为 40 MHz，对于此实现，两种通信中的波特率均为 10 Mbps。

为了提高 MCU 的性能，我们将采用 eDMA，以避免使用 CPU 读取和写入数据到外设缓冲区。通常，每个串行通信需要两个 eDMA 描述符，一个用于 TX（内部 RAM 到发送缓冲区），另一个用于 RX（接收缓冲区到内部 RAM）。但在本用例中，我们将试着修改内部 Rx 或 Tx 数据的地址。

如下图所示，红色方块代表内部 RAM 缓冲区，黄色箭头代表 eDMA 将执行的 4 个传输。eDMA 模块可以从缓冲区 1 或缓冲区 2 中获取数据，也可以将数据发送到缓冲区 1 或缓冲区 2。

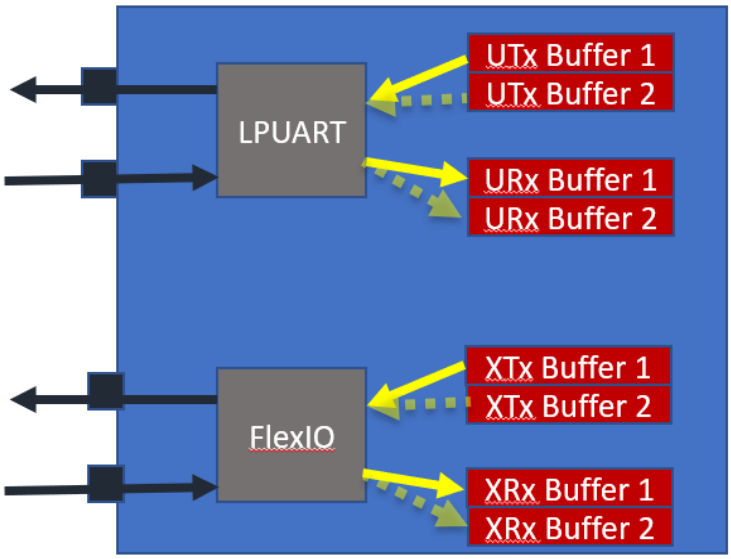

图 5. LPUART 和 FlexIO 通信

为了实现这一点，我们需要使用 4 个 eDMA 通道（2 个用于 Tx，2 个用于 Rx），通过分散聚集，我们可以更改 RAM 缓冲区的位置。

有 3 个影响此实现的要点需要考虑：通道数量、波特率和启用分散聚集。

由于我们需要 2 个全双工通信，因此我们将使用 4 个通道，这对 eDMA 来说是一个很大的工作负载，因为在 10 Mbps 的情况下，我们有 1 MBps（8 个数据位 + 起始位 + 停止位 = 每次 UART 传输 10 位），这只给了我们 1μS 的“时隙请求”。时隙请求是指对同一通道的两次请求之间的时间间隔。

这意味着为了避免 Tx 线路中的延迟（欠载）或 Rx 线路中可能的信息丢失（溢出），eDMA 应该能够在不到 1μS 的时间内（在来自同一通道的另一个请求到达之前）接收 4 个 eDMA 请求、处理它们和传输数据。

此外，我们还需要考虑当 TCD 被重新载入时，分散聚集功能将在 eDMA 过程中造成延迟，从而使我们对 RAM 总线的可用性产生依赖。这里的目标是优化 eDMA 传输，以便为 4 个 eDMA 通道提供足够的时间。

