Application Note

调整 TDA4x 和 AM6x 器件上关于 DDR 带宽优化的 QoS 和 CoS 设置

摘要

TDA4x 和 AM6x 器件都包含服务质量 (QoS) 方案，可优先处理特定事务并平衡不同应用和 IP 的负载。如果未针对设备和应用调整 QoS 设置，可能会导致意外或不良行为。本文档中涵盖的案例研究演示了这种缺少调优导致的不良行为。在运行自动代客泊车 (AVP) 演示时，显示器出现同步丢失错误。这是由于缺少正确的 QoS 和服务等级 (CoS) 设置，导致显示器未能获得应有的优先级。一旦 DDR 控制器内的 CoS 设置设为正确的值，就不再观察到同步丢失问题。

内容

商标

1 TDA4VH 内的数据移动

备注

请阅读 TDA4VH TRM 的 **3. 系统互连**小节，了解更多详细信息。

要了解如何平衡和优先处理事务，就必须全面掌握系统互连和数据流动的方式。在以下案例研究中，我们将主要关注显示子系统 (DSS) 和 C7x 到 DDR 子系统 (DDRSS) 的数据路由，但最好对所有启动器和目标有大致的了解。

图 1-1（取自 TDA4VH TRM）显示了启动器和目标及其请求方向的总体示意图。DSS 位于“启动器”模块内。TDA4VH TRM 中的 **3.2.6 启动器-目标连接**包含**连接矩阵**，进一步详细说明了系统内启动器和目标之间的关系。

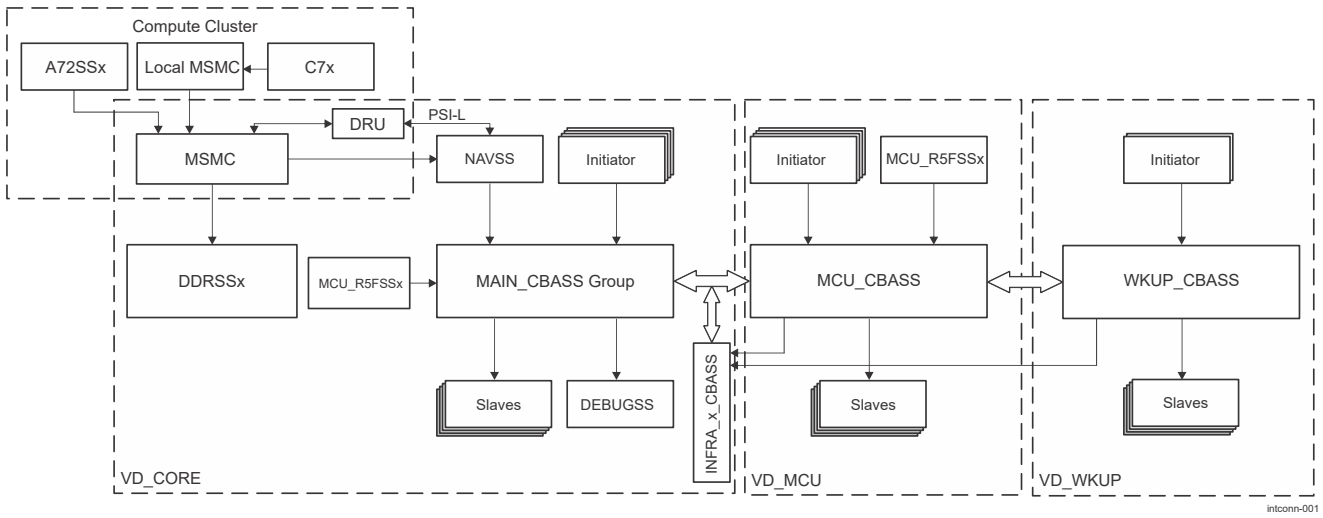

图 1-1. 器件系统互连概述

C7x 到 DDR 的请求主要遵循以下路径： C7x → MSMC → DDRSS

DSS 到 DDR 的请求主要遵循以下路径：DSS → 主 CBASS → NAVSS → MSMC → DDRSS

1.1 通用总线架构子系统 (CBASS)

系统器件中的所有模块和子系统通过系统互连相互通信。内容划分为以下几部分：

- CBASS0 互连
- INFRA_CBASS0 互连
- MCU_CBASS0 互连
- WKUP_CBASS0 互连

大多数模块连接到 MAIN 域中的 CBASS0 互连。CBASS 包含某些模块的服务质量机制，将在后面的章节中进行说明。

1.2 导航器子系统 (NAVSS)

备注

请阅读 TDA4VH TRM 的 **10.2.10 NAVSS 北桥 (NB)** 小节，了解更多详细信息。

小心

本节将讨论主 NAVSS (NAVSS0)，而不是 MCU NAVSS。

NAVSS0 包含以下元件：

- 统一 DMA 子系统 (UDMASS)
- 模块子系统 (MODSS)
- 北桥子系统 (NBSS)
- 虚拟化子系统 (VirtSS)
- ECC 聚合器

图 1-2（取自 TDA4VH TRM）展示了 NAVSS0 硬件元件及其集成。

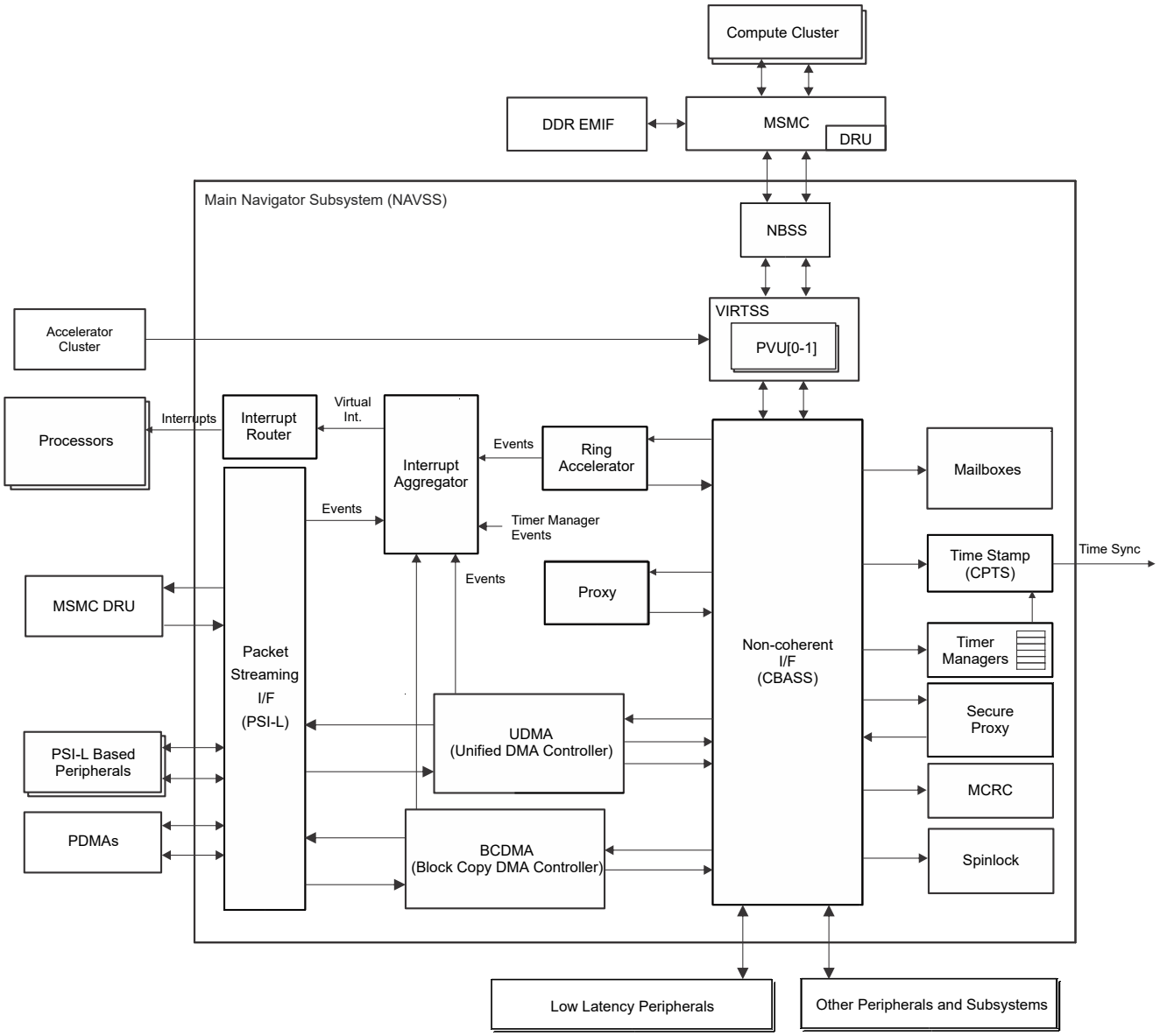

图 1-2. NAVSS0 顶层方框图

1.2.1 NAVSS 北桥 (NB)

备注

请阅读 TDA4VH TRM 的 **10.2.10 NAVSS 北桥 (NB)** 小节，了解更多详细信息。

NB 在 VBUSM 接口和 VBUSM.C 接口之间桥接。换句话说，它桥接 CBASS (VBUSM) 和 MSMC (VBUSM.C)。NAVSS 包含 2 个北桥：NB0 和 NB1。

图 1-3（取自 DRA829/TDA4VM TRM）显示了 NAVSS0 的 NB0 和 NB1 的总体结构。

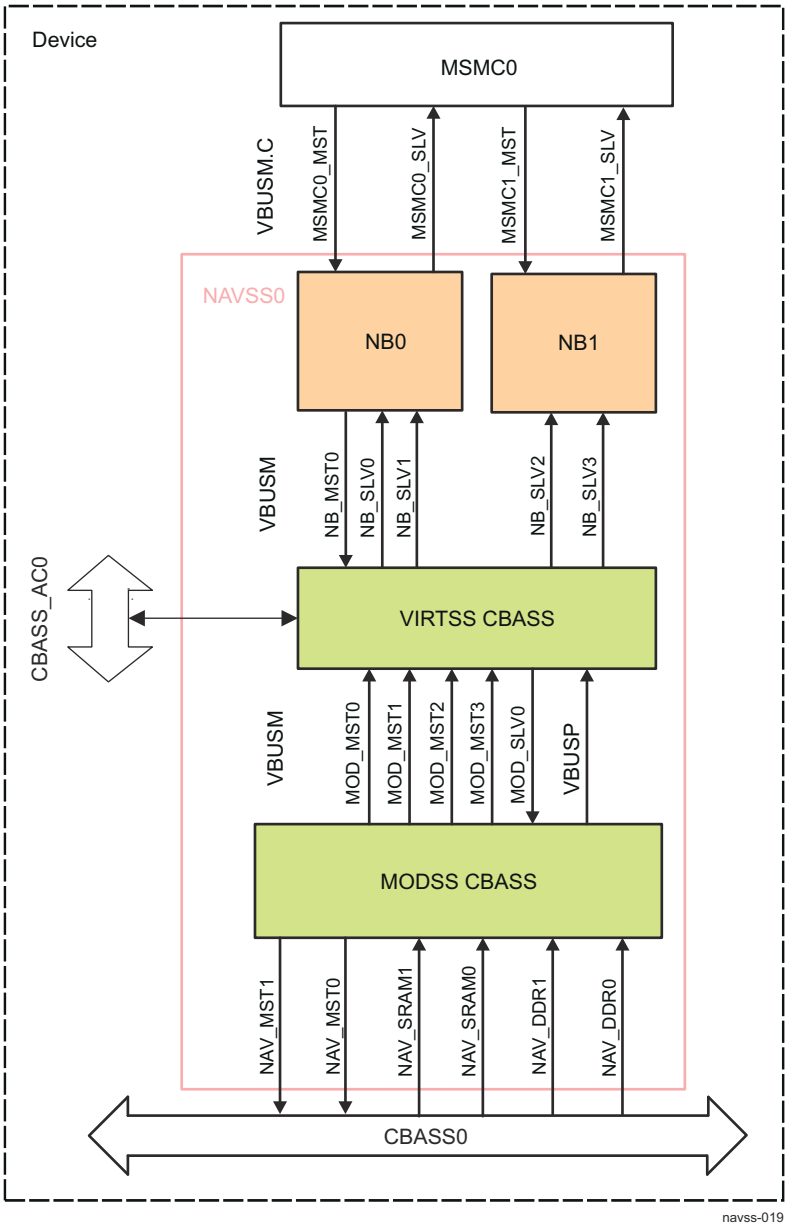

图 1-3. NB 概述

NB 的服务质量机制将在后面的章节中进行说明。

1.3 多核共享内存控制器 (MSMC)

备注

请阅读 TDA4VH TRM 的 **8.1 多核共享存储器控制器 (MSMC)** 小节，了解更多详细信息。

MSMC 提供与计算集群和系统其余部分的内部处理元件之间的高带宽数据移动和资源访问。

图 1-4（取自 TDA4VH TRM）展示了 MSMC 及其周围模块的概述。

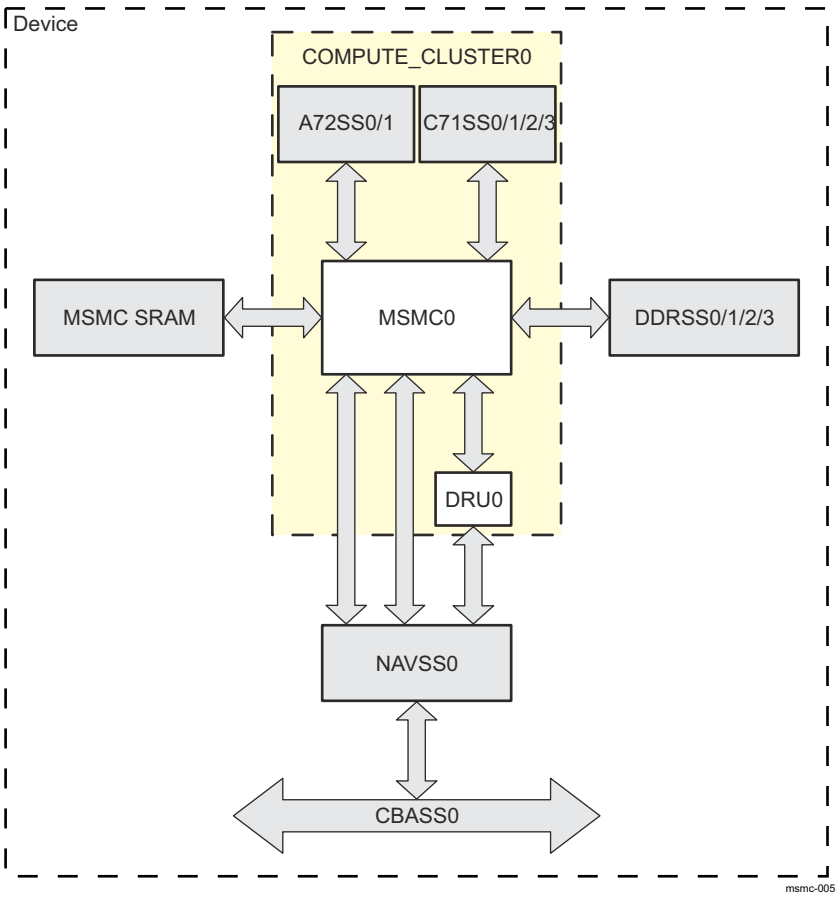

图 1-4. MSMC 概述

MSMC 的服务质量机制将在后面的章节中进行说明。

2 服务质量 (QoS)

备注

请阅读 TDA4VH TRM 的 **3.2.1 服务质量 (QoS)** 小节，了解更多详细信息。

服务质量是指通过一系列机制对网络内的流量进行管控。在本例中，网络是 TDA4VH 器件。有两种系统级的服务质量实现方式：一种是基于顺序 ID，另一种是基于优先级的仲裁机制。顺序 ID 用于控制事务主设备及其通道（若存在多个通道）的映射关系，从而实现不同路径间的流量均衡。优先级机制提供仲裁功能来改善延迟和带宽。

事务的优先级相对容易理解。优先级范围为 0 至 7；其中 0 代表最高优先级，7 代表最低优先级。优先级较高的事务将先于优先级较低的事务进行处理。设置请求优先级的方法因 IP 而异，但 IP 中通常有一个 CTRL MMR 或寄存器来设置优先级。例如，DSS 的 DSS_DISPC_0_COMMON_M_DSS_CBA_CFG 寄存器用于控制 DSS 请求的优先级。

表 2-1. DSS 优先级寄存器

寄存器	字段	位	说明
DSS_DISPC_0_COMMON_M_DSS_CBA_CFG	PRI_HI	5:3	PRI_HI 总线上从 DSS 发送到 CBA 的值用于指示高优先级 [MFLAG] 事务的优先级。 • 值 0x0 表示最高优先级 • 值 0x7 表示最低优先级
	PRI_LO	2:0	PRI_LO 总线上从 DSS 发送到 CBA 的值用于指示正常 [非 MFLAG] 事务的优先级。 • 值 0x0 表示最高优先级 • 值 0x7 表示最低优先级

可以在启动器的寄存器或 CBASS 寄存器中设置事务的顺序 ID。顺序 ID 的效果不如优先级机制简单，数据移动架构中的不同子系统使用顺序 ID 通过不同路径来路由和平衡负载。

备注

默认情况下，所有主设备都会发送顺序 ID = 0 的事务。

2.1 NAVSS0

NAVSS0 内部的互连使用顺序 ID 来提供多条（至少两条）到 DDR 的并行路径，以及一组到 SRAM 的多条（至少两条）并行路径。NAVSS0 还为到 SoC 级别的 DMA 流量提供多条（至少两条）并行路径，这可以提供隔离式 DMA 流量路径。

在 NAVSS0 内，两个北桥会路由进出 MSMC 的 SRAM 流量和 DDR 流量。北桥 0 会路由 SRAM 流量，北桥 1 会路由 DDR 流量。

2.1.1 NAVSS0 北桥

备注

请阅读 TDA4VH TRM 的 **10.2.10.2.10 服务质量**小节，了解更多详细信息。

每个北桥接收多个来源，按顺序 ID 进行分隔。对于北桥 0：源 0 接收所有顺序 ID 为 0-7 的事务，源 1 接收所有顺序 ID 为 8-15 的事务。对于北桥 1：源 0 接收所有顺序 ID 为 0-4 的事务，源 1 接收顺序 ID 为 5-9 的事务，源 2 接收顺序 ID 为 10-15 的事务。这些并行路径通过顺序 ID 对事务负载进行分配。每个源接收的顺序 ID 不能由用户进行编程设定。

小心

每个源收到的顺序 ID 可以在不同的设备之间发生变化。例如，TDA4VM 仅为北桥 0 和 1 的两个源进行。

顺序 ID 也会影响命令的顺序。从具有特定顺序 ID 值的 VBUSM 接口接收到的每个读取命令都将以完全相同的顺序返回其读取数据，即使命令来自不同的主设备也是如此。如果读取时使用不同的顺序 ID 值，则可以按任何顺序返回该读取数据，以 VBUSM.C 接口上首先接收到的为准。

要将来自 VBUSM.C 的返回流量路由回正确的 VBUSM 源，则使用顺序 ID。因此，源的顺序 ID 不能重叠。由于源会固有地路由不同的顺序 ID，因此不会出现重叠的问题。

2.1.1.1 正常流量与实时流量的区别

北桥使用 3 个线程将流量分隔给 MSMC：

- 线程 0：指向 VBUSM.C 的命令
- 线程 1：来自 VBUSM.C 的命令
- 线程 2：指向 VBUSM.C 的实时命令

为了支持服务质量，北桥提供了寄存器来将源映射到普通线程 (0) 或实时线程 (2)。映射到实时线程的任何源将在正常线程之前进行仲裁。如果有多个源映射到同一个线程，则根据优先级对它们进行仲裁；如果它们具有相同的优先级，则通过轮询方式进行仲裁。通过这种方式，北桥能够按顺序 ID 分配流量和优先级（并以某种方式增加可能的优先级数）。

在 NAVSS North Bridge MMR 寄存器，特别是 NAVSS_NORTH_x_NBSS_NBx_MMRS_threadmap（x 表示 0 或 1）寄存器中，对顺序 ID 是映射到正常线程还是实时线程进行编程设定。

表 2-2. NAVSS 北桥线程映射寄存器

寄存器	字段	位	说明
NAVSS_NORTH_0_NBSS_NB0_MMRS_threadmap	保留	31:3	保留
	线程映射	1	将顺序 ID 8-15 映射到 VBUSM.C 线程编号： • 0：VBUSM.C 线程 0（非实时流量） • 1：VBUSM.C 线程 2（实时流量）
		0	将顺序 ID 0-7 映射到 VBUSM.C 线程编号： • 0：VBUSM.C 线程 0（非实时流量） • 1：VBUSM.C 线程 2（实时流量）

表 2-2. NAVSS 北桥线程映射寄存器（续）

寄存器	字段	位	说明
NAVSS_NORTH_1_NBSS_NB1_MMRS_threadmap	保留	31:3	保留
	线程映射	2	将顺序 ID 10-15 映射到 VBUSM.C 线程编号： • 0：VBUSM.C 线程 0（非实时流量） • 1：VBUSM.C 线程 2（实时流量）
		1	将顺序 ID 5-9 映射到 VBUSM.C 线程编号： • 0：VBUSM.C 线程 0（非实时流量） • 1：VBUSM.C 线程 2（实时流量）
		0	将顺序 ID 0-4 映射到 VBUSM.C 线程编号： • 0：VBUSM.C 线程 0（非实时流量） • 1：VBUSM.C 线程 2（实时流量）

