Unified DMA Controller (UDMA) Overview

K3 Processors
18th March 2019
V1.1

Texas Instruments

Agenda

- UDMA (NAVSS) Overview
  - Features (Comparison with EDMA)
  - Introduction to terminology: Channels, UTC, DRU, RA, IA, IR, Events, Proxy
  - TR and TRPD Formats
- UDMA Software Architecture
  - API Sequence
- UDMA Memcpy Example
  - Code Walk Through
  - UDMA LLD API Overview
- Back-up
  - UDMA Hardware (In Details: Covered by Driver)

Texas Instruments

UDMA Features vs EDMA

Features	UDMA	EDMA
Address Range	64b source and destination DMA buffer address	32b source and destination DMA buffer address
Transfer Dimension	Two modes of operations • Packet mode (Peripherals and scatter gather) • Transfer Request (TR) mode (Memcpy, Video/Audio peripherals) Up to four transfer dimensions in TR mode (DIM0, DIM1, DIM2, DIM3)	Three transfer dimensions (A, B, C)
Transfer Synchronization	• DIM0 synchronized transfer: 1D • DIM1 synchronized transfer: 2D • DIM2 synchronized transfer: 3D • DIM3 synchronized transfer: 4D	• A-synchronized transfers: one-dimension serviced per event • AB-synchronized transfers: two-dimensions serviced per event
Buffer Indexing	Independent indexes and count on source and destination for DIM0/1/2/3 – This gives flexibility to read/write differently based on memory layout (gives optimized memory access)	Independent indexes on source and destination. Same ACNT, BCNT, CCNT on source and destination
Addressing Mode	Increment or FIFO transfer addressing modes	Increment or FIFO transfer addressing modes
Linking	TR Descriptor and ring accelerator allows multiple DMA transfers to be sequenced on the same UDMA channel limited only by available system memory space to store descriptors	Linking mechanism allows multiple DMAs to be sequenced on the same EDMA Channel limited by number of PaRAM entries
Chaining	Chaining allows multiple transfers to execute simultaneously on multiple UDMA channels with one event. Chaining achieved using event steering from event source to event sink on Event Transport Lane (ETL)	Chaining allows multiple transfers to execute simultaneously on multiple EDMA channels with one event

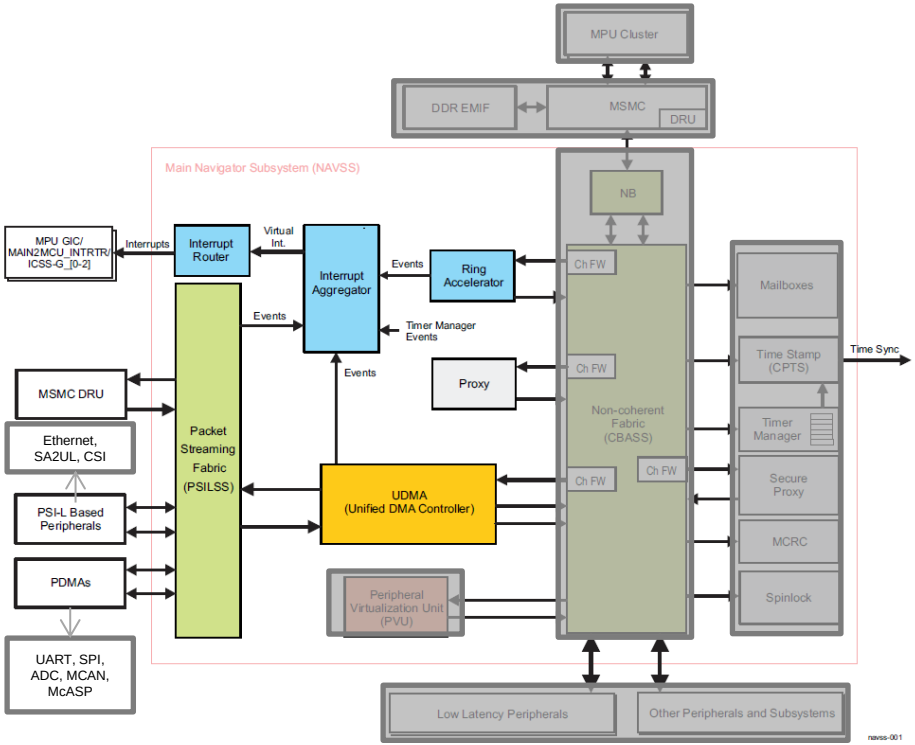
UDMA Features vs EDMA Contd…

Features	UDMA	EDMA
Events and Interrupts	Event generation for the following • Transfer completion, intermediate transfer completion • Error conditions Events can be converted to interrupts using Interrupt Aggregator and Interrupt Router	Interrupt generation for the following: • Transfer completion, intermediate transfer completion • Error conditions
Logical Channels	• 140 TX channels (memory to peripheral) in J721E Main NAVSS • 140 RX channels (peripheral to memory) in J721E Main NAVSS • A pair of RX+TX channels is used for memory to memory DMA **NOTE:** The number of RX and TX channel can change from SoC to SoC and from Main NAVSS to MCU NAVSS. Refer to SoC TRM for exact numbers	16 ~ 128 EDMA channels depending on the SoC Same channel can be used for RX or TX
Triggers	Synchronization based on • Manual synchronization (CPU write to UDMA channel trigger register). • Chain synchronization (completion of one transfer triggers another transfer using events)	Synchronization based on • Event from peripheral • Manual synchronization (CPU write to event set registers EDMA_TPCC_ESR and EDMA_TPCC_ESRH) • Chain synchronization (completion of one transfer triggers another transfer)
Performance	32 DRU DMA channels for high throughput memory to memory DMA	8 QDMA channels
Parameter Memory	DMA operation described by a packet descriptor or TR descriptor in system memory. Number of descriptors only limited by amount of system memory	DMA operation described by a PaRAM set, up to 512 PaRAM set in a EDMA controller

NAVSS Architecture (UDMA Focused) and Terminology

- NAVSS (Navigator Subsystem)
  - Container which groups together components which are involved in providing DMA services in a SoC
- UDMA-C (Controller)
  - Triggers request and receives response from UTC (Channel controller), DRU, PDMA
- UTC (Unified Transfer Controller)
  - Received Transfer Request from UDMA and performs actual transfers (Transfer Controller)
- DRU (Data Routing Unit)
  - Special UTC for high performance data movement esp meant for C7x algorithms
- PDMA (Peripheral DMA)
  - Located close to peripherals
- PSILSS (Packet Streaming Interface)
  - Switch fabric: Pairing source/destination, routing PSI-L packets
- Key Features:
  - Split DMA: Needs a paired TX and RX channel and transfer request contains independent params for TX and RX
  - Multiple Instances in an SOC: Main NAVSS and MCU NAVSS (* varies from SOC to SOC)

