System Tuning & BW monitoring/control on TDA2xx

Agenda

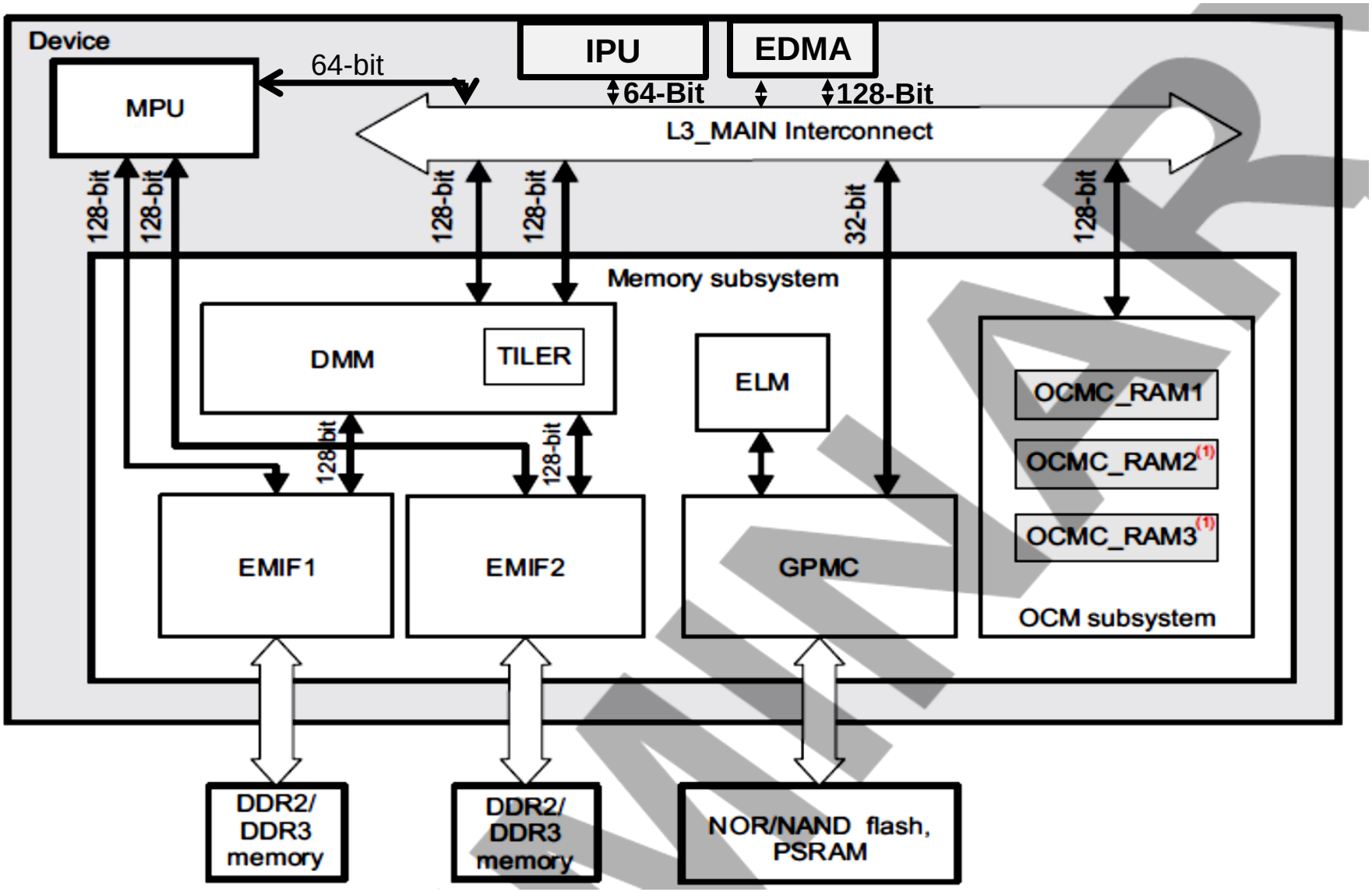
- Introduction
  - System Level Memory Architecture
- System Instrumentation and Measurement Strategies
  - GP Timers & L3 Statistic Collectors
  - CPU Timers/Counters
    - MPU/EVE/DSP
- Bandwidth Knobs at the IP level.
  - MFLAG
  - Control Module: L3 Pressure
- Bandwidth Knobs at the interconnect level.
  - Bandwidth Regulator
  - Bandwidth Limiter
- Bandwidth Knobs at the DMM/EMIF level
  - DMM Peg Priority
  - DMM Emergency
  - EMIF CoS and other QoS knobs

Refer TDA2xx TRM for exact register programming details

Overview : Basic Terminology

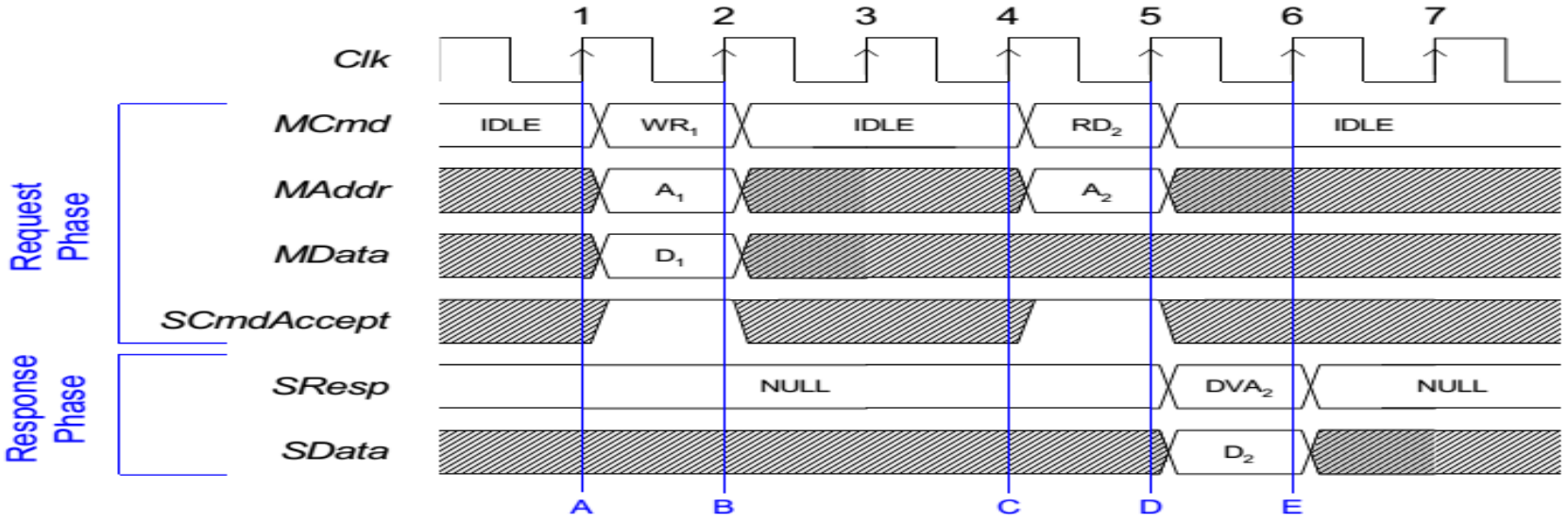
- Master IP – Initiates bus requests
- Slave IP – Responds to bus requests
- L3 Interconnect – Routes/arbitrates bus requests between Masters and Slaves
- Dynamic Memory Manager (DMM)
  - Provides interleaved view of two EMIF's in single address space
  - Provides non-interleaved view of single EMIF in single address space
- External Memory Interface (EMIF) - Queues/schedules requests to DDR
- BW (Bandwidth)/Throughput –
  - These two terms are used interchangeably to indicate the number of bytes transferred between initiators and slaves every second.
  - Represented in MB/s or GB/s.

Overview: TDA2xx Memory Subsystem



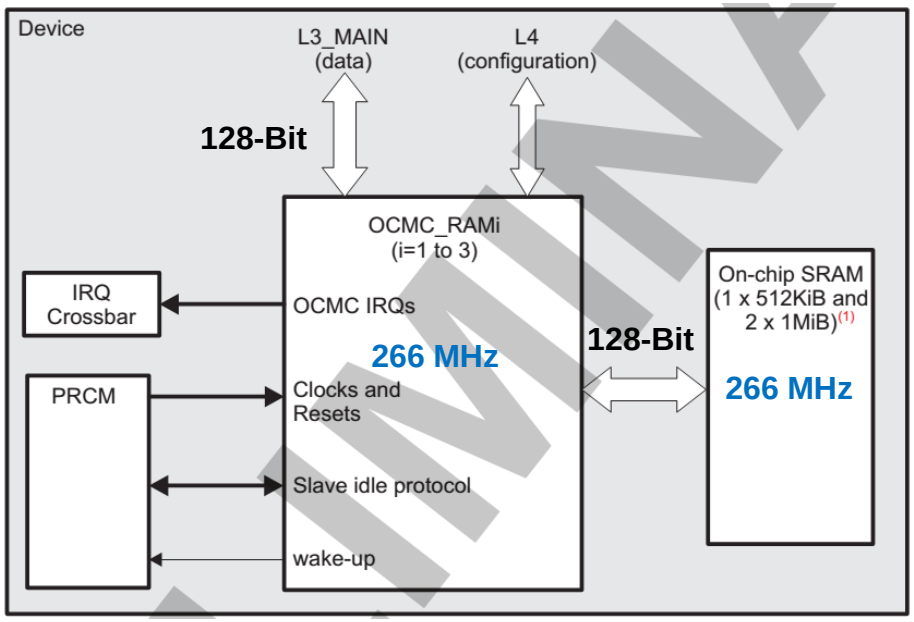
Overview : Concept of theoretical BW

- *Ideal Throughput = Frequency of Limiting Port * Data Bus Width in Bytes*
- *Utilization or Efficiency = (Measured Throughput/ Ideal Throughput) * 100*
- When the source and destination memories are the same (eg. OCMC0 to OCMC0 EDMA transfer.), then the measured bandwidth is multiplied with 2.



