Direct Memory Access and Scatter Gather (v2.01a)

DS416 April 24, 2009 **Product Specification**

Introduction

The DMA SG service is a continuation of the Xilinx family of IBM CoreConnect compatible LogiCORE products. It provides direct memory access (DMA) allowing for a bounded number of sequential data transfers to take place between regions in the address space, typically between memory and an I/O device, without processor management of individual transfers. This service also provides Scatter Gather (SG) functionality allowing a sequence of DMA operations to be pre-specified by software and performed automatically without further processor intervention. The Xilinx DMA SG is available for use as a service within the OPB and PLB V3 Master with Local Link.

Features

- Available as a service in selected Master IPIFs
- Supports up to 2 physical channels
- Supports Split bus operation for simultaneous independent Transmit and Receive DMA operations
- Supports Simple DMA, Simple SG, and Packet SG
- Interrupt event reporting for each channel
- Supports optional Interrupt coalescing
- Supports user definable Local Link Headers and Footers for use with functions such as checksum off loading
- Supports Scatter Gather Buffer Descriptor inserting and appending on active Buffer Descriptor chains

LogiCORE™ IP Facts		
Core Specifics		
Supported Device Family	Virtex®-4	
Version of Core	dma_sg	v2.01a
Resources Used		
	Min	Max
Slices	185	1089
LUTs	176	1296
FFs	216	1160
Block RAMs	0	1
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	None	
Verification	EDK Simulation Support	
Instantiation Template	Wizard Support	
Design Tool Requirements		
Xilinx Implementation Tools	Xilinx EDK 1.1	
Verification	Mentor Graphics ® ModelSim® 6.4be or later	
Simulation	ModelSim 6.4bor later	

Functional Description

The block diagram in Figure 1 shows an overview of the DMA SG, while a detailed block diagram of the DMA SG is shown in Figure 2.

The DMA SG System is partitioned into several functional blocks which are shown in Figure 2 and described in the subsequent sections.

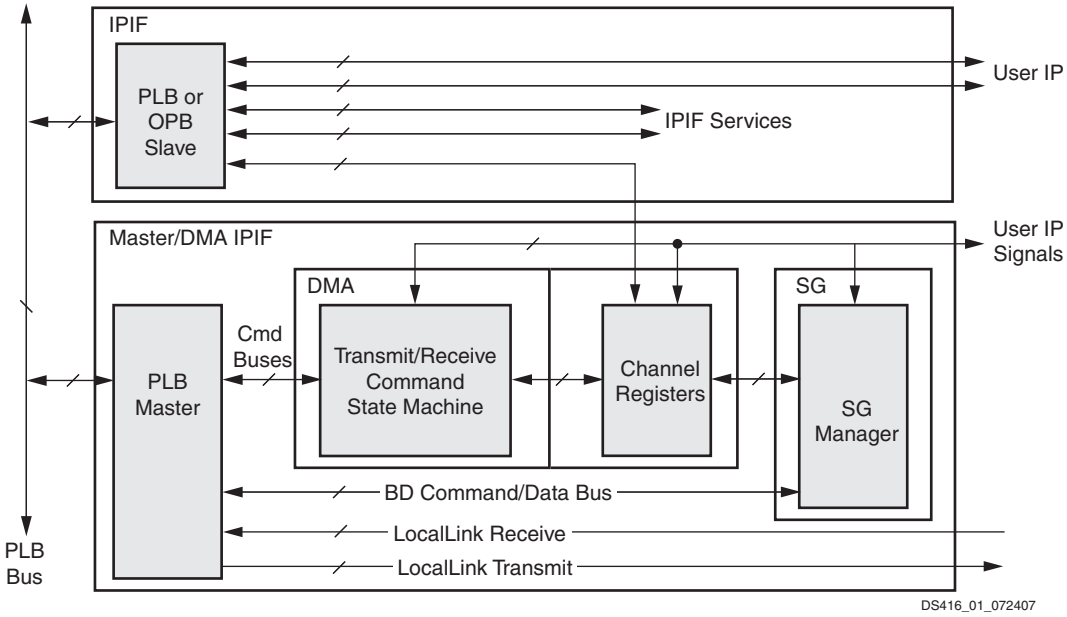

Figure 1: **DMA SG Block Diagram**

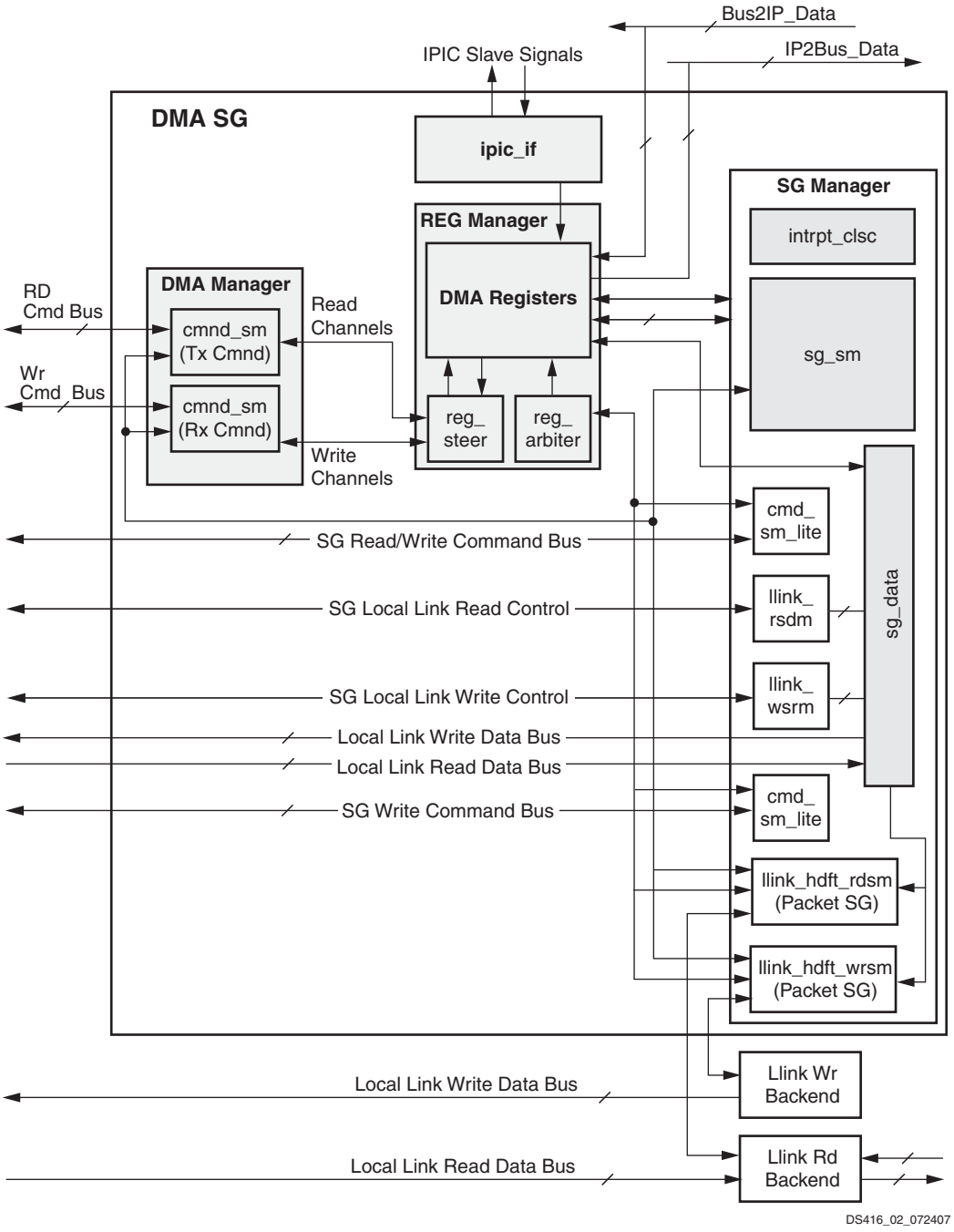

Figure 2: **DMA SG System Detail Block Diagram**

DMA Manager Module: Handles interfacing with the IPIF Master to perform DMA operations, as well as controlling two command buses with each bus operating independently from the other.

DMA Registers Module: Contains all the registers for controlling and presenting status of all DMA operations. This block also contains the Event registers for capturing DMA and SG events and for generating interrupts to the IPIF.

SG Manager Module: Manages fetching and updating Buffer Descriptors (BD), as well as updating and monitoring Channel Registers. The SG Manager maintains its own independent command bus between itself and the IPIF Master. This allows the IPIF Master to properly arbitrate between performing DMA Read and Write operations and performing BD Fetch and Update operations. If there are no channels configured for SG operation then the SG manager is optimized out of the DMA SG logic reducing FPGA resource utilization, See Table 14.

To perform SG operations, the SG Manager utilizes a dedicated command and local link interface for fetching and updating buffer descriptors.

Simple DMA Operation

When used under programmed IO, a DMA operation for a channel is set up and started by writing values into the following DMA registers.

- DMACR - DMA Control Register
  - AINC - Address Increment. If the DMA Address is a "keyhole" (1) register, then AINC is set to 0. If the DMA Address should increment for each byte transferred, AINC is set to 1.
  - Device Select - Specifies which channel this DMA cycle is associated with. This will drive the appropriate DMA Device Select, DMA2IP_Device_Sel, bits to the User IP.
  - Type - Type specifies the type of the DMA transfer.
    - 000 = Reserved
    - 001 = Bounded Fixed Length Burst (Length / Master parameterization sets burst length) - EOP Terminates Burst
    - 010 = Bounded Indeterminate Burst with single termination (EOP terminates burst)
    - 011 = Reserved
    - 100 = Reserved
    - 101 = Reserved
    - 110 = Reserved
    - 111 = Reserved
  - DSize - DSize specifies the data size of the DMA transfer.
    - 000 = byte (1 byte)
    - 001 = half-word (2 bytes)
    - 010 = word (4 bytes)
    - 011 = double-word (8 bytes)
    - 100 = reserved
    - 101 = reserved
    - 110 = reserved
    - 111 = reserved
- MSBA, LSBA- Most Significant and Least Significant Bus Address. The bus address for the transfer is written to the Bus Address registers.
- LENGTH - Length Register. The number of bytes to transfer. Writing of the register is the event that starts the DMA operation, so it must be done last(2).

1. A keyhole register is a single address associated with a sequence of values. An example is a FIFO.
2. The other registers can be written, if they need to change, in any order.

The status of the DMA operation is available in the DMA Status Register, DMASR. The DMABSY bit becomes a 1 when the DMA engine starts to process the channel, which may be delayed from the time that LENGTH register is written. Because of the delay, the recommended way to poll for completion is to wait for the DMADONE bit to become a 1, rather than to wait for DMABSY to become 0, because a false completion could be detected before DMA even starts. Alternatively, interrupts on DMA Done or DMA Error can be used to indicate completion. The DMA Bus Error (DBE), and DMA Bus Time-out (DBT), status bits indicate errors.

Scatter Gather Buffer Descriptors

Scatter Gather requires a common memory-resident data structure that holds the list of DMA operations to be performed. This list is shared by software and the SG hardware. The Buffer Descriptor shown in Table 1 is the basis for organizing the DMA operations as a linked list. Buffer descriptor are fetched through the SG Command and Local Link Data interfaces. A similar mechanism is used for performing Buffer Descriptor updates, which are done for SG Packet channels.

Table 1: **Buffer Descriptor**

Buffer	Name	Description
DMASR	DMA Status Register	Status results for the DMA transfer described by this Buffer Descriptor.
DMACR	DMA Control Register	Control bits for controlling the DMA transfer described by this Buffer Descriptor.
MSBA	Most Significant Bus Address	Specifies the most significant starting address of the Buffer to which this buffer descriptor describes. Used only for 36 bit address buses, ignored otherwise.
LSBA	Least Significant Bus Address	Specifies the least significant starting address of the Buffer to which this buffer descriptor describes.
BDA	Buffer Descriptor Address (Link to next Buffer Descriptor)	Link pointer, specifying to the DMA Controller the address of the next Buffer Descriptor to fetch.
LENGTH	Length	Specifies number of bytes in buffer to transmit or specifies available bytes in buffer for receive. For Packet SG Write (Receive) after the Buffer Descriptor is updated, will specify number of payload bytes received.
SR	Packet Status Register	Updated with first word of Local Link Footer for Packet SG Write (Receive) channels. Field used to give status of receive packet and is user defined.
RSVD	Reserved For Future Growth	
USR0	User Field 0	Inserted into Local Link Header for Packet SG Read (Transmit). Stripped and stored from Local Link Header for Packet SG Write (Receive) Unused for all other channel types
USR1	User Field 1	Inserted into Local Link Header for Packet SG Read (Transmit). Stripped and stored from Local Link Header for Packet SG Write (Receive) Unused for all other channel types
USR2	User Field 2	Inserted into Local Link Header for Packet SG Read (Transmit). Stripped and stored from Local Link Header for Packet SG Write (Receive) Unused for all other channel types

Table 1: **Buffer Descriptor *(Cont'd)***

Buffer	Name	Description
USR3	User Field 3	Inserted into Local Link Header for Packet SG Read (Transmit). Stripped and stored from Local Link Header for Packet SG Write (Receive) Unused for all other channel types
USR4	User Field 4	Inserted into Local Link Footer for Packet SG Read (Transmit). Unused for all other channel types
USR5	User Field 5	Inserted into Local Link Footer for Packet SG Read (Transmit). Stripped and stored from Local Link Footer for Packet SG Write (Receive) Unused for all other channel types

Each field of the Buffer Descriptor is four bytes in length and corresponds to either one of the DMA SG registers or, in one case, to a Device Status Register. SG automation is the process of initializing a DMA operation from the fields of the buffer descriptor, starting the operation, recording the completion status to the Buffer Descriptor, and then following the link to the next buffer descriptor and repeating the process, which is described in more detail in the next section.