小心

NAVSS_NORTH_x_NBSS_NBx_MMRS_threadmap 寄存器的字段因器件而异。上表代表了 TDA4VH。

2.2 多核共享内存控制器 (MSMC)

备注

请阅读 TDA4VH TRM 的 **8.1.2.11 MSMC 服务质量**小节，了解更多详细信息。

MSMC 提供两类流量：实时 (RT) 和非实时 (NRT)。这两类流量对应于北桥内的实时线程 (2) 和正常线程 (0)。MSMC 在每个仲裁点提供专用缓冲，只能由 RT 流量消耗，因此 NRT 流量无法完全耗尽 RT 请求。

无法对 MSMC QoS 硬件进行软件控制。不支持 QoS 功能的接口会将所有流量表示为非实时流量。

2.3 DDR 子系统 (DDRSS)

DDRSS 包含通过 MSMC2DDR 桥接器实现的服务质量机制，同时还包含其自己独特的机制：服务等级 (CoS)。DDR 控制器利用其 CoS 机制将 VBUSM.C 线程和优先级映射到其内部线程和优先级。

2.3.1 MSMC2DDR 桥接器

备注

请阅读 TDA4VH TRM 的 **8.2.3.1 DDRSS MSMC2DDR 桥接器**和 **8.2.3.1.1 VBUSM.C 线程**小节，了解更多详细信息

MSMC2DDR 桥接器支持 2 个线程：

- 高优先级线程 (HPT)：VBUSM.C 线程 2 上接收到的流量属于 HPT

- 低优先级线程 (LPT)：VBUSM.C 线程 0 上接收到的流量属于 LPT

HPT 的优先级高于 LPT，来自命令队列的命令在执行时可能顺序异常。这可确保即使 LPT 被阻止，HPT 也能得到保证执行。

由于 MSMC2DDR 桥接器保持线程间的数据一致性，因此可能会出现优先级反转。由于地址冲突而依赖于 LPT 事务的任何 HPT 事务都会被阻止，直到执行相应的 LPT 事务。

2.3.2 服务等级 (CoS)

备注

请阅读 TDA4VH TRM 的 **8.2.3.1.2 服务等级 (CoS)** 小节，了解更多详细信息。

服务等级特定于 DDRSS，并且用于控制系统 (VBUSM.C) 优先级如何映射到 DDRSS 内部优先级。MSMC2DDR 桥接器具有以下寄存器来将 VBUSM.C 优先级映射到 DRR 控制器优先级：

- 范围匹配寄存器：
  - DDRSS_V2A_R1_MAT_REG
  - DDRSS_V2A_R2_MAT_REG
  - DDRSS_V2A_R3_MAT_REG
- 优先级映射寄存器：
  - DDRSS_V2A_LPT_DEF_PRI_MAP_REG
  - DDRSS_V2A_LPT_R1_PRI_MAP_REG
  - DDRSS_V2A_LPT_R2_PRI_MAP_REG
  - DDRSS_V2A_LPT_R3_PRI_MAP_REG
  - DDRSS_V2A_HPT_DEF_PRI_MAP_REG
  - DDRSS_V2A_HPT_R1_PRI_MAP_REG
  - DDRSS_V2A_HPT_R2_PRI_MAP_REG
  - DDRSS_V2A_HPT_R3_PRI_MAP_REG

图 2-1（取自 TDA4VH TRM）展示了优先级映射寄存器如何将传入的优先级映射到相应的 DDR 优先级。

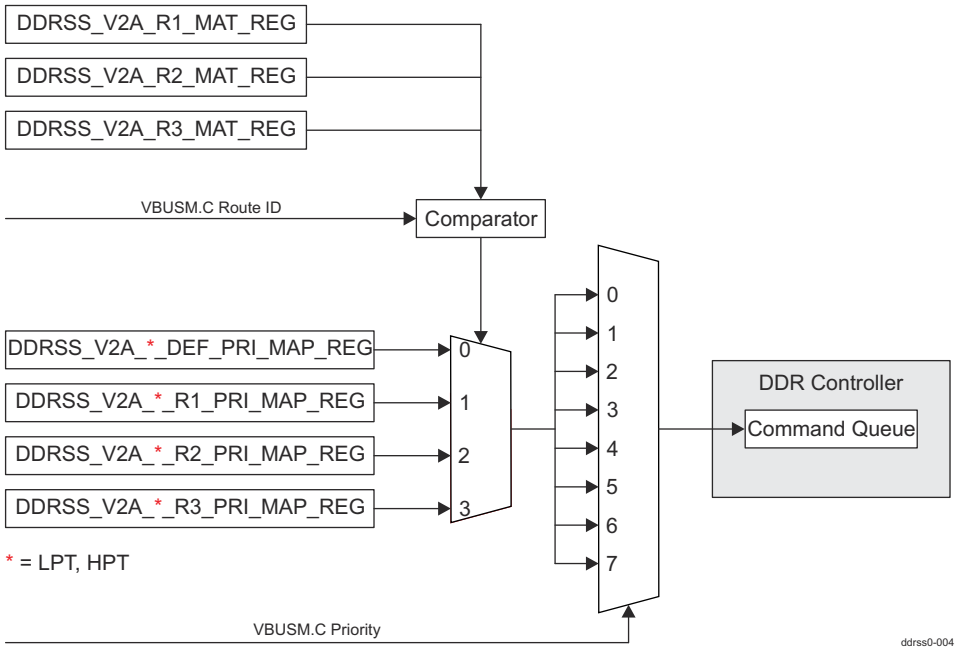

图 2-1. DDRSS CoS 映射图

2.4 QoS 摘要

我们已经讨论了负责控制 QoS 的机制和 IP，接下来有必要将所有详细信息整合在一起，从高层次的角度了解其整体含义。

如前所述，QoS 通过两种机制实现：顺序 ID 和优先级。顺序 ID 是请求的可编程字段，会影响在不同 IP 之间路由请求的方式。顺序 ID 的重点是提供一种机制来平衡数据在不同路径之间的流动方式。优先级用于控制为请求提供服务的顺序。

数据移动架构子系统还引入了非实时 (NRT) 和实时 (RT) 线程的概念。NRT 和 RT 属性的名称因 IP 而异。

表 2-3. 不同 IP 的非实时和实时命名

IP	非实时 (NRT)	实时 (RT)
MSMC	非实时	实时
NAVSS 北桥	线程 0（正常）	线程 2（实时）
DDR 控制器	低优先级线程 (LPT)	高优先级线程 (HPT)

RT 线程在 NRT 线程之前进行仲裁，因此 RT 线程上的事务被赋予更高的优先级。事务是属于 NRT 线程还是 RT 线程，由顺序 ID 和北桥编程设定方式决定。当您将 NRT 和 RT 线程的概念与优先级结合时，可以按下表查看请求的层次结构。

表 2-4. 不同非实时和实时请求的优先级

NRT 或 RT 请求	优先级	优先级
实时	0	最高优先级
	1	优先级按列表顺序逐级降低
	2	
	3	
	4	
	5	
	6	
	7	
非实时	0	
	1	
	2	
	3	
	4	
	5	
	6	
	7	最低优先级

3 案例研究：显示同步丢失问题

让我们来看一个案例研究，了解 QoS 机制如何平衡设备上的负载。

3.1 问题说明

在插入 4k 显示屏的情况下运行 AVP（自动代客泊车）演示时，显示屏会出现频繁的同步丢失。

3.2 设置和重新创建

3.2.1 要求

表 3-1. 硬件和软件要求

条目	链接	注释
J784S4XEVM	https://www.ti.com/tool/J784S4XEVM	不适用
SD 卡	不适用	不适用
4K 30fps 显示器	不适用	将显示器连接至 Display Port1
PROCESSOR-SDK-RTOS-J784S4	ti-processor-sdk-rtos-j784s4-evm-09_02_00_05.tar.gz	版本 09.02
SOC 通用 TI 示例输入数据集	psdk_rtos_ti_data_set_09_02_00.tar.gz	在 PROCESSOR-SDK-RTOS-J784S4 页面中列出
特定于 SOC 的 tidl 模型	psdk_rtos_ti_data_set_09_02_00_j784s4.tar.gz	在 PROCESSOR-SDK-RTOS-J784S4 页面中列出
PROCESSOR-SDK-LINUX-J784S4	ti-processor-sdk-linux-adas-j784s4-evm-09_02_00_05-Linux-x86-Install.bin	版本 09.02
RTOS 补丁	rtos-patches.tar.xz • 0001-vision_apps-Remove-the-DSS-application-from-MCU2_0.patch • 0002-vision_apps-Remove-display-use-from-the-AVP-demo.patch	应用于 vision_apps 存储库的补丁
Linux 补丁	linux-patches.tar.xz • 0001-arm64-dts-ti-k3-j784s4-vision-apps-Re-enable-DSS-for.patch	应用于 ti-linux-kernel 存储库的补丁

3.2.1.1 RTOS 补丁

3.2.1.1.1 0001-vision_apps-Remove-the-DSS-application-from-MCU2_0.patch

```
From d5bd0778612110390ed7a20e7bb9afb4c95f0c25 Mon Sep 17 00:00:00 2001
From: Jared McArthur <j-mcarthur@ti.com>
Date: Tue, 28 Jan 2025 11:14:48 -0600
Subject: [PATCH 1/2] vision_apps: Remove the DSS application from MCU2_0

Remove the DSS application from MCU2_0 to give Linux control of the
display driver.

Signed-off-by: Jared McArthur <j-mcarthur@ti.com>
---
 platform/j784s4/rtos/common/app_cfg_mcu2_0.h | 13 ++++++++----
 1 file changed, 9 insertions(+), 4 deletions(-)

diff --git a/platform/j784s4/rtos/common/app_cfg_mcu2_0.h b/platform/j784s4/rtos/common/
app_cfg_mcu2_0.h
index a18c41b..8a96d2e 100755
--- a/platform/j784s4/rtos/common/app_cfg_mcu2_0.h
+++ b/platform/j784s4/rtos/common/app_cfg_mcu2_0.h
@@ -78,7 +78,7 @@
 #ifdef BUILD_MCU_BOARD_DEPENDENCIES
```

```
     #define ENABLE_CSI2RX
-    #define ENABLE_CSI2TX
+    // #define ENABLE_CSI2TX
     #undef  ENABLE_DSS_HDMI

     /* IMPORANT NOTE:
@@ -86,8 +86,8 @@
      * - When ENABLE_DSS_SINGLE is defined, only one of ENABLE_DSS_DSI or ENABLE_DSS_EDP should be
defined
      * - When ENABLE_DSS_DUAL is defined, ENABLE_DSS_DSI and ENABLE_DSS_EDP are not used, both EDP
and DSI are enabled unconditionally
      */
-    #define ENABLE_DSS_SINGLE
-    #undef  ENABLE_DSS_DUAL
+    // #define ENABLE_DSS_SINGLE
+

 #if defined(ENABLE_DSS_DUAL)
     #undef  ENABLE_DSS_SINGLE
@@ -102,7 +102,12 @@
     #undef  ENABLE_CSI2TX
 #endif
     #define ENABLE_I2C
-    #define ENABLE_BOARD
+    // #define ENABLE_BOARD
+    #undef  ENABLE_DSS_DUAL
+    #undef  ENABLE_DSS_SINGLE
+    #undef  ENABLE_DSS_DSI
+    #undef  ENABLE_DSS_EDP
+    #undef  ENABLE_DSS_HDMI
 #else

     #undef ENABLE_CSI2RX
--
2.34.1
```

3.2.1.1.2 0002-vision_apps-Remove-display-use-from-the-AVP-demo.patch

```
From c8059ede6e8a5cb38933f01b6c275a84539cd267 Mon Sep 17 00:00:00 2001
From: Jared McArthur <j-mcarthur@ti.com>
Date: Tue, 28 Jan 2025 11:29:07 -0600
Subject: [PATCH 2/2] vision_apps: Remove display use from the AVP demo

Remove display calls from the auto valey parking (AVP) demo. The demo
traditionally outputs to a display; remove this functionality so Linux
can own the display driver.

Disabling the display within the AVP demo allows for testing Linux's
display driver while the C7x cores are loaded.

Signed-off-by: Jared McArthur <j-mcarthur@ti.com>
---
 .../app_tidl_avp/avp_img_mosaic_module.c      |  73 ++++++++++-
 apps/dl_demos/app_tidl_avp/concerto.mak       |   2 +-
 apps/dl_demos/app_tidl_avp/main.c             | 119 ++++++++++++++----
 3 files changed, 97 insertions(+), 97 deletions(-)

diff --git a/apps/dl_demos/app_tidl_avp/avp_img_mosaic_module.c b/apps/dl_demos/app_tidl_avp/
avp_img_mosaic_module.c
index 21ace33..d43f7ad 100644
--- a/apps/dl_demos/app_tidl_avp/avp_img_mosaic_module.c
+++ b/apps/dl_demos/app_tidl_avp/avp_img_mosaic_module.c
@@ -61,7 +61,14 @@
  */

 #include "avp_img_mosaic_module.h"
-
+#include <fcntl.h>
+#include <linux/fb.h>
+#include <stdio.h>
+#include <stdlib.h>
+#include <string.h>
+#include <sys/ioctl.h>
+#include <sys/mman.h>
+#include <unistd.h>
```

```
 vx_status app_init_img_mosaic(vx_context context, ImgMosaicObj *imgMosaicObj, vx_int32 bufq_depth)
 {
@@ -143,7 +150,68 @@ void app_create_graph_img_mosaic(vx_graph graph, ImgMosaicObj *imgMosaicObj,
vx_

     return;
 }
+/*
+#include <fcntl.h>
+#include <linux/fb.h>
+#include <stdio.h>
+#include <stdlib.h>
+#include <string.h>
+#include <sys/ioctl.h>
+#include <sys/mman.h>
+#include <unistd.h>
+
+int main(int argc, char *argv[]) {
+int fb_fd = open("/dev/fb0", O_RDWR);
+if (fb_fd == -1) {
+perror("Error: cannot open framebuffer device");
+return 1;
+}
+
+struct fb_var_screeninfo vinfo;
+struct fb_fix_screeninfo finfo;
+
+// Get fixed screen information
+if (ioctl(fb_fd, FBIOGET_FSCREENINFO, &finfo)) {
+perror("Error reading fixed information");
+return 2;
+}
+
+// Get variable screen information
+if (ioctl(fb_fd, FBIOGET_VSCREENINFO, &vinfo)) {
+perror("Error reading variable information");
+return 3;
+}

+int screensize = vinfo.yres_virtual * finfo.line_length;
+
+// Map framebuffer to user memory
+char *fbp = (char *)mmap(0, screensize, PROT_READ | PROT_WRITE, MAP_SHARED, fb_fd, 0);
+if ((intptr_t)fbp == -1) {
+perror("Error: failed to map framebuffer device to memory");
+return 4;
+}
+
+// Open the image file (raw RGB data)
+FILE *img = fopen("image.rgb", "rb");
+if (!img) {
+perror("Error: cannot open image file");
+munmap(fbp, screensize);
+close(fb_fd);
+return 5;
+}
+
+// Read image data into framebuffer memory
+fread(fbp, 1, screensize, img);
+
+// Cleanup
+fclose(img);
+munmap(fbp, screensize);
+close(fb_fd);
+
+return 0;
+
+}
+*/
 vx_status writeMosaicOutput(char* file_name, vx_image out_img)
 {
     vx_status status;
@@ -194,6 +262,8 @@ vx_status writeMosaicOutput(char* file_name, vx_image out_img)
                 data_ptr += image_addr.stride_y;
             }

+
+
```

```
             if(num_bytes != (img_width*img_height))
             {
                 printf("Luma bytes written = %d, expected = %d\n", num_bytes,
img_width*img_height);
@@ -224,6 +294,7 @@ vx_status writeMosaicOutput(char* file_name, vx_image out_img)
                 printf("CbCr bytes written = %d, expected = %d\n", num_bytes,
img_width*img_height/2);
             }

+
             vxUnmapImagePatch(out_img, map_id);

         }
diff --git a/apps/dl_demos/app_tidl_avp/concerto.mak b/apps/dl_demos/app_tidl_avp/concerto.mak
index fab60bc..b4b4677 100644
--- a/apps/dl_demos/app_tidl_avp/concerto.mak
+++ b/apps/dl_demos/app_tidl_avp/concerto.mak
@@ -3,7 +3,7 @@ ifeq ($(TARGET_CPU),$(filter $(TARGET_CPU), x86_64 A72 A53))
 include $(PRELUDE)

 TARGET      := vx_app_tidl_avp
-CSOURCES    := main.c avp_scaler_module.c avp_pre_proc_module.c avp_tidl_module.c
avp_post_proc_module.c fisheye_angle_table.c avp_img_mosaic_module.c avp_draw_detections_module.c
avp_display_module.c
+CSOURCES    := main.c avp_scaler_module.c avp_pre_proc_module.c avp_tidl_module.c
avp_post_proc_module.c fisheye_angle_table.c avp_img_mosaic_module.c avp_draw_detections_module.c

 ifeq ($(HOST_COMPILER),GCC_LINUX)
 CFLAGS += -Wno-unused-function
diff --git a/apps/dl_demos/app_tidl_avp/main.c b/apps/dl_demos/app_tidl_avp/main.c
index 9d23faf..5b0a673 100644
--- a/apps/dl_demos/app_tidl_avp/main.c
+++ b/apps/dl_demos/app_tidl_avp/main.c
@@ -78,7 +78,6 @@
 #include "avp_post_proc_module.h"
 #include "avp_draw_detections_module.h"
 #include "avp_img_mosaic_module.h"
-#include "avp_display_module.h"
 #include "avp_test.h"

 #ifndef x86_64
@@ -108,8 +107,6 @@ typedef struct {

     ImgMosaicObj imgMosaicObj;

-    DisplayObj displayObj;
-
     vx_char input_file_path[APP_MAX_FILE_PATH];
     vx_char output_file_path[APP_MAX_FILE_PATH];
     vx_char input_file_list[APP_MAX_FILE_PATH];
@@ -188,9 +185,7 @@ static void app_update_param_set(AppObj *obj);
 static void add_graph_parameter_by_node_index(vx_graph graph, vx_node node, vx_uint32
node_parameter_index);
 static void app_pipeline_params_defaults(AppObj *obj);
 static void app_find_object_array_index(vx_object_array object_array[], vx_reference ref, vx_int32
array_size, vx_int32 *array_idx);
-#ifndef x86_64
-static void app_draw_graphics(Draw2D_Handle *handle, Draw2D_BufInfo *draw2dBufInfo, uint32_t
update_type);
-#endif
+
 #ifdef AVP_ENABLE_PIPELINE_FLOW
 static vx_status app_run_graph_for_one_frame_pipeline(AppObj *obj, vx_int32 frame_id);
 #else
@@ -228,7 +223,7 @@ static void app_run_task(void *app_var)
     AppObj *obj = (AppObj *)app_var;
     vx_status status = VX_SUCCESS;

-    while(!obj->stop_task && (status == VX_SUCCESS))
+    while((status == VX_SUCCESS))
     {
         status = app_run_graph(obj);
     }
@@ -626,15 +621,6 @@ static void app_parse_cfg_file(AppObj *obj, vx_char *cfg_file_name)
                 }
             }
             else
-            if(strcmp(token, "display_option")==0)
```

```
-            {
-                token = strtok(NULL, s);
-                if(token != NULL)
-                {
-                    obj->displayObj.display_option = atoi(token);
-                }
-            }
-            else
             if(strcmp(token, "delay_in_msecs")==0)
             {
                 token = strtok(NULL, s);
@@ -718,7 +704,6 @@ static void app_parse_cfg_file(AppObj *obj, vx_char *cfg_file_name)

     if (obj->test_mode == 1)
     {
-        obj->displayObj.display_option = 1;
         obj->is_interactive = 0;
         obj->num_iterations = 1;
         /* if testing, just run the number of frames that are found in the expected
@@ -770,7 +755,6 @@ static void app_parse_cmd_line_args(AppObj *obj, vx_int32 argc, vx_char *argv[])
     if (set_test_mode == vx_true_e)
     {
         obj->test_mode = 1;
-        obj->displayObj.display_option = 1;
         obj->is_interactive = 0;
         obj->num_iterations = 1;
         /* if testing, just run the number of frames that are found in the expected
@@ -780,7 +764,6 @@ static void app_parse_cmd_line_args(AppObj *obj, vx_int32 argc, vx_char *argv[])
     }

     #ifdef x86_64
-    obj->displayObj.display_option = 0;
     obj->is_interactive = 0;
     #endif

@@ -876,7 +859,6 @@ static void add_graph_parameter_by_node_index(vx_graph graph, vx_node node, vx_u
 static vx_status app_init(AppObj *obj)
 {
     vx_status status = VX_SUCCESS;
-    app_grpx_init_prms_t grpx_prms;

     /* Create OpenVx Context */
     obj->context = vxCreateContext();
@@ -994,21 +976,10 @@ static vx_status app_init(AppObj *obj)
     {
         status = app_init_img_mosaic(obj->context, &obj->imgMosaicObj, AVP_BUFFER_Q_DEPTH);
     }
-    if(status == VX_SUCCESS)
-    {
-        status = app_init_display(obj->context, &obj->displayObj, "display_obj");
-    }
+
     appPerfPointSetName(&obj->total_perf , "TOTAL");
     appPerfPointSetName(&obj->fileio_perf, "FILEIO");

-    #ifndef x86_64
-    if(obj->displayObj.display_option == 1)
-    {
-        appGrpxInitParamsInit(&grpx_prms, obj->context);
-        grpx_prms.draw_callback = app_draw_graphics;
-        appGrpxInit(&grpx_prms);
-    }
-    #endif

     return status;
 }
@@ -1041,14 +1012,6 @@ static void app_deinit(AppObj *obj)

     app_deinit_img_mosaic(&obj->imgMosaicObj, AVP_BUFFER_Q_DEPTH);

-    app_deinit_display(&obj->displayObj);
-
-    #ifndef x86_64
-    if(obj->displayObj.display_option == 1)
-    {
-        appGrpxDeInit();
-    }
-    #endif
```

```
     tivxTIDLUnLoadKernels(obj->context);
     tivxImgProcUnLoadKernels(obj->context);
@@ -1086,7 +1049,6 @@ static void app_delete_graph(AppObj *obj)

     app_delete_img_mosaic(&obj->imgMosaicObj);

-    app_delete_display(&obj->displayObj);

     vxReleaseGraph(&obj->graph);
 }
@@ -1180,10 +1142,6 @@ static vx_status app_create_graph(AppObj *obj)

     app_create_graph_img_mosaic(obj->graph, &obj->imgMosaicObj, NULL);

-    if(status == VX_SUCCESS)
-    {
-        status = app_create_graph_display(obj->graph, &obj->displayObj, obj-
>imgMosaicObj.output_image[0]);
-    }

 #ifdef AVP_ENABLE_PIPELINE_FLOW
     /* Scalar Node - input is in Index 0 */
@@ -1547,8 +1505,11 @@ static vx_status app_run_graph_for_one_frame_pipeline(AppObj *obj, vx_int32
fram
                 APP_PRINTF("App Writing Outputs Start...\n");

                 snprintf(output_file_name, APP_MAX_FILE_PATH, "%s/
mosaic_output_%010d_1920x1080.yuv", obj->output_file_path, (frame_id - AVP_BUFFER_Q_DEPTH));
+                // printf("output_file_name=%s\n",output_file_name);
                 status = writeMosaicOutput(output_file_name, mosaic_output_image);
-
+                /*Kangjia get perception algorithm result*/
+                // printf("-------------------------------------\n");
+
                 APP_PRINTF("App Writing Outputs Done!\n");
             }
             /* Enqueue output */
@@ -1629,10 +1590,10 @@ static vx_status app_run_graph(AppObj *obj)
             APP_PRINTF("app_avp: Frame ID %d of %d ... Done.\n", frame_id, obj->start_frame + obj-
>num_frames);

             /* user asked to stop processing */
-            if((obj->stop_task) || (status == VX_FAILURE))
-            {
-                break;
-            }
+            //if((obj->stop_task) || (status == VX_FAILURE))
+            //{
+            //    break;
+            //}
         }
         printf("app_avp: Iteration %d of %d ... Done.\n", x, obj->num_iterations);
         appPerfPointPrintFPS(&obj->total_perf);
@@ -1644,17 +1605,17 @@ static vx_status app_run_graph(AppObj *obj)
         }

         /* user asked to stop processing */
-        if((obj->stop_task) || (status == VX_FAILURE))
-        {
-          break;
-        }
+        //if((obj->stop_task) || (status == VX_FAILURE))
+        //{
+        //  break;
+        //}
     }

 #ifdef AVP_ENABLE_PIPELINE_FLOW
     vxWaitGraph(obj->graph);
 #endif

-    obj->stop_task = 1;
+    //obj->stop_task = 1;

     return status;
 }
@@ -1838,8 +1799,8 @@ static void set_img_mosaic_defaults(AppObj *obj, ImgMosaicObj *imgMosaicObj)
```

```
 {
     vx_int32 idx = 0;
     vx_int32 in = 0;
-    imgMosaicObj->out_width    = 1920;
-    imgMosaicObj->out_height   = 1080;
+    imgMosaicObj->out_width    = 800;
+    imgMosaicObj->out_height   = 600;
     imgMosaicObj->num_inputs   = obj->enable_psd + obj->enable_vd + obj->enable_sem_seg;

     tivxImgMosaicParamsSetDefaults(&imgMosaicObj->params);
@@ -1848,8 +1809,8 @@ static void set_img_mosaic_defaults(AppObj *obj, ImgMosaicObj *imgMosaicObj)
     /* Right camera - PSD output */
     if(obj->enable_psd == 1)
     {
-        imgMosaicObj->params.windows[idx].startX  = 840;
-        imgMosaicObj->params.windows[idx].startY  = 200;
+        imgMosaicObj->params.windows[idx].startX  = 100;
+        imgMosaicObj->params.windows[idx].startY  = 50;
         imgMosaicObj->params.windows[idx].width   = 512;
         imgMosaicObj->params.windows[idx].height  = 512;
         imgMosaicObj->params.windows[idx].input_select   = in++;
@@ -1860,8 +1821,8 @@ static void set_img_mosaic_defaults(AppObj *obj, ImgMosaicObj *imgMosaicObj)
     /* Right camera - PSD output */
     if(obj->enable_vd == 1)
     {
-        imgMosaicObj->params.windows[idx].startX  = 1380;
-        imgMosaicObj->params.windows[idx].startY  = 200;
+        imgMosaicObj->params.windows[idx].startX  = 100;
+        imgMosaicObj->params.windows[idx].startY  = 50;
         imgMosaicObj->params.windows[idx].width   = 512;
         imgMosaicObj->params.windows[idx].height  = 512;
         imgMosaicObj->params.windows[idx].input_select   = in++;
@@ -1872,8 +1833,8 @@ static void set_img_mosaic_defaults(AppObj *obj, ImgMosaicObj *imgMosaicObj)
     /* Front camera - semantic segmentation output */
     if(obj->enable_sem_seg == 1)
     {
-        imgMosaicObj->params.windows[idx].startX  = 40;
-        imgMosaicObj->params.windows[idx].startY  = 250;
+        imgMosaicObj->params.windows[idx].startX  =20;
+        imgMosaicObj->params.windows[idx].startY  = 100;
         imgMosaicObj->params.windows[idx].width   = 768;
         imgMosaicObj->params.windows[idx].height  = 384;
         imgMosaicObj->params.windows[idx].input_select   = in++;
@@ -1887,10 +1848,6 @@ static void set_img_mosaic_defaults(AppObj *obj, ImgMosaicObj *imgMosaicObj)
     imgMosaicObj->params.clear_count  = 4;
 }

-static void set_display_defaults(DisplayObj *displayObj)
-{
-    displayObj->display_option = 0;
-}

 static void app_pipeline_params_defaults(AppObj *obj)
 {
@@ -1918,7 +1875,6 @@ static void app_default_param_set(AppObj *obj)
     obj->vdDrawDetectionsObj.params.num_classes = 1;
     obj->vdDrawDetectionsObj.params.class_id[0] = 1;

-    set_display_defaults(&obj->displayObj);

     app_pipeline_params_defaults(obj);

@@ -1972,31 +1928,4 @@ static void app_find_object_array_index(vx_object_array object_array[],
vx_refer
         vxReleaseImage(&img_ref);
     }
 }
-#ifndef x86_64
-static void app_draw_graphics(Draw2D_Handle *handle, Draw2D_BufInfo *draw2dBufInfo, uint32_t
update_type)
-{
-    appGrpxDrawDefault(handle, draw2dBufInfo, update_type);
-
-    if(update_type == 0)
-    {
-        Draw2D_FontPrm sHeading;
-        Draw2D_FontPrm sAlgo1;
-        Draw2D_FontPrm sAlgo2;
```

```
-        Draw2D_FontPrm sAlgo3;
-
-        sHeading.fontIdx = 0;
-        Draw2D_drawString(handle, 560, 5, "Analytics for Auto Valet Parking", &sHeading);
-
-        sAlgo1.fontIdx = 2;
-        Draw2D_drawString(handle, 270, 640, "Semantic Segmentation", &sAlgo1);

-        sAlgo2.fontIdx = 2;
-        Draw2D_drawString(handle, 920, 720, "Parking Spot Detection", &sAlgo2);
-
-        sAlgo3.fontIdx = 2;
-        Draw2D_drawString(handle, 1490, 720, "Vehicle Detection", &sAlgo3);
-    }
-
-  return;
-}
-#endif
--
2.34.1
```

