AWR1xxx Data Path

This application report is a guide to the XWR1 data path programming.

Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

This purpose of this document is to capture the details of the various configurations and the programming model of the modules that are involved in the storage and transfer, and relay all the data of interest that is generated (raw and/or processed) locally within the device over high speed interfaces.

Figure 1 depicts a typical data flow in AWR16xx device variants that includes:

- Data sources - analog-to-digital converter (ADC) buffer, chirp profile, chirp quality buffer, transfer buffer
- Data movement and transfer across modules - over eDMA
-
  - DSP that performs multi-dimensional FFT processing on the analog-to-digital converter (ADC) data and generates an object list in transfer buffer.
  - CBUFF that is responsible for relaying the data (raw and/or processed) generated from within the device externally over the high-speed interfaces CSI2 and/or LVDS.
  Data sinks

The high-speed interfaces supported in device variants are:

- AWR12xx family of devices have support for LVDS and CSI2 transfers (any one active at a time) over up to four lanes.
- On AWR14xx family of devices, up to four lanes of LVDS transfer are supported.
- On the AWR16x family of devices, only up to two lanes of LVDS transfer are supported.
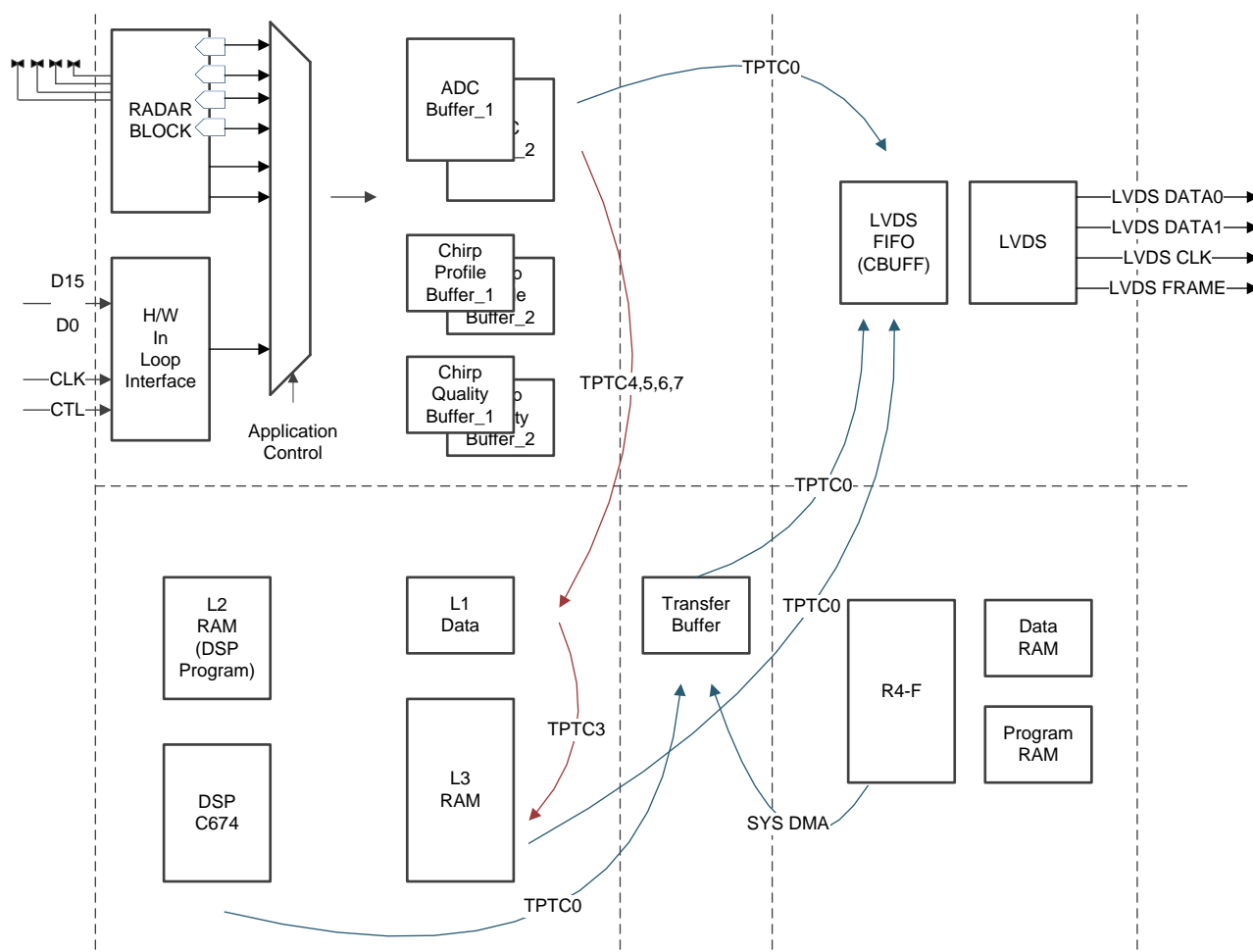

Figure 1. Typical Data Flow in AWR16xx Devices

2   Data Sources

2.1   ADC Buffer

The analog signals received on each of the configured receive (Rx) channels in the device passes through a pre-conditioning over the Analog and Digital Front End (DFE) and the resulting data at the configured sampling rate is stored in the ADC buffer. Data corresponding to all the configured Rx channels is stored within this buffer.

The ADC buffer is implemented as a double buffering (ping-pong) mechanism that allows for one buffer to be written to (filled) while the other one is being read out (emptied).

The size of the ADC buffer is:

- 16 KB for each ping and pong buffers on AWR12xx and AWR14xx devices
- 32 KB for each ping and pong buffers on AWR16xx

2.1.1   Input Sources

The source of the data filled in the ADC buffer is selectable and can be:

- The data as received over the digital front-end chain corresponding to a live scenario of device operation
- In case of AWR16xx, the data fed in over the hardware in the loop (HIL) mechanism corresponding to a playback mode of operation

Register Name	Register Field	Description
DSS_REG:DMMSWINT1 (Applicable for AWR16xx)	DMMADCBUFWREN	0: DFE Samples 1: HIL Samples
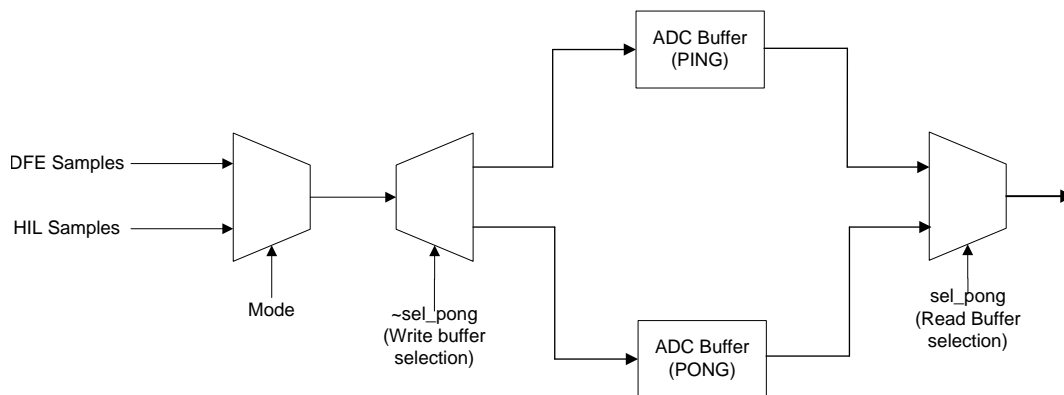
Figure 2. ADC Buffer Write Sources

2.1.2   Interleaved and Non-Interleaved Modes

The data corresponding to all the configured receive (Rx) channels are stored in the ADC buffer. There are two possible storage formats of the data in the ADC buffer with each format best suited (also recommended) for use with a particular variant:

- On AWR12xx and XWR14xx device variants, the interleaved mode of storage is recommended as it also facilitates easy mapping and transfer of each Rx channel data over a corresponding lane.
- On the AWR16xx device variants, only the non-interleaved mode of storage is supported, which makes it more conducive for the processing of the ADC data by the DSP processor.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG1	ADCBUFRL2CHINTRL	0: Interleaved mode 1: Non-interleaved mode

2.1.2.1   Interleaved Mode (AWR12xx/AWR14xx Mode)

The storage of the ADC samples in the ADC buffer for the sample interleaving configuration is as depicted in Figure 3.

- In case of three ADC channels, only 96 bit of a 128-bit row of buffer is filled. Hence, the amount of data transferred over eDMA has to also account for the additional invalid samples.
- The light and dark shades of the same color correspond to I and Q samples.
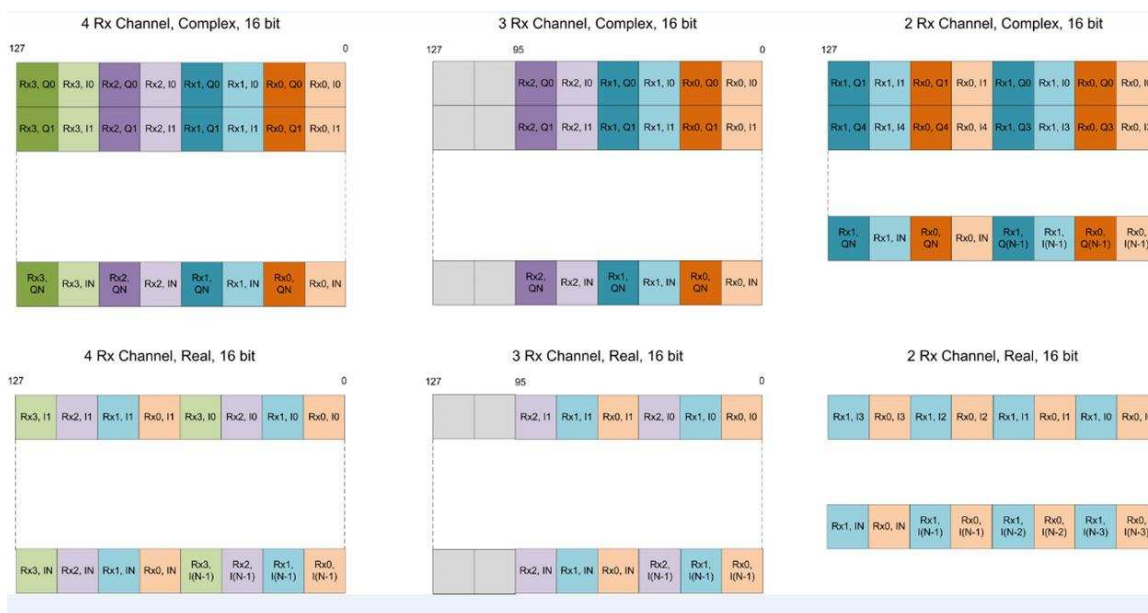
Figure 3. Interleaved Mode of Storage

2.1.2.2   Non-Interleaved Mode (AWR16xx Mode)

In the non-interleaved mode of storage, the ADC data corresponding to each Rx channel are grouped and stored together allowing easy processing of the related data corresponding to each channel. The storage offset for each of the channels is configurable. Also depending on the number of channels configured, the offset to store the data can be moved to allow for larger amount of data to be stored within the same buffer for reduced number of Rx channels.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG2	ADCBUFADDRX0, ADCBUFADDRX1	128-bit address offset to be added to the internal address pointer for Rx0, Rx1 writes in non-interleaved mode

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG3	ADCBUFADDRX2, ADCBUFADDRX3	128-bit address offset to be added to the internal address pointer for Rx2, Rx3 writes in non-interleaved mode
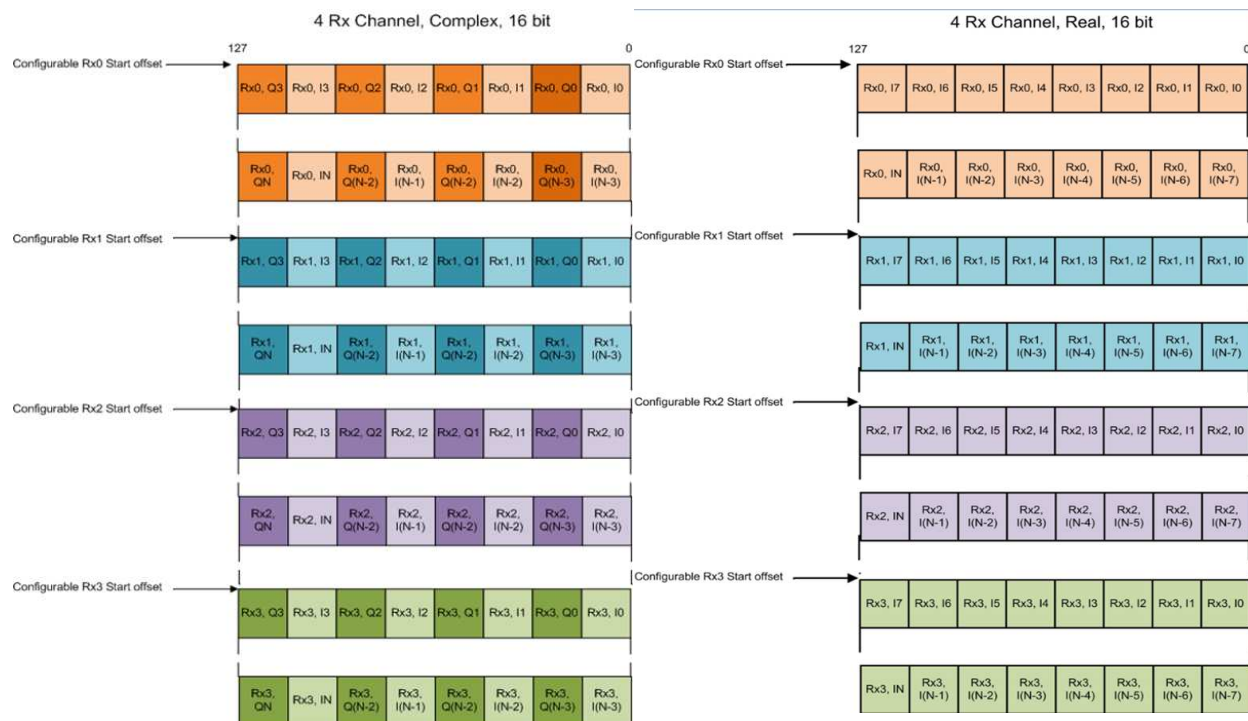
Figure 4. Non-Interleaved Mode of Storage