The SR field serves to report the status of the packet reception and is only valid for write channels i.e. receive (type 3). The definition of the value is user-determined. The status value is obtained from the footer of the Receive Local Link data stream. (See <RD Red>"Stopping and Starting SG Operation" on page 12 for more information about Local Link Headers and Footers). If a receive footer does not exists (i.e. C_WR_FTR_SIZE = 0) then the SR field is set to a value of zero. For read channels, i.e. transmit (type 2), the SR field will retain what was stored there by the software application.

Figure 3 shows Buffer Descriptors organized into a linked list. Scatter Gather hardware will successively perform the DMA operations specified in the descriptors up to and including the descriptor with SGS=1 (all others in the figure are assumed to have SGS=0).

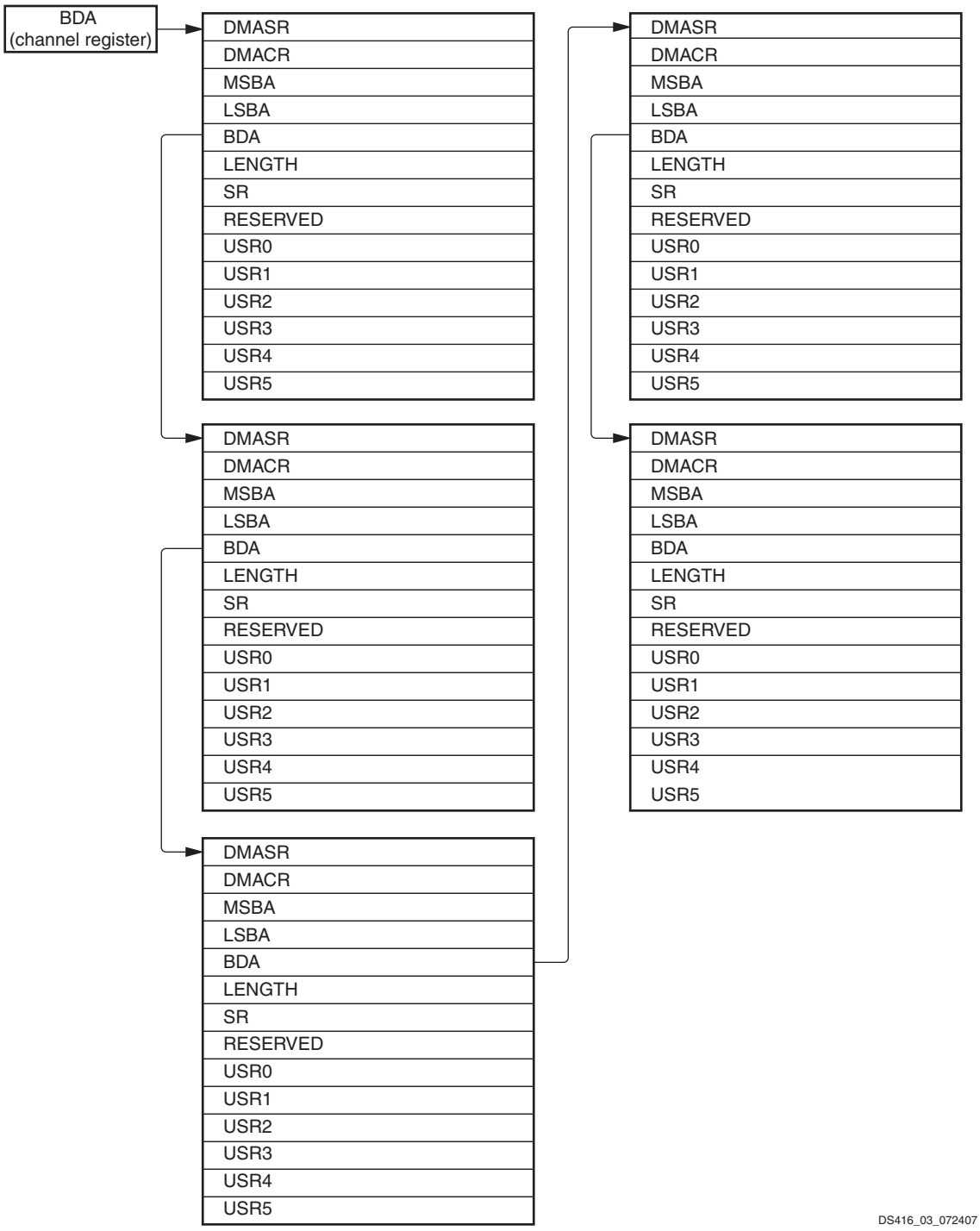

Figure 3: **Linked List of Buffer Descriptors**

Figure 4 shows Buffer Descriptors organized into a buffer ring. The buffer ring is for a channel of type 2 as evidenced by LAST=1 tags (other Buffer Descriptors are assumed in the figure to have LAST=0). Note that some packets are specified by a single Buffer Descriptor and others by more than one consecutive Buffer Descriptor. The last ready packet is marked with SGS=1, giving a sentinel position in the

ring. Also note that even if Buffer Descriptors are contiguously allocated, they are required to be linked through the BDA field.

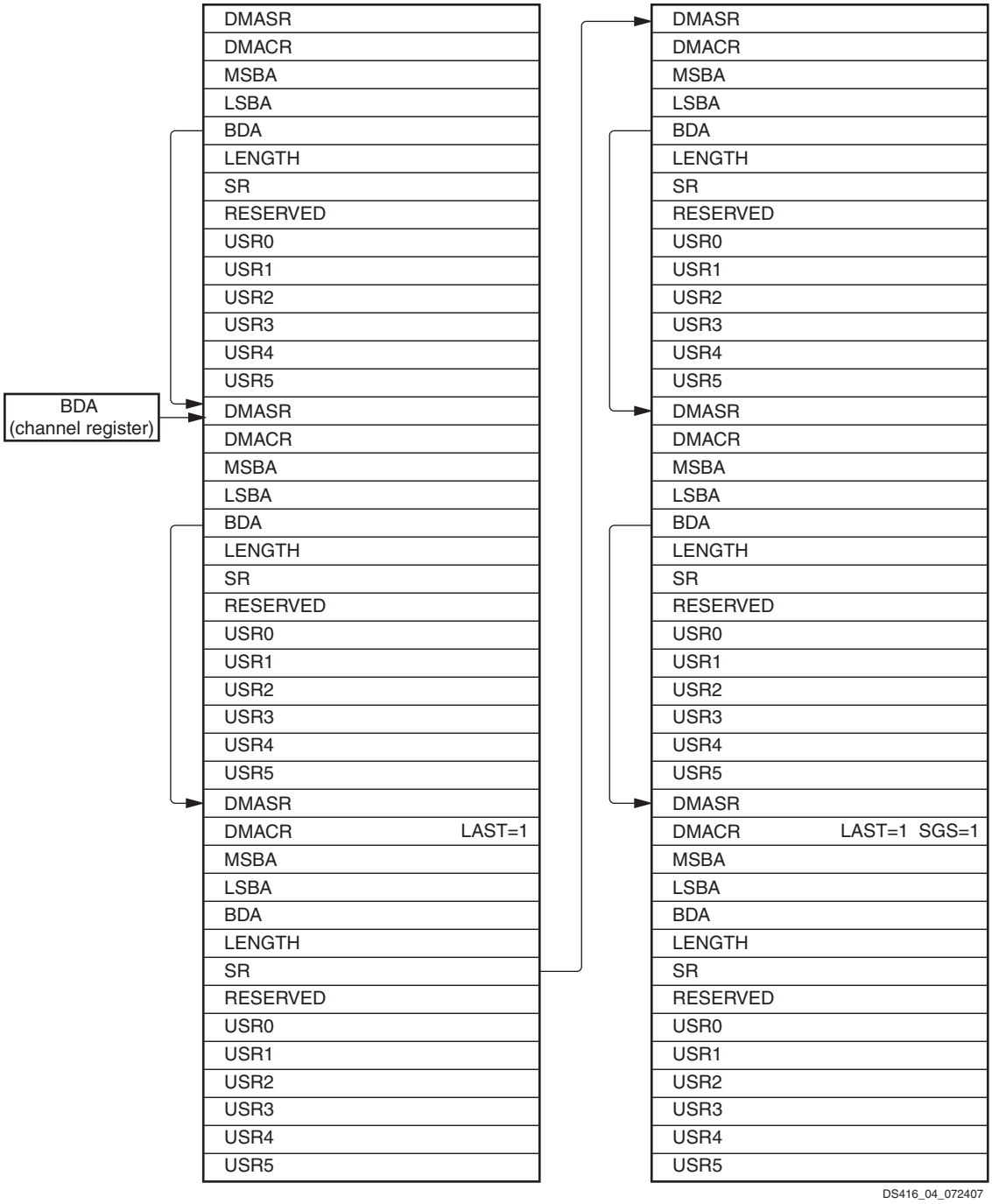

Figure 4: **Buffer Descriptors Organized into a Buffer Ring**

Simple Scatter Gather Operation

When Scatter Gather is enabled (SWCR.SGE=1, SWCR.SGD = 0, and DMACR.SGS=0), the SG automation will process the list of buffers referenced by the channel's Buffer Descriptor Address register

(BDA). This processing will continue until it is either synchronously stopped by reaching a Buffer Descriptor with SGS=1 or asynchronously stopped by a write of 1 to the SGD bit of the Software Control Register (SWCR). It should be noted that DMACR, BA, LENGTH, and BDA registers become "read only" to software when SWCR.SGE = 1 and DMACR.SGS=0.

The following is a list of the steps required/performed to execute a simple scatter gather operation.

1. Software creates a chain of Buffer Descriptors.
2. Software prepares the DMA Channel Registers,
   a. Write the address to the first buffer descriptor to the BDA register
   b. If utilizing a 36-bit address then write the buffer descriptor page address to the DMACR register (bits 20 to 23) .
3. Software starts SG Automation by writing a 1 to SWCR.SGE and a 0 to SWCR.SGD
4. DMA SG Hardware will request the first buffer descriptor pointed to by the BDA register.
5. Upon completion of the buffer descriptor fetch, the buffer descriptor will be transferred to the currently active channel registers. This will trigger the DMA cycle to begin.
6. If the currently fetched buffer descriptor does NOT have the SGS bit set to a 1 in the DMACR, will then fetch the next buffer descriptor in the chain.
7. This process will continue until a buffer descriptor with DMACR.SGS = 1 is fetched. DMA SG automation will stop after the completion of the buffer descriptor with DMACR.SGS = 1.
8. Note that DMACR.LAST is for Packet Scatter Gather Channels and is not used in Simple Scatter Gather.
9. At the completion of SG Automation, i.e. when DMACR.SGS=1, the DMA SG controller will stop processing buffer descriptors.

Transmit/Receive Packet Scatter Gather Operation

When packets are supported by a DMA SG channel, additional considerations apply. Firstly, each channel is associated with a direction, either transmit or receive. A packet-transmit channel passes packets from memory to a consuming device, often referred to as a DMA Read in that data is read from memory. A packet-receive channel passes packets from a producing device to memory, often referred to as a DMA Write in that data is written to memory.

Secondly each packet-SG channel handles a status value. The status value is generated by the hardware device that receives packets. The Status Register (SR) field of the packet's last Buffer Descriptor receives the status value once it has been made available by the device. The device feeds the status value to SG automation via the footer in the Local Link data stream.

Thirdly, packets need to be delineated. A Last bit (LAST) is associated with the last Buffer Descriptor for a packet(1). For packet-transmit channels, the LAST bit is in the DMA Control Register; for packet-receive channels it is in the DMA Status Register(2).

1. Because whole packets can correspond to just one Buffer Descriptor, the first Buffer Descriptor, the one that gets a valid SR value, and the last Buffer Descriptor, the one that gets the LAST bit, may be the same Buffer Descriptor.
2. For packet-transmit, the LAST value in the DMA Control Register is copied to the DMA Status Register as part of the status for the Buffer Descriptor.

Finally, packet boundaries need to be communicated between the device and the DMA SG Controller. This is accomplished differently depending on whether the channel is configured for Receive or configured for Transmit.

For receive channels, when a packet has been completely received the producing device stores the packet status in the footer of the local link data stream. From here the DMA SG Controller acquires the packet length and status and writes these values to the Length field and SR field respectively of the last Buffer Descriptor. DMA SG also sets the Last bit in the DMA Status Register (DMASR) indicating to the software that the current receive buffer as described by the Buffer Descriptor contains the last of the packet data.

For packet-transmit channels, DMA SG uses the LAST bit in the DMACR as discussed above. The DMA SG Controller will drive the ChainReq signal to the IPIF Master with a logical 1 until a Buffer Descriptor is fetched with the LAST bit set to 1. When DMA SG determines that the LAST bit is set in the DMACR then the DMA SG drives the ChainReq signal to a logical 0. When the ChainReq signal is set to a 1, the IPIF Master will chain the data payloads together on the Local Link Read Data interface and suppress the EOP and EOF to the consuming device. When ChainReq is set to a 0 for a transfer request, the IPIF Master will complete the currently requested transfer and then terminate the Local Link transfer with an EOF. Note that for both receive and transmit SG channels the device (UserIP) has the obligation to maintain byte counts.

SG Packet Transmit Channel Operation

The following is a list of the steps required/performed to execute a packet transmit operation.