Figure 10-1. NAVSS Overview

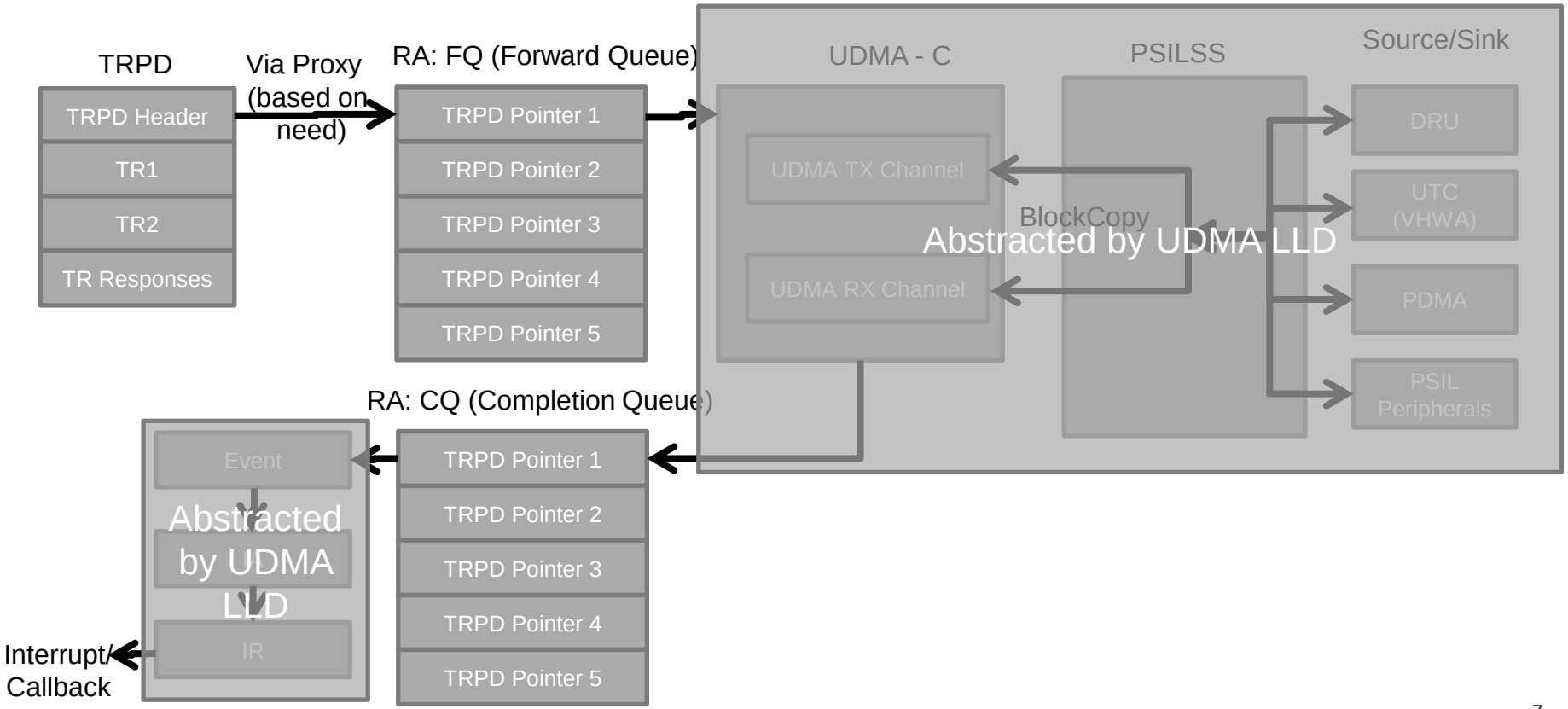


Texas Instruments

NAVSS Architecture (UDMA Focused) and Terminology Contd…

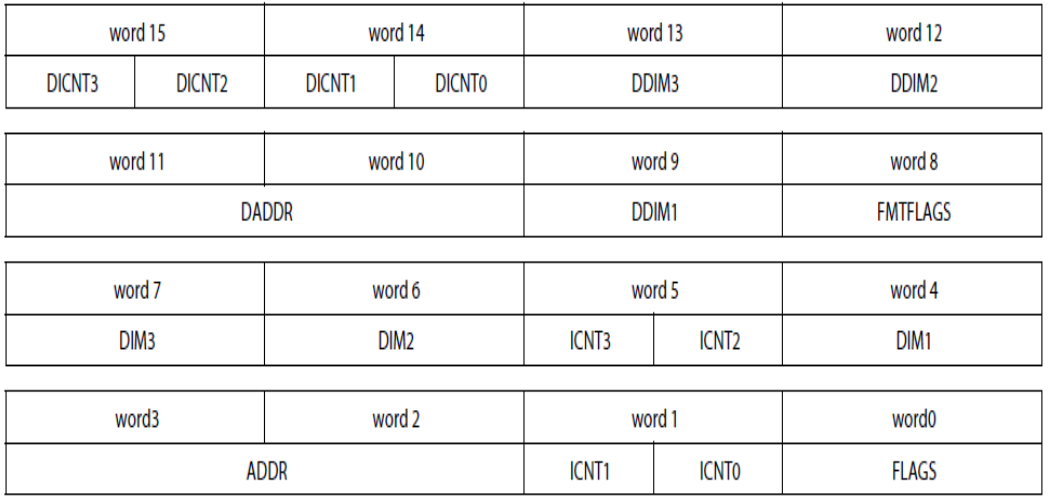
- RA (Ring Accelerator)
  - Mechanism to submit request to UDMA and get response back from UDMA
  - Manages queue and state for producer and consumer to exchange data (Note: not limited to UDMA alone)
- Proxy
  - Mechanism to access RA in an atomic way: Ex: 32-bit CPU like R5 can't perform a 64-bit single atomic write to RA
- Events
  - 16-bit unique global entity which can be generated by specific sources
  - Can be used to trigger interrupt to CPU via IA/IR or to trigger other transfers
  - Sources include: RA, DMA channel, Ring monitors, various error events
- IA (Interrupt Aggregator)
  - With so many events possible in a system (up to 64K), it is not possible to route all events to the CPU interrupt (which are usually limited and in ~100 range)
  - Mechanism to aggregate events: Up to 64 event aggregation per VINT (Virtual interrupt: not yet an interrupt to the CPU, see IR)
  - Supports polling mode as well without generating interrupts
- IR (Interrupt Router)
  - M:N mux (cross bar) to generate interrupt to various CPU in the system
    - where M is larger compared to N with sources from VINT, other NAVSS modules like Timer Manager
    - N interrupts to the CPU (N usually in the range of 128 to MPU/GIC, 32 to each of R5FSS; specific value varies from SOC to SOC)
  - There are multiple IR in the system which are identical in functionality – only source and destination differs. We will cover only NAVSS IR specifics in this training

UDMA Setup/Flow – At a High Level



Texas Instruments

Transfer Request (TR) Record

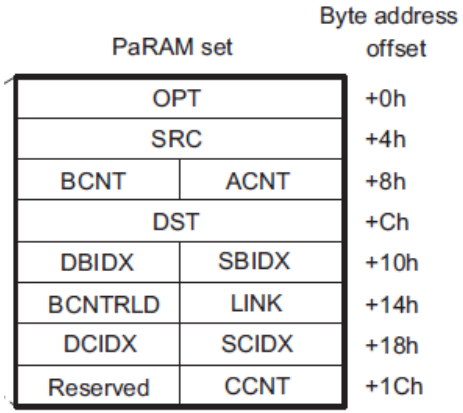


Size of TR is variable from 16 bytes to 64 bytes.
Specified via TR Type in FLAGS field

TR Type	Descriptrion
Type 0	1D (word0-3)
Type 1	2D (word0-4)
Type 2	3D (word0-6)
Type 3	4D (word0-8)
Type 5	Cache warm (word0-15) (MSMC DRU ONLY)
Type 8	4D Block Copy (word0-15)
Type 9	4D Block Copy with reformatting (word0-15) (MSMC DRU ONLY)
Type 10	2D Block Copy (word0-15)
Type 11	2D Block Copy with reformatting (word0-15) (MSMC DRU ONLY)
Type 15	4D Block Copy with reformatting and indirection (word0-15) (MSMC DRU ONLY)

