KeyStone Training

Multicore Navigator
Overview

Overview Agenda

- **What is Navigator?**
  - Definition
  - Architecture
  - Queue Manager Sub-System (QMSS)
  - Packet DMA (PKTDMA)
  - Descriptors and Queuing
- **What can Navigator do?**
  - Data movement
  - InterProcessor Communication
  - Job management

What is Navigator?

- **What is Navigator?**
  - Definition
  - Architecture
  - Queue Manager Sub-System (QMSS)
  - Packet DMA (PKTDMA)
  - Descriptors and Queuing
- What can Navigator do?
  - Data movement
 - InterProcessor Communication
  - Job management

Definition

- Multicore Navigator is a collection of hardware components that facilitate data movement and multi-core control.
- The major components within the Navigator domain are:
  - A hardware Queue Manager (QM).
  - Specialized packet DMAs, called PKTDMA.
  - Data structures to describe packets, called *descriptors*.
  - A consistent API to manipulate descriptors and hardware.
- Navigator is the primary data movement engine in Nyquist, Turbo Nyquist and Shannon devices.
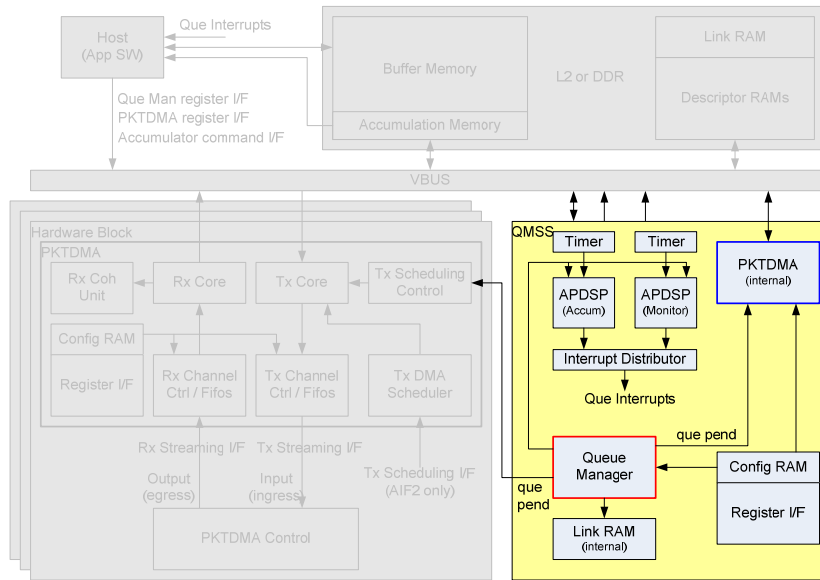- Designed to be a "fire and forget" system – load the data and the system handles the rest, without CPU intervention.
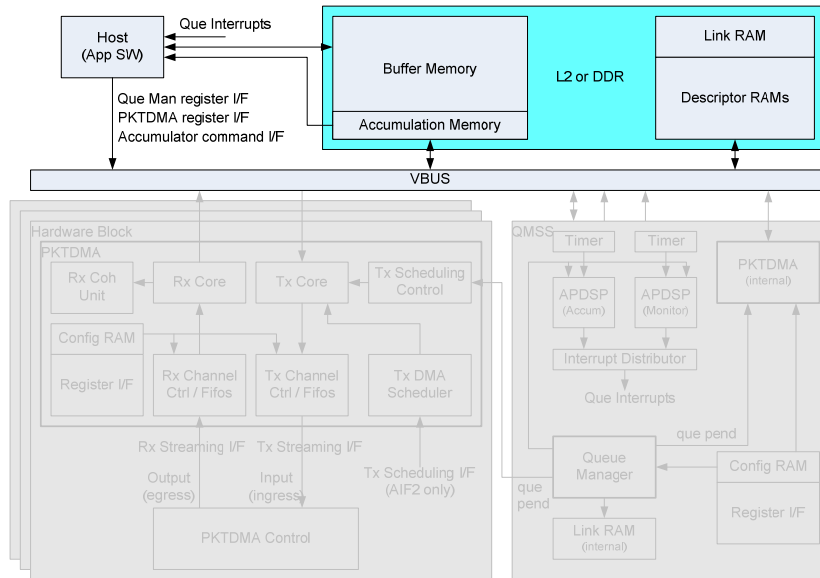
Navigator Architecture
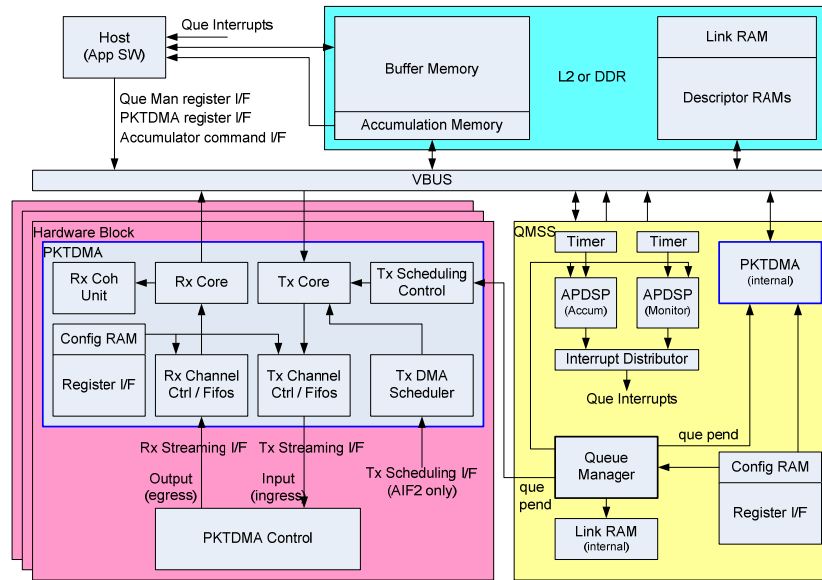
Navigator Architecture: PKTDMA
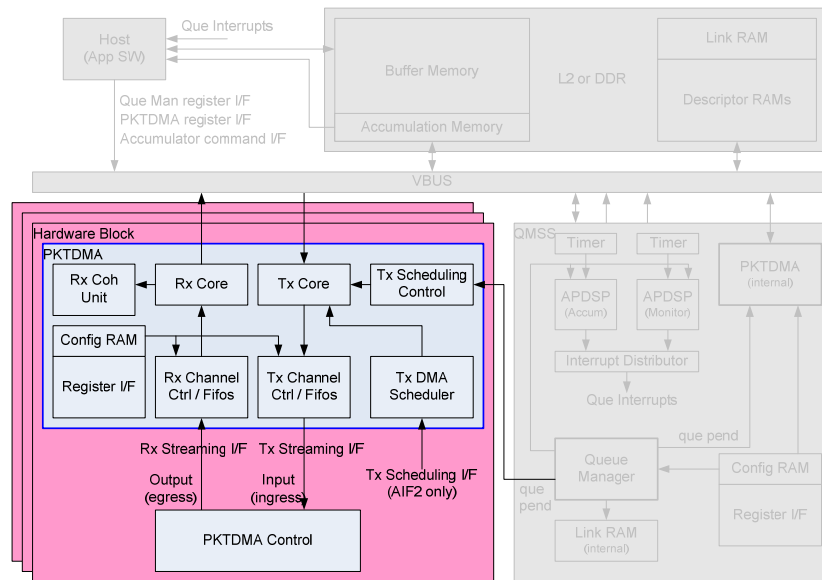
Navigator Architecture: QMSS