1. Software creates a chain of Buffer Descriptors, specifying in the Buffer Descriptor the packet boundaries using the DMACR.LAST bit. See <RD Red>"Scatter Gather Buffer Descriptors" on page 5 and <RD Red>Figure 4 on page 8. An DMACR.SGS bit can be set to allow for a synchronous stop to SG automation. <RD Red>"Stopping and Starting SG Operation" on page 12.
2. Software prepares the DMA Channel Registers,
   a. Write the address to the first buffer descriptor to the BDA channel register
   b. If utilizing a 36-bit address then write the buffer descriptor page address to the DMACR register (bits 20 to 23).
3. Software starts SG Automation by writing a 1 to SWCR.SGE and a 0 to SWCR.SGD
4. DMA SG Hardware will request the first buffer descriptor pointed to by the BDA register.
5. Upon completion of the buffer descriptor fetch, the buffer descriptor will be transferred to the currently active channel registers. This will trigger the DMA cycle to begin.
6. If a Local Link header is defined, i.e. C_RD_HDR_SIZE = 16 then USR0, USR1, USR2, and USR3 of the first buffer descriptor will be used as the header of the current Local Link Data stream. See <RD Red>"Stopping and Starting SG Operation" on page 12 and specifically <RD Red>Figure 6 on page 16.
7. If the currently fetched buffer descriptor does NOT have the SGS bit set to a 1 in the DMACR, then the next buffer descriptor in the chain will be fetched.
8. At the completion of each buffer descriptor the channel register information will be updated to the corresponding buffer descriptor memory location. The User fields, USR0 to USR5 are not updated for transmit channels. These fields in the BD will be ignored and will remain unchanged.
9. When DMACR.LAST=1 for a fetched buffer descriptor, the Master IPIF is informed to terminate the Local Link transfer at the completion of the currently requested transfer marking the end of a packet.

10. If a Local Link footer is defined, i.e. C_RD_FTR_SIZE = 8 then USR4 and USR5 of the last buffer descriptor will be used as the footer of the current Local Link Data stream. See <RD Red>"Stopping and Starting SG Operation" on page 12 and specifically <RD Red>Figure 6 on page 16.
11. At the completion of SG Automation, i.e. when DMACR.SGS=1, the most recently completed buffer descriptor will be updated and the DMA SG controller will stop processing buffer descriptors except for the User fields as discussed in step <RD Red>[8].

SG Packet Receive Channel Operation

The following is a list of the steps required/performed to execute a packet receive operation.

1. Software creates a chain of Buffer Descriptors, specifying in the Length field of the Buffer Descriptor the size available to each buffer for receiving data.
    a. Each Buffer Descriptor describes a packet. This version of the DMA SG Controller does not support multiple buffer descriptors being used to describe a single packet. Therefore the DMASR.LAST bit in each Buffer Descriptor of the chain will be set indicating the end of a packet.
    b. The Length field in the Buffer Descriptor must specify a byte count that is large enough to hold an entire packet.
    c. A DMACR.SGS bit can be set to allow for a synchronous stop. See <RD Red>"Stopping and Starting SG Operation" on page 12.
2. Software prepares the DMA Channel Registers,
    a. Write the address to the first buffer descriptor to the BDA register
    b. If utilizing a 36-bit address then write the buffer descriptor page address to the DMACR register (bits 20 to 23).
3. Software starts SG Automation by writing a 1 to SWCR.SGE and a 0 to SWCR.SGD
4. DMA SG Hardware will request the first Buffer Descriptor pointed to by the BDA register. For receive channels the User fields are not fetched and will be updated in the BD during the BD update phase of processing.
5. Upon completion of the buffer descriptor fetch, the buffer descriptor will be transferred to the currently active channel registers. This will trigger the DMA cycle to begin.
6. If a Local Link header is defined, i.e. C_WR_HDR_SIZE = 16 then USR0, USR1, USR3, and USR3 of the first buffer descriptor will be updated with the header information received. See <RD Red>"Stopping and Starting SG Operation" on page 12.
7. The DMA Controller will make a request to the Master IPIF to process receive data. When the receive data is received from the producing device (UserIP) the Master IPIF will acknowledge the DMA engine's request.
8. When the IPIF Master indicates the end of a packet, DMA SG automation will fetch the length and status of the received packet, set DMASR.LAST=1, and update the information in the associated Buffer Descriptor. See step <RD Red>[9].
9. If a Local Link footer is defined, i.e. C_WR_FTR_SIZE = 8 then the SR and USR5 of the last Buffer Descriptor will be updated with the footer of the current Local Link Data stream. USR4 will contain a copy of the SR data. See <RD Red>"Stopping and Starting SG Operation" on page 12.
10. The DMA Controller will continue to fetch and process Buffer Descriptors until a Buffer Descriptor with DMA.SGS=1 is fetched. At which point Scatter Gather automation will complete processing of the Buffer Descriptor, set the SWCR.SGE=0 and halt.

Stopping and Starting SG Operation

Starting SG Operation

Scatter Gather operations will start when the software application sets SWCR.SGD=0, SWCR.SGE=1, and DMACR.SGS=0. When this condition is met, the DMA SG Controller will fetch the first Buffer Descriptor pointed to by its BDA register.

Stopping SG Operation (Synchronous stop)

Scatter Gather processing of Buffer Descriptors will continue until finished processing a descriptor that has DMACR.SGS=1. When the DMA SG controller reaches this stop condition the SWCR.SGE bit will be set to 0 and all processing will halt.

Stopping SG Operation (Asynchronous stop)

There may also be times when software wishes to stop SG operation without reaching the sentinel in the Buffer Descriptor list. Situations where this is the case, might include

- High priority Buffer Descriptors need to be inserted ahead of already scheduled Buffer Descriptors.
- New Buffer Descriptors need to be inserted at the end of the Buffer Descriptor list.
- Scatter Gather activity needs to be paused or aborted.

To affect an asynchronous stop, 1 is written to the SWCR.SGD bit, which essentially asynchronously inserts a new sentinel into the list. Scatter Gather automation will stop after completing any DMA operation (or any packets, if the channel is a packet channel) that it might have started. While Scatter Gather operations are completing both SGE and SGD will be set to a '1' indicating the DMA Controller is in the process of stopping.

Once stopped the SWCR.SGE bit will be automatically cleared to '0'. If the stop is synchronous, the Scatter Gather End (SGEND) interrupt will be generated and if asynchronous, the Scatter Gather Disable Acknowledge (SGDA) interrupt. If an asynchronous-stop point coincides with a synchronous-stop point, both interrupts will be generated.

Re-Starting SG Operation

With SG disabled (SGD=1), software may make any needed insertions or deletions to the Buffer Descriptor linked list (See <RD Red>"Inserting or Appending A Buffer Descriptor Chain" on page 41). If SG Automatic Restart is enabled (SWCR.DSGAR=0) then when software wishes SG operation to restart, it simply needs to set SWCR.SGD=0. If Automatic Restart is disabled (SWCR.DSGAR=1) then the software needs to verify that SG processing has halted, either by receipt of SGDA and/or SGEND interrupt or by pulling for SWCR.SGE=0 before attempting to restart the SG process. Once it has been determined that SG processing has halted in response to SWCR.SGD = 1, to restart simply establish the condition, SWCR.SGE=1, SWCR.SGD=0, and DMACR.SGS=0.

When SG restarts, processing will resume from the point of discontinuance.

Aborting SG Operation (Scatter Gather Automatic Restart Disabled)

If SG is disabled (SGD=1) during SG operation and the DMA Controller reaches a stop point (i.e. end of a packet) where SGE=0, then setting SWCR.DSGAR = 1 and SWCR.SGD=0 in the same register write will allow for an abort of Scatter Gather activity. In other words the DMA controller will NOT automatically restart Scatter Gather operations.

Management of Buffer Descriptors

Prior to starting Scatter Gather operations, the software application must set up a Buffer Descriptor or chain of Buffer Descriptors. Once the DMA Controller begins processing the Buffer Descriptors, it will fetch, process, and then update the Buffer Descriptors. By analyzing the Buffer Descriptors, the software application can read status on the associated DMA transfer, fetch user information on receive channels (See <RD Red>"Local Link Headers and Footers" on page 14), and determine completion of the transfer. With this information the software application can manage the Buffer Descriptors and data buffers.

The first field in a Buffer Descriptor is the DMASR field. Once a buffer descriptor has been processed by the DMA Controller this field will be updated with status concerning the transfer. When the software application sets up the Buffer Descriptor chain the DMASR field of each Buffer Descriptor should be set to zero. This will allow the software application to easily determine when a Buffer Descriptor has been processed and whether or not there was an errors during processing.

As each Buffer Descriptor is updated into remote memory, DMASR.DMADONE will be set to 1. By looking at DMASR.DMADONE bit and walking through the Buffer Descriptor chain, the software application can determine which Buffer Descriptors have been completed and which ones have not. Figure 5 shows remote memory where software has constructed a Buffer Descriptor chain. The DMA SG Controller, shown on the right, fetches Buffer Descriptors, processes them, and then updates the Buffer Descriptor in remote memory providing status. As can be seen DMASR.DMADONE=1 for all of the Buffer Descriptors that have been processed.

For further clarity, on receive channels, by monitoring the DMASR.DMADONE bit in the Buffer Descriptor, the software application can determine which data buffers, as described by the Buffer Descriptor, have received data and need to be processed. For transmit channels, DMASR.DMA-

DONE=1 indicates to the software application that the data in the associated buffer has been transmitted and is free to be modified.

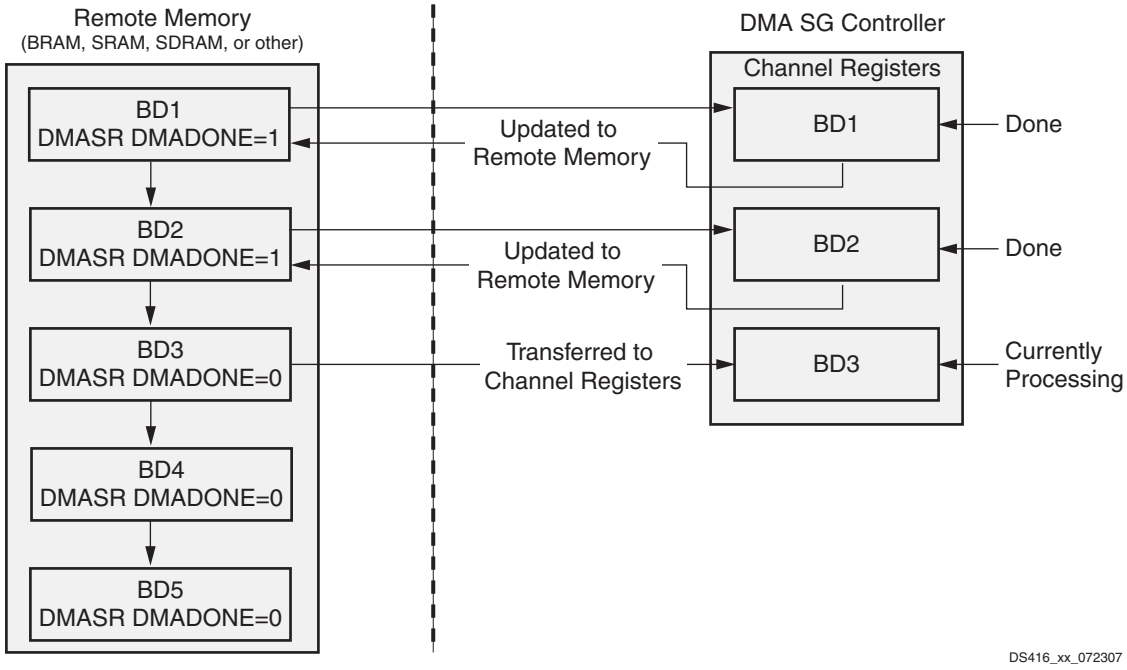

Figure 5: **Software - Hardware BD Processing**

If an error occurs during the DMA Transfer, either during the Buffer Descriptor fetch or during the actual requested DMA transfer, the DMASR.DBE bit will be set. If an error occurs during a transfer, an DMA Error (DE) interrupt will be generated, the SWCR.SGE bit will be cleared to 0, and Scatter Gather operations will be halted.

Local Link Headers and Footers

Packet Scatter Gather DMA channels utilize Local Link Headers and Footers to pass data in and out of the Buffer Descriptor User Fields, when headers and footers are enabled. This allows the software application to pass user defined data to and from UserIP via the Local Link data stream. To enable Headers and Footers for both a transmit and receive channel set C_RD_HDR_SIZE=16, C_RD_FTR_SIZE=8, C_WR_HDR_SIZE=16, and C_WR_FTR_SIZE=8.

The user fields, USR0, USR1, USR2, and USR3 of the first buffer descriptor are passed in and out of the header of the Local Link data stream. For transmit channels (i.e. Read channels), user fields 0 to 3 are written to the Local Link header as it is transmitted (See Figure 6). For receive channels (i.e. Write channels), user fields, 0 to 3 are read from the receive Local Link header (See Figure 7).

The user fields, USR4 and USR5 of the last buffer descriptor are passed in and out of the footer of the Local Link data stream. For transmit channels (i.e. Read channels), user fields 4 and 5 are written to the Local Link footer as it is transmitted (See Figure 6). For receive channels (i.e. Write channels), user field 5 is read from the footer. Note user field 4 for receive channels will contain a copy of the CR value when the buffer descriptor is updated (See Figure 7). It should be noted that on receive channels, each buffer descriptor defines a packet. In other words, multiple buffer descriptors cannot be used to describe a single packet for a receive channel.

