参考資料

Application Note

TDA4x および AM6x デバイスにおける DDR 帯域最適化のための QoS および CoS 設定の調整

Jared McArthur

概要

TDA4x および AM6x デバイスには、特定のトランザクションに優先度を与え、さまざまなアプリケーションや IP 間で負荷をバランスさせるためのサービス品質 (QoS) スキームが備わっています。QoS 設定がデバイスやアプリケーションに合わせて調整されていない場合、予期しない、または望ましくない動作が発生する可能性があります。この資料で取り上げているケース スタディでは、この調整不足によって生じる望ましくない動作を示しています。自動バレー パーキング (AVP) デモが実行されているときに、ディスプレイに同期喪失エラーが発生します。これは、ディスプレイに優先度を確保するための適切な QoS およびサービスのクラス (CoS) 設定が不足していることが原因です。DDR コントローラ内の CoS 設定が正しい値に設定されると、同期喪失の問題は確認されなくなります。

目次

商標

1 TDA4VH 内でのデータ移動

注

セクション **3** をお読みください。詳細については、TDA4VH TRM のシステム インターコネクトを参照してください。

トランザクションがどのように負荷分散され、優先順位付けされるかを理解するには、システム インターコネクトとデータの流れについて大まかに把握しておくことが重要です。以下のケーススタディでは、主にディスプレイ サブシステム (DSS) と C7x から DDR サブシステム (DDRSS) へのデータ ルーティングに焦点を当てますが、すべてのイニシエータとターゲットについて大まかに理解しておくことが望ましいです。

図 1-1 (TDA4VH TRM から取得) は、イニシエータとターゲットの概略図と、要求の方向を示します。DSS は「イニシエータ」ブロック内にあります。TDA4VH TRM 内のセクション **3.2.6「イニシエータ — ターゲット接続」**には、システム内のイニシエータとターゲットとの関係をさらに詳しく説明する**コネクティビティ マトリクス**が含まれています。

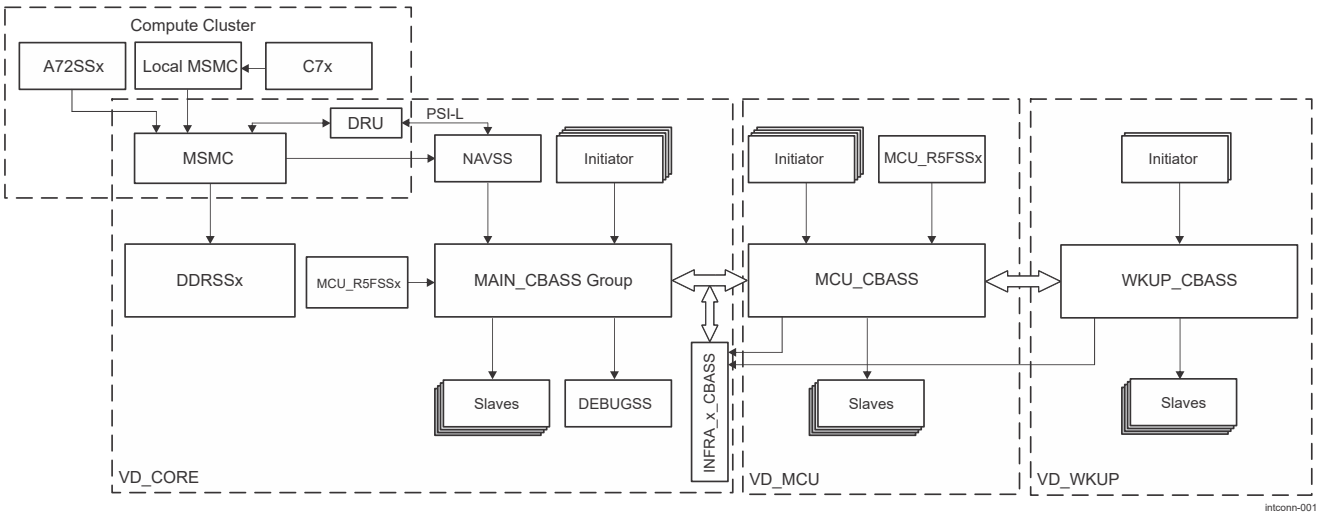

図 1-1. デバイス システム インターコネクトの概要

C7x から DDR へのリクエストは、主に次の経路をたどります:C7x → MSMC → DDRSS

DSS から DDR へのリクエストは、主に次の経路をたどります:DSS → メイン CBASS → NAVSS → MSMC → DDRSS

1.1 共通バス アーキテクチャ サブシステム (CBASS)

システムでは、デバイス内のすべてのモジュールおよびサブシステムが、システム インターコネクトを介して相互に通信します。次のセクションに分かれています:

- CBASS0 インターコネクト
- INFRA_CBASS0 インターコネクト
- MCU_CBASS0 インターコネクト
- WKUP_CBASS0 インターコネクト

ほとんどのモジュールは、MAIN ドメイン内の CBASS0 インターコネクトに接続されています。CBASS には、後のセクションで説明する一部のモジュールのサービス品質のメカニズムが含まれています。

1.2 ナビゲータ サブシステム (NAVSS)

注

詳細については、TDA4VH TRM のセクション **10.2.10 NAVSS North Bridge (NB)** をお読みください。

注意
このセクションでは、マイコン NAVSS ではなく、Main NAVSS (NAVSS0) について説明します。

NAVSS0 は、以下のコンポーネントで構成されます:

- 統合 DMA サブシステム (UDMASS)
- モジュール サブシステム (MODSS)
- North bridge サブシステム (NBSS)
- 仮想化サブシステム (VirtSS)
- ECC アグリゲータ

図 1-2 (TDA4VH TRM から取得) に、NAVSS0 ハードウェア コンポーネントとその統合を示します。

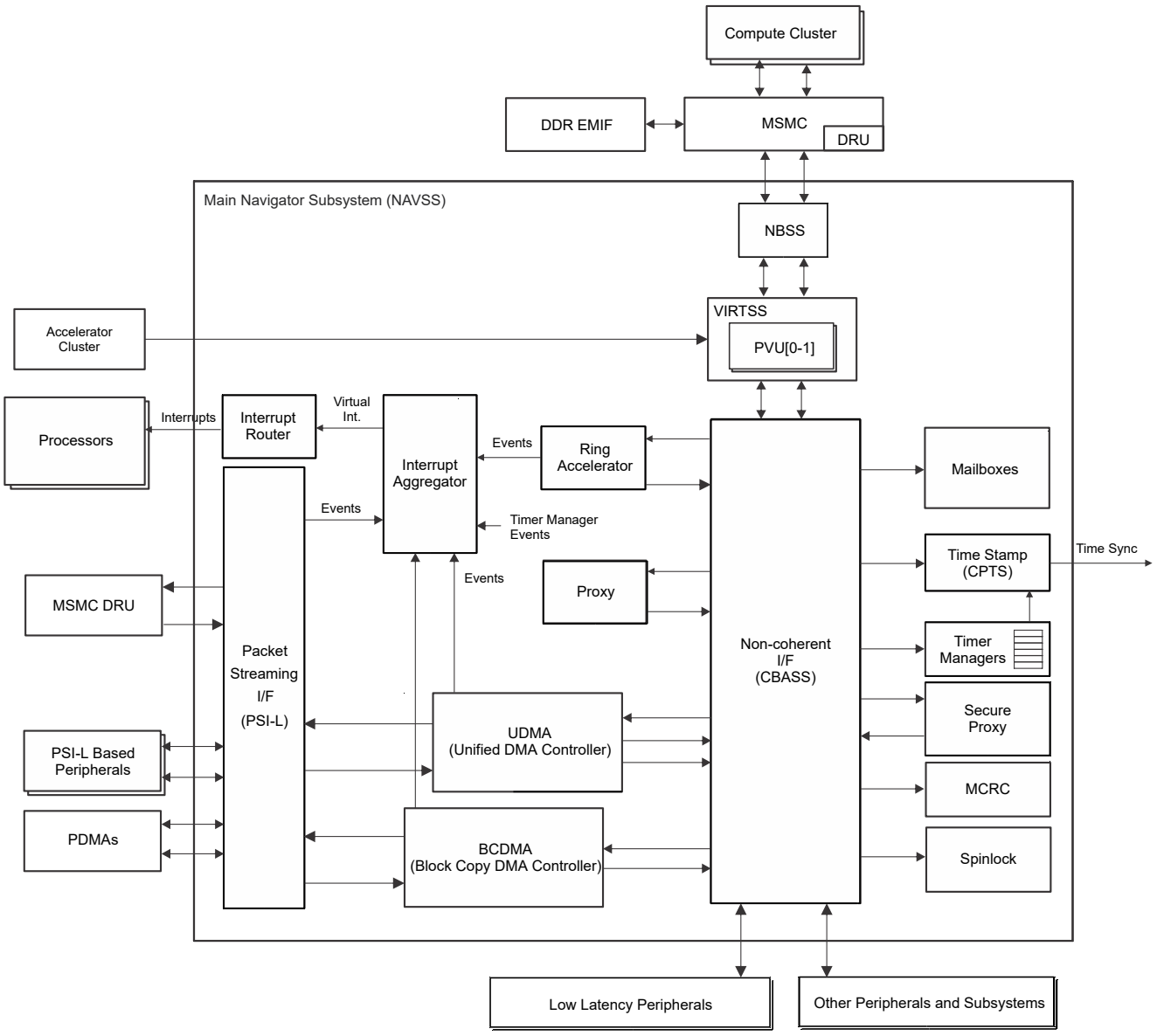

図 1-2. NAVSS0 最上位ブロック図

1.2.1 NAVSS North Bridge (NB)

注

詳細については、TDA4VH TRM のセクション **10.2.10 NAVSS North Bridge (NB)** をお読みください。

NB は、VBUSM インターフェイスと VBUSM.C インターフェイスの間をブリッジします。つまり、CBASS (VBUSM) と MSMC (VBUSM.C) をブリッジします。NAVSS には、2 つの North Bridge が含まれています:(NB0、NB1) で構成されています。

図 1-3 (DRA829/TDA4VM TRM から取得) に、NAVSS0 の NB0 および NB1 のハイレベル構造が表示されています。

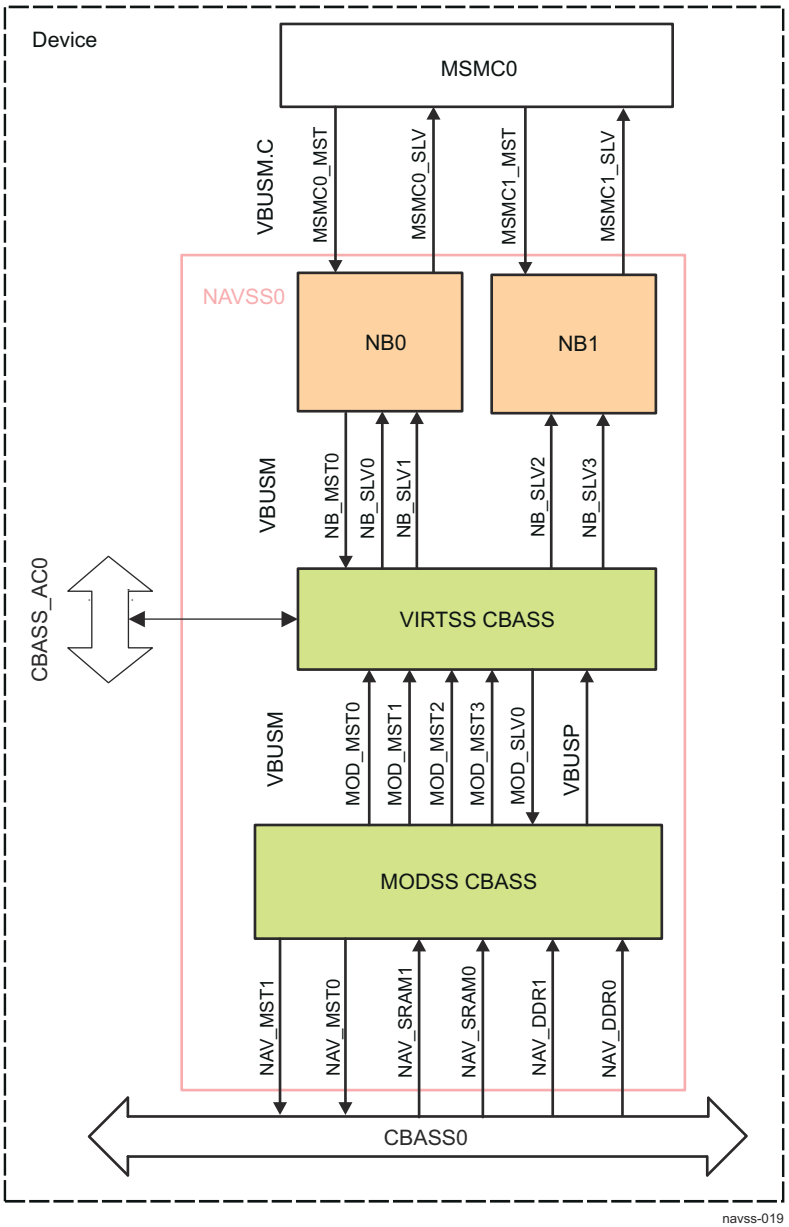

図 1-3. NB の概要

NB のサービス品質メカニズムについては、以降のセクションで説明します。

1.3 マルチコア共有メモリ コントローラ (MSMC)

注

詳細については、TDA4VH TRM の **8.1 マルチコア共有メモリ コントローラ (MSMC)** セクションをお読みください。

MSMC は、コンピュート クラスタ内の内部処理エレメントとシステムの他の部分との間で、高帯域幅のデータ転送およびリソース アクセスを提供します。

図 1-4 (TDA4VH TRM から取得) に、MSMC と周辺モジュールの概要を示します。

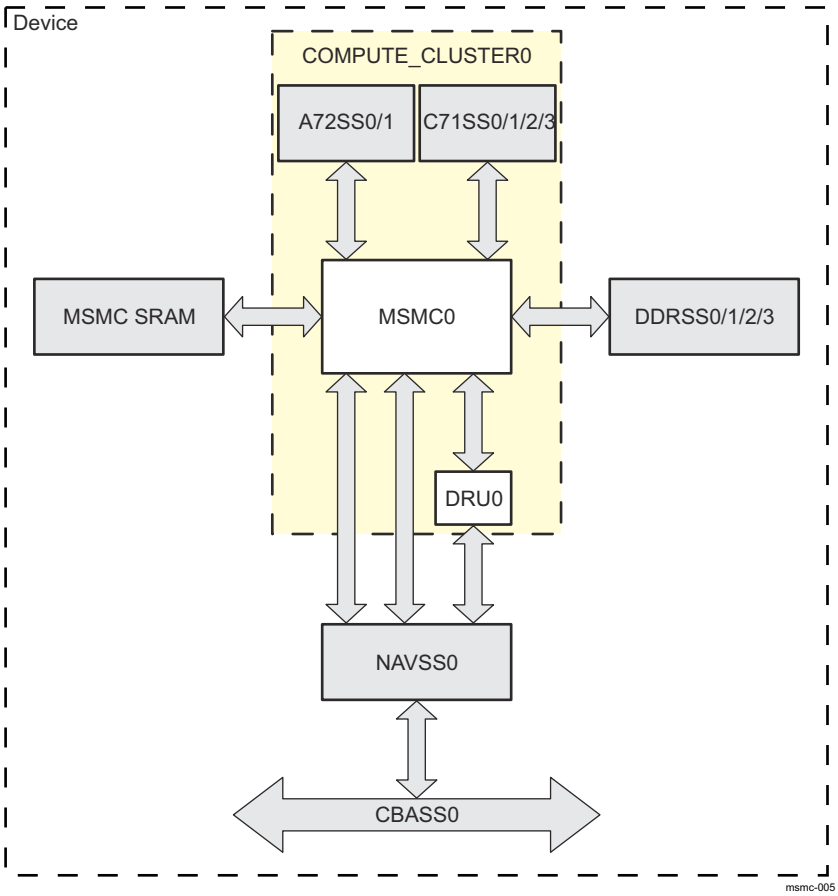

図 1-4. MSMC の概要

MSMC のサービス品質メカニズムについては、以降のセクションで説明します。

2 サービス品質 (QoS)