2.1.3 Data Format

2.1.3.1 Real and Complex Samples

This field allows for the configuration of the ADC sample stored in the ADC buffer to be real or complex.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG1	ADCBUFREALONLYMODE	0: Complex data mode 1: Real data mode

2.1.3.2 IQ Swap

This field allows for a swap in the storage of the I and Q samples.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG4	ADCBUFIQSWAP	0: I is stored in LSB and Q in MSB 1: Q is stored in LSB and I in MSB

2.1.4   Ping-Pong Switch Select

The completion of filling of an ADC buffer and a subsequent switch to the other buffer for filling the data is determined by the completion of the filling of data corresponding to the configuration of the number of chirps of data to be stored in each of the ping pong buffers.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG4	ADCBUFNUMCHRPPING, ADCBUFNUMCHRPPONG	Number of chirps to be stored in Ping, Pong buffer. This register should be programmed with one less than the actual number needed. This is used when data is written to Pong, Ping Memory. The value written to this field should be the same as that configured for Pong, Ping.

> **NOTE:** The maximum number of chirps supported in each memory is 8.

2.2   Chirp Parameters Information

2.2.1   Fields

Table 1. Chirp Parameter Fields

Field	Description	
Field-1 [15:0]	Chirp Number [11:0]	In legacy frame configuration, chirp number for starts from 1 and increments for each chirp within the frame and resets to 0 for the next frame.
		In advanced frame configuration chirp number starts from 1 and increments for each chirp within the burst and resets to 0 for the next burst.
	Unused [15:12]	Unused
Field-2 [15:0]	Channel Number [1:0]	The receive channel number that is encoded
		00 – RX0, 01 – RX1, 10 – RX2, 11 – RX3
	Chirp Profile Index [5:2]	The profile number to which the chirp belongs
	Reserved [11:6]	Reserved
	Unused[15:12]	Unused

2.2.2   CP Buffer Select

The chirp parameters are stored in hardware registers when the information is updated by the DFE. In case of updating the information over HIL interface, the CP memory is selected using the mux select. The address mapping however remains the same for the application.
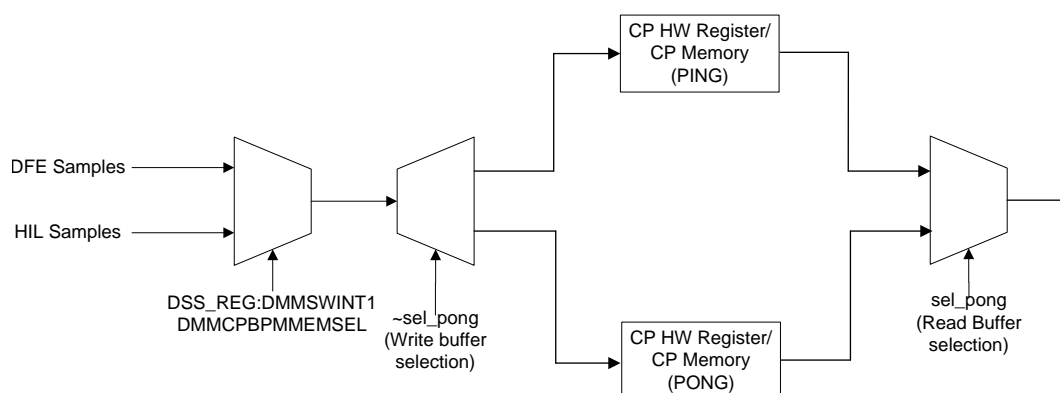

Figure 5. CP Buffer Select

2.2.3 Storage Details

The chirp parameters follow the storage format of the ADC samples. CPREG[0-15] are registers in the DSS_REG module.
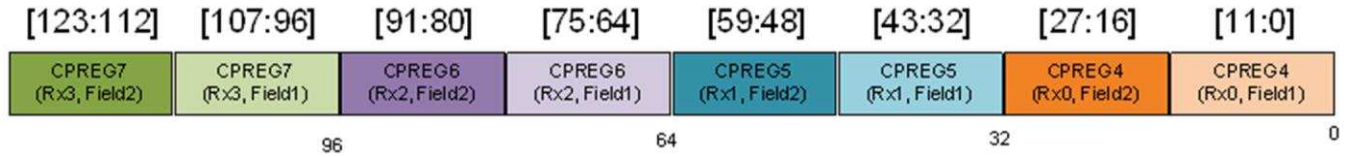
Interleaved mode:

Figure 6. CP Storage - ADC Non-Interleaved, 4 Rx Channel
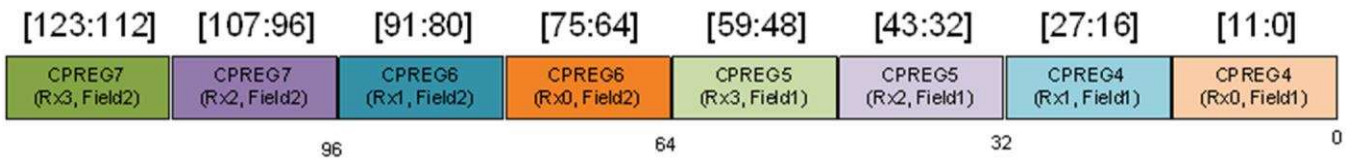
Figure 7. CP Storage - ADC Interleaved, Real, 4 Rx Channel
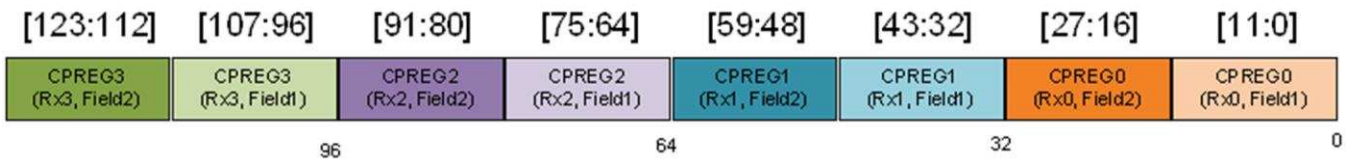
Figure 8. CP Storage - ADC Non-Interleaved, 4 Rx Channel

In case of AWR16xx devices, in the non-interleaved mode and ping-pong switch of the ADC buffer based on multiple chirps (upto 8), the CP data is stored as above in subsequent locations:

DSS_REG_VBUSM – CH[0-7]CPREG[0-3]

Figure 6 captures the configuration for 1 chirp and corresponds to:

DSS_REG_VBUSM – CH0CPREG[0-3]

2.3 Chirp Quality Information (TBD)

2.3.1 Fields

Table 2. Chirp Quality Fields

Chirp Quality Information	Default Base Address (as viewed by eDMA)	Description
Chirp Quality 0	0x21028000 (AWR12/XWR14/XWR16)	Interference monitoring (TBD)
Chirp Quality 1	0x21028200 (AWR12/XWR14) 0x21288800 (XWR16)	Interference monitoring (TBD)
Chirp Quality 2	0x21028400 (AWR12/XWR14) 0x21289000 (XWR16)	Saturation monitoring (TBD)

2.3.2 CQ Data Input Select

The source of the chirp quality data can be configured to be populated by DFE or in case of the playback over the HIL interface. The CQ data also follows the ping-pong scheme similar to the ADC buffer.

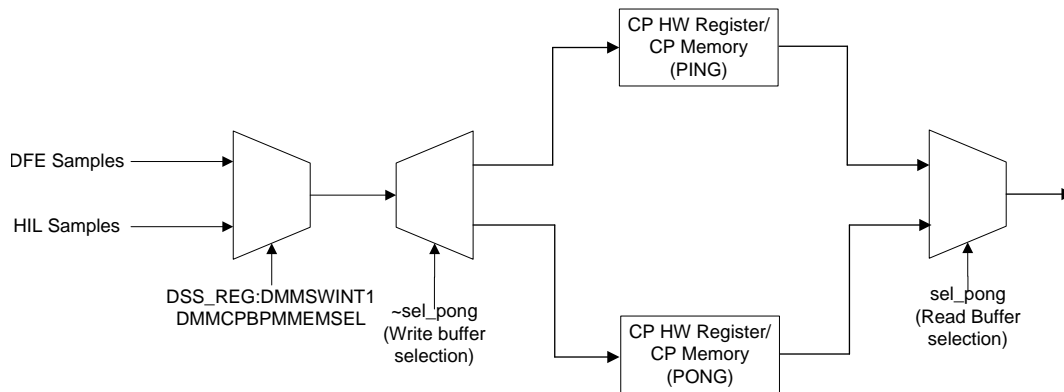
Figure 9. CQ Data Input Select

2.3.3 Storage details (TBD)

The chirp quality data information is available on a per chirp basis and is not per Rx channel. Therefore, the chirp quality data is transferred over all the lanes in a chunk. The chirp quality data information also follows the storage format as in the ADC buffer.

The data corresponding to each of the CQ type can be placed at a configurable offset in the CQ memory. For the case of multiple chirps, the CQ data is stored back to back at the next 128-bit aligned address within each CQ type.

The data format of CQ is also configurable (12 bit, 14 bit, 16 bit) and can be different from that of the ADC data.

Format	[127:112]	[111:96]	[95:80]	[79:64]	[63:48]	[47:32]	[31:16]	[15:0]
Raw8/LVDS-16b	16bits valid	16bits valid	16bits valid	16bits valid	16bits valid	16bits valid	16bits valid	16bits valid
Raw14/LVDS-14b	14bits valid	14bits valid	14bits valid	14bits valid	14bits valid	14bits valid	14bits valid	14bits valid
Raw12/LVDS-12b	12bits valid	12bits valid	12bits valid	12bits valid	12bits valid	12bits valid	12bits valid	12bits valid

Figure 10. CQ Data Packing in Memory

Format	[127:112]	[111:96]	[95:80]	[79:64]	[63:48]	[47:32]	[31:16]	[15:0]
Raw8/LVDS-16b	Unused	Unused	16bits valid	16bits valid	16bits valid	16bits valid	16bits valid	16bits valid
Raw14/LVDS-14b	Unused	Unused	14bits valid	14bits valid	14bits valid	14bits valid	14bits valid	14bits valid
Raw12/LVDS-12b	Unused	Unused	12bits valid	12bits valid	12bits valid	12bits valid	12bits valid	12bits valid

Figure 11. CQ Data Packing in Memory for 3 Channel 3 Lane

2.3.4 Data Handling Requirement on Receiver

As the CQ data is inherently 16-bit information, when sent over the lanes in the 12-bit or 14-bit mode, the information is split over the 12-bit or 14-bit data. The receiver of the data has to appropriately reconstruct the 16-bit value by packing the information in the 12- or 14-bit data.

In order to avoid this overhead on the receiver, it is recommended to send CQ data as a separate packet of 16 bit if the ADC data has to be 12 bit or 14 bit.

2.4 Application Data

In addition to the data generated by the hardware and populated in the sources (ADC buffer, CP registers and CQ memories), the data generated or determined by the application can also be sent out on the HSI lanes. This may include the results of the DSP processing (in case of AWR16xx devices), metadata/header information to tag along with the data.

There are memories available for such transfers as determined by the application software. Examples of such memories are: Dedicated memories (transfer buffer (8 KB)) as well as application memories such as R4F TCMB, DSP L2-L3 memories, handshake memories.

The generation and transfer of the data is the responsibility of the application.