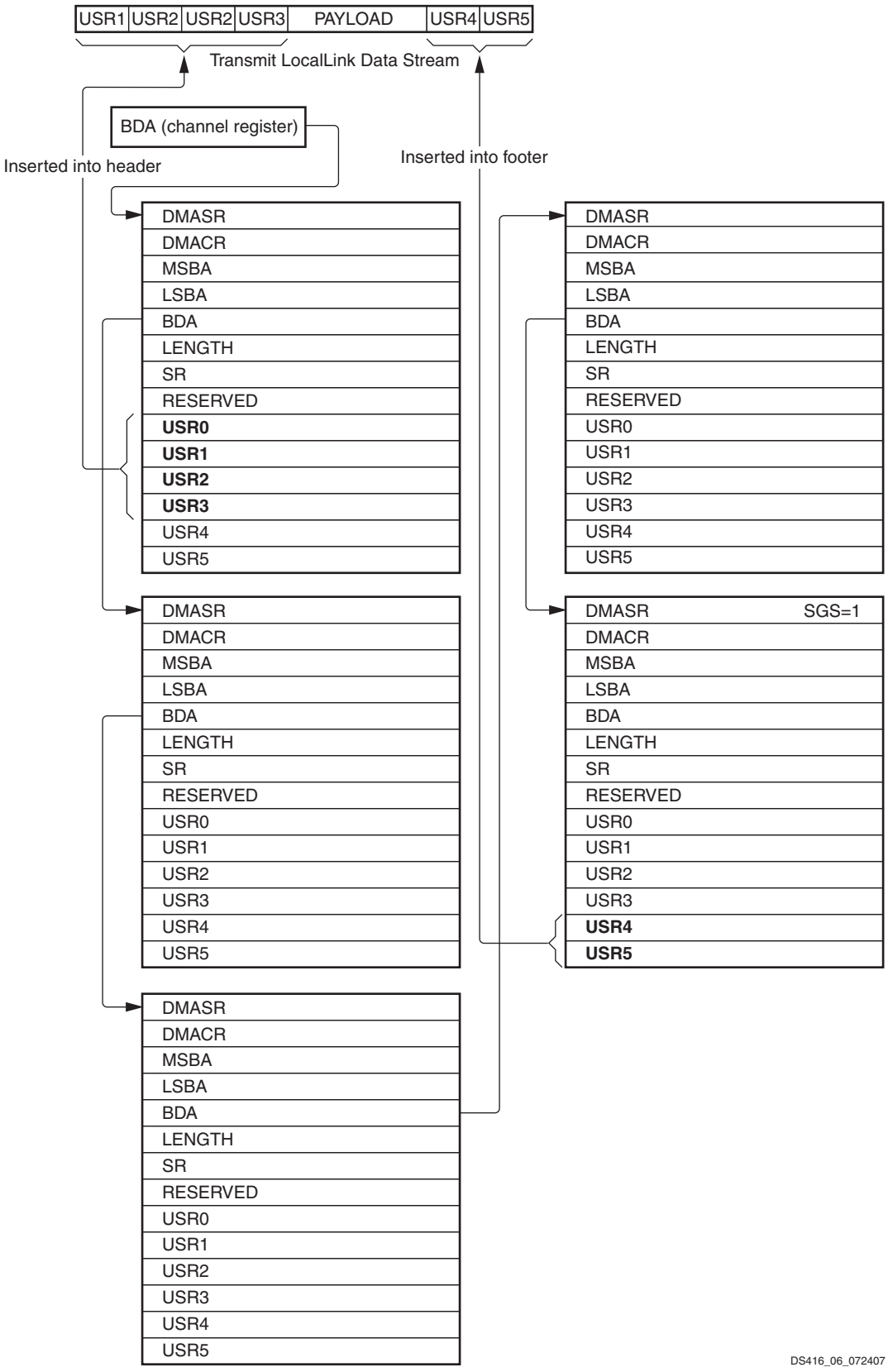

Figure 6: **Local Link Transmit Data Stream Header-Footer Assignment**

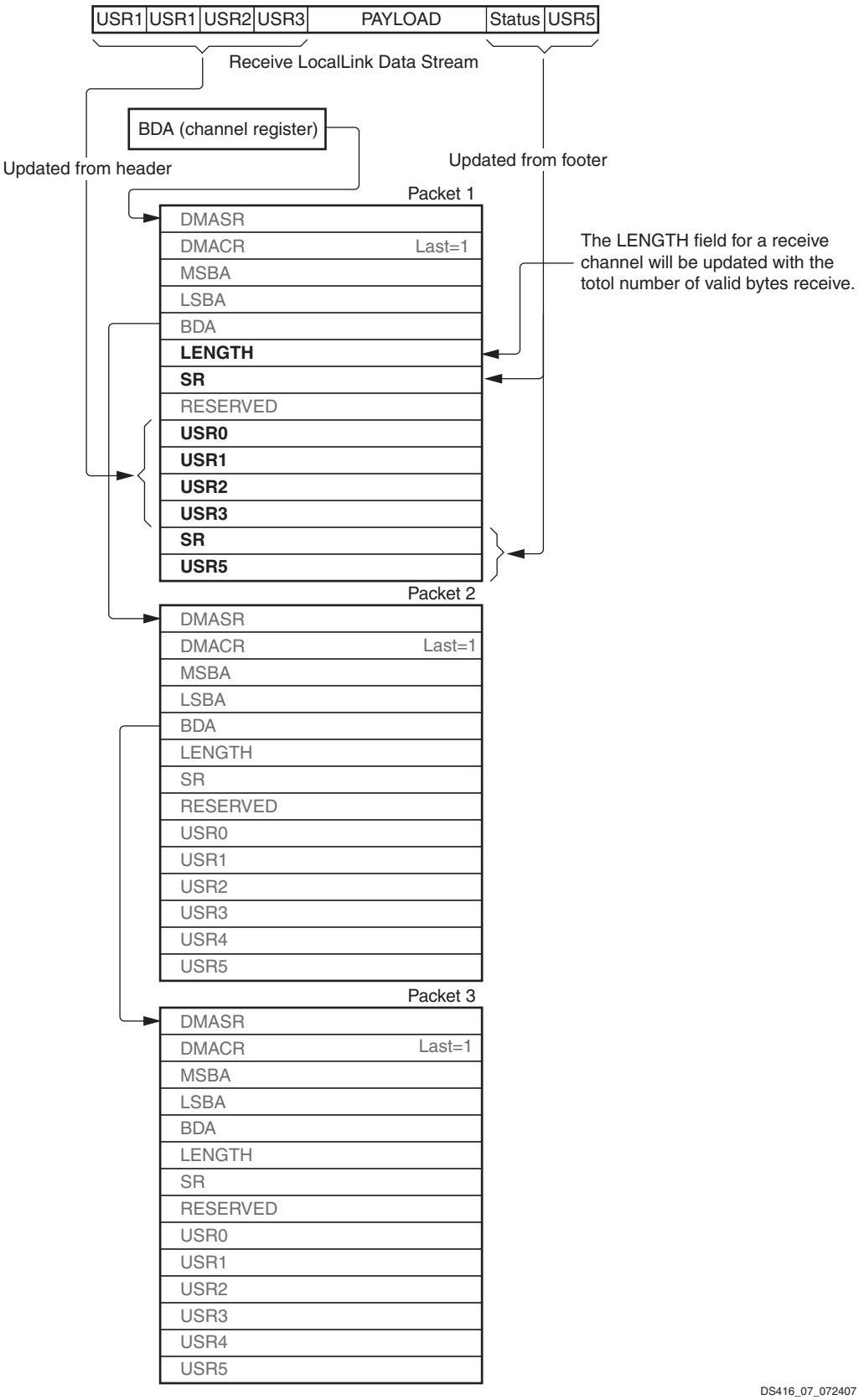

Figure 7: **Local Link Receive Data Stream Header - Footer Assignment**

DMA Controller Interrupt Description

Each channel may generate one or more events that are of interest to the user. Events are reported via the Interrupt Status Registers (ISR) for the individual channels, Figure 8. The ISR generates a system interrupt when any bit is set. Reading of the individual ISR's allows the software to determine which event occurred and for which channel the event was logged. Writing a 1 to the bit position of the event that was logged clears that event from the associated ISR register.

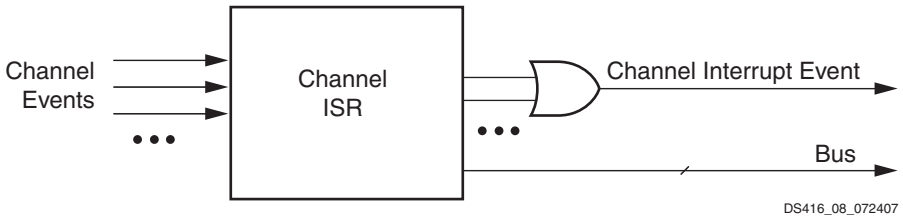

Figure 8: **Interrupt Status Register**

Interrupt Coalescing

Under SG operation, many packet transfers and the underlying DMA operations of which they are composed, proceed automatically. Nevertheless, software needs to receive reports of packets that have been handled by DMA SG so that it can perform its part of the process, e.g. posting of completion notifications to client processes, management of buffer resources, etc. For low packet-rate devices, an interrupt per packet might be acceptable. For high packet-rate devices, however, this may result in an undesirably high interrupt rate. Reducing the interrupt rate to below one per packet is called *interrupt coalescing*.

The user specifies the number of packets to be reported in aggregate by setting the value of the Packet Count Threshold (PCT). SG keeps a count of the number of packets it has handled but not yet reported. Whenever this unreported packet count grows equal to the Packet Count Threshold value, the Packet Count Threshold Reached (PCTR) event is set in the channel Interrupt Status Register (ISR) for the respective channel and the count is reset to zero.

Coalescing packets and reporting them in aggregate has an undesirable side effect, however. If the number of unreported packets does not grow large enough to reach the Packet Count Threshold, as could happen, for example, if the channel enters a period of inactivity, packets might wait indefinitely before being reported.

To solve this case, activity of a channel is monitored by a Packet Wait Timer. Whenever the unreported packet count is zero, the timer is inactive. Whenever the unreported packet count goes from zero to one, the timer starts counting toward a time-out at the Packet Wait Bound (PWB) value. If the time-out is reached, a Packet Wait Bound Reached (PWBR) event is set in the channel Interrupt Status Register (ISR) for the respective channel.

DMA SG Parameter Detailed Descriptions

C_IPIF_AWIDTH

This integer parameter is used by DMA to size bus address related components within the DMA.

C_IPIF_DWIDTH

This integer parameter is used by DMA to size the data bus related components within the DMA.

C_DMA_CHAN_TYPE

This integer array specifies the DMA channel type. Specify a DMA channel type for each channel. In other words, there should be an array element for each channel.

C_SPLIT_BUS_TYPE

This integer array specifies the transfer type for each channel in the Split Bus configuration. This is used to properly assign DMA channels at build time to the two command buses in the Split Bus configuration. This allows for the creation of two independent DMA channels in order to utilize the independent read and write controller in the PLB IPIF Master. This gives the DMA engine the ability to perform simultaneous read and write operations. There should be an array element for each channel.

Also, the channel transfer type is required to match the C_DMA_CHAN_TYPE when DMA SG is configured for Scatter Gather Packet DMA. In other words, if C_DMA_CHAN_TYPE is set to Packet Transmit (Type 2 = DMA Read) then the corresponding element of C_SPLIT_BUS_TYPE must be set to 0 (or Read). Likewise for a C_DMA_CHAN_TYPE of 3 (Packet Receive) which is a write channel, C_SPLIT_BUS_TYPE must be set to 1(or write). An assertion will be made if the types in C_DMA_CHAN_TYPE and C_SPLIT_BUS_TYPE do not match (See <RD Red>"Mismatch Array Elements" on page 40)

C_NUM_RX_CHANNELS

Number of receive channels configured for DMA. This parameter sizes the IP2SG_Rx_Length and IP2SG_Rx_LenFIFO_WrCnt ports and must equal the total number of receive channels specified in C_SPLIT_BUS_TYPE array.

C_NUM_TX_CHANNELS

Number of transmit channels configured for DMA. This parameter sizes the IP2SG_Tx_LenFIFO_WrCnt port and must equal the total number of transmit channels specified in C_SPLIT_BUS_TYPE array.

C_DMA_LENGTH_WIDTH

This integer value is used by DMA to size the DMA Command Length bus which feeds the IPIF Master.

C_REM_WIDTH

This integer value specifies the size of the DMA Local Link Remainder Bus. The value of this generic depends on the setting of the C_REM_CODING generic. If C_REM_CODING = 1 then the remainder bus is a encoded bus and C_REM_WIDTH is the $\log_2$(Local Link Data Bus Width). If C_REM_CODING = 2 then the remainder bus is a mask bus and C_REM_WIDTH is the (Local Link Data Bus Width)/8.

C_REM_CODING

This integer specifies the format of the remainder bus. Setting C_REM_CODING = 0 indicates that there is no remainder bus. Setting C_REM_CODING = 1 indicates that the remainder bus is in an encoded format (i.e. the value of the remainder specifies the number of bytes in the associated data word is valid. If C_REM_CODING is set to 2 then the remainder bus is in a mask format. (i.e. a 1 represents that the associated byte in the data bus is valid)

C_RD_HDR_SIZE

This integer value is used by DMA Read Header insert logic to specify the length in bytes of the DMA Local Link Header. Setting this generic to a value of 0 will remove the logic for inserting a header into the read data stream.

C_RD_FTR_SIZE

This integer value is used by DMA Read Footer insert logic to specify the length in bytes of the DMA Local Link Footer . Setting this generic to a value of 0 will remove the logic for inserting a footer into the read data stream.

C_WR_HDR_SIZE

This integer value is used by DMA Write Header stripping logic to specify the length in bytes of the DMA Local Link Header. Setting this generic to a value of 0 will remove the logic for stripping a header from the write data stream.

C_WR_FTR_SIZE

This integer value is used by DMA Write Footer stripping logic to specify the length in bytes of the DMA Local Link Footer . Setting this generic to a value of 0 will remove the logic for stripping a footer from the write data stream.

C_BD_DWIDTH

This integer value is used by SG to size the SG Buffer Descriptor Local Link Data Bus.

C_BD_REM_WIDTH

This integer value is used by SG to size the SG Buffer Descriptor Local Link Remainder Bus. The Buffer Descriptor Local Link Read REM bus, Bus2SG_CmdRd_rem, is not used for this revision of the DMA SG core.