注

詳細については、TDA4VH TRM のセクション **3.2.1 サービス品質 (QoS)** を参照してください。

サービス品質とは、ネットワーク内のトラフィックを制御するためのメカニズムを使用することです。この場合、ネットワークは TDA4VH デバイスです。システム全体で提供されるサービス品質の方式は 2 つあります。それは order ID と優先度によるアービトレーションです。Order ID は、トランザクションのマスターと、そのマスターのチャネル (複数ある場合) のマッピングを制御します。これにより、異なるパス間でトラフィックを分散できます。優先度は、レイテンシと帯域幅を改善するためのアービトレーション機能を提供します。

トランザクションの優先度は、比較的理解しやすい概念です。優先度は 0 ～ 7 の範囲で、0 は最も高い優先度、7 は最も低い優先度です。優先度の高いトランザクションは、優先度の低いトランザクションよりも先に処理されます。リクエストの優先度の設定方法は IP から IP にかけて異なりますが、通常は優先度を設定するための CTRL MMR またはレジスタが IP 内に用意されています。例えば、DSS の DSS_DISPC_0_COMMON_M_DSS_CBA_CFG レジスタは、DSS リクエストの優先度レベルを制御します。

表 2-1. DSS 優先度レジスタ

登録	フィールド	ビット	説明
DSS_DISPC_0_COMMON_M_DSS_CBA_CFG	PRI_HI	5:3	DSS から CBA へ PRI_HI バスで送信される値は、高優先度 [MFLAG] トランザクションの優先度レベルを示します。 • 0x0 の値は、最も高い優先度を示します • 0x7 の値は、優先度が最も低いことを示します
	PRI_LO	2:0	DSS から CBA へ PRI_LO バスで送信される値は、通常 [non-MFLAG] トランザクションの優先度レベルを示します。 • 0x0 の値は、最も高い優先度を示します • 0x7 の値は、優先度が最も低いことを示します

トランザクションの order ID は、イニシエータのレジスタまたは CBASS レジスタ内で設定できます。order ID の効果は優先度ほど単純ではなく、データ移動アーキテクチャ内のさまざまなサブシステムが、異なるパスを通じてトラフィックをルーティングし負荷を分散するために order ID を使用します。

注

デフォルトでは、すべてのマスタは order ID = 0 でトランザクションを送信します。

2.1 NAVSS0

NAVSS0 内のインターコネクトでは、Order ID を使用して DDR への複数 (少なくとも 2 本) の並列パスと、SRAM への別の複数 (少なくとも 2 本) の並列パスを提供しています。NAVSS0 は、DMA トラフィックを SoC レベルへ送るための複数 (少なくとも 2 本) の並列パスも提供しており、DMA トラフィック用の独立した経路を確保できます。

NAVSS0 内では、2 つの North Bridge が SRAM と DDR のトラフィックを MSMC との間で送受信できるようルーティングします。North Bridge 0 は SRAM トラフィックをルーティングし、North Bridge 1 は DDR トラフィックをルーティングします。

2.1.1 NAVSS0 North Bridge

注

詳細については、TDA4VH TRM のセクション **10.2.10.2.10 サービス品質**を参照してください。

各 North Bridge は、order ID によって分離された複数のソースを受け取ります。North Bridge 0: source 0 が order ID 0 ～ 7 のすべてのトランザクションを受け取り、source 1 が order ID 8 ～ 15 のすべてのトランザクションを受け取ります。North Bridge 1: source 0 が order ID 0 ～ 4 のすべてのトランザクションを受け取り、source 1 が order ID 5 ～ 9、source 2 が order ID 10 ～ 15 のトランザクションを受け取ります。これらの並列パスは、order ID によってトランザクションの負荷を分散します。各ソースが受け取る order ID は、ユーザーが設定することはできません。

注意

各ソースが受け取る order ID は、デバイスによって異なる場合があります。例えば、TDA4VM では North Bridge 0 と 1 の両方で、2 つのソースのみがルーティングされます。

Order ID は、コマンドの順序にも影響します。特定の order ID 値を持つ VBUSM インターフェイスから受信した各読み取りコマンドは、たとえ異なるマスターからのコマンドであっても、読み取りデータがまったく同じ順序で返されます。読み取りが異なる order ID 値を使用している場合、読み取りデータは任意の順序で返される可能性があり、VBUSM.C インターフェイスで先に受信されたものから返されます。

VBUSM.C からの戻りトラフィックを正しい VBUSM ソースへルーティングするために、order ID が使用されます。このため、各ソースの order ID は重複することができません。ただし、各ソースはもともと異なる order ID をルーティングするため、これは問題にはなりません。

2.1.1.1 通常のトラフィックとリアルタイムのトラフィックの関係

North Bridge は、MSMC へのトラフィックを分離するために 3 つのスレッドを使用します:

- スレッド 0: VBUSM.C へのコマンド
- スレッド 1: VBUSM.C からのコマンド
- スレッド 2: VBUSM.C へのリアルタイム コマンド

サービス品質をサポートするために、North Bridge には、ソースをノーマル スレッド (0) またはリアルタイム スレッド (2) のいずれかにマッピングするレジスタが用意されています。リアルタイム スレッドにマップされたソースは、通常のスレッドよりも先にアービトレーションされます。同じスレッドに複数のソースがマッピングされている場合、それらは優先度に基づいてアービトレーションされ、優先度が同じ場合はラウンド ロビン方式で処理されます。このようにして、North Bridge はトラフィックを分割し、order ID によって優先度を割り当てることができます (結果として、利用可能な優先度の数を拡張することにもなります)。

order ID をノーマル スレッドにマッピングするかリアルタイム スレッドにマッピングするかの設定は、NAVSS North Bridge の MMR レジスタで行われ、具体的には NAVSS_NORTH_x_NBSS_NBx_MMRS_threadmap (x は 0 または 1 を示す) レジスタで設定されます。

表 2-2. NAVSS North Bridge Threadmap レジスタ

登録	フィールド	ビット	説明
NAVSS_NORTH_0_NBSS_NB0_MMRS_threadmap	予約済み	31:3	予約済み
	THREADMAP	1	order ID 8 ～ 15 を VBUSM.C のスレッド番号にマッピングします: • 0:VBUSM.C スレッド 0 (非リアルタイム トラフィック) • 1:VBUSM.C スレッド 2 (リアルタイム トラフィック)
		0	order ID 0 ～ 7 を VBUSM.C のスレッド番号にマッピングします: • 0:VBUSM.C スレッド 0 (非リアルタイム トラフィック) • 1:VBUSM.C スレッド 2 (リアルタイム トラフィック)
NAVSS_NORTH_1_NBSS_NB1_MMRS_threadmap	予約済み	31:3	予約済み
	THREADMAP	2	order ID 10 ～ 15 を VBUSM.C のスレッド番号にマッピングします: • 0:VBUSM.C スレッド 0 (非リアルタイム トラフィック) • 1:VBUSM.C スレッド 2 (リアルタイム トラフィック)
		1	order ID 5 ～ 9 を VBUSM.C のスレッド番号にマッピングします: • 0:VBUSM.C スレッド 0 (非リアルタイム トラフィック) • 1:VBUSM.C スレッド 2 (リアルタイム トラフィック)
		0	order ID 0 ～ 4 を VBUSM.C のスレッド番号にマッピングします: • 0:VBUSM.C スレッド 0 (非リアルタイム トラフィック) • 1:VBUSM.C スレッド 2 (リアルタイム トラフィック)

注意

NAVSS_NORTH_x_NBSS_NBx_MMRS_threadmap レジスタのフィールドは、デバイスによって異なります。上の表は TDA4VH を代表したものです。

2.2 マルチコア共有メモリ コントローラ (MSMC)

注

詳細については、TDA4VH TRM のセクション **8.1.2.11 MSMC のサービス品質**を参照してください。

MSMC には、リアルタイム (RT) とリアルタイム以外 (NRT) の 2 つのトラフィック クラスがあります。これら 2 種類のトラフィックは、North Bridge 内のリアルタイム スレッド (2) と通常スレッド (0) に対応します。MSMC では、各アービトレーション

ポイントに、RT トラフィックのみが使用できる専用バッファが用意されています。そのため、NRT トラフィックによって RT リクエストが完全に枯渇することはありません。

MSMC QoS ハードウェアに対するソフトウェア制御はありません。QoS 機能をサポートしていないインターフェースでは、すべてのトラフィックが非リアルタイムとして扱われます。

2.3 DDR サブシステム (DDRSS)

DDRSS には、MSMC2DDR ブリッジを介したサービス品質メカニズムがあり、さらに独自のメカニズムであるサービス クラス (CoS) も備わっています。DDR コントローラは、CoS メカニズムを使用して、VBUSM.C スレッドおよび優先度を内部のスレッドおよび優先度にマッピングします。

2.3.1 MSMC2DDR ブリッジ

注

詳細については、TDA4VH TRM の **8.2.3.1 DDRSS MSMC2DDR ブリッジ**および **8.2.3.1.1 VBUSM.C スレッド**の各セクションを参照してください

MSMC2DDR ブリッジは、次の 2 つのスレッドをサポートします:

- 高優先度スレッド (HPT): VBUSM.C スレッド 2 で受信されたトラフィックは HPT に属します
- 低優先度スレッド (LPT): VBUSM.C スレッド 0 で受信されたトラフィックは LPT に属します

HPT は LPT よりも優先度が高く、コマンド キューからのコマンド実行は順が異なる場合があります。これにより、LPT がブロック済みでも HPT の実行が保証されます。

MSMC2DDR ブリッジはスレッド間でデータのコヒーレンシーを維持するため、優先度逆転が発生する可能性があります。アドレス競合により LPT トランザクションに依存する HPT トランザクションは、対応する LPT トランザクションが実行されるまでブロックされます。

2.3.2 サービス クラス (CoS)

注

詳細については、TDA4VH TRM のセクション **8.2.3.1.2 サービスクラス (CoS)** をお読みください。

サービス クラスは DDRSS 固有であり、システム (VBUSM.C) の優先度が DDRSS 内部優先度にどのようにマップされるかを制御します。MSMC2DDR ブリッジには、VBUSM.C の優先度を DDR コントローラの優先度に対応付けるための次のレジスタがあります:

- 範囲一致レジスタ:
  - DDRSS_V2A_R1_MAT_REG
  - DDRSS_V2A_R2_MAT_REG
  - DDRSS_V2A_R3_MAT_REG
- 優先マップ レジスタ:
  - 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

図 2-1 (TDA4VH TRM から取得) に、優先度マップレジスタが受信優先度を適切な DDR 優先度にマッピングする方法を示します。

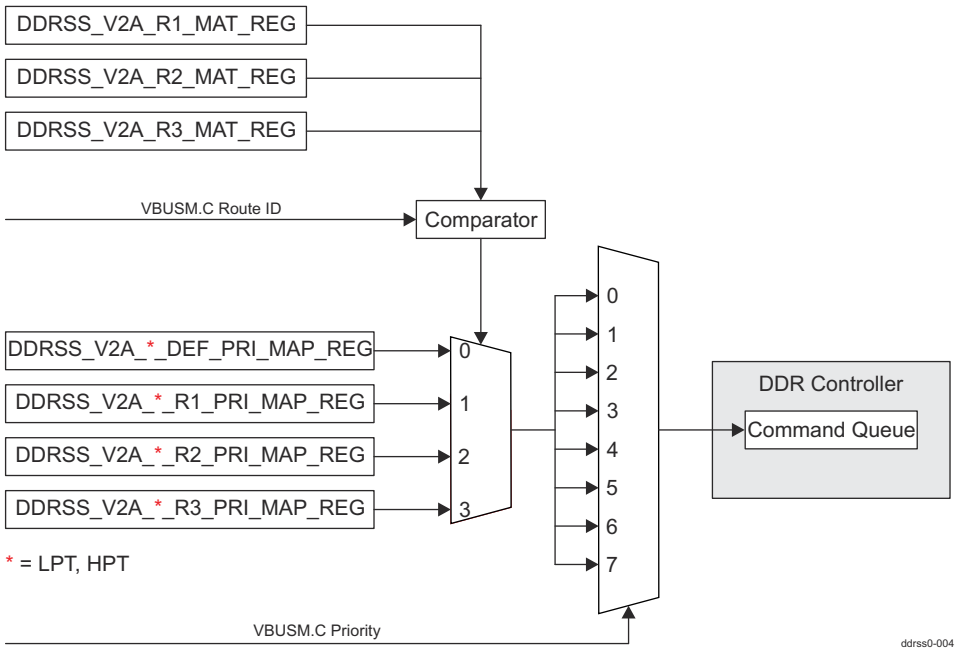

図 2-1. DDRSS CoS マッピング図

2.4 QoS の概要

QoS を制御するメカニズムと IP について説明してきましたので、ここでそれらの詳細をまとめ、高レベルの観点からその意味を理解することが重要です。

前述したように、QoS は 2 つのメカニズム (order ID と優先度) によって実現されます。Order ID は、リクエストに対して設定可能なフィールドであり、リクエストが異なる IP 間でどのようにルーティングされるかに影響します。オーダー ID の主な目的は、異なるパス間でデータの流れを分散させるためのメカニズムを提供することです。優先度は、リクエストが処理されるオーダーを制御します。

データ移動アーキテクチャ サブシステムは、非リアルタイム (NRT) スレッドとリアルタイム (RT) スレッドの概念も導入しました。NRT および RT 属性の名前は、IP から IP にかけて異なります。

表 2-3. IP 間でのリアルタイム以外およびリアルタイムの命名

IP	リアルタイム以外 (NRT)	リアルタイム (RT)
MSMC	リアルタイム以外	リアルタイム
NAVSS North Bridge	スレッド 0 (標準)	スレッド 2 (リアルタイム)
DDR コントローラ	低優先度スレッド (LPT)	高優先度スレッド (HPT)

RT スレッドは NRT スレッドより先にアービトレーションされるため、RT スレッド上のトランザクションにはより高い優先度が与えられます。トランザクションが NRT スレッドに入るか RT スレッドに入るかは、order ID と North Bridge のプログラミング方法によって決まります。NRT スレッドと RT スレッドの概念を優先度と組み合わせると、要求の階層を次の表に示します。

表 2-4. 非リアルタイムおよびリアルタイムリクエスト間の優先度レベル

NRT または RT リクエスト	優先順位	優先レベル
リアルタイム	0	最高の優先順位
	1	優先度は、リストの下に行くほど低くなります
	2	
	3	
	4	
	5	
	6	
	7	
リアルタイム以外	0	
	1	
	2	
	3	
	4	
	5	
	6	
	7	最も低い優先度