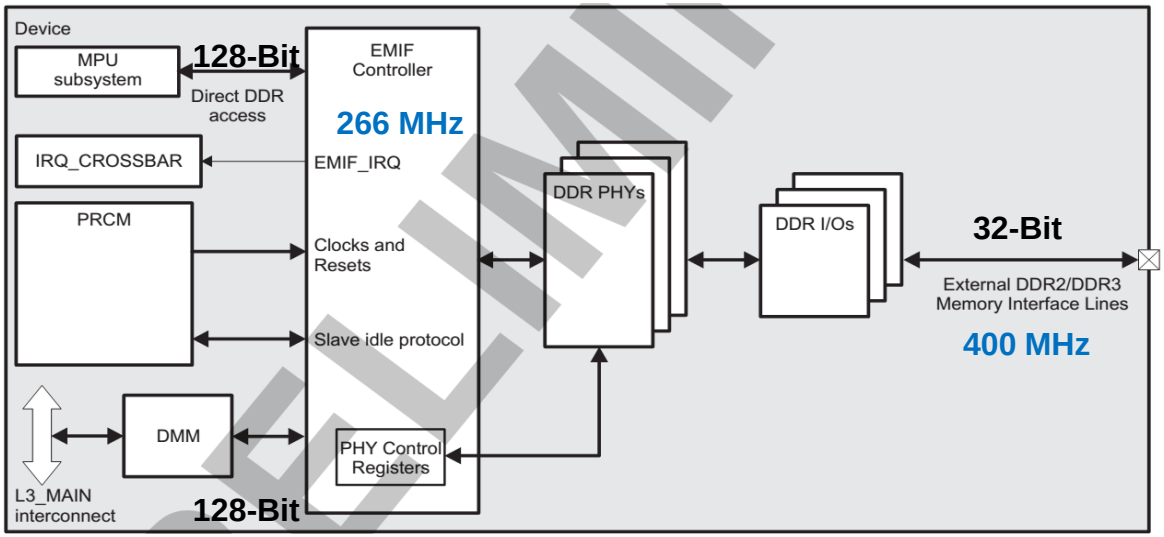
Theoretical Max Bandwidth Examples – OCMC RAM

- For different memories in the system the theoretical bandwidth can be calculated as below:
  - OCMC operating @ 266 MHz
    - OCMC bus width = 128 bits = 16 bytes
    - OCMC Frequency = 266 MHz
    - Theoretical Max bandwidth = 266 * 16 = **4256 MBps**



Theoretical Max Bandwidth Examples – DDR

- For different memories in the system the theoretical bandwidth can be calculated as below:
  - DDR3 Operating @ 400 MHz
    - DDR3 bus width = 32 bits = 4 bytes
    - DDR3 Frequency of operation = 400 MHz
    - Theoretical Max Bandwidth = 400 * 4 * 2 = **3200 MBps**
    - (The extra 2 is because data can be sent on the rising edge and falling edge of the clock)



Overview: Quality of Service Mechanisms

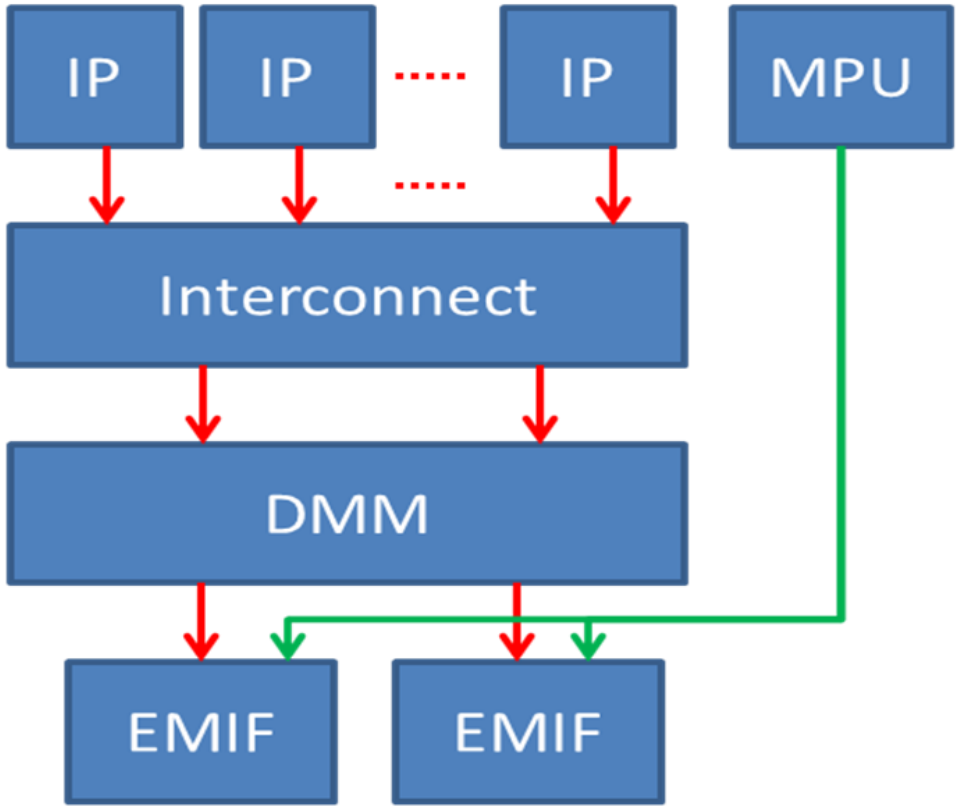
QoS mechansims

MFlag/MReqPrio Generation
- IP driven
- Control Module

Arbitration Based on
- Bandwidth Regulator (BR)
- Bandwidth Limiter (BL)
- MFlag

Arbitration Based on Emergency
PEG Priority Generation

Arbitration Based on
- CoS
- EMIF algorithm
- MreqPrio (generated from DMM PEG)



How to measure system performance?

- Different methods to measure system performance.

- Broadly define system performance along the following vectors:
  - Frames/second achieved by real time IPs, multimedia and vision IPs.
  - Average and Peak DDR bandwidth and utilization.
  - Latency of access from master to slave memories.
  - Average and peak throughput of peripheral IPs.

- Different measurement techniques available in TDA2xx:
  - **General Purpose Timers/CPU Timers**
  - **L3 Statistic collectors**
  - **EMIF counters**
  - SCTM/Processor Trace/Watch Points etc.

GP Timers/CPU Timers

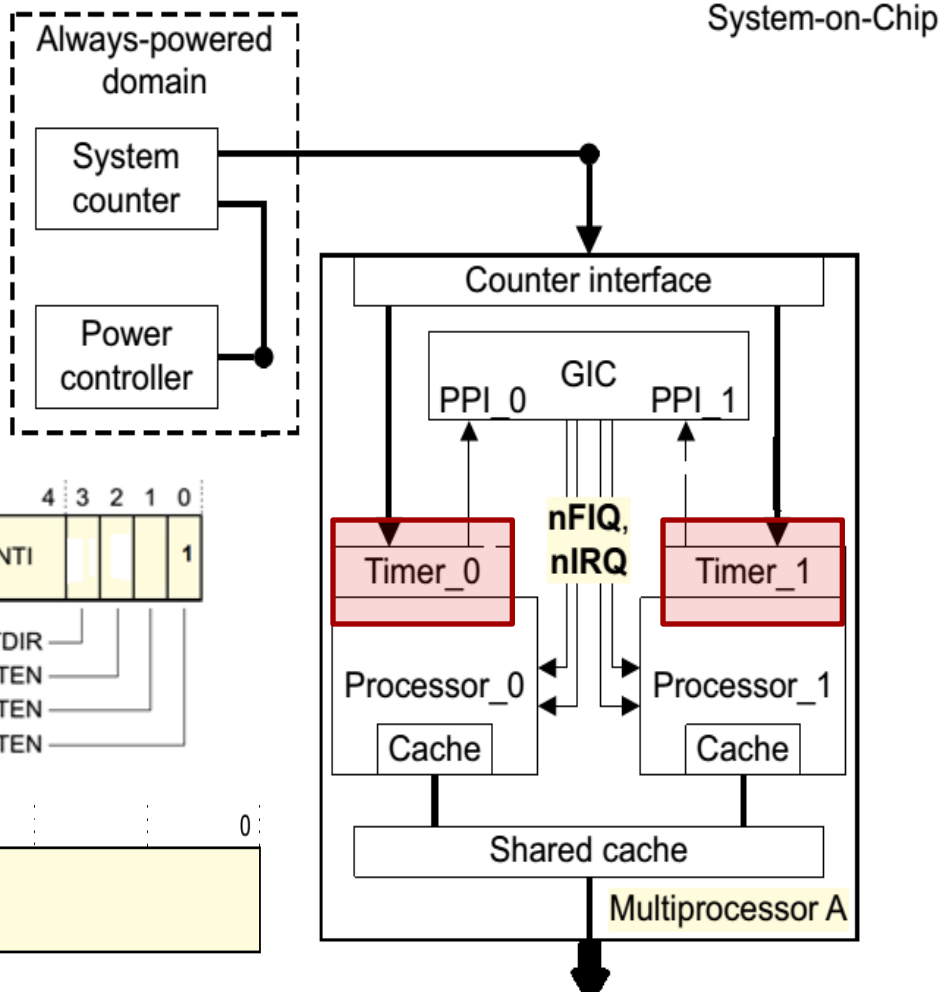
- Provides a quick way to measure throughput.
- Provides a coarse measurement of initiator bandwidth and FPS for real time traffic IPs (VIP/DSS), multimedia IPs (IVAHD/VPE) and vision IPs (EVE/DSP).
- Can’t measure latency. Can’t know instantaneous traffic peaks.
- Typical flow:

```
/*Get the Timer Start Stamp*/
timerStartStamp = timerRead(TIMER_NUM);

/*Start the transfer*/
transferStart();

/*Wait for Transfer Completion*/
waitForTransferCompletion();

/*Get the Time Stamp*/
timerEndStamp  = timerRead(TIMER_NUM);
```