C_BD_REM_CODING

This integer specifies the format of the Buffer Descriptor remainder bus. Setting *C_BD_REM_CODING* = 0 indicates that there is no remainder bus. Setting C_REM_CODING = 1 indicates that the remainder bus is in an encoded format (i.e. the value of the remainder specifies the number of bytes in the associated data word is valid. If *C_BD_REM_CODING* is set to 2 then the remainder bus is in a mask format. (i.e. a 1 represents that the associated byte in the data bus is valid). The Buffer Descriptor Local Link REM bus is not used for this revision of the DMA SG core.

C_BD_LENGTH_WIDTH

This integer value is used by SG to size the SG Buffer Descriptor Command Length bus which feeds the IPIF Master.

C_BD_BE_WIDTH

This integer value is used by SG to size the SG Buffer Descriptor Command Byte Enable Bus. This value should be set to C_BD_DWIDTH/8.

C_BD_RDCNT_WIDTH

This integer value is used by SG to size the SG Buffer Descriptor Read FIFO Read Count. The Read FIFO Read Count is currently not being used.

C_BD_WRCNT_WIDTH

This integer value is used by SG to size the SG Buffer Descriptor Write FIFO Write Count. The Write FIFO Write Count is currently not being used.

C_INTR_COALESCE

Enables or disables interrupt coalescing. Interrupt coalescing allows for a reduction of interrupts for high packet-rate devices (See <RD Red>"Interrupt Coalescing" on page 18).

C_CLK_PERIOD_PS

The period of the bus clock in ps. Knowledge of the period is only needed to divide the clock to form C_PACKET_WAIT_UNIT_NS.

C_PACKET_WAIT_UNIT_NS

Gives the unit in which packet-wait bounds are measured. The default value of 100,000ns (100us) is a typical setting.

C_FAMILY

This parameter is defined as a string. It specifies the target FPGA technology for implementation of the DMA SG. This parameter is required by the channel registers which utilizes Distributed or BRAM primitives.

DMA SG I/O Signal

The DMA SG service has 5 major interfaces. These are the Command Bus (CMD Bus), the DMA IP Interconnect (DMA IPIC), the Buffer Descriptor Local Link Interface (BD LLINK), the Local Link Header / Footer Interface (HdfFtr IF), and the User IP interface (USER IP). The device also generates an interrupt for use by a processing element. The I/O signals for the design are listed in Table 2.

Table 2: **DMA SG I/O Signals**

Port	Signal Name	Interface	I/O	Description
DMA Read Command Interface				
	DMA2Bus_CmdRd_Req	CMD Bus	O	Setting Read Request to a logical '1' instructs the IPIF Master to begin a DMA Read Cycle. This signal is held high until acknowledged by Bus2DMA_CmdRd_CmdAck or Bus2DMA_CmdRd_Retry.
	DMA2Bus_CmdRd_ChainReq	CMD Bus	O	Setting Chain Request to a logical 1 instructs the IPIF Master that the current and next command request should be chained together into 1 Local Link transfer. This signal should be driven high when DMA2Bus_CmdRd_Req is driven high.
	DMA2Bus_CmdRd_BE(0 to (C_IPIF_DWIDTH/8) -1)	CMD Bus	O	Command Bus Byte enables
	DMA2Bus_CmdRd_Addr (0 to **C_IPIF_AWIDTH** - 1)	CMD Bus	O	The address is the bus address from which the IPIF Master reads data in a DMA Read Cycle.

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
	DMA2Bus_CmdRd_AddrIncr	CMD Bus	O	The Address Increment specifies to the IPIF Master that the Source address should increment during burst cycles or not. Setting DMA2Bus_CmdRd_AddrIncr='0' specifies that the Source is a key whole address and the IPIF Master should not increment the address during bursts.
	DMA2Bus_CmdRd_Length(0 to **C_DMA_LENGTH_WIDTH** - 1)	CMD Bus	O	The Length specifies to the IPIF Master, the number of bytes to transfer during a DMA transaction.
	DMA2Bus_CmdRd_ DSize(0 to 2)	CMD Bus	O	DSize specifies the size of the DMA transfer, 000=byte (1 byte) 001=half-word (2 bytes) 010=word (4 bytes) 011=double-word (8 bytes) 100=quad-word (16 bytes) etc.
	DMA2Bus_CmdRd_ Type(0 to 2)	CMD Bus	O	Instructs the IPIF Master as to what type of bus transfer should be made, 000=Reserved 001=Bounded Fixed Length Burst 010=Bounded Indeterminate Burst 011=Reserved 100=Reserved 101=Reserved 110=Reserved 111= Reserved
	DMA2Bus_CmdRd_Lock	CMD Bus	O	Currently Not Used - Driven to '0'
	DMA2Bus_CmdRd_Rst	CMD Bus	O	Command Reset allows the DMA SG controller to reset the Master.
	DMA2Bus_CmdRd_DRE_Rst	CMD Bus	O	DRE Reset instructs the master to reset the Data Realignment Engine in preparation for a new DMA transfer.
	DMA2Bus_CmdRd_DRE_ Push2FIFO	CMD Bus	O	Currently Not Used - Driven to '0'
	DMA2Bus_CmdRd_DRE_ PassThru	CMD Bus	O	Setting DRE Pass Through to a logical 1 instructs the Master IPIF to bypass the Data Realignment Engine.
	Bus2DMA_CmdRd_CmdAck	CMD Bus	I	Command Acknowledge signals to the DMA SG that the read request has been acknowledged by the IPIF Master.
	Bus2DMA_CmdRd_Cmplt	CMD Bus	I	Command Complete signals to the DMA SG that the IPIF Master has completed the DMA cycle
	Bus2DMA_CmdRd_Retarbitrate	CMD Bus	I	Currently Not Used
	Bus2DMA_CmdRd_Error	CMD Bus	I	Command Error signals the DMA SG that an error occurred in performing the DMA transfer.
	Bus2DMA_CmdRd_Length_is_ Zero	CMD Bus	I	Currently Not Used

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
	Bus2DMA_CmdRd_Aborted	CMD Bus	I	The IPIF Master will set this signal to a logical 1 to issue an abort of the currently requested cycle.
DMA Write Command Interface				
	DMA2Bus_CmdWr_Req	CMD Bus	O	Setting Write Request to a logical '1' instructs the IPIF Master to begin a DMA Write Cycle. This signal is held high until acknowledged by Bus2DMA_CmdWr_CmdAck or Bus2DMA_CmdWr_Retry.
	DMA2Bus_CmdWr_ChainReq	CMD Bus	O	Setting Chain Request to a logical 1 instructs the IPIF Master that the current and next command request should be chained together into 1 Local Link transfer. This signal should be driven high when DMA2Bus_CmdRd_Req is driven high.
	DMA2Bus_CmdWr_BE(0 to (C_IPIF_DWIDTH/8) -1)	CMD Bus	O	Command Bus Byte enables
	DMA2Bus_CmdWr_Addr (0 to **C_IPIF_AWIDTH** - 1)	CMD Bus	O	The address is the bus address to which the IPIF Master writes data to in a DMA Write Cycle.
	DMA2Bus_CmdWr_AddrIncr	CMD Bus	O	The Address Increment specifies to the IPIF Master that the Destination address should increment during burst cycles or not. Setting DMA2Bus_CmdWr_AddrIncr='0' specifies that the Destination is a key whole address and the IPIF Master should not increment the address during bursts.
	DMA2Bus_CmdWr_Length(0 to **C_DMA_LENGTH_WIDTH** - 1)	CMD Bus	O	The Length specifies to the IPIF Master, the number of bytes to transfer during a DMA transaction.
	DMA2Bus_CmdWr_DSize (0 to 2)	CMD Bus	O	DSize specifies the size of the DMA transfer, 000=byte (1 byte) 001=half-word (2 bytes) 010=word (4 bytes) 011=double-word (8 bytes) 100=quad-word (16 bytes) etc.
	DMA2Bus_CmdWr_Type (0 to 2)	CMD Bus	O	Instructs the IPIF Master as to what type of bus transfer should be made, 000=Reserved 001=Bounded Fixed Length Burst 010=Bounded Indeterminate Burst 011=Reserved 100=Reserved 101=Reserved 110=Reserved 111= Reserved
	DMA2Bus_CmdWr_Lock	CMD Bus	O	Currently Not Used - Driven to '0'
	DMA2Bus_CmdWr_Rst	CMD Bus	O	Command Reset allows the DMA SG controller to reset the Master.

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
	DMA2Bus_CmdWr_DRE_Rst	CMD Bus	O	DRE Reset instructs the master to reset the Data Realignment Engine in preperation for a new DMA transfer.
	DMA2Bus_CmdWr_DRE_ Push2FIFO	CMD Bus	O	Currently Not Used - Driven to '0'
	DMA2Bus_CmdWr_DRE_ PassThru	CMD Bus	O	Setting DRE Pass Through to a logical 1 instructs the Master IPIF to bypass the Data Realignment Engine.
	Bus2DMA_CmdWr_CmdAck	CMD Bus	I	Command Acknowledge signals to the DMA SG that the write request has been acknowledged by the IPIF Master.
	Bus2DMA_CmdWr_Cmplt	CMD Bus	I	Command Complete signals to the DMA SG that the IPIF Master has completed the DMA cycle.
	Bus2DMA_CmdWr_Rearbitrate	CMD Bus	I	Currently Not Used
	Bus2DMA_CmdWr_Error	CMD Bus	I	Command Error signals the DMA SG that an error occurred in performing the DMA transfer.
	Bus2DMA_CmdWr_Length_is_ Zero	CMD Bus	I	Currently Not Used
	Bus2DMA_CmdWr_Aborted	CMD Bus	I	The IPIF Master will set this signal to a logical 1 to issue an abort of the currently requested cycle
	MstWr2SG_Xfer_Pending	CMD Bus	I	Currently Not Used
DMA LLink Read Header / Footer Insert				
	MstRd2SG_Rdy_for_Hdr	HdfFtr IF	I	The IPIF Master will set this signal to a logical 1 to indicate that the Local Link Read Back end is ready for a header
	MstRd2SG_Rdy_for_Ftr	HdfFtr IF	I	The IPIF Master will set this signal to a logical 1 to indicate that the Local Link Read Backend is ready for a footer
	SG2MstRd_Hdr_Done	HdfFtr IF	O	The DMA SG Controller will set this signal to a logical 1 to indicate that the header transfer is complete.
	SG2MstRd_Ftr_Done	HdfFtr IF	O	The DMA SG Controller will set this signal to a logical 1 to indicate that the footer transfer is complete.
	MstRd2SG_LLFIFO_Full	HdrFtr IF	I	The IPIF Master will set this signal to a logical 1 to indicate that the Local Link Read Backend FIFO is Full and cannot except data.
	SG2MstRd_LLFIFO_WEN	HdrFtr IF	O	Read Local Link FIFO Write Enable. Header data is written to the IPIF MAster's Local Link Read Backend FIFO when the write enable is set to a logical 1.
	SG2MstRd_LLFIFO_SOF	HdrFtr IF	O	Read Local Link FIFO Start of Frame. When SG2MstRd_LLFIFO_WEN is a logical 1 then setting SOF to a 1 will write a Start of Frame into the Local Link Backend FIFO. This signal marks the beginning of a frame and specifically the beginning of the Local Link Header.

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
	SG2MstRd_LLFIFO_EOF	HdrFtr IF	O	Read Local Link FIFO End of Frame. When SG2MstRd_LLFIFO_WEN is a logical 1 then setting EOF to a 1 will write a End of Frame into the Local Link Backend FIFO. This signal marks the end of a frame and specifically the end of the Local Link Footer.
	SG2MstRd_LLFIFO_REM (0 to **C_REM_WIDTH** - 1)	HdrFtr IF	O	Read Local Link FIFO Remainder Bus. This bus indicates which bytes of the SG2MstRd_LLFIFO_Data bus are valid.
	SG2MstRd_LLFIFO_Data (0 to **C_IPIF_DWIDTH** - 1)	HdrFtr IF	O	Read Local Link FIFO Data Bus. This bus passes the header and footer data to the Local Link Read Backend when SG2MstRd_LLFIFO_WEN is set to a logical 1.
DMA LLink Write Header / Footer Strip				
	MstWr2SG_Rdy_for_Hdr	HdrFtr IF	I	The IPIF Master will set this signal to a logical 1 to indicate that the Local Link Write Backend has a header ready for stripping.
	MstWr2SG_Rdy_for_Ftr	HdrFtr IF	I	The IPIF Master will set this signal to a logical 1 to indicate that the Local Link Read Backend has a footer ready for stripping.
	SG2MstWr_Hdr_Done	HdrFtr IF	O	The DMA SG Controller will set this signal to a logical 1 to indicate that the header transfer is complete.
	SG2MstWr_Ftr_Done	HdrFtr IF	O	The DMA SG Controller will set this signal to a logical 1 to indicate that the footer transfer is complete.
	MstWr2SG_LLFIFO_DValid	HdrFtr IF	I	The IPIF Master will set this signal to a logical 1 to indicate that there is valid data in the local link receive FIFO.
	SG2MstWr_LLFIFO_REN	HdrFtr IF	O	Write Local Link FIFO Read Enable. Header data is read from the IPIF MAster's Local Link Write Backend FIFO when the read enable is set to a logical 1.
	SG2MstWr_LLFIFO_SOF	HdrFtr IF	I	Write Local Link FIFO Start of Frame. When SG2MstWr_LLFIFO_REN is a logical 1, an SOF of a 1 marks the beginning of a frame and specifically the beginning of the Local Link Header.
	SG2MstWr_LLFIFO_EOF	HdrFtr IF	I	Write Local Link FIFO End of Frame. When SG2MstWr_LLFIFO_REN is a logical 1, an EOF of a 1 marks the end of a frame and specifically the end of the Local Link Footer.
	SG2MstWr_LLFIFO_SOP	HdrFtr IF	I	Write Local Link FIFO Start of Payload. When SG2MstWr_LLFIFO_REN is a logical 1, an SOP of a 1 marks the beginning of a payload and specifically the end of the Local Link Header.
	SG2MstWr_LLFIFO_EOP	HdrFtr IF	I	Write Local Link FIFO End of Payload. When SG2MstWr_LLFIFO_REN is a logical 1, an EOP of a 1 marks the end of a payload and specifically the beginning of the Local Link Footer.

