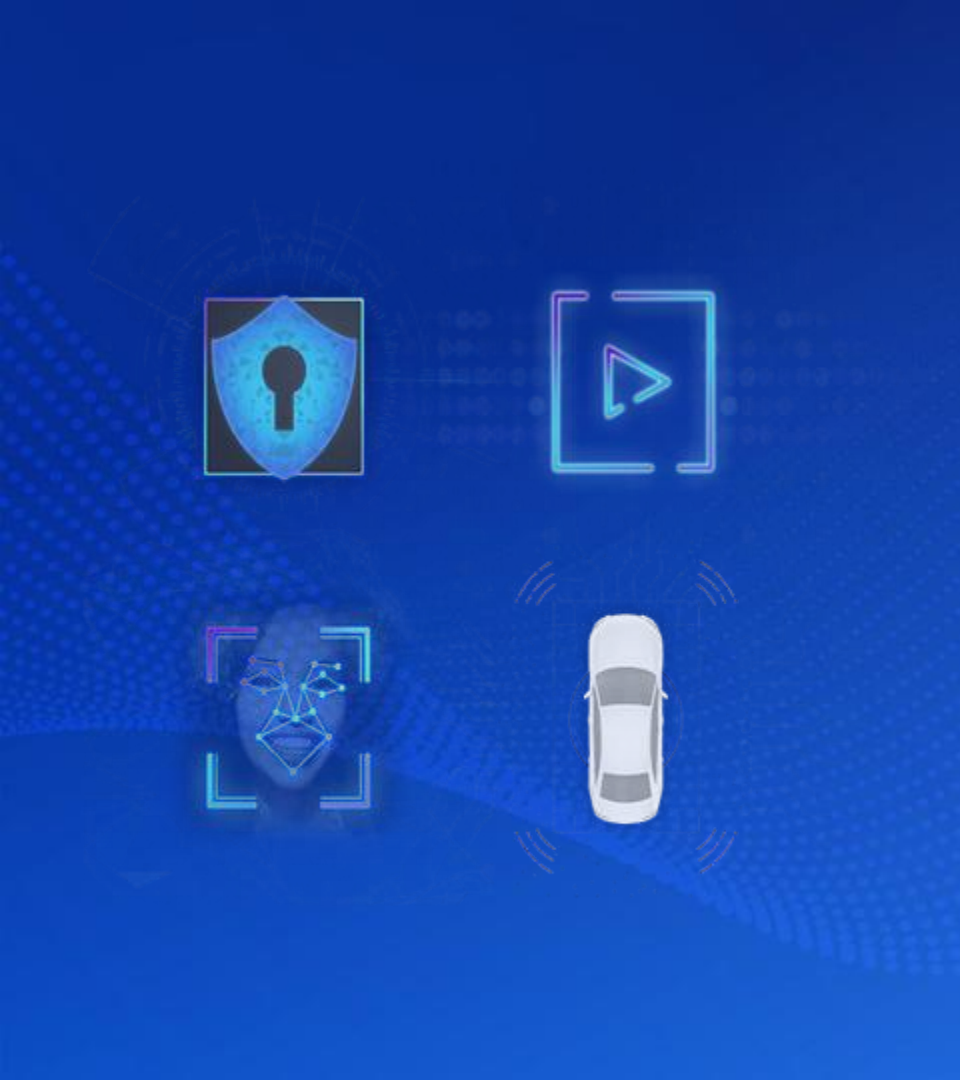


multicoreware

UDMA TUTORIAL

UDMA Overview

- UDMA stands for **Unified Direct Memory Access**
- The UDMA is intended to perform functions as the packet oriented DMA used on TI SoC devices.
- The UDMA module supports the transmission and reception of various packet types. The UDMA is architected to facilitate the segmentation and reassembly of SoC DMA data structure compliant packets to/from smaller data blocks that are natively compatible with the specific requirements of each connected peripheral.
- Multiple **Tx** and **Rx** channels are provided within the DMA which allow multiple segmentation or reassembly operations to be ongoing.
- The DMA controller maintains state information for each of the channels which allows packet segmentation and reassembly operations.

UDMA ARCHITECTURE

UDMA MODEL - DESCRIPTOR

- A UDMA DMA request is specified by user constructing a data structure in memory called “**Descriptor**”. Once a “**Descriptor**” is constructed by SW, it is submitted into a “**Ring**” to trigger the DMA operation.