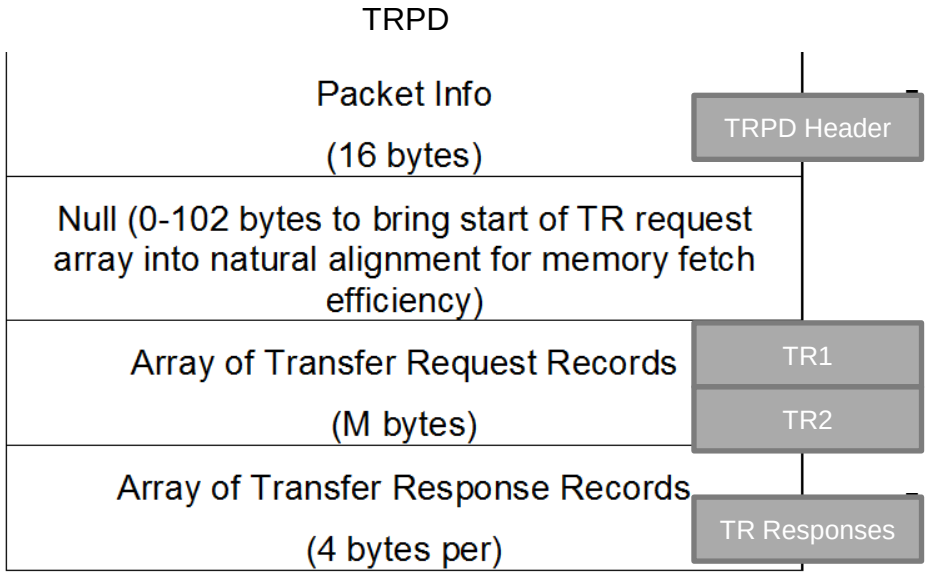
Comparison of TR Record with EDMA PaRAM

UDMA TR	EDMA PaRAM	Remarks
ADDR	SRC	64b address
DADDR	DST	64b address
ICNT0	ACNT	ICNT0 is 1st DIM for data @ ADDR
ICNT1	BCNT	ICNT1 is 2nd DIM for data @ ADDR
ICNT2	CCNT	ICNT2 is 3rd DIM for data @ ADDR
ICNT3	na	ICNT3 is 4th DIM for data @ ADDR
DIM1	SBIDX	2nd DIM stride in bytes for data @ ADDR
DIM2	SCIDX	3rd DIM stride in bytes for data @ ADDR
DIM3	Na	4th DIM stride in bytes for data @ ADDR
DICNT0/1/2/3	Na	ICNTx for DIMx for data @ DADDR
DDIM1/2/3	DBIDX / DCIDX / na	DIMx Stride in bytes for data @ DADDR
FLAGS	OPT	TR Type, Trigger type, Event condition
na	LINK	No equivalent in TR. Linking done via Ring Accelerator.
na	BCNTRLD	No equivalent in TR.
FMTFLAGS	na	Data reformatting options (used by MSMC DRU)



EDMA Params

TRPD (TR Packet Descriptor) Layout



- Packet Info important fields
  - Number of TR records
  - Reload enable
    - Loop to index “IDX” within TR records
  - TR Record Size
  - *Packet Return Queue (CQ RA)*
- TR Response
  - Status of TR – no error, transfer error, aborted
  - CMDID from TR.FLAGS – helps associate TR response to TR

Texas Instruments

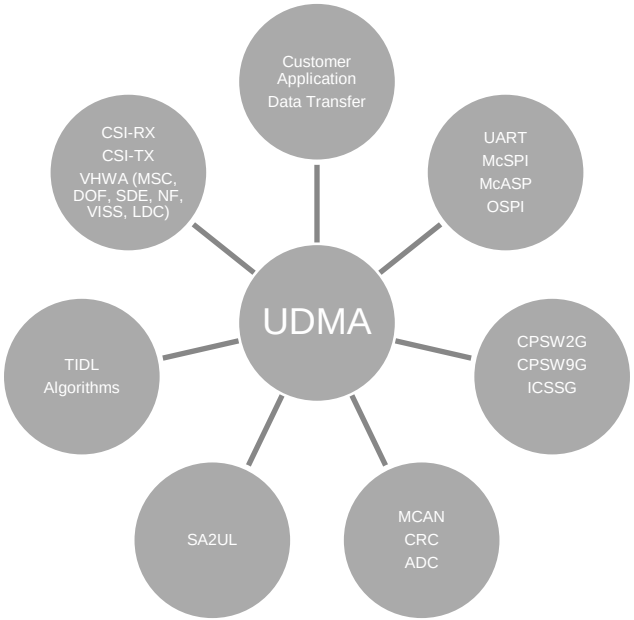
UDMA and Peripherals

Peripheral	DMA Type	DMA descriptor	Remarks
McSPI*	PDMA + UDMA-P	HOST Descriptor	
UART*		HOST Descriptor	
MCAN**		HOST Descriptor	
McASP*		TR Descriptor	Special mode in PDMA for MCAN
ADC**		TR Descriptor	
CPSWx*	Native PSI + UDMA-P	HOST Descriptor	
ICSSG*		HOST Descriptor	
SA2UL*		HOST Descriptor	
CSI2 TX / RX*		TR Descriptor	
VPAC / DMPAC*	VPAC/DMPAC UTC + UDMA-P	TR Descriptor	UDMA-P only used to transfer TR
C7x/MMA**	MSMC DRU + UDMA-P	TR Descriptor	UDMA-P only used to transfer TR
OSPI*	UDMA-P	HOST/TR Descriptor	UDMA-P used in block copy (memory to memory) mode. Peripheral acts as memory mapped region
PCIe*		HOST/TR Descriptor	
CRC**		TR Descriptor	

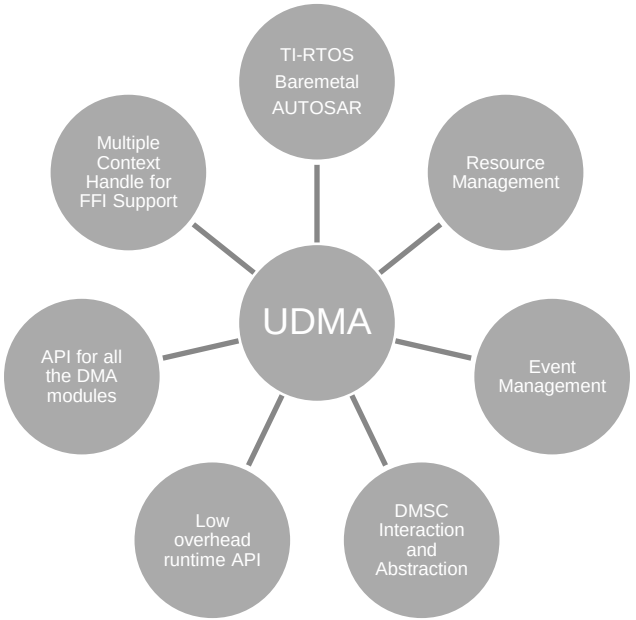
NOTE:

1. * Already supported/abstracted by TI RTOS drivers
2. ** CSL-FL based examples or DMA utils provided for DMA demonstration
3. DSS, GPU, D55xx encode/decode, VPE, USB, MMCSD, FlexRay, MLB, UFS have embedded DMA, i.e no UDMA/NavSS interaction