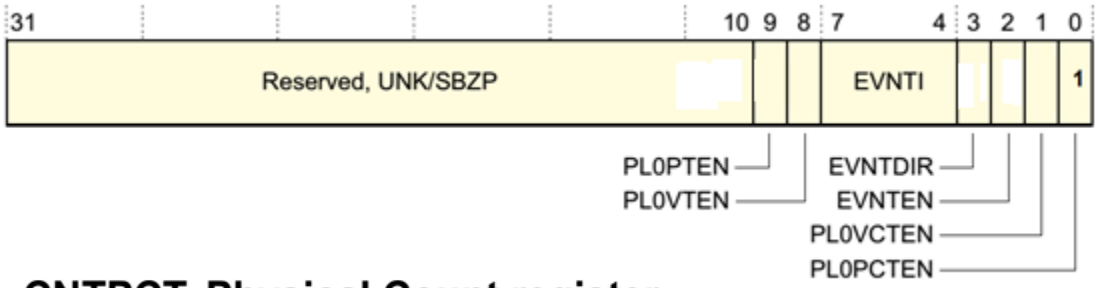
```
/* Calculate the bandwidth - Note that one needs to
 * multiply x2 from the calculated value to get
 * Bandwidth when the source and destination memory
 * are the same       */
if (timerEndStamp > timerStartStamp)
        BW = (((float) (TRANSFER_SIZE)/ (float)
        (timerEndStamp - timerStartStamp)) * TIMER_FREQ);
else
        BW = (((float) (TRANSFER_SIZE)/
                (float) (timerEndStamp - timerStartStamp +
                0xFFFFFFFF)) * TIMER_FREQ);
```

Cortex A15 CPU Timer Configuration

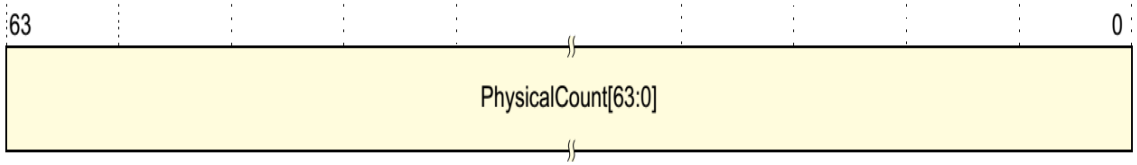
- 64 bit Generic Timer
- Counts at sys_clk Frequency (external crystal frequency) = 20 MHz.
- Setting up the timer:



CNTKCTL, Timer PL1 Control register



CNTPCT, Physical Count register,



DSP CPU Timer Configuration

- 64 bit Generic Timer
- Counts at CPU Clock Frequency.
- **Initialization:** The counter is cleared to 0 after reset, and counting is disabled.
- **Enabling Counting:** The counter is enabled by writing to TSCL. The value written is ignored.

```
MVC B0,TSCL ; Start TSC
MVC TSCL,B0 ; B0 = 0
MVC TSCL,B1 ; B1 = 1
```

- **Disabling Counting:** Once enabled, counting cannot be disabled under program control. Counting is disabled in the following cases:
  - After exiting the reset state.
  - When the CPU is fully powered down.

- **Reading the counter:**

```
BNOP TSC_Read_Done, 3
MVC TSCL,B0 ; Read the low half first
; high half copied to TSCH
MVC TSCH,B1 ; Read the snapshot of
;the high half
TSC_Read_Done:
```

EVE SCTM

- 8 32-bit counters, out of which 2 can be configured as timers.
- The SCTM module operates at half the clock rate (EVEx_GFCLK), CLK2 = 0.5 × CLK1.

IP Source	Name	Type	SCTM Mode	Native Clock Freq	SCTM Event?
ARP32_Pcache	cache_miss_count	Pulse	Duration	CLK2	1
ARP32_Pcache	cache_hit_count	pulse	Duration	CLK2	2
ARP32_Pcache	cache_miss_stall	duration	Duration or Event	CLK2	3
ARP32_Pcache	prefetch_compulsory_count	pulse	Duration	CLK2	4
ARP32_Pcache	Prefetch_lookahead_count	pulse	Duration	CLK2	5
ARP32_Pcache	prefetch_hit_count	pulse	Duration	CLK2	6
…..					

- For full list of SCTM events refer TDA2xx TRM and EVE Programming Guide.

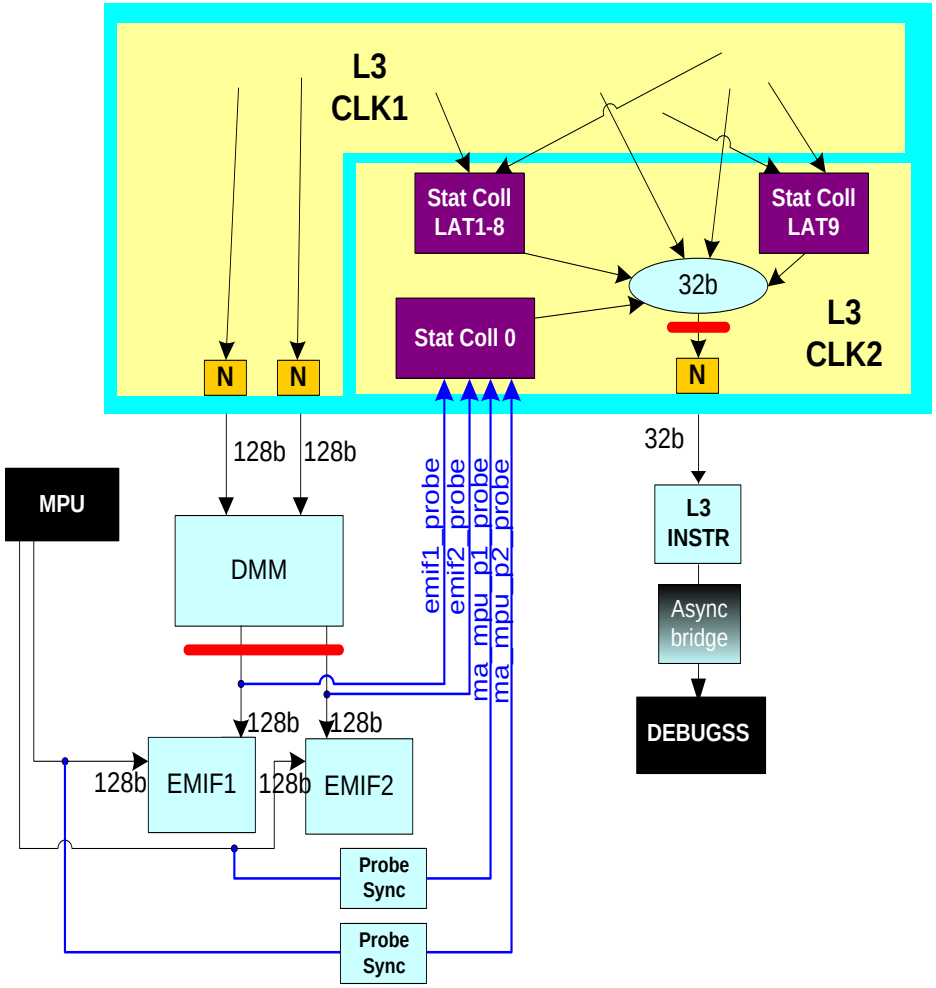
L3 StatColl Overview

- Provides ability to probe OCP (Open Core Bus Protocol) or NTTP (Arteris L3 Interconnect Packet Protocol) links.
- Transmitting results to a debug unit through a dedicated NTTP link.
- Software controlled at run time through the service network.
- Non intrusive monitoring
- Up to 8 probes for monitoring NTTP or OCP links.
- Programmable filters and counters.
- Collect results at programmable time interval
- Provides metrics such as throughput and latency on some data flows.
- Additional filtering capabilities to focus on certain initiators or targets.

L3 Statistic Collectors (StatColl)

- 10 statcoll instances available on TDA2xx to measure traffic statistics of different subsystems.
- Static Mapping from subsystems to statistic collector.

StatColl_0			
Probe #	**Description**	**Link**	**Port #**
0	EMIF1_SYS	OCP REQ	0
		OCP RSP	1
1	EMIF2_SYS	OCP REQ	2
		OCP RSP	3
2	MA_MPU_P1	OCP REQ	4
		OCP RSP	5
3	MA_MPU_P2	OCP REQ	6
		OCP RSP	7



Refer the TDA2xx TRM (Section 30.10.7.1 L3 Target Load Monitoring) to know the complete Initiator to statcoll mapping.

Statcoll Probe to Initiator/Slave Mapping

StatColl_0

Probe #	Description	Link	Port #
0	EMIF1_SYS	OCP REQ	0
		OCP RSP	1
1	EMIF2_SYS	OCP REQ	2
		OCP RSP	3
2	MA_MPU_P1	OCP REQ	4
		OCP RSP	5
3	MA_MPU_P2	OCP REQ	6
		OCP RSP	7

StatColl_1

Probe #	Description	Link	Port #
0	MPU	NTTP REQ	0
		NTTP RSP	1
1	MMU1	NTTP REQ	2
		NTTP RSP	3
2	EDMA_TC0_RD	NTTP REQ	4
		NTTP RSP	5
3	EDMA_TC0_WR	NTTP REQ	6
		NTTP RSP	7
4	EDMA_TC1_RD	NTTP REQ	8
		NTTP RSP	9
5	EDMA_TC1_WR	NTTP REQ	10
		NTTP RSP	11

StatColl_5

3	IPU1	NTTP REQ	6
		NTTP RSP	7

StatColl_7

Probe #	Description	Link	Port #
0	GMAC SW	NTTP REQ	0
		NTTP RSP	1

StatColl_9

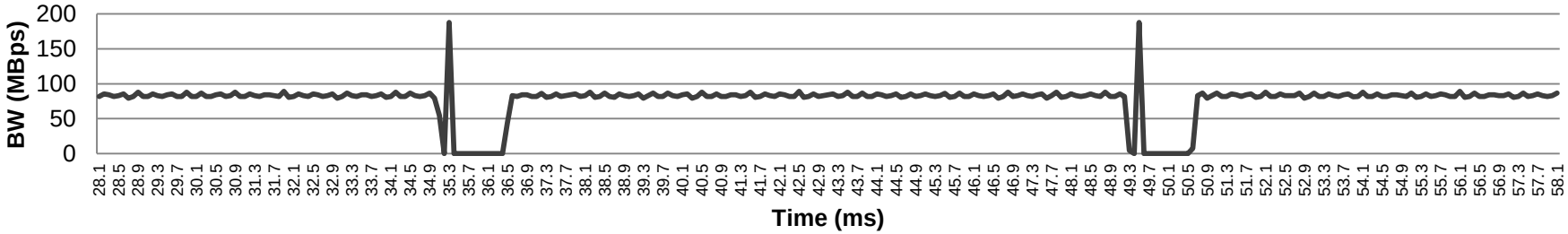
Probe #	Description	Link	Port #
0	OCMC RAM1	NTTP REQ	0
		NTTP RSP	1
1	OCMC RAM2	NTTP REQ	2
		NTTP RSP	3
2	OCMC RAM3	NTTP REQ	4
		NTTP RSP	5

L3 Statcoll - A Small Caveat

- If CPU has cache enabled, the data read from the L3 Statcoll and actual bytes transferred may differ.
- This is because the L3 statcoll reads bytes at the L3/EMIF interface and lot of data may be cached already.
- ARM performs speculative reads. Not all reads are used by the CPU.