3 ケース スタディ: ディスプレイ同期喪失の問題

では、QoS メカニズムがデバイス全体の負荷をどのように分散できるかを示すケース スタディを見ていきましょう。

3.1 問題提起

4k ディスプレイを接続した状態で AVP (自動バレー パーキング) デモを実行すると、ディスプレイで同期の損失が頻繁に発生します。

3.2 セットアップと再現

3.2.1 要件

表 3-1. ハードウェアとソフトウェアの要件

項目	リンク	コメント
J784S4XEVM	https://www.ti.com/tool/J784S4XEVM	該当なし
SD カード	該当なし	該当なし
4K 30fps モニタ	該当なし	モニタを Display Port1 に接続します
PROCESSOR-SDK-RTOS-J784S4	ti-processor-sdk-rtos-j784s4-evm-09_02_00_05.tar.gz	バージョン 09.02
SOC 汎用 TI サンプル入力データ セット	psdk_rtos_ti_data_set_09_02_00.tar.gz	PROCESSOR-SDK-RTOS-J784S4 ページ内に記載されています
SOC 固有の tidl モデル	psdk_rtos_ti_data_set_09_02_00_j784s4.tar.gz	PROCESSOR-SDK-RTOS-J784S4 ページ内に記載されています
PROCESSOR-SDK-LINUX-J784S4	ti-processor-sdk-linux-adas-j784s4-evm-09_02_00_05-Linux-x86-Install.bin	バージョン 09.02
RTOS パッチ	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	vision_apps リポジトリに適用するパッチ
Linux パッチ	linux-patches.tar.xz • 0001-arm64-dts-ti-k3-j784s4-vision-apps-Re-enable-DSS-for.patch	ti-linux-kernel リポジトリに適用するためのパッチ

3.2.1.1 RTOS パッチ

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 パッチ

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 ホスト セットアップ

ホスト PC で次のコマンドが実行されます。

すべての依存関係をインストールします。

```
# 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
```

ビルド変数をエクスポートします。

```
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>
```

PSDK RTOS を設定します。

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

SD カードをセットアップします。この例では、SD カードが /dev/sdb にあることを前提としています。

```
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
```

デモ アプリケーションを編集、ビルド、インストールします。

```
# 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
```

デバイス ツリーを編集、ビルド、およびインストールします。

```
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 ターゲット セットアップ

以下のコマンドは、ターゲット上で実行されます。

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

3.2.4 再現

以下の写真は、kmstest を AVP デモと同時に実行した状態で撮影されたものです。

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

AVP デモを実行すると、画面で同期の損失が発生します。これにより、ディスプレイに予期しないアーチファクトが発生します。

図 3-1. kmstest と AVP デモ実行時の同期の損失

3.3 QoS のデバッグ

3.3.1 CPTracer

CCS (および Lauterbach) は CPTracer と呼ばれるツールを提供しており、開発者はシステム内のトラフィックのプロファイルを作成できます。この場合は、DDR トラフィックをプロファイリングします。

3.3.1.1 構成

1. セクション 7.4 内の CCS を設定する手順に従います。プロセッサ SDK RTOS の A72 (Linux/QNX) で動作する HLOS によるデバッグ 。

注意

ここで説明している PSDK RTOS の特定のバージョンは、以下のとおりです: PSDK RTOS J784S4 10.01.00

2. トラフィック プロファイリング ウィンドウは、SoC Analysis タブから開くことができます。

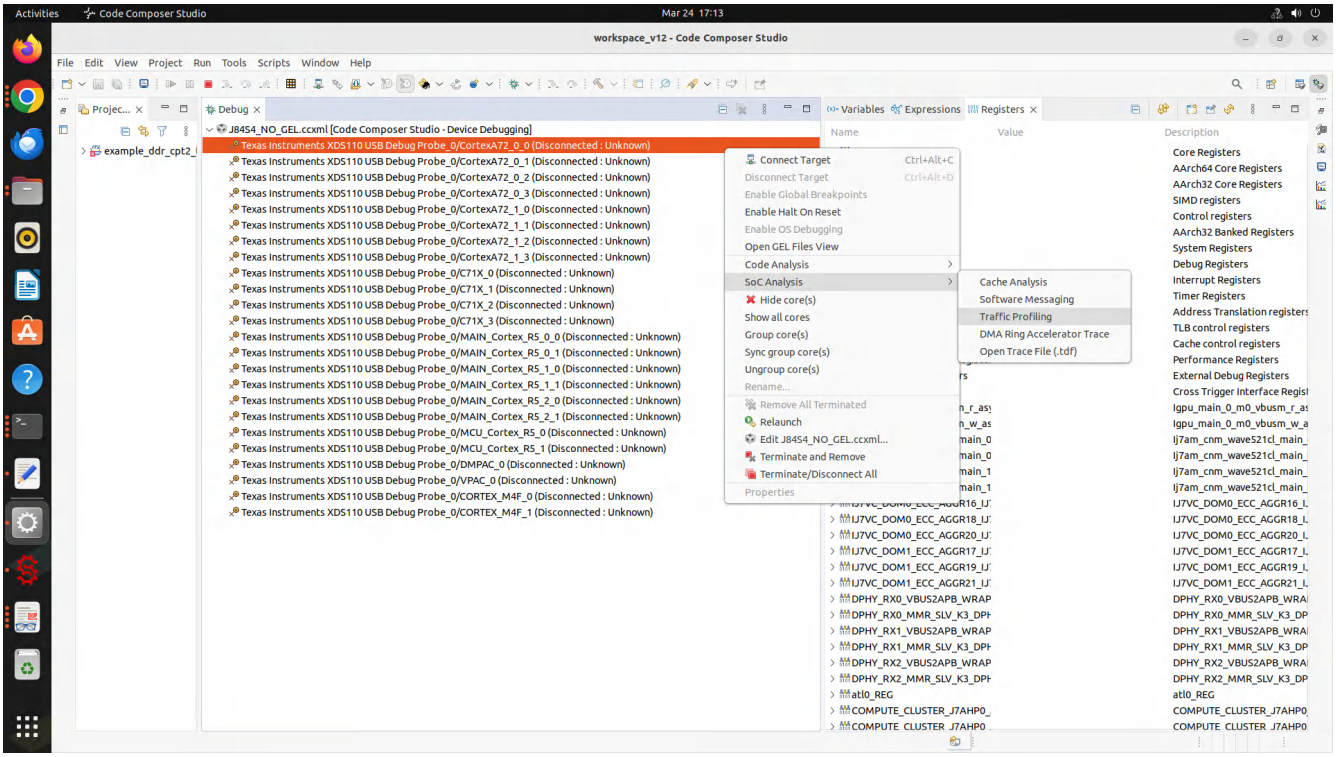

図 3-2. CCS でのトラフィック プロファイリングの選択

3. CPTracer を開くと表示されるウィンドウです:

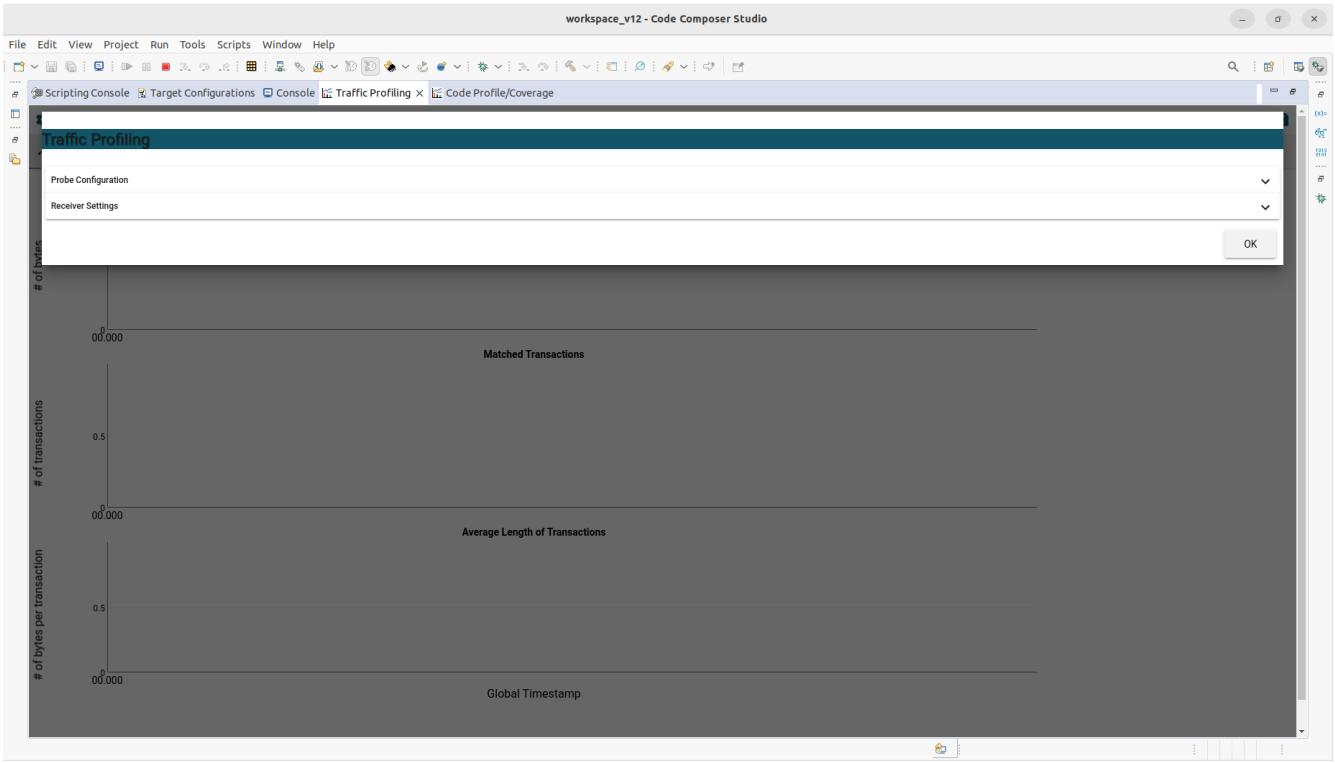

図 3-3. CCS における CPTracer の設定

4. どのトランザクションをプロファイル対象とするかを絞り込むためのフィルタが多数用意されています。これらのフィルタは、対応するイニシエータの歯車アイコンをクリックして編集できます。

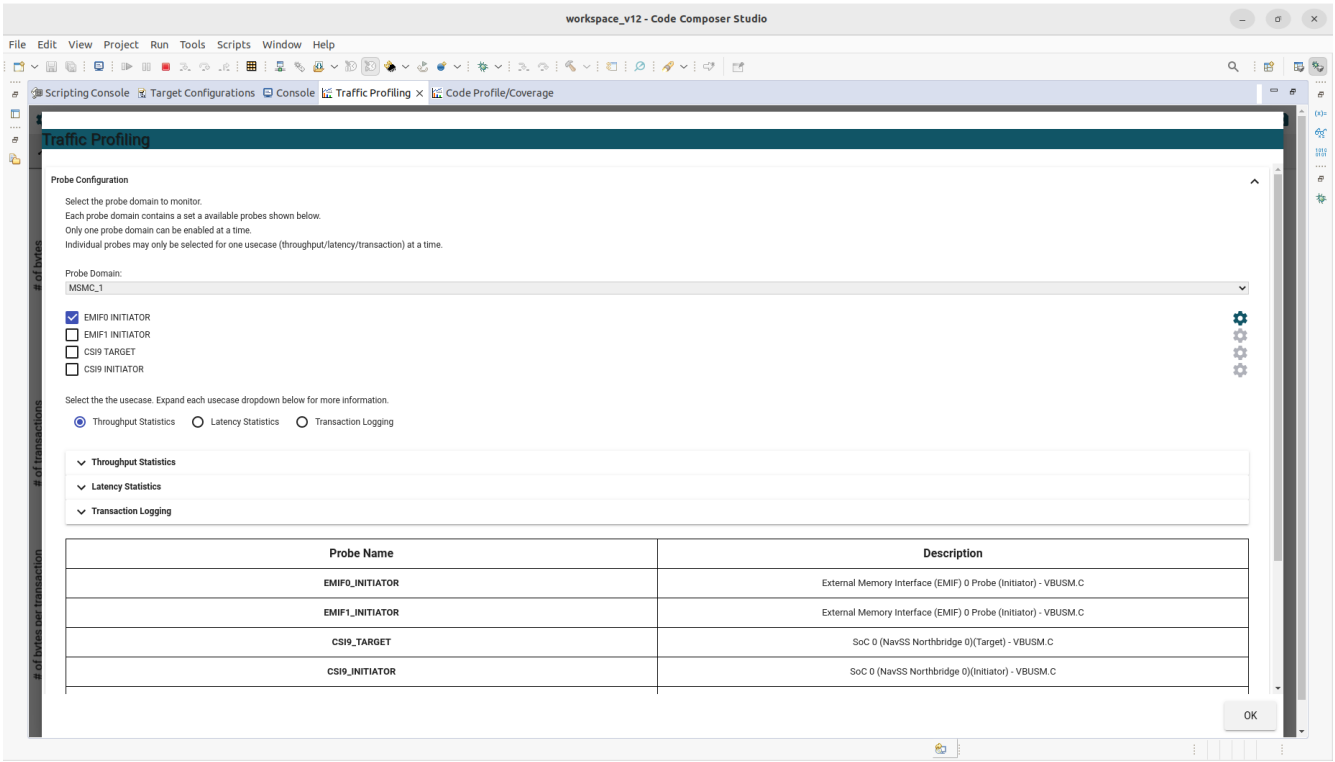

図 3-4. CCS における CPTracer フィルタ

3.3.1.2 プロファイリングのスループット

CPTracer は、一定期間内のトランザクションの総スループットをプロファイルする方法を提供します。csv がエクスポートされると、次のように出力されます:

- **マスタ ID**: トランザクションを実行するイニシエータの ID
- **マスタ名**: イニシエータの名前 (マスタ ID に対応)
- **データ メッセージ**: 行の説明 (クロック サイクル内の周期を含む)
- **グローバル タイム スタンプ**: サンプル ウィンドウが閉じている時間 (GTC (200MHz クロック) に基づく)
- **トレース ステータス**: トレースの開始および終了時を記録します
- **バイト トランザクション**: 設定されたフィルタに一致するプローブで観測された合計バイト数 (期間内に送信されたバイト数)
- **一致**: 設定されたフィルタに一致するプローブで観測されたトランザクションの合計数 (期間内)
- **平均値長さ**: 設定されたフィルタに一致するトランザクションについて、プローブで観測された一定期間内に送信されたトランザクション当たりの平均バイト数

3.3.1.3 プロファイリング レイテンシ

また、CPTracer では、一定期間内のトランザクションの遅延をプロファイルする方法も提供しています。csv がエクスポートされると、次のように出力されます:

- **マスタ ID**: トランザクションを実行するイニシエータの ID
- **マスタ名**: イニシエータの名前 (マスタ ID に対応)
- **データ メッセージ**: 行の説明 (クロック サイクル内の周期を含む)
- **グローバル タイム スタンプ**: サンプル ウィンドウが閉じている時間 (GTC (200MHz クロック) に基づく)
- **トレース ステータス**: トレースの開始および終了時を記録します
- **追跡済み**: イニシエータからの期間内のトランザクションの合計数
- **一致したトランザクション**: 設定されたフィルタに一致するプローブで観測されたトランザクションの合計数 (期間内)
- **最大待機時間**: 設定されたフィルタに一致するプローブで観測された 1 つのトランザクションの最大レイテンシ測定 (期間内)
- **合計待機**: 設定されたフィルタに一致するプローブで観測された合計レイテンシ (すべてのレイテンシ測定値の合計) (期間内)
- **クレジット待機**: 設定されたフィルタに一致するクレジット ベースのバスのプローブで観測された合計クレジット遅延 (一定期間内)

3.3.1.4 トランザクションのプロファイリング

CPTracer は、観察されたトランザクションに関する詳細情報を提供します。CSV をエクスポートすると、次の項目が出力されます (以下は関連する列のみを抜粋した一覧です):

- **ルート ID**: トランザクションのルート ID
- **バイト数**: トランザクションのバースト サイズ
- **優先度**: トランザクションの優先度
- **QOS**: トランザクションのサービス品質
- **Order ID**: トランザクションの order ID

注

カラムの完全なリストは、CPTracer マニュアルの高度プローブ フィルタセクションにあります

3.3.1.5 関連ルートのプロファイリング

現在確認している問題は、C7x が DSS に影響を与えていることによって発生しています。そのため、C7x と DSS のルートをプロファイルします。

DSS および C7x トランザクションのルート ID は次のとおりです:

表 3-2. 関連するイニシエータとルート ID

イニシエータ	ルート ID
C7x_1 コア	0x20
C7x_1 DRU0	0x21
C7x_1 DRU1	0x22
C7x_1 CMMU	0x23
C7x_2 コア	0x24
C7x_2 DRU0	0x25
C7x_2 DRU1	0x26
C7x_2 CMMU	0x27
C7x_3 コア	0x28
C7x_3 DRU0	0x29
C7x_3 DRU1	0x2A
C7x_3 CMMU	0x2B
C7x_4 コア	0x2C
C7x_4 DRU0	0x2D
C7x_4 DRU1	0x2E
C7x_4 CMMU	0x2F
DSS_INST0_VBUSM_DMA	0xA20
DSS_INST0_VBUSM_FBDC	0xA21

注

ルート ID は、TDA4VH TRM の付録に記載されています

CCS 版の CPTracer では、MSMC_1 プローブ ドメイン内に EMIF0 と EMIF1 のみが含まれています (EMIF2 と 3 は含まれていません)。ただし、1 つの EMIF のスループットを 4 倍することで、全体のスループットをおおよそ推定できます。

3.3.1.6 DSS スループットのプロファイリング

DSS トランザクションのみをプロファイルするために使用したフィルタは、次のとおりです:

- **ルート ID 値**:0xA20
- **ルート ID マスク**:0xFFE
- **サンプル ウィンドウ**:0x4000

EMIF0 のみがプロファイルされました。

フレームごとに送信される合計バイト数は、**dss-frame-thru-calc.py** で計算されました