3.2.1.2 Linux 补丁

3.2.1.2.1 0001-arm64-dts-ti-k3-j784s4-vision-apps-Re-enable-DSS-for.patch

```
From 1245202b0656ae2d2f52b76951c4f2341c8290fb Mon Sep 17 00:00:00 2001
From: Jared McArthur <j-mcarthur@ti.com>
Date: Tue, 28 Jan 2025 14:04:34 -0600
Subject: [PATCH 1/1] arm64: dts: ti: k3-j784s4-vision-apps: Re-enable DSS for
 Linux

Re-enable DSS within the Linux domain when running vision_apps
applications.

The display driver is usually controlled by the MCU2_0 core when
vision_apps applications are run, and the Linux display driver is
disabled. Revert this behavior.

Signed-off-by: Jared McArthur <j-mcarthur@ti.com>
---
 .../boot/dts/ti/k3-j784s4-vision-apps.dtso    | 20 --------------------
 1 file changed, 20 deletions(-)

diff --git a/arch/arm64/boot/dts/ti/k3-j784s4-vision-apps.dtso b/arch/arm64/boot/dts/ti/k3-j784s4-
vision-apps.dtso
index 83b500405..ba226e32c 100644
--- a/arch/arm64/boot/dts/ti/k3-j784s4-vision-apps.dtso
+++ b/arch/arm64/boot/dts/ti/k3-j784s4-vision-apps.dtso
@@ -10,36 +10,16 @@

 #include <dt-bindings/mux/ti-serdes.h>

-&main_r5fss0_core0_shared_memory_queue_region {
-    status = "disabled";
-};
-
-&main_r5fss0_core0_shared_memory_bufpool_region {
-    status = "disabled";
-};
-
 #include "k3-j784s4-rtos-memory-map.dtsi"

 &main_i2c1 {
     status = "disabled";
 };

-&main_i2c4 {
-    status = "disabled";
-};
-
 &main_i2c5 {
     status = "disabled";
 };

-&dss {
-    status = "disabled";
```

```
-};
-
-&serdes_wiz4 {
-    status = "disabled";
-};
-
 &ti_csi2rx0 {
     status = "disabled";
 };
--
2.34.1
```

3.2.2 主机设置

在主机 PC 上执行以下命令。

安装所有依赖项。

```
# install the PROCESSOR-SDK-RTOS-J784S4
# install the PROCESSOR-SDK-LINUX-J784S4
# download the data set tar files
# download and untar the patch tars
# insert SD card
```

导出构建变量。

```
export PSDKR_PATH=<path-to-rtos-sdk>
export PSDKL_PATH=<path-to-linux-sdk>
export DATA_SET_PATH=<path-to-directory-where-data-sets-are-stored>
export RTOS_PATCHES=<path-to-rtos-patches>
export LINUX_PATCHES=<path-to-linux-patches>
```

设置 PSDK RTOS。

```
# set up PSDK RTOS
cd $PSDKR_PATH
./sdk_builder/scripts/setup_psdk_rtos.sh
```

设置 SD 卡。本示例假设 SD 卡位于 /dev/sdb。

```
umount /dev/sdb?*
cd $PSDKR_PATH
sudo sdk_builder/scripts/mk-linux-card.sh /dev/sdb
./sdk_builder/scripts/install_to_sd_card.sh
cd /media/$USER/rootfs/
mkdir -p opt/vision_apps
cd opt/vision_apps
tar --strip-components=1 -xf $DATA_SET_PATH/psdk_rtos_ti_data_set_09_02_00.tar.gz
tar --strip-components=1 -xf $DATA_SET_PATH/psdk_rtos_ti_data_set_09_02_00_j784s4.tar.gz
sync
```

编辑、构建和安装演示应用程序。

```
# edit and build demo app
cd $PSDKR_PATH/vision_apps
git init
git add -A
git commit -m "SDK 09.02.00.05 release"
git am $RTOS_PATCHES/*.patch
cd ../sdk_builder
./make_sdk.sh
make linux_fs_install_sd
```

编辑、构建和安装设备树。

```
export PATH=$PATH:$PSDKL_PATH/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/aarch64-oe-linux/
cd $PSDKL_PATH/board-support/ti-linux-kernel-6.1.80+gitAUTOINC+2e423244f8-ti
git add -A
git commit -m "SDK 09.02.00.05 release"
```

```
git am $LINUX_PATCHES/*.patch
make ARCH=arm64 CROSS_COMPILE=aarch64-oe-linux- defconfig ti_arm64_prune.config
make ARCH=arm64 CROSS_COMPILE=aarch64-oe-linux- DTC_FLAGS=-@ ti/k3-j784s4-vision-apps.dtbo
sudo mv /media/$USER/rootfs/boot/dtb/ti/k3-j784s4-vision-apps.dtbo /media/$USER/rootfs/
boot/dtb/ti/k3-j784s4-vision-apps.dtbo.old
sudo cp arch/arm64/boot/dts/ti/k3-j784s4-vision-apps.dtbo /media/$USER/rootfs/boot/dtb/ti/
```

3.2.3 目标设置

以下命令在目标上执行。

```
cd /opt/vision_apps
source ./vision_apps_init.sh
./run_app_tidl_avp.sh
```

3.2.4 重新创建

以下照片是在 AVP 演示与 kmstest 一起运行时拍摄的。

```
systemctl stop weston
kmstest & ./run_app_tidl_avp.sh
```

运行 AVP 演示时，屏幕出现同步丢失。这会导致显示器出现不可预测的伪影。

图 3-1. 运行 kmstest 和 AVP 演示时出现同步丢失问题

3.3 调试 QoS

3.3.1 CPTracer

CCS（和 Lauterbach）提供了一个名为 CPTracer 的工具，可让开发人员分析系统内的流量。在本例中，我们将对 DDR 流量进行分析。

3.3.1.1 设置

1. 按照第 7.4 节中的步骤来设置 CCS。使用在 Processor SDK RTOS 的 A72 (Linux/QNX) 上运行的 HLOS 进行调试 。

> **小心**
>
> 此处提到的 PSDK RTOS 的具体版本是： PSDK RTOS J784S4 10.01.00

2. 可以从 SoC Analysis 选项卡打开流量分析窗口。

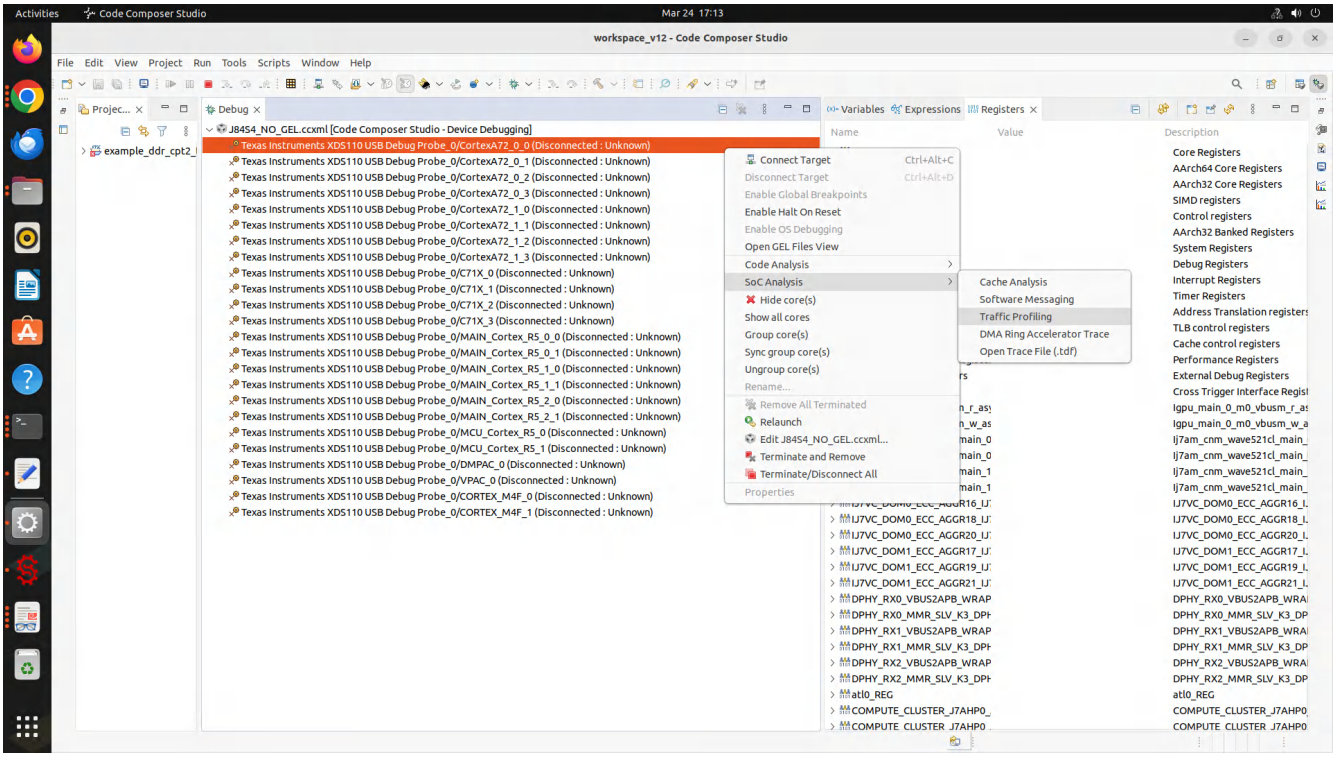

图 3-2. CCS 中的流量分析选择

3. 这是打开 CPTracer 时应出现的窗口：

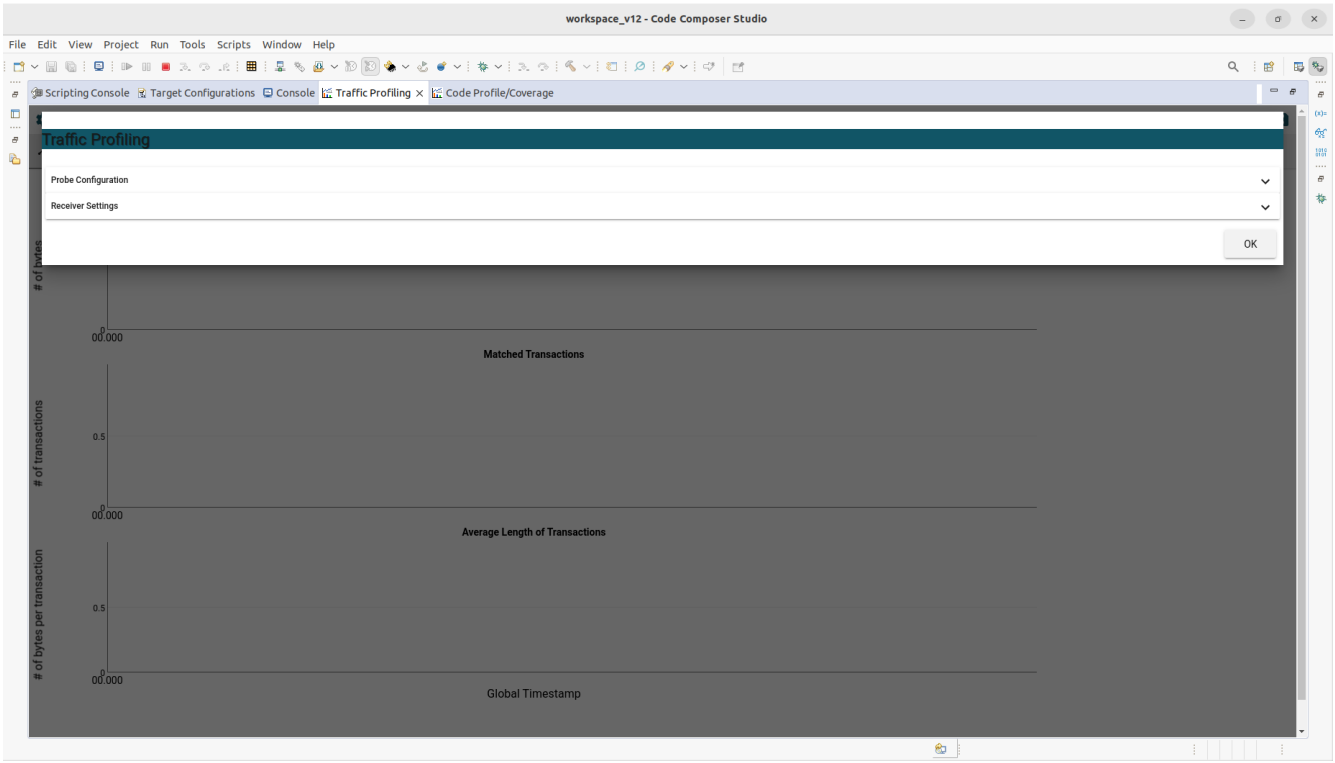

图 3-3. CCS 中的 CPTracer 设置

4. 有许多过滤器可用于调整要分析的事务，并且可通过单击相应启动器的齿轮图标对其进行编辑。

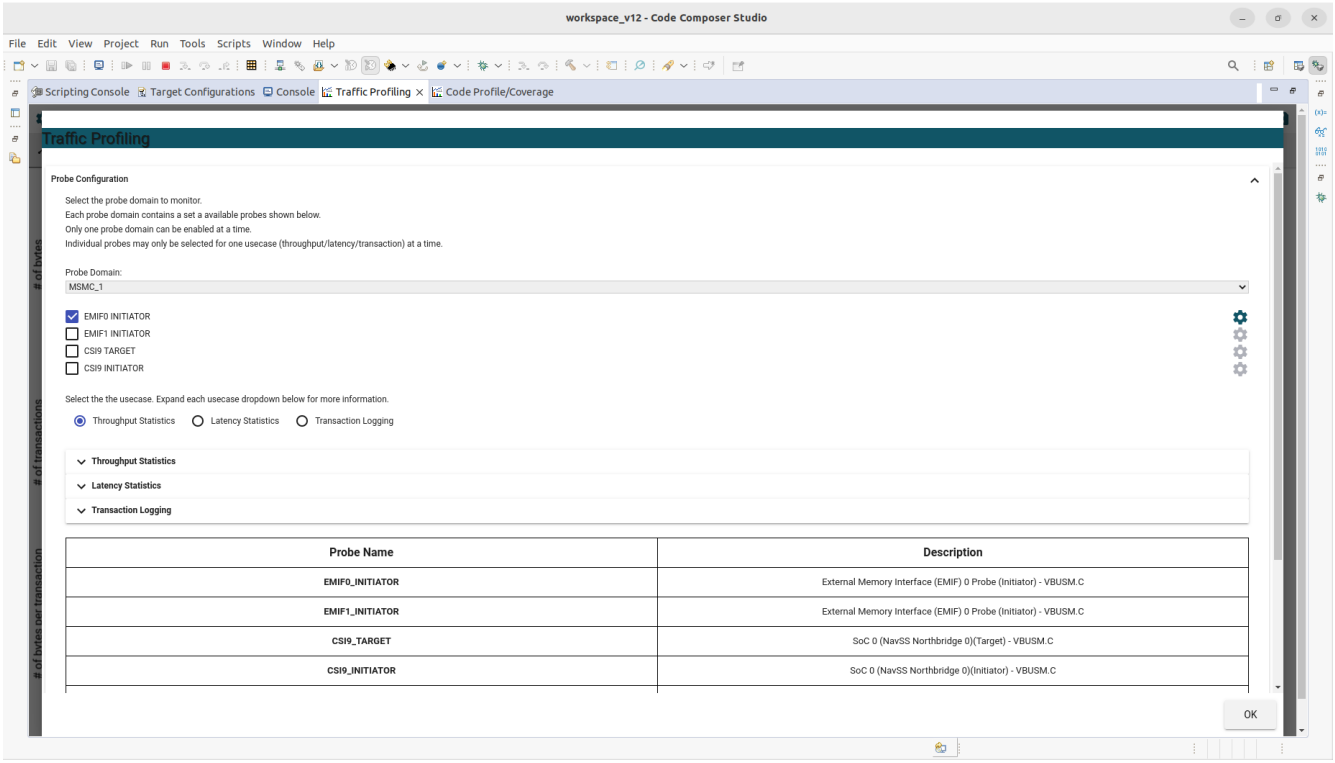

图 3-4. CCS 中的 CPTracer 过滤器

3.3.1.2 分析吞吐量

CPTracer 提供了描述一个周期时间内事务总吞吐量的方法。以下是导出 csv 时的输出：

- **主 ID**：进行事务的启动器的 ID
- **主名称**：启动器的名称（与主 ID 相对应）
- **数据消息**：行说明（包含以时钟周期为单位的周期时间）
- **全局时间戳**：采样窗口关闭的时间（基于 GTC（200MHz 时钟））
- **跟踪状态**：记录跟踪开始和结束的时间
- **字节事务**：探测中观察到的与设置的过滤器匹配的总字节数（在一个周期时间内发送）
- **匹配**：探测中观察到的与设置的过滤器匹配的事务总数（在一个周期时间内）
- **平均长度**：探测中观察到的与设置的过滤器匹配的每个事务的平均字节数（在一个周期时间内发送）