UDMA Driver: Dependencies and Features



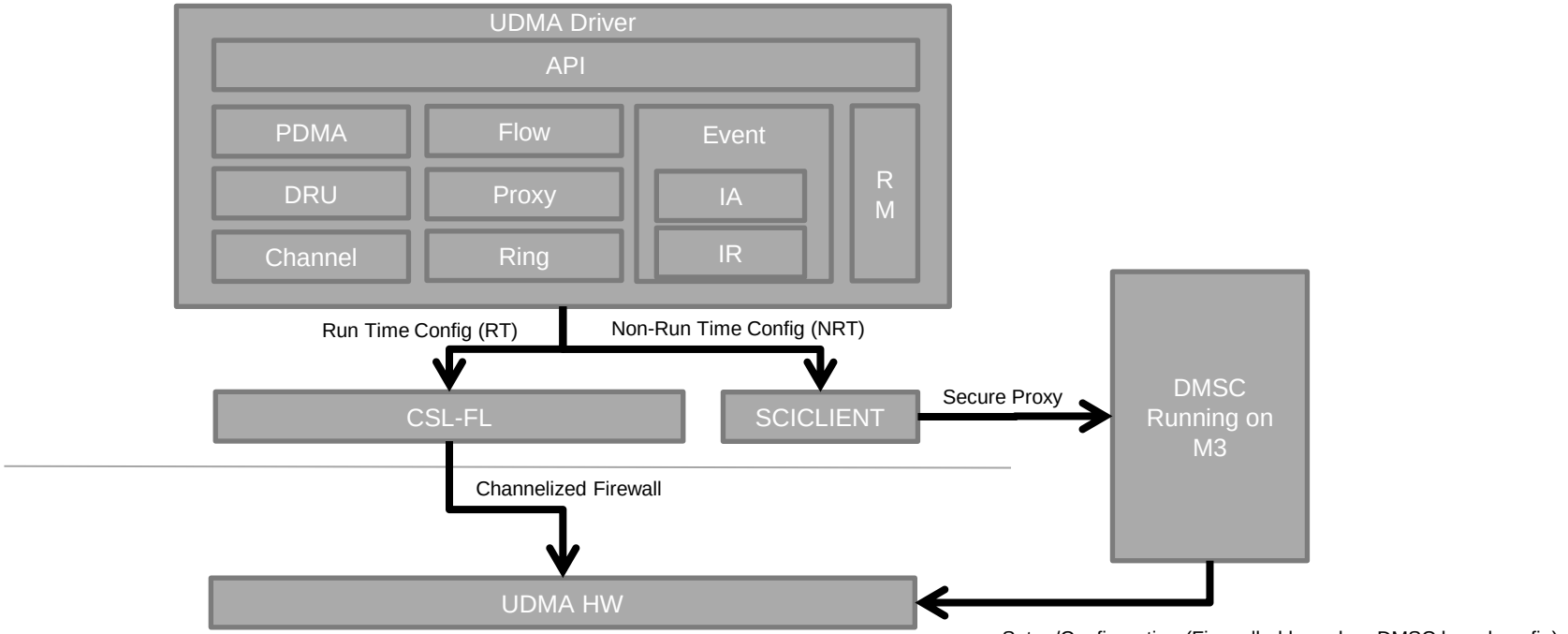
Dependent SW Modules



Features

Texas Instruments

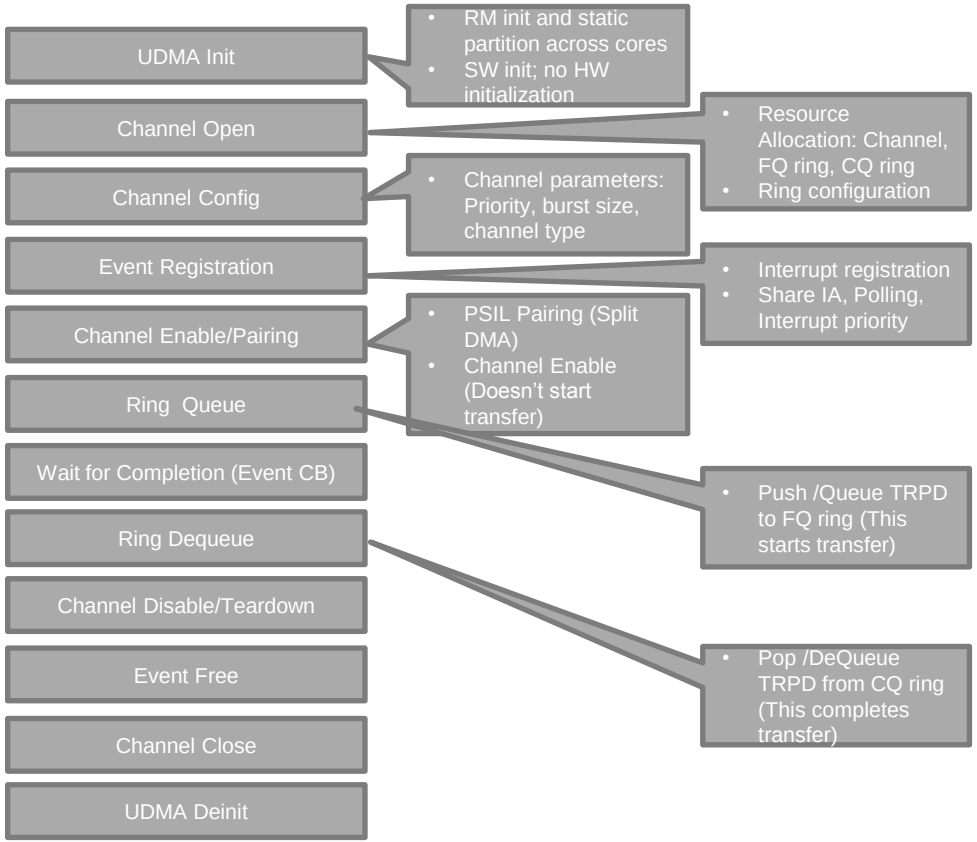
UDMA SW Architecture



13

Texas Instruments

UDMA LLD API Flow




```
UdmaInitPrms_init(instId, &initPrms);
Udma_init(drvHandle, &initPrms);

UdmaChPrms_init(&chPrms, chType);
Udma_chOpen(drvHandle, chHandle, chType, &chPrms);

UdmaChTxPrms_init(&txPrms, chType);
Udma_chConfigTx(chHandle, &txPrms);

UdmaEventPrms_init(&eventPrms);
Udma_eventRegister(drvHandle, eventHandle, &eventPrms);

Udma_chEnable(chHandle);

Udma_ringQueueRaw(Udma_chGetFqRingHandle(chHandle), (uint64_t) trpdMem);
/* Wait for event */
Udma_ringDequeueRaw(Udma_chGetCqRingHandle(chHandle), &pDesc);

Udma_chDisable(chHandle, UDMA_DEFAULT_CH_DISABLE_TIMEOUT);
Udma_eventUnRegister(eventHandle);
Udma_chClose(chHandle);

Udma_deinit(drvHandle);
```


14

© Copyright 2019 Texas Instruments Incorporated. All rights reserved.