```
# 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 理論上の DSS スループット

理論上の DSS スループット (1 フレーム当たり **265420800** ビット) は、**3840x2160@30fps** の **XR32-888** ディスプレイを基に計算されています。

表 3-3. 理論上の DSS スループット

パラメータ	値
高さ	3840
幅	2160
fps	30
ピクセルあたりのビット数	32
フレームあたりのビット数	265420800
フレームごとの Mib	253.125
データ レート (bps)	7962624000
データ レート (Mibps)	7593.75

3.3.1.6.2 通常の DSS スループット

次の図は、AVP デモを実行していない DSS トランザクションのスループットを示しています。

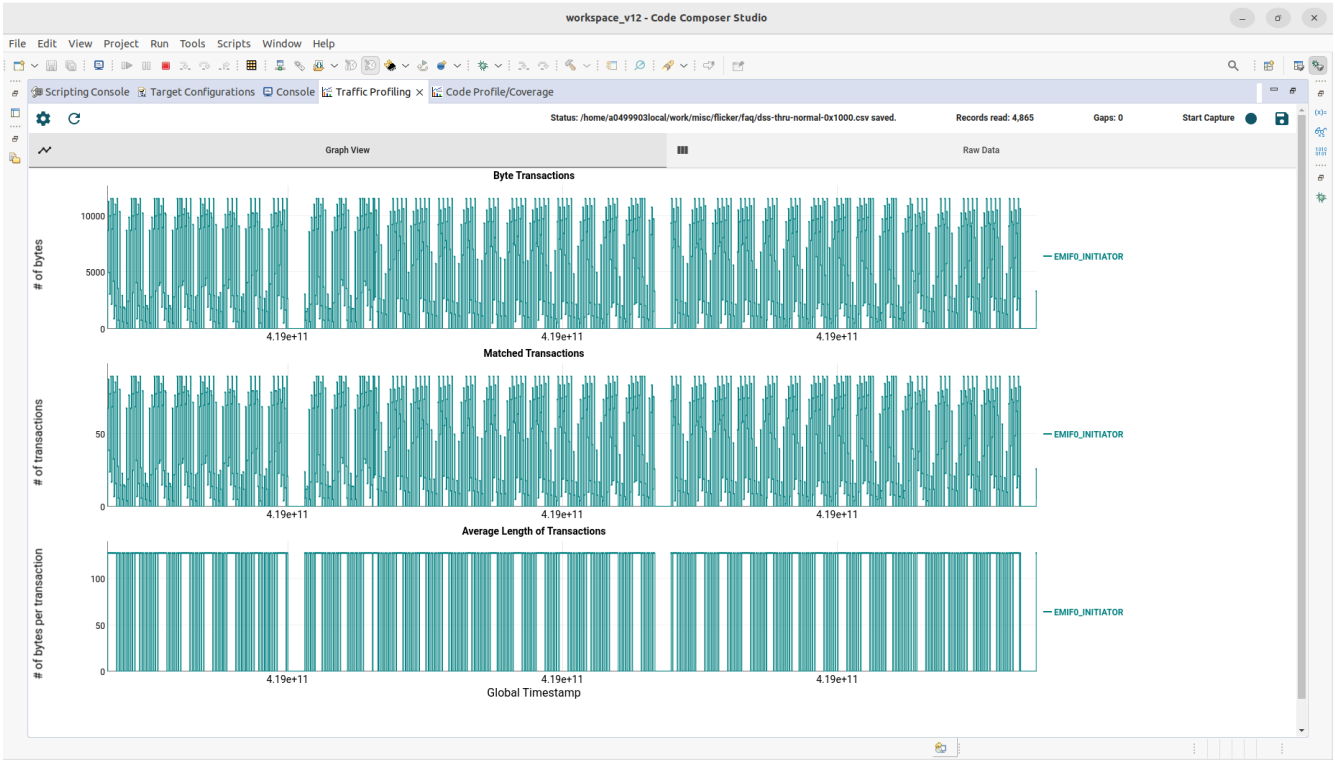

図 3-5. 通常の DSS スループット

スループットの測定値は、1 つのフレーム内に含まれるすべての期間のスループットを合計することで算出されました。測定値は、スループットの CSV を保存し、それらを Python スクリプトで処理することで取得しました。

表 3-4. 通常の測定された DSS スループット

	1 つの EMIF の DSS スループット	4 つの EMIF のスループットを訂正
フレームあたりのバイト数	8294400	33177600
バイトあたりのビット数	8	
fps	30	
測定した EMIF の数	1	4
フレームあたりのビット数	66355200	265420800
フレームごとの Mib	63.28125	253.125
データ レート (bps)	1990656000	7962624000
データ レート (Mibps)	1898.4375	7593.75

1 つのフレームの合計スループットは、**3840x2160@30fps** の画面 (フレームあたり **66355200** ビット) で予想されるスループットの 1/4 に一致します。プロファイルされたのは 4 つの EMIF のうち 1 つだけで、これが予想される結果です。測定値が修正されると、スループットは理論スループット (フレームあたり **265420800** ビット) に等しくなります。

3.3.1.6.3 AVP デモを実行した場合の DSS スループット

次の図は、AVP デモ実行時の DSS トランザクションのスループットを示しています。

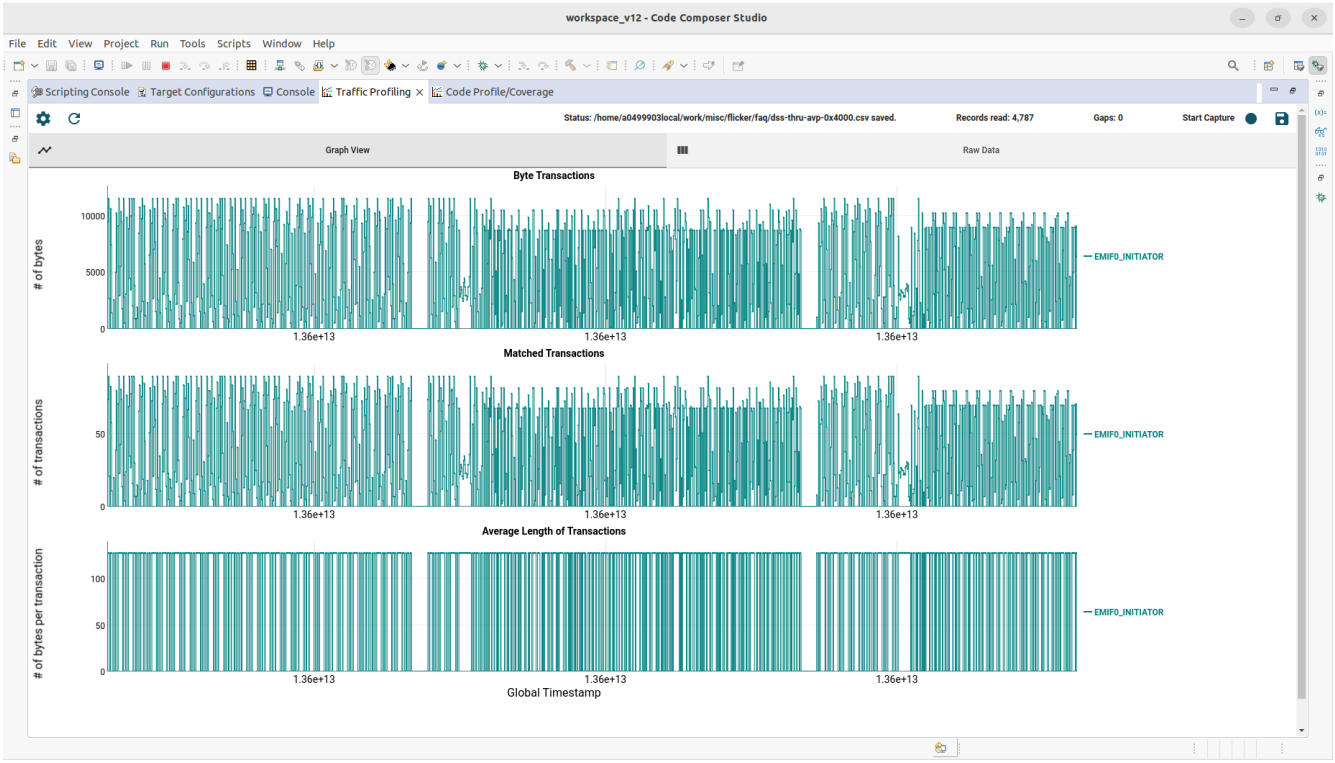

図 3-6. AVP デモ DSS スループット

表 3-5. AVP デモを使用した DSS スループットの測定

	1 つの EMIF の DSS スループット	4 つの EMIF のスループットを訂正
フレームあたりのバイト数	8257536	33030144
バイトあたりのビット数	8	
fps	30	
測定した EMIF の数	1	4
フレームあたりのビット数	66060288	264241152
フレームごとの Mib	63	252
データ レート (bps)	1981808640	7927234560
データ レート (Mibps)	1890	7560

合計スループットは、**3840x2160@30fps** の表示に必要なものには達しません。これが同期が失われる原因です。1 つのフレームで送信されたビット数は、**265420800** ビットの代わりに **264241152** ビットしかありませんでした。

3.3.1.7 DSS レイテンシのプロファイリング

仮に、DSS のトランザクションがストールしている場合、合計レイテンシが大きくスパイクする可能性があります。この仮説を検証するため、AVP デモを実行している場合と実行していない場合の両方で DSS のレイテンシをプロファイルします。