3.3.1.3 分析延迟

CPTracer 还提供了分析一段时间内事务延迟的方法。以下是导出 csv 时的输出：

- **主 ID**：进行事务的启动器的 ID
- **主名称**：启动器的名称（与主 ID 相对应）
- **数据消息**：行说明（包含以时钟周期为单位的周期时间）
- **全局时间戳**：采样窗口关闭的时间（基于 GTC（200MHz 时钟））
- **跟踪状态**：记录跟踪开始和结束的时间
- **已跟踪**：启动器在一段时间内的事务总数
- **匹配的事务**：探测中观察到的与设置的过滤器匹配的事务总数（在一个周期时间内）
- **最长等待时间**：探测中察到的与设置的过滤器匹配的单个事务（在一个周期时间内）的最大延迟测量值
- **总等待时间**：探测中察到的与设置的过滤器匹配的总延迟（所有延迟测量值加在一起）（在一个周期时间内）
- **信用等待时间**：探测中察到的与设置的过滤器匹配的基于信用的总线的总信用延迟（在一个周期时间内）

3.3.1.4 分析事务

CPTracer 提供有关观察到的事务的详细信息。以下是导出 csv 时的输出（以下是精简后的相关列的列表）：

- **路由 ID**：事务的路由 ID
- **字节计数**：事务的突发大小
- **优先级**：事务的优先级
- **QOS**：事务的服务质量
- **顺序 ID**：事务的顺序 ID

备注

可在 CPTracer 文档的 Advanced Probe Filters（高级探头过滤器）部分找到完整的列的列表

3.3.1.5 分析相关路由

我们正在查看的问题是由于 C7x 干扰了 DSS，因此我们需要对 C7x 和 DSS 路由进行分析。

DSS 和 C7x 事务的路由 ID 如下所示：

表 3-2. 相关的启动器和路由 ID

启动器	路由 ID
C7x_1 内核	0x20
C7x_1 DRU0	0x21
C7x_1 DRU1	0x22
C7x_1 CMMU	0x23
C7x_2 内核	0x24
C7x_2 DRU0	0x25
C7x_2 DRU1	0x26

表 3-2. 相关的启动器和路由 ID （续）

启动器	路由 ID
C7x_2 CMMU	0x27
C7x_3 内核	0x28
C7x_3 DRU0	0x29
C7x_3 DRU1	0x2A
C7x_3 CMMU	0x2B
C7x_4 内核	0x2C
C7x_4 DRU0	0x2D
C7x_4 DRU1	0x2E
C7x_4 CMMU	0x2F
DSS_INST0_VBUSM_DMA	0xA20
DSS_INST0_VBUSM_FBDC	0xA21

备注

可以在 TDA4VH TRM 的附录中找到路由 ID

CCS 版本的 CPTracer 在 MSMC_1 探针域中仅包含 EMIF0 和 EMIF1（缺少 EMIF2 和 3），但可以通过将一个 EMIF 的吞吐量乘以 4 来估算总吞吐量。

3.3.1.6 分析 DSS 吞吐量

用于仅分析 DSS 事务的过滤器如下：

- **路由 ID 值**：0xA20
- **路由 ID 掩码**：0xFFE
- **采样窗口**：0x4000

仅对 EMIF0 进行了分析。

每帧发送的总字节通过以下公式计算：**dss-frame-thru-calc.py**

```
# Author: Jared McArthur

import csv
import matplotlib.pyplot as plt
from argparse import ArgumentParser
from textwrap import dedent

def main():
    total_bytes = []
    time_stamps = []

    parser = ArgumentParser(prog="dss-frame-thru-calc.py")
    parser.add_argument("file", type=str)
    parser.add_argument("frame_start", type=float)
    parser.add_argument("frame_end", type=float)

    args = parser.parse_args()
    start = args.frame_start
    end = args.frame_end

    with open(args.file, "r") as csvfile:
        data = csv.DictReader(csvfile)

        first_stamp = 0

        for row in data:
            if row.get("Master ID") != "":
                total_byte = int(row.get("Byte Transactions"), base=16)
```

```
                time_stamp = int(row.get("Global Timestamp"), base=16)

                if len(time_stamps) == 0:
                    first_stamp = time_stamp

                total_bytes.append(total_byte)
                time_stamps.append((time_stamp - first_stamp) / (1000000000 / 5))

    stamps_len = len(time_stamps)
    for index, stamp in enumerate(reversed(time_stamps)):
        if stamp < start or stamp > end:
            time_stamps.pop(stamps_len - 1 - index)
            total_bytes.pop(stamps_len - 1 - index)

    bytes_in_frame = 0
    for val in total_bytes:
        bytes_in_frame += val

    print(dedent(f"""
        Num periods in segment: {len(time_stamps)}
        Time elapsed in segment: {time_stamps[-1] - time_stamps[0]}
        Bytes sent in segment: {bytes_in_frame}"""))

    plt.plot(time_stamps, total_bytes)
    plt.show()

if __name__ == "__main__":
    main()
```

3.3.1.6.1 理论 DSS 吞吐量

计算 **3840x2160@30fps XR32-888** 显示屏的理论 DSS 吞吐量（**265420800** 位/帧）。

表 3-3. 理论 DSS 吞吐量

参数	值
高度	3840
宽度	2160
fps	30
位/像素	32
位/帧	265420800
Mib/帧	253.125
数据速率 (bps)	7962624000
数据速率 (Mibps)	7593.75

3.3.1.6.2 正常 DSS 吞吐量

下图显示了未运行 AVP 演示时 DSS 事务的吞吐量。

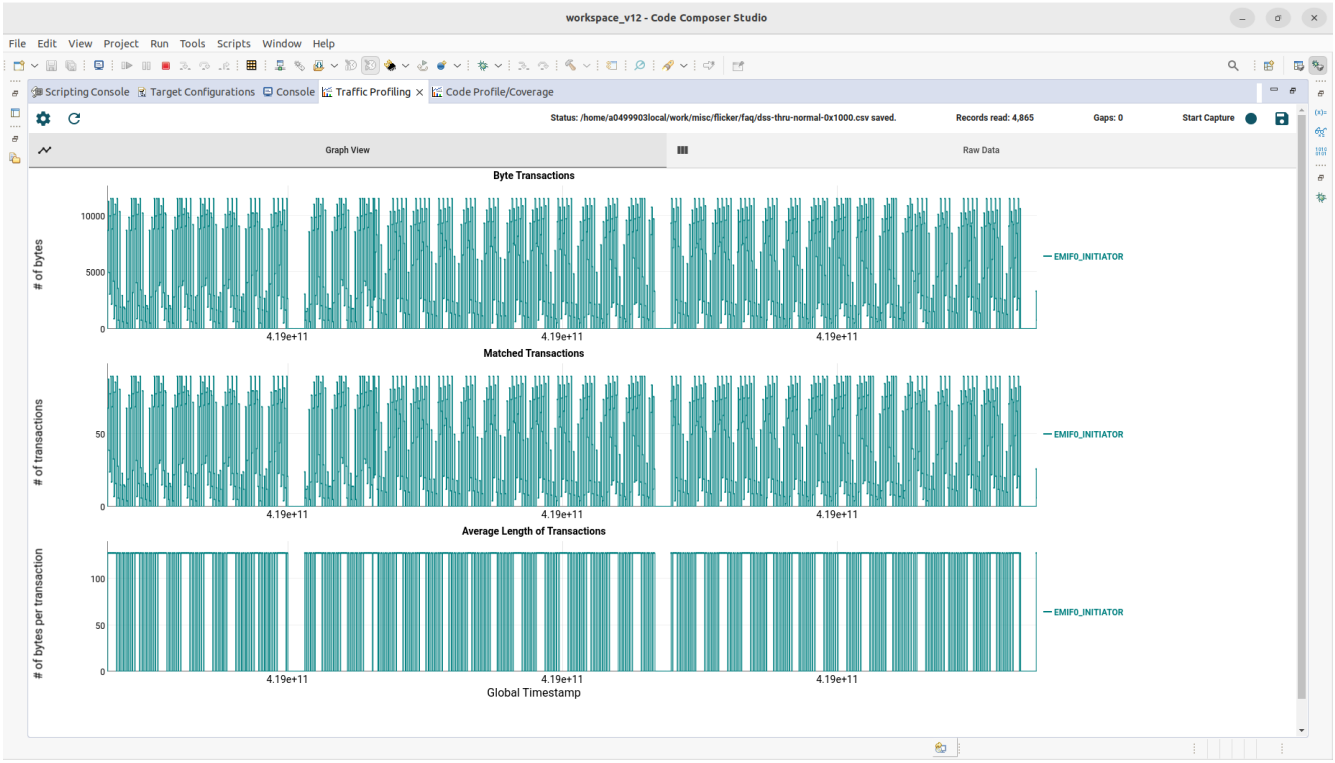

图 3-5. 正常 DSS 吞吐量

吞吐量测量值是通过将一个帧内的所有周期吞吐量相加而得出的。通过保存吞吐量 CSV 文件并利用 python 脚本来汇总测量值。

表 3-4. 正常测量的 DSS 吞吐量

	1 个 EMIF 的 DSS 吞吐量	更正了 4 个 EMIF 的吞吐量
字节/帧	8294400	33177600
位/字节	8	
fps	30	
测量的 EMIF 数量	1	4
位/帧	66355200	265420800
Mib/帧	63.28125	253.125
数据速率 (bps)	1990656000	7962624000
数据速率 (Mibps)	1898.4375	7593.75

单个帧的总吞吐量与 **3840x2160@30fps** 屏幕（**66355200** 位/帧）的 1/4 预期吞吐量相匹配。仅对四个 EMIF 中的一个进行了分析，因此这个结果符合预期。校正测量值后，吞吐量等于理论吞吐量（**265420800** 位/帧）。

3.3.1.6.3 AVP 演示运行时的 DSS 吞吐量

下图显示了 AVP 演示运行时 DSS 事务的吞吐量。

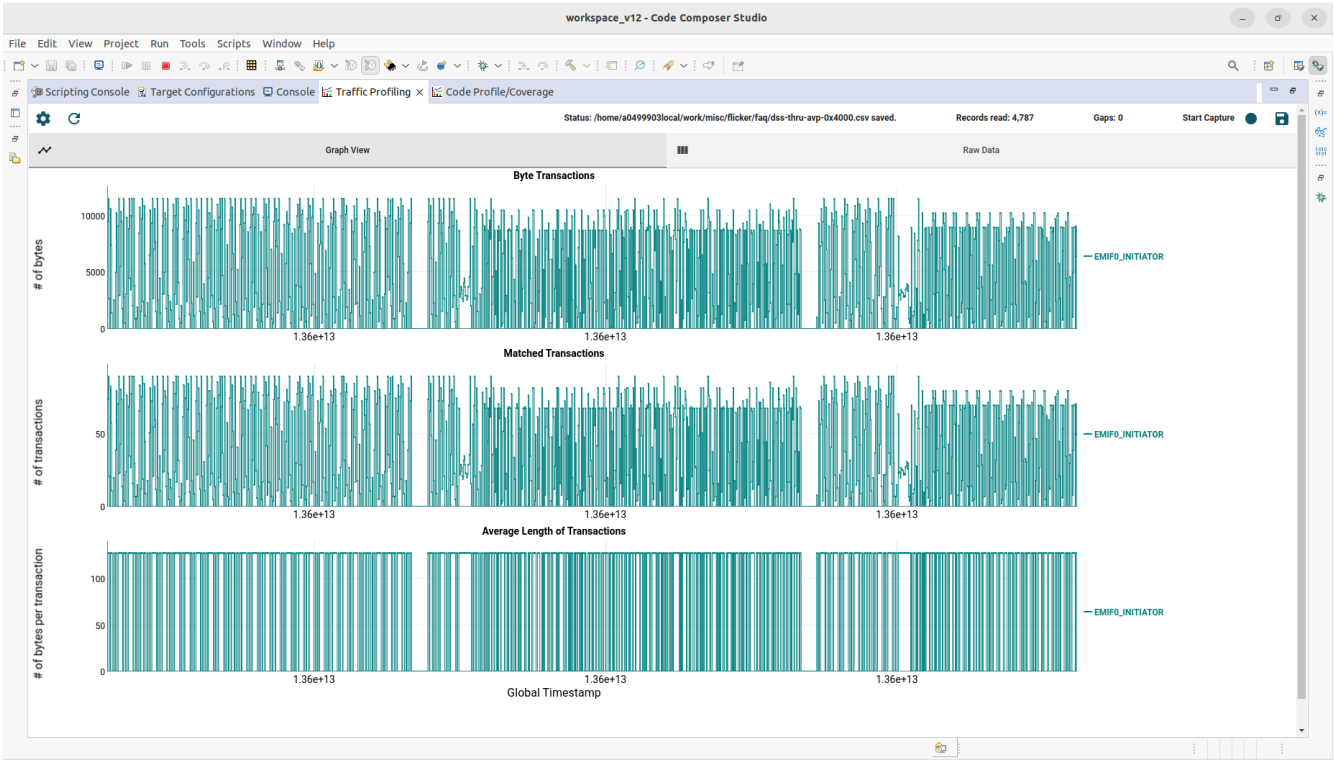

图 3-6. AVP 演示 DSS 吞吐量

表 3-5. 使用 AVP 演示时测量到的 DSS 吞吐量

	1 个 EMIF 的 DSS 吞吐量	更正了 4 个 EMIF 的吞吐量
字节/帧	8257536	33030144
位/字节	8	
fps	30	
测量的 EMIF 数量	1	4
位/帧	66060288	264241152
Mib/帧	63	252
数据速率 (bps)	1981808640	7927234560
数据速率 (Mibps)	1890	7560

总吞吐量低于显示 **3840x2160@30fps** 所需的吞吐量。这是导致同步丢失的原因。一个帧中仅发送 **264241152** 位，而不是发送 **265420800** 位。

3.3.1.7 分析 DSS 延迟

假设 DSS 事务处于停滞状态，导致总延迟大幅上升。为了验证该假设，我们在运行和不运行 AVP 演示的情况下分别分析 DSS 延迟。

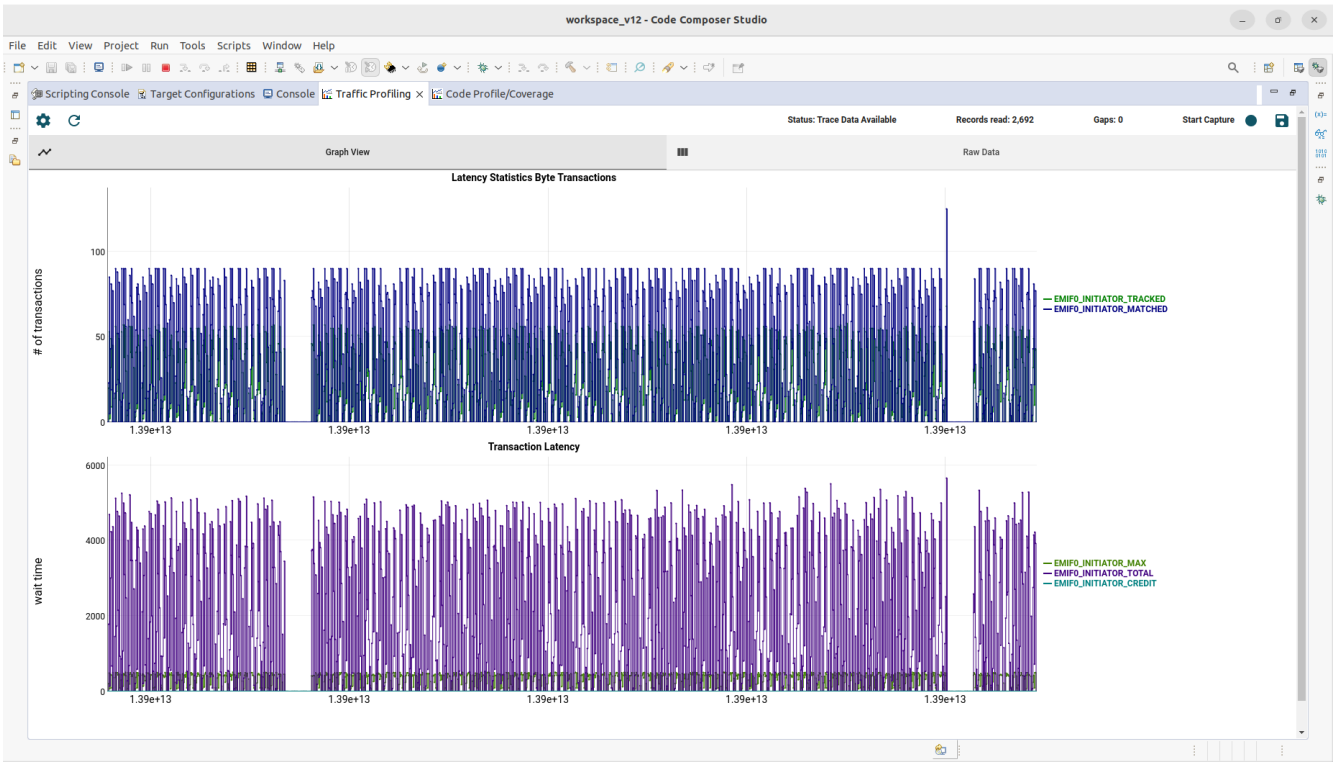

图 3-7. 正常 DSS 延迟

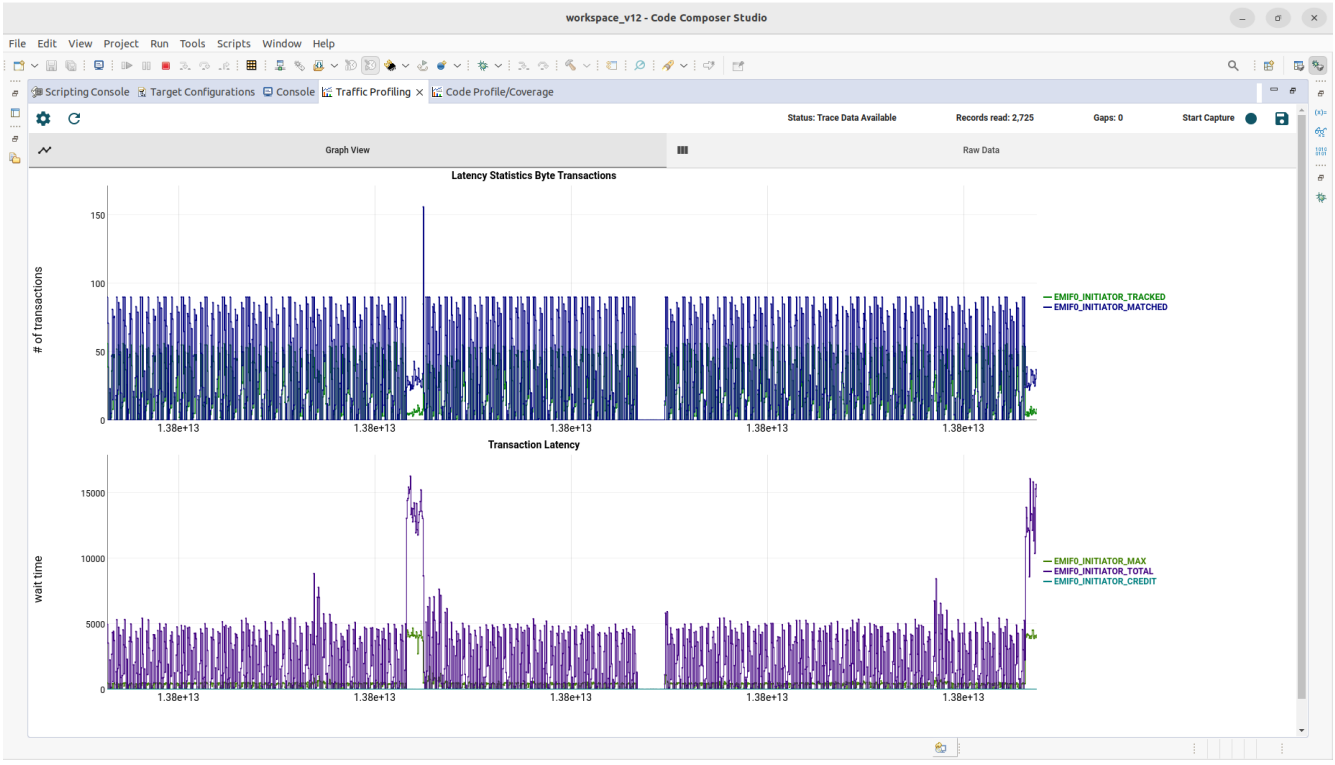

图 3-8. AVP 演示 DSS 延迟

从延迟图可以明显看出，DSS 事务正处于停滞状态。我们现在可以分析 C7x 事务并尝试缩小停滞源的范围。

3.3.1.8 分析 C7x 吞吐量

分析每个单独 C7x 内核的吞吐量可以发现导致 DSS 停滞的确切原因。

高吞吐量和高事务量的的区可能会导致停滞。

表 3-6 包含过滤单独 C7x 内核所需的设置。

表 3-6. C7x 路由 ID 值和掩码

C7x 内核	路由 ID	路由 ID 值	路由 ID 掩码
所有内核	0x20 至 0x2F	0x020	0xFF0
1	0x20 至 0x23	0x020	0xFFC
2	0x24 至 0x27	0x024	0xFFC
3	0x28 至 0x2B	0x028	0xFFC
4	0x2C 至 0x2F	0x02C	0xFFC