This material is provided strictly “as-is,” for informational purposes only, and without any warranty.
Use of this material is subject to TI’s **Terms of Use,** viewable at TI.com

Back-up (Advance users)

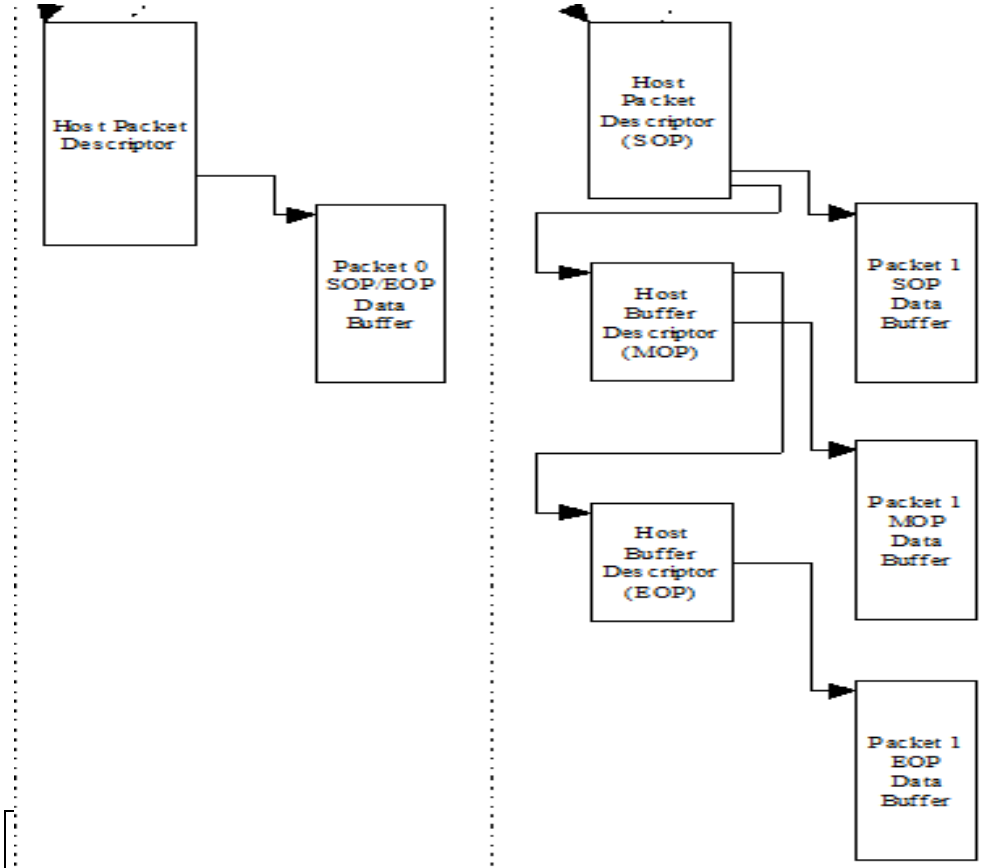
Texas Instruments

UDMA Setup

- **Describe the DMA transfer**
  - **Using Descriptors – HOST descriptors, TR descriptors**
- Submit the DMA transfer
  - Using Proxy, RING Accelerator
- Execute the DMA transfer
  - Using UDMA-P, PDMA, PSI
- Wait for DMA transfer completion
  - Using Events, Interrupt Aggregator, Interrupt Router

Texas Instruments

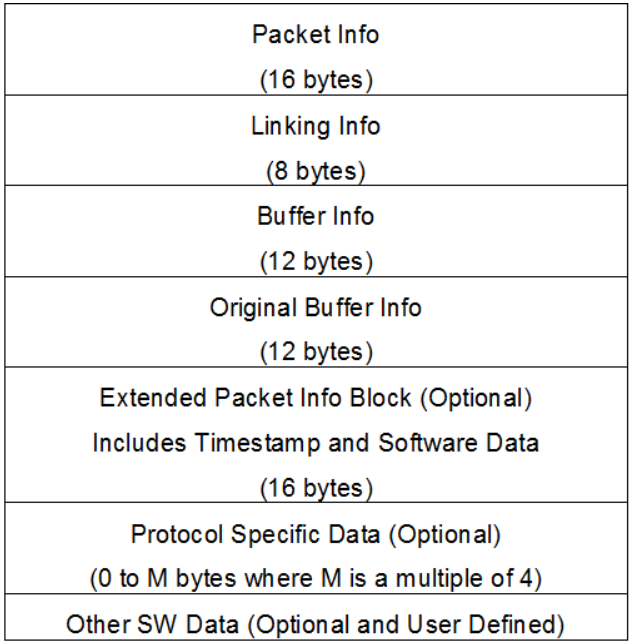
HOST Descriptor (Used in peripherals)



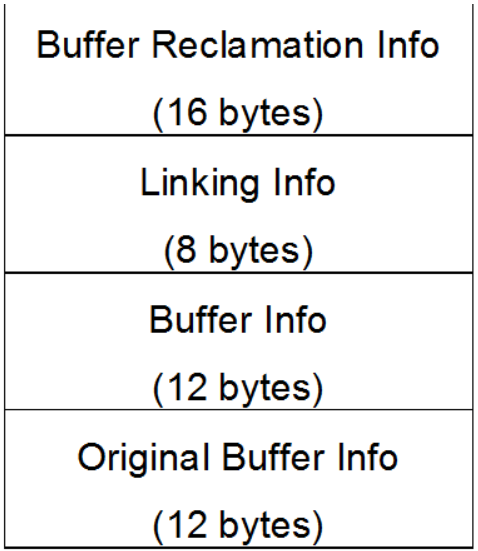
- Useful to describe “packet” like data structures
  - Ex, networking packets
- Useful to describe scatter gather data structures

Texas Instruments

HOST Descriptor



HOST Packet Descriptor



HOST Buffer Descriptor

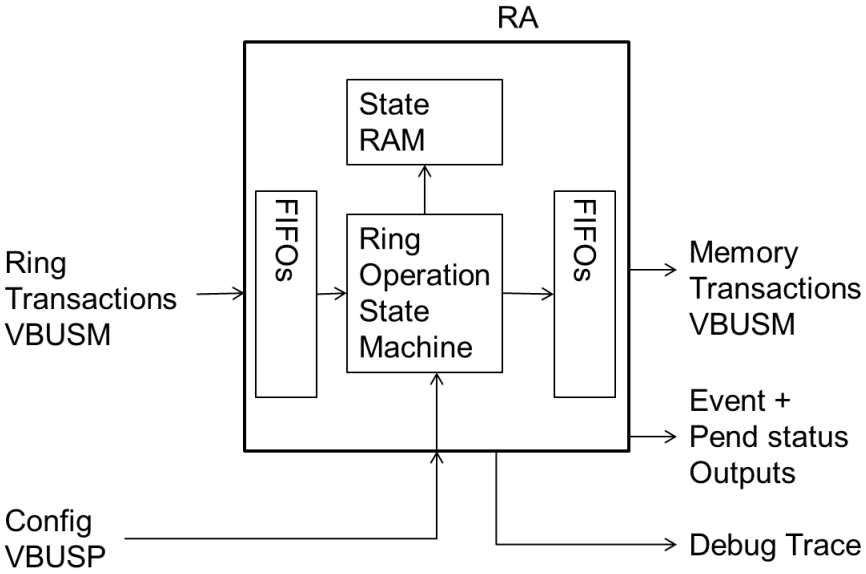
Texas Instruments

To use UDMA

- Describe the DMA transfer
  - Using Descriptors – HOST descriptors, TR descriptors
- **Submit the DMA transfer**
  - **Using Proxy, RING Accelerator**
- Execute the DMA transfer
  - Using UDMA-P, PDMA, PSI
- Wait for DMA transfer completion
  - Using Events, Interrupt Aggregator, Interrupt Router