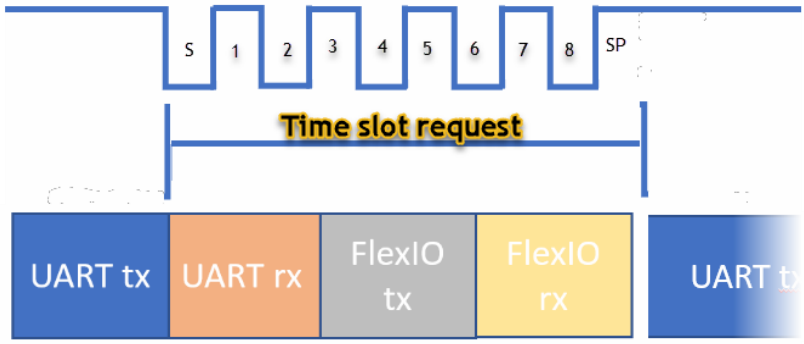

图 6. 时隙请求

通常，Tx 线路中的延迟在串行通信中不是严重的问题（例如，它们可以与超时关联，作为错误检测），但另一方面，Rx 线路中的数据丢失是必须要避免的关键问题。S32K1 的 LPUART 模块中的一个选项是 Rx 中的 FIFO 缓冲区，启用它可以在 Rx 缓冲区未被读取的情况下增加存储接收数据的空间，从而最大限度地降低溢出和数据丢失的风险。由于 eDMA 请求是由水印值（Watermark value）触发的，我们可以将此值保留为 0，这样 eDMA 请求仍然会被触发，直到 Rx FIFO 缓冲区为空。

结果

下一个示波器显示该实现的传输结果。把 UART Tx 连接到 Rx，把 FlexIO Tx 连接到 Rx。

UART Tx eDMA 通道从“UTx 缓冲区 1”执行 4 次传输，然后使用 SGA 功能重新载入 TCD。在此之后，eDMA 从“UTx 缓冲区 2”中提取数据，然后传输另外 8 个数据。

UART Rx eDMA 通道执行从 UART 缓冲区到“URx 缓冲区 1”的 4 次传输，然后重新载入 TCD。之后，它将接收的其他 8 个数据保存在“URx 缓冲区 2”中。

FlexIO Tx eDMA 通道从“XTx 缓冲区 1”执行 4 次传输，然后使用 SGA 功能重新载入 TCD。之后，eDMA 指向“XTx 缓冲区 2”，然后再传输另外 8 个数据。

我们可以观察到，分散聚集的重新载入干预导致 UART 输出延迟，但在检查内部缓冲区“URx 缓冲区 1”时，我们没有发现数据丢失。至于 FlexIO，我们既没有观察到延迟也没发现数据丢失。

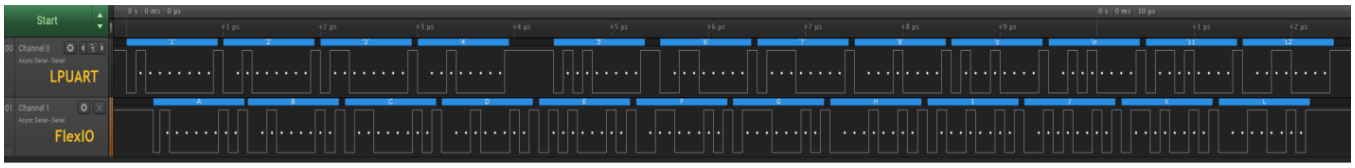

图 7. 使用分散聚集功能的 UART 和 FlexIO 通信

3.1.2. SAI

本文档将介绍一个在 I2S 模式下的、16 KHz、16 位的 2 个 Tx 通道的用例。有关 SAI 模块中的其他 eDMA 用法（乒乓缓冲器、分离通道等），请参见 AN12202。

I2S（或 TDM 或任何其他音频协议通信）是同步通信的一种特殊情况，其中数据数组中可以有不同的格式。

在经典的 Tx I2S 通信中，每个帧（左和右）在一行中有两个数据字，在某些情况下，音频数据应在每个通道中分别处理，对于这些场景，应用代码已分隔了“左”和“右”缓冲区，并且 eDMA 用于将数据从不同地址（左和右缓冲区）传输到一个缓冲区地址（内部 SAI tx 缓冲区）。为此，可以将 TCD 配置为每次传输都具有偏移。

现在，本用例的每个帧都在两行中有 4 个数据字（Ch0 L，Ch0 R，Ch1 L 和 Ch1 R），通常用户可以为每个音频通道使用 2 个 eDMA 通道，但是 eDMA 通道需要在次循环传输中链接，正如我们已经讲述过的，这会减少可以在 TCD 配置上设置的传输数量。S32K148 SAI 模块具有“组合”功能，该功能允许“多路复用”SAI 通道，因此基本上 SAI 通道 0 缓冲区中的每个写入都被分散到所有启用的通道中。