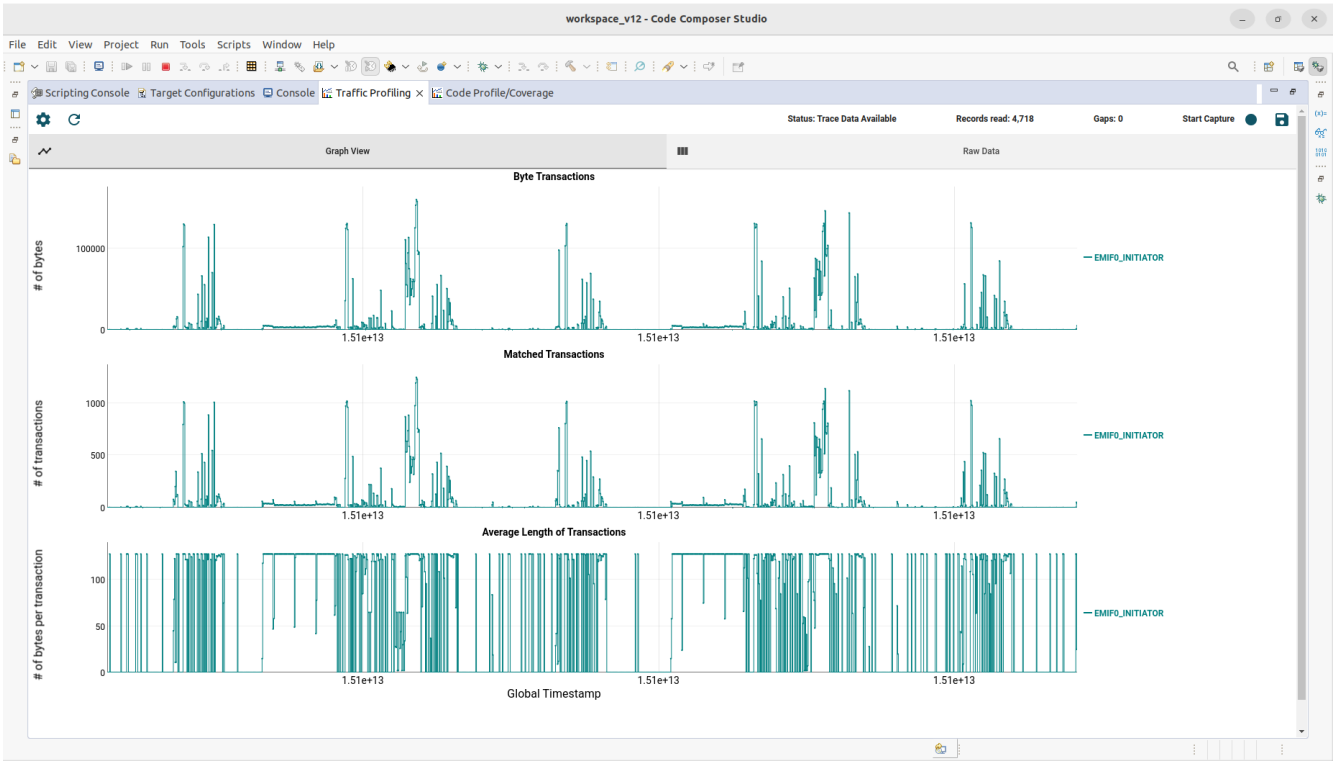

图 3-9. 所有 C7x 吞吐量

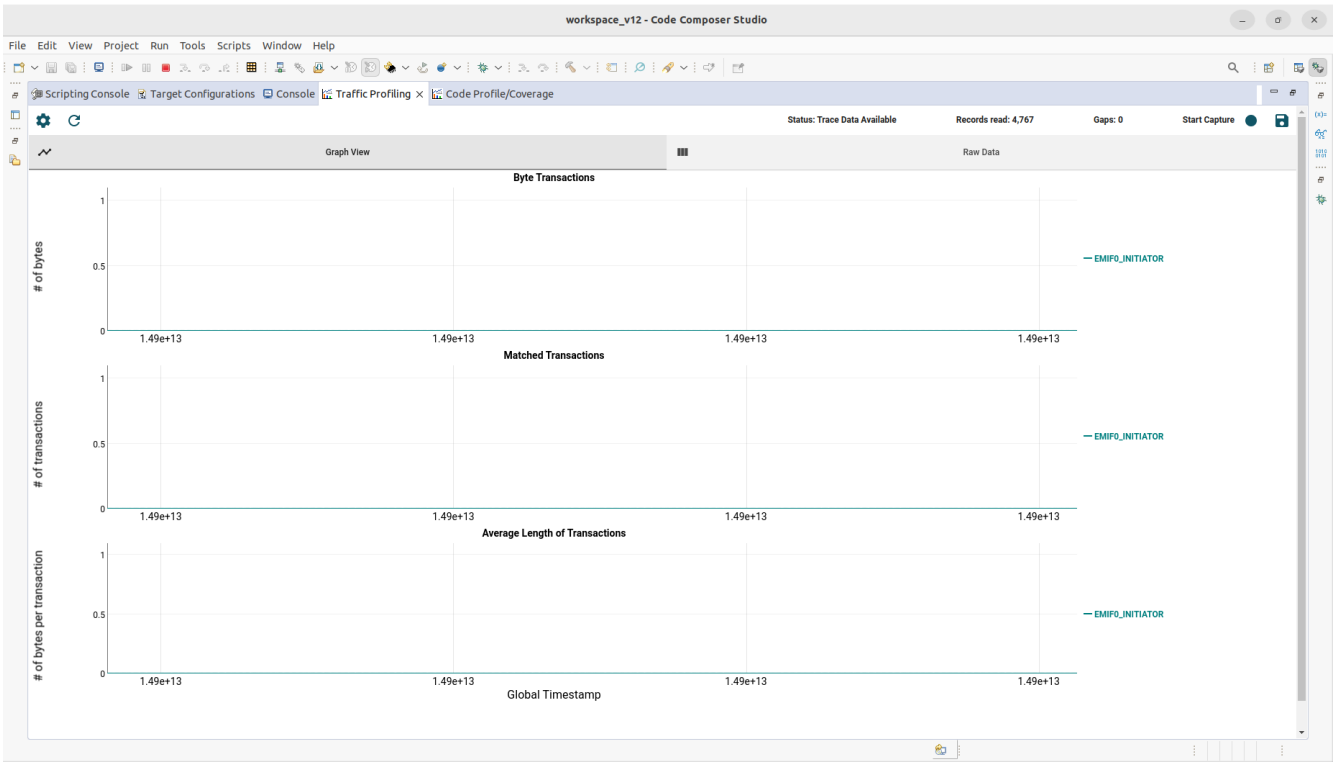

图 3-10. C7x_1 吞吐量

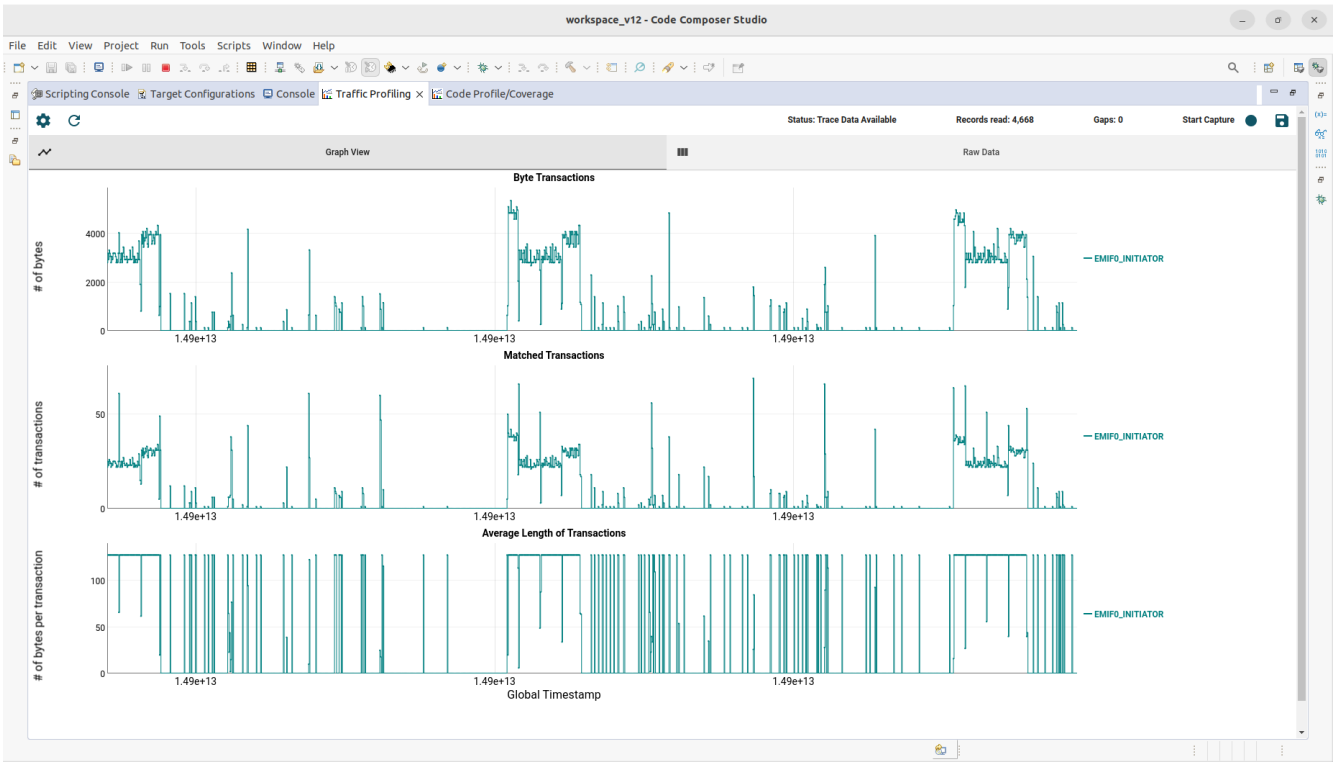

图 3-11. C7x_2 吞吐量

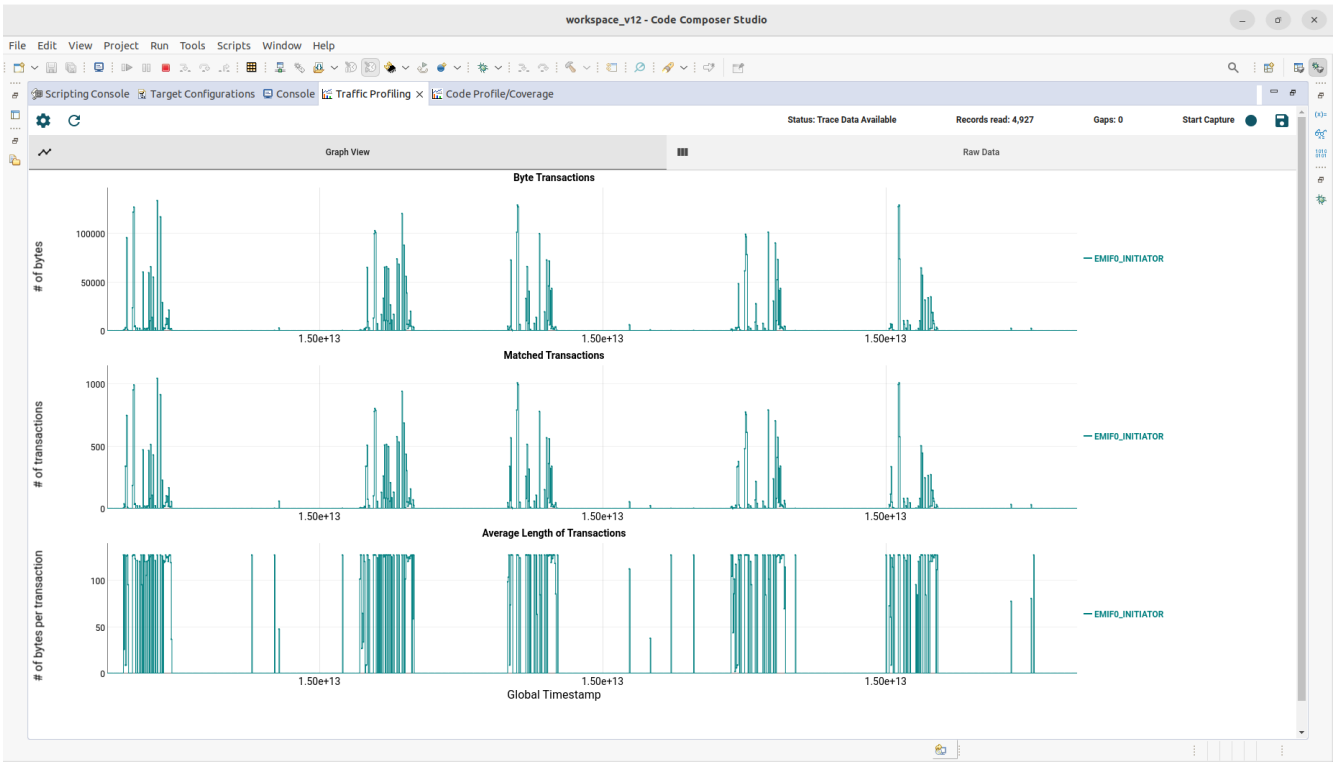

图 3-12. C7x_3 吞吐量

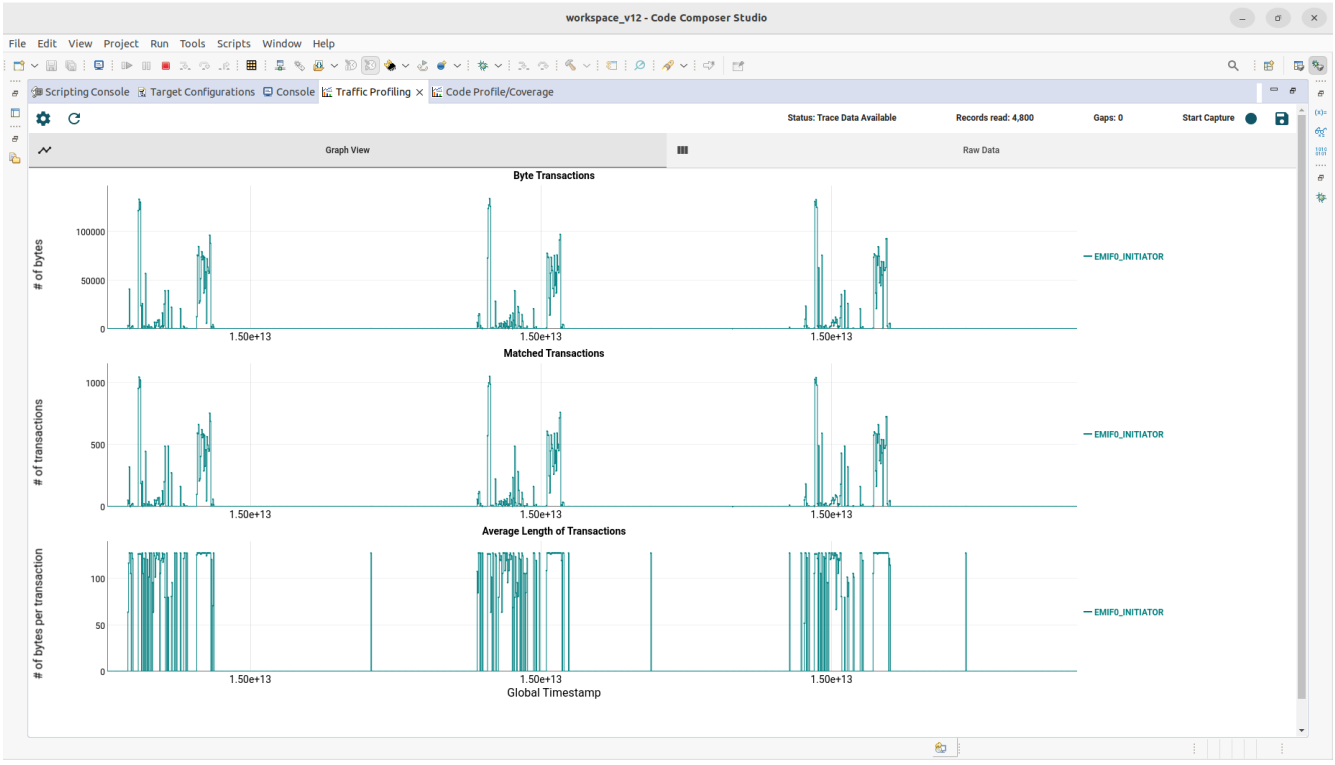

图 3-13. C7x_4 吞吐量

观察以上各图发现，似乎是 C7x_4 导致 DSS 中出现停滞。在 DSS 吞吐量模式下，高吞吐量区段看起来与之成反比。

由于从 C7x_4 发送的高吞吐量事务似乎导致 DSS 停滞，因此我们来仔细研究一下 C7x_4。

3.3.1.9 分析 C7x 吞吐量与 DSS 延迟

很遗憾，CCS 的 CPTracer 不允许同时显示事务吞吐量和事务延迟。但是，Lauterbach 的 CPTracer 可以。

您可以分析 EMIF0 上的吞吐量和 EMIF1 上的延迟，从而同时查看吞吐量和延迟。

借助此方法，我们可以验证在 DSS 延迟较高且 DSS 吞吐量受限的情况下，具体是哪一条 C7x_4 路由路径具有非零吞吐量。

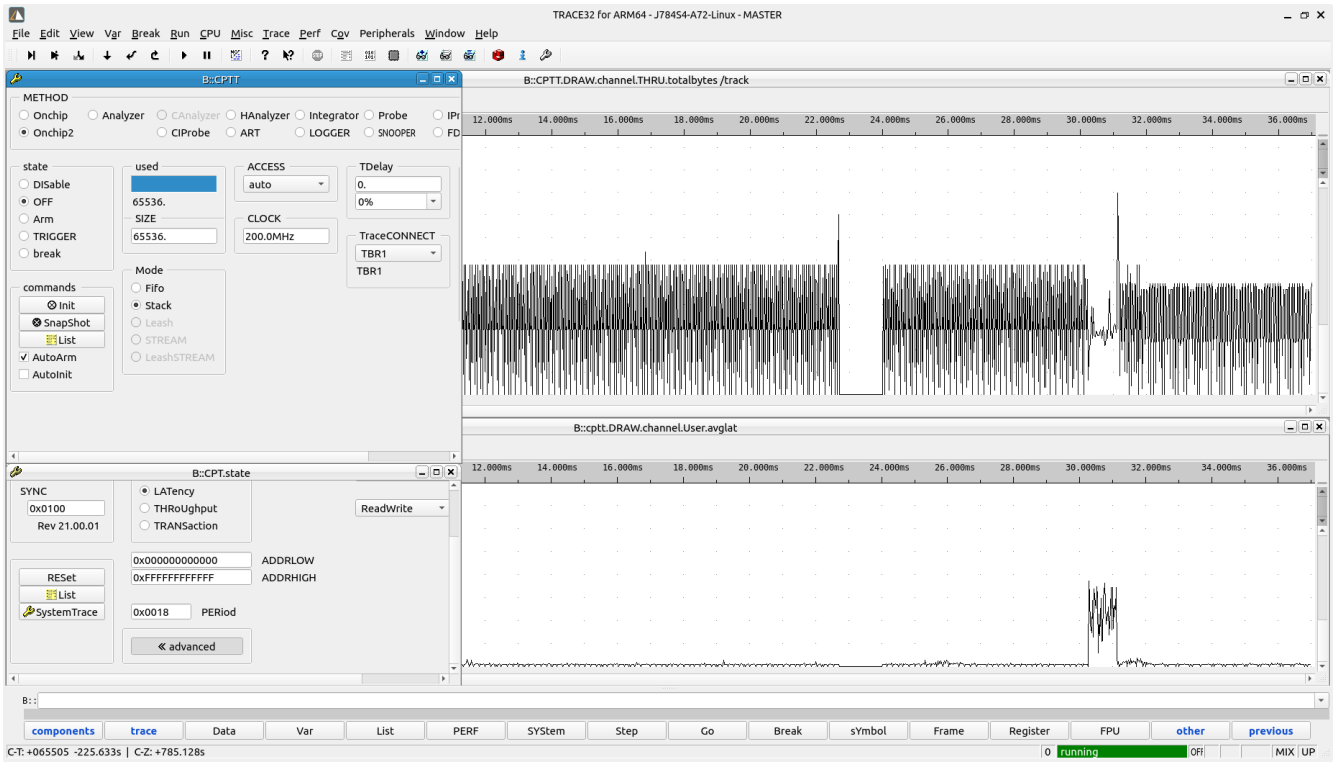

图 3-14. DSS 吞吐量与 DSS 延迟之间的关系

C7x_4 内核（路由 0x2C）事务与 DSS 事务的延迟比较。

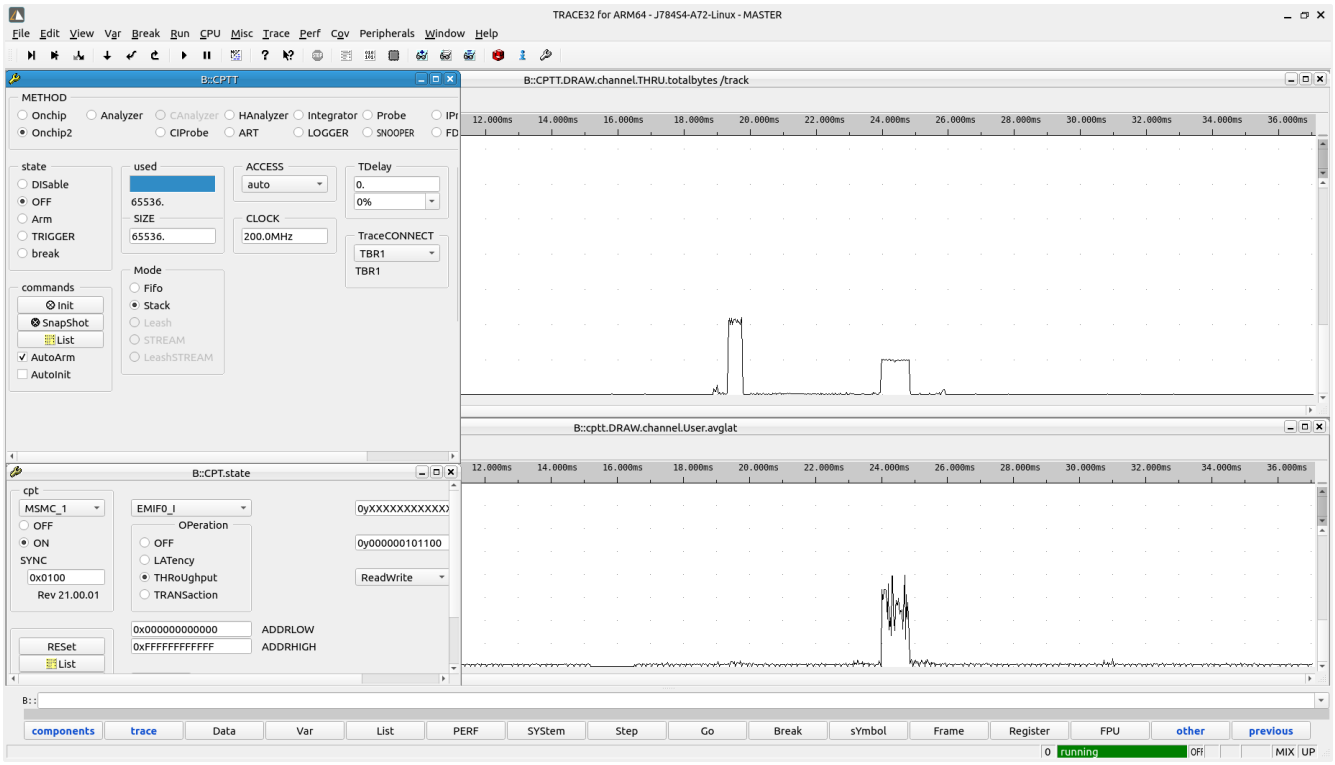

图 3-15. C7x_4 内核吞吐量与 DSS 延迟之间的关系

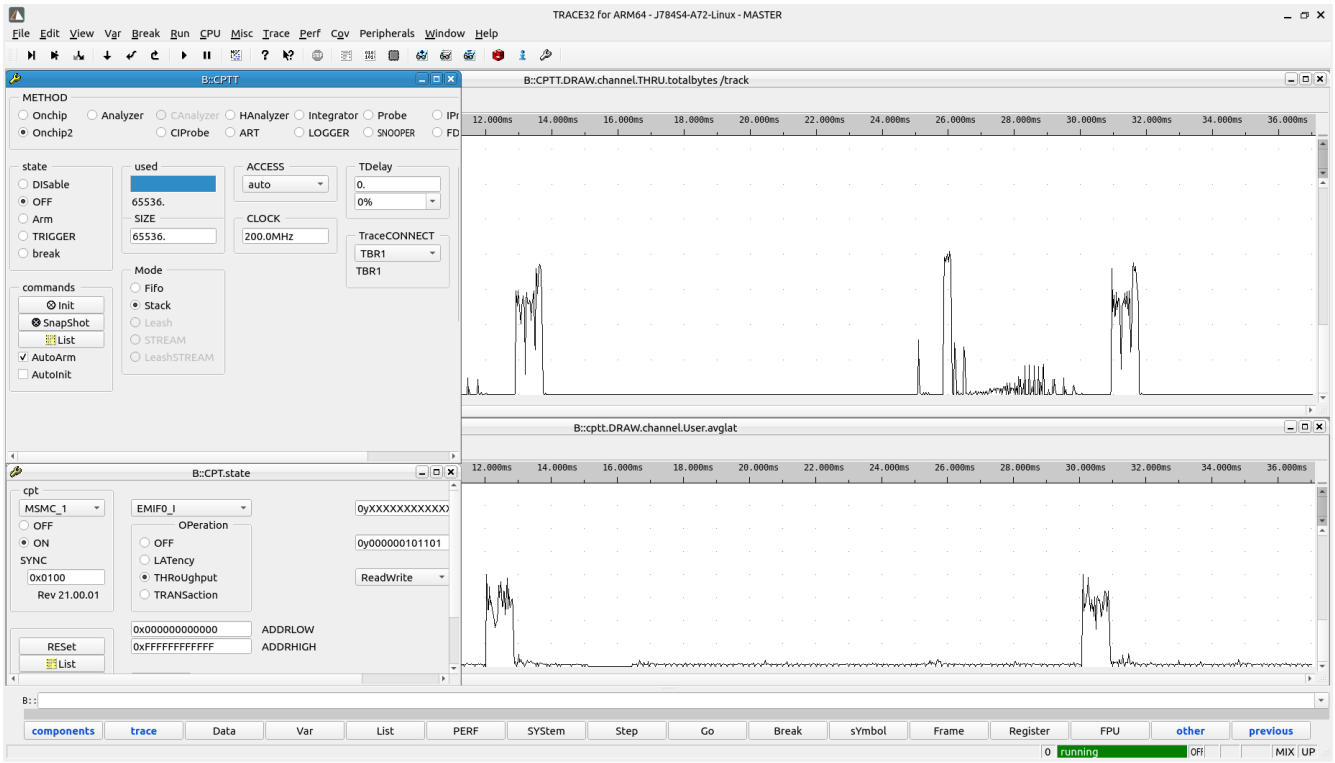

图 3-16. C7x_4 DRU0 吞吐量与 DSS 延迟之间的关系

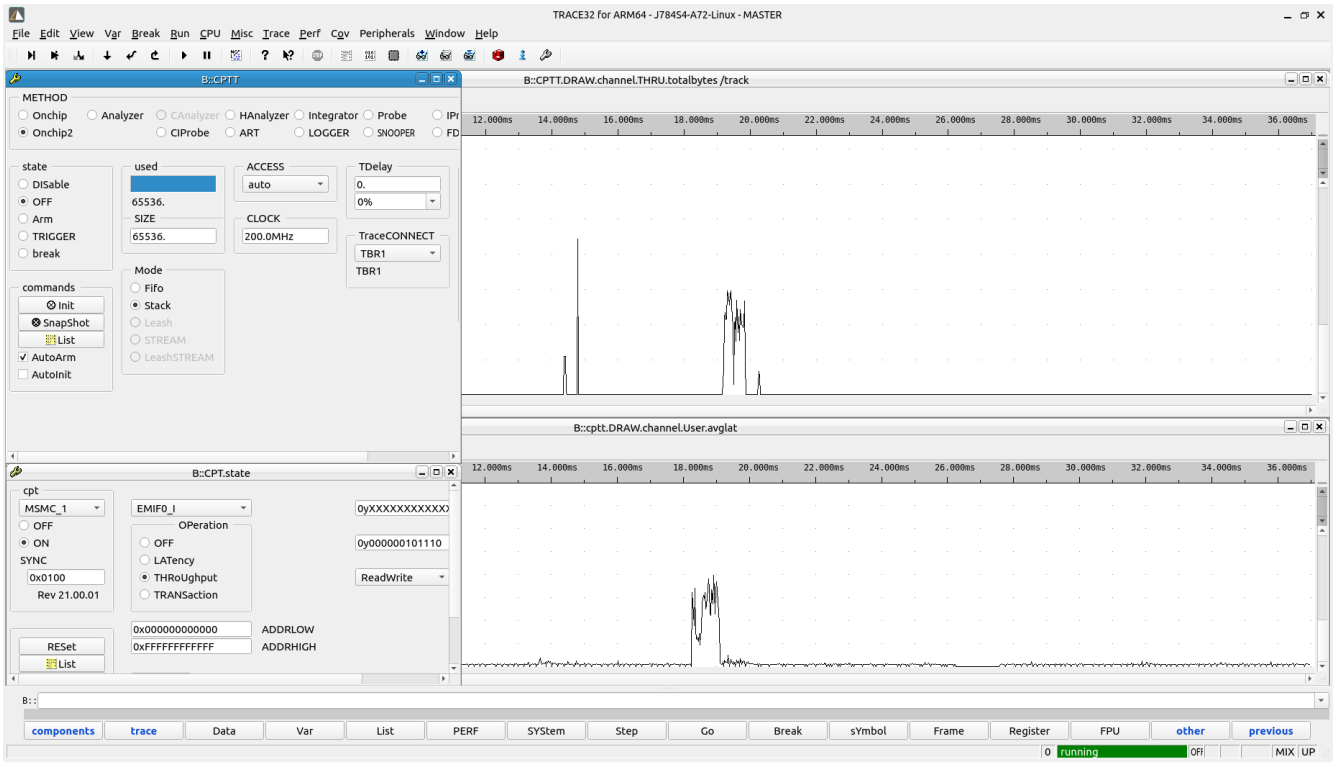

图 3-17. C7x_4 DRU1 内核吞吐量与 DSS 延迟之间的关系

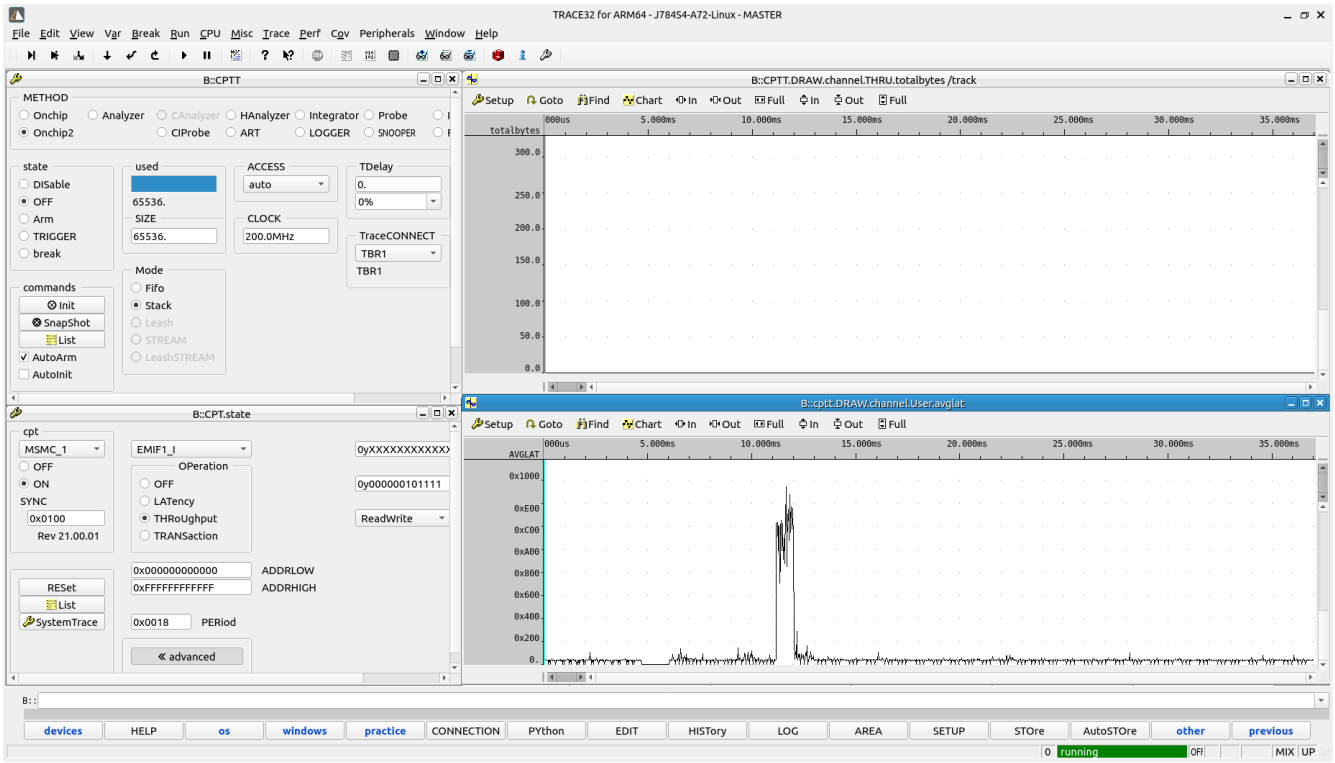

图 3-18. C7x_4 CMMU 吞吐量与 DSS 延迟之间的关系

从前面的图中可以看出，问题似乎出在 C7x_4 内核 (0x2D) 事务上。

3.3.1.10 分析 C7x_4 内核事务

使用 CPTracer 的事务分析功能，我们可以查看 C7x_4 内核和 DSS 事务的 QoS 设置。

C7x_4 内核事务：

- **路由 ID**：0x002D
- **优先级**：0x03
- **QoS**：0x00
- **顺序 ID**：0x00

DSS 事务：

- **路由 ID**：0xA20
- **优先级**：0x00 或 0x01
- **QoS**：0x00
- **顺序 ID**：0x0F

从优先级来看，C7x_4 内核事务不得高于 DSS 事务。

3.3.2 编辑 QoS 设置

尽管所有 QoS 设置看似正确，但为确保记录完整，我们仍将说明如何设置或修改这些参数。我们还需要验证 DSS 事务是否路由到 RT 线程而不是 NRT 线程。

设置相关 QoS 设置后，可以使用 CPTracer 的事务分析功能来验证所做的更改。

备注

可以使用 SYSCONFIG 附带的 **Keystone3 资源分区工具**生成用于编辑 QoS 设置的代码。

3.3.2.1 编辑顺序 ID

通过更改特定事务的顺序 ID，可以将其路由到 NRT 或 RT 线程。这是在 CBASS 配置寄存器和独立 IP（取决于 IP）内完成的。

3.3.2.1.1 DSS 顺序 ID

DSS 的顺序 ID 在 CBASS 配置寄存器中设置。有一些寄存器用于控制 DSS 中每个通道的顺序 ID，还有一些寄存器用于将这些顺序 ID 映射到 VBUSM 顺序 ID。DSS 包含一个主要端口（用“DMA”表示）和辅助端口（用“帧缓冲区解压缩 (FBDC)”表示）。如果为特定通道启用了压缩，则该通道的所有事务都会通过辅助端口。

表 3-7. DSS 顺序 ID 寄存器

寄存器	地址	位	字段	说明
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_dma_slv_linkgrp_1_grp_map1	0x45DC2000	不适用	不适用	组映射寄存器为组 slv_linkgrp_1 定义了启动器 lk3_dss_main_0.dss_inst0_vbusm_dma 的最终顺序 ID。
		31:28	ORDERID7	针对 7 的顺序 ID 信号
		27:24	ORDERID6	针对 6 的顺序 ID 信号