Texas Instruments

RING Accelerator

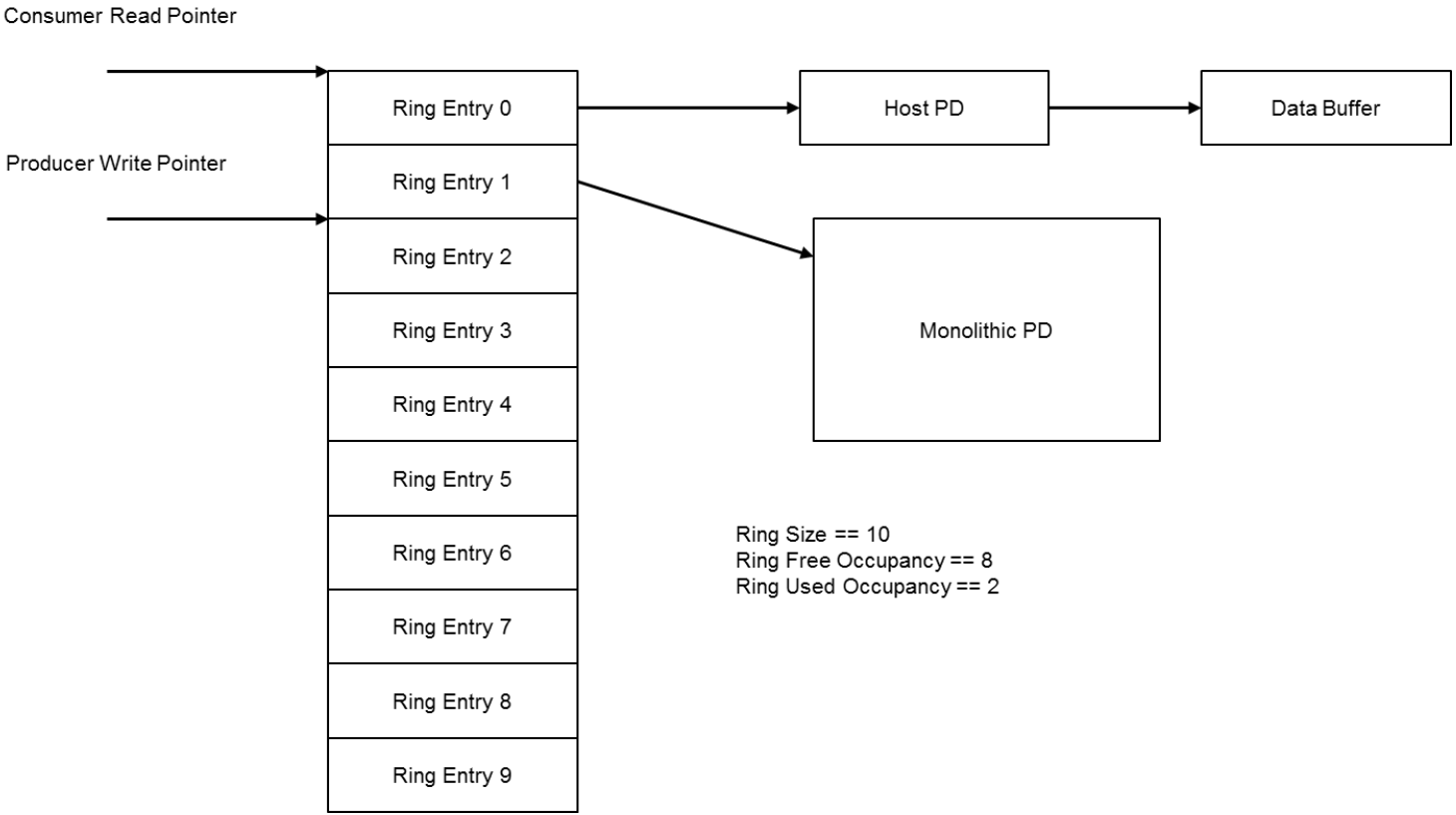


- The Ring Accelerator implements a HW Queue
  - Used to submit descriptors to UDMA-P
  - Used to exchange arbitrary messages between SW entities (i.e IPC)
- N independent RINGs are provided
  - Ex, 1024 RINGs in J721E (Varies across SOC)
- Each queue is implemented as a circular buffer in memory which is external to the RA
  - Any size ring from 1 entry to 1M-1 entries is supported
  - Each element on the ring can be up to 256 bytes
  - Contents of a ring element are as follows:
    - The message data (can be an actual data value set or a reference to a set of data)
    - Optional credentials (priv, privid,secure,virtid)
    - Optional message length

NOTE: Bottom “N” (equal to channel count) RING instance to UDMA-P channel association is fixed

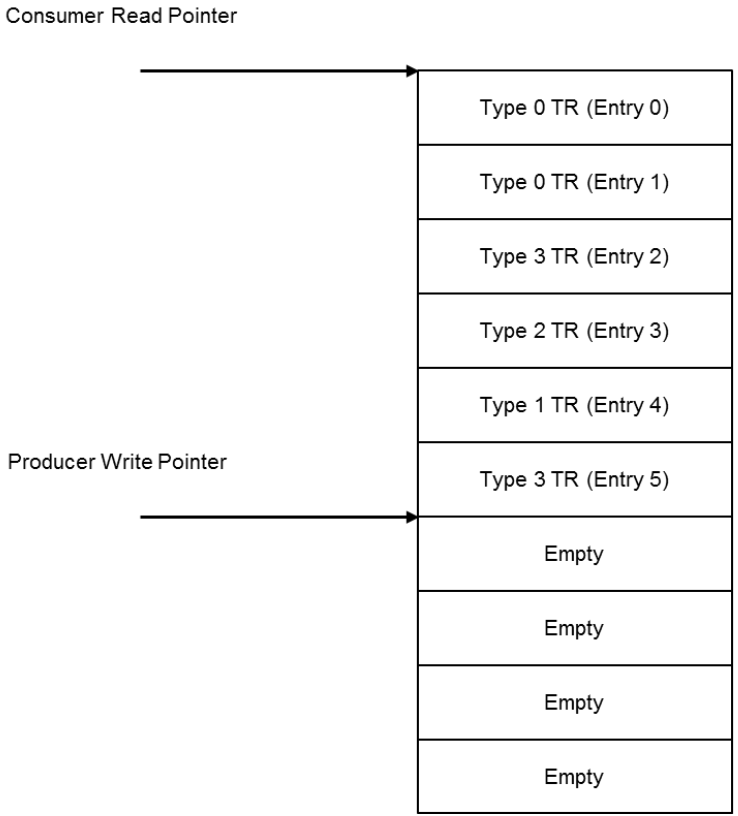
Texas Instruments

Pass By Reference RING mode



Texas Instruments

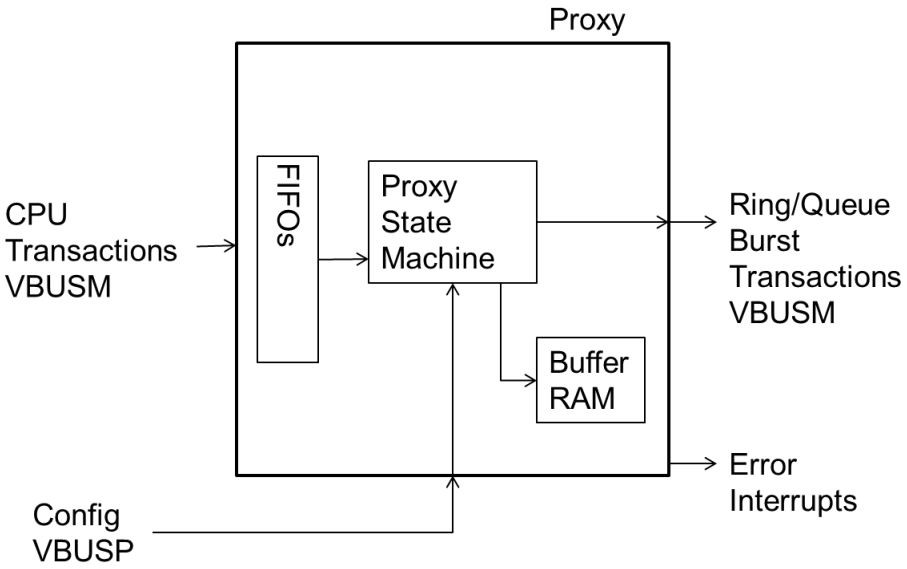
Pass By Value RING mode



Ring Size == 10
Element Size = 32 bytes
Ring Free Occupancy == 4
Ring Used Occupancy == 6