3 Data Transfer

3.1 Enahanced DMA

- Fully orthogonal transfer description
  - Three transfer dimensions
  - A-synchronized transfers: 1 dimension serviced per event
  - AB-synchronized transfers: 2 dimensions serviced per event
  - Independent indexes on source and destination
  - Chaining feature allows 3-D transfer based on single event
- Flexible transfer definition
  - Increment or constant addressing modes
  - Linking mechanism allows automatic PaRAM set update
  - Chaining allows multiple transfers to execute with one event
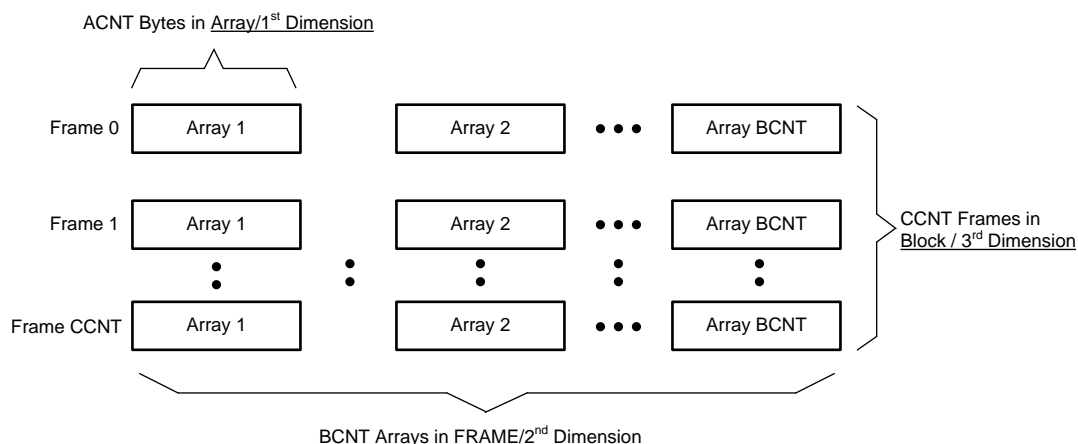

Figure 12. EDMA Channel Transfer Terminology

- 1st Dimension or Array (A): The first dimension in a transfer consists of ACNT contiguous bytes.
- 2nd Dimension or Frame (B): The second dimension in a transfer consists of BCNT arrays of ACNT bytes. Each array transfer in the second dimension is separated from each other by an index programmed via SBIDX or DBIDX.

- 3rd Dimension or Block (C): The third dimension in a transfer consists of CCNT frames of BCNT arrays of ACNT bytes. Each transfer in the third dimension is separated from the previous by an index programmed via SCIDX or DCIDX.

3.1.1 Chaining Transfers

Chaining is a mechanism by which the completion of one transfer automatically sets the event for another channel. This is particularly useful to transfer data from different sources based on a single trigger event.

An example usecase is chaining the EDMA transfers of data associated to a single chirp, available at different sources (from ADC buffer, chirp parameter information and chirp quality information), whenever a chirp available hardware event is triggered. The chirp available hardware event can trigger the EDMA chain (via CBUFF) and this can cause the data to be transferred from the chained sources onto the CBUFF and subsequently out on the high speed interface lanes without any CPU intervention.

3.1.2 Linking Transfers

The EDMA provides a mechanism known as linking, which allows the entire PaRAM set to be reloaded from a location within the PaRAM memory map (for both DMA and QDMA channels). Linking is especially useful for maintaining ping-pong buffers, circular buffering, and repetitive or continuous transfers all with no CPU intervention. Upon completion of a transfer, the current transfer parameters are reloaded with the parameter set pointed to by the 16-bit link address field (of the current parameter set).

An example usecase is the linking of EDMA PaRAM sets to allow for the configuration information to be automatically reloaded when the EDMA chain corresponding to one chirp information is completed. The reconfiguration for the next chirp data is automatically performed without any CPU intervention and this process repeats for the transfer of all the "N" chirp information associated with a frame.

4   Data Sink and Transfer Over HSI

4.1   *Common Buffer*
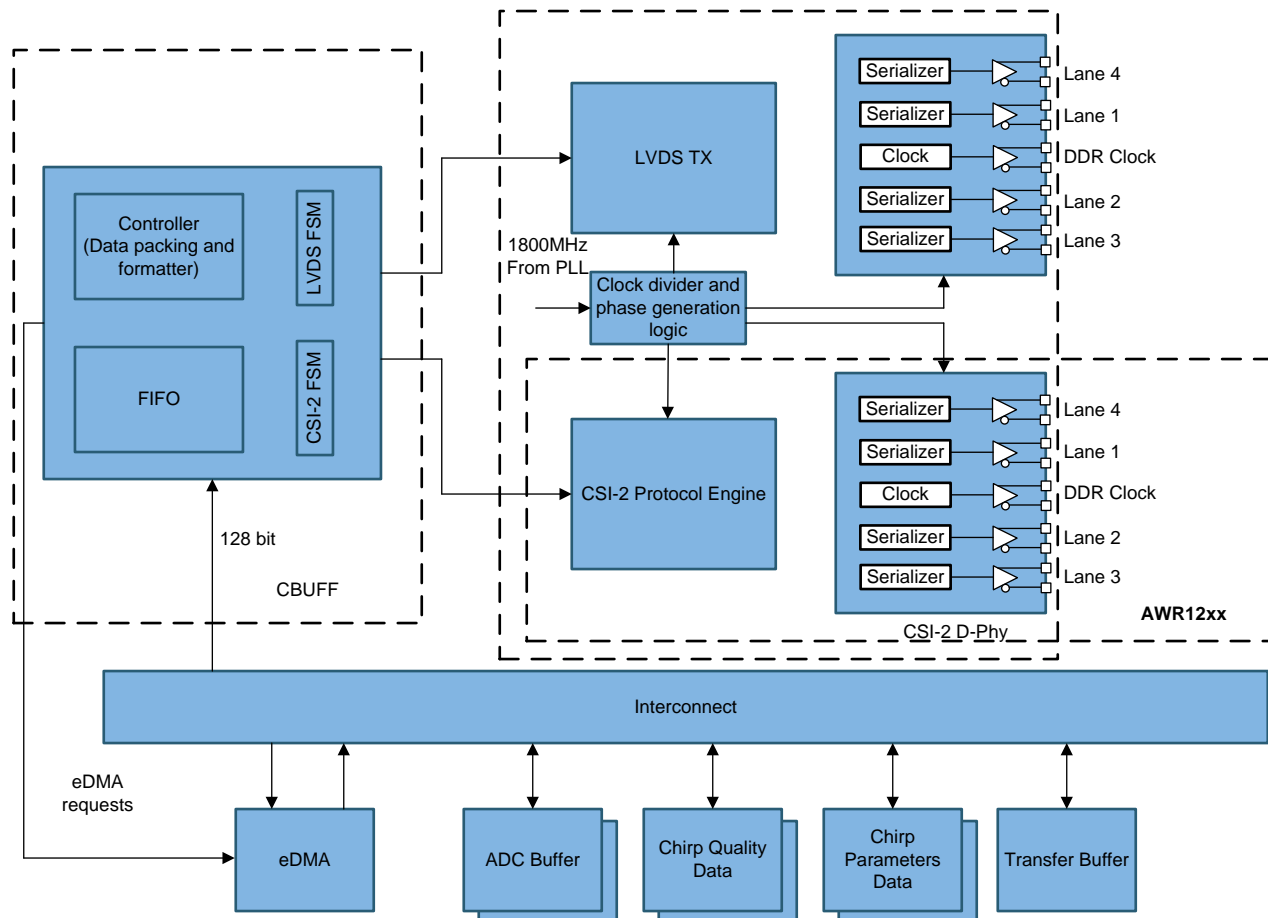
Note:
CSI2 interface : Supported only in AWR12xx
LVDS interface : In case of AWR16xx, is a 2 lane data interface. Lanes 3&4 are not available at device boundary.

Figure 13. CBUFF Overview

The Common buffer (CBUFF) controller is responsible for the transfer of data generated from within the device over the high speed interfaces (HSI), CSI2 or LVDS. It is a common data sink that has additional logic to format and packetize the information filled in its 128-bit wide FIFO in accordance with the interfacing requirements of the CSI2 or LVDS protocol and physical layers.

For CSI2, additional IPs such as the CSI2 Protocol Engine and the CSI D-Phy need to also be configured to transmit the data.

For LVDS, the LVDS TX IP is closely integrated with the configuration of CBUFF and does not need to be configured separately.

CBUFF has the necessary flow control mechanisms to manage the different rates of eDMA filling the FIFO and the contents in the FIFO being emptied out over the HSI lanes.

CBUFF always handles external data at 16-bit boundary (1 CBUFF Unit).

4.1.1 State Machine

The CBUFF controller state machines inputs are designed to be in sync with the chirping scheme of the radar device; the hardware triggers to the state machine are generated from the DFE. The output of the CBUFF state machine is primarily tuned to the interfacing and handshake mechanism of the CSI2 protocol engine. So, while the state machine captures the CSI2 handshake, the sate transitions are mimicked in case of LVDS. However, on the LVDS lanes, only the data packets are available, not any of the sync packets (VSync Start-End, HSync Start-End, Long Packet Header).
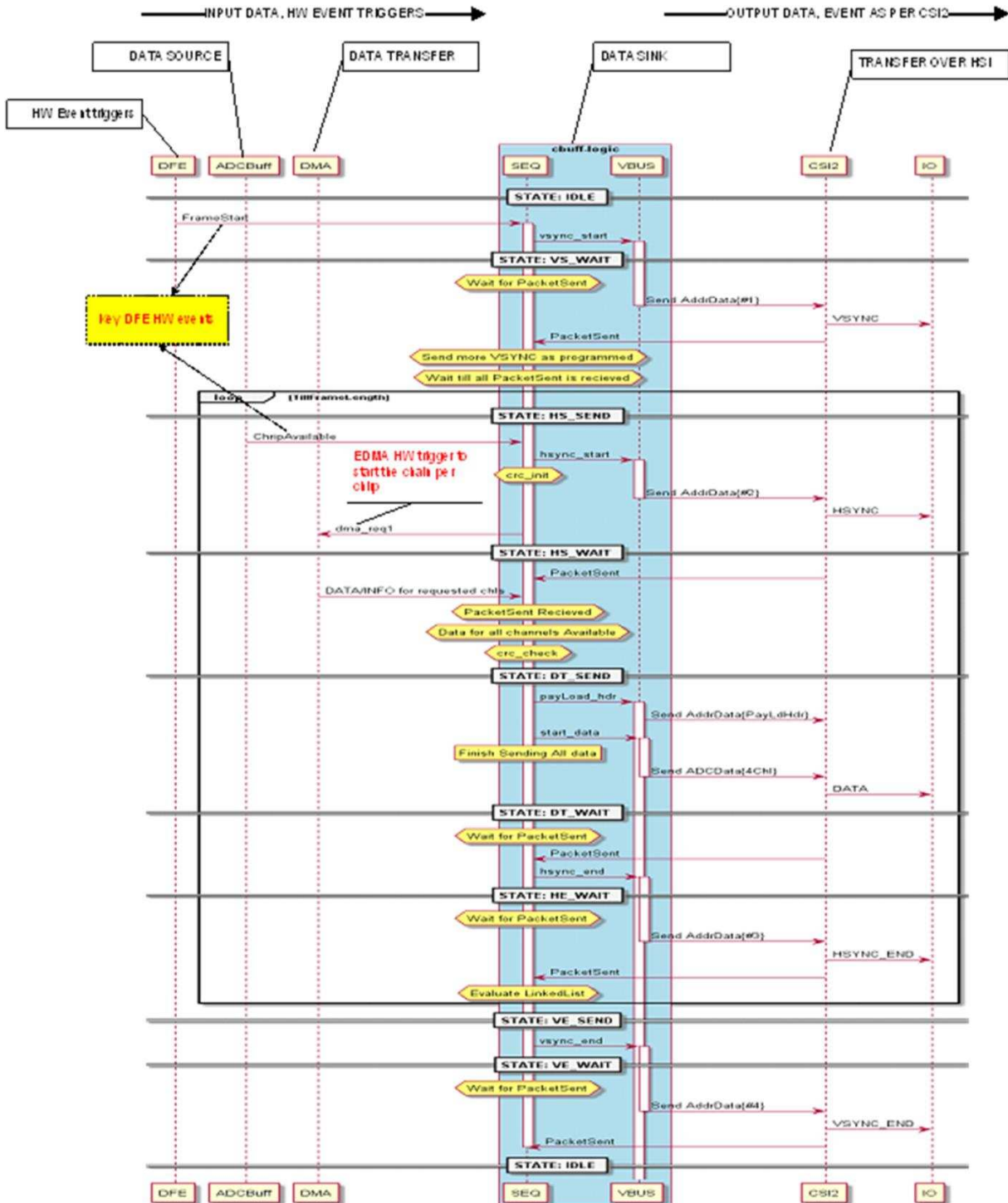

Figure 14. CBUFF State Machine

The key DFE events that trigger the state machine are the "Frame Start" and "Chirp(s) Available" events. The "Chirp(s) Available" event is generated during every ping-pong switch of the ADC buffer. On every "Chirp(s) Available" event, the CBUFF kicks the eDMA configured eDMA channel for the transfer and the flow control mechanism ensures the complete transfer of the data over the HSI interface.