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
	MstWr2SG_LLFIFO_REM (0 to **C_REM_WIDTH** - 1)	HdrFtr IF	I	Write Local Link FIFO Remainder Bus. This bus indicates which bytes of the SG2MstWr_LLFIFO_Data bus are valid.
	MstWr2SG_LLFIFO_Data (0 to **C_IPIF_DWIDTH** - 1)	HdrFtr IF	I	Write Local Link FIFO Data Bus. This bus passes the header and footer data to the DMA SG Controller from the Local Link Read Backend when SG2MstWr_LLFIFO_REN is set to a logical 1.
	MstWr_LLFIFO_REN	HdrFtr IF	I	Master Write Controller's Local Link FIFO Read Enable. This signal is monitored to calculate the payload length of receive data.
SG Read Command Interface				
	SG2Bus_CmdRd_Req	CMD Bus	O	Setting Read Request to a logical '1' instructs the IPIF Master to begin a Buffer Descriptor Read Cycle. This signal is held high until acknowledged by Bus2SG_CmdRd_CmdAck or Bus2SG_CmdRd_Retry.
	SG2Bus_CmdRd_BE(0 to **C_BD_BE_WIDTH** - 1)	CMD Bus	O	SG Buffer Descriptor Command Bus Byte enables
	SG2Bus_CmdRd_Addr(0 to **C_IPIF_AWIDTH** - 1)	CMD Bus	O	The address is the Buffer Descriptor address from which the IPIF Master reads Buffer Descriptor data.
	SG2Bus_CmdRd_Length(0 to **C_BD_LENGTH_WIDTH** - 1)	CMD Bus	O	The Length specifies to the IPIF Master, the number of bytes to transfer during a BD Fetch.
	SG2Bus_CmdRd_DSize(0 to 2)	CMD Bus	O	DSize specifies the size of the BD transfer, 000=byte (1 byte) 001=half-word (2 bytes) 010=word (4 bytes) 011=double-word (8 bytes) 100=quad-word (16 bytes) etc.
	SG2Bus_CmdRd_Type(0 to 2)	CMD Bus	O	Instructs the IPIF Master as to what type of bus transfer should be made, 000=Reserved 001=Bounded Fixed Length Burst 010=Bounded Indeterminate Burst 011=Reserved 100=Reserved 101=Reserved 110=Reserved 111= Reserved
	SG2Bus_CmdRd_Lock	CMD Bus	O	Currently Not Used - Driven to '0'
	SG2Bus_CmdRd_Rst	CMD Bus	O	Command Reset allows the DMA SG Controller to reset the Master.
	Bus2SG_CmdRd_CmdAck	CMD Bus	I	Command Acknowledge signals to the DMA SG that the read request has been acknowledged by the IPIF Master.
	Bus2SG_CmdRd_Cmplt	CMD Bus	I	Command Complete signals to the DMA SG that the IPIF Master as completed the BD cycle.

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
	Bus2SG_CmdRd_Rearbitrate	CMD Bus	I	Currently Not Used
	Bus2SG_CmdRd_Error	CMD Bus	I	Command Error signals the DMA SG that an error occurred in performing the BD transfer.
	Bus2SG_CmdRd_Length_is_Zero	CMD Bus	I	Command Amount of Data Transferred flag. At Command Complete if set to '0' then all data was NOT transferred. If set to '1' then all data was transferred.
	Bus2SG_CmdRd_Aborted	CMD Bus	I	The IPIF Master will set this signal to a logical 1 to issue an abort of the currently requested cycle
SG Write Command Interface				
	SG2Bus_CmdWr_Req	CMD Bus	O	Setting Write Request to a logical '1' instructs the IPIF Master to begin a Buffer Descriptor Write Cycle. This signal is held high until acknowledged by Bus2SG_CmdWr_CmdAck or Bus2SG_CmdWr_Retry.
	SG2Bus_CmdWr_BE(0 to **C_BD_BE_WIDTH** - 1)	CMD Bus	O	SG Buffer Descriptor Command Bus Byte enables
	SG2Bus_CmdWr_Addr(0 to **C_IPIF_AWIDTH** - 1)	CMD Bus	O	The address is the Buffer Descriptor address from which the IPIF Master Writes Buffer Descriptor data.
	SG2Bus_CmdWr_Length(0 to **C_BD_LENGTH_WIDTH** - 1)	CMD Bus	O	The Length specifies to the IPIF Master, the number of bytes to transfer during a BD Fetch.
	SG2Bus_CmdWr_DSize(0 to 2)	CMD Bus	O	DSize specifies the size of the BD transfer, 000=byte (1 byte) 001=half-word (2 bytes) 010=word (4 bytes) 011=double-word (8 bytes) 100=quad-word (16 bytes) etc
	SG2Bus_CmdWr_Type(0 to 2)	CMD Bus	O	Instructs the IPIF Master as to what type of bus transfer should be made, 000=Reserved 001=Bounded Fixed Length Burst 010=Bounded Indeterminate Burst 011=Reserved 100=Reserved 101=Reserved 110=Reserved 111= Reserved
	SG2Bus_CmdWr_Lock	CMD Bus	O	Currently Not Used - Driven to '0'
	SG2Bus_CmdWr_Rst	CMD Bus	O	Command Reset allows the DMA SG Controller to reset the Master.
	Bus2SG_CmdWr_CmdAck	CMD Bus	I	Command Acknowledge signals to the DMA SG that the write request has been acknowledged by the IPIF Master.
	Bus2SG_CmdWr_Cmplt	CMD Bus	I	Command Complete signals to the DMA SG that the IPIF Master as completed the BD cycle.
	Bus2SG_CmdWr_Rearbitrate	CMD Bus	I	Currently Not Used

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
SG Local Link Write				
	Bus2SG_CmdWr_Error	CMD Bus	I	Command Error signals the DMA SG that an error occurred in performing the BD transfer.
	Bus2SG_CmdWr_Length_is_Zero	CMD Bus	I	Command Amount of Data Transferred flag. At Command Complete if set to '0' then all data was NOT transferred. If set to '1' then all data was transferred.
	Bus2SG_CmdWr_Aborted	CMD Bus	I	The IPIF Master will set this signal to a logical 1 to issue an abort of the currently requested cycle
	Bus2SG_CmdWr_WrCnt(0 to **C_BD_WRCNT_WIDTH** - 1)	BD LLINK	I	Currently Not Used
	SG2Bus_CmdWr_sof_n	BD LLINK	O	Active low signal marking the start of a local link frame
	SG2Bus_CmdWr_eof_n	BD LLINK	O	Active low signal marking the end of a local link frame.
	SG2Bus_CmdWr_src_rdy_n	BD LLINK	O	Active low signal indicating that the Master IPIF is ready to accept data
	SG2Bus_CmdWr_src_dsc_n	BD LLINK	O	Active low signal indicating that the Master IPIF is disconnecting
	Bus2SG_CmdWr_dst_rdy_n	BD LLINK	I	Active low signal indicating that the SG Engine is ready to send data
SG Local Link Read				
	Bus2SG_CmdWr_dst_dsc_n	BD LLINK	I	Active low signal indicating that the SG Engine is disconnecting
	SG2Bus_CmdWr_rem(0 to **C_BD_REM_WIDTH** - 1)	BD LLINK	O	Local Link write remainder bus specifying valid bytes of the CmdWr_data bus.
	SG2Bus_CmdWr_data(0 to **C_BD_DWIDTH** - 1)	BD LLINK	O	Local link write data bus used for updating buffer descriptor data to remote memory.
	Bus2SG_CmdRd_RdCnt(0 to **C_BD_RDCNT_WIDTH** - 1)	BD LLINK	I	Currently Not Used
	Bus2SG_CmdRd_sof_n	BD LLINK	I	Active low signal marking the start of a local link frame
	Bus2SG_CmdRd_eof_n	BD LLINK	I	Active low signal marking the end of a local link frame.
	Bus2SG_CmdRd_src_rdy_n	BD LLINK	I	Active low signal indicating that the Master IPIF is ready to send data
	Bus2SG_CmdRd_src_dsc_n	BD LLINK	I	Active low signal indicating that the Master IPIF is disconnecting
	SG2Bus_CmdRd_dst_rdy_n	BD LLINK	O	Active low signal indicating that the SG Engine is ready to accept data
DMA Slave Interconnect				
	SG2Bus_CmdRd_dst_dsc_n	BD LLINK	O	Active low signal indicating that the SG Engine is disconnecting
	Bus2SG_CmdRd_rem(0 to **C_BD_REM_WIDTH** - 1)	BD LLINK	I	Currently Not Used

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
	Bus2SG_CmdRd_data(0 to **C_BD_DWIDTH**-1)	BD LLINK	I	Local link read data bus used for fetching buffer descriptor data to local channel registers.
	Bus2IP_Clk	DMA IPIC	I	IPIC clock pass-through from Bus_Clk.
	Bus2IP_Reset	DMA IPIC	I	Active high reset for use by the DMA SG.
	Bus2IP_Freeze	DMA IPIC	I	Currently Not Used
	Bus2IP_Data(0 to **C_IPIF_DWIDTH**-1)	DMA IPIC	I	Write data bus to the DMA SG. Write data is accepted by the DMA SG by assertion of the IP2Bus_Ack signal at the rising edge of the Bus2IP_Clk.
	Bus2IP_Addr(0 to **C_IPIF_AWIDTH**-1)	DMA IPIC	I	Address bus indicating the desired address of the requested read or write operation.
	Bus2IP_RNW	DMA IPIC	I	This signal indicates the sense of a requested operation with the DMA SG. High is a read, low is a write.
	Bus2IP_BE(0 to (**C_IPIF_DWIDTH**/8)-1)	DMA IPIC	I	Byte enable qualifiers for the requested read or write operation with the DMA SG. Bit 0 corresponds to Byte lane 0, Bit 1 to Byte lane 1, and so on.
	Bus2IP_RdReq	DMA IPIC	I	Active high read request of the DMA SG Registers.
	Bus2IP_WrReq	DMA IPIC	I	Active high write request of the DMA SG Registers.
	Bus2IP_WrCE	DMA IPIC	I	Active high write chip enable of the DMA SG Registers.
	Bus2IP_CS	DMA IPIC	I	Active high chip select for DMA register access. Assertion of a chip select indicates an active transaction request to the DMA registers.
	Bus2IP_PselHit	DMA IPIC	I	Active high early indicator that a IPIC slave access is in process
	IP2Bus_Data(0 to **C_IPIF_DWIDTH** -1)	DMA IPIC	O	Input read data bus from the DMA SG. Data is qualified with the assertion of IP2Bus_Ack signal and the rising edge of the Bus2IP_Clk.
	IP2Bus_AddrAck	DMA IPIC	O	Active high signal that advances the IPIF Address counter during multiple data beat transfers.
	IP2Bus_WrAck	DMA IPIC	O	Active high write data acknowledgement For a write transaction, data on the Bus2IP_Data bus is deemed accepted by the DMA SG at the rising edge of the Bus2IP_Clk whenever IP2Bus_WrAck is asserted and an IPIC write transaction is active.
	IP2Bus_RdAck	DMA IPIC	O	Active high read data acknowledgement For a read transaction, data on the IP2Bus_Data bus is deemed accepted by the DMA SG at the rising edge of the Bus2IP_Clk whenever IP2Bus_RdAck is asserted and an IPIC read transaction is active.
	IP2Bus_Busy	DMA IPIC	O	Active high indication that the DMA SG is busy and cannot accept a slave cycle.

Table 2: **DMA SG I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
	IP2Bus_InterEvent(0 to C_DMA_CHAN_TYPE'length - 1)	DMA IPIC	O	DMA Interrupt Event. Provides one interrupt output for each channel indicating an enabled event for a particular channel as occurred.
DMA User Interface				
	IP2Bus_Retry	DMA IPIC	O	Active high signal indicating the DMA SG is requesting a retry of an active operation.
	IP2Bus_Error	DMA IPIC	O	Active high signal indicating the DMA SG has encountered an error with the requested operation. This signal is asserted in conjunction with IP2Bus_Ack.
	IP2Bus_ToutSup	DMA IPIC	O	Active high signal requesting suppression of the transaction time-out function in the IPIF for the active read or write operation.
	IP2SG_RxLength(0 to **C_NUM_RX_CHANNELS** - 1)			Number of bytes in the received packet. Must be valid from the start of frame to the end of frame. This port is a vhdl type of RXLENGTH_ARRAY_TYPE which is found in dma_sg_pkg of the dma_sg_library. RXLENGTH_ARRAY_TYPE is an array of 32-Bit wide vectors of which only the C_DMA_LENGTH_WIDTH least significant bits of each vector are used.
	IP2SG_Rx_LenFIFO_WrCnt(0 to **C_NUM_RX_CHANNELS** - 1)			Write count from the Receive Length FIFO. Bit 0 should be connected to the Receive Length FIFO empty flag. This port is a vhdl type of FIFO_WRCNT_ARRAY_TYPE which is found in dma_sg_pkg of the dma_sg library. FIFO_WRCNT_ARRAY_TYPE is an array of 8-Bit wide vectors.
	IP2SG_Tx_LenFIFO_WrCnt(0 to **C_NUM_TX_CHANNELS** - 1)			Currently Not Used Write count from the Transmit Length FIFO. This port is a vhdl type of FIFO_WRCNT_ARRAY_TYPE which is found in dma_sg_pkg of the dma_sg library. FIFO_WRCNT_ARRAY_TYPE is an array of 8-Bit wide vectors.
	DMA2IP_Device_Sel(0 to **C_DMA_CHAN_TYPE**'length - 1**)**	User IP	O	User device currently being targeted for the DMA operation. This port is a vhdl type of DEVICE_SEL_ARRAY_TYPE which is found in dma_sg_pkg of the dma_sg library. DEVICE_SEL_ARRAY_TYPE is an array of 8-Bit wide vectors.

