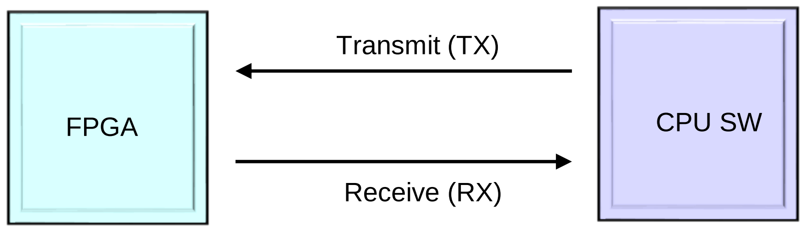
Basic design principles to design with the Smartlogic DMA IP Cores for PCIe

Thomas Zerrer

Data flow definition RX and TX



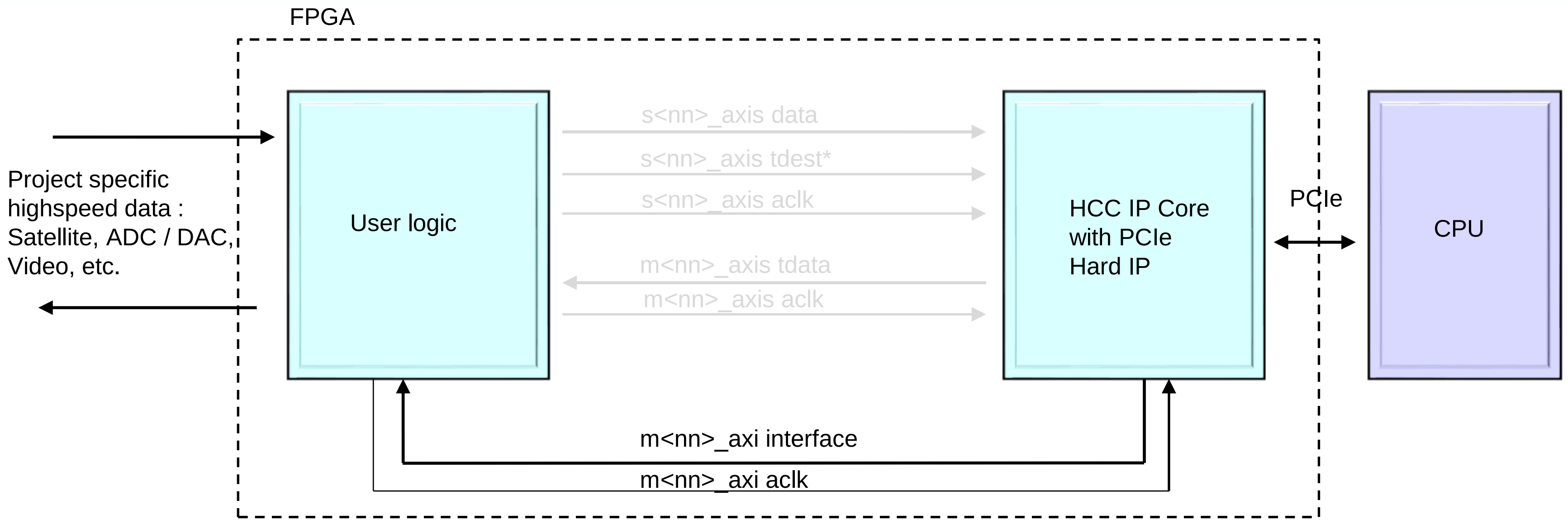
Receive (RX)

- FPGA transmits data to CPU via DMA
- AXI Stream Slaves receive data from the user logic
- Memory Write TLPs are issued by the FPGA
- User Guide defines this as „DMA Write“

Transmit (TX)

- FPGA reads data from CPU via DMA
- AXI Stream Masters present data to the user logic
- Memory Read TLPs are issued by the FPGA
- User guide defines this as „DMA Read“