4.1.2 CBUFF Configurations

4.1.2.1 General Configuration

All configurations are performed by holding the CBUFF controller in reset state. The key configurations of the CBUFF are:

- Selection of the HSI interface (LVDS/CSI2)
- Trigger sources are configured to be generated by hardware when the transfers over HSI are intended to happen at a chirp periodicity (in sync with the Radar timing engine operation for the configured chirping scheme).
- Trigger sources are configured to be generated by software when the transfers over HSI are not necessarily aligned with the chirp periodicity. In these cases, it is the applications responsibility to setup the input triggers to the CBUFF controller state machine appropriately to initiate the transfer of data.

Table 3. CBUFF General Configuration Parameters

Register Name	Register Field	Description
CONFIG_REG_0	CFG_1LVDS_0CSi	0: Send data over CSI-2 1: Send data over LVDS
	CFG_ECC_EN	0: Disable ECC on the CBUF FIFO 1: Enable ECC on the CBUF FIFO
	cftrigen	Select Frame Start Trigger Source 0: Frame trigger is generated by hardware 1: Frame trigger is generated by software
	CFG_SW_TRIG_EN	Select Chirp Available Trigger Source 0: Chirp available trigger is generated by hardware 1: Chirp available trigger is generated by software
	CFG_CHIRP_AVAIL_TRIG	Software trigger generation: write 1 to this bit to generate a chirp available software trigger
	CFG_FRAME_START_TRIG	Software trigger generation: write 1 to this bit to generate a frame start software trigger
	cswcrst	CBUFF controller software reset 1 => RESET the CBUFF controller 0 => RELEASE RESET for CBUFF controller

4.1.2.2 Packetization Configurations

The packetization configurations are captured in Table 4.

Being a common HSI interface, while there is a correspondence of the state machine operation and transitions with the CSI2 data flow, there are certain configuration requirements while programming for the transfers over LVDS that are predominantly driven by the need to mimic certain state transitions, and does not really translate to any action or data on the LVDS lanes.

However, the values in some of these registers for LVDS are overloaded to also indicate a choice while also being used for the FSM operation. For example, the selection of sending CRC data over the lanes is determined by the value in the CFG_CMD_HSVAL register.

Table 4. Packet Marker Configuration Parameters

Register Name	Register Field	Description
CFG_SPHDR_ADDRESS	CFG_SPHDR_ADDRESS	CSI2 programming: Configure the CSI_PROTOCOL_ENGINE_CSI_VC_SHORT_PACKET_HEADER Address in the CSI Protocol engine LVDS programming: Configure with the static value: 0x55555555
CFG_CMD_HSVAL	CFG_CMD_HSVAL	CSI2 programming: Configure the HSync Start Short Packet Value LVDS programming: If LVDS CRC is enabled: Configure with the static value: 0x55555555 If LVDS CRC is disabled: Configure with the static value: 0xAAAAAAAA
CFG_CMD_HEVAL	CFG_CMD_HEVAL	CSI2 programming: Configure the HSync End Short Packet Value LVDS programming: If LVDS CRC is enabled: Configure with the static value: 0x33333333 If LVDS CRC is disabled: Configure with the static value: 0xAAAAAAAA
CFG_CMD_VSVAL	CFG_CMD_VSVAL	CSI2 programming: Configure the VSync Start Short Packet Value LVDS programming: Configure with the static value: 0xAAAAAAAA
CFG_CMD_VEVAL	CFG_CMD_VEVAL	CSI2 programming: Configure the VSync End Short Packet Value LVDS programming: Configure with the static value: 0xAAAAAAAA
CFG_LPHDR_ADDRESS	CFG_LPHDR_ADDRESS	CSI2 programming: Configure the CSI_PROTOCOL_ENGINE_CSI_VC_LONG_PACKET_HEADER address in the CSI Protocol engine LVDS programming: Configure with the static value: 0x55555555
CFG_LPPYLD_ADDRESS	CFG_LPPYLD_ADDRESS	CSI2 only programming: Configure the CSI_PROTOCOL_ENGINE_CSI_VC_LONG_PACKET_PAYLOAD address in the CSI Protocol engine

4.1.2.3 Chirps Per Frame Configuration

The CBUFF controller state machine is designed to be in sync with the chirping scheme configured as the primary usecase of the module is to transfer the raw ADC data over the HSI interface. In line with the same, the FSM also loops through the linked list for number of chirps configured per frame.

Table 5. Chirps Per Frame Configuration

Register Name	Register Field	Description
CFG_CHIRPS_PER_FRAME	CFG_CHIRPS_PER_FRAME	Configure the number of chirps in a frame

4.1.2.4 Linked List

The chain of linked list in the CBUFF is analogous to the chain of parameter sets in the eDMA. Each node of the linked list contains the necessary information that triggers certain actions or state transitions on the CBUFF controller. Each node primarily controls the output actions and responses of the state machine and has the necessary information to maintain the flow control.

There are up to 32 linked list entries that can be configured.

Table 6. Linked List Configuration Parameters

Register Name	Register Field	Description
CFG_DATA_LL0	LL0_VALID	0: Linklist entry is invalid 1: Linklist entry is valid
	LL0_HE	CSI-2: 0: Do not send an Hsync End packet after sending this data 1: Send an Hsync End Packet after sending this data LVDS; 0: Entry is not the last data of LVDS frame 1: Entry is the first data in the LVDS frame
	LL0_HS	CSI-2: 0: Do not send an Hsync Start packet after sending this data 1: Send an Hsync Start Packet after sending this data LVDS; 0: Entry is not the last data of LVDS frame 1: Entry is the first data in the LVDS frame
	LL0_VCNUM	CSI-2 Configure the Virtual Channel Number for the Long Packet over which this data is sent
	LL0_FMT	Specify the LVDS/CSI2 output format 00 - 16 bit 01 - 14 bit 10 - 12 bit
	LL0_FMT_MAP	LVDS only: 0: Choose CFG_LVDS_MAPPING_LANEx_FMT_0_v 1: Choose CFG_LVDS_MAPPING_LANEx_FMT_1_v
	LL0_FMT_IN	0: The infoming data sources for this linklist is aligned to 128 bit
	LL0_SIZE	Configure the size of the data in terms of the number of samples (not in terms of number of bytes). Sample refers to a 16-bit CBUFF unit

Table 6. Linked List Configuration Parameters (continued)

Register Name	Register Field	Description
CFG_DATA_LL0 (cont.)	LL0_LPHDR_EN	CSI-2:
		0: Entry is start of a new CSI-2 packet. Send the LP payload header before sending data corresponding to this linklist
		1: Linklist is not the start of a long packet but part of a previous packet, directly send data
		LVDS programming;
		0: Entry is the start of a new LVDS fram
		1: Entry is not the start of a new LVDS frame
	LL0_CRC_EN	0: CRC is disabled
		1: This linklist corresponds to ADC buffer data. Enable the CRC check from the ADC buffer to CBUFF.
CFG_DATA_LL0_LPHDR_VAL	LL0_LPHDR_VAL	CSI-2 programming:
		Configure the long packet header to be sent to the protocol engine if the LPHDR_EN field is for the linklist.
		LVDS programming:
		Configure with the static value: 0xBBBBBBBB
CFG_DATA_LL0_THRESHOLD	LL0_RD_THRESHOLD	Confgure the CBUFF read threshold to be reached before sending the data over CSI2/LVDS and start draining the CBUFF FIFO. Static configuration.
	LL0_WR_THRESHOLD	Configure the CBUFF FIFO write threshold over which CBUFF will stall the DMA write to the CBUFF. Static configuration.
	ll0dmam	If the long packet header is enabled, CBUFF can generate a DMA request to trigger the DMA transfer for the new packet.
		0: Send a request on DMA hardware req output line 0
		1: Send a request on DMA hardware req output line 1
		2: Send a request on DMA hardware req output line 2
		3: Send a request on DMA hardware req output line 3
		4: Send a request on DMA hardware req output line 4
		5: Send a request on DMA hardware req output line 5
		6: Send a request on DMA hardware req output line 6
		7: Do not generate DMA trigger

NOTE: Set the LL[X]_RD_THRESHOLD as 0x4 and LL[X]_WR_THRESHOLD as 0x40.

Figure 15 maps the different fields of a linked list node to the actions/transitions in the state machine.

- Broadly they translate to triggers to send packets over CSI2 (for example, HSync Start, Long Packet Header, HSync end, and so forth) and also determine the amount of data to handshake and receive over eDMA to be sent out over HSI.
- For LVDS, it includes information of the format mapping to be applied.
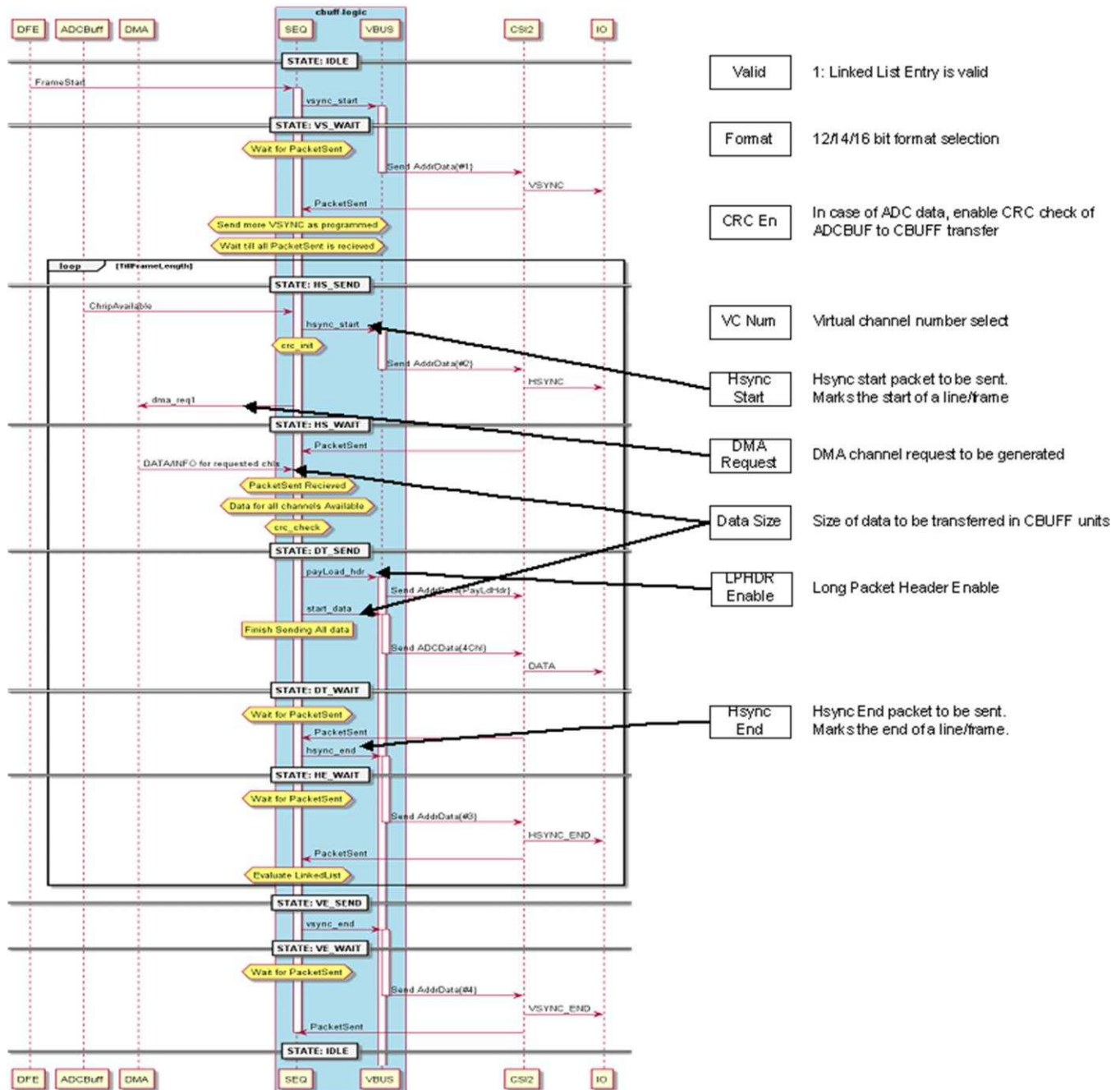

Figure 15. Mapping Linked List Fields to FSM

Limitations:

- A linked list group marked by HS to HE nodes needs to have the same common configurations (Format, VC Num).
- The Size field is 14-bit data and can be a max. of 0x3FFF. Therefore, the maximum size that can be transferred using one node is 0x3FFF CBUFF units (= 32768 – 2 Bytes).