Allowable Parameter Combinations

The DMA SG supports up to 2 DMA channels. The channel type is specified with the C_DMA_CHAN_TYPE generic.

C_SPLIT_BUS_TYPE specifies the transfer direction of the DMA channel. The elements of this integer array must match in type to the elements of C_DMA_CHAN_TYPE. For example, if a channel in C_DMA_CHAN_TYPE is specified to be a receive channel (DMA Write) then the corresponding index in C_SPLIT_BUS_TYPE must be of type receive i.e. equal to 1.

C_SPLIT_BUS_TYPE must match in length to C_DMA_CHAN_TYPE.

Parameter - Port Dependencies

The DMA parameterization has effects on many of it's I/O port sizes. These are indicated in the port definitions presented in Table 2. There are cases where DMA SG ports are rendered unused based on parameter settings. These dependencies are summarized in Table 3. Unused DMA SG Input Ports need to be tied to logic '0'. Unused DMA SG Output Ports need to be tied to 'open'.

Table 3: **Parameter/Port Dependencies**

Name	Affects	Depends	Relationship Description
Design Parameters			
C_IPIF_AWIDTH	Bus2IP_Addr DMA2Bus_CmdRd_Addr DMA2Bus_CmdWr_Addr SG2Bus_CmdRd_Addr SG2Bus_CmdWr_Addr		Width of the Address Buses vary based on the IPIF address width.
C_IPIF_DWIDTH	Bus2IP_Data Bus2IP_BE IP2Bus_Data DMA2Bus_CmdRd_BE DMA2Bus_CmdWr_BE SG2MstRd_LLFIFO_Data MstWr2SG_LLFIFO_Data		Width of the Data Buses vary based on the IPIF data width.
C_DMA_CHAN_TYPE	DMA2IP_Device_Select SG2Bus_CmdRd_x[1] Bus2SG_CmdRd_x[1] SG2Bus_CmdWr_x[1] Bus2SG_CmdWr_x[1] SG2MstRd_x[1] MstRd2SG_x[1] SG2MstWr_x[1] MstWr2SG_x[1] IP2Bus_InterEvent		Width of DMA2IP_Device_Select and IP2Bus_IntrEvent varies based on C_DMA_CHAN_TYPE'length SG Command buses are enabled if at least one channel is configured for SG operation, C_DMA_CHANNEL_TYPE = 1, 2, and/or 3 Header/Footer Buses, SG2MstRd, MstRd2SG, SG2MstWr, and MstWr2SG, are enabled if at least one channel is configured for Packet SG
C_DMA_LENGTH_WIDTH	DMA2Bus_CmdRd_Length DMA2Bus_CmdWr_Length		Width of the Length Buses vary based on the DMA Length Width.
C_BD_LENGTH_WIDTH	SG2Bus_CmdRd_Length SG2Bus_CmdWr_Length		Width of the Length Buses vary based on the SG Length Width.

Table 3: **Parameter/Port Dependencies *(Cont'd)***

Name	Affects	Depends	Relationship Description
C_BD_DWIDTH	Bus2SG_CmdRd_data SG2Bus_CmdWr_data		The data bus widths are sized by C_BD_DWIDTH
C_BD_BE_WIDTH	SG2Bus_CmdRd_BE SG2Bus_CmdWr_BE		The BE width is sized by C_BD_BE_DWIDTH.
C_BD_REM_WIDTH	Bus2SG_CmdRd_rem SG2Bus_CmdWr_rem		The REM Bus widths are sized by C_BD_REM_WIDTH
C_RD_HDR_SIZE	SG2MstRd_x MstRd2SG_x		If C_RD_HDR_SIZE = 0 and C_RD_FTR_SIZE = 0 then outputs are driven to inactive and inputs are ignored.
C_RD_FTR_SIZE	SG2MstRd_x MstRd2SG_x		If C_RD_HDR_SIZE = 0 and C_RD_FTR_SIZE = 0 then outputs are driven to inactive and inputs are ignored.
C_WR_HDR_SIZE	SG2MstWr_x MstWr2SG_x		If C_WR_HDR_SIZE = 0 and C_WR_FTR_SIZE = 0 then outputs are driven to inactive and inputs are ignored.
C_WR_FTR_SIZE	SG2MstWr_x MstWr2SG_x		If C_WR_HDR_SIZE = 0 and C_WR_FTR_SIZE = 0 then outputs are driven to inactive and inputs are ignored.
C_REM_WIDTH	SG2MstRd_LLFIFO_REM MstWr2SG_LLFIFO_REM		Width of the Remainder buses vary based on C_REM_WIDTH.
I/O Signals			
Bus2IP_Addr		C_IPIF_AWIDTH	Width of the Address Bus vary based on the IPIF address width.
DMA2Bus_CmdRd_Addr		C_IPIF_AWIDTH	Width of the Address Bus vary based on the IPIF address width This port is disabled when C_SPLIT_BUS_ENABLE=0
DMA2Bus_CmdWr_Addr		C_IPIF_AWIDTH	Width of the Address Bus vary based on the IPIF address width
Bus2IP_Data		C_IPIF_DWIDTH	Width of the Data Busses vary based on the IPIF data width.
Bus2IP_BE		C_IPIF_DWIDTH	Width of byte enables vary based on $\log_2$(C_IPIF_DWIDTH)
DMA2Bus_CmdRd_BE		C_IPIF_DWIDTH	Width of byte enables vary based on $\log_2$(C_IPIF_DWIDTH) This port is disabled when C_SPLIT_BUS_ENABLE=0
DMA2Bus_CmdWr_BE		C_IPIF_DWIDTH	Width of byte enables vary based on $\log_2$(C_IPIF_DWIDTH)
SG2Bus_CmdRd_BE		C_BD_BE_WIDTH C_DMA_CHAN_TYPE	Width of byte enables vary based on C_BD_BE_WIDTH SG type buses are enabled when at least one channel is of type Scatter Gather.

Table 3: **Parameter/Port Dependencies *(Cont'd)***

Name	Affects	Depends	Relationship Description
SG2Bus_Cmd Wr_BE		C_BD_BE_ WIDTH C_DMA_ CHAN_TYPE	Width of byte enables vary based on C_BD_BE_WIDTH SG type buses are enabled when at least one channel is of type Scatter Gather. There must also be at least 1 channel of type Packet SG, C_DMA_CHAN_TYPE = 2 or 3, for this bus to be enabled.
DMA2IP_Device_S elect		C_DMA_ CHAN_TYPE	Width of Select bus varies based on C_DMA_CHAN_TYPE'length
DMA2Bus_Cmd Rd_Length		C_DMA_LENG TH_WIDTH	Width of the Length Buses vary based on the DMA Length Width.
DMA2Bus_Cmd Wr_Length		C_DMA_LENG TH_WIDTH	Width of the Length Buses vary based on the DMA Length Width.
SG2Bus_Cmd Rd_Length		C_BD_LENGT H_WIDTH C_DMA_CHAN _TYPE	Width of byte enables vary based on C_BD_LENGTH_WIDTH SG type buses are enabled when at least one channel is of type Scatter Gather. This port is disabled when C_SPLIT_BUS_ENABLE=0
SG2Bus_Cmd Wr_Length		C_BD_LENGT H_WIDTH C_DMA_CHAN _TYPE	Width of byte enables vary based on C_BD_LENGTH_WIDTH SG type buses are enabled when at least one channel is of type Scatter Gather. There must also be at least 1 channel of type Packet SG, C_DMA_CHAN_TYPE = 2 or 3, for this bus to be enabled.
DMA2Bus_Cmd Rd_DSize		C_SPLIT_BUS_ ENABLE	This port is disabled when C_SPLIT_BUS_ENABLE=0
DMA2Bus_Cmd Wr_DSize		C_SPLIT_BUS_ ENABLE	This port is disabled when C_SPLIT_BUS_ENABLE=0
SG2Bus_Cmd Rd_DSize		C_SPLIT_BUS_ ENABLE C_DMA_CHAN _TYPE	This port is disabled when C_SPLIT_BUS_ENABLE=0 and at least 1 channel is configured for SG operation, C_DMA_CHANNEL_TYPE = 1, 2, and/or 3
SG2Bus_Cmd Wr_DSize		C_SPLIT_BUS_ ENABLE C_DMA_CHAN _TYPE	This port is disabled when C_SPLIT_BUS_ENABLE=0 and at least 1 channel is configured for Packet SG operation, C_DMA_CHANNEL_TYPE = 2 or 3
DMA2Bus_Cmd Rd_Type		C_SPLIT_BUS_ ENABLE	This port is disabled when C_SPLIT_BUS_ENABLE=0
DMA2Bus_Cmd Wr_Type		C_SPLIT_BUS_ ENABLE	This port is disabled when C_SPLIT_BUS_ENABLE=0
SG2Bus_Cmd Rd_Type		C_SPLIT_BUS_ ENABLE C_DMA_CHAN _TYPE	This port is disabled when C_SPLIT_BUS_ENABLE=0 and at least 1 channel is configured for SG operation, C_DMA_CHANNEL_TYPE = 1, 2, and/or 3

Table 3: **Parameter/Port Dependencies *(Cont'd)***

Name	Affects	Depends	Relationship Description
SG2Bus_Cmd Wr_Type		C_SPLIT_BUS_ENABLE C_DMA_CHAN_TYPE	This port is disabled when C_SPLIT_BUS_ENABLE=0 and at least 1 channel is configured for Packet SG operation, C_DMA_CHANNEL_TYPE = 2 or 3
SG2Bus_Cmd Rd_x		C_DMA_CHAN_TYPE	Ports are enabled if at least one channel is configured for SG operation, C_DMA_CHANNEL_TYPE = 1, 2, and/or 3
Bus2SG_Cmd Rd_x		C_DMA_CHAN_TYPE	Ports are enabled if at least one channel is configured for SG operation, C_DMA_CHANNEL_TYPE = 1, 2, and/or 3
SG2Bus_Cmd Wr_x		C_DMA_CHAN_TYPE	Ports are enabled if at least one channel is configured for Packet SG operation, C_DMA_CHANNEL_TYPE = 2 or 3
Bus2SG_Cmd Wr_x		C_SPLIT_BUS_ENABLE C_DMA_CHAN_TYPE	Ports are enabled if at least one channel is configured for SG operation, C_DMA_CHANNEL_TYPE = 1, 2, and/or 3
SG2Bus_Cmd Wr_rem		C_BD_REM_WIDTH C_BD_REM_CODING	C_BD_REM_WIDTH specifies the width of the rem ports. C_BD_REM_CODING specifies the format of the rem ports. 0=no rem, 1=encoded rem, and 2 = mask rem.
SG2Bus_Cmd Wr_data		C_BD_DWIDTH	C_BD_DWIDTH specifies the width of the data port
Bus2SG_Cmd Rd_rem		C_BD_REM_WIDTH C_BD_REM_CODING	C_BD_REM_WIDTH specifies the width of the rem ports. C_BD_REM_CODING specifies the format of the rem ports. 0=no rem, 1=encoded rem, and 2 = mask rem.
Bus2SG_Cmd Rd_data		C_BD_DWIDTH	C_BD_DWIDTH specifies the width of the data port
MstRd2SG_x		C_DMA_CHAN_TYPE C_RD_HDR_SIZE C_RD_FTR_SIZE	Ports are enabled if at least one channel is configured for Packet SG Read operation, C_DMA_CHANNEL_TYPE = 2 and C_RD_HDR_SIZE and C_RD_FTR_SIZE are greater than zero
SG2MstRd_x		C_DMA_CHAN_TYPE C_RD_HDR_SIZE C_RD_FTR_SIZE	Ports are enabled if at least one channel is configured for Packet SG Read operation, C_DMA_CHANNEL_TYPE = 2 and C_RD_HDR_SIZE and C_RD_FTR_SIZE are greater than zero

Table 3: **Parameter/Port Dependencies *(Cont'd)***

Name	Affects	Depends	Relationship Description
MstWr2SG_x		C_DMA_CHAN_TYPE C_WR_HDR_SIZE C_WR_FTR_SIZE	Ports are enabled if at least one channel is configured for Packet SG Write operation, C_DMA_CHANNEL_TYPE = 3 and C_WR_HDR_SIZE and C_WR_FTR_SIZE are greater than zero
SG2MstWr_x		C_DMA_CHAN_TYPE C_WR_HDR_SIZE C_WR_FTR_SIZE	Ports are enabled if at least one channel is configured for Packet SG Write operation, C_DMA_CHANNEL_TYPE = 3 and C_WR_HDR_SIZE and C_WR_FTR_SIZE are greater than zero

Register-Interface Descriptions

The following section discusses the user application interface to the various registers provided by the DMA SG. Depending on the user parameter assignments, these registers may or may not exist in the user's implementation.