		23:20	ORDERID5	针对 5 的顺序 ID 信号
		19:16	ORDERID4	针对 4 的顺序 ID 信号
		15:12	ORDERID3	针对 3 的顺序 ID 信号
		11:8	ORDERID2	针对 2 的顺序 ID 信号
		7:4	ORDERID1	针对 1 的顺序 ID 信号
		3:0	ORDERID0	针对 0 的顺序 ID 信号
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_dma_slv_linkgrp_1_grp_map2	0x45DC2004	不适用	不适用	组映射寄存器为组 slv_linkgrp_1 定义了启动器 lk3_dss_main_0.dss_inst0_vbusm_dma 的最终顺序 ID。
		31:28	ORDERID15	针对 15 的顺序 ID 信号
		27:24	ORDERID14	针对 14 的顺序 ID 信号
		23:20	ORDERID13	针对 13 的顺序 ID 信号
		19:16	ORDERID12	针对 12 的顺序 ID 信号
		15:12	ORDERID11	针对 11 的顺序 ID 信号
		11:8	ORDERID10	针对 10 的顺序 ID 信号
		7:4	ORDERID9	针对 9 的顺序 ID 信号
		3:0	ORDERID8	针对 8 的顺序 ID 信号

表 3-7. DSS 顺序 ID 寄存器（续）

寄存器	地址	位	字段	说明
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_dma_mapx	0x45DC2100 + (x * 4); 其中，x = 0 - 9	7:4	ORDERID	通道 x 的顺序 ID 信号选择用于负载平衡的路由（0-7 使用一条路由，8-15 使用另一条路由）。 也可用于 DDR4/LPDDR4 重新排序，以更大限度地提高吞吐量。仅当顺序 ID 相同时，事务的顺序才能得到保证
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_fbdc_slv_linkgrp_1_grp_map1	0x45DC2404	不适用	不适用	组映射寄存器为组 slv_linkgrp_1 定义了启动器 lk3_dss_main_0.dss_inst0_vbusm_fbdc 的最终顺序 ID。
		31:28	ORDERID7	针对 7 的顺序 ID 信号
		27:24	ORDERID6	针对 6 的顺序 ID 信号
		23:20	ORDERID5	针对 5 的顺序 ID 信号
		19:16	ORDERID4	针对 4 的顺序 ID 信号
		15:12	ORDERID3	针对 3 的顺序 ID 信号
		11:8	ORDERID2	针对 2 的顺序 ID 信号
		7:4	ORDERID1	针对 1 的顺序 ID 信号
		3:0	ORDERID0	针对 0 的顺序 ID 信号
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_fbdc_slv_linkgrp_1_grp_map2	0x45DC2408	不适用	不适用	组映射寄存器为组 slv_linkgrp_1 定义了启动器 lk3_dss_main_0.dss_inst0_vbusm_fbdc 的最终顺序 ID。
		31:28	ORDERID15	针对 15 的顺序 ID 信号
		27:24	ORDERID14	针对 14 的顺序 ID 信号
		23:20	ORDERID13	针对 13 的顺序 ID 信号
		19:16	ORDERID12	针对 12 的顺序 ID 信号
		15:12	ORDERID11	针对 11 的顺序 ID 信号
		11:8	ORDERID10	针对 10 的顺序 ID 信号
		7:4	ORDERID9	针对 9 的顺序 ID 信号
		3:0	ORDERID8	针对 8 的顺序 ID 信号

表 3-7. DSS 顺序 ID 寄存器（续）

寄存器	地址	位	字段	说明
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_fbdc_map x	0x45DC2500 + (x * 4); 其中，x = 0 - 9	7:4	ORDERID	通道 x 的顺序 ID 信号选择用于负载平衡的路由（0-7 使用一条路由，8-15 使用另一条路由）。 也可用于 DDR4/LPDDR4 重新排序，以更大限度地提高吞吐量。仅当顺序 ID 相同时，事务的顺序才能得到保证

用于设置 DSS 通道顺序 ID 的代码可在 ti-u-boot 存储库中找到，位置为 arch/arm/mach-k3/r5/j784s4/j784s4_qos_uboot.c。该文件用于设置常规 QoS 参数，可使用 SYSCONFIG 生成。