Example Configuration 1

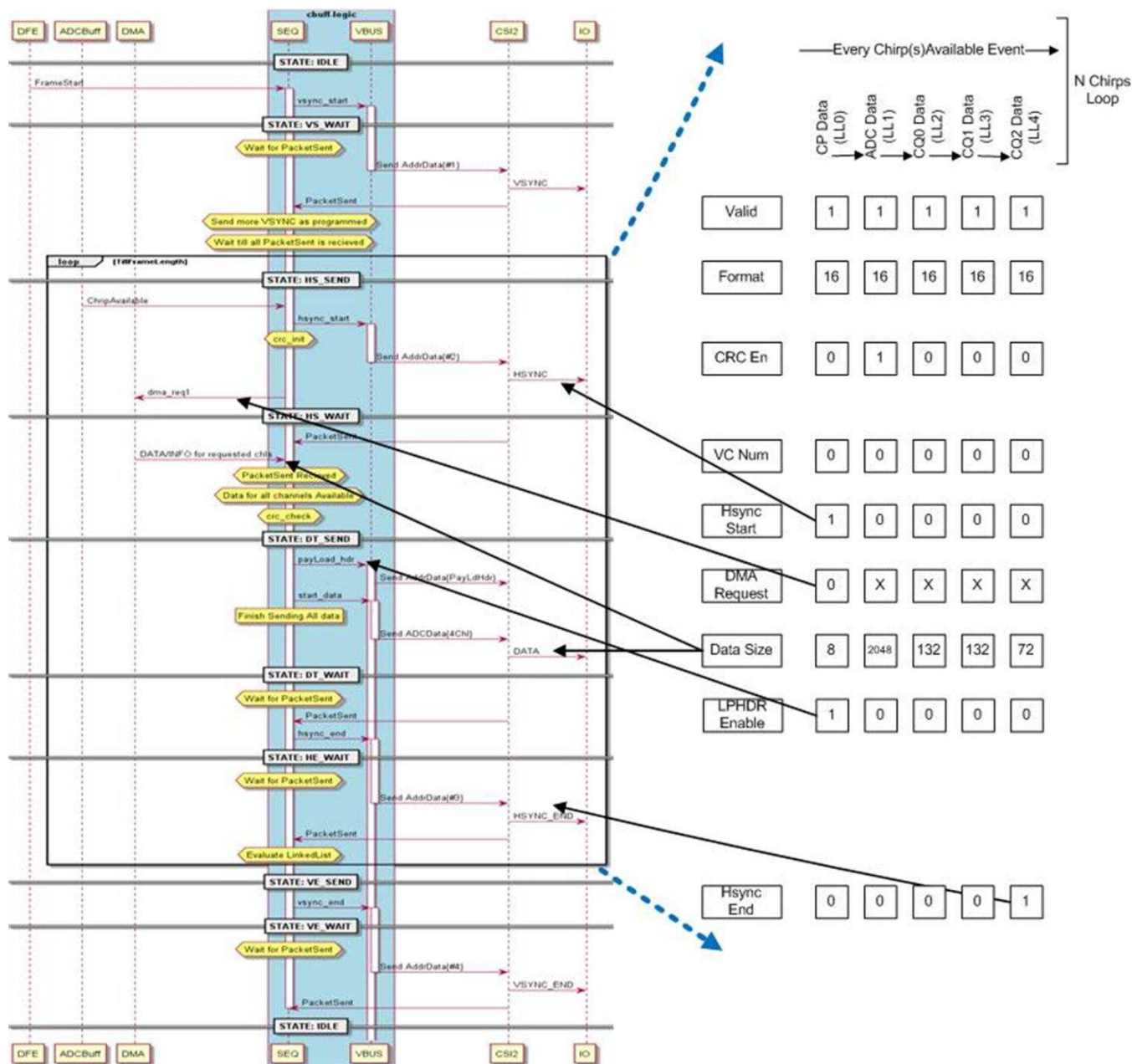Figure 16 depicts the key linked list node configurations to transfer the data for the usecase.

Figure 16. Linked List Setup - Interleaved Mode

The configurations are: four channels, 256 ADC complex samples, interleaved, 16-bit data.

> **NOTE:** The CRC enable is only set for the node transferring ADC data. This translates to ensure that the CRC, computed at the ADC buffer matches the CRC computed at the CBUFF to ensure the integrity of the ADC buffer transfers.

Example Configuration 2 (Non interleaved)

CP (Rx0) → ADC Buffer (Rx0, Non Interleaved) → CP (Rx1) → ADC Buffer (Rx1, Non Interleaved) → CP (Rx2) → ADC Buffer (Rx2, Non Interleaved) → CP (Rx3) → ADC Buffer (Rx3, Non Interleaved) → CQ0 → CQ1 → CQ2

NOTE: It is recommended to have one linked list node per Rx channel data transfer to ensure accurate CRC computation and matching for ADC data transfer.

	CP Data (LL0)	ADC Data (LL1)	CP Data (LL2)	ADC Data (LL3)	CP Data (LL4)	ADC Data (LL5)	CP Data (LL6)	ADC Data (LL7)	CQ0 Data (LL8)	CQ0 Data (LL9)	CQ0 Data (LL10)
Valid	1	1	1	1	1	1	1	1	1	1	1
Format	16	16	16	16	16	16	16	16	16	16	16
CRC En	0	1	0	1	0	1	0	1	0	1	0
VC Num	0	0	0	0	0	0	0	0	0	0	0
Hsync Start	1	0	0	0	0	0	0	0	0	0	0
DMA Request	0	0	0	0	0	0	0	0	0	0	0
Data Size	2	512	2	512	2	512	2	512	2	512	132
LPHDR Enable	1	1	1	1	1	1	1	1	1	1	1
Hsync End	0	0	0	0	0	0	0	0	0	0	0

Figure 17. Linked List Setup - Non-Interleaved Mode

Example Configuration 3 (Multiple packets)

Packet 1: CP → ADC Buffer (Interleaved) → CQ0 → CQ1 → CQ2

Packet 2: CP → CQ0 → CQ1 → Q2

> **NOTE:** A value of 7 for DMA request ensures no new eDMA trigger request is generated. Assumption is that the eDMA chain takes care of transferring all the packets after the first eDMA channel (0) is triggered by CBUFF.

	CP Data (LL0)	ADC Data (LL1)	CP Data (LL2)	ADC Data (LL3)	CP Data (LL4)	ADC Data (LL5)	CP Data (LL6)	CQ1 Data (LL7)	CQ2 Data (LL8)
Valid	1	1	1	1	1	1	1	1	1
Format	16	16	16	16	16	16	16	16	16
CRC En	0	1	0	1	0	1	0	1	0
VC Num	0	0	0	0	0	0	0	0	0
Hsync Start	1	0	0	0	0	0	0	0	0
DMA Request	0	0	0	0	0	0	0	0	0
Data Size	8	2048	132	132	72	8	132	132	72
LPHDR Enable	1	1	1	1	1	1	1	1	1
Hsync End	0	0	0	0	0	0	0	0	0

Figure 18. Linked List Configuration - Multiple Packets

4.1.3 Simplified View of the FSM

A simplified view of the CBUFF state machine is as depicted in Figure 19.

Once configured and out of reset:

- Step 1. The state machine is awaiting the reception of a frame start.
- Step 2. On a frame start (DFE or software), a VSync start is triggered and the state transitions to awaiting a "Chirp Available" event.
- Step 3. For the "number of chirps" configured:
  - Every "Chirp Available" event trigger, the CBUFF state machine always runs through the valid linked list chain sending out control and data packets and in the process also handshaking and receiving data over the eDMA.
  - At the end of transfer of every linked list chain per chirp, a "chirp done" interrupt is generated to R4F and DSP.
- Step 4. At the end of all "number of chirps" loops and transfers, a VSync end is triggered.
  - A "frame done" interrupt is generated to R4F and DSP.
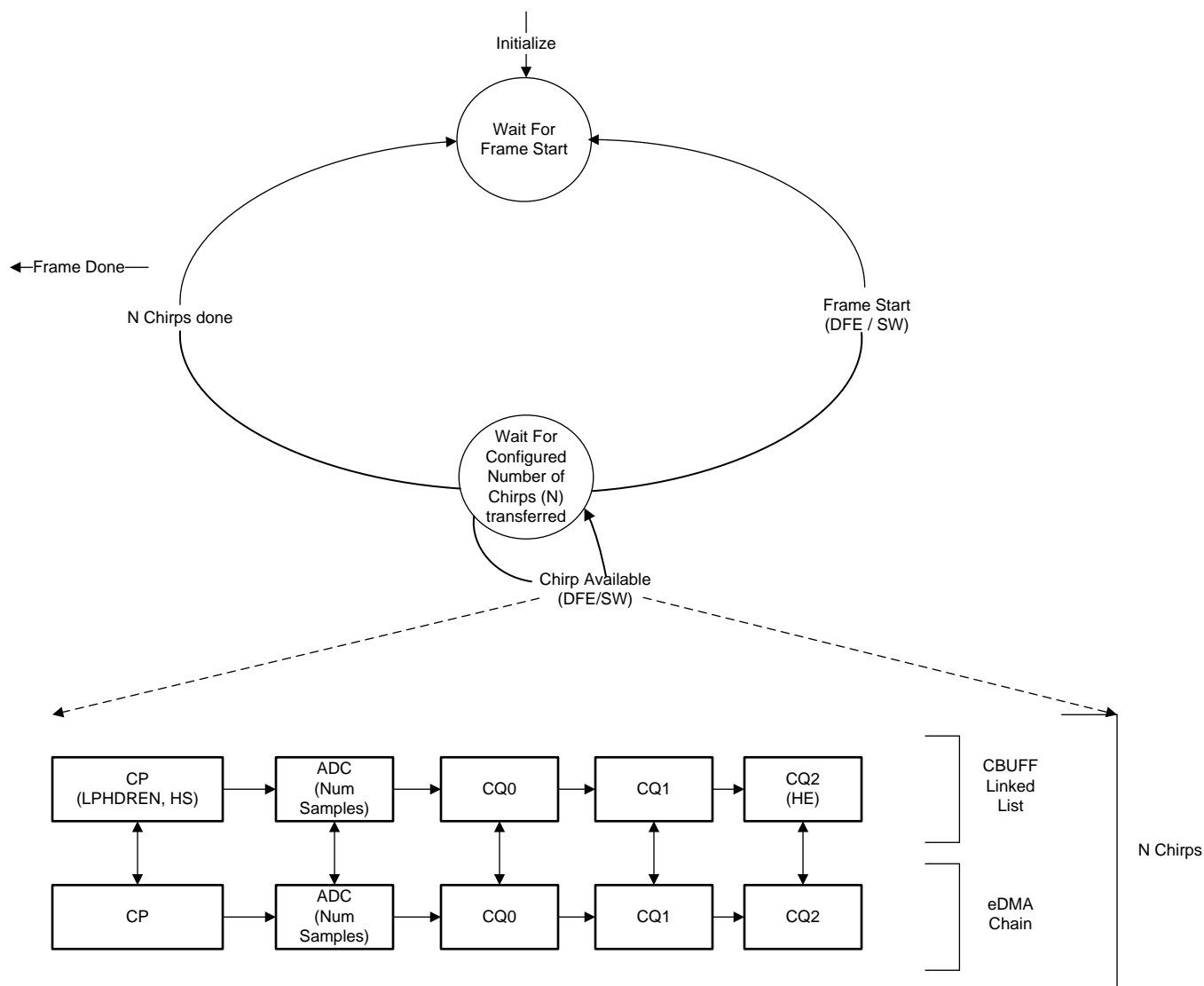- Step 5. Go to step 1.

Figure 19. Simplified View of CBUFF FSM

4.1.3.1   *Reconfiguration @ Chirp Periodicity*

Figure 20 captures the timing events that include the input triggers (Frame start and chirp available) and output responses from the CBUFF (chirp done and frame done) indicating the completion of the transfers over HSI.

Also captured is a need for reconfiguration at a frame boundary due to changes in the chirping scheme to be applied.
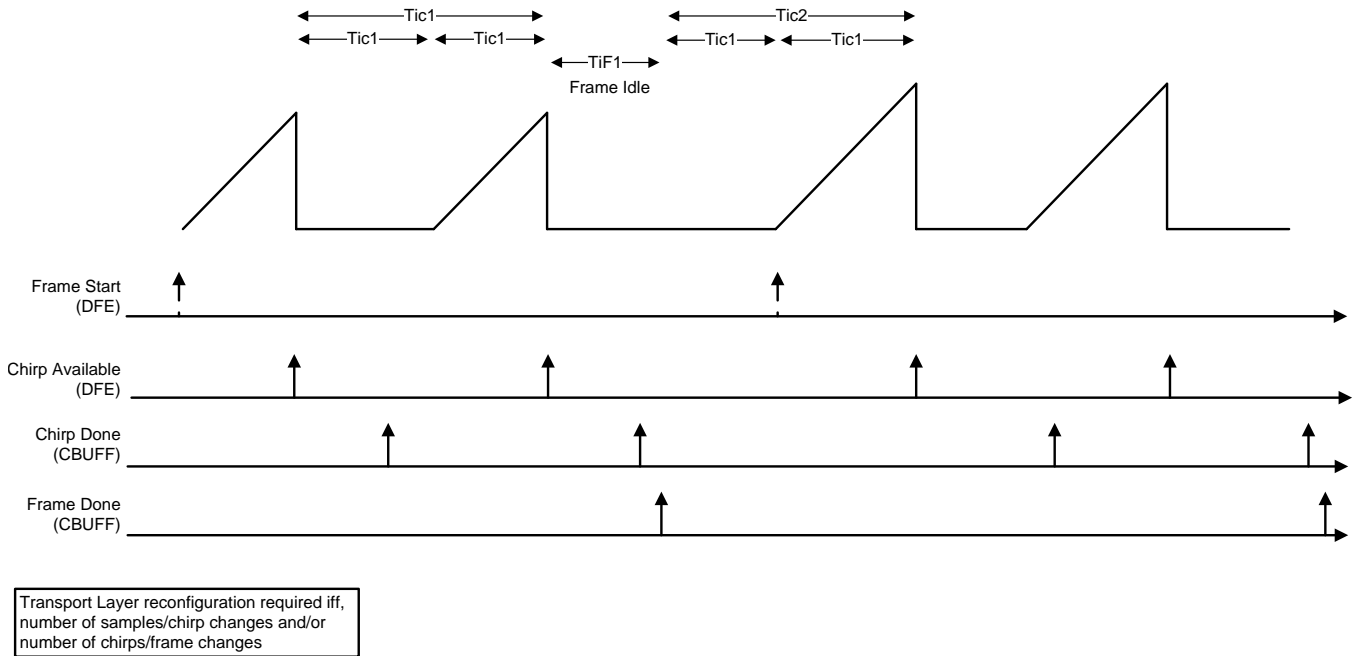

Figure 20. Timing Diagram of Events

If the data packets being sent out at the chirp periodicity is the same and the chirping scheme varies, a reconfiguration is required if:

- Number of samples and chirp changes
- Number of chirps and frame changes

Referring to Figure 14, the key fields that need to be updated (highlighted in bold) are the "number of chirps" configured and the "number of samples" in the linked list node and eDMA parameter set (for ADC buffer).