L3 Statistic Collectors from Software

- In a production board where DEBUG connectivity is difficult to achieve, a spare CPU core and spare timer can be used to read the StatColl registers at regular intervals.
- Eg. To capture the BW profile of a given initiator use the following steps:
  - Set up the StatColl for the initiator to capture number of bytes transferred.
  - Configure a timer to maintain ‘x’ us time gap.
  - Read the number of bytes transferred every ‘x’ us at the initiator ports (read + write) or the destination memory from the StatColl register.
  - The number of bytes obtained is divided by ‘x’ us to get the average BW within the ‘x’ us.
  - The process is repeated multiple times to generate a bandwidth profile.
- x=100 us is found to give good granularity for analyzing the traffic.



EMIF Counters

- The EMIF_PERFORMANCE_COUNTER_1 and EMIF_PERFORMANCE_COUNTER_2 registers are used to monitor or calculate the EMIF Controller bandwidth and efficiency.
- Controller using EMIF_PERFORMANCE_COUNTER_CONFIG

Bit	31	30	29 28 27 26 25 24 23 22 21 20	19 18 17 16	15	14	13 12 11 10 9 8 7 6 5 4	3 2 1 0
Field	CNTR2_MCONNID_EN	CNTR2_REGION_EN	Reserved	CNTR2_CFG	CNTR1_MCONNID_EN	CNTR1_REGION_EN	Reserved	CNTR1_CFG
Reset Value	0	0	0	0001	0	0	0	0000
Type	rw	rw	ro	rw	rw	rw	ro	rw

cntrN_cfg	cntrN_region_en	cntrN_mconnid_en	Description
0x0	0x0	0x0 or 0x1	**Count total SDRAM accesses.**
0x1	0x0	0x0 or 0x1	**Count total SDRAM activates.**
0x2	0x0 or 0x1	0x0 or 0x1	**Count total reads.**
0x3	0x0 or 0x1	0x0 or 0x1	**Count total writes.**
0x4	0x0	0x0	**Count number of m_clk cycles OCP Command FIFO is full.**
0x5	0x0	0x0	**Count number of m_clk cycles OCP Write Data FIFO is full.**
0x6	0x0	0x0	**Count number of m_clk cycles OCP Read Data FIFO is full.**
0x7	0x0	0x0	**Count number of m_clk cycles OCP Return Command FIFO is full.**
0x8	0x0 or 0x1	0x0 or 0x1	**Count number of priority elevations.**
0x9	0x0	0x0	**Count number of m_clk cycles that a command was pending.**
0xA	0x0	0x0	**Count number of m_clk cycles for which the memory data bus was transferring data.**
0xB – 0xF	**0x0**	**0x0**	**Reserved for future use.**

EMIF Counter Filtered on an ID

- EMIF_PERFORMANCE_COUNTER_MASTER_REGION_SELECT:

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	MCONNID2								Reserved						REGION_SEL2		MCONNID1								Reserved						REGION_SEL1	
Reset Value	00000000								0						00		00000000								0						00	
Type	Rw								ro						rw		rw								ro						rw	

- Details on conn ID can be seen from the Table 30-53. Statistics Collector Master Address Mapping of the TRM.
- For details on region select look at Table 14-101. MAddrSpace Mapping in the TDA2x TRM.
- No Debug SS support is present to capture this counter. A spare core can be used to monitor this value every ‘x’ us.

Bandwidth Knobs at IP Level

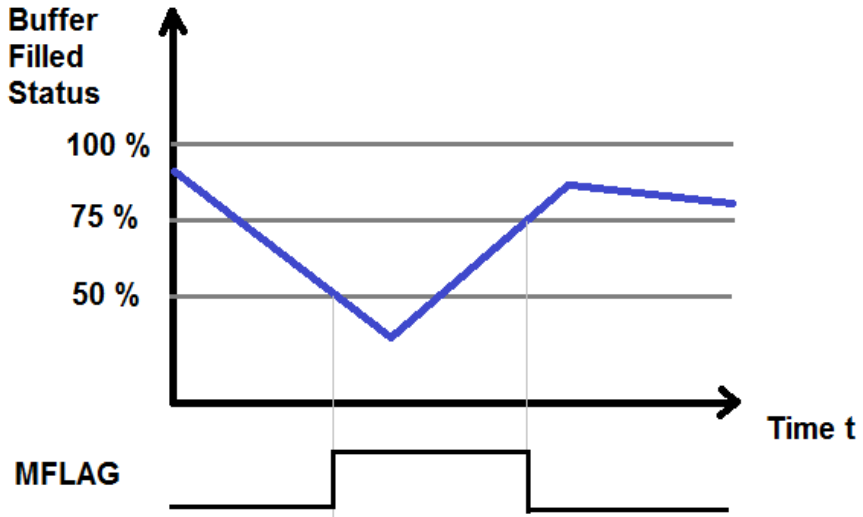
- Two main knobs at the IP level which can given priority to the traffic it generates:
  - **IP Driven MFLAG/MReqPrio:** Valid for DSS, VIP, VPE, DSP MDMA, EDMA and EVE TC0/TC1.
  - **Control Module: L3 Pressure:** Valid for MPU, DSP1, DSP2, IPU1, PRUSS1, GPU P1, GPU P2

IP Controlled MFLAG

- **DSP EDMA + MDMA**
  - EVTOUT[31] and EVTOUT[30] are used for generation of MFLAGs dedicated to the DSP MDMA and EDMA ports, respectively.
  - EVTOUT[31/30] = 1 → Corresponding MFLAG is high.
- **EVE TC0/TC1**
  - For EVE port 1 and port 2 (EVE TC0 & TC1) MFlag is driven by **evex_gpout[63]** and **evex_gpout[62]** respectively.
  - evex_gpout[63] is connected to DMM_P1 and EMIF.
  - evex_gpout[62] is connected to DMM_P2 and EMIF.
- **VIP/VPE**
  - In the VIP/VPE Data Packet Descriptor Word 3, can set the priority in [11:9] bits.
  - This value is mapped to OCP Reqinfo bits.
  - 0x0 – Highest Priority, 0x7 – Lowest Priority.
  - VIP Has Dynamic Mflag specific scheme based on internal FIFO status
    - Based on HW set margins to overflow/underflow
    - Enable by default, no MMR control

IP Controlled MFLAG

- **DSS**
  - DSS has 4 display read pipes (Graphics , Vid1 , Vid2 , Vid3) and 1 write pipe (WB) .
  - DSS will drive MFlag if any of read pipes are made high priority and FIFO levels are below low threshold for high priority display pipe.
  - VIDx have 32 KB FIFO & GFX has 16 KB FIFO.
  - FIFO threshold is measured in terms of 16 byte word.
  - Recommended settings for high and low threshold are 75% and 50 % respectively.