摘录中包含视频平面 1 (VID1) 的设置，但文件还包含视频平面 2 (VID2)、视频精简平面 1 (VIDL1) 和视频精简平面 2 (VIDL2) 的设置。每个平面都与两个通道相关联。

```
...

struct k3_qos_data qos_data[] = {
        /* DSS_PIPE_VID1 - 2 endpoints, 2 channels */
        {
                .reg = K3_QOS_REG(K3_DSS_MAIN_0_DSS_INST0_VBUSM_DMA, 0),
                .val = K3_QOS_VAL(0, 15, 0, 0, 0, 0),
        },
        {
                .reg = K3_QOS_REG(K3_DSS_MAIN_0_DSS_INST0_VBUSM_DMA, 1),
                .val = K3_QOS_VAL(0, 15, 0, 0, 0, 0),
        },
        {
                .reg = K3_QOS_REG(K3_DSS_MAIN_0_DSS_INST0_VBUSM_FBDC, 0),
                .val = K3_QOS_VAL(0, 15, 0, 0, 0, 0),
        },
        {
                .reg = K3_QOS_REG(K3_DSS_MAIN_0_DSS_INST0_VBUSM_FBDC, 1),
                .val = K3_QOS_VAL(0, 15, 0, 0, 0, 0),
        },

...

        /* Following registers set 1:1 mapping for orderID MAP1/MAP2
         * remap registers. orderID x is remapped to orderID x again
         * This is to ensure orderID from MAP register is unchanged
         */

        /* K3_DSS_MAIN_0_DSS_INST0_VBUSM_DMA - 1 groups */
        {
                .reg = K3_QOS_GROUP_REG(K3_DSS_MAIN_0_DSS_INST0_VBUSM_DMA, 0),
                .val = K3_QOS_GROUP_DEFAULT_VAL_LOW,
        },
        {
                .reg = K3_QOS_GROUP_REG(K3_DSS_MAIN_0_DSS_INST0_VBUSM_DMA, 1),
                .val = K3_QOS_GROUP_DEFAULT_VAL_HIGH,
        },

        /* K3_DSS_MAIN_0_DSS_INST0_VBUSM_FBDC - 1 groups */
        {
                .reg = K3_QOS_GROUP_REG(K3_DSS_MAIN_0_DSS_INST0_VBUSM_FBDC, 0),
                .val = K3_QOS_GROUP_DEFAULT_VAL_LOW,
        },
        {
                .reg = K3_QOS_GROUP_REG(K3_DSS_MAIN_0_DSS_INST0_VBUSM_FBDC, 1),
                .val = K3_QOS_GROUP_DEFAULT_VAL_HIGH,
        },

...
```

在 arch/arm/mach-k3/include/mach/k3-qos.h 中，定义了 K3_QOS_REG、K3_QOS_VAL、K3_QOS_GROUP_REG 和 K3_QOS_GROUP_DEFAULT_VAL_LOW/HIGH：

```
...

/* K3_QOS_REG: Registers to configure the channel for a given endpoint */

#define K3_QOS_REG(base_reg, i)     (base_reg + 0x100 + (i) * 4)

#define K3_QOS_VAL(qos, orderid, asel, epriority, virtid, atype) \
    (qos        << 0  | \
     orderid    << 4  | \
     asel       << 8  | \
     epriority  << 12 | \
     virtid     << 16 | \
     atype      << 28)

/*
 * K3_QOS_GROUP_REG: Registers to set 1:1 mapping for orderID MAP1/MAP2
 * remap registers.
 */
#define K3_QOS_GROUP_REG(base_reg, i)   (base_reg + (i) * 4)

#define K3_QOS_GROUP_DEFAULT_VAL_LOW    0x76543210
#define K3_QOS_GROUP_DEFAULT_VAL_HIGH   0xfedcba98
struct k3_qos_data {
    u32 reg;
    u32 val;
};

...
```

3.3.2.1.2 C7x 顺序 ID

每个 C7x 的顺序 ID 可使用 DRU 队列配置寄存器进行配置。

表 3-8. C7x 顺序 ID 寄存器

C7x 子系统	地址	寄存器	位	字段	说明
COMPUTE_CLUSTER0_C71SS0	0x68A08000 + (j * 8); 其中 j = 0 - 6	DRU_QUEUE_cfg_j	7:4	ORDERID	这将配置队列的顺序 ID。
COMPUTE_CLUSTER0_C71SS1	0x69A08000 + (j * 8); 其中 j = 0 - 6	DRU_QUEUE_cfg_j	7:4	ORDERID	这将配置队列的顺序 ID。
COMPUTE_CLUSTER0_C71SS2	0x6AA08000 + (j * 8); 其中 j = 0 - 6	DRU_QUEUE_cfg_j	7:4	ORDERID	这将配置队列的顺序 ID。
COMPUTE_CLUSTER0_C71SS3	0x6BA08000 + (j * 8); 其中 j = 0 - 6	DRU_QUEUE_cfg_j	7:4	ORDERID	这将配置队列的顺序 ID。

用于设置 DRU 队列配置寄存器和顺序 ID 的代码位于 vision_apps 存储库 (platform/j784s4/rtos/c7x_4/main.c) 的相应 main.c 中。

```
...

/* DRU configuration */
uint32_t gDruQoS_Enable    = 1;
uint32_t gQoS_DRU_Priority = 3;
uint32_t gQoS_DRU_OrderID  = 0;

void setup_dru_qos(void)
{
   uint64_t DRU_BASE = CSL_COMPUTE_CLUSTER0_MMR_DRU7_MMR_CFG_DRU_BASE;
   volatile uint64_t* queue0CFG     = (uint64_t*)(DRU_BASE + 0x8000);

   if(gQoS_DRU_Priority > 7 || (gDruQoS_Enable == 0))
   {
     gQoS_DRU_Priority = 0;
   }
```

```
    if(gQoS_DRU_OrderID > 15 || (gDruQoS_Enable == 0))
    {
      gQoS_DRU_OrderID = 0;
    }

    uint64_t queue0CFG_VAL = 0x0;
    queue0CFG_VAL |= ((uint64_t)gQoS_DRU_OrderID)<<4;
    queue0CFG_VAL |= ((uint64_t)gQoS_DRU_Priority);

    *queue0CFG = queue0CFG_VAL;
}

...
```

3.3.2.2 NRT 和 RT 路由

观察 C7x_4 内核 (0) 和 DSS (15) 事务的顺序 ID 后发现，可以将 DSS 事务设置为 RT，同时将 C7x_4 内核事务保持为 NRT。顺序 ID 0-4 应路由到 NRT 线程，而顺序 ID 10-15 应路由到 RT 线程。

检查 DSS 事务是否路由到 RT 线程其实很简单，我们可以读取 NAVSS_NORTH_x_NBSS_NBx_MMRS_threadmap 寄存器（使用 devmem2）。

表 3-9. 北桥线程映射寄存器

寄存器	地址	值
NAVSS_NORTH_0_NBSS_NB0_MMRS_threadmap	0x03702010	0x00000002
NAVSS_NORTH_1_NBSS_NB1_MMRS_threadmap	0x03703010	0x00000004

这些值表示顺序 ID 为 0-7 的 SRAM 和 DDR 事务映射到 NRT 线程，而顺序 ID 为 8-15 的 SRAM 事务和顺序 ID 为 10-15 的 DDR 事务映射到 RT 线程。换句话说，C7x_4（顺序 ID 0）事务映射到 NRT 线程，而 DSS（顺序 ID 15）事务映射到 RT 线程。因此，DSS 事务始终应比 C7x_4 事务具有更高的优先级。

3.3.2.2.1 U-Boot 中的 NRT 和 RT 路由

对于 J784S4，RT 和 NRT 映射添加到 ti-u-boot-2025.01 分支/版本中。在早期版本中，相关代码是通过 SDK 内部打包的未提交修改引入的。相关代码可以在 arch/arm/mach-k3/j784s4/j784s4_init.c 中找到。

下面的代码将顺序 ID 为 8-15 的 SRAM 事务和顺序 ID 为 10-15 的 DDR 事务映射到 RT 线程。

```
...

/* NAVSS North Bridge (NB) */
#define NAVSS0_NBSS_NB0_CFG_MMRS                0x03702000
#define NAVSS0_NBSS_NB1_CFG_MMRS                0x03703000
#define NAVSS0_NBSS_NB0_CFG_NB_THREADMAP        (NAVSS0_NBSS_NB0_CFG_MMRS + 0x10)
#define NAVSS0_NBSS_NB1_CFG_NB_THREADMAP        (NAVSS0_NBSS_NB1_CFG_MMRS + 0x10)
/*
 * Thread Map for North Bridge Configuration
 * Each bit is for each VBUSM source.
 * Bit[0] maps orderID 0-3 to VBUSM.C thread number
 * Bit[1] maps orderID 4-9 to VBUSM.C thread number
 * Bit[2] maps orderID 10-15 to VBUSM.C thread number
 * When bit has value 0: VBUSM.C thread 0 (non-real time traffic)
 * When bit has value 1: VBUSM.C thread 2 (real time traffic)
 */
#define NB_THREADMAP_BIT0                               BIT(0)
#define NB_THREADMAP_BIT1                               BIT(1)
#define NB_THREADMAP_BIT2                               BIT(2)

...

/* Setup North Bridge registers to map ORDERID 10-15 to RT traffic */
static void setup_navss_nb(void)
{
        writel(NB_THREADMAP_BIT1, (uintptr_t)NAVSS0_NBSS_NB0_CFG_NB_THREADMAP);
        writel(NB_THREADMAP_BIT2, (uintptr_t)NAVSS0_NBSS_NB1_CFG_NB_THREADMAP);
}
```

```
...
```

3.3.2.3 编辑优先级

各个 IP 的优先级在各自的驱动程序中设置。

3.3.2.3.1 DSS 优先级

DSS 包含两个可能的事务优先级：MFLAG 事务的高优先级和非 MFLAG 事务的低优先级。MFLAG 机制允许 DSS 在其 DMA 读取缓冲区接近下溢时提高其流量的优先级。

表 3-10. DSS 优先级寄存器

寄存器	地址	位	字段	说明
DSS_DISPC_0_COMMON_M_DSS_CBA_CFG	0x04A000A4	5:3	PRI_HI	PRI_HI 总线上从 DSS 发送到 CBA 的值用于指示高优先级 [MFLAG] 事务的优先级。值 0x0 表示最高优先级 值 0x7 表示最低优先级
DSS_DISPC_0_COMMON_M_DSS_CBA_CFG	0x04A000A4	2:0	PRI_LO	PRI_LO 总线上从 DSS 发送到 CBA 的值用于指示正常 [非 MFLAG] 事务的优先级。值 0x0 表示最高优先级 值 0x7 表示最低优先级

用于设置 DSS 优先级的代码可在其 Linux 驱动程序 (drivers/gpu/drm/tidss/tidss_dispc.c) 中找到。此代码将 MFLAG 事务的优先级设置为 0，将非 MFLAG 事务的优先级设置为 1。

```
...

        u32 cba_lo_pri = 1;
        u32 cba_hi_pri = 0;

        dev_dbg(dispc->dev, "%s()\n", __func__);

        REG_FLD_MOD(dispc, DSS_CBA_CFG, cba_lo_pri, 2, 0);
        REG_FLD_MOD(dispc, DSS_CBA_CFG, cba_hi_pri, 5, 3);

...
```

3.3.2.3.2 C7x 优先级

与顺序 ID 一样，每个 C7x 的优先级也可以使用 DRU 队列配置寄存器进行配置。

表 3-11. C7x 优先级寄存器

C7x	寄存器	寄存器	位	字段	说明
COMPUTE_CLUSTER0_C71SS0	0x68A08000 + (j * 8); 其中 j = 0 - 6	DRU_QUEUE_cfg_j	2:0	PRI	这将配置 QUEUE0 的优先级。对于此队列中的所有命令，这将是外部总线上显示的优先级。
COMPUTE_CLUSTER0_C71SS1	0x69A08000 + (j * 8); 其中 j = 0 - 6	DRU_QUEUE_cfg_j	2:0	PRI	这将配置 QUEUE0 的优先级。对于此队列中的所有命令，这将是外部总线上显示的优先级。

表 3-11. C7x 优先级寄存器（续）

C7x	寄存器	寄存器	位	字段	说明
COMPUTE_CLUSTER0_C71SS2	0x6AA08000 + (j * 8); 其中 j = 0 - 6	DRU_QUEUE_cfg_j	2:0	PRI	这将配置 QUEUE0 的优先级。对于此队列中的所有命令，这将是外部总线上显示的优先级。
COMPUTE_CLUSTER0_C71SS3	0x6BA08000 + (j * 8); 其中 j = 0 - 6	DRU_QUEUE_cfg_j	2:0	PRI	这将配置 QUEUE0 的优先级。对于此队列中的所有命令，这将是外部总线上显示的优先级。

与顺序 ID 一样，用于设置优先级的代码可以在 vision_apps 存储库 (platform/j784s4/rtos/c7x_4/main.c) 的相应 main.c 中找到。

```
...

/* DRU configuration */
uint32_t gDruQoS_Enable    = 1;
uint32_t gQoS_DRU_Priority = 3;
uint32_t gQoS_DRU_OrderID  = 0;

void setup_dru_qos(void)
{
   uint64_t DRU_BASE = CSL_COMPUTE_CLUSTER0_MMR_DRU7_MMR_CFG_DRU_BASE;
   volatile uint64_t* queue0CFG     = (uint64_t*)(DRU_BASE + 0x8000);

   if(gQoS_DRU_Priority > 7 || (gDruQoS_Enable == 0))
   {
     gQoS_DRU_Priority = 0;
   }
   if(gQoS_DRU_OrderID > 15 || (gDruQoS_Enable == 0))
   {
     gQoS_DRU_OrderID = 0;
   }

   uint64_t queue0CFG_VAL = 0x0;
   queue0CFG_VAL |= ((uint64_t)gQoS_DRU_OrderID)<<4;
   queue0CFG_VAL |= ((uint64_t)gQoS_DRU_Priority);

   *queue0CFG = queue0CFG_VAL;
}

...
```

3.3.3 编辑 CoS 映射

由于所有 QoS 设置都是正确的并优先处理 DSS 事务（而不是 C7x_4 内核事务），因此问肯定出在 DDR 控制器的优先级映射中。

3.3.3.1 CoS 映射寄存器

DDRSS 包含一系列多路复用器，用于将 VBUSM.C 优先级映射到 AXI 优先级。用于控制映射的寄存器如下：

- **路由 ID 过滤器**：
  - emif_ew_sscfg_V2A_R1_MAT_REG：允许过滤路由 ID 并将其路由到范围 1 映射
  - emif_ew_sscfg_V2A_R2_MAT_REG：允许过滤路由 ID 并将其路由到范围 2 映射
  - emif_ew_sscfg_V2A_R3_MAT_REG：允许过滤路由 ID 并将其路由到范围 3 映射
- **优先级映射**：
  - **LPT（低优先级线程）**：
    - emif_ew_sscfg_V2A_LPT_DEF_PRI_MAP_REG：默认 VBUSM.C 到 AXI 优先级映射
    - emif_ew_sscfg_V2A_LPT_R1_PRI_MAP_REG：范围 1 VBUSM.C 到 AXI 优先级映射

- emif_ew_sscfg_V2A_LPT_R2_PRI_MAP_REG：范围 2 VBUSM.C 到 AXI 优先级映射
- emif_ew_sscfg_V2A_LPT_R3_PRI_MAP_REG：范围 3 VBUSM.C 到 AXI 优先级映射

– **HPT（高优先级线程）**：

- emif_ew_sscfg_V2A_HPT_DEF_PRI_MAP_REG：默认 VBUSM.C 到 AXI 优先级映射
- emif_ew_sscfg_V2A_HPT_R1_PRI_MAP_REG：范围 1 VBUSM.C 到 AXI 优先级映射
- emif_ew_sscfg_V2A_HPT_R2_PRI_MAP_REG：范围 2 VBUSM.C 到 AXI 优先级映射
- emif_ew_sscfg_V2A_HPT_R3_PRI_MAP_REG：范围 3 VBUSM.C 到 AXI 优先级映射

备注

确切的寄存器地址和字段可在 TDA4VH TRM 中找到

采用假设的 **LPT** 事务。如果其路由 ID 位于过滤器内，例如**范围 1**，则它将使用 **emif_ew_sscfg_V2A_LPT_R1_PRI_MAP_REG** 映射来映射其优先级。如果它不属于任何过滤器，它将使用 **emif_ew_sscfg_V2A_LPT_DEF_PRI_MAP_REG** 寄存器来映射其优先级。这种复用方式如 图 2-1 所示。

3.3.3.2 检查 CoS 映射

下面再次说明 DSS 和 C7x_4 内核事务的 QoS 设置：

- **DSS**：
  - 路由 ID：0xA20
  - 顺序 ID：0x0F
  - NRT 或 RT：RT
  - 优先级：0x00 或 0x01
- **C7x_4 内核**：
  - 路由 ID：0x02D
  - 顺序 ID：0x00
  - NRT 或 RT：NRT
  - 优先级：0x03

下表包含 CoS 寄存器的值（寄存器组是我用于对寄存器进行分类的项）：

表 3-12. CoS 寄存器

寄存器组	寄存器	值
路由 ID 过滤器	emif_ew_sscfg_V2A_R1_MAT_REG	0x00000000
	emif_ew_sscfg_V2A_R2_MAT_REG	0x00000000
	emif_ew_sscfg_V2A_R3_MAT_REG	0x00000000
LPT 优先级映射	emif_ew_sscfg_V2A_LPT_DEF_PRI_MAP_REG	0x00000000
	emif_ew_sscfg_V2A_LPT_R1_PRI_MAP_REG	0x23456677
	emif_ew_sscfg_V2A_LPT_R2_PRI_MAP_REG	0x23456677
	emif_ew_sscfg_V2A_LPT_R3_PRI_MAP_REG	0x23456677

表 3-12. CoS 寄存器（续）

寄存器组	寄存器	值
HPT 优先级映射	emif_ew_sscfg_V2A_HPT_DEF_PRI_MAP_REG	0x00000000
	emif_ew_sscfg_V2A_HPT_R1_PRI_MAP_REG	0x00112345
	emif_ew_sscfg_V2A_HPT_R2_PRI_MAP_REG	0x00112345
	emif_ew_sscfg_V2A_HPT_R3_PRI_MAP_REG	0x00112345

未启用任何**路由 ID 过滤器**。这意味着所有优先级都将使用默认优先级映射进行映射。

LPT 和 HPT 事务的默认优先级映射都被均衡为 0。这意味着所有事务在 DDRSS 中都具有相同的优先级。因此，DSS 和 C7x_4 内核事务在 DDR 中具有相同的优先级；这解释了了 C7x_4 内核事务如何使 DSS 事务停滞。

3.4 修复 DSS 同步丢失问题

找到了根本原因后，接下来需要决定如何解决问题。

使 DDR 控制器内的所有优先级都得到均衡，可带来若干益处。也就是说，当优先级较高的线程进入队列时，不会逐出任何线程，这有助于防止页面抖动。然而，在这种情况下，不遵守优先级所带来的弊端远大于其益处。

有几种方法可以解决同步丢失问题：

1. 重新映射 C7x_4 内核事务
2. 在所有事务中遵守优先级

仅重映射 C7x_4 内核事务将解决同步丢失问题，同时使所有其他事务能够取代 DSS 事务。在所有事务中遵守优先级将强制 DDRSS 以其 VBUSM.C 优先级相应地处理所有事务。

由于只有八种不同的 AXI 优先级设置，因此在尝试在所有事务中遵守优先级时，HPT 和 LPT 之间存在一些重叠。

表 3-13.