Smartlogic Multichannel DMA IP core – interfacing user I/O

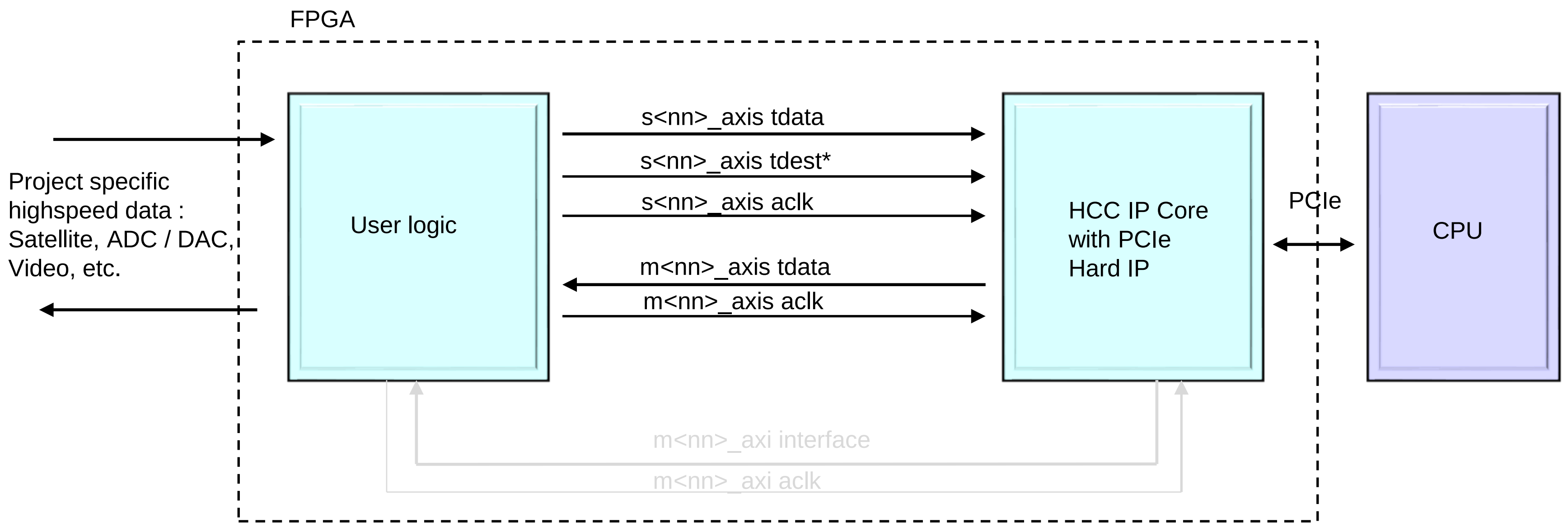


AXI (memorymapped) Master interfaces m<nn>_axi:

- used to configure user logic registers
- each interface is clocked with a dedicated (asynchronous) user clock
- datawidth configurable in powers of 2
- up to 8 AXI Masters available

* Only available for HCC Core

Smartlogic multichannel DMA IP core streaming interfaces



AXI Stream slave interfaces (s<nn>_axis) for RX:

- user logic transmits data to the RX data buffers via one of the 16 available s<nn>_axis interfaces
- each interface is clocked with a dedicated (asynchronous) clock input
- each interface contains the TDEST sideband signal to identify the destination of the transfer*
- databitwidth adjustable in powers of 2

AXI Stream master interfaces (m<nn>_axis) for TX:

- IP core reads TX data to the user logic via one of the 16 available m<nn>_axis interfaces
- each interface is clocked with a dedicated (asynchronous) clock input
- databitwidth adjustable in powers of 2

* Only available for HCC Core

Use case 1 : Fixed size data transmission

The Smartlogic PCIe DMA IP cores support fixed size and variable sized datatransmission.

In this section the fixed size data transmission is explained in detail.

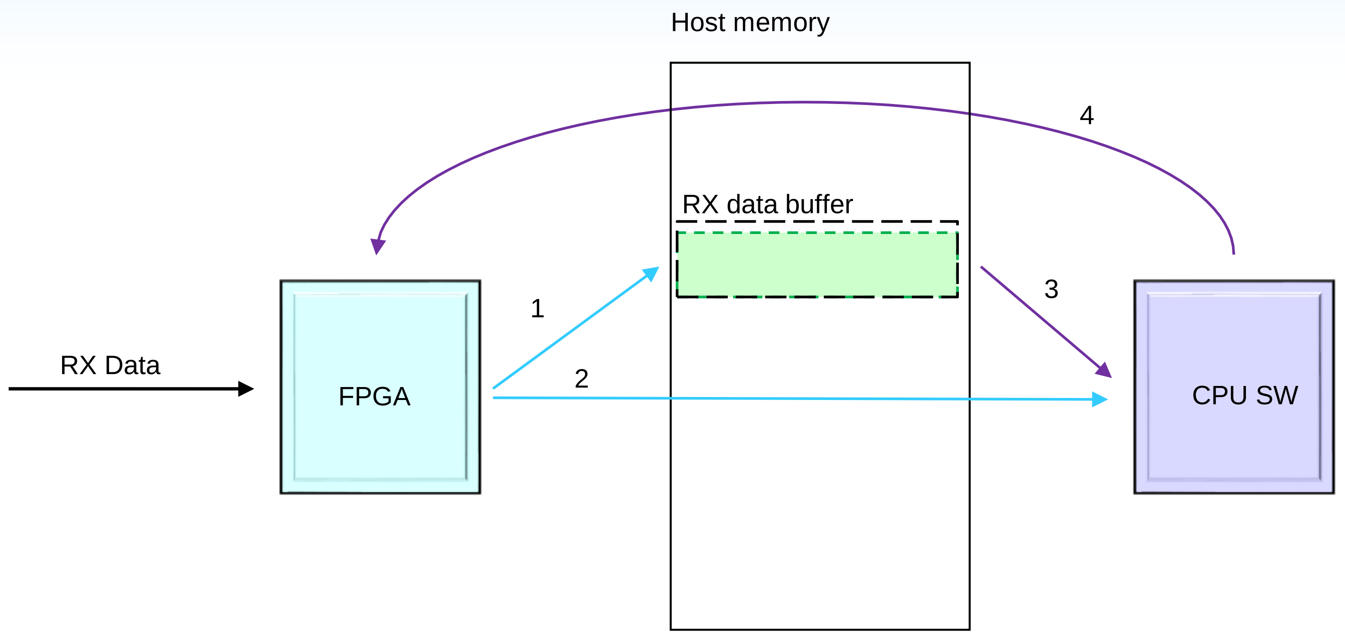
Fixed size transmission transfers blocks of data with a fixed size

Examples:

- Video Frames (downstream and or upstream)
- Supplying data for DACs (typically downstream)

The following slides show the details for this use case. The variable sized use case is discussed in the section following the fixed size section.