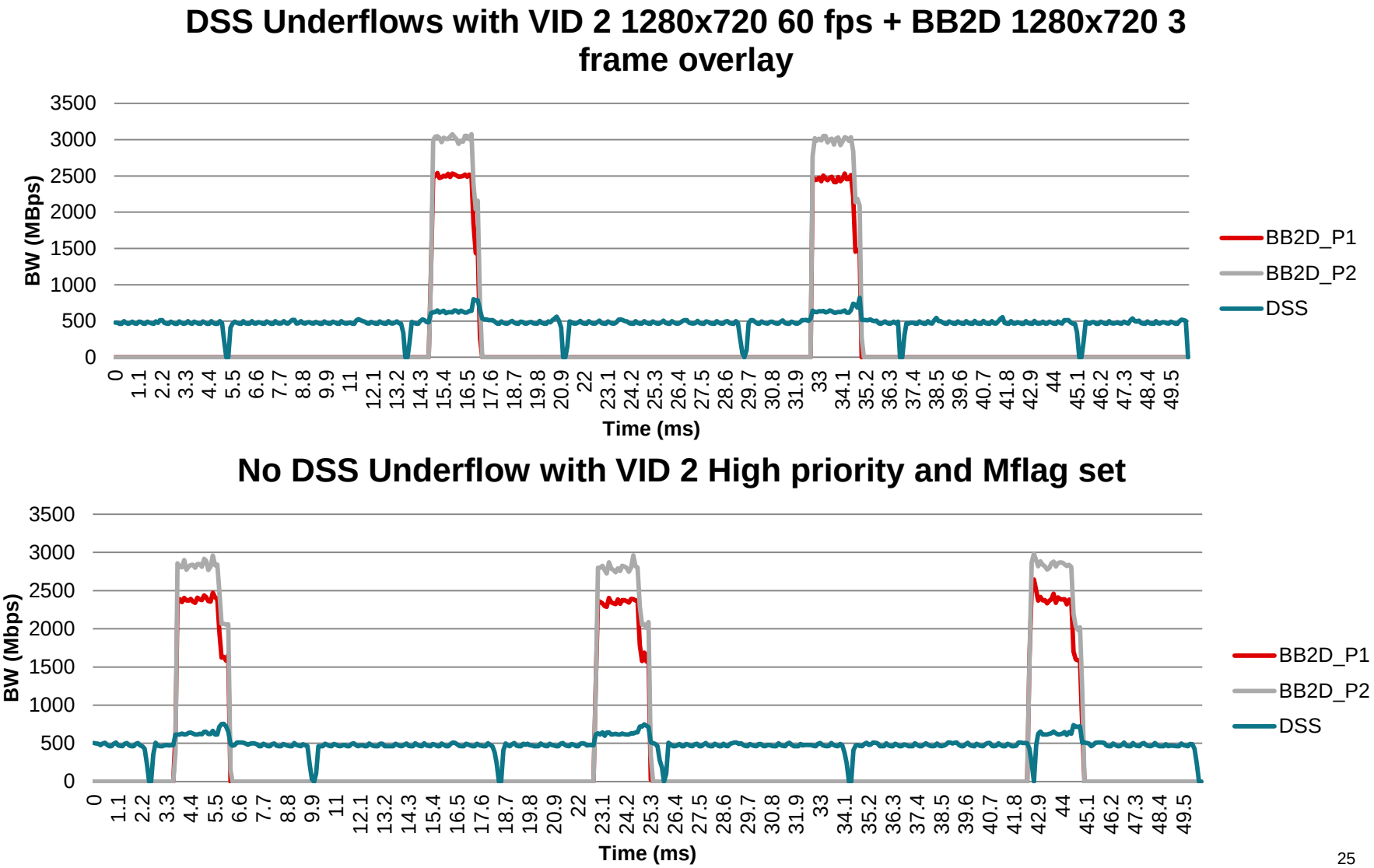


DSS MFLAG Programming Model

- Enable MFlag Generation DISPC_GLOBAL_MFLAG_ATTRIBUTE
  - `DISPC_GLOBAL_MFLAG_ATTRIBUTE = 0x2;`
- Set Video Pipe as High Priority DISPC_VIDx_ATTRIBUTES[23]
  - `DISPC_VID1_ATTRIBUTES | = (1<<23);`
  - `DISPC_VID2_ATTRIBUTES | = (1<<23);`
  - `DISPC_VID3_ATTRIBUTES | = (1<<23);`
- Set Graphics Pipe as High Priority DISPC_GFX_ATTRIBUTES[14]
  - `DISPC_GFX_ATTRIBUTES | = (1<<14);`

- GFX threshold 75 % HT , 50 % LT
  - `DISPC_GFX_MFLAG_THRESHOLD = 0x03000200;`
- VIDx threshold 75 % HT , 50 % LT
  - `DISPC_VID1_MFLAG_THRESHOLD = 0x06000400;`
  - `DISPC_VID2_MFLAG_THRESHOLD = 0x06000400;`
  - `DISPC_VID3_MFLAG_THRESHOLD = 0x06000400;`

Example – DSS MFLAG



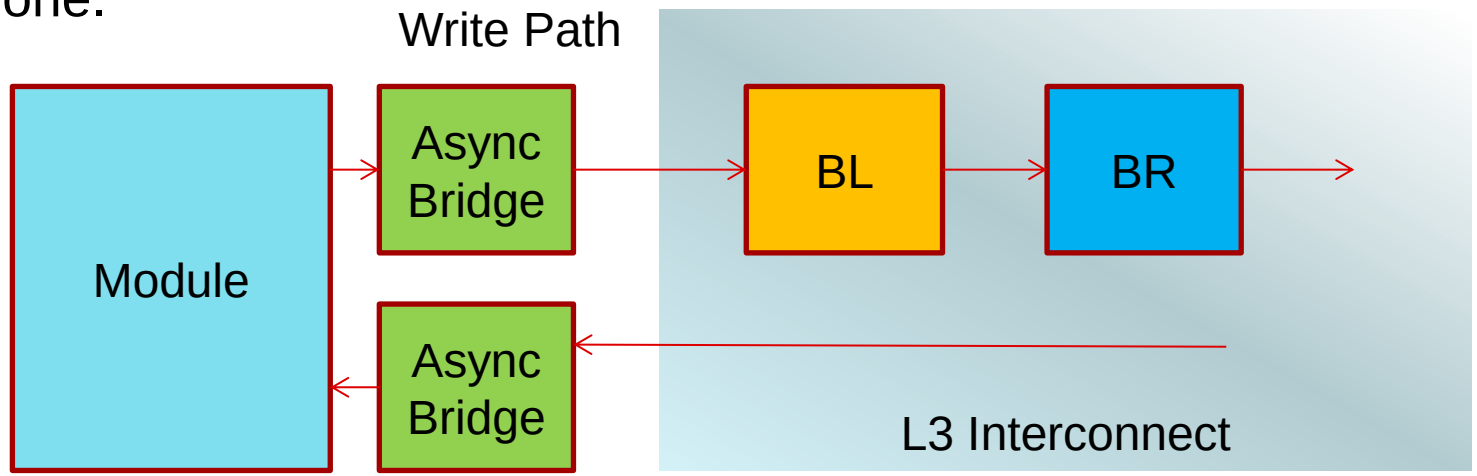
Control Module Initiator Pressure

- The **CTRL_CORE_L3_INITIATOR_PRESSURE_1** to **CTRL_CORE_L3_INITIATOR_PRESSURE_4** registers are used for controlling the priority of certain initiators on the **L3_MAIN**.
- 0x3 – Highest Priority/Pressure
- 0x0 – Lowest Priority/Pressure
- For MPU, DSP1, DSP2, IPU1, PRUSS1, GPU P1, GPU P2

Bandwidth knobs at the Interconnect

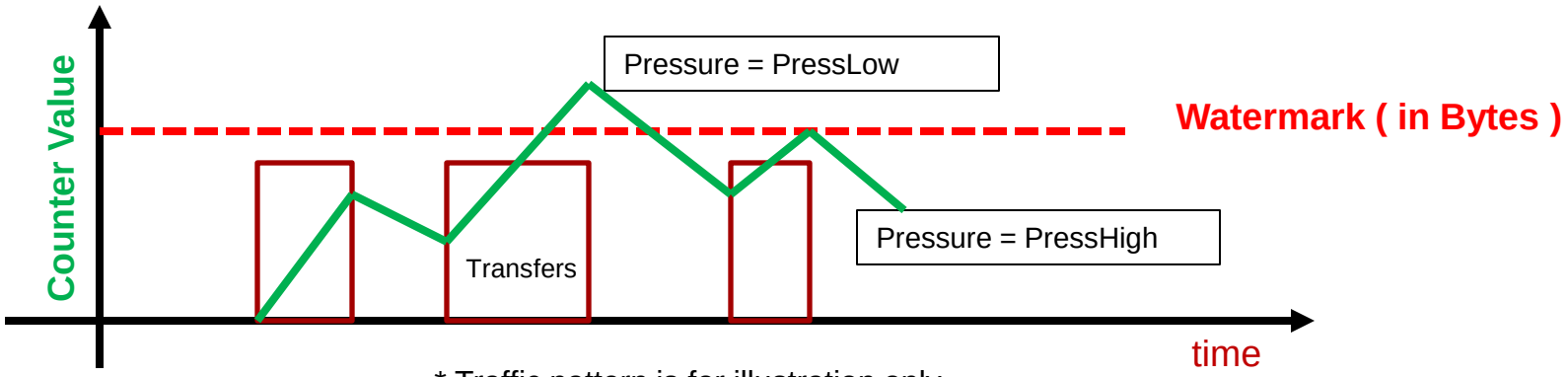
- Bandwidth Regulators (BR) – Used to regulate the BW from an initiator with in a range.
- Bandwidth Limiters (BL) – Used to limit the BW from an Initiator to the given BW cap.

- These are placed at the Initiator Write ports.
- Some initiators may have both BL followed by BR. This is because in the low pressure region of BR only a best case effort of maintaining the BW is done.



Bandwidth Regulator

- For a given initiator.
- Increases pressure when the actual consumed bandwidth is lower than expected bandwidth
- Decreases pressure once the expected bandwidth is reached.
- Mechanism
  - A counter is incremented by number of bytes transferred (read + write)
  - A Watermark/threshold value for the counter is programmed.
  - When counter value is less than Watermark high pressure (as define by PressHigh) is applied for the given packet.
  - Else low pressure (as defined by PressLow) is applied for the given packet.