Navigator Architecture: Host
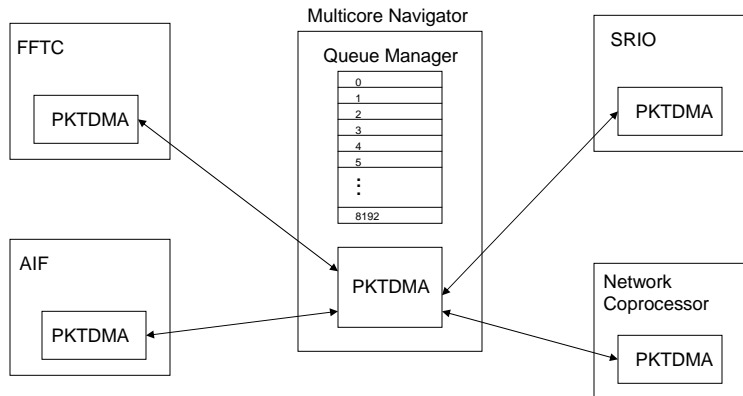
Queue Manager Subsystem (QMSS)

- Features:
  - 8192 total hardware queues, some dedicated to qpend signals.
    - HW signals route to Tx DMA channels and chip level CPINTC.
  - 20 Memory regions for descriptor storage (LL2, MSMC, DDR)
  - 2 Linking RAMs for queue linking/management
    - Up to 16K descriptors can be handled by internal Link RAM.
    - Second Link RAM can be placed in L2 or DDR.
  - Up to 512K descriptors supported in total.
- Major hardware components:
  - Queue Manager
  - PKTDMA (Infrastructure DMA)
  - 2 PDSPs (Packed Data Structure Processors) for:
    - Descriptor Accumulation / Queue Monitoring
    - Load Balancing and Traffic Shaping
  - Interrupt Distributor (INTD) module

Queue Mapping

- This table shows the mapping of queue number to functionality.
- Queues associated with queue pend signals should not be used for general use, such as free descriptor queues (FDQs). Others can be used for any purpose.

Queue Range	Count	Hardware Type	Purpose
0 to 511	512	pdsp/firmware	Low Priority Accumulation queues
512 to 639	128	queue pend	AIF2 Tx queues
640 to 651	12	queue pend	PA Tx queues (PA PKTDMA uses the first 9 only)
652 to 671	20	queue pend	CPintC0/intC1 auto-notification queues
672 to 687	16	queue pend	SRIO Tx queues
688 to 695	8	queue pend	FFTC_A and FFTC_B Tx queues (688..691 for FFTC_A)
696 to 703	8		General purpose
704 to 735	32	pdsp/firmware	High Priority Accumulation queues
736 to 799	64		Starvation counter queues
800 to 831	32	queue pend	QMSS Tx queues
832 to 863	32		Queues for traffic shaping (supported by specific firmware)
864 to 895	32	queue pend	vUSR queues for external chip connections
896 to 8191	7296		General Purpose

Packet DMA Topology

Multicore Navigator

FFTC

PKTDMA

Queue Manager

	0
	1
	2
	3
	4
	5
	⋮
	8192

SRIO

PKTDMA

AIF

PKTDMA

PKTDMA

Network Coprocessor
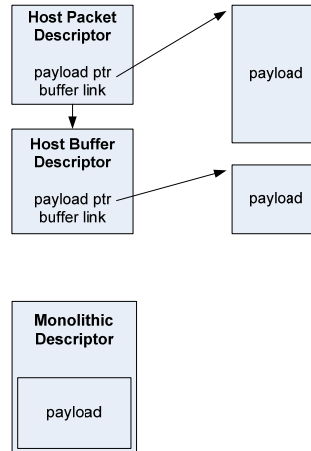
PKTDMA

Multiple Packet DMA instances in KeyStone devices:

• PA and SRIO instances for all KeyStone devices.

• AIF2 and FFTC (A and B) instances are only in KeyStone devices for wireless applications.

TEXAS INSTRUMENTS   CI Training

Packet DMA (PKTDMA) Features

- Independent Rx and Tx cores:
  - Tx Core:
    - Tx channel triggering via hardware qpend signals from QM.
    - Tx core control is programmed via descriptors.
    - 4 level priority (round robin) Tx Scheduler
      - Additional Tx Scheduler Interface for AIF2 (wireless applications only)
  - Rx Core:
    - Rx channel triggering via Rx Streaming I/F.
    - Rx core control is programmed via an "Rx Flow" (more later)
- 2x128 bit symmetrical Streaming I/F for Tx output and Rx input
  - These are wired together for loopback in QMSS' PKTDMA.
  - Connects to peripheral's matching streaming I/F (Tx->Rx, Rx->Tx)
- Packet based, so neither Rx or Tx cores care about payload format.