Fixed size : RX – FPGA writes data to the host memory



FPGA

- Step 1:
FPGA transmits valid data to the next empty RX data buffer

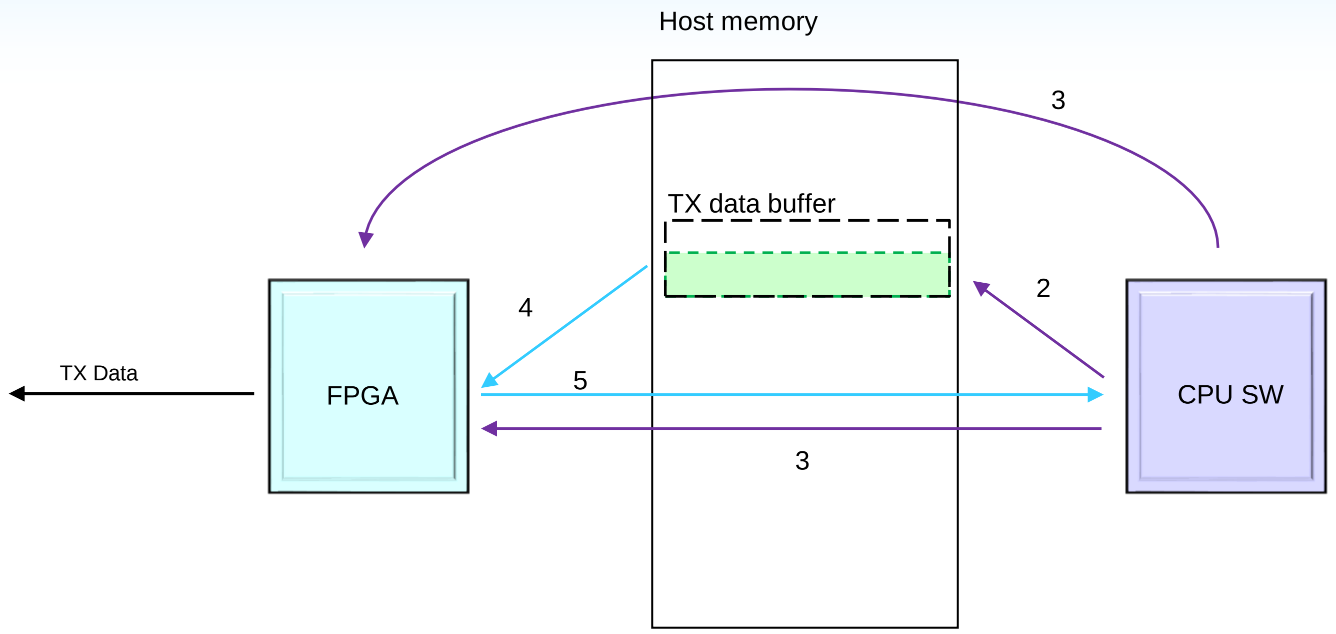
- Step 2:
FPGA issues an interrupt

Smartlogic Driver

- Step 3:
Driver reacts to the interrupt and the data buffer with fixed block length is offered to the user application

- Step 4 : Driver transfers the start address of the now empty RX buffer back to the FPGA

Fixed size : TX – FPGA reads data from the host memory



FPGA

- Step 4:
FPGA reads the data buffer
- Step 5:
FPGA issues an interrupt

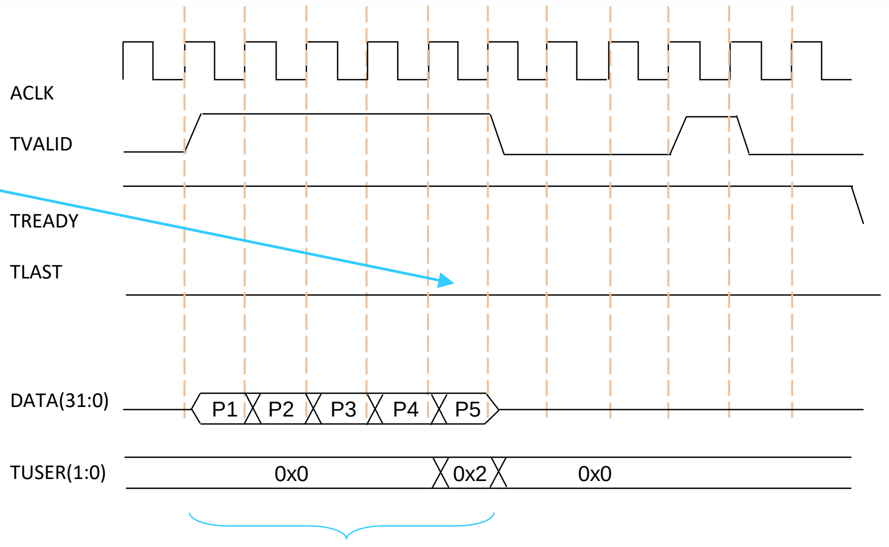
User Application / Smartlogic Driver

- Step 1:
User app checks if there is an empty TX data buffer
- Step 2:
User app fills the data buffer
- Step 3: Driver informs the FPGA to read the data buffer by writing the startaddress. Length is fixed

Fixed size : How RX data is passed to the FLEX IP core

Timing diagram for s<nn>_axis:

Data for DB is fed in first without TLAST.