In addition, the "Long packet payload size" also needs an update in case of CSI2.

Copyright © 2017, Texas Instruments Incorporated

4.1.4 LVDS Configurations

4.1.4.1 General Configurations

The key LVDS configurations include:

- Enabling the lanes (up to four lanes in AXR12xx/AWR14xx and two lanes in AWR16xx
- Selection of the clocking mode: single data rate, double data rate
- Selection of MSB first or LSB first
- Selection of CRC enable at the end of every LVDS frame. This choice is a global choice and applies for all the LVDS frames transferred on the lanes.
- Enable of 3 channel 3 lanes mode of operation

Table 7. LVDS General Configurations

Register Name	Register Field	Description
CFG_LVDS_GEN_0	CFG_LVDS_LANE0_EN	LVDS only programming: 0: LVDS lane 0 is disabled 1: LVDS lane 0 is enabled
	CFG_LVDS_LANE1_EN	LVDS only programming: 0: LVDS lane 0 is disabled 1: LVDS lane 0 is enabled
	CFG_LVDS_LANE2_EN	LVDS only programming: 0: LVDS lane 0 is disabled 1: LVDS lane 0 is enabled
	CFG_LVDS_LANE3_EN	LVDS only programming: 0: LVDS lane 0 is disabled 1: LVDS lane 0 is enabled
	CFG_BIT_CLK_MODE	Bit clock mode: 0: SDR clocking mode 1: DDR clocking mode
	cmsbf	1: Data is sent out on the LVDS lane MSB first 0: Data is sent out on the LVDS lane LSB first
	cbcrcen	LVDS frame CRC: 0: CRC is not sent at the end of the LVDS frame 1: CRC is sent at the end of the LVDS frame
	cpz	LVDS clock config 1: Clock alignment enabled Others: Internal clock alignment not enabled This needs to be set to 0x1 for correct functionality
CFG_LVDS_GEN_1	c3c3l	LVDS only programming: 0: Regular operation 1: Enable 3CH-3Lane mode in LVDS.

NOTE: Set the LVDS FIFO Initial Threshold to 8.

4.1.4.2   Format Mapping

The format mapping configuration in the linked list node determines the mapping of the data from the sources received over the eDMA onto the configured lanes.

The format mapping works on mapping each of the CBUFF units (16-bit data), out of each row of memory comprising of 8 CBUFF units (128 bit), onto the different LVDS lanes. Each lane consists of two format mapping registers and the linked list node has an entry to select the format mapping to be applied for the transfer.

Example 1 – Non-interleaved mode (AWR16xx usecase)

Figure 21 depicts the transfer of the ADC data stored in non-interleaved mode in the ADC buffer being mapped onto the available two LVDS lanes in AWR16xx devices.

The eight CBUFF units, from one row in memory, are mapped to the two lanes. There are two example mappings shown:

- Mapping IQ samples together and then the IQ pairs are interleaved over the two LVDS channels
- Mapping all the I samples on one lanes and all the Q samples on another lane
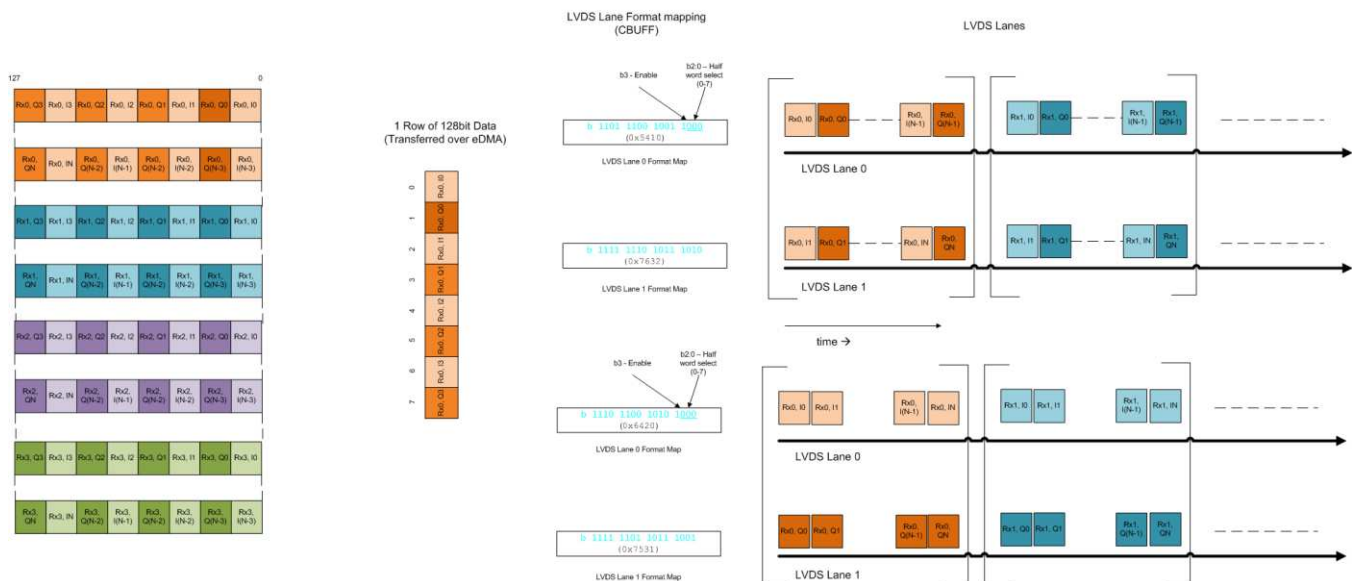
Figure 21. LVDS Format Mapping - Non Interleaved Mode

Example 2 – Interleaved Mode (AWR12xx/AWR14xx usecase)

Figure 22 depicts the transfer of the ADC data stored in interleaved mode in the ADC buffer being mapped on to the available four LVDS lanes in AWR12xx/AWR14xx devices.

The format mapping configuration maps two CBUFF units each (out of the 8 in a row) onto each of the four LVDS lanes. In turn this selection maps the data corresponding to a particular receive channel (Rx) onto a particular LVDS lane.
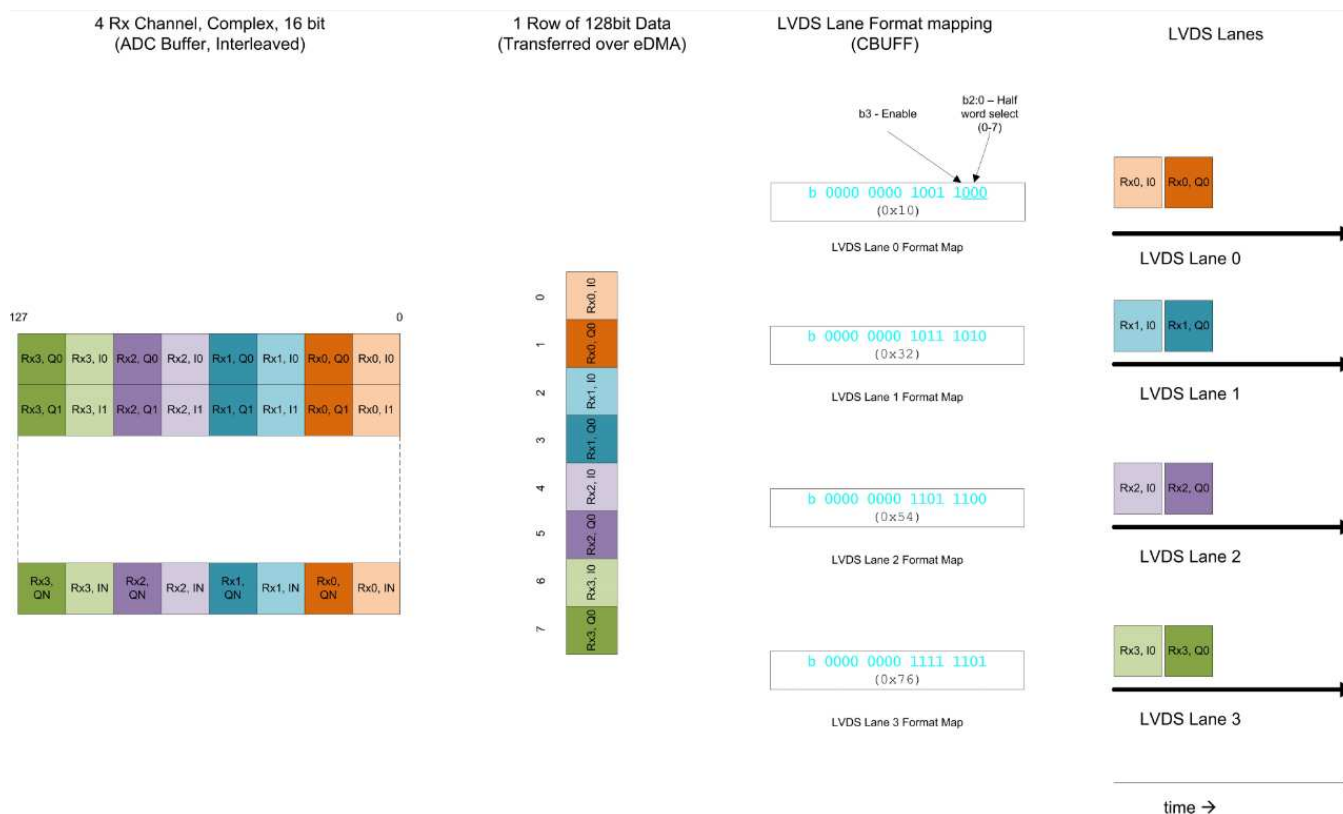
Figure 22. LVDS Format Mapping - Interleaved Mode

Register Description

The registers to be programmed for the format mapping are listed below. Each of the fields (nibbles) marked from _A to _H represent the possibility of mapping up to eight CBUFF units on one lane. Depending on the number of lanes configured, the number of valid entries (determined by a validity field) would scale down.

The selection of the format map to be used out of the two options per lane (CFG_LVDS_MAPPING_LANE[X]_FMT0 or CFG_LVDS_MAPPING_LANE[X]_FMT1), is made in the linked list node.

Table 8. LVDS Lane Mapping Configuration

Register Name	Register Field	Description
CFG_LVDS_MAPPING LANE[03-]_FMT_[0-1]	CFG_LVDS_MAPPING_ LANE[03-]_FMT_[0-1]_A	Lane [0-3] mapping if format [0-1] is selected Bit [2-0]: 0-7: Selected the CBUFF unit from the eight CBUFF units to be sent on Lane 0 Bit 3 0: Entry is not valid 1: Entry is valid
	CFG_LVDS_MAPPING_ LANE[03-]_FMT_[0-1]_B	Lane [0-3] mapping if format [0-1] is selected Bit [2-0]: 0-7: Selected the CBUFF unit from the eight CBUFF units to be sent on Lane 0 Bit 3 0: Entry is not valid 1: Entry is valid
	CFG_LVDS_MAPPING_ LANE[03-]_FMT_[0-1]_C	Lane [0-3] mapping if format [0-1] is selected Bit [2-0]: 0-7: Selected the CBUFF unit from the eight CBUFF units to be sent on Lane 0 Bit 3 0: Entry is not valid 1: Entry is valid
	CFG_LVDS_MAPPING_ LANE[03-]_FMT_[0-1]_D	Lane [0-3] mapping if format [0-1] is selected Bit [2-0]: 0-7: Selected the CBUFF unit from the eight CBUFF units to be sent on Lane 0 Bit 3 0: Entry is not valid 1: Entry is valid

Table 8. LVDS Lane Mapping Configuration (continued)

Register Name	Register Field	Description
	CFG_LVDS_MAPPING_ LANE[03-]_FMT_[0-1]_E	Lane [0-3] mapping if format [0-1] is selected Bit [2-0]: 0-7: Selected the CBUFF unit from the eight CBUFF units to be sent on Lane 0 Bit 3 0: Entry is not valid 1: Entry is valid
	CFG_LVDS_MAPPING_ LANE[03-]_FMT_[0-1]_F	Lane [0-3] mapping if format [0-1] is selected Bit [2-0]: 0-7: Selected the CBUFF unit from the eight CBUFF units to be sent on Lane 0 Bit 3 0: Entry is not valid 1: Entry is valid
	CFG_LVDS_MAPPING_ LANE[03-]_FMT_[0-1]_G	Lane [0-3] mapping if format [0-1] is selected Bit [2-0]: 0-7: Selected the CBUFF unit from the eight CBUFF units to be sent on Lane 0 Bit 3 0: Entry is not valid 1: Entry is valid
	CFG_LVDS_MAPPING_ LANE[03-]_FMT_[0-1]_H	Lane [0-3] mapping if format [0-1] is selected Bit [2-0]: 0-7: Selected the CBUFF unit from the eight CBUFF units to be sent on Lane 0 Bit 3 0: Entry is not valid 1: Entry is valid

4.1.5 CSI2 Configurations (AWR12xx specific)

4.1.5.1 Virtual Channel Mapping

The sending of the CSI2 data packets on the different virtual channels (0 to 3) is enabled by performing two sets of configurations:

- Configuring sending the "VSync Start" and "VSync End" packets on the virtual channel. This is done by programming the fields of the register described below.