TEXAS INSTRUMENTS   CI Training

Descriptor Types

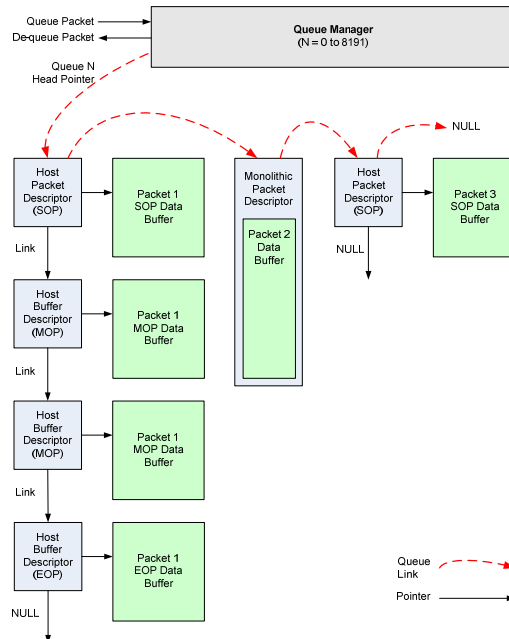- Two descriptor types are used within Navigator:
  - Host type provide flexibility, but are more difficult to use
    - Contains a header with a pointer to the payload.
    - Can be linked together (packet length is the sum of payload (buffer) sizes).

  - Monolithic type are less flexible, but easier to use
    - Descriptor contains the header and payload.
    - Cannot be linked together.
    - All payload buffers are equally sized (per region).

Host Packet Descriptor

payload ptr
buffer link

payload

Host Buffer Descriptor

payload ptr
buffer link

payload

Monolithic Descriptor

payload

Descriptor Queuing

This diagram shows several descriptors queued together. Things to note:

•Only the Host Packet is queued in a linked Host descriptor.

•A Host Packet is always used at SOP, followed by zero or more Host Buffer types.

•Multiple descriptor types may be queued together, though not commonly done in practice.

Queue Packet
De-queue Packet

Queue Manager
(N = 0 to 8191)

Queue N
Head Pointer

NULL

Host Packet Descriptor (SOP)

Packet 1 SOP Data Buffer

Link

Monolithic Packet Descriptor

Packet 2 Data Buffer

Host Packet Descriptor (SOP)

NULL

Packet 3 SOP Data Buffer

Host Buffer Descriptor (MOP)

Packet 1 MOP Data Buffer

Link

Host Buffer Descriptor (MOP)

Packet 1 MOP Data Buffer

Link

Host Buffer Descriptor (EOP)

Packet 1 EOP Data Buffer

NULL

Queue Link

Pointer

What is Navigator?

- What is Navigator?
  - Definition
  - Architecture
  - Queue Manager Sub-System (QMSS)
  - Packet DMA (PKTDMA)
  - Descriptors and Queuing
- **What Can Navigator Do?**
  - Data Movement
  - InterProcessor Communication
  - Job Management

Navigator Functionality

- Three major areas:
  - Data Movement
    - Peripheral input and output
    - Infrastructure, or core-to-core transfers
    - Chaining of transfers (output of PKTDMA A triggers PKTDMA B)
  - Inter-Processor Communication (IPC)
    - Task/Core synchronization
    - Task/Core notification
  - Job Management
    - Resource Sharing
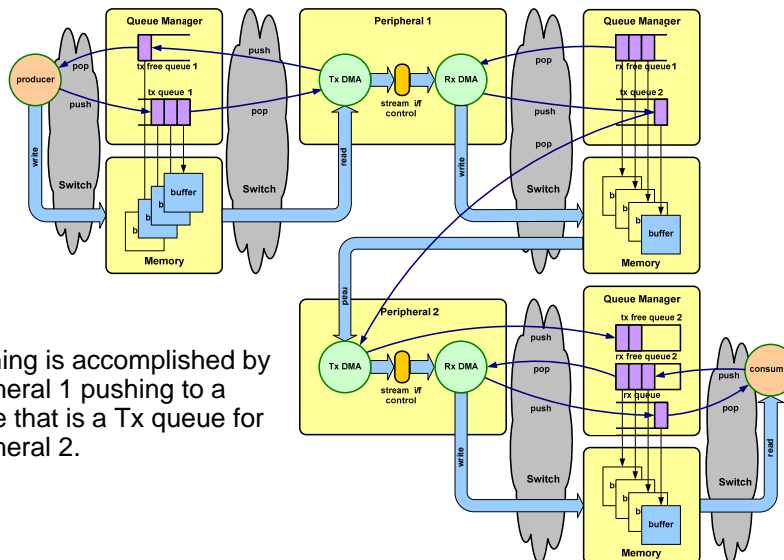   - Load Balancing