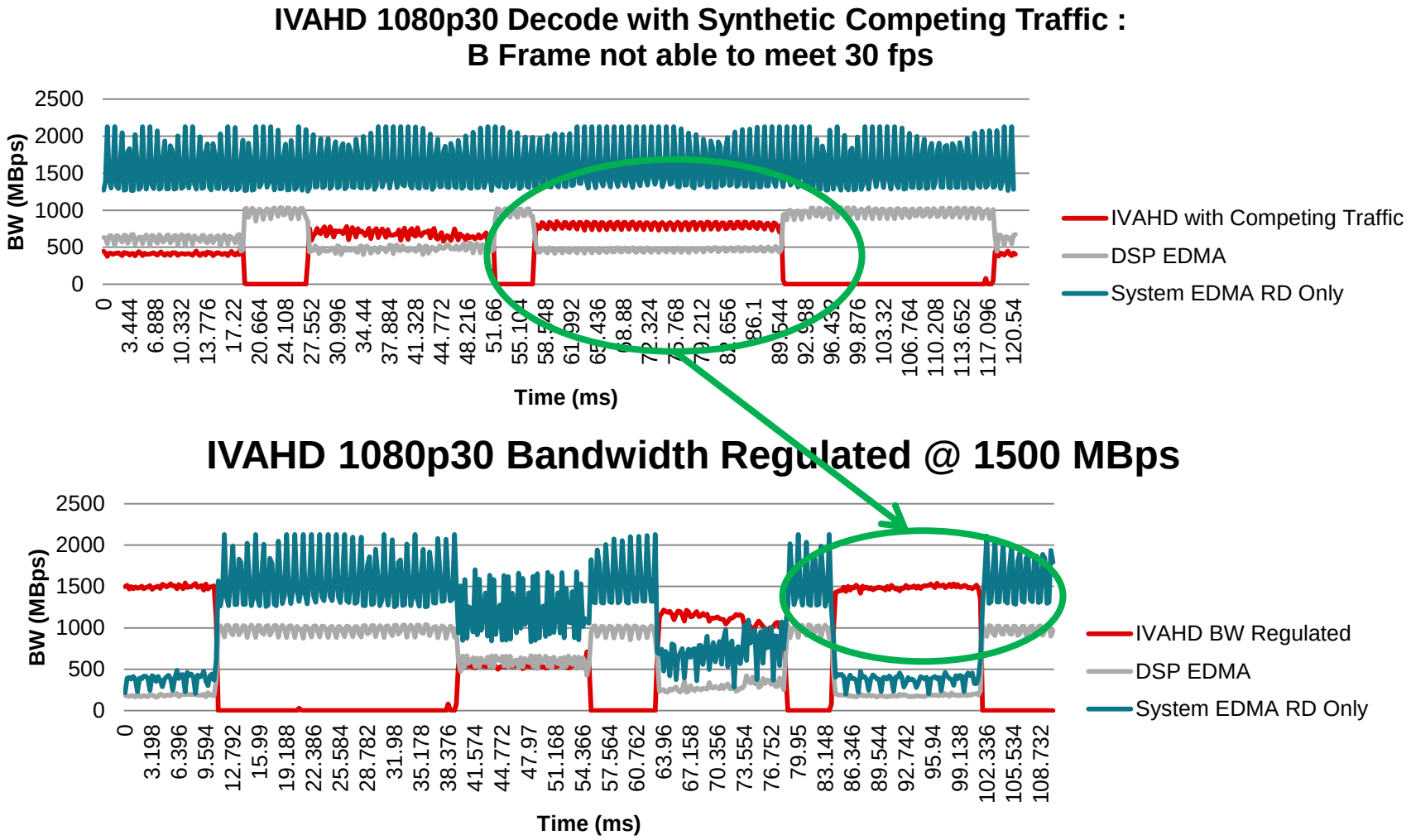


* Traffic pattern is for illustration only

Bandwidth Regulator Programming Model

- Program the BR register with desired bandwidth divided by the resolution.
- Resolution = L3 Frequency/ 2^5 = 266MHz/ 32 = 8.3125.
  - **`L3_BW_REGULATOR_BANDWIDTH =`**
    `(int)(ceil(Required Bandwidth/8.3125));`
- Set the watermark register by multiplying the required bandwidth in MBps with the time window size in micro seconds:
  - **`L3_BW_REGULATOR_WATERMARK =`**
    `(Window size in us * Required Bandwidth);`
- The pressure that is applied to the initiator packets when the counter value is lower than watermark (High_pres) and higher than watermark (Low_press) can be read from **`L3_BW_REGULATOR_PRESS.`**
  - By default : 0x3 – High Level, 0x0 – Low level.
- Write to the clear history to take in the new values.
  - **`L3_BW_REGULATOR_CLEARHISTORY = 0x1;`**

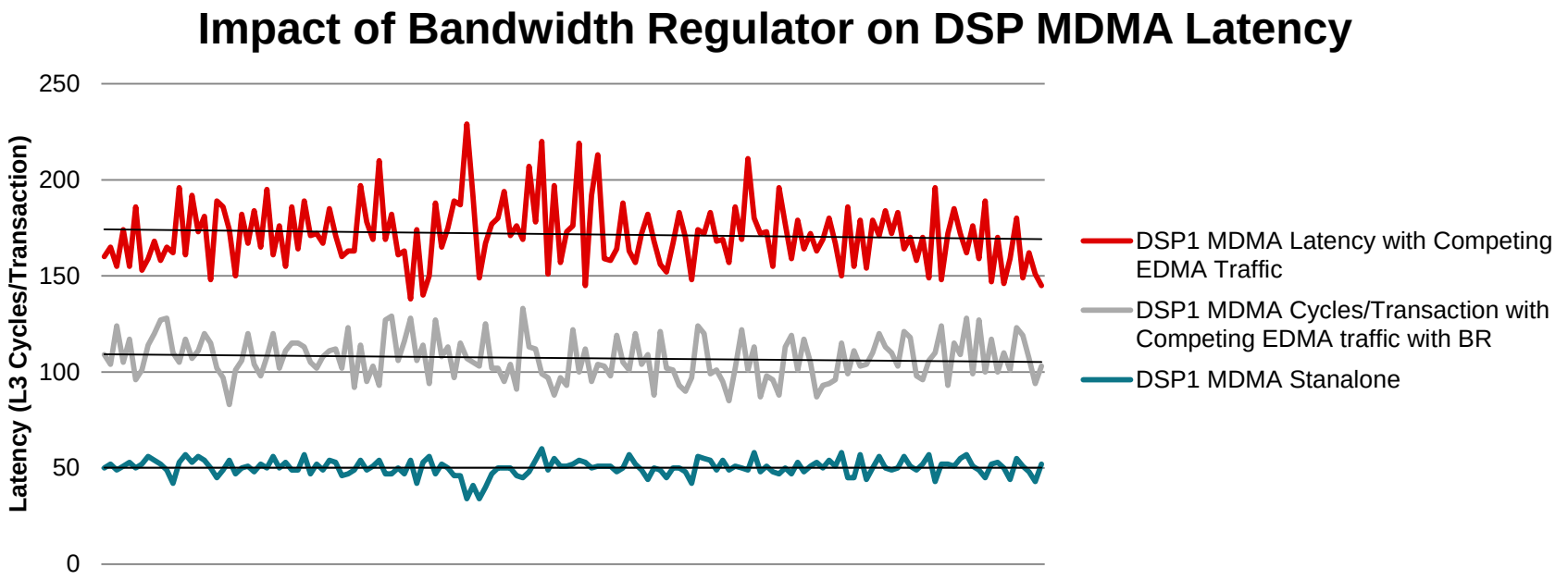
Example 1 : BR on IVAHD (Impact on BW)



L3 Frequency = 266 MHz, Resolution = 266/2^5 = 266/32 = 8.3125
L3_BW_REGULATOR_BANDWIDTH[15:0] BANDWIDTH = ceil(1500 MBps / 8.3125) = 181d = 0xB5
L3_BW_REGULATOR_WATERMARK[11:0] WATERMARK = 1500MBps * 1us = 1500d = 0x5DC.

Example 2: DSP MDMA BR (Impact on Latency)

- Optimized memcopy is executed on DSP : average BW of 2.5 GBps standalone traffic.
- Add a 2 TC edma transfer running from the DSP EDMA
- Observed the DSP1 MDMA bandwidth drops to 981 MBps.
- Setting the BW regulator to 1.5 GBps (with a window of 1 us) leads the latency of the DSP1 MDMA port to improve from ~180 cycles/transaction to ~110 cycles/transaction.



IPs supporting Bandwidth Regulator

- Bandwidth regulator available for below IPs
  - MMU2
  - EVE1, EVE2, EVE3, EVE4 – both TC0 TC1
  - DSP1, DSP2 MDMA (CPU access port)
  - DSP1, DSP2 EDMA
  - IVA
  - GPU
  - GMAC
  - PCIe
- Note, Bandwidth regulator will override any Mflag setting done via Control module for the IP
- Refer to TRM for exact bandwidth regulator base addresses