在下图中，您可以看到如何将 SAI 模块的组合模式和 eDMA 模块的次要环路偏移相结合，用于带有单独的缓冲区的两个通道的左右配置。

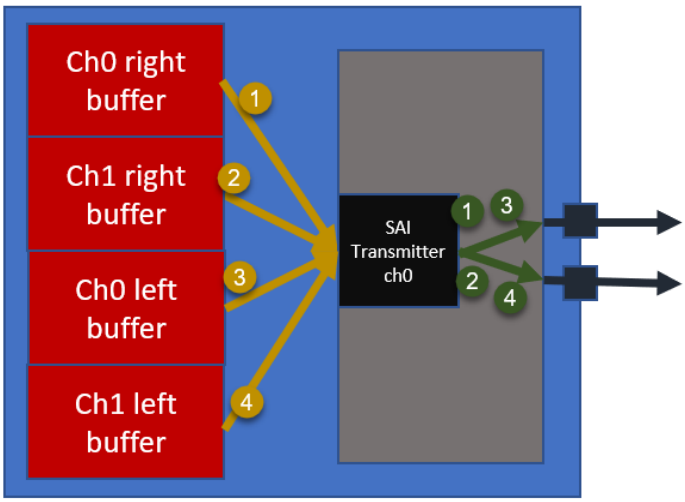

图 8. SAI 模块中的组合模式

这种实现方式的优点是只使用 1 个 eDMA 通道，减少了 eDMA 资源占用，而且不会增加分散聚集的时间，也不会迫使用户减少链接通道的数据长度。

我们这里不讲述采样频率，但在 SAI 模块中要注意，虽然没有明确的最小总线时钟频率规范，但总线时钟必须足够快（相对于位时钟），以确保 FIFO 可以得到服务，而不会产生发送器 FIFO 欠载或接收器 FIFO 溢出的情况。

结果

下面的示波器显示该实现正常工作，可以看到 0x000x 表示通道 0 L 数据，0x100x 表示通道 0 R 数据，0x200x 表示通道 1 L 数据，0x300x 表示通道 1 R 数据。所有这些都在一个帧中（一个帧周期）。

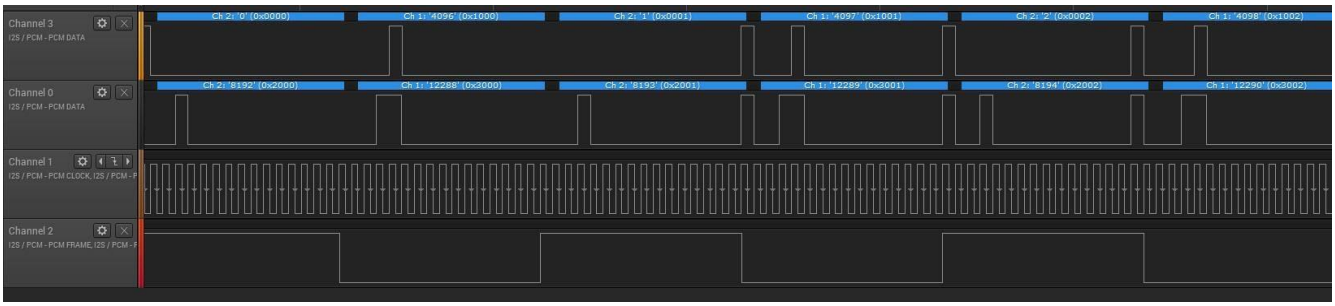

图 9. 采用 eDMA 的 2 个 Tx SAI 通道

3.2. 通过 eDMA 进行 ADC 读数

众所周知，ADC 读数可以通过 eDMA 完成，这就避免了 CPU 干预 ADC_R 寄存器和清除标志。这样做通常的缺点是，对一个固定的 ADC 通道，要改变它，就需要使用 CPU。

有一些模块专门用于自动化所有 ADC 过程，包括 ADC 通道更改（如其他汽车器件中的 BCTU），但我们可以使用 eDMA 的某些功能来模拟这种功能。灵活的扫描模式允许执行 ADC 转换，将值保存在内部缓冲器中，并自动更改 ADC 通道，所有这些都通过 2 个 eDMA 通道和 1 个 ADC 转换器来实现。

基本过程可分为以下三个部分：

1. 转换完成标志 COCO 请求一次对 eDMA 通道 1 的 eDMA 传输。
2. eDMA 通道 1 将 ADC 结果值传输到 SRAM 缓冲区。
3. 将通道 1 链接到 eDMA 通道 0，通道 0 将一个新的 ADC 通道从 RAM 多路复用阵列传输到 ADCx_SC（选择器通道）。重复步骤 1。