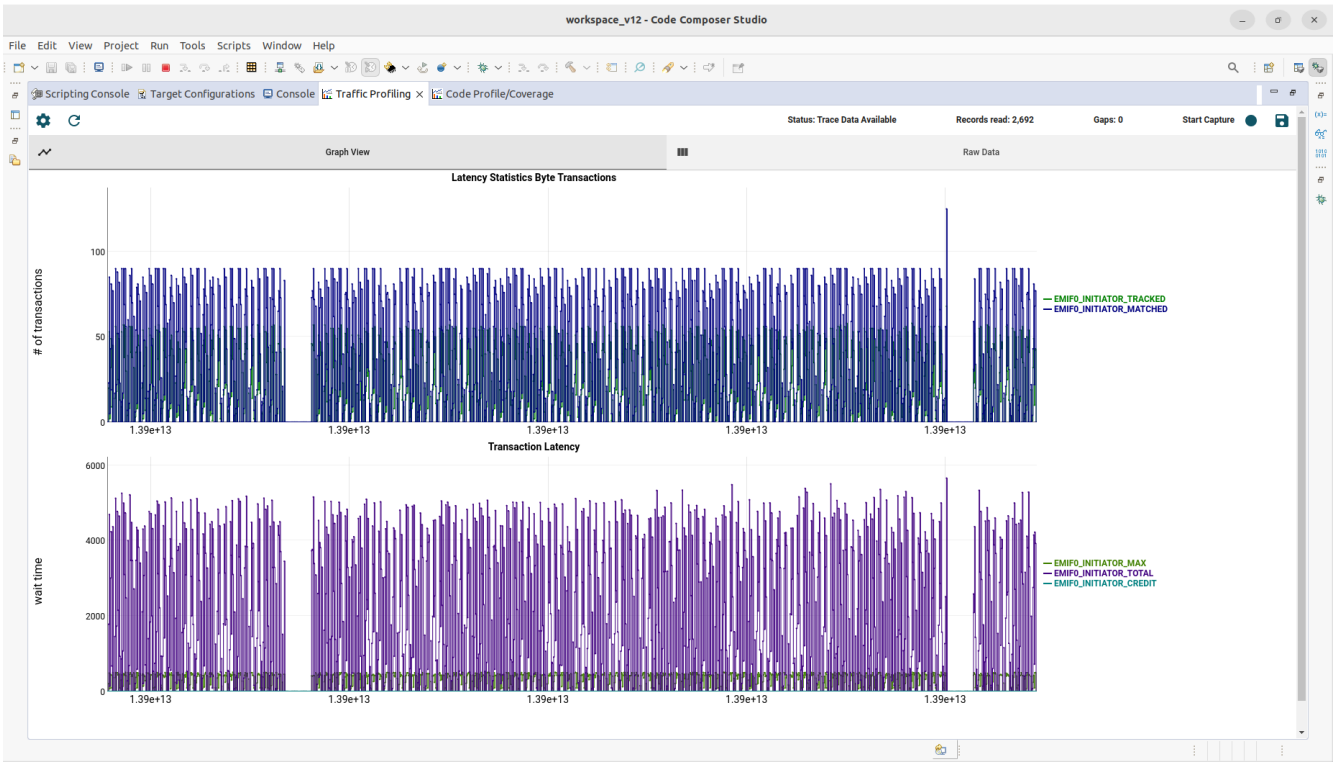

図 3-7. 通常の DSS レイテンシ

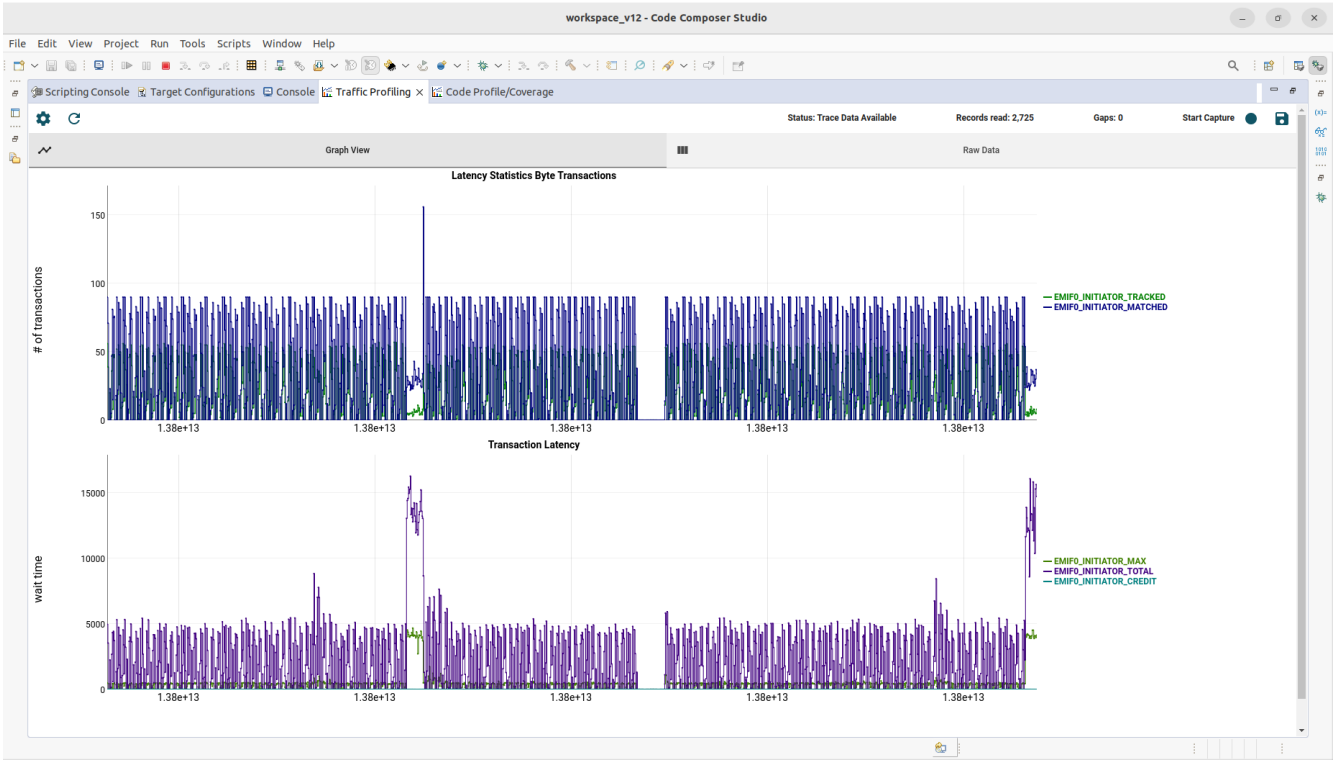

図 3-8. AVP デモ DSS の遅延

レイテンシ グラフを見ると、DSS トランザクションが停止していることは明らかです。これで、C7x のトランザクションをプロファイルし、ストールの原因を絞り込むことができます。

3.3.1.8 C7x スループットのプロファイリング

各 C7x コアのスループットを個別にプロファイリングすることで、DSS のストールの原因を正確に特定できます。

スループットが高く、トランザクション セクションの数が多いと、ストールが発生する可能性があります。

表 3-6 には、個別の C7x コアをフィルタリングするために必要な設定が含まれています。

表 3-6. C7x ルート ID の値とマスク

C7x コア	ルート ID	ルート ID 値	ルート ID マスク
すべてのコア	0x20～0x2F	0x020	0xFF0
1	0x20～0x23	0x020	0xFFC
2	0x24～0x27	0x024	0xFFC
3	0x28～0x2B	0x028	0xFFC
4	0x2C～0x2F	0x02C	0xFFC