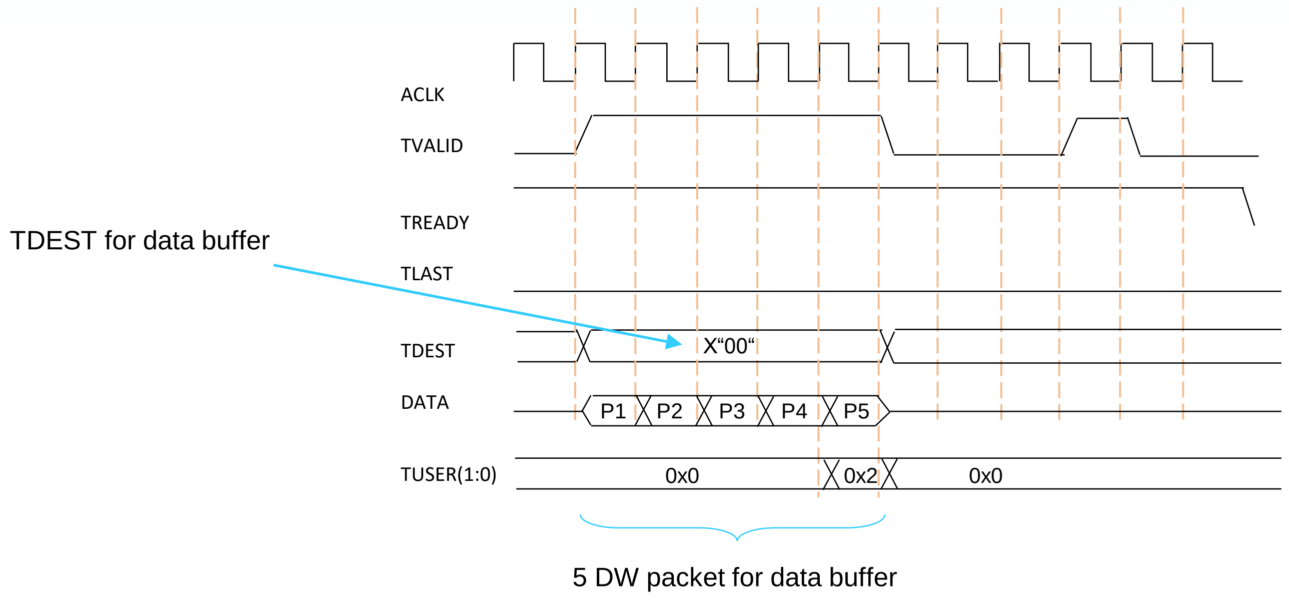
5 DW packet for data buffer with EOF and without TLAST

- Back to back transmission is possible.
- Tuser must be driven with 0x2 in the last dataphase of data
- In case of an uneven byte count for DB, the remaining bytes of the last data phase are automatically padded by the core with 0x0
- Example shows 32-Bit s_axis data bitwidth. Other data bit widths are allowed.

Fixed size : How RX data is passed to the HCC IP core

SMARTLOGIC

Timing diagram for s<nn>_axis:



- Back to back transmission is possible
- Tuser must be driven with 0x2 in the last dataphase to switch to the next DB
- TDEST must not change during a packet
- In case of an uneven byte count for DB, the remaining bytes of the last data phase are automatically padded by the core with 0x0
- Example shows 32-Bit s_axis data bitwidth. Other data bit widths (powers of 2) are allowed. See UG for details

Fixed size : DB channel mapping in sldma_test.cpp

The reference design DMA_Demo3 features up to 8 upstream channels that transmit data.

The following table shows the default mapping of the TDEST values for data :

s_axis if #	TDEST for DB #	
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	

If this mapping is kept, the sldma_test.cpp adaptions can be held at a minimum.

Use case 2 : Variable size data transmission

The Smartlogic IP cores also support variable size data transmission

This type of data transmission transfers blocks of data with a variable block length

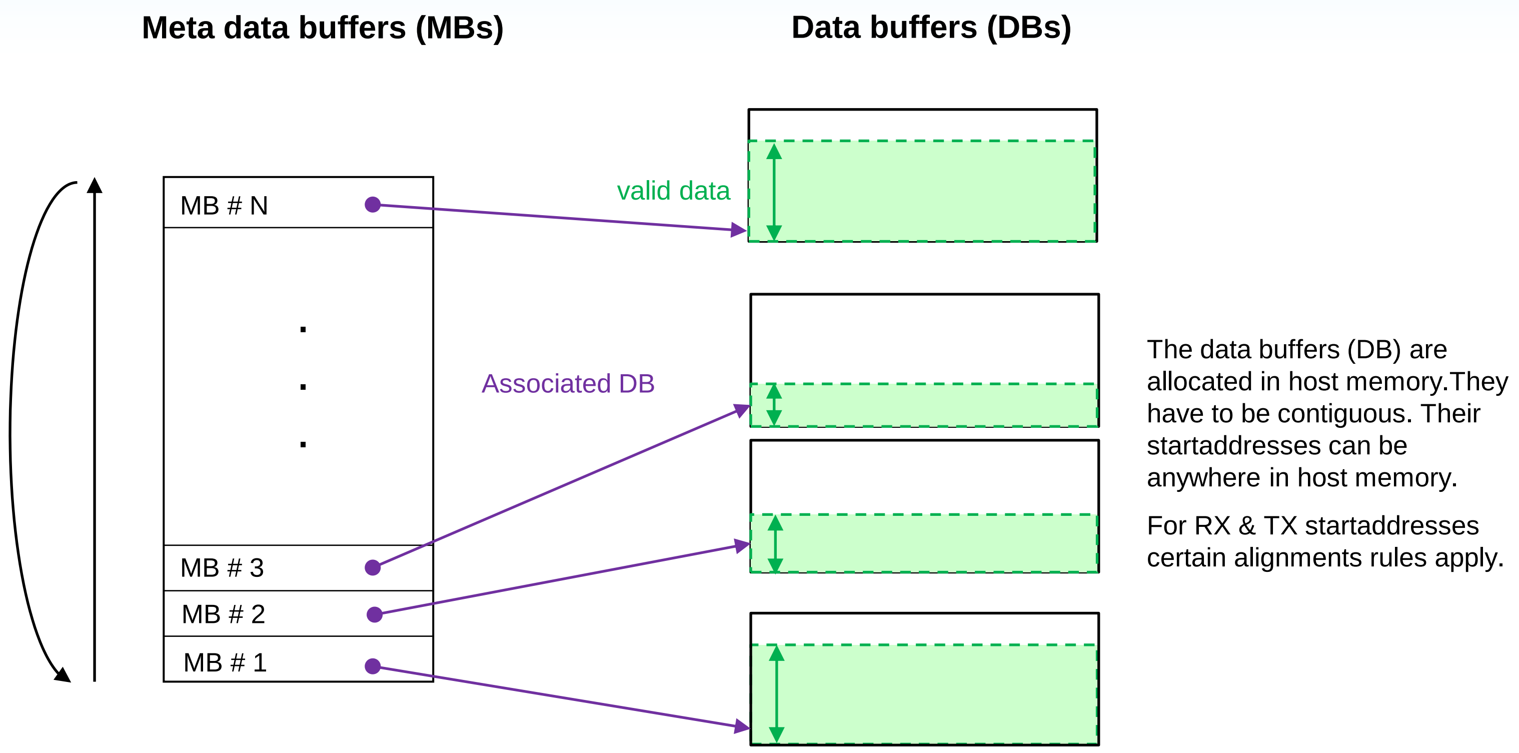
Examples:

- Ethernet frames (downstream and or upstream)

The following slides show the details for this use case.

Note : The Flex IP Core supports variable sized transfers only upstream

Variable size : Definition of terms „MB“ and „DB“



The meta data buffers (MB) reside in host memory and must be contiguous. They provide the length in the first DW and user defined side data.

The data buffers (DB) are allocated in host memory.They have to be contiguous. Their startaddresses can be anywhere in host memory.

For RX & TX startaddresses certain alignments rules apply.

First Dword in each MB must have the following format:

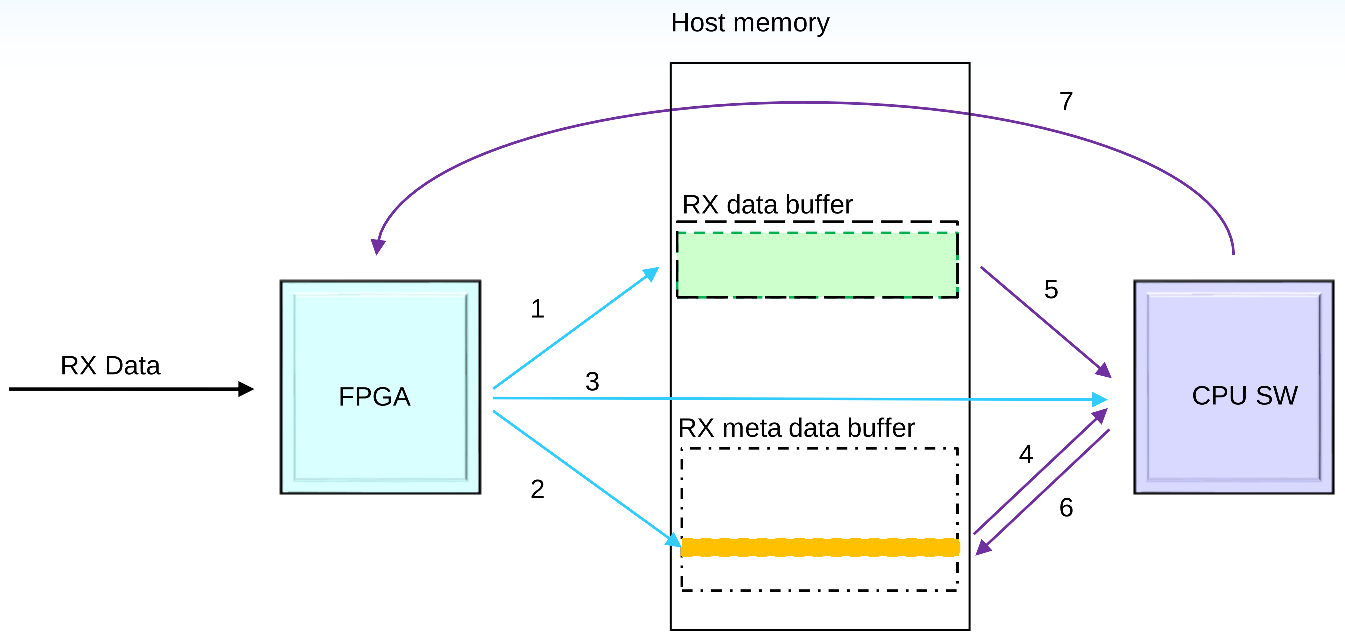
31	30 … 0
F	Length N in bytes of valid data

F = Full = ‚1‘, identifies that the associated DB contains valid data with length N
‚0‘ identifies an empty data buffer

The user is free to define the following:

- Number of MBs and associated DBs per channel
- Meta data buffer content is user defined (timestamp, tags, identifier) except for the first DW
- 64-Bit Core : MB length can be any DW number
- 256-Bit Core : MB length can be either 1 DW or multiples of 8 DW