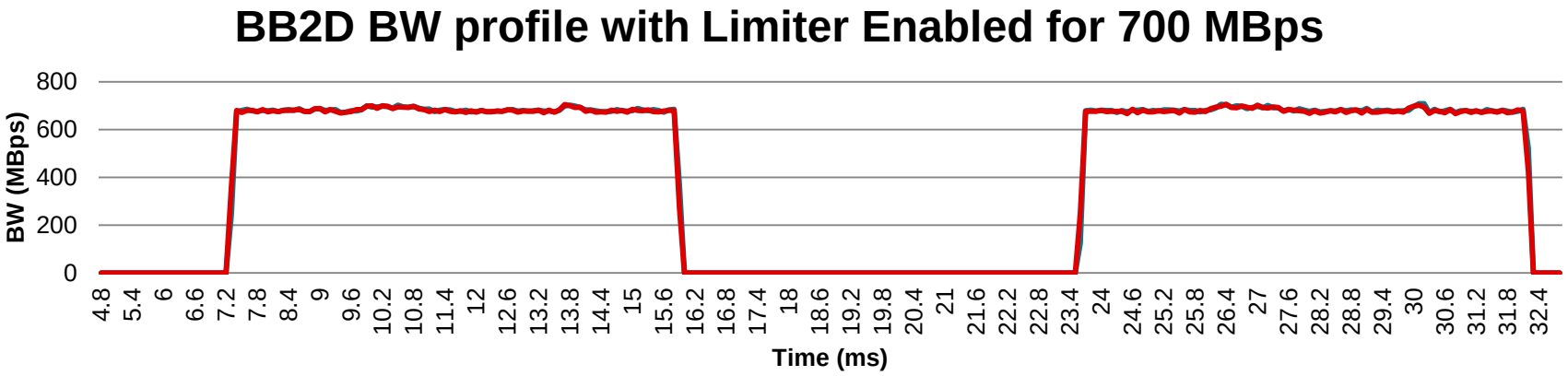
Bandwidth Limiter

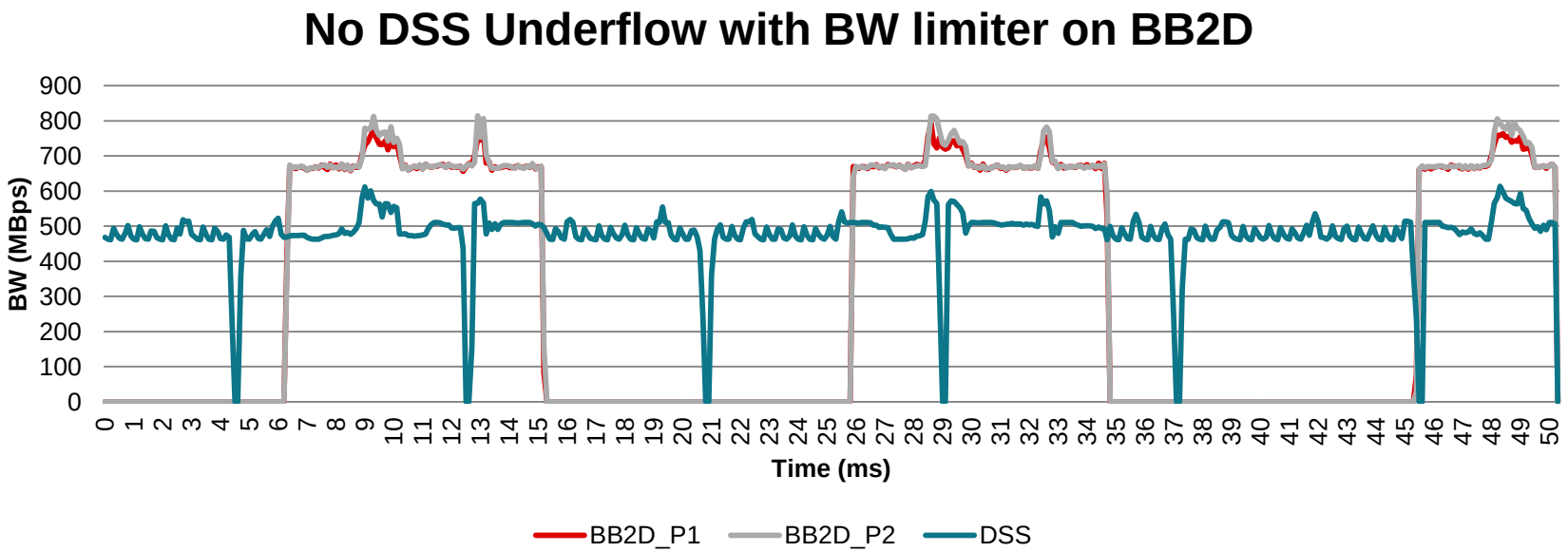
- Bandwidth Limiter mechanism limits a particular initiator from consuming excessive bandwidth.
- This mechanism is implemented for
  - MMU1
  - SYSTEM_EDMA
  - VPE
  - GPU
- The bandwidth limiter regulates the packet flow in the L3_MAIN interconnect by applying flow control when a user-defined bandwidth limit is reached.
- The next packet is served only after an internal timer expires, thus ensuring that traffic does not exceed the allocated bandwidth.
- Bandwidth limiter can be used with a watermark mechanism that allows traffic to temporarily exceed the peak bandwidth.

Bandwidth Limiter Programming Model

- Calculate the Bandwidth integer and fractional numbers with respect to 2^5 = 32.
  - `bandwidth = int(Required Limit bandwidth / 8.3125);`
  - `bandwidth_int = ( bandwidth & 0xFFFFFFE0 ) >> 5;`
  - `bandwidth_frac = ( bandwidth & 0x1F );`
- Program the integer and fractional values in the registers.
  - `L3_BW_LIMITER_BANDWIDTH_FRACTIONAL = bandwidth_frac;`
  - `L3_BW_LIMITER_BANDWIDTH_INTEGER = bandwidth_int;`
- Program the watermark to have 0x0 to not allow additional traffic than the required.
  - `L3_BW_LIMITER_WATERMARK = 0x0;`
- Clear the history to fill the newly populated values:
  - `L3_BW_LIMITER_CLEARHISTORY = 0x1;`

Bandwidth Limiter: Example





Bandwidth Knobs at DMM

- The DMM has the following BW knobs:

- **DMM PEG : Priority Extention Generator**
  - Forwards the packet from the initiator to the EMIF with an updated MReqPrio (OCP MReqInfo)

- **DMM Emergency**
  - Gives higher priority than rest of the initiators to real time traffic of VIP and DSS based on whether MFLAG is high.

DMM PEG Priority

- Reads the required priority from the PEG LUT programmed by the software.
- Each initiator has a 3 bit field (0 ... 7) PEG Priority , 0 is highest priority, 7 is lowest
- Priority determines prioritization of data transfers in EMIF

Configuring DMM PEG

DMM_PEG_PRIO7 : 0x63C															
31	30…28	27	26…24	23	22…20	19	18…16	15	14…12	11	10…8	7	6…4	3	2…0
$PRIO_{63}$		$PRIO_{62}$		$PRIO_{61}$		$PRIO_{60}$		$PRIO_{59}$		$PRIO_{58}$		$PRIO_{57}$		$PRIO_{56}$	
W7	P7	W6	P6	W5	P5	W4	P4	W3	P3	W2	P2	W1	P1	W0	P0

DMM_PEG_PRIO0 : 0x620															
31	30…28	27	26…24	23	22…20	19	18…16	15	14…12	11	10…8	7	6…4	3	2…0
$PRIO_{7}$		$PRIO_{6}$		$PRIO_{5}$		$PRIO_{4}$		$PRIO_{3}$		$PRIO_{2}$		$PRIO_{1}$		$PRIO_{0}$	
W7	P7	W6	P6	W5	P5	W4	P4	W3	P3	W2	P2	W1	P1	W0	P0

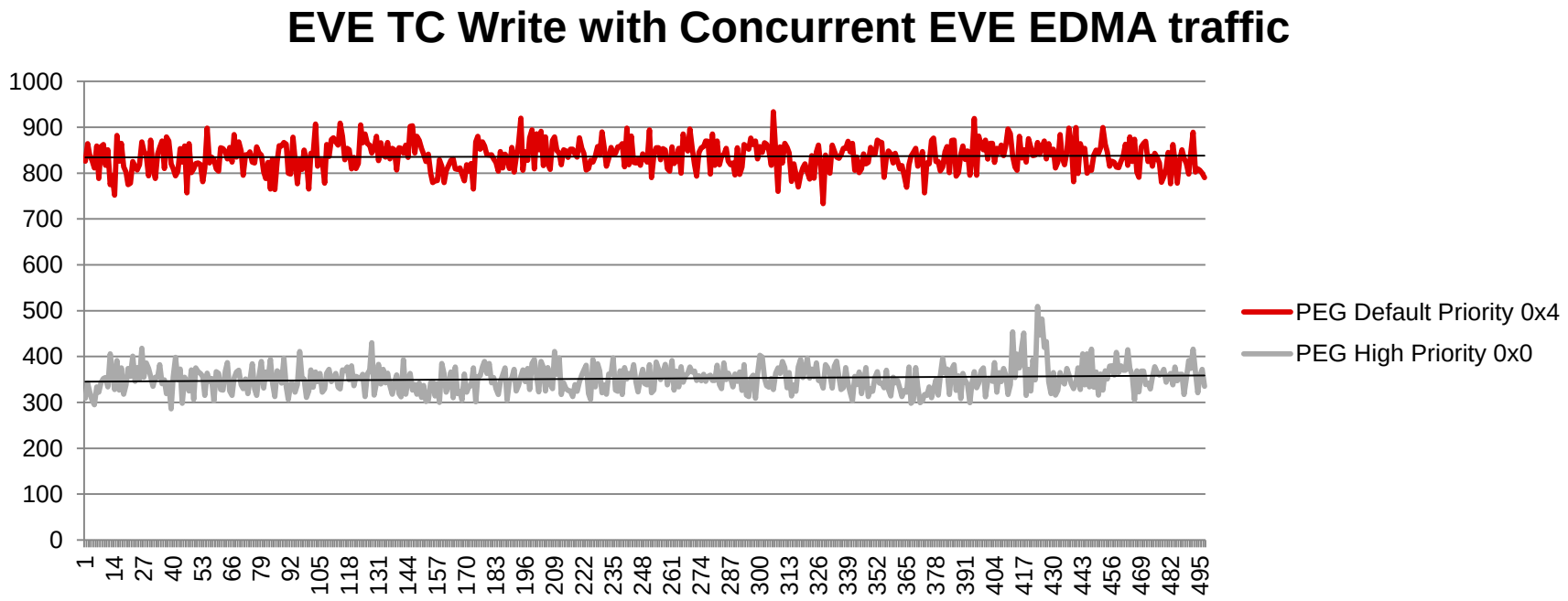
the 3-bit priority coded on the 3 least significant bits (0 is the higher priority)
A “W” field-specific active-high local write enable bit, always read as 0

The role of the W bit is to allow the modification of a single entry without requiring a read-modify-write sequence.

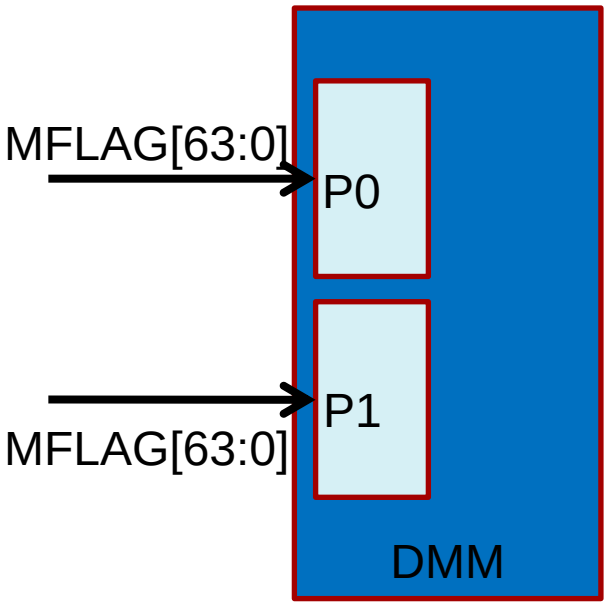
Refer to TRM Section to “PEG Description” under DMM to know the mapping from 6 bit MConnId to field of DMM_PEG_PRIO Registers.

DMM Peg Priority : Example

- Concurrent traffic from 4 x EDMA TC channels.
- 1 TC doing write to DDR and 3 TC doing read from DDR.
- Avg. Write Latency Increased to ~850 L3 cycles/Transaction.
- Setting DMM PEG to 0x0 for EVE TC helps decrease latency to ~350 L3 Cycles/Transaction.