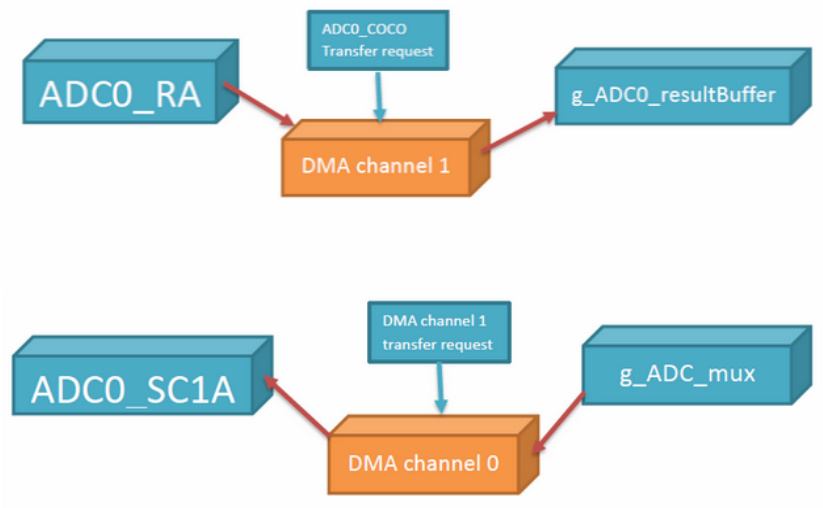

图 10. 采用 eDMA 的 2 个 Tx SAI 通道

所有这些过程都由定时器控制，定时器触发 ADC 转换（周期性触发）。这些步骤可以根据需要重复任意次，并且可以根据需要保存任意数量的 ADC 通道，因此这将标记 ADC 转换的次数和 eDMA 将执行的传输次数。

例如，如果有 3 个 ADC 通道，每个通道需要 4 个测量值，那么内部 SRAM 结果缓冲区将保存 3x4=12 个值。现在，如果用 0.5 s 的 PDB 定时器触发 ADC，这个过程将需要 0.5x12 = 6s。

这里的一个重要因素是，触发转换的定时器也会设置时隙请求，这意味着对于我们的 ADC 转换示例，必须在 500 ms 内发生 2 次 eDMA 传输。创建 ADC SAR 的目的不是用于快速转换，但用户必须确保有足够的时间进行 ADC 转换和 2 次 eDMA 传输（以及在其他应用中使用 eDMA 导致可能增加的时间）。

结果

下表显示该实现的一个示例，其中 3 个 ADC 通道（电位计、VREFH 和 VREFL）和 4 个测量值保存在内部 SRAM 存储器中。

ADC_Results	volatile uint32_t [20]	0x2000001c <ADC_Results>
ADC_Results[0]	volatile uint32_t	1425
ADC_Results[1]	volatile uint32_t	4095
ADC_Results[2]	volatile uint32_t	0
ADC_Results[3]	volatile uint32_t	1424
ADC_Results[4]	volatile uint32_t	4095
ADC_Results[5]	volatile uint32_t	0
ADC_Results[6]	volatile uint32_t	1424
ADC_Results[7]	volatile uint32_t	4095
ADC_Results[8]	volatile uint32_t	0
ADC_Results[9]	volatile uint32_t	1425
ADC_Results[10]	volatile uint32_t	4095
ADC_Results[11]	volatile uint32_t	0

3.3. 其他因素

现在让我们来评估还有哪些因素可能影响 SoC 级别的 eDMA 性能。

假设需要将大量数据从 Flash 移动到 SRAM 上部区域，然后逐个处理数据，一张[320x240]像素的图像可以存储在 76,800 字节的数组中（在实际场景中，这些数据可以通过用户代码采集来自传感器（例如相机）的数据或其他设备传输的数据来获得）。

为了获得 eDMA 执行这些操作所花费时间的测量值，要在单个请求操作中将整个数组传输到单个 SRAM-U 位置，传输将由软件启动，相应的通道中断将被启用。

MCU 将从 8 MHz SIRC 进入 VLPR 模式，内核和总线时钟将设置为 1 MHz，闪存时钟将设置为 0.25 MHz。启动 eDMA 传输后，CPU 将进入无操作的无限循环。