Texas Instruments

PROXY



- Used to indirectly access a RING.
- Allow use-case of multiple producer – single consumer
- Typically one proxy used per CPU or per VM
- Ex, 64 proxies in J721E

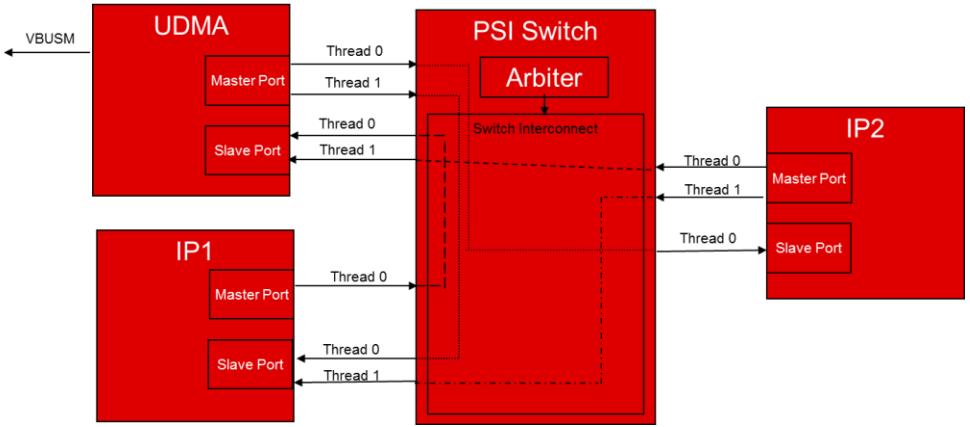
- The Proxy receives small CPU transactions to write or read parts of messages (as CPU transactions cannot handle a single 64 byte message), while accessing the RA in atomic message bursts to simplify queuing
- For writes the Proxy stores the partial message as it is built in the Buffer RAM
  - When the message is completed (by writing the final message byte) the Proxy sends the full burst to the RA
- For reads the Proxy reads the full message from the RA (upon CPU initial read) into the Buffer RAM
  - Then each CPU read will read from the buffer RAM message until completed (by reading the final message byte)
- The Proxy IP is built to support N threads of execution and M size messages
  - Each thread maps to a CPU thread that can independently access a message
  - Each thread can store up to a single max sized message, M bytes, in the Buffer RAM
  - Each thread has a 64KB (MMU aligned) memory range to access its window, with the offset determining which queue to access
- The Proxy IP will detect errors when thread access changes rings/queues while in the middle of a message
  - Once the proxy is in error, it will no longer send or receive messages until the error status is cleared

To use UDMA

- Describe the DMA transfer
  - Using Descriptors – HOST descriptors, TR descriptors
- Submit the DMA transfer
  - Using Proxy, RING Accelerator
- **Execute the DMA transfer**
  - **Using UDMA-P, PDMA, PSI**
- Wait for DMA transfer completion
  - Using Events, Interrupt Aggregator, Interrupt Router

Texas Instruments

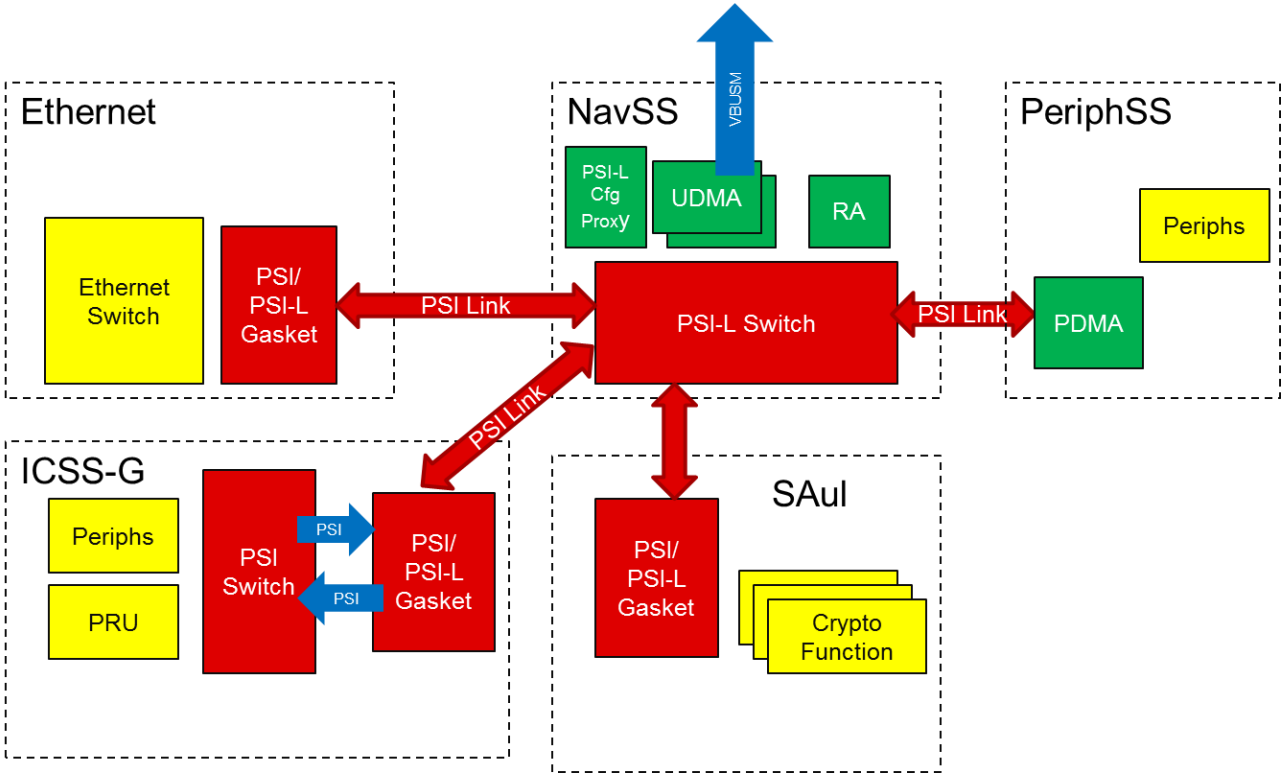
PSI



- PSI is a push-based interface for routing packets of data between peripherals
  - Packets of data can be any generic packet of work
- Features
  - Multi-threaded
  - Point-to-Point
  - Non-blocking
  - Highly Efficient
    - Single cycle arbitration
    - Data transfers every cycle
  - Easy to pipeline with no throughput penalty

Texas Instruments

UDMA – PSI – PDMA System in K3



Texas Instruments

To use UDMA

- Describe the DMA transfer
  - Using Descriptors – HOST descriptors, TR descriptors
- Submit the DMA transfer
  - Using Proxy, RING Accelerator
- Execute the DMA transfer
  - Using UDMA-P, PDMA, PSI
- **Wait for DMA transfer completion**
  - **Using Events, Interrupt Aggregator, Interrupt Router**

Texas Instruments

Events and Interrupts

- Events provide information to indicate that a condition has asserted or de-asserted
  - Ex, TR complete event or TR ICNT1 complete event or RING not empty event or peripheral FIFO threshold reached
- Events can be of two types
  - Local events
    - Each event is signaled on a dedicated pin in the PDMA or UTC
    - Local events are typically used between peripherals and PDMA or UTC
  - Global events
    - Signaled as message on Event Transport Lanes (ETL)
- **Events in are not the same as interrupts !!!**
  - **Interrupt Aggregator (IA) used to convert event to interrupt for SW notification**