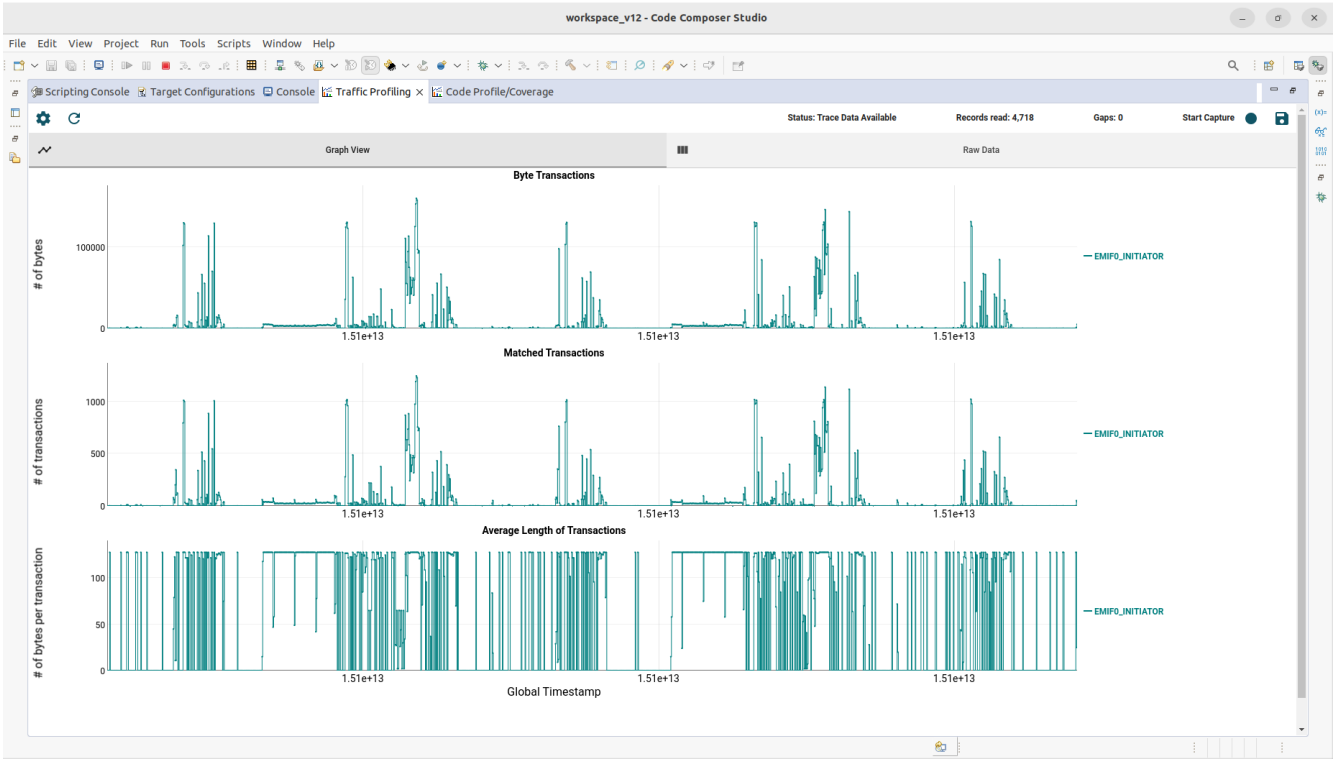

図 3-9. すべての C7x スループット

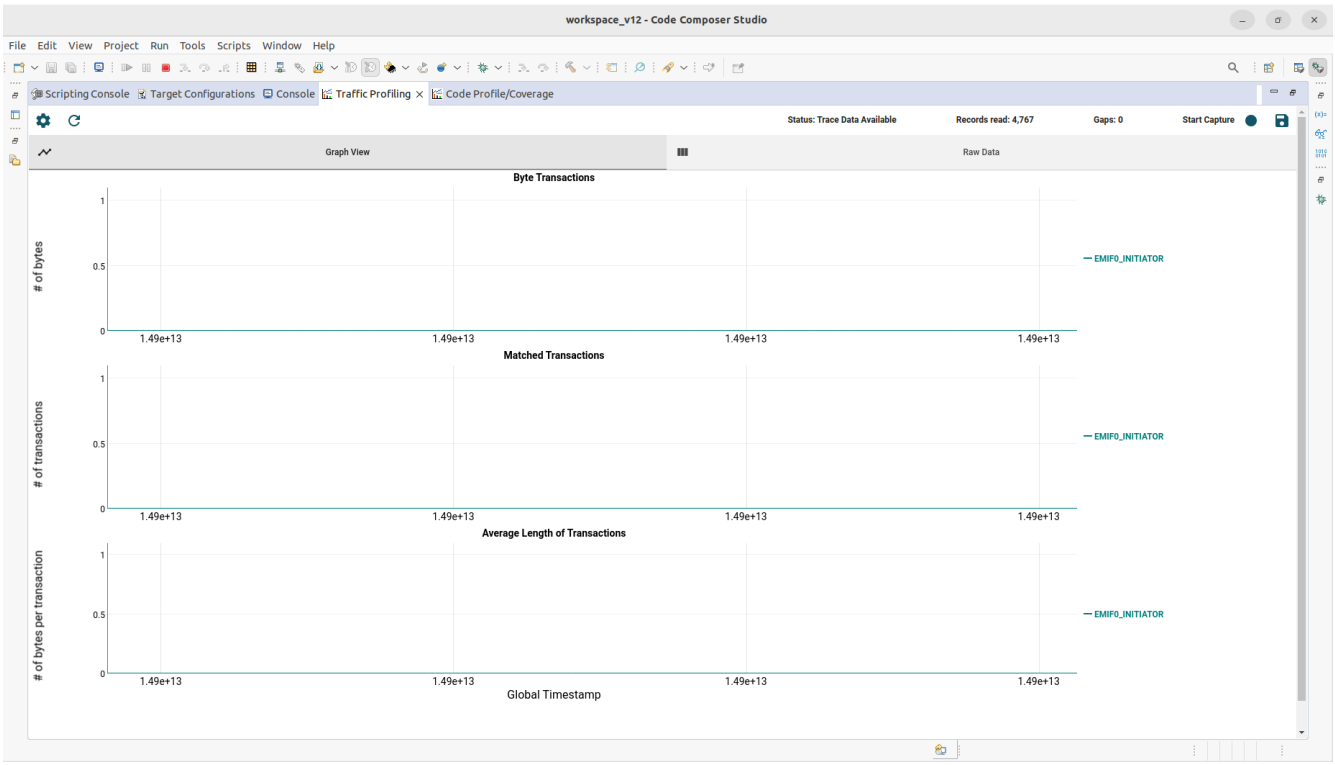

図 3-10. C7x_1 スループット

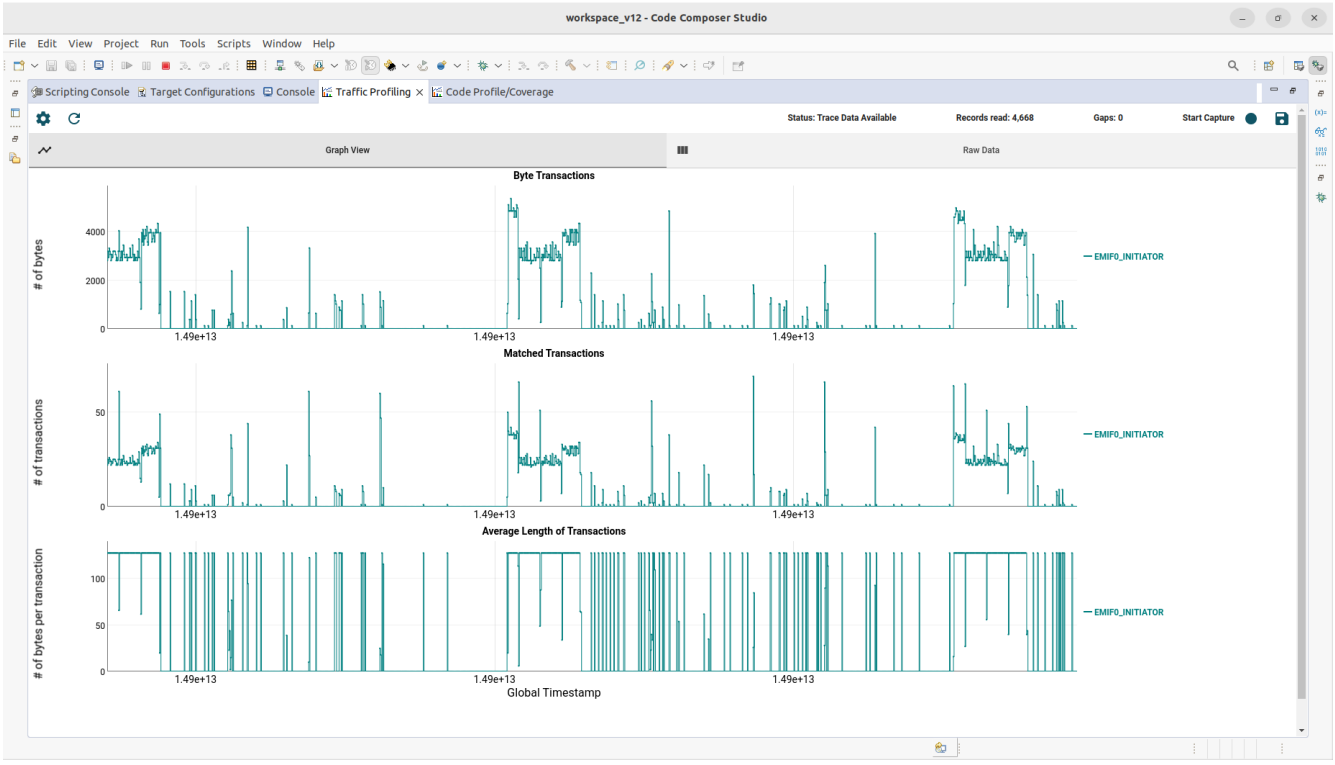

図 3-11. C7x_2 スループット

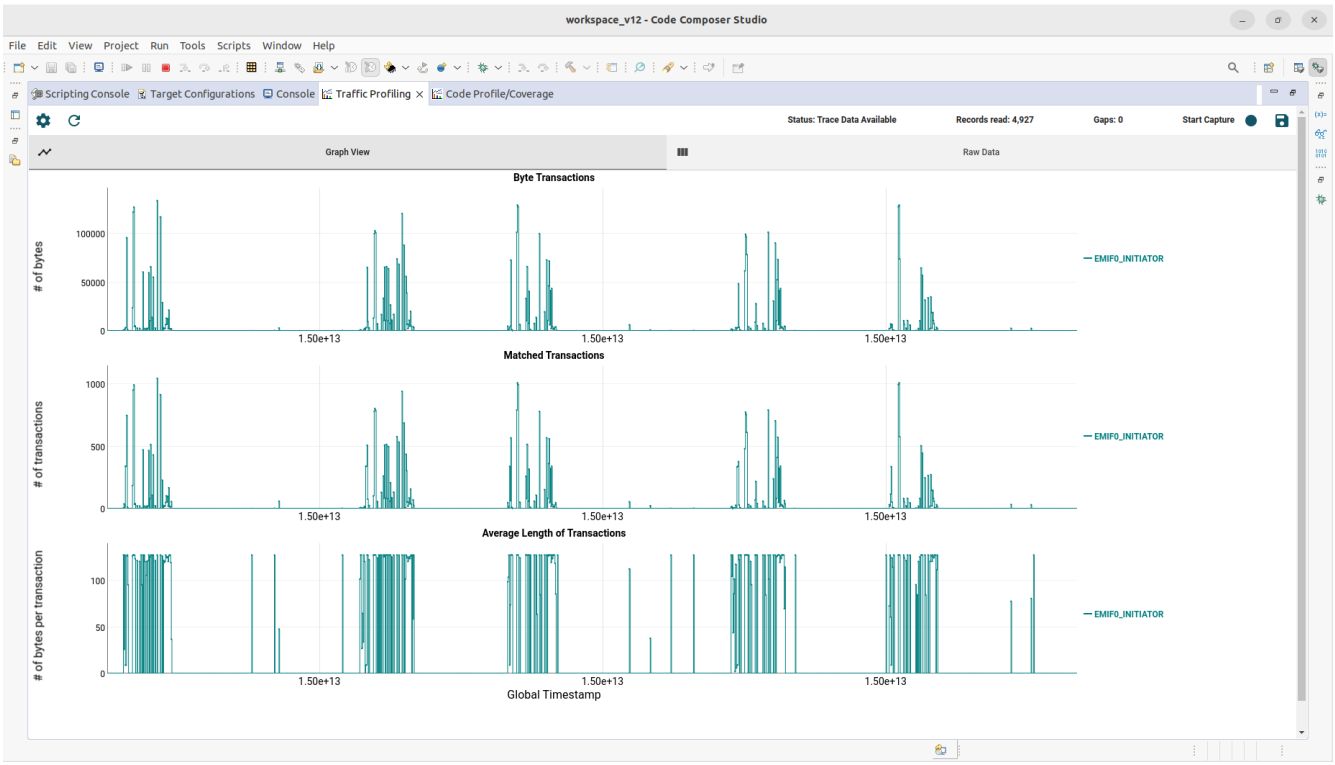

図 3-12. C7x_3 スループット

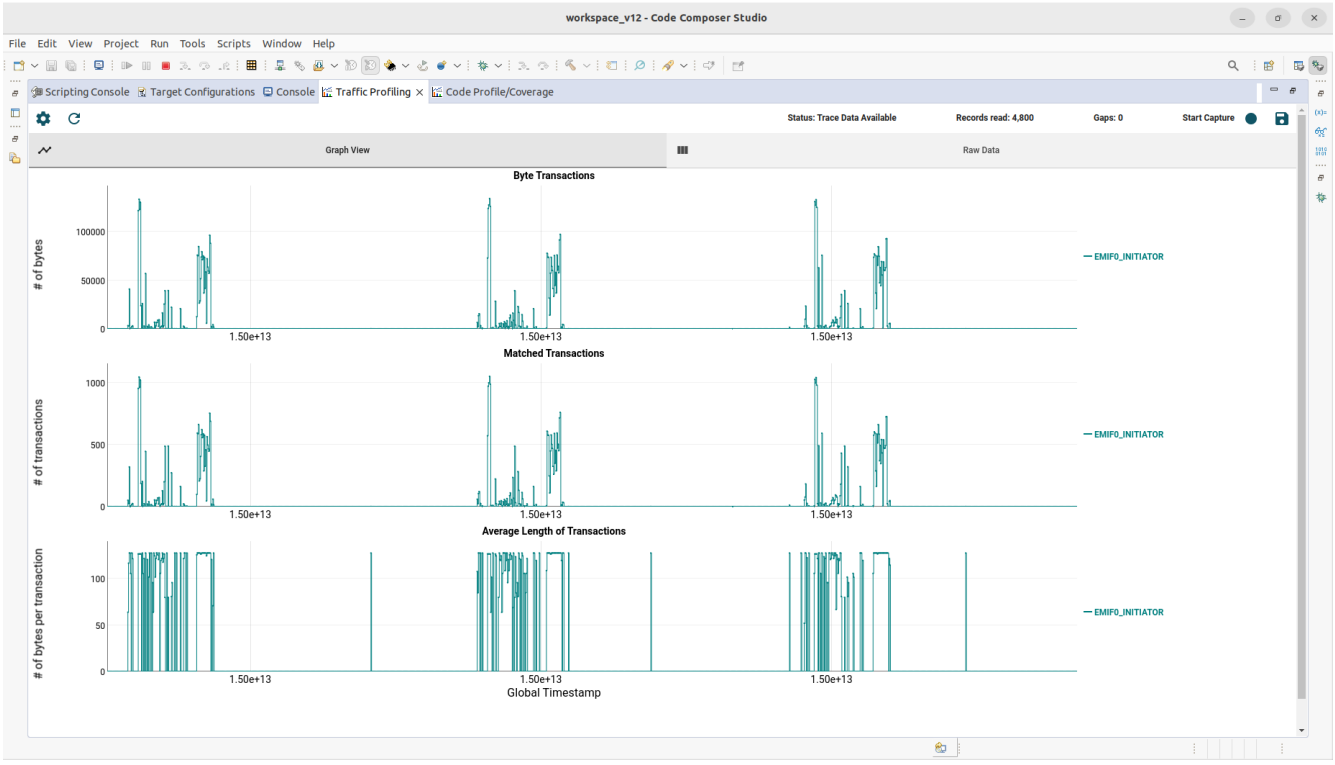

図 3-13. C7x_4 スループット

上記のプロットを見ると、C7x_4 が DSS のストールを引き起こしているように見えます。高スループットの区間は、DSS のスループット パターンと逆の関係になっているように見えます。

DSS が C7x_4 から送信される高スループットのトランザクションによってストールしている可能性があるため、C7x_4 をさらに詳しく調べます。

3.3.1.9 C7x のスループットと DSS のレイテンシのプロファイリング

残念ながら、CCS の CPTracer では、トランザクション スループットとトランザクション レイテンシを同時に可視化することはできません。一方で、Lauterbach の CPTracer ではそれが可能です。

EMIF0 でスループットをプロファイリングし、EMIF1 でレイテンシをプロファイリングすることで、同一または異なるルートのスループットとレイテンシの両方を同時に確認できます。

この方法を用いることで、DSS のレイテンシが高く、DSS のスループットがボトルネックになっている状況で、どの C7x_4 の特定ルートに非ゼロのスループットがあるかを確認できます。