- There are various types of descriptors, the most commonly used descriptors are listed below

  1. **Host Packet Descriptor** – these descriptors typically describe **1D** “packet” buffers that need to be RX’ed or TX’ed with a peripheral or used in memory to memory copy transfer (Block Copy)

  2. **Transfer Request** (**TR**) **Descriptor** – these descriptors typically describe an up to **4D** data buffer that needs to be RX’ed or TX’ed with a peripheral or used in memory to memory copy transfer (Block Copy)

UTC (Unified Transfer Controller)

- The **UTC engine** is generally classified as a third party DMA. This designation comes from the fact that the engine is not actually the source or sink of the data which is being moved but is instead an intermediary 3rd party that performs the data move on behalf of the source and sink.
- The UTC engine accepts Transfer Response messages from the UDMA via a **PSI-L** interface which provide instructions to copy data between a source read interface and a destination write interface.
- The sequence of operations that can be instructed includes up to **4-dimensional** nested loops. Multiple types of Transfer Request messages are specified and each UTC instance in the system may support all types or any specified subset.
- When a Transfer Request has been completed, the UTC returns a Transfer Response message back to the originating UDMA.

TRANSFER SYNCHRONIZATION

- The sequence of operations that can be instructed includes up to 4-dimensional nested loops that comes under transfer synchronization as follows,
    - **DIM0 synchronized transfer: 1D**
    - **DIM1 synchronized transfer: 2D**
    - **DIM2 synchronized transfer: 3D**
    - **DIM3 synchronized transfer: 4D**
- The 4-dimensional nested loops are configured based on the data arrangement which comes as part of the descriptor for memory to memory block transfer

UDMA TRANSFER CONFIGURATION

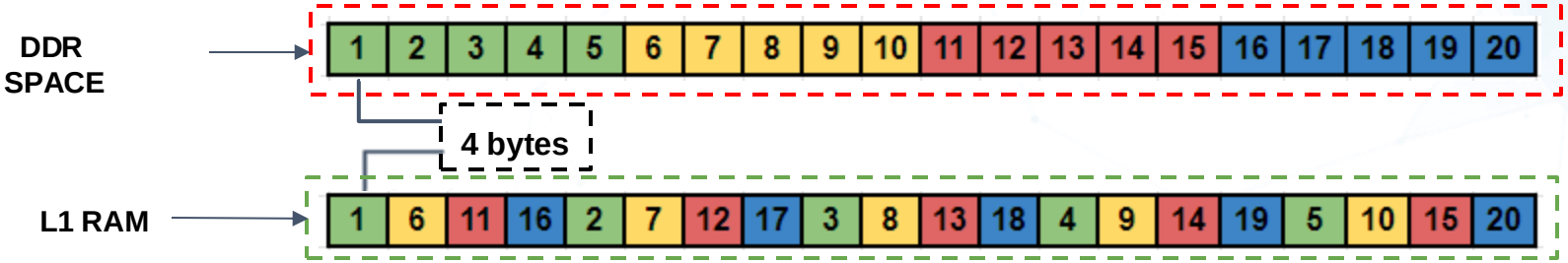
UDMA TR (4D Block Copy TR)	PURPOSE
ICNT0 (16b) , DICNT0 (16b)	1st dimension loop count
ICNT1 (16b) , DICNT1 (16b)	2nd dimension loop count
ICNT2 (16b) , DICNT2 (16b)	3rd dimension loop count
ICNT3 (16b) , DICNT3 (16b)	4th dimension loop count
DIM1(signed 32b)	2nd dimension stride at source
DDIM1 (signed 32b)	2nd dimension stride at destination
DIM2 (signed 32b)	3rd dimension stride at source

UDMA TRANSFER CONFIGURATION

UDMA TR (4D Block Copy TR)	PURPOSE
DDIM2 (signed 32b)	3rd dimension stride at destination
DIM3 (signed 32b) DDIM3 (signed 32b)	4th dimension stride at source and destination
ADDR (64b)	Source address
DAADR (64b)	Destination address

UDMA EXAMPLE

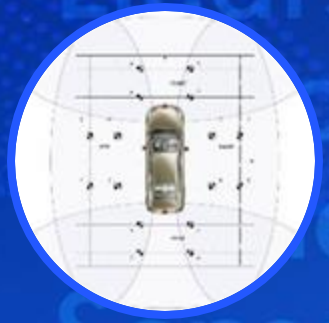
UDMA TRANSFER



ICNT0 (16b) | DICNT0 (16b)	**4 bytes | 4 bytes**
ICNT1 (16b) | DICNT1 (16b)	**4 elements per transfer | 5 blocks**
ICNT2 (16b) | DICNT2 (16b)	**5 blocks | 4 elements per transfer**
ICNT3 (16b) , DICNT3 (16b)	**0U**
DIM1(signed 32b)	**4 bytes**
DIM2(signed 32b)	**4 elements * 4bytes**

DIM3 (signed 32b)	**1U**
DDIM1 (signed 32b)	**4 elements * 4 bytes**
DDIM2 (signed 32b)	**4 bytes**
DDIM3 (signed 32b)	**1U**
ADDR (64b)	**0x00C00000**
DAADR (64b)	**0x00070000**

THANK YOU



connect with us

www.multicorewareinc.com

www.facebook.com/multicoreware

www.twitter.com/multicoreware

www.linkedin.com/company/multicoreware-inc

www.instagram.com/multicoreware

www.youtube.com/multicoreware