Table 9. Virtual Channel Configuration

Register Name	Register Field	Description
CONFIG_REG_0	cvc0en	CSI2 only programming:
		0: No VSync packet is sent at frame boundary
		1: A VSync start packet on virtual channel 0 is generated at beginning of frame
		2: A VSync end packet on virtual channel 0 is generated at end of frame
		3: A VSync start packet on virtual channel 0 is generated at beginning of frame. A VSync end packet on virtual channel 0 is generated at end of frame
	cv1en	CSI2 only programming:
		0: No VSync packet is sent at frame boundary
		1: A VSync start packet on virtual channel 1 is generated at beginning of frame
		2: A VSync end packet on virtual channel 1 is generated at end of frame
		3: A VSync start packet on virtual channel 1 is generated at beginning of frame. A VSync end packet on virtual channel 1 is generated at end of frame
	cv2en	CSI2 only programming:
		0: No VSync packet is sent at frame boundary
		1: A VSync start packet on virtual channel 2 is generated at beginning of frame
		2: A VSync end packet on virtual channel 2 is generated at end of frame
		3: A VSync start packet on virtual channel 2 is generated at beginning of frame. A VSync end packet on virtual channel 2 is generated at end of frame
	cv3en	CSI2 only programming:
		0: No VSync packet is sent at frame boundary
		1: A VSync start packet on virtual channel 3 is generated at beginning of frame
		2: A VSync end packet on virtual channel 3 is generated at end of frame
		3: A VSync start packet on virtual channel 3 is generated at beginning of frame. A VSync end packet on virtual channel 3 is generated at end of frame

- Mapping each linked list node entry to a virtual channel. Each linked list node has an entry to associate the transfer of the data corresponding to that entry to a virtual channel.

4.1.6 CRC

There are two following configurations of the CRC meant to address different objectives of ensuring data integrity:

- Data Integrity of ADC data (ADC Buffer to CBUFF transfer)

  As a safety mechanism, to ensure integrity of data transfer from ADC buffer to CBUFF, there is a CRC computed and check for integrity in hardware at both the source (ADC buffer) and destination (CBUFF).

  Configuration: This data integrity check is enabled by setting a field in the CBUFF linked list configuration register.

  > **NOTE:** This feature is applicable only for linked list entries for ADC buffer transfers. In case of non-interleaved mode, if this feature has to be enabled, ensure that there is a linked list entry for each Rx channel. This ensures that the generation and check at both the ADC buffer and CBUFF are in sync.

- Data integrity on the lanes
  - For CSI2, the protocol itself has an associated CRC on the data transferred and hence checks for integrity.
  - For LVDS, the CRC on the lanes have to be explicitly enabled. This is done by performing the following two configurations:
    - Setting CRC enable in CFG_LVDS_GEN_0 (see Section 4.1.4.1)
    - Setting the CFG_CMD_HSVAL and CFG_CMD_HEVAL as specified in Section 4.1.2.2

4.1.7 Clock Configuration

The high speed interface clock is setup by invoking an API - rlDeviceSetHsiConfig of the mmWaveLink library. This in turn translates to the invocation of AR_HIGHSPEEDINTFCLK_CONF_SET_SB API to the BIST Sub-system. For further details, see the XWR1xx_Radar_Interface_Control (available as a part of the DFP package (http://www.ti.com/tool/mmwave-dfp).

5 Usecase Examples

5.1 Packet Formats (AWR12xx)

The packet formats supported on AWR12xx devices is shown below and is selected by issuing the data path configuration API.

Packet 0

- On CSI2, the data can be RAW8/RAW12/RAW14 format and sent on any of the four configured virtual channels.
- On LVDS, the data format can be 12/14/16 bit and are mapped onto the different lanes based on a format mapping selection done (using LVDS configuration API).

PACKET 0 – Can be RAW8/RAW12/RAW14, Virtual Channel 0/1/2/3

Figure 23. Packet 0 - Configurable Formats

Packet 1

- Packet 1 is always sent out as 16 bit data. This is because the CQ data is natively 16 bit information and the receiver need not perform the packing.
- On CSI2 lanes, this information is sent as RAW8 information always but can be on a different virtual channel if different from the Packet 0.

PACKET 1 – Always RAW8, Virtual Channel 0/1/2/3
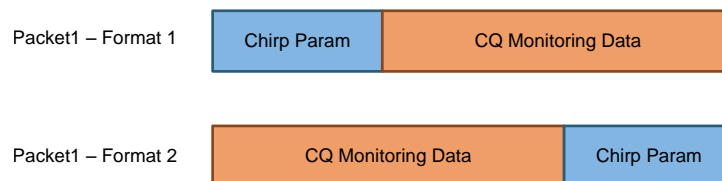
Figure 24. Packet 1 - Configurable Formats

5.2 Transfer Periodicity – Chirp(s) Available (Ping Pong switch) Aligned

In the use case described in this section, the transfer periodicity is assumed to be aligned with the "Chirp(s) available" interrupt from the DFE, which is also indicative of the ping-pong switch and a filled buffer data being available

This event indicates the availability of all the related data corresponding to the chirp(s) that include ADC buffer, chirp parameters and chirp quality data. Operating the CBUFF FSM in hardware (DFE) triggered mode can in turn kick the eDMA to perform the readout of data and transfer to CBUFF FIFO, which in turn gets relayed onto the HSI interface.

The advantage of the transfer periodicity being aligned to the "Chirp(s) Available" interrupt is that, once the ADC buffer-eDMA-CBUFF configurations are setup, the transfer is performed by hardware triggers without the need for any software intervention.

In addition to transferring hardware generated data (ADC, CP, CQ), application data (from transfer buffer, RAMs) can also be transferred by extending the chain of the transfers (both in eDMA as well as CBUFF linked list) appropriately. It is the application's responsibility to ensure that the data handling is managed appropriately as the rate of the "Chirp(s) Available" will not match the application data generation. A possible approach could be to have header information in the application data that can indicate the validity and/or newer updates to the data being transferred.

Example Use Case

The example assumes the configuration to be the following. There are 6 data packets being transferred during every "Chirp(s) Available" event:

Table 10. Data Packets to be Transferred

Data packet 1	HDR1 → ADC (Rx0, 4 Chirps) → ADC (Rx1, 4 Chirps) → ADC (Rx2, 4 Chirps) → ADC (Rx3, 4 Chirps)
Data packet 2	HDR2 → CP Data (4 Rx channels, 4 Chirps)
Data packet 3	HDR3 → CQ Data (4 Chirps)
Data packet 4	HDR4 → Transfer Buffer[0] (4 KB)
Data packet 5	HDR5 → Transfer Buffer[1] (4 KB)
Data packet 6	HDR6 → DSP Measurement data in L3 (5 KB)

The configurations of the data path modules are:

Table 11. Data Path Configurations

ADC Buffer	Four Rx channels, non-interleaved, 2K complex samples/channel, Ping-Pong switch based on availability of four chirps
eDMA	15 eDMA channels, all 15 channels chained to transfer one after another, all 15 channels linked to restore the configuration end of every iteration
CBUFF Linked list	15 linked list (LL) entries - data packet 1 (5 LLs), data packet 2 (2 LLs), data packet 3 (2 LLs), data packet 4 (2 LLs), data packet 5 (2 LLs), data packet 6 (2 LLs)
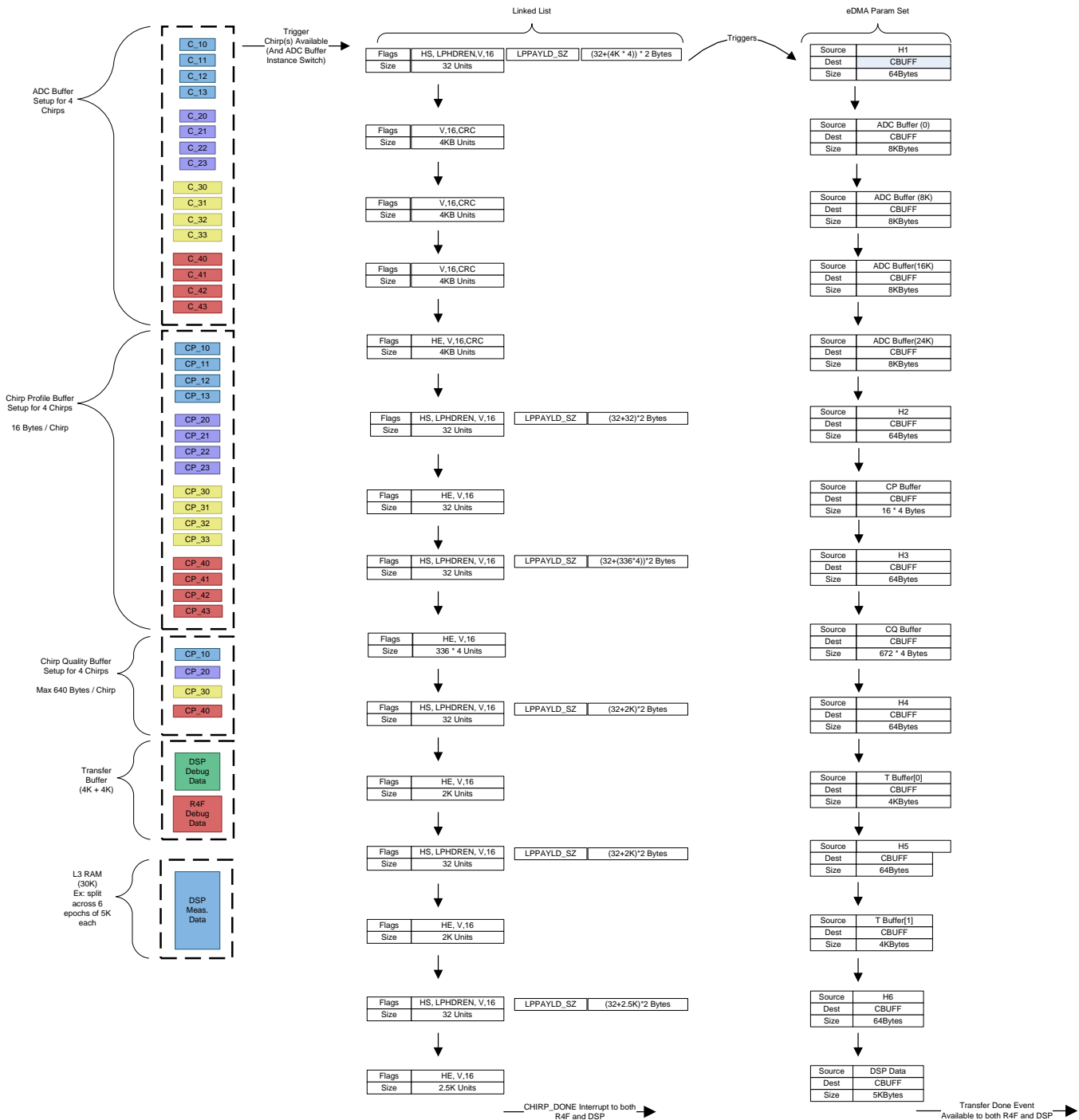
Figure 25. Data Transfer @ Chirp Periodicity

- A packet is referred to as signified by the markers HSync Start to HSync End in the linked list. A group of linked list entries constitute a packet.

- The A3. Only the processing of the first linked list packet (Data packet 1) causes the eDMA chain to be triggered. For subsequent packets, the DMA trigger is set to 7 (meaning do not trigger DMA). This is done as the eDMA chaining itself takes care of triggering all the configured transfers for all the packets.

- DC buffer transfers of each Rx channel data is captured as a separate linked list entry to ensure the CRC computation for the data integrity check between ADC buffer and CBUFF is correct.