Data Movement: Normal

- Peripheral input and output:
  - Drive data through IP block using QM and PKTDMA
  - Simple transmit is shown

- Infrastructure or core-to-core transfers:
 - Transfer payload from L2 to L2, or DDR
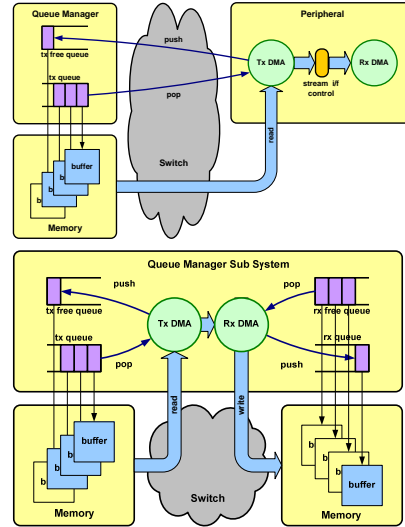
Data Movement: Chaining

- Chaining of IP transfers (output of PKTDMA 1 triggers PKTDMA 2).

- Chaining is accomplished by peripheral 1 pushing to a queue that is a Tx queue for peripheral 2.

IPC

- Using QM without a PKTDMA
- Synchronization
  - Queues are used by tasks or cores as a sync resource.
  - Multiple receivers (slaves) can sync on the same trigger.
  - Can also be done using the QMSS Interrupt Distributor, or the CpIntC queue pend queues.

- Notification
  - Zero copy messaging using shared memory
  - Producers and consumers may be on different cores or the same core.
  - Notification can be interrupt or polling.
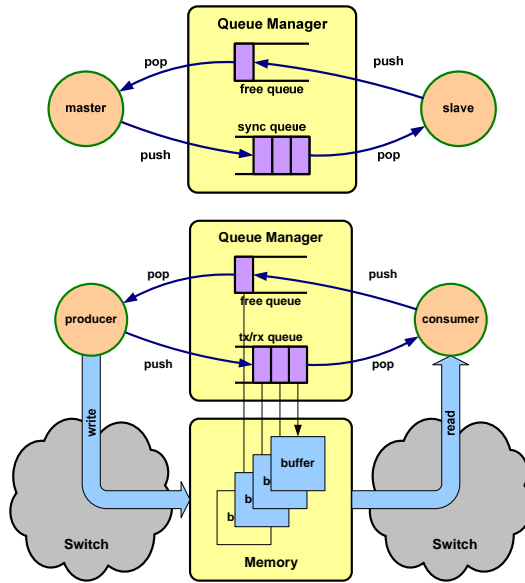
Job Management

- Two main variations:
  - Resource Sharing
    - Multiple job queues are scheduled for a single shared resource.
    - A resource can be a peripheral (e.g. FFTC) or a DSP task/core.

  - Load Balancing
    - Single job queue is fanned out to several resources.
    - The goal is to send a job to the least busy core.

  - Distributed (multi) schedulers are another variation.
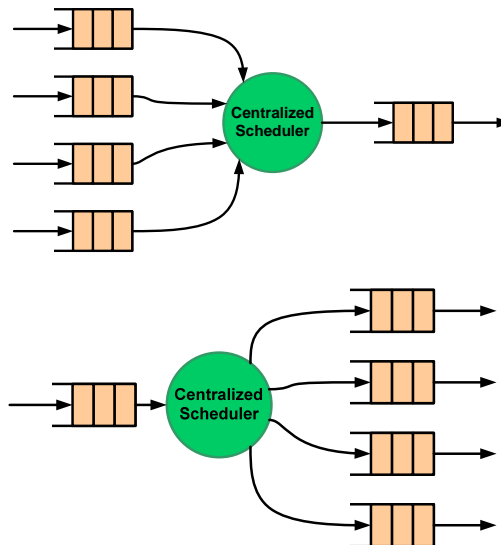
For More Information

- For more information, refer to the to Multicore Navigator User Guide http://www.ti.com/lit/SPRUGR9

- For questions regarding topics covered in this training, visit the support forums at the TI E2E Community website.