Table 4: **DMA Channel Register Summary**

Register Name	Abbreviation	Address Offset from Service's Base Address Assignment(3)	Access
DMA Status Register	DMASR	dma_chan*0x40 + 0x00	Read
DMA Control Register	DMACR	dma_chan*0x40 + 0x04	Read/Write(1)
Most Significant Bus Address	MSBA	dma_chan*0x40 + 0x08	Read/Write(2)
Least Significant Bus Address	LSBA	dma_chan*0x40 + 0x0C	Read/Write(2)
Buffer Descriptor Address	BDA	dma_chan*0x40 + 0x10	Read/Write(1)
DMA Length	LENGTH	dma_chan*0x40 + 0x14	Read/Write(2)
Interrupt Status Register	ISR	dma_chan*0x40 + 0x18	Read/TOW(4)
Interrupt Enable Register	IER	dma_chan*0x40 + 0x1C	Read/Write
Software Control Register	SWCR	dma_chan*0x40 + 0x20	Read/Write
Reserved		dma_chan*0x40 + 0x24	
Reserved		dma_chan*0x40 + 0x28	
Reserved		dma_chan*0x40 + 0x2C	
Reserved		dma_chan*0x40 + 0x30	
Reserved		dma_chan*0x40 + 0x34	
Reserved		dma_chan*0x40 + 0x38	
Reserved		dma_chan*0x40 + 0x3C	

Notes: Notes:

1. If the channel is designated as a Scatter Gather type channel then these registers become Read Only while Scatter Gather is in operation, (i.e. SWCR.SGE = 1, SWCR.SGD = 0 and DMASR.SGBSY = 1).
2. If the channel is designated as a Scatter Gather type channel then these registers become Read Only.
3. The Base Address is assigned by C_ARD_ADDR_RANGE_ARRAY generic for the OPB or PLB IPIF
4. Toggle each bit position to which a 1 is written.

Channel Register Summary

There are nine channel registers per channel. These are summarized in Table 5.

Table 5: **Channel Registers**

Register		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	DMASR	DMABSY	DBE	DBT	DMADONE	SGBSY	LAST	rsvd	SGDONE	rsvd		DMACNFG		Reserved																			
1	DMACR	AINC	rsvd	BPDRE	rsvd	SGS	LAST	rsvd		Device Select								Reserved				BD Page(1)				rsvd		TYPE		rsvd		DSIZE	
2	MSBA(1)	Reserved																													MSBA		
3	LSBA	Least Significant Bus Address																															
4	BDA(2)	Buffer Descriptor Address																															
5	LENGTH	DMA Transfer Length (in Bytes)																															
6	ISR	Reserved																								DD	DE	PD	PCTR	PWBR	SGDA	SGEND	
7	IER	Reserved																								EDD	EDE	EPD	EPCTR	EPWBR	ESGDA	ESGEND	
8	SWCR(2)	SGE	SGD	DSGAR	rsvd	Reserved				PWB(3) (Packet Wait Bound)												PCT(3) (Packet Count Threshold)										

Notes:
1. MSBA and BD Page are reserved for future growth.
2. BDA and SWCR are only applicable for channels of type SG DMA (C_DMA_CHAN_TYPE = 1, 2, or 3)
3. PWB and PCT are only applicable if interrupt coalescing is enabled, i.e. C_INTERRUPT_COALESCE = 1.

Channel Register BIT Description

Table 6 to Table 13 give the BIT descriptions for the Channel Registers.

Table 6: **DMA Status Register (DMASR)**

Bit(s)	Name	Description	Properties
0	DMABSY	DMA Busy. This bit is set when hardware reaches a point that it recognizes and acts on a newly established LENGTH-not-equal-to-zero condition. There is, in general, some delay from the establishment of the condition to the assertion of DMABSY as DMA processing start for the Channel. Therefore, polling this bit as a test for DMA completion is not recommended. Instead test for DMADONE=1	r
1	DBE	DMA Bus Error	r
2	DBT	DMA Bus Timeout.	r
3	DMADONE	DMA Done. This bit is set when the hardware completes the DMA Cycle. This bit is cleared when a new DMA Cycle is started by writing a new value into the LENGTH register.	r

Table 6: **DMA Status Register (DMASR)**

Bit(s)	Name	Description	Properties
4	SGBSY	Scatter Gather Busy. This bit is set when hardware reaches a point that it recognizes and acts on a newly established SGE=1, SGD=0, and SGS = 0 condition. There is, in general, some delay from the establishment of the condition to the assertion of SGBSY as SG processing starts for the channel. Therefore, polling this bit as a test for SG completion is not recommended. Instead, test for SGDONE = 1 or test for SGE=0, because SGE is cleared by hardware as SG completes.	r
5	LAST	Last. This bit indicates that the buffer associated with this Buffer Descriptor contains the last byte of the received packet.	r
6		Reserved - read as zero.	r
7	SGDONE	Scatter Gather Done. This bit is set when the hardware completes the SG Cycle if Simple SG or if Packet SG then this bit is set when the Packet is completed.	r
8 - 9		Reserved - read as zero.	r
10 - 11	DMACNFG	DMA Channel Configuration. 00 = Simple DMA, 01 = Simple SG, 10 = Packet Transmit SG, and 11 = Packet Receive SG	r
12 - 31		Reserved - read as zero.	r

Table 7: **DMA Control Register (DMACR)**

Bit(s)	Name	Description	Properties
0	AINC	Address Increment. During DMA operation, increment Address by one for each byte transferred.	rw
1		Reserved - read as zero. Writing has no effect.	rw
2	BPDRE	Bypass DRE: 1 = Bypass Data Realignment Engine. 0 = Do NOT bypass Data Realignment Engine.	rw
3		Reserved - read as zero. Writing has no effect.	r
4	SGS	Scatter Gather Stop. SGS is used to tell the SG automation to stop as soon as a stop condition holds. For simple SG channels, the stop condition is established by completion of the buffer descriptor for which the SGS bit is set. For packet SG channels, the stop condition is established by completion of all partially completed packets.	rw
5	LAST	Last. This bit indicates to the hardware that the buffer associated with this descriptor represents the last of a packet. This bit is relevant for when a packet is being sourced, or transmitted, from the buffer.	rw
6 - 7		Reserved - read as zero. Writing has no effect.	r
8 - 15	Device Select	Specifies to the UserIP, which device the current DMA transfer is targeting.	rw
16 - 19		Reserved - read as zero. Writing has no effect.	rw
20 - 23	BD Page	Reserved for future growth.	rw
24		Reserved - read as zero. Writing has no effect.	rw

Table 7: **DMA Control Register (DMACR) *(Cont'd)***

Bit(s)	Name	Description	Properties
25 - 27	Type	Instructs the IPIF Master as to what type of bus transfer should be made, 000=Reserved 001=Bounded Fixed Length Burst - Instructs the Master Attachment to perform fixed length burst on the host bus whenever possible. 010=Bounded Indeterminate Burst - Instructs the Master Attachment to perform indeterminate bursts on the host bus whenever possible. 011=Reserved 100=Reserved 101=Reserved 110=Reserved 111= Reserved	rw
28		Reserved - read as zero. Writing has no effect.	rw
29 - 31	DSize	Specifies the width of the DMA transfer. 000=byte (8 bits), 001=half_word (16 bits), 010=word (32 bits), 011=double-word (64 bits), 100=quadword (128 bits), etc.	rw

Table 8: **Bus Address (MSBA and LSBA)**

Bit(s)	Name	Description	Properties
0 to 27	MSBA	Reserved - Read as 0	r
28 to 31	MSBA	Reserved for future growth	rw
0 to 31	LSBA	Least Significant portion of bus address for DMA	rw

Table 9: **Buffer Descriptor Address (BDA)**

Bit(s)	Name	Description	Properties
0 to 31	BDA	Buffer Descriptor Address for DMA. This value points to the next buffer descriptor to be processed by the DMA Controller.	rw

Table 10: **Length (LENGTH)**

Bit(s)	Description	Properties
0 to 31(1)	This register is used to specify the requested number of bytes to transfer. For channels of a Simple DMA type, writing a non-zero value to this register allows the DMA operation to start, so the register should be written last when setting up a DMA operation.	rw

Note: Note:

1. The length value occupies the least significant C_DMA_LENGTH_WIDTH bits of the register. For example if C_DMA_LENGTH_WIDTH = 10 then the valid length bits of the register are bits 22 to 31. Bits 0 to 21 would be ignored.

Table 11: **Interrupt Status Register (ISR)**

Bit(s)	Name	Description	Properties
0 - 24		Reserved - read as zero. Writing has no effect.	r
25	DD	DMA Done Event. A DMA operation has finished either normally or with an error.	r/TOG
26	DE	DMA Error Event. A DMA operation has finished with an error. The channel and any associated FIFOs or devices must be re-initialized.	r/TOG

Table 11: **Interrupt Status Register (ISR) *(Cont'd)***

Bit(s)	Name	Description	Properties
27	PD	Packet Done event. The handling of a packet has been completed.	r/TOG
28	PCTR	Packet Count Threshold Reached event. The number of unreported packets has reached the Packet Count Threshold.	r/TOG
29	PWBR	Packet Wait Bound Reached event. The Packet Wait Bound has been reached before the Packet Count Threshold was reached.	r/TOG
30	SGDA	SG Disable Acknowledge event. SG operation has stopped as a result of SWCR.SGD = 1	r/TOG
31	SGEND	SG End event. Sg operation has stopped as a result of finishing packet when DMACR.SGS =1 on the last BD.	r/TOG

Table 12: **Interrupt Enable Register (IER)**

Bit(s)	Name	Description	Properties
0 - 24		Reserved - read as zero. Writing has no effect.	r
25	EDD	Enable DMA Done Event.	rw
26	EDE	Enable DMA Error Event.	rw
27	EPD	Enable Packet Done event.	rw
28	EPCTR	Enable Packet count Threshold Reached event.	rw
29	EPWBR	Enable Packet Wait Bound Reached event.	rw
30	ESGDA	Enable SG Disable Acknowledge event.	rw
31	ESGEND	Enable SG End event.	rw

Table 13: **Software Control Register (**SWCR**)**

Bit(s)	Name	Description	Properties
0	SGE	Scatter Gather Enable. SG automation does not begin unless SGE=1 and DMACR.SGS=0. SG automation can be stopped while active by setting SGD=1. SGE will be automatically cleared to 0 when SGD=1 and stop conditions have been reached. See <RD Red>"Stopping and Starting SG Operation" on page 12. SGE will be automatically set to '1' when SGD is cleared to '0' and DSGAR = '0' facilitating the insertion or appending of a new BD chain to an already active BD chain. (See <RD Red>"Inserting or Appending A Buffer Descriptor Chain" on page 41)	rw
1	SGD	Scatter Gather Disable. Setting SGD=1 will disable SG automation. SG Automation will stop with the completion of the current Buffer Descriptor if the channel is defined as Simple SG (C_DMA_CHAN_TYPE=1) or at the end of the packet for Packet SG (C_DMA_CHAN_TYPE=2 or 3). Whenever SG automation stops, whether caused by SGS=1 or SGD=1, SGE is cleared by the hardware.	rw
2	DSGAR	Disable Scatter Gather Auto-Restart. Setting DSGAR=1 will disable SG process from auto-restarting when SGD negates from 1 to 0. Setting DSGAR=0 will allow the SG process to auto-restart when SGD transitions from a 1 to a 0. This bit is applicable when SG processes have halted as a result of SGD being set to '1' and facilitates in allowing the software application to insert or append a new BD chain into an already active BD chain. (See <RD Red>"Inserting or Appending A Buffer Descriptor Chain" on page 41)	rw

Table 13: **Software Control Register (**SWCR**) *(Cont'd)***

Bit(s)	Name	Description	Properties
2 - 7		Reserved - read as zero. Writing has no effect.	rw
8 - 19	PWB[1]	Packet Wait Bound. The maximum amount of time that an unreported packet is required to wait until unreported packets are reported via a PWBR event. (Must remain unchanged for the duration of an SG operation.)	rw
20 - 31	PCT[1]	Packet Count Threshold. The number of unreported packets that must accumulate before they are reported via a PCTR event. (Must remain unchanged for the duration of an SG operation.)	rw

Notes:
1. Only valid if C_INTR_COALESCE=1; otherwise, these fields are ignored.

User Application Topics

Assert Error Messages

A mechanism has been put into place to help users locate invalid configurations of the DMA SG that may still compile successfully, yet will not function correctly. This is achieved using 'assert' logic that will abort simulation loading if an invalid configuration is specified for the DMA SG. The following is a description of the possible error messages.

Unequal Array Elements

ERROR - Unequal Array Elements: The number of elements in C_SPLIT_BUS_TYPE does NOT equal the number of elements in C_DMA_CHAN_TYPE."

The number of elements specified in C_DMA_CHAN_TYPE array do not match the number of elements specified in C_SPLIT_BUS_TYPE. For the Split bus mode, there must be the same amount of elements specified in each respective array. If there are two channels specified in C_DMA_CHAN_TYPE then there must be two elements specified in C_SPLIT_BUS_TYPE.

Mismatch Array Elements

ERROR - Mismatch Array Elements: Scatter Gather types in C_DMA_CHAN_TYPE do NOT match transmit and receive types specified in C_SPLIT_BUS_TYPE.

This error occurs when the transmit and receive SG channels specified in C_DMA_CHAN_TYPE do not match the transmit and receive specifiers in *C_SPLIT_BUS_TYPE*. For example, if channel 0 (array index 0) of C_DMA_CHAN_TYPE is specified to be a SG Transmit Packet type and channel 0 (array index 0) of C_SPLIT_BUS_TYPE is specified to be a Receive channel then this will produce a conflict. In the Split Bus configuration a channel cannot be both a transmit and receive.

Invalid Rx Channel Number

ERROR - Invalid RX Channel Number: C_NUM_RX_CHANNELS must equal the number of receive channels specified in C_SPLIT_BUS_TYPE.

This error occurs if the number of receive channels indicated in C_NUM_RX_CHANNELS does not equal the number of receive channels indicated in C_SPLIT_BUS_TYPE.

Invalid Tx Channel Number

ERROR - Invalid TX Channel Number: C_NUM_TX_CHANNELS must equal the number of transmit channels specified in C_SPLIT_BUS_TYPE.

This error occurs if the number of transmit channels indicated in C_NUM_TX_CHANNELS does not equal the number of transmit channels indicated in C_SPLIT_BUS_TYPE.

Invalid Address Width

ERROR - Invalid Address Width: IPIC Address width NOT equal to 32.

This error occurs if C_IPIF_AWIDTH is set to anything other than 32. This version of the core does not support a Bus2IP_Addr width of anything but 32.

Invalid Data Width

ERROR - Invalid Data Width: IPIC data width NOT equal to