- The start of every packet has the LPH (Long Packet Header Enable) bit set. In addition to setting this bit, the CFG_DATA_LL[X]_LPHDR_VAL also has to be appropriately set. In case of CSI2, the Long Packet Payload size (LPPAYLD_SZ) has to be a total number of bytes to be sent in the data packet.

- The application data is assumed to be sent every "Chirp(s) Available" event.

- For the above configuration, if the transfer of data was performed over LVDS with CRC on the packets enabled, the data would appear on the lanes as depicted below. Each data packet would be appended with a CRC as delimited by the HSync Start and HSync End markers in the linked list entires.
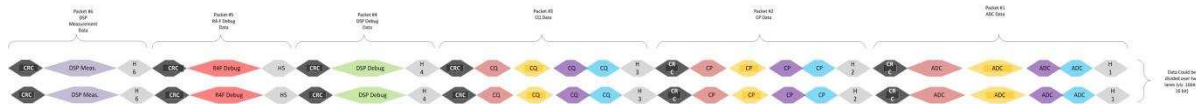
Figure 26. LVDS Data Over the Lanes for Above Configuration

5.3   *Transfer Periodicity – Non-Chirp Aligned (software-based triggering)*

There is no hardware support in the CBUFF to transmit packets at a periodicity other than when aligned to "Chirp(s) Available" event (for example, a single packet to be transferred at inter-frame boundary).

It is the responsibility of the application software to perform the necessary steps, as listed below, to send out the data over the high speed interface.

CBUFF supports software triggering of a packet by allowing for "Frame Start" and "Chirp Available" to be triggered by software.
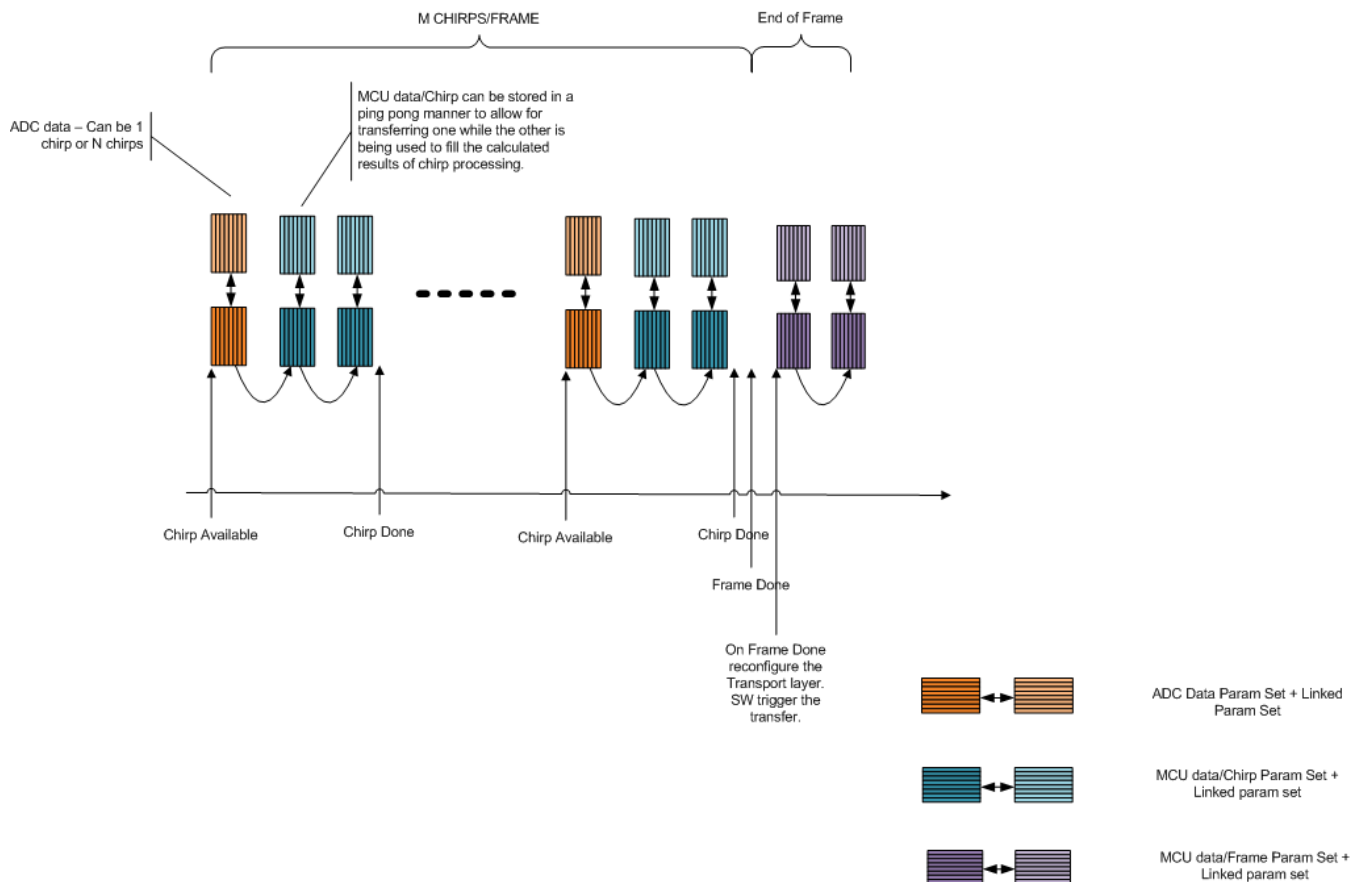
Figure 27. Transfer Periodicity – Non-Chirp Aligned

- Identify a period of inactivity on the HSI lanes
  – For example, to support transferring application data, at the inter-frame boundary, software needs to wait for CBUFF "Frame done" interrupt indicating the VE sync packet for the last frame has been sent.
- Save the existing CBUFF and eDMA configuration
  – This can also be predefined configurations that are saved and restored as per the context.
- Configure the CBUFF (linked list entries, number of chirps/frame (usually 1)) and eDMA channels parameter set appropriately for sending application data packet
  – This is setting up the new context for the transfer of application data (both in CBUFF as well as the eDMA chain).
- Trigger sending of the packet
  – This is done by first triggering a "Frame Start" followed by a "Chirp Available" in software. This would mimic the hardware operation and the event triggers for the CBUFF state machine to perform the transfer.
- Wait for Packet to be transferred over CSI-2/LVDS
  – Await the chirp done and frame done interrupts from CBUFF
- Restore CBUFF configuration so that ADC/CQ data will be sent out on the next frame start
- It is the application software responsibility to ensure it completes above sequence before next frame start interrupt from DFE

5.4 Continuous Streaming Mode

There are 2 possible use cases of the continuous streaming mode supported in the device:

- Option A: Rx path characterization

  The FMCW transceiver is configured to output a single frequency tone in the range of 76-81 GHz. Analysis of the receive path involves feeding a tone via a waveguide interface, with an offset for evaluation, to analyze the Rx channel data that is continuously captured in the ADC buffer and transmitted out over the HSI interface.

- Option B: Sanity test of the Data path

  To test the connectivity of the HSI interface (LVDS/CSI2) and to ensure the correct configuration of the data path from (ADC Buffer → eDMA → CBUFF → HSI interface), a test pattern can be configured to be generated in the ADC buffer that is then transferred over the HSI interface to be tested for correctness of the reception at the receiver.

  When configured in the continuous streaming mode, the ADC buffer supports a ping-pong switch based on a threshold value set based on the number of ADC samples filled in the ADC buffer (in contrast to the ping-pong switch based on the number of chirps available). The ADC samples as received from the configured Rx channel is continuously filled in the ADC buffer until the threshold is reached. When the threshold is reached, a ping-pong switch occurs that in turn triggers the CBUFF-eDMA chain to transfer the data out over the HSI interface.

  Usage of the continuous streaming mode involves the steps discussed in the following sections.

5.4.1 Continuous Streaming Mode Setup

Use the following sequence to setup the data path to operate in the continuous streaming mode:

1. Configure the eDMA and CBUFF linked list to transfer the desired number of samples "N" per (Rx channel) per (ping-pong switch).
2. Set the "ChirpsPerFrame" field to 0xFFFFFFFF (see Section 4.1.2.3) in CBUFF to perform the transfers infinitely.
3. Enable continuous streaming mode in ADC buffer.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG1	ADCBUFCONTMODEEN	0: Continuous mode disable for ADC Buffer. 1: Continuous mode enable for ADC Buffer.

4. In case of continuous streaming mode, as there is no chirping, the "frame start" required for the CBUFF state machine operation, has to be enabled in software (see Section 4.1.2.1).
5. Setup the threshold "N" per Rx channel to cause a ping-pong switch.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG4	ADCBUFSAMPCNT	No of samples to store in each Ping and Pong register in continuous mode of ADC Buffer. In real only mode this refers to the number of real samples and in complex mode, this refers to number of complex samples. This refers to the number of samples per channel. The internal counter increments once for every new sample from DFE (as long as 1 or more channels are enabled). The max allowed value varies depending on other configurations (No of channels enabled and real/complex data). Continuous mode is expected to be only used for Rx Characterization and ADC Buffer Testpattern mode.

6. Start the continuous streaming mode.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG1	ADCBUFCONTSTRTPL	Start Pulse for Continuous mode. The data capture will start from Address 0 once this register is set. All the other configurations like Enable, Sample Count are expected to be programmed before this pulse. Continuous mode is expected to be only used for Rx Characterization and ADC Buffer Testpattern mode.

5.4.2 Setting up of the Input Source

Option A: Rx path characterization

In addition to setting up the waveguide connectivity to the Rx path, there are APIs to be issued to the Radar Subsystem (RadarSS) setup the continuous streaming mode. The key API to be invoked is "rlSetContModeConfig".

For the sequence of API invocation, see the *Continuous streaming mode* section in the *AWR1xx Radar Interface Control Reference Guide* (SWRU431) (available as a part of the DFP package (http://www.ti.com/tool/mmwave-dfp)).

Option B: Sanity test of the Data path

The configuration of the test pattern involves setting up the following registers:

Register Name	Register Field	Description
DSS_REG:TESTPATTERNRX[1-4]ICFG, TESTPATTERNRX[1-4]QCFG	TSTPATRX[1-4]IOFFSET, TSTPATRX[1-4]QOFFSET	Offset value to be used for the first sample for the test pattern data in I-Q channel Rx channel [0-3]. In this register the naming convention for the 4 Rx channel indices are from 1 to 4 instead of 0 to 3.

Register Name	Register Field	Description
DSS_REG:TESTPATTERNRX[1-4]ICFG, TESTPATTERNRX[1-4]QCFG	TSTPATRX[1-4]IINCR, TSTPATRX[1-4]QINCR	Value to be added for each successive sample for the test pattern data in I-Q channel Rx channel [0-3]. In this register the naming convention for the 4 Rx channel indices are from 1 to 4 instead of 0 to 3.

The configurations setup the start offset and the increment to the test pattern value to be filled in the ADC buffer.

5.4.3 Trigger the Start of Filling Data

Option A: Rx path characterization

To start the RadarSS to receive the Rx data in continuous streaming mode, issue the "rlEnableContMode" API. The ADC data will start filling into the ADC buffer and whenever the threshold set for each Rx is reached a ping-pong switch occurs and the data is transferred out over the HSI lanes.

Option B: Sanity test of the Data path

For the HSI operation, ensure that the "rlDeviceSetHsiClk" API is issued to the RadarSS.

In order to start the test pattern generation, the configurations shown below have to be performed:

Register Name	Register Field	Description
DSS_REG:TESTPATTERNVLDCFG	TSTPATGENEN	Enable for test pattern generator. This is used to Mux with the functional data from DFE. 000 → Disable, 111 → Enable, Others are reserved.

Register Name	Register Field	Description
DSS_REG:TESTPATTERNVLDCFG	TSTPATVLDCNT	Number of DSS Interconnect clocks (200 MHz) between successive samples for the test pattern gen.

Once TSTPATGENEN is set, the ADC buffer is filled with the configured pattern, at a rate as determined by the TSTPATVLDCNT configuration.

Every time the configured threshold for the ping-pong switch is reached, the transfer of the ADC buffer contents (test pattern) happen over the HSI interface.

5.4.4 Stopping the Continuous Streaming Mode

The continuous streaming mode is disabled by following the steps below:

Option A: Rx path characterization:

Issue the "rlEnableContMode(disable)" API to RadarSS to stop the continuous streaming mode.

Option B: Sanity test of the Data path

Disable the test pattern generator by clearing the DSS_REG:TESTPATTERNVLDCFG:TSTPATGENEN field (see Section 5.4.3Section 1.1.3)

Once the above settings are performed, disable the continuous streaming mode in the ADC buffer by setting the below field.

Register Name	Register Field	Description
DSS_REG:ADCBUFCFG1	ADCBUFCONTSTOPPL	Stop Pulse for Continuous mode. The data capture will stop once this register is set. Continuous mode is expected to be only used for Rx Characterization and ADC Buffer Testpattern mode.

6 References

- *TMS320C6000 DSP Enhanced Direct Memory Access (EDMA) Controller Reference Guide*
- *EDMA3* wiki page