Variable size : RX – FPGA writes data to the host memory



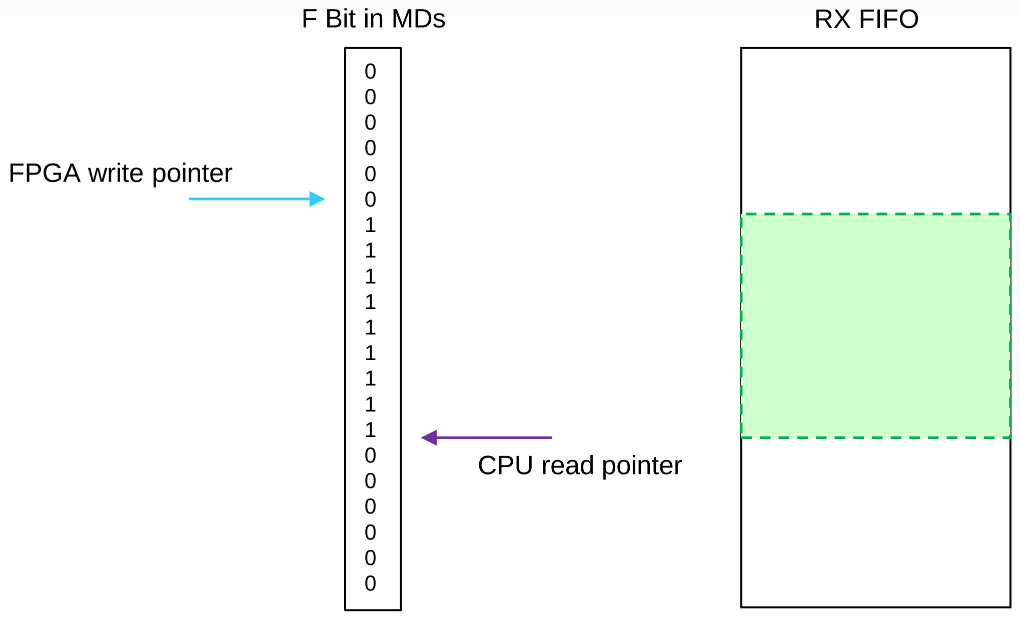
FPGA

- Step 1:
FPGA transmits valid data to the next empty data buffer
- Step 2:
FPGA transmits the meta data and qualifies the associated data buffer as „full“
- Step 3:
FPGA issues an interrupt

Smartlogic Driver

- Step 4:
Driver reacts to the interrupt and examines the first DW of meta data
- Step 5:
If the first DW has the F-Bit set to ‚1‘, the data buffer with length N is offered to the user application
- Step 6:
After the data is processed, the driver clears the F-Bit in the first meta data DW in order to indicate, that the databuffer is empty
- Step 7 : Driver transfers the start address of the now empty buffer back to the FPGA

Variable size : RX can be considered as a FIFO



RX overrun protection

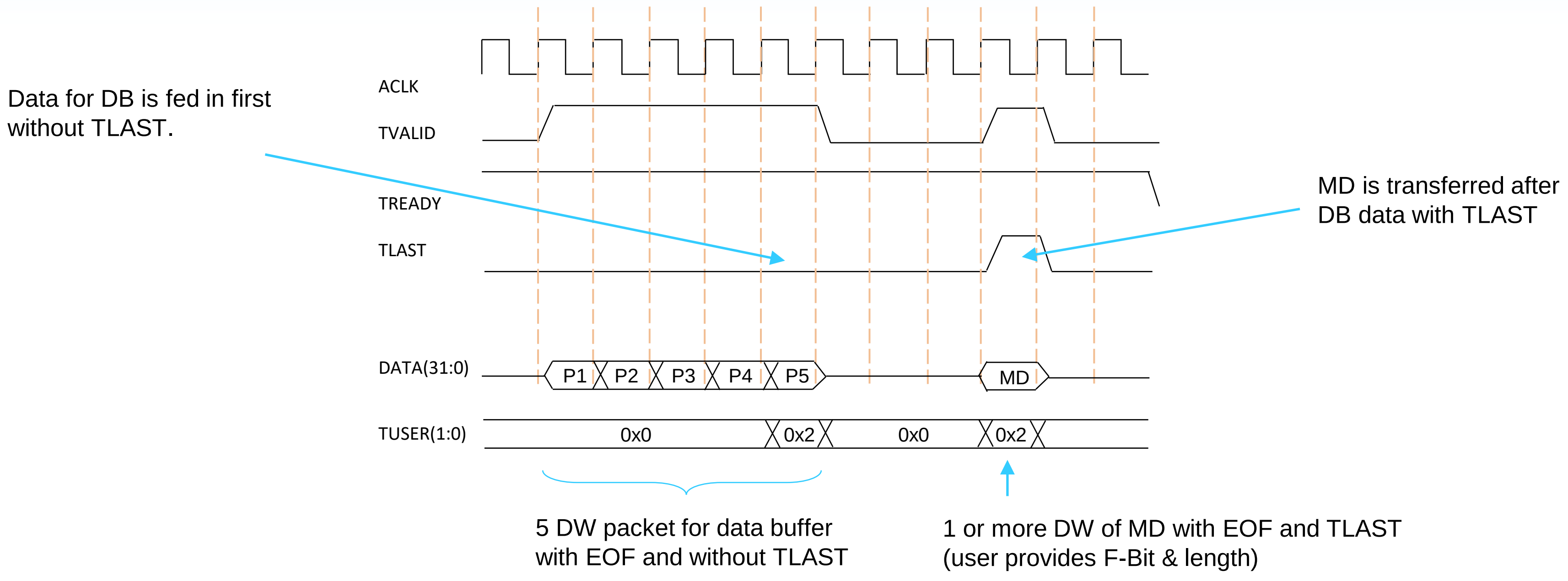
- The FPGA fills only databuffers for which it has permission
- Permission to fill a specific DB is granted by the CPU in the following way:

In the beginning, all startaddresses are transmitted to the FPGA, so that the FPGA is allowed to fill each buffer one time. After one data buffer is filled, the corresponding address is cleared out of the address FIFO.

After the CPU has processed the data in the receive data buffer the cpu gives write permission again to this buffer by writing the address of this buffer back in the FPGA address FIFO of the core.