3.3.1. CACHE 高速缓存

高速缓冲是包含地址信息（标签）和相关数据的高速存储器位置块。其目的是减少存储器访问的平均时间。

在这种情况下，我们对高速缓存运行第一个局部性原则感兴趣，即空间局部性：“访问了一个位置之后，接下来很可能会继续访问相邻位置”。

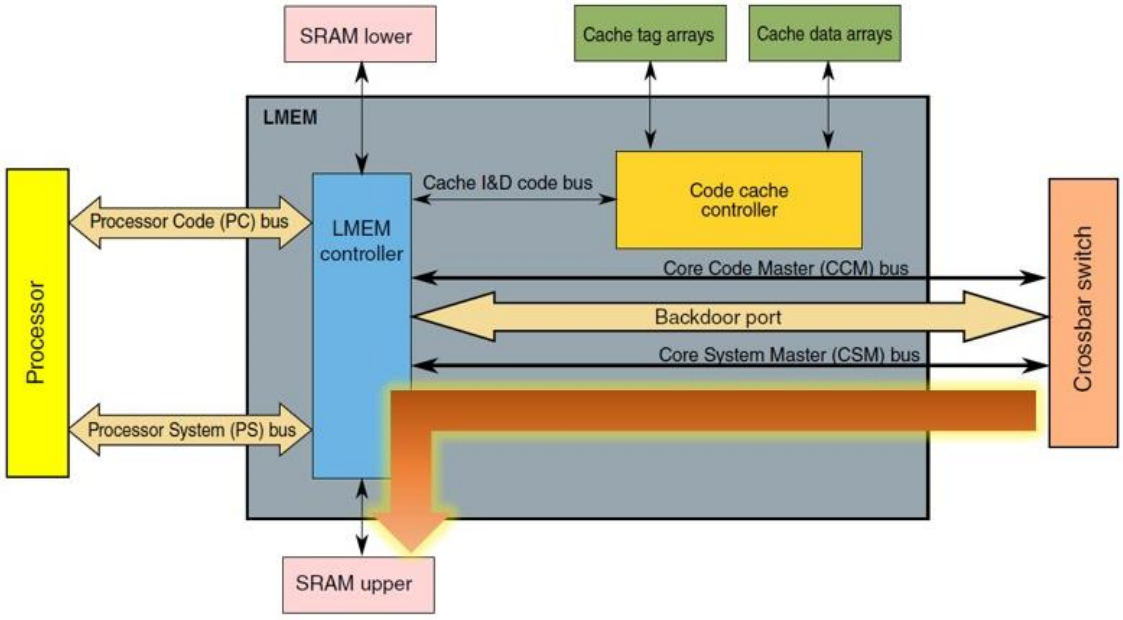

图 11. 将数据从 FLASH 移至 SRAM 上部

空间局部性属性将多个位置分组到同一个标签下。当数据被载入高速缓存后，后续加载的访问时间会缩短，从而提高整体性能。

3.3.2. 推测性闪存读取

闪存控制器（FMC）有一个缓冲区，可提前读取闪存中的下一个字。

启用推测读取后，闪存控制器会在读取完成后立即请求下一个顺序地址，这有助于减少甚至消除访问顺序代码和/或数据时的等待状态。

注意

闪存加速功能默认为启用。

结果

复位后，高速缓存被禁用，而数据预取和闪存推测选项被启用。让我们禁用数据闪存预取 *（MSCM_OCMDR[OCM1]）*，看看需要多长时间。

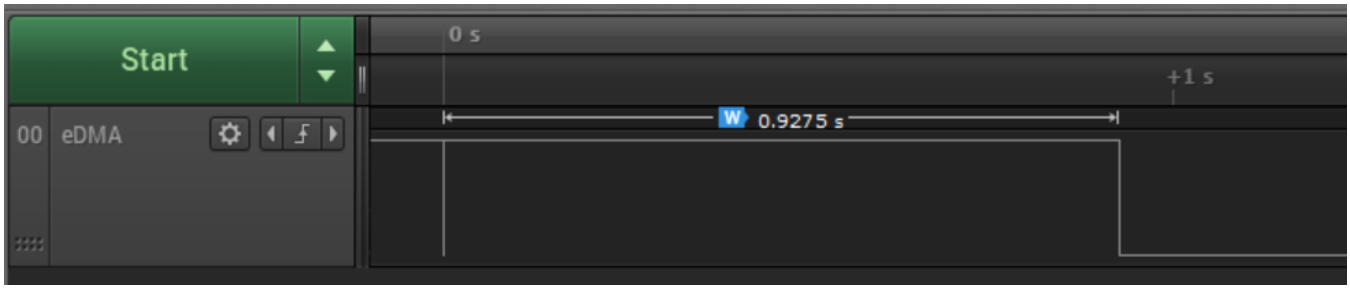

图 12. eDMA 传输 76,800 字节

这用了 0.9275 秒，让我们启用数据推测（复位后的默认设置）。

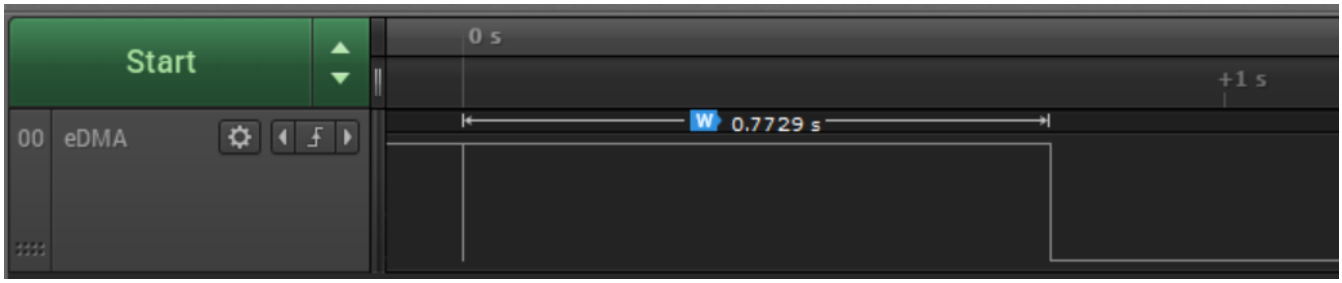