实时或非实时	VBUSM.C 优先级	AXI 优先级
实时	0	0
实时	1	0
实时	2	1
实时	3	1
实时	4	2
非实时	0	2
实时	5	3
非实时	1	3
实时	6	4
非实时	2	4
实时	7	5
非实时	3	5
非实时	4	6
非实时	5	6
非实时	6	7
非实时	7	7

CoS 映射在电路板初始化期间添加到 U-Boot。为 ti-u-boot-2023.04 和 ti-u-boot-2025.01 分支编写了补丁。

3.4.1 重新映射 C7x_4 内核事务

以下 U-Boot 补丁将 C7x_4 内核事务映射到 LPT 范围 1 优先级映射。这样可以避免更改任何其他事务的优先级。

备注

如果想将所有 C7x 事务设置为 LPT 范围 1 优先级映射，则应将 0xf02d0000 替换为 0x80208028

3.4.1.1 ti-u-boot-2023.04

以下补丁应用于提交的内容： 2bedcd265ca6 ("configs: am57xx_hs_evm_defconfig: Enable configs needed for Early Boot")

0001-arm-mach-k3-j784s4-Remove-priority-equalization-for-.patch

```
From 6a8d46d01e482cff6678189087155f2dd2ddafc9 Mon Sep 17 00:00:00 2001
From: Jared McArthur <j-mcarthur@ti.com>
Date: Thu, 28 Aug 2025 18:33:22 -0500
Subject: [PATCH 1/1] arm: mach-k3: j784s4: Remove priority equalization for
 C7x_4 core transactions

Add C7x_4 core transactions to the low priority thread (LPT) R1 mux
within the DDR controller. By default, the J784S4's MSMC2DDR bridge
maps all VBUSM priorities to 0 and all transactions travel through the
default mux [0].

By default, the LPT R1 mux honors VBUSM priorities. Move C7x_4 core
transactions to the LPT R1 mux, so they will honor VBUSM priorities.
All other transactions priorities remain unchanged from default
behavior.

[0] https://www.ti.com/lit/zip/spruj52

Signed-off-by: Jared McArthur <j-mcarthur@ti.com>
---
 arch/arm/mach-k3/j784s4_init.c | 33 +++++++++++++++++++++++++++++++++
 1 file changed, 33 insertions(+)

diff --git a/arch/arm/mach-k3/j784s4_init.c b/arch/arm/mach-k3/j784s4_init.c
index af0f46e2ab5..4bcc83dadaa 100644
--- a/arch/arm/mach-k3/j784s4_init.c
+++ b/arch/arm/mach-k3/j784s4_init.c
@@ -22,6 +22,26 @@
 #include <mmc.h>
 #include <remoteproc.h>

+/* DDRSS Config */
+#define DDRSS0_EMIF_EW_SSCFG    0x02980000
+#define DDRSS1_EMIF_EW_SSCFG    0x029A0000
+#define DDRSS2_EMIF_EW_SSCFG    0x029C0000
+#define DDRSS3_EMIF_EW_SSCFG    0x029E0000
+#define DDRSS0_V2A_R1_MAT_REG    (DDRSS0_EMIF_EW_SSCFG + 0x24)
+#define DDRSS1_V2A_R1_MAT_REG    (DDRSS1_EMIF_EW_SSCFG + 0x24)
+#define DDRSS2_V2A_R1_MAT_REG    (DDRSS2_EMIF_EW_SSCFG + 0x24)
+#define DDRSS3_V2A_R1_MAT_REG    (DDRSS3_EMIF_EW_SSCFG + 0x24)
+
+/*
+ * Move the C7x_4 core transactions to the LPT range 1 priority mappings.
+ * This decreases the priority of specifically C7x core transactions,
+ * but doesn't impact the priority of any other transactions.
+ */
+#define DDRSS0_V2A_R1_MAT    0xf02d0000
+#define DDRSS1_V2A_R1_MAT    0xf02d0000
+#define DDRSS2_V2A_R1_MAT    0xf02d0000
+#define DDRSS3_V2A_R1_MAT    0xf02d0000
+
 struct fwl_data infra_cbass0_fwls[] = {
     { "PSC0", 5, 1 },
     { "PLL_CTRL0", 6, 1 },
@@ -139,6 +159,17 @@ static void store_boot_info_from_rom(void)

 #define J784S4_MAX_CONTROLLERS    4

+/* Setup DDRSS CoS (Class of Service) registers to remove priority equalization
+ * for C7x_4 core transactions
```

```
+ */
+static void setup_ddrss_cos(void)
+{
+    writel(DDRSS0_V2A_R1_MAT, (uintptr_t)DDRSS0_V2A_R1_MAT_REG);
+    writel(DDRSS1_V2A_R1_MAT, (uintptr_t)DDRSS1_V2A_R1_MAT_REG);
+    writel(DDRSS2_V2A_R1_MAT, (uintptr_t)DDRSS2_V2A_R1_MAT_REG);
+    writel(DDRSS3_V2A_R1_MAT, (uintptr_t)DDRSS3_V2A_R1_MAT_REG);
+}
+
 void board_init_f(ulong dummy)
 {
     struct udevice *dev;
@@ -241,6 +272,8 @@ void board_init_f(ulong dummy)
     }

     spl_enable_dcache();
+
+    setup_ddrss_cos();
 }

 u32 spl_mmc_boot_mode(struct mmc *mmc, const u32 boot_device)
--
2.34.1
```

3.4.1.2 ti-u-boot-2025.01

以下补丁应用于提交的内容：f3f8c664b300 ("PSTREAM: board: ti: common: Kconfig: add CMD_CACHE")

0001-arm-mach-k3-j784s4-Remove-priority-equalization-for-.patch

```
From 734130b26838b77759e7bcbb0f8af79330e48ec7 Mon Sep 17 00:00:00 2001
From: Jared McArthur <j-mcarthur@ti.com>
Date: Thu, 28 Aug 2025 17:35:44 -0500
Subject: [PATCH 1/1] arm: mach-k3: j784s4: Remove priority equalization for
 C7x_4 core transactions

Add C7x_4 core transactions to the low priority thread (LPT) R1 mux
within the DDR controller. By default, the J784S4's MSMC2DDR bridge
maps all VBUSM priorities to 0 and all transactions travel through the
default mux [0].

By default, the LPT R1 mux honors VBUSM priorities. Move C7x_4 core
transactions to the LPT R1 mux, so they will honor VBUSM priorities.
All other transactions priorities remain unchanged from default
behavior.

[0] https://www.ti.com/lit/zip/spruj52

Signed-off-by: Jared McArthur <j-mcarthur@ti.com>
---
 arch/arm/mach-k3/j784s4/j784s4_init.c | 32 ++++++++++++++++++++++++++++++++
 1 file changed, 32 insertions(+)

diff --git a/arch/arm/mach-k3/j784s4/j784s4_init.c b/arch/arm/mach-k3/j784s4/j784s4_init.c
index 9897f4fb921..8557d3b8ddf 100644
--- a/arch/arm/mach-k3/j784s4/j784s4_init.c
+++ b/arch/arm/mach-k3/j784s4/j784s4_init.c
@@ -45,6 +45,26 @@
 #define NB_THREADMAP_BIT1               BIT(1)
 #define NB_THREADMAP_BIT2               BIT(2)

+/* DDRSS Config */
+#define DDRSS0_EMIF_EW_SSCFG    0x02980000
+#define DDRSS1_EMIF_EW_SSCFG    0x029A0000
+#define DDRSS2_EMIF_EW_SSCFG    0x029C0000
+#define DDRSS3_EMIF_EW_SSCFG    0x029E0000
+#define DDRSS0_V2A_R1_MAT_REG    (DDRSS0_EMIF_EW_SSCFG + 0x24)
+#define DDRSS1_V2A_R1_MAT_REG    (DDRSS1_EMIF_EW_SSCFG + 0x24)
+#define DDRSS2_V2A_R1_MAT_REG    (DDRSS2_EMIF_EW_SSCFG + 0x24)
+#define DDRSS3_V2A_R1_MAT_REG    (DDRSS3_EMIF_EW_SSCFG + 0x24)
+
+/*
+ * Move the C7x_4 core transactions to the LPT range 1 priority mappings.
+ * This decreases the priority of specifically C7x core transactions,
+ * but doesn't impact the priority of any other transactions.
+ */
+#define DDRSS0_V2A_R1_MAT    0xf02d0000
```

```
+#define DDRSS1_V2A_R1_MAT    0xf02d0000
+#define DDRSS2_V2A_R1_MAT    0xf02d0000
+#define DDRSS3_V2A_R1_MAT    0xf02d0000
+
 struct fwl_data infra_cbass0_fwls[] = {
     { "PSC0", 5, 1 },
     { "PLL_CTRL0", 6, 1 },
@@ -123,6 +143,17 @@ static void setup_navss_nb(void)
     writel(NB_THREADMAP_BIT2, (uintptr_t)NAVSS0_NBSS_NB1_CFG_NB_THREADMAP);
 }

+/* Setup DDRSS CoS (Class of Service) registers to remove priority equalization
+ * for C7x_4 core transactions
+ */
+static void setup_ddrss_cos(void)
+{
+    writel(DDRSS0_V2A_R1_MAT, (uintptr_t)DDRSS0_V2A_R1_MAT_REG);
+    writel(DDRSS1_V2A_R1_MAT, (uintptr_t)DDRSS1_V2A_R1_MAT_REG);
+    writel(DDRSS2_V2A_R1_MAT, (uintptr_t)DDRSS2_V2A_R1_MAT_REG);
+    writel(DDRSS3_V2A_R1_MAT, (uintptr_t)DDRSS3_V2A_R1_MAT_REG);
+}
+
 /* Execute and check results of BIST executed on MCU1_x and MCU4_0 */
 static void run_bist_j784s4(struct udevice *dev)
 {
@@ -328,6 +359,7 @@ void board_init_f(ulong dummy)
         setup_navss_nb();

     setup_qos();
+    setup_ddrss_cos();
 }

 u32 spl_mmc_boot_mode(struct mmc *mmc, const u32 boot_device)
--
2.34.1
```

3.4.2 遵守所有优先级

以下 U-Boot 补丁会将默认 LPT 和 HPT 优先级映射更改为与范围 1-3 映射相同的映射。这是 TDA4VM/J721E 的默认行为。

3.4.2.1 ti-u-boot-2023.04

以下补丁应用于提交的内容：2bedcd265ca6 ("configs: am57xx_hs_evm_defconfig: Enable configs needed for Early Boot")

0001-arm-mach-k3-j784s4-Remove-priority-equalization-and-.patch

```
From 6ebd1448e30ed1861492738759680b90209fcc22 Mon Sep 17 00:00:00 2001
From: Jared McArthur <j-mcarthur@ti.com>
Date: Thu, 28 Aug 2025 18:33:22 -0500
Subject: [PATCH 1/1] arm: mach-k3: j784s4: Remove priority equalization and
 honor VBUSM priorities

Give the DDR controller's high priority thread (HPT) priority over the
low priority thread (LPT) and give weight to transactions' individual
priorities as well.

By default, the J784S4's MSMC2DDR bridge maps all VBUSM priorities to
0. This is done with priority mapping registers.

DDRSSX_V2A_LPT_DEF_PRI_MAP_REG: default VBUSM to DDR controller
priority mapping for LPT

DDRSSX_V2A_HPT_DEF_PRI_MAP_REG: default VBUSM to DDR controller
priority mapping for HPT

Set DDRSSX_V2A_LPT_DEF_PRI_MAP_REG as 0x23456677 and
DDRSSX_V2A_HPT_DEF_PRI_MAP_REG as 0x00112345. The values are taken
from the default values for the J721E [0] and are also the default values
for the J784S4 priority map range muxes [1].

[0] https://www.ti.com/lit/zip/spruil1
[1] https://www.ti.com/lit/zip/spruj52

Signed-off-by: Jared McArthur <j-mcarthur@ti.com>
```

```
---
 arch/arm/mach-k3/j784s4_init.c | 36 ++++++++++++++++++++++++++++++++++
 1 file changed, 36 insertions(+)

diff --git a/arch/arm/mach-k3/j784s4_init.c b/arch/arm/mach-k3/j784s4_init.c
index af0f46e2ab5..eb45ccd0cb3 100644
--- a/arch/arm/mach-k3/j784s4_init.c
+++ b/arch/arm/mach-k3/j784s4_init.c
@@ -22,6 +22,27 @@
 #include <mmc.h>
 #include <remoteproc.h>

+/* DDRSS Config */
+#define DDRSS0_EMIF_EW_SSCFG        0x02980000
+#define DDRSS1_EMIF_EW_SSCFG        0x029A0000
+#define DDRSS2_EMIF_EW_SSCFG        0x029C0000
+#define DDRSS3_EMIF_EW_SSCFG        0x029E0000
+#define DDRSS0_V2A_LPT_DEF_PRI_MAP_REG    (DDRSS0_EMIF_EW_SSCFG + 0x30)
+#define DDRSS0_V2A_HPT_DEF_PRI_MAP_REG    (DDRSS0_EMIF_EW_SSCFG + 0x4C)
+#define DDRSS1_V2A_LPT_DEF_PRI_MAP_REG    (DDRSS1_EMIF_EW_SSCFG + 0x30)
+#define DDRSS1_V2A_HPT_DEF_PRI_MAP_REG    (DDRSS1_EMIF_EW_SSCFG + 0x4C)
+#define DDRSS2_V2A_LPT_DEF_PRI_MAP_REG    (DDRSS2_EMIF_EW_SSCFG + 0x30)
+#define DDRSS2_V2A_HPT_DEF_PRI_MAP_REG    (DDRSS2_EMIF_EW_SSCFG + 0x4C)
+#define DDRSS3_V2A_LPT_DEF_PRI_MAP_REG    (DDRSS3_EMIF_EW_SSCFG + 0x30)
+#define DDRSS3_V2A_HPT_DEF_PRI_MAP_REG    (DDRSS3_EMIF_EW_SSCFG + 0x4C)
+
+/*
+ * New thread priority mapping to remove complete priority equalization
+ * of threads coming from VBUSM space to DDRSS space.
+ */
+#define DDRSS_V2A_LPT_DEF_PRIMAP    0x23456677
+#define DDRSS_V2A_HPT_DEF_PRIMAP    0x00112345
+
 struct fwl_data infra_cbass0_fwls[] = {
     { "PSC0", 5, 1 },
     { "PLL_CTRL0", 6, 1 },
@@ -139,6 +160,19 @@ static void store_boot_info_from_rom(void)

 #define J784S4_MAX_CONTROLLERS    4

+/* Setup DDRSS CoS (Class of Service) registers to remove priority equalization */
+static void setup_ddrss_cos(void)
+{
+    writel(DDRSS_V2A_LPT_DEF_PRIMAP, (uintptr_t)DDRSS0_V2A_LPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_HPT_DEF_PRIMAP, (uintptr_t)DDRSS0_V2A_HPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_LPT_DEF_PRIMAP, (uintptr_t)DDRSS1_V2A_LPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_HPT_DEF_PRIMAP, (uintptr_t)DDRSS1_V2A_HPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_LPT_DEF_PRIMAP, (uintptr_t)DDRSS2_V2A_LPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_HPT_DEF_PRIMAP, (uintptr_t)DDRSS2_V2A_HPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_LPT_DEF_PRIMAP, (uintptr_t)DDRSS3_V2A_LPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_HPT_DEF_PRIMAP, (uintptr_t)DDRSS3_V2A_HPT_DEF_PRI_MAP_REG);
+}
+
 void board_init_f(ulong dummy)
 {
     struct udevice *dev;
@@ -241,6 +275,8 @@ void board_init_f(ulong dummy)
     }

     spl_enable_dcache();
+
+    setup_ddrss_cos();
 }

 u32 spl_mmc_boot_mode(struct mmc *mmc, const u32 boot_device)
--
2.34.1
```

3.4.2.2 ti-u-boot-2025.01

以下补丁应用于提交的内容：f3f8c664b300 ("PSTREAM: board: ti: common: Kconfig: add CMD_CACHE")

0001-arm-mach-k3-j784s4-Remove-priority-equalization-and-.patch

```
From 7ac3b82eca22d6d28ca767d400c8aa5830704a59 Mon Sep 17 00:00:00 2001
From: Jared McArthur <j-mcarthur@ti.com>
Date: Thu, 28 Aug 2025 17:35:44 -0500
```

```
Subject: [PATCH 1/1] arm: mach-k3: j784s4: Remove priority equalization and
 honor VBUSM priorities

Give the DDR controller's high priority thread (HPT) priority over the
low priority thread (LPT) and give weight to transactions' individual
priorities as well.

By default, the J784S4's MSMC2DDR bridge maps all VBUSM priorities to
0. This is done with priority mapping registers.

DDRSSX_V2A_LPT_DEF_PRI_MAP_REG: default VBUSM to DDR controller
priority mapping for LPT

DDRSSX_V2A_HPT_DEF_PRI_MAP_REG: default VBUSM to DDR controller
priority mapping for HPT

Set DDRSSX_V2A_LPT_DEF_PRI_MAP_REG as 0x23456677 and
DDRSSX_V2A_HPT_DEF_PRI_MAP_REG as 0x00112345. The values are taken
from the default values for the J721E [0] and are also the default values
for the J784S4 priority map range muxes [1].

[0] https://www.ti.com/lit/zip/spruil1
[1] https://www.ti.com/lit/zip/spruj52

Signed-off-by: Jared McArthur <j-mcarthur@ti.com>
---
 arch/arm/mach-k3/j784s4/j784s4_init.c | 35 ++++++++++++++++++++++++++++++++++
 1 file changed, 35 insertions(+)

diff --git a/arch/arm/mach-k3/j784s4/j784s4_init.c b/arch/arm/mach-k3/j784s4/j784s4_init.c
index 9897f4fb921..f66a8fd1774 100644
--- a/arch/arm/mach-k3/j784s4/j784s4_init.c
+++ b/arch/arm/mach-k3/j784s4/j784s4_init.c
@@ -45,6 +45,27 @@
 #define NB_THREADMAP_BIT1               BIT(1)
 #define NB_THREADMAP_BIT2               BIT(2)

+/* DDRSS Config */
+#define DDRSS0_EMIF_EW_SSCFG        0x02980000
+#define DDRSS1_EMIF_EW_SSCFG        0x029A0000
+#define DDRSS2_EMIF_EW_SSCFG        0x029C0000
+#define DDRSS3_EMIF_EW_SSCFG        0x029E0000
+#define DDRSS0_V2A_LPT_DEF_PRI_MAP_REG    (DDRSS0_EMIF_EW_SSCFG + 0x30)
+#define DDRSS0_V2A_HPT_DEF_PRI_MAP_REG    (DDRSS0_EMIF_EW_SSCFG + 0x4C)
+#define DDRSS1_V2A_LPT_DEF_PRI_MAP_REG    (DDRSS1_EMIF_EW_SSCFG + 0x30)
+#define DDRSS1_V2A_HPT_DEF_PRI_MAP_REG    (DDRSS1_EMIF_EW_SSCFG + 0x4C)
+#define DDRSS2_V2A_LPT_DEF_PRI_MAP_REG    (DDRSS2_EMIF_EW_SSCFG + 0x30)
+#define DDRSS2_V2A_HPT_DEF_PRI_MAP_REG    (DDRSS2_EMIF_EW_SSCFG + 0x4C)
+#define DDRSS3_V2A_LPT_DEF_PRI_MAP_REG    (DDRSS3_EMIF_EW_SSCFG + 0x30)
+#define DDRSS3_V2A_HPT_DEF_PRI_MAP_REG    (DDRSS3_EMIF_EW_SSCFG + 0x4C)
+
+/*
+ * New thread priority mapping to remove complete priority equalization
+ * of threads coming from VBUSM space to DDRSS space.
+ */
+#define DDRSS_V2A_LPT_DEF_PRIMAP    0x23456677
+#define DDRSS_V2A_HPT_DEF_PRIMAP    0x00112345
+
 struct fwl_data infra_cbass0_fwls[] = {
     { "PSC0", 5, 1 },
     { "PLL_CTRL0", 6, 1 },
@@ -123,6 +144,19 @@ static void setup_navss_nb(void)
     writel(NB_THREADMAP_BIT2, (uintptr_t)NAVSS0_NBSS_NB1_CFG_NB_THREADMAP);
 }

+/* Setup DDRSS CoS (Class of Service) registers to remove priority equalization */
+static void setup_ddrss_cos(void)
+{
+    writel(DDRSS_V2A_LPT_DEF_PRIMAP, (uintptr_t)DDRSS0_V2A_LPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_HPT_DEF_PRIMAP, (uintptr_t)DDRSS0_V2A_HPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_LPT_DEF_PRIMAP, (uintptr_t)DDRSS1_V2A_LPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_HPT_DEF_PRIMAP, (uintptr_t)DDRSS1_V2A_HPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_LPT_DEF_PRIMAP, (uintptr_t)DDRSS2_V2A_LPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_HPT_DEF_PRIMAP, (uintptr_t)DDRSS2_V2A_HPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_LPT_DEF_PRIMAP, (uintptr_t)DDRSS3_V2A_LPT_DEF_PRI_MAP_REG);
+    writel(DDRSS_V2A_HPT_DEF_PRIMAP, (uintptr_t)DDRSS3_V2A_HPT_DEF_PRI_MAP_REG);
+}
+
```

```
 /* Execute and check results of BIST executed on MCU1_x and MCU4_O */
 static void run_bist_j784s4(struct udevice *dev)
 {
@@ -328,6 +362,7 @@ void board_init_f(ulong dummy)
         setup_navss_nb();

     setup_qos();
+    setup_ddrss_cos();
 }

 u32 spl_mmc_boot_mode(struct mmc *mmc, const u32 boot_device)
--
2.34.1
```

4 总结

本应用手册概述了 Jacinto 器件中的不同 QoS 机制（尤其是 TDA4VH），阐明了如何使用这些机制在不同 IP 之间实现负载平衡。顺序 ID 会影响事务处理所采用的路径及其实时和非实时分配。为事务分配不同的优先级和实时/非实时分配会影响其接受服务的顺序，并用于平衡和优化所有应用及其各自硬件的性能。

在该案例研究中，我们观察到 C7x_4 内核事务会使 DSS 事务停滞并导致显示子系统内出现同步丢失错误。DSS 事务的优先级高于 VBUSM 和 VBUSM.C 空间内的 C7x_4 内核事务，但在 DDRSS 内，所有优先级都受到了均衡处理。移除此均衡并遵守 DDR 控制器内的 VBUSM.C 优先级后，消除了同步丢失错误。

5 参考资料

1. 德州仪器 (TI)，*J784S4 J742S2 技术参考手册*，技术参考手册。
2. 德州仪器 (TI)，*J721E DRA829/TDA4VM 处理器器件版本 2.0、1.1 技术参考手册*，技术参考手册
3. 德州仪器 (TI)，*PROCESSOR-SDK-LINUX-J784S4*，软件开发套件
4. 德州仪器 (TI)，*PROCESSOR-SDK-RTOS-J784S4*，软件开发套件
5. 德州仪器 (TI)，*使用 CP Tracers V2 进行流量分析*，TI 软件下载
6. 德州仪器 (TI)， *0001-arm-mach-k3-j784s4-Remove-priority-equalization-for-.patch*，ti-u-boot-2023.04.y 补丁
7. 德州仪器 (TI)，*0001-arm-mach-k3-j784s4-Remove-priority-equalization-for-.patch*，ti-u-boot-2023.04.y 补丁
8. 德州仪器 (TI)，*0001-arm-mach-k3-j784s4-Remove-priority-equalization-and-.patch*，ti-u-boot-2025.01.y 补丁
9. 德州仪器 (TI)，*0001-arm-mach-k3-j784s4-Remove-priority-equalization-and-.patch*，ti-u-boot-2025.01.y 补丁

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