Variable size : How RX data and meta data are passed (FLEX IP)

Timing diagram for s<nn>_axis:

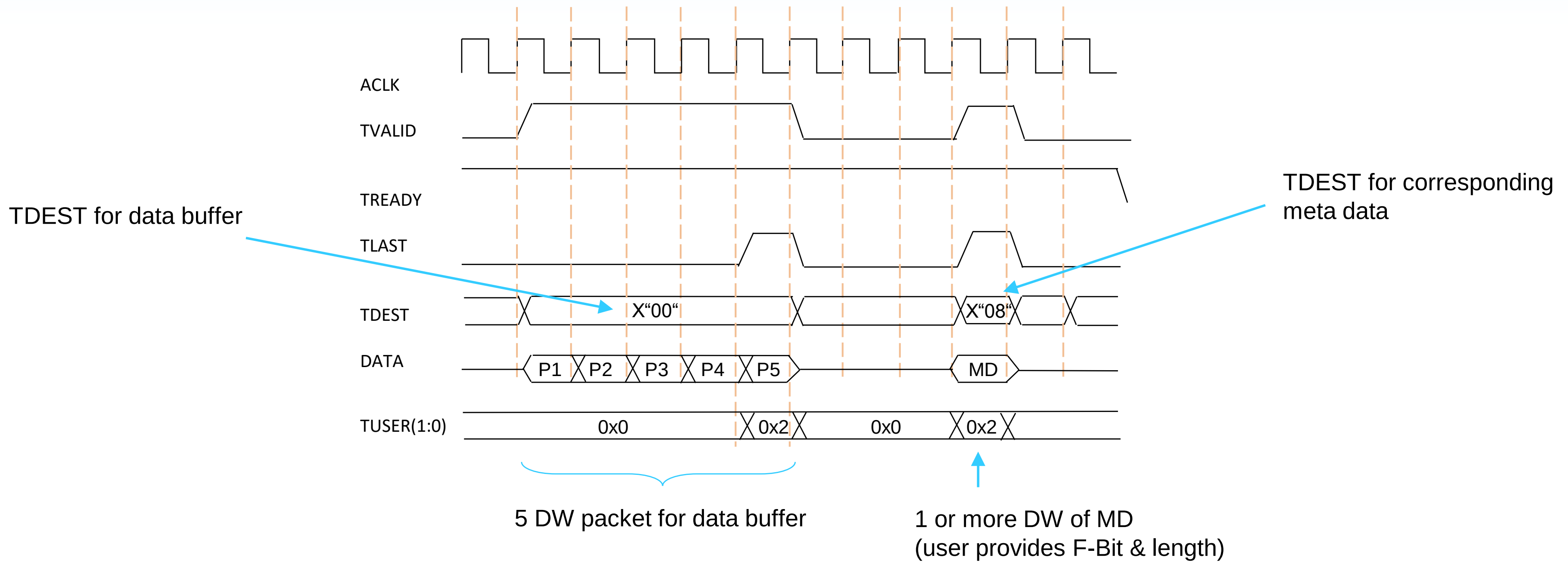

Data for DB is fed in first without TLAST.
ACLK
TVALID
TREADY
TLAST
DATA(31:0)
P1
P2
P3
P4
P5
MD
TUSER(1:0)
0x0
0x2
0x0
0x2
MD is transferred after DB data with TLAST
5 DW packet for data buffer with EOF and without TLAST
1 or more DW of MD with EOF and TLAST (user provides F-Bit & length)

- Metadata is transmitted on the same s_axis interface after the data buffer data.
- Back to back transmission is possible.
- The end of a MD must be signaled with tlast.
- Tuser must be driven with 0x2 in the last dataphase of data and the last dataphase of MD.
- In case of an uneven byte count for DB, pad the last dataphase with dummy data. This is uncritical, since the true byte length is presented to the CPU via the MD length field.
- Example shows 32-Bit s_axis data bitwidth. Other data bit widths are allowed, but each MD must contain at least 1 DW of data.

Variable size : How RX data and meta data are passed (HCC IP)

SMARTLOGIC

Timing diagram for s<nn>_axis:


ACLK
TVALID
TREADY
TLAST
TDEST
DATA
TUSER(1:0)
TDEST for data buffer
TDEST for corresponding meta data
X"00"
X"08"
P1
P2
P3
P4
P5
MD
0x0
0x2
0x0
0x2
5 DW packet for data buffer
1 or more DW of MD (user provides F-Bit & length)

- MDs are transmitted on the same s_axis interface after the data buffer data
- Back to back transmission is possible
- The end of a packet of a MD must be signaled with tlast and tuser = 0x2.
- Tuser must be driven with 0x2 in the last dataphase of a DB to switch to the next DB
- TDEST must not change during a packet
- In case of an uneven byte count, just pad the last dataphase with dummy data. This is uncritical, since the true byte length is presented to the CPU via the MD length field
- Example shows 32-Bit s_axis data bitwidth. Other data bit widths are allowed, but each MD must contain at least 1 DW of data.