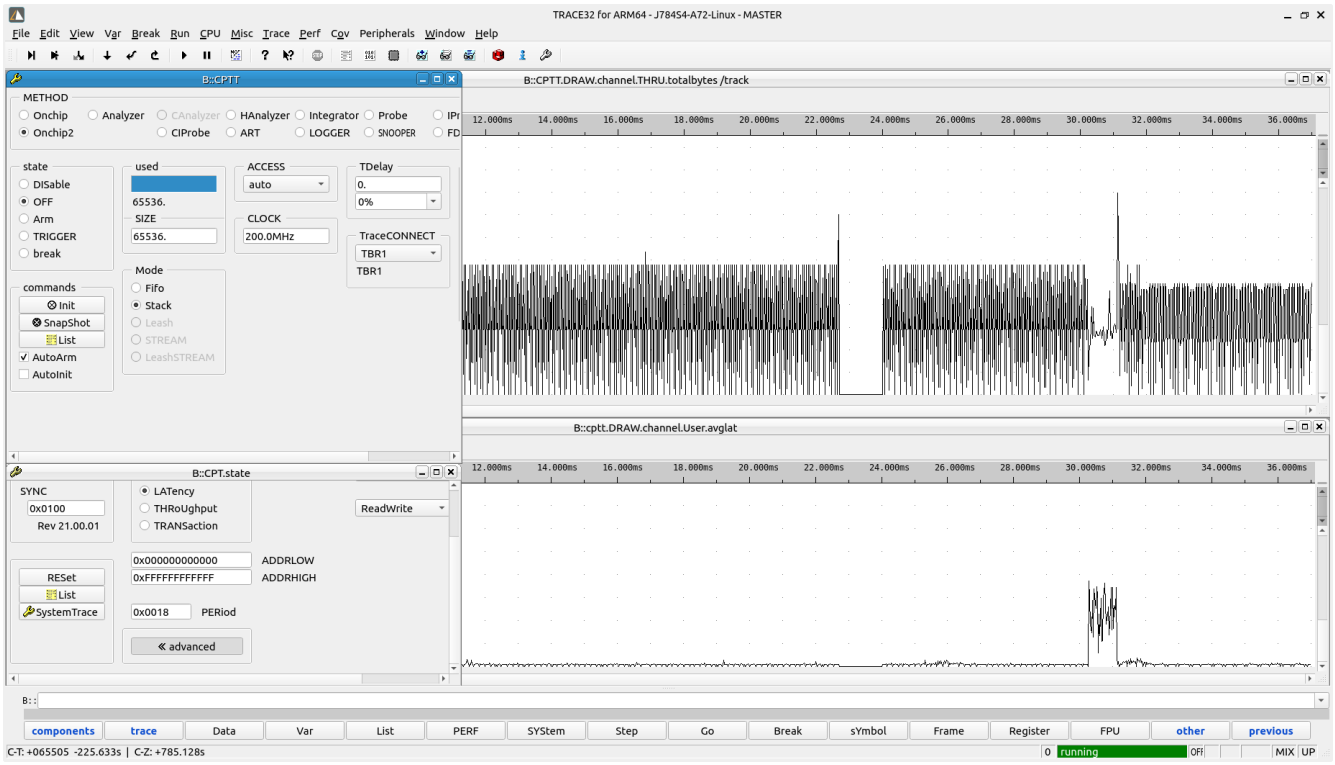

図 3-14. DSS スループットと DSS レイテンシとの関係

C7x_4 コア (ルート 0x2C) のトランザクションのレイテンシと DSS のトランザクションのレイテンシとの関係。

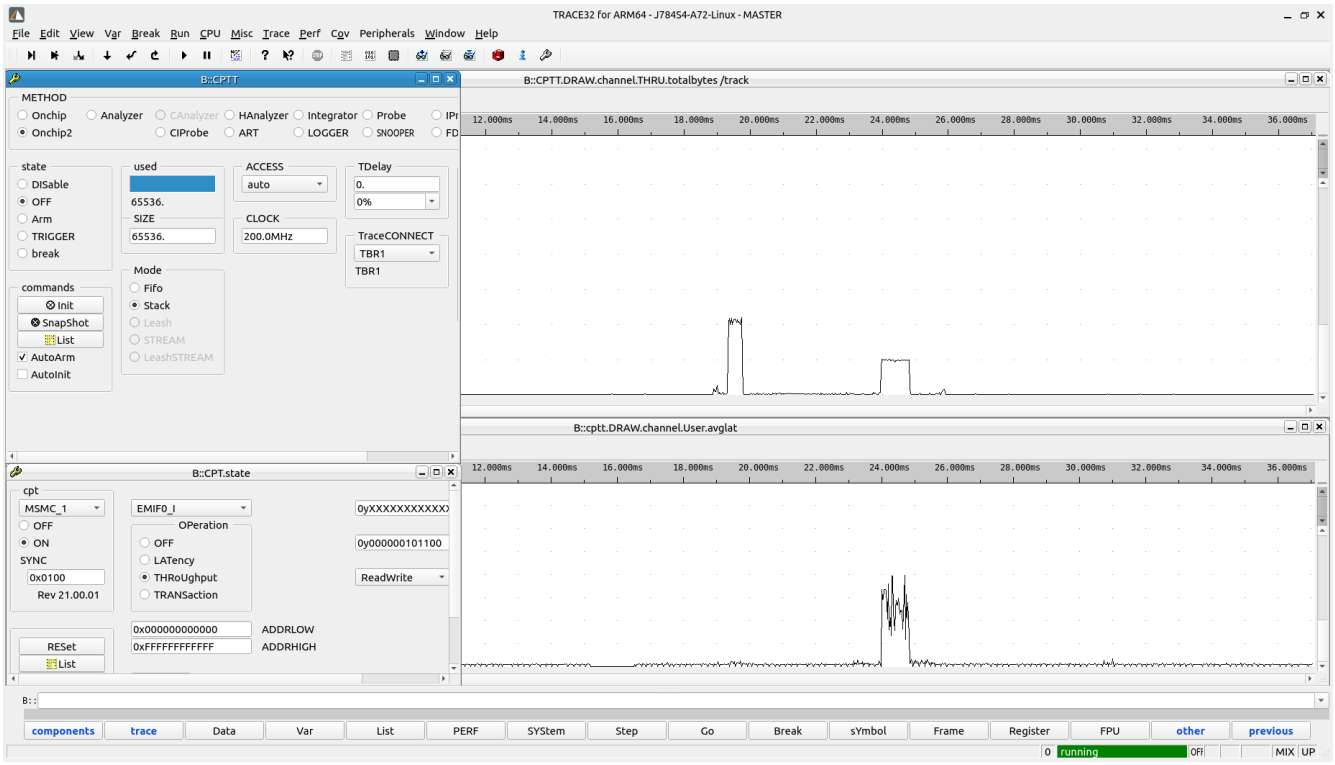

図 3-15. C7x_4 コア スループットと DSS レイテンシとの関係

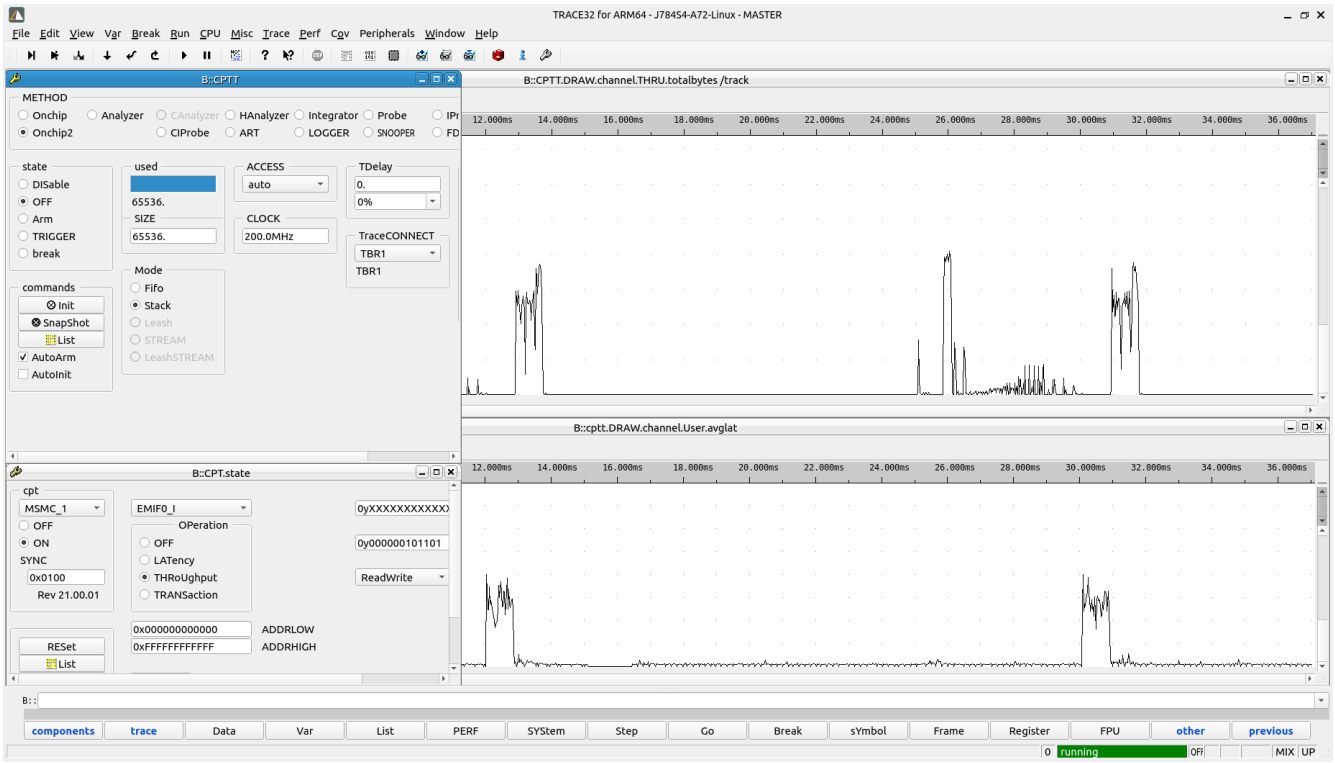

図 3-16. C7x_4 DRU0 のスループットと DSS レイテンシとの関係

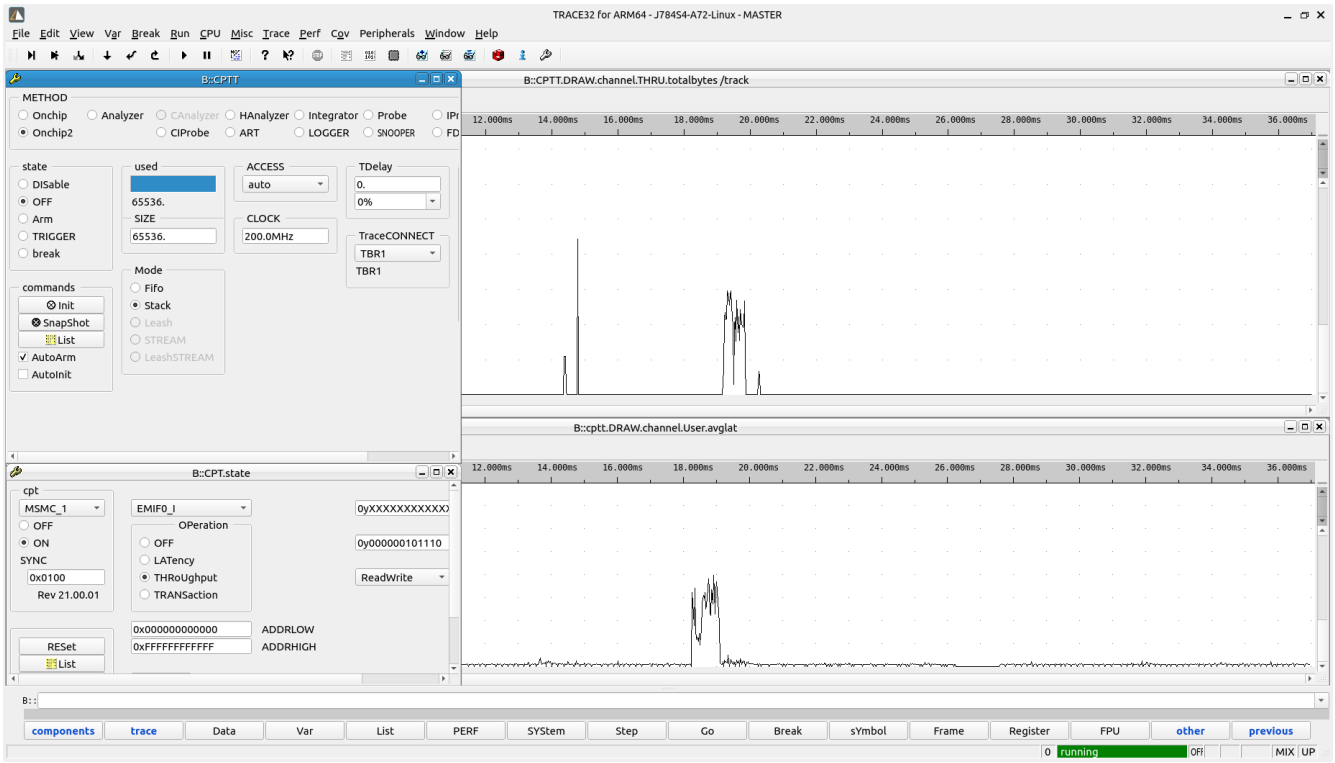

図 3-17. C7x_4 DRU1 コアのスループットと DSS レイテンシとの関係

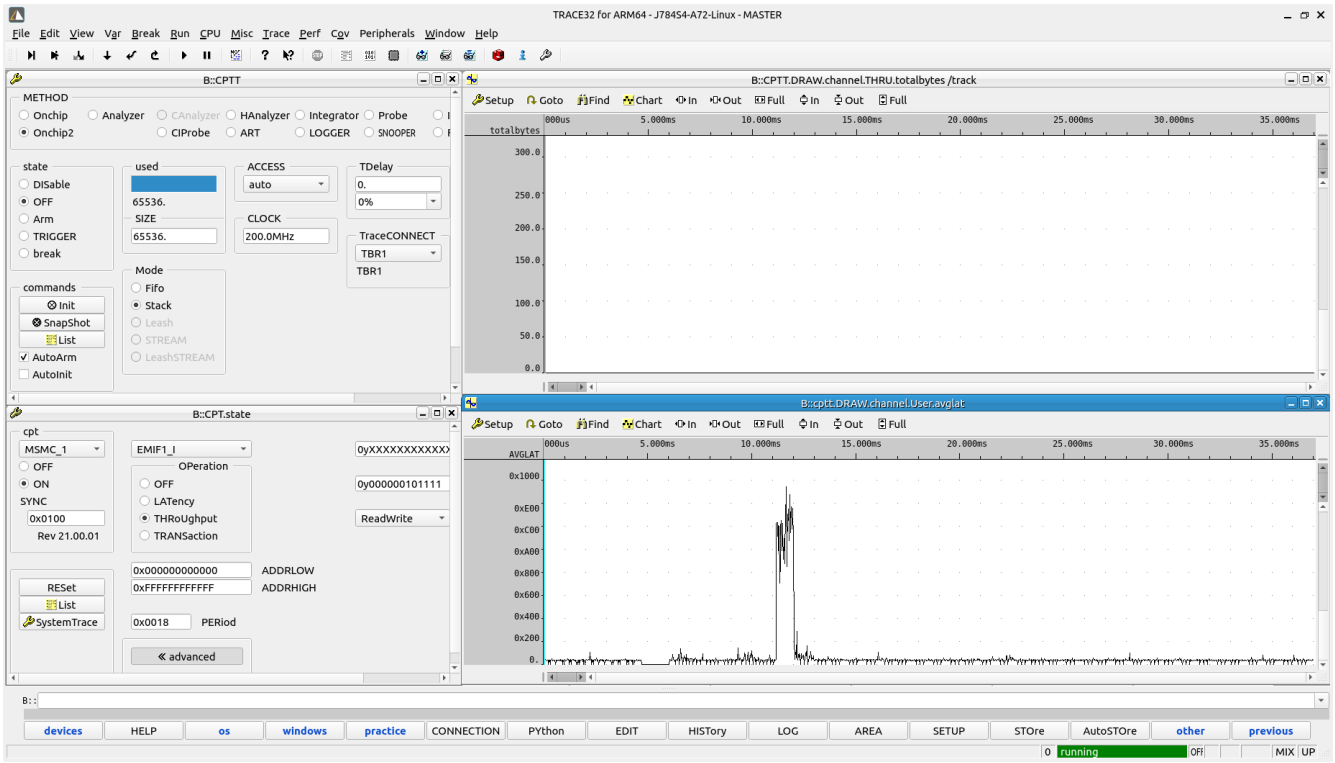

図 3-18. C7x_4 CMMU スループットと DSS レイテンシとの関係

前のプロットを見ると、問題は C7x_4 コア (0x2D) のトランザクションにあるようです。

3.3.1.10 C7x_4 コア トランザクションのプロファイリング

CPTracer のトランザクション プロファイリング機能を使用すると、C7x_4 コアおよび DSS トランザクションの QoS 設定を確認できます。

C7x_4 コア トランザクション:

- **ルート ID**: 0x002D
- **優先度**: 0x03
- **QoS**: 0x00
- **Order ID**: 0x00

DSS トランザクション:

- **ルート ID**: 0xA20
- **優先度**: 0x00 または 0x01
- **QoS**: 0x00
- **Order ID**: 0x0F

優先度を確認すると、C7x_4 コアのトランザクションが DSS トランザクションを上回ることはないはずです。

3.3.2 QoS 設定の編集

すべての QoS 設定は正しく設定されているように見えますが、念のため、これらの設定をどのように設定し、変更するかについて確認しておきましょう。さらに、ユーザは、DSS のトランザクションが NRT スレッドではなく RT スレッドへルーティングされていることを確認する必要があります。

関連する QoS 設定を行った後、ユーザーは、CPTracer のトランザクション プロファイリング機能を使用して変更内容を確認できます。

注

QoS 設定を編集するコードは、SYSCONFIG に付属の **Keystone3 リソース パーティショニング ツール** を使用して生成できます。

3.3.2.1 Order ID の編集

特定のトランザクションの order ID を変更することで、それらを NRT または RT スレッドにルーティングできます。これは、CBASS の構成レジスタおよび各独立 IP 内 (IP によって異なる) で設定されます。

3.3.2.1.1 DSS Order ID

DSS のオーダー ID は、CBASS 構成レジスタ内で設定されます。DSS 内の各チャネルの order ID を制御するためのレジスタと、それらの order ID を VBUSM の order ID にマッピングするためのレジスタがあります。DSS には、プライマリ ポート (「DMA」で示される) とセカンダリ ポート (「フレーム バッファ伸長 (FBDC)」で示される) の両方が含まれています。特定のチャネルで圧縮が有効になっている場合、そのチャネルのすべてのトランザクションはセカンダリ ポートを経由します。

表 3-7. DSS Order ID レジスタ

登録	アドレス	ビット	フィールド	説明
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_dma_slv_linkgrp_1_grp_map1	0x45DC2000	該当なし	該当なし	グループ マップ レジスタは、グループ slv_linkgrp_1 に対するイニシエータ lk3_dss_main_0.dss_inst0_vbusm_dma の最終 order ID を定義します。
		31:28	ORDERID7	7 の order ID 信号
		27:24	ORDERID6	6 の order ID 信号
		23:20	ORDERID5	5 の order ID 信号
		19:16	ORDERID4	4 の order ID 信号
		15:12	ORDERID3	3 の order ID 信号
		11:8	ORDERID2	2 の order ID 信号
		7:4	ORDERID1	1 の order ID 信号
		3:0	ORDERID0	0 の order ID 信号
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_dma_slv_linkgrp_1_grp_map2	0x45DC2004	該当なし	該当なし	グループ マップ レジスタは、グループ slv_linkgrp_1 に対するイニシエータ lk3_dss_main_0.dss_inst0_vbusm_dma の最終 order ID を定義します。
		31:28	ORDERID15	15 の order ID 信号
		27:24	ORDERID14	14 の order ID 信号
		23:20	ORDERID13	13 の order ID 信号
		19:16	ORDERID12	12 の order ID 信号
		15:12	ORDERID11	11 の order ID 信号
		11:8	ORDERID10	10 の order ID 信号
		7:4	ORDERID9	9 の order ID 信号
		3:0	ORDERID8	8 の order ID 信号

表 3-7. DSS Order ID レジスタ (続き)

登録	アドレス	ビット	フィールド	説明
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_dma_mapx	0x45DC2100 + (x * 4); x = 0 ～ 9	7:4	ORDERID	チャネル x の Order ID 信号。負荷分散のためのルートを選択します (0 ～ 7 は 1 つのルート、8 ～ 15 は別のルートを使用)。スループットを最大化するために、DDR4/LPDDR4 のリオーダリングでも使用されます。トランザクションの順序は、同じ order ID の場合にのみ保証されます
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_fbdc_slv_linkgrp_1_grp_map1	0x45DC2404	該当なし	該当なし	グループ マップ レジスタは、グループ slv_linkgrp_1 に対するイニシエータ lk3_dss_main_0.dss_inst0_vbusm_fbdc の最終 order ID を定義します。
		31:28	ORDERID7	7 の order ID 信号
		27:24	ORDERID6	6 の order ID 信号
		23:20	ORDERID5	5 の order ID 信号
		19:16	ORDERID4	4 の order ID 信号
		15:12	ORDERID3	3 の order ID 信号
		11:8	ORDERID2	2 の order ID 信号
		7:4	ORDERID1	1 の order ID 信号
		3:0	ORDERID0	0 の order ID 信号
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_fbdc_slv_linkgrp_1_grp_map2	0x45DC2408	該当なし	該当なし	グループ マップ レジスタは、グループ slv_linkgrp_1 に対するイニシエータ lk3_dss_main_0.dss_inst0_vbusm_fbdc の最終 order ID を定義します。
		31:28	ORDERID15	15 の order ID 信号
		27:24	ORDERID14	14 の order ID 信号
		23:20	ORDERID13	13 の order ID 信号
		19:16	ORDERID12	12 の order ID 信号
		15:12	ORDERID11	11 の order ID 信号
		11:8	ORDERID10	10 の order ID 信号
		7:4	ORDERID9	9 の order ID 信号
		3:0	ORDERID8	8 の order ID 信号

表 3-7. DSS Order ID レジスタ (続き)

登録	アドレス	ビット	フィールド	説明
CBASS_AC_NONSAFE_QOS_lk3_dss_main_0_dss_inst0_vbusm_fbdc_mapx	0x45DC2500 + (x * 4); x = 0 ～ 9	7:4	ORDERID	チャネル x の Order ID 信号。負荷分散のためのルートを選択します (0 ～ 7 は 1 つのルート、8 ～ 15 は別のルートを使用)。スループットを最大化するために、DDR4/LPDDR4 のリオーダリングでも使用されます。トランザクションの順序は、同じ order ID の場合にのみ保証されます

DSS チャネルの order ID を設定するコードは、ti-u-boot リポジトリの arch/arm/mach-k3/r5/j784s4/j784s4_qos_uboot.c にあります。このファイルは一般的な QoS 設定を行うもので、SYSCONFIG を使用して生成できます。

この抜粋にはビデオ プレーン 1 (VID1) の設定が含まれていますが、このファイルにはビデオ プレーン 2 (VID2)、ビデオ ライト プレーン 1 (VIDL1)、および ビデオ ライト プレーン 2 (VIDL2) の設定も含まれています。各プレーンは 2 つのチャネルに関連付けられています。

```
...

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,
```

```
        },
...
```

K3_QOS_REG、K3_QOS_VAL、K3_QOS_GROUP_REG および K3_QOS_GROUP_DEFAULT_VAL_LOW/HIGH は、arch/arm/mach-k3/include/mach/k3-qos.h で定義されています:

```
...
/* 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 Order ID

各 C7x の order ID は、DRU キュー設定レジスタを使用して構成できます。

表 3-8. C7x Order ID レジスタ

C7x サブシステム	アドレス	登録	ビット	フィールド	説明
COMPUTE_CLUSTER0_C71SS0	0x68A08000 + (j * 8); ここで、j = 0 ～ 6 です	DRU_QUEUE_cfg_j	7:4	ORDERID	この設定により、QUEUE の order ID が設定されます。
COMPUTE_CLUSTER0_C71SS1	0x69A08000 + (j * 8); ここで、j = 0 ～ 6 です	DRU_QUEUE_cfg_j	7:4	ORDERID	この設定により、QUEUE の order ID が設定されます。
COMPUTE_CLUSTER0_C71SS2	0x6AA08000 + (j * 8); ここで、j = 0 ～ 6 です	DRU_QUEUE_cfg_j	7:4	ORDERID	この設定により、QUEUE の order ID が設定されます。
COMPUTE_CLUSTER0_C71SS3	0x6BA08000 + (j * 8); ここで、j = 0 ～ 6 です	DRU_QUEUE_cfg_j	7:4	ORDERID	この設定により、QUEUE の order ID が設定されます。

DRU キュー構成レジスタと order ID を設定するコードは、vision_apps リポジトリ (platform/j784s4/rtrost/c7x_4/main.c) の該当する main.c にあります。

```
...
/* 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 および RT ルーティング

C7x_4 コア (0) と DSS (15) のトランザクションの order ID を考慮すると、C7x_4 コアのトランザクションを NRT のままにしつつ、DSS のトランザクションを RT に設定することが可能です。order ID 0 ～ 4 は NRT スレッドにルーティングし、order ID 10 ～ 15 は RT スレッドにルーティングする必要があります。