图 13. eDMA 传输 76,800 字节，启用数据闪存预取功能

传输时间缩短了 16.66%，但我们还必须检查高速缓存在这种情况下的作用，为此，我们将使用 LMEM 控制器启用高速缓存。

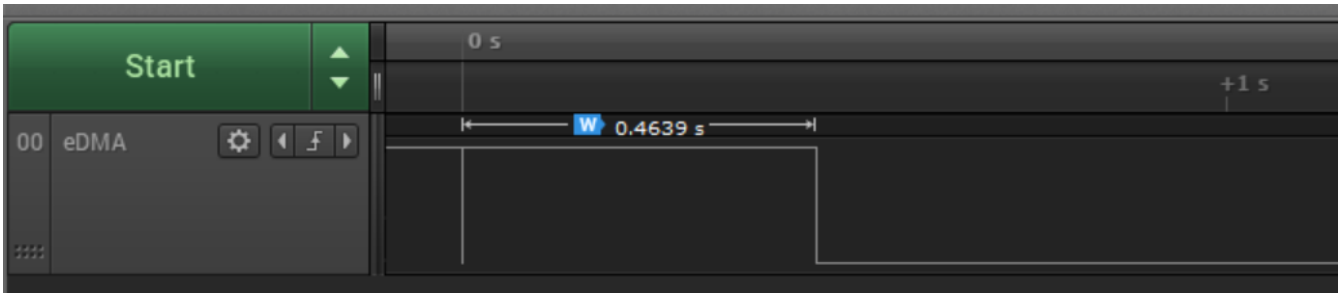

图 14. eDMA 传输 76,800 字节，启用了高速缓存

可以看到时间减少到 0.4639 秒（原来时间的一半）。

4. 结论

- 在分散-聚集模式下，TCD 数据传输结构会导致数据传输的小延迟。
- eDMA 模块中包含一些功能，可以改善数据传输性能。
- eDMA 模块外部有一些缓存，可以使能这些缓存来提高某些应用的性能。
- 尽管 eDMA 传输需要几个时钟周期，但整个应用的时间可能主要受到总线流量的影响，而不是受 eDMA 相关问题的影响。
- 用户应该考虑到最坏情况的一些关键点，并尝试使用不同的功能和实现来管理它们。

5. 参考资料

- S32K1xx 数据手册。2020
- S32K14x 系列参考手册。2020
- https://www.nxp.com/docs/en/application-note/AN4590.pdf

How to Reach Us:

Home Page:
nxp.com.cn

Web Support:
nxp.com.cn/support

Document Number: AN12972
Rev. 0
02/2021