Variable size : DB and MB channel mapping in sldma_test.cpp

SMARTLOGIC

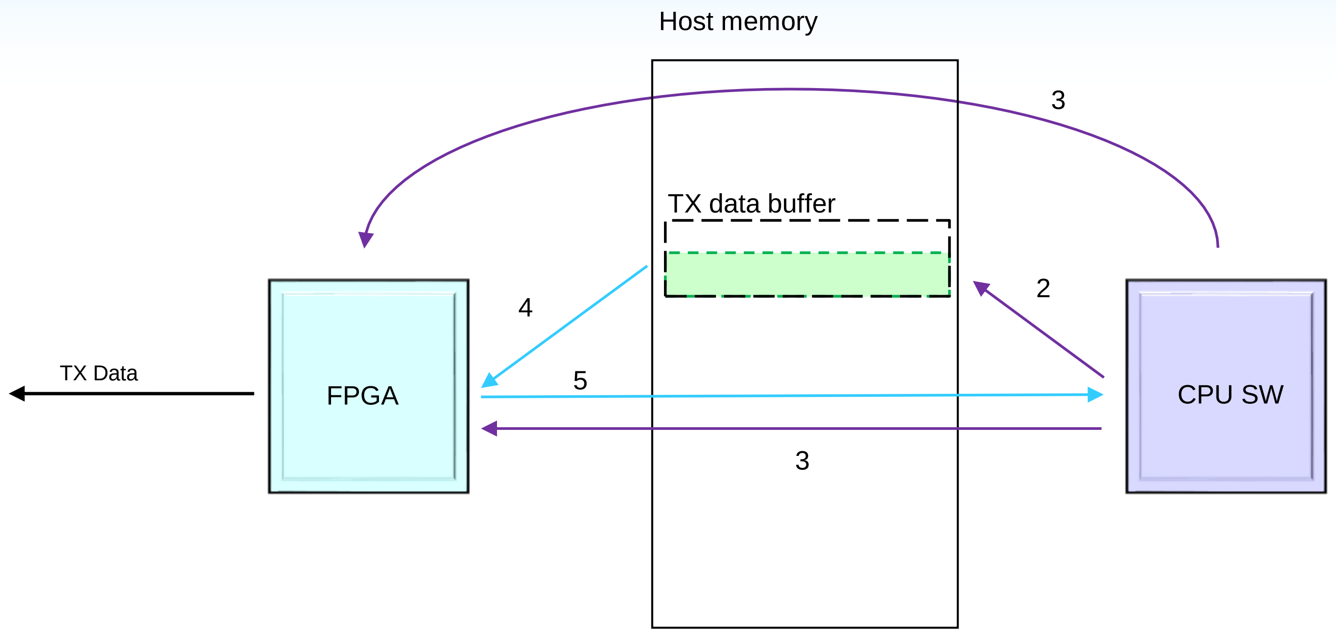
The reference design DMA_Demo3 features 8 upstream channels that transmit data and meta data.

The following table shows the default mapping of the TDEST values for data and meta data :

s_axis if #	TDEST for DB #	TDEST for MB #
0	0	8
1	1	9
2	2	10
3	3	11
4	4	12
5	5	13
6	6	14
7	7	15

If this mapping is kept, the sldma_test.cpp adaptions can be held at a minimum.

Variable size : TX – FPGA reads data from the host memory



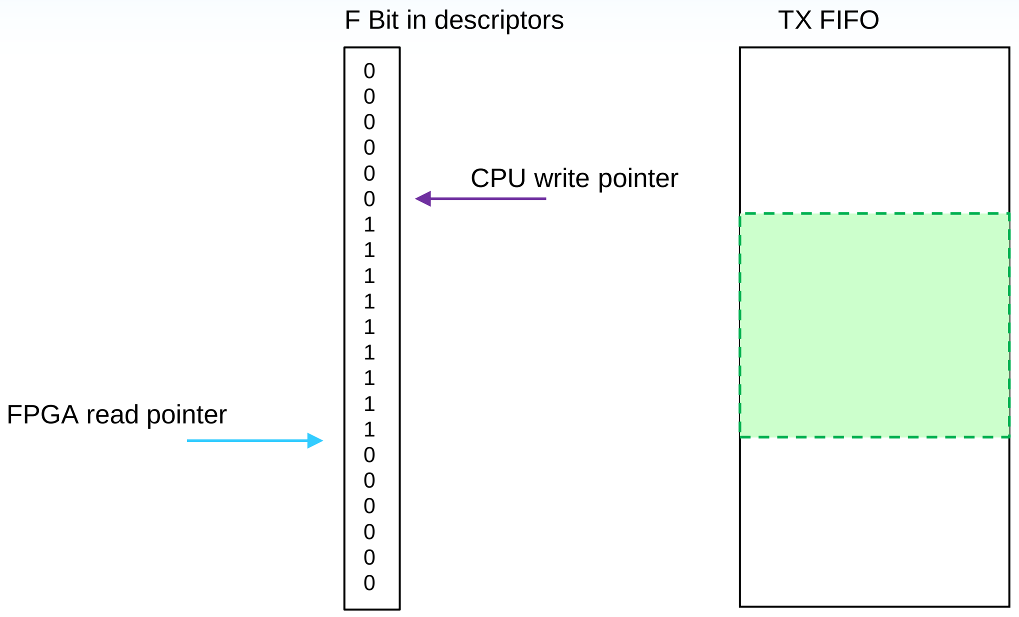
FPGA

- Step 4:
FPGA reads the data buffer
- Step 5:
FPGA issues an interrupt

User Application / Smartlogic Driver

- Step 1:
User app checks if there is an empty TX data buffer
- Step 2:
User app fills the data buffer
- Step 3: Driver informs the FPGA to read the data buffer by writing startaddress and length of TX DB

TX can be considered as a FIFO



TX overrun protection

- The user app has to issue a request for a TX data buffer pointer
- Permission to reuse a TX data buffer is granted by the FPGA in the following way:

In the beginning, user app has the permission to fill all TX data buffers one time

After the FPGA has read the data in a transmit data buffer, the FPGA gives write permission again to the CPU for this buffer by issueing an interrupt to the driver. The interrupt indicates that the TX data buffer has been successfully transmitted and can be re-used again.

16 channels : DB and MB channel mapping

In case more than 8 channels are required, the following TDEST mapping is required for data and meta data:

s_axis if #	TDEST for DB #	TDEST for MB #
0	0	16
1	1	17
2	2	18
3	3	19
4	4	20
5	5	21
…	…	…
15	15	31

Please make sure to set interrupt_scheme_g to 4 and set write_tdest_width_g to 5