DSS のトランザクションが RT スレッドにルーティングされているかどうかは簡単に確認できます。devmem2 を使用して NAVSS_NORTH_x_NBSS_NBx_MMRS_threadmap レジスタを読み取ることで確認できます。

表 3-9. North Bridge スレッド マッピング レジスタ

登録	アドレス	値
NAVSS_NORTH_0_NBSS_NB0_MMRS_threadmap	0x03702010	0x00000002
NAVSS_NORTH_1_NBSS_NB1_MMRS_threadmap	0x03703010	0x00000004

これらの値は、order ID 0 ～ 7 の SRAM および DDR トランザクションが NRT スレッドにマッピングされ、order ID 8 ～ 15 の SRAM トランザクションと order ID 10 ～ 15 の DDR トランザクションが RT スレッドにマッピングされることを意味します。つまり、C7x_4 (order ID 0) のトランザクションは NRT スレッドにマッピングされ、DSS (order ID 15) のトランザクションは RT スレッドにマッピングされます。したがって、DSS トランザクションは常に C7x_4 トランザクションよりも優先度が高い必要があります。

3.3.2.2.1 U-Boot での NRT および RT ルーティング

J784S4 では、RT と NRT のマッピングが ti-u-boot–2025.01 分岐 / リリースに追加されました。以前のリリースでは、そのコードは SDK に含まれる未コミットの編集としてパッケージ化され、追加されていました。関連するコードは、arch/arm/mach-k3/j784s4/j784s4_init.c にあります。

以下のコードは、order ID 8 ～ 15 の SRAM トランザクションと、order ID 10 ～ 15 の DDR トランザクションを RT スレッドにマッピングします。

```
...

/* 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 優先度の編集

個別の IP の優先度は、それぞれのドライバ内で設定されます。

3.3.2.3.1 DSS 優先度

DSS では、トランザクションに対して 2 種類の優先度があり、MFLAG トランザクションには高優先度、MFLAG 以外のトランザクションには低優先度が設定されます。MFLAG メカニズムにより、DMA 読み取りバッファがアンダーフローに近づいたときに、DSS は自身のトラフィックの優先度を上げることができます。

表 3-10. DSS 優先度レジスタ

登録	アドレス	ビット	フィールド	説明
DSS_DISPC_0_COMMON_M_DSS_CBA_CFG	0x04A000A4	5:3	PRI_HI	DSS から CBA へ PRI_HI バスで送信される値は、高優先度 [MFLAG] トランザクションの優先度レベルを示します。値 0x0 は最高優先度を示し、値 0x7 は最低優先度を示します
DSS_DISPC_0_COMMON_M_DSS_CBA_CFG	0x04A000A4	2:0	PRI_LO	DSS から CBA へ PRI_LO バスで送信される値は、通常 [non-MFLAG] トランザクションの優先度レベルを示します。値 0x0 は最高優先度を示し、値 0x7 は最低優先度を示します

DSS の優先度を設定するコードは、Linux ドライバ (drivers/gpu/drm/tidss/tidss_dpc.c) にあります。このコードは、MFLAG トランザクションの優先度レベルを 0 に、MFLAG 以外のトランザクションを 1 に設定します。

```
...

        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 優先度

order ID と同様に、各 C7x の優先度は DRU キュー構成レジスタを使用して設定できます。

表 3-11. C7x 優先度レジスタ

C7x	登録	登録	ビット	フィールド	説明
COMPUTE_CLUSTER0_C71SS0	0x68A08000 + (j * 8); ここで、j = 0 ～ 6 です	DRU_QUEUE_cfg_j	2:0	PRI	これにより、QUEUE0 の優先度が設定されます。これは、このキューからのすべてのコマンドに対して外部バスに表示される優先度になります。
COMPUTE_CLUSTER0_C71SS1	0x69A08000 + (j * 8); ここで、j = 0 ～ 6 です	DRU_QUEUE_cfg_j	2:0	PRI	これにより、QUEUE0 の優先度が設定されます。これは、このキューからのすべてのコマンドに対して外部バスに表示される優先度になります。
COMPUTE_CLUSTER0_C71SS2	0x6AA08000 + (j * 8); ここで、j = 0 ～ 6 です	DRU_QUEUE_cfg_j	2:0	PRI	これにより、QUEUE0 の優先度が設定されます。これは、このキューからのすべてのコマンドに対して外部バスに表示される優先度になります。
COMPUTE_CLUSTER0_C71SS3	0x6BA08000 + (j * 8); ここで、j = 0 ～ 6 です	DRU_QUEUE_cfg_j	2:0	PRI	これにより、QUEUE0 の優先度が設定されます。これは、このキューからのすべてのコマンドに対して外部バスに表示される優先度になります。

注文 ID と同様に、優先度レベルを設定するコードは、vision_apps リポジトリ (platform/j784s4/rtrost/c7x_4/main.c) の該当する main.c にあります。

```
...

/* 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 CoS マッピングの編集

すべての QoS 設定が正しく、C7x_4 コアのトランザクションではなく DSS トランザクションが優先されるようになっているため、問題は DDR コントローラの優先度マッピングにあるはずです。

3.3.3.1 CoS マッピング レジスタ

DDRSS には、VBUSM.C の優先度を AXI の優先度にマッピングするための一連のマルチプレクサが含まれています。マッピングを制御するレジスタは次のとおりです:

- **ルート ID フィルタ**:
  - emif_ew_sscfg_V2A_R1_MAT_REG: ルート ID をフィルタリングし、範囲 1 マッピングへルーティングできるようにします
  - emif_ew_sscfg_V2A_R2_MAT_REG: ルート ID をフィルタリングし、範囲 2 マッピングへルーティングできるようにします
  - emif_ew_sscfg_V2A_R3_MAT_REG: ルート ID をフィルタリングし、範囲 3 マッピングへルーティングできるようにします
- **優先度マッピング**:
  - **LPT (低優先度スレッド)**:
    - emif_ew_sscfg_V2A_LPT_DEF_PRI_MAP_REG: デフォルトの VBUSM.C から AXI への優先度のマッピング
    - emif_ew_sscfg_V2A_LPT_R1_PRI_MAP_REG: 範囲 1 VBUSM.C から AXI への優先度マッピング
    - emif_ew_sscfg_V2A_LPT_R2_PRI_MAP_REG: 範囲 2 VBUSM.C から AXI への優先度マッピング
    - emif_ew_sscfg_V2A_LPT_R3_PRI_MAP_REG: 範囲 3 VBUSM.C から AXI への優先度マッピング
  - **HPT (高優先度スレッド)**:
    - emif_ew_sscfg_V2A_HPT_DEF_PRI_MAP_REG: デフォルトの VBUSM.C から AXI への優先度のマッピング
    - emif_ew_sscfg_V2A_HPT_R1_PRI_MAP_REG: 範囲 1 VBUSM.C から AXI への優先度マッピング
    - emif_ew_sscfg_V2A_HPT_R2_PRI_MAP_REG: 範囲 2 VBUSM.C から AXI への優先度マッピング
    - emif_ew_sscfg_V2A_HPT_R3_PRI_MAP_REG: 範囲 3 VBUSM.C から AXI への優先度マッピング

注

正確なレジスタのアドレスとフィールドは、TDA4VH TRM に記載されています

仮定の **LPT** トランザクションを取ります。ルート ID がフィルタ (**範囲 1** など) 内にある場合は、**emif_ew_sscfg_V2A_LPT_R1_PRI_MAP_REG** マッピングを使用して優先順位をマッピングします。フィルタに含まれない場合、**emif_ew_sscfg_V2A_LPT_DEF_PRI_MAP_REG** レジスタを使用して優先度がマッピングされます。このマルチプレクシングを、図 2-1 に示します。

3.3.3.2 CoS マッピングの確認

DSS と C7x_4 コアのトランザクションに関する QoS 設定を、あらためて確認します:

- **DSS**:
  - ルート ID: 0xA20
  - Order ID: 0x0F
  - NRT または RT: RT
  - 優先度: 0x00 または 0x01
- **C7x_4 コア**:
  - ルート ID: 0x02D
  - Order ID: 0x00
  - NRT または RT: NRT

– 優先度: 0x03

次の表には、CoS レジスタの値を示します (レジスタ グループという呼び方は、レジスタを分類するために私が付けた用語です):

表 3-12. CoS レジスタ

レジスタ グループ	登録	値
ルート ID フィルタ	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 優先度マッピング	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
HPT 優先度マッピング	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

ルート ID フィルタはいずれもイネーブルになっていません。これは、すべての優先度がデフォルトの優先度マッピングを使用してマッピングされることを意味します。

LPT トランザクションと HPT トランザクションの両方のデフォルト優先度マッピングは、0 に等しく設定されています。これは、DDRSS 内ですべてのトランザクションの優先度が同じであることを意味します。したがって、DDR 内では DSS と C7x_4 コアのトランザクションは同じ優先度になります。これにより、C7x_4 コアのトランザクションが DSS のトランザクションを停滞させている理由が説明できます。

3.4 DSS の同期ロスの修正

問題の根本原因が特定されたので、次にどのように修正するかを決める必要があります。

DDR コントローラ内ですべての優先度を均等化することには利点があります。つまり、より高い優先度のスレッドがキューに入っても既存のスレッドは追い出されないため、ページ スラッシングの発生を防ぐことができます。しかしこの場合、優先度を尊重しないことは、利益よりも大きな不利益をもたらします。

同期喪失の問題を解決するには、いくつかのオプションがあります:

1. C7x_4 コア トランザクションの再マッピング
2. すべてのトランザクションで優先順位を維持します

C7x_4 コアのトランザクションのみを再マッピングすることで、同期の損失の問題を解決でき、他のすべてのトランザクションは引き続き DSS のトランザクションより優先されます。すべてのトランザクションで優先度を尊重すると、DDRSS は各トランザクションをそれぞれの VBUSM.C 優先度に従って処理することになります。

AXI の優先度設定は 8 種類しかないため、すべてのトランザクションで優先度を尊重しようとすると、HPT と LPT の間で一部重複が発生します。

表 3-13. HPT および LPT から AXI への優先順位マッピング

リアルタイムまたはリアルタイム以外	VBUSM.C 優先度	AXI 優先度
リアルタイム	0	0
リアルタイム	1	0
リアルタイム	2	1
リアルタイム	3	1
リアルタイム	4	2
リアルタイム以外	0	2
リアルタイム	5	3
リアルタイム以外	1	3
リアルタイム	6	4
リアルタイム以外	2	4
リアルタイム	7	5
リアルタイム以外	3	5
リアルタイム以外	4	6
リアルタイム以外	5	6
リアルタイム以外	6	7
リアルタイム以外	7	7

CoS マッピングは、ボードの初期化中に U-Boot に追加されます。ti-u-boot-2023.04 と ti-u-boot-2025.01 ブランチの両方向けに、パッチが書き込まれています。

3.4.1 C7x_4 コア トランザクションの再マッピング

以下の U-Boot パッチでは、C7x_4 コアのトランザクションを LPT 範囲 1 の優先度マッピングに割り当てます。これにより、他のトランザクションの優先度を変更することを回避できます。

注

すべての C7x トランザクションを LPT 範囲 1 の優先度マッピングに設定する場合は、0xf02d0000 を 0x80208028 に置き換えます

3.4.1.1 ti-u-boot-2023.04

次のパッチは、コミットの最上位に適用されます: 2bedcd265ca6 ("configs: am57xx_hs_evm_defconfig: Early Boot に必要な設定を有効化する)

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

以下のパッチは、コミットの先頭に適用されます: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 すべての優先度を尊重

以下の U-Boot パッチは、デフォルトの LPT および HPT の優先度マッピングを、範囲 1 ～ 3 のマッピングと同じ設定に変更します。これは TDA4VM/J721E のデフォルト動作です。

3.4.2.1 ti-u-boot-2023.04

次のパッチは、コミットの最上位に適用されます: 2bedcd265ca6 ("configs: am57xx_hs_evm_defconfig: Early Boot に必要な設定を有効化する)

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

以下のパッチは、コミットの先頭に適用されます: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_0 */
 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 まとめ

このアプリケーション ノートでは、Jacinto デバイス (特に TDA4VH) における各種 QoS メカニズムの概要と、それらを用いて異なる IP 間で負荷分散を行う方法について説明します。Order ID は、トランザクションが通る経路と、リアルタイムおよびリアルタイム以外の割り当てに影響します。トランザクションに異なる優先度やリアルタイム / リアルタイム以外の割り当てを設定することで、処理される順序に影響を与え、すべてのアプリケーションとそれぞれのハードウェアの性能をバランスよく最適化することができます。

ケース スタディでは、C7x_4 コアのトランザクションが DSS のトランザクションを停止させ、ディスプレイ サブシステム内で同期喪失エラーを引き起こしていることが確認されました。VBUSM および VBUSM.C の領域では、DSS のトランザクションは C7x_4 コアのトランザクションより優先されていましたが、DDRSS 内ではすべての優先度が同一に扱われていました。この均一化を取り除き、DDR コントローラ内で VBUSM.C の優先度を尊重するようにしたことで、同期喪失エラーは解消されました。

5 参考資料

1. テキサス インスツルメンツ、J784S4 J742S2 テクニカル リファレンス マニュアル、テクニカル リファレンス マニュアル
2. テキサス インスツルメンツ、J721E DRA829/TDA4VM プロセッサ、シリコン リビジョン 2.0、1.1 テクニカル リファレンス マニュアル、テクニカル リファレンス マニュアル。
3. テキサス インスツルメンツ、PROCESSOR-SDK-LINUX-J784S4、ソフトウェア開発キット。
4. テキサス インスツルメンツ、PROCESSOR-SDK-RTOS-J784S4、ソフトウェア開発キット。
5. テキサス インスツルメンツ、CP Tracers V2 を使用したトラフィック プロファイリング、TI ソフトウェア ダウンロード。
6. テキサス インスツルメンツ、*0001-arm-mach-k3-j784s4-Remove-priority-equalization-for-.patch*、ti-u-boot-2023.04.y パッチ
7. テキサス インスツルメンツ、*0001-arm-mach-k3-j784s4-Remove-priority-equalization-for-.patch*、ti-u-boot-2023.04.y パッチ
8. テキサス インスツルメンツ、*0001-arm-mach-k3-j784s4-Remove-priority-equalization-and-.patch*、ti-u-boot-2025.01.y パッチ
9. テキサス インスツルメンツ、*0001-arm-mach-k3-j784s4-Remove-priority-equalization-and-.patch*、ti-u-boot-2025.01.y パッチ

6 改訂履歴

Changes from Revision * (March 2026) to Revision A (March 2026)	Page
• タイトルを更新	0
• 表 3-13 に表タトルを追加	45
• ドキュメント全体にわたって表、図、相互参照の採番方法を更新	53

重要なお知らせと免責事項

テキサス・インスツルメンツは、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、テキサス・インスツルメンツ製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した テキサス・インスツルメンツ製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている テキサス・インスツルメンツ製品を使用するアプリケーションの開発の目的でのみ、テキサス・インスツルメンツはその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。テキサス・インスツルメンツや第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、テキサス・インスツルメンツおよびその代理人を完全に補償するものとし、テキサス・インスツルメンツは一切の責任を拒否します。

テキサス・インスツルメンツの製品は、テキサス・インスツルメンツの販売条件、または ti.com やかかる テキサス・インスツルメンツ製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。テキサス・インスツルメンツがこれらのリソースを提供することは、適用されるテキサス・インスツルメンツの保証または他の保証の放棄の拡大や変更を意味するものではありません。

お客様がいかなる追加条項または代替条項を提案した場合でも、テキサス・インスツルメンツはそれらに異議を唱え、拒否します。

郵送先住所:Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、 TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月