DMM Emergency Field



- The initiators with MFLAG set will be classified as higher priority.
- Weighted round robin algorithm is used for arbitration between high priority and other initiators.
- Set DMM_EMERGENCY[0] to run this arbitration scheme.
- Weight is set in the field DMM_EMERGENCY[20:16] WEIGHT

Bandwidth Knobs at EMIF

- The bandwidth knobs at EMIF are as follows:

- **EMIF_OCP_CONFIG:** Helps balance the traffic between the MPU and the system traffic.

- **EMIF Class of Service:** Helps prioritize the traffic from an initiator. Filtering is done on the Initiator Conn ID.

- **EMIF Read vs Write Threshold:** Can prioritize read commands over writes and vice versa.

EMIF_OCP_CONFIG

Bit	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved			SYS_THRESH_MAX				MPU_THRESH_MAX				Reserved				Reserved																
Reset Value	0			0111				0111				0				0																
Type	ro			rw				rw				ro				ro																

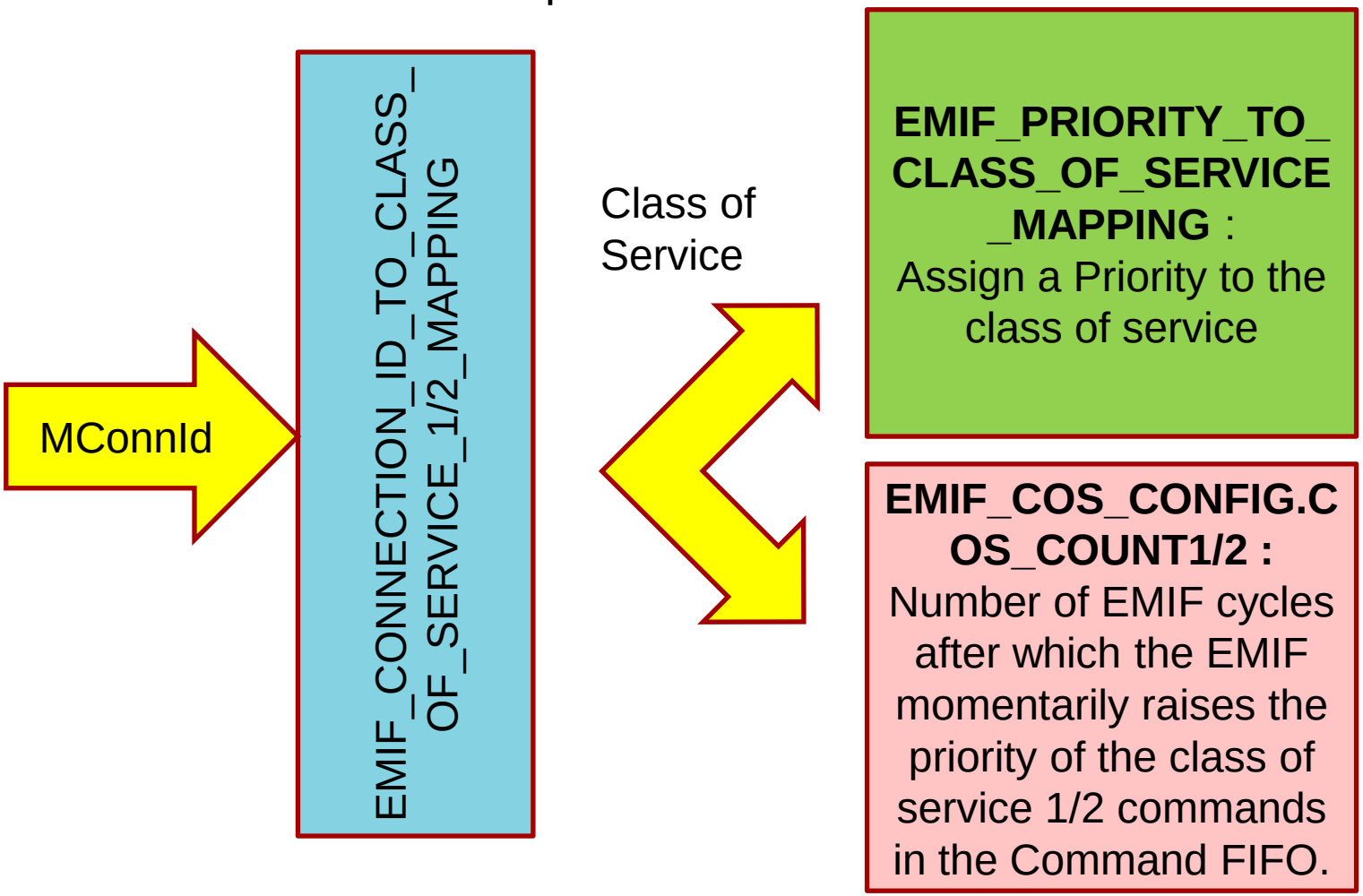
The command FIFO stores all the commands coming in on the local command interface.

The allocation of entries in the command FIFO is programmable per local interface using the following bit fields:

- EMIF_OCP_CONFIG[27:24] SYS_THRESH_MAX
- EMIF_OCP_CONFIG[23:20] MPU_THRESH_MAX

EMIF Class of Service (CoS)

- Two Class of Services present in EMIF.



EMIF_READ_WRITE_EXECUTION_THRESHOLD

Bit	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Field	Reserved																		WR_THRSH					Reserved			RD_THRSH				
Reset Value	0																		00011					0			00101				
Type	ro																		rw					ro			rw				

Usually reads are given more preference than writes:

RD_THRSH: Number of SDRAM read bursts after which the EMIF arbitration will switch to executing write commands.

WR_THRSH: Number of SDRAM write bursts after which the EMIF arbitration will switch to executing read commands.

Resources for BW optimization

- Vayu_CCS_Trace_Debug.pdf: https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.4960.9266
- CCS_DEBUG_SCREEN_CAST1.7z: https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.4992.43328
- CCS_DEBUG_SCREEN_CAST2.7z: https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.48672.25740
- CCS_DEBUG_SCREEN_CAST3.7z: https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.26816.14527
- A Guide to Debugging with CCS on the DRA7xx TDA2xx and TDA3xx family of deviceshttps://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.4848.9376
- https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.43616.1592 – TDA2xx Performance Application Note
- https://cdds.ext.ti.com/ematrix/common/emxNavigator.jsp?objectId=28670.42872.40204.59666 - SPRABX1 Quality of Service Knobs for TDA2x Family of Devices

THANK YOU

BACKUP

L3 Statcoll Initial Programming for BW Measurement

- Refer Chapter on On-Chip Debug Support -> System instrumentation -> L3 NOC Statistics Collector for mapping of the initiator/slave port to the StatColl instance.
- L3_STCOL_EN. EN = 1
- L3_STCOL_SOFTEN. SOFTEN = 1
- L3_STCOL_REQEVT. REQEVT = 0x5 -> For bandwidth (list of valid values given in the TRM)
- L3_STCOL_RSPEVT. RSPEVT = 0x5 -> For bandwidth (list of valid values given in the TRM)
- L3_STCOL_EVTMUX_SEL0/1/2/3 = Mux Select Value (Read this from initiator/slave port mapping).
- L3_STCOL_OP_i_SEL = 0x2 -> Add to counter the selected event info value (list of valid values given in the TRM)
- L3_STCOL_OP_i_EVTINFOSEL = 0x0 -> Select len from event info list (gets the number of bytes in each packet)
- L3_STCOL_FILTER_i_GLOBALEN. FILTER_i_GLOBALEN = 1 -> Enable filter.
- L3_STCOL_FILTER_i_EN_k. FILTER_i_EN0 = 1 -> Enable Filter stage 0.
- L3_STCOL_DUMP_MODE. DUMP_MANUAL = 1
- To read the register:
  - L3_STCOL_SOFTEN. SOFTEN = 0
  - Read L3_STCOL_DUMP_CNTi
  - L3_STCOL_SOFTEN. SOFTEN = 1

L3 Statcoll Initial Programming for Latency Measurement

- Refer Chapter on On-Chip Debug Support -> System instrumentation -> L3 NOC Statistics Collector for mapping of the initiator/slave port to the StatColl instance.
- L3_STCOL_EN. EN = 1
- L3_STCOL_SOFTEN. SOFTEN = 1
- L3_STCOL_REQEVT. REQEVT = 0x8 -> For Latency (list of valid values given in the TRM)
- L3_STCOL_RSPEVT. RSPEVT = 0x8 -> For Latency (list of valid values given in the TRM)
- L3_STCOL_EVTMUX_SELi = Mux Select Value (Read this from initiator/slave port mapping).
- L3_STCOL_OP_i_SEL = 0x2 -> Add to counter the selected event info value (list of valid values given in the TRM)
- L3_STCOL_OP_i_EVTINFOSEL = 0x2 -> Select latency if available from event info list
- L3_STCOL_FILTER_i_GLOBALEN. FILTER_i_GLOBALEN = 1 -> Enable filter.
- L3_STCOL_FILTER_i_EN_k. FILTER_i_EN0 = 1 -> Enable Filter stage 0.

- L3_STCOL_OP_(i+1)_SEL = 0x0 -> Increment counter on each mask/match filter hit (list of valid values given in the TRM)
- L3_STCOL_OP_(i+1)_EVTINFOSEL = 0x2 -> Select latency if available from event info list
- L3_STCOL_FILTER_(i+1)_GLOBALEN. FILTER_i_GLOBALEN = 1 -> Enable filter.
- L3_STCOL_FILTER_(i+1)_EN_k. FILTER_i_EN0 = 1 -> Enable Filter stage 0.
- L3_STCOL_DUMP_MODE. DUMP_MANUAL = 1

- To read the register:
  - L3_STCOL_SOFTEN. SOFTEN = 0
  - Read L3_STCOL_DUMP_CNTi
  - Read L3_STCOL_DUMP_CNT(i+1)
  - L3_STCOL_SOFTEN. SOFTEN = 1