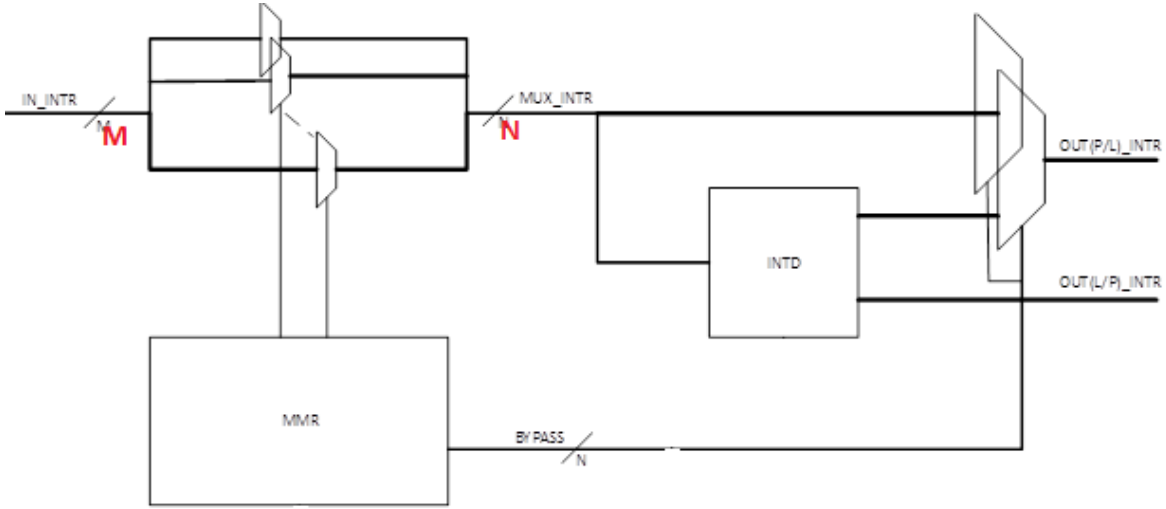
Global Events

- Global Event represented as 16b index, i.e there can be 64K distinct events
  - The index value is generated using a direct lookup table which is controlled by SW.
  - When an IP block needs to generate a global event it provides the ability to set the index value
  - Once set, the index becomes the destination address to which the event is to be sent
  - All event sinks in the system are mapped into a unified event map which contains up to 64K different destination slots.
- Example,
  - DMA channel chaining can be implemented using events
    - In TR.FLAGS.EVENT_SIZE set the condition for generating an event, TR complete, ICNT0/1/2 complete
    - In UDMAP.TXCHAN[a].TOES = CHAINED_CHANNEL_B_EVENT_ID
      - Specifies the event to generate (or indirectly channel to trigger) when event is generated

Interrupt Aggregator (IA) - Global Event to Interrupt Mapping

- For each received event, the IA performs a lookup into an interrupt mapping table which specifies:
  - Interrupt Status Register Number (regnum)
  - Interrupt Status Bit Number (bitnum)
- For each up event which is received, the IA will set bit ‘bitnum’ in ISR ‘regnum’
- For each down event which is received, the IA will clear bit ‘bitnum’ in ISR ‘regnum’
- Provides an active high level sensitive **virtual interrupt line** for each ISR which is asserted anytime any enabled bit is set in the ISR

Interrupt Router (IR)

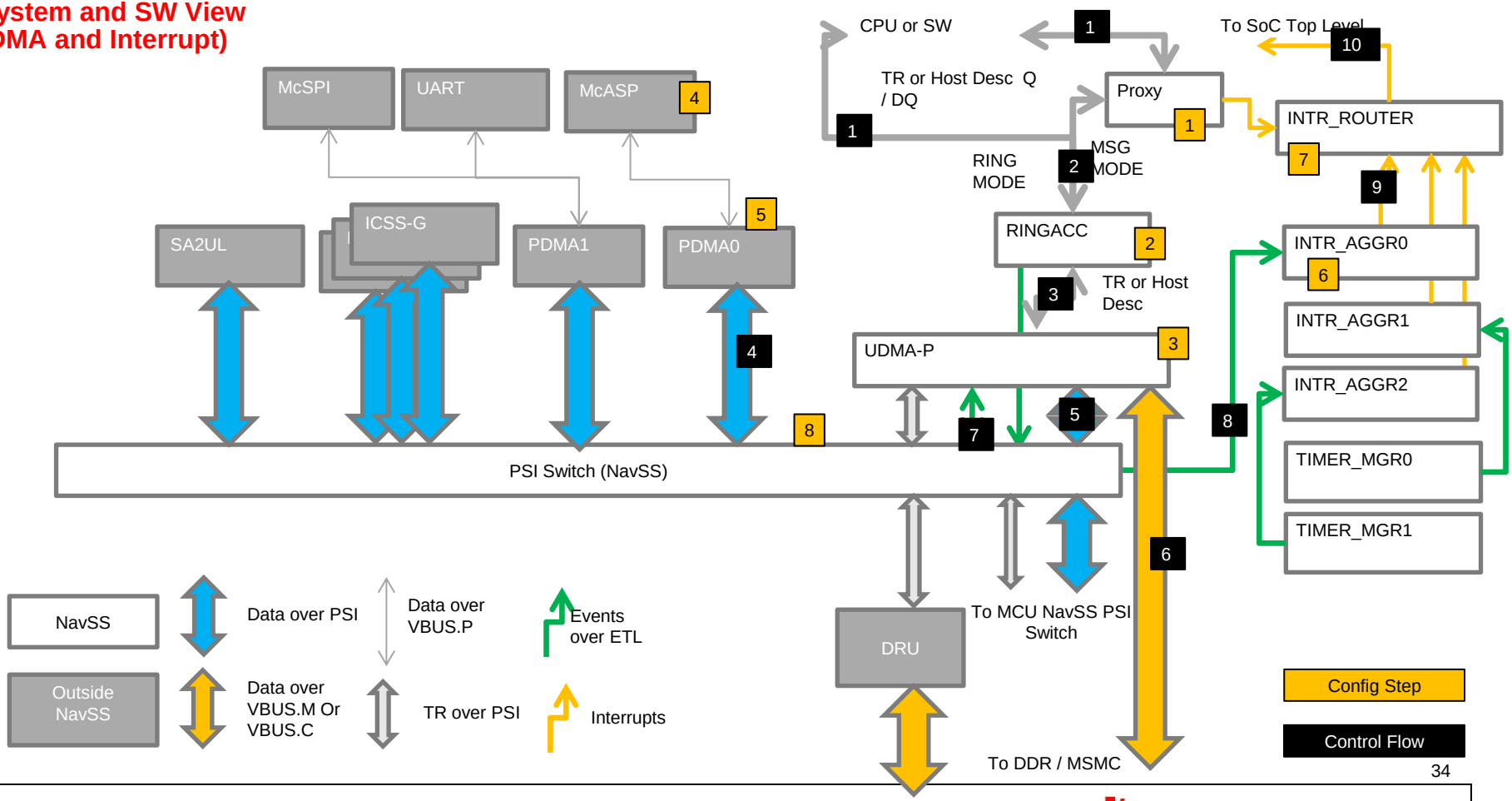


- Interrupt Router is a M to N mux
- Virtual Interrupt from IA + interrupts for other sources within NavSS example MailBox, feed as input to Interrupt router
- N interrupts from IR appear at NavSS boundary and feed GIC (A72) or CLEC (C7x) or VIM (R5F)

Putting it all together …

Texas Instruments

System and SW View (DMA and Interrupt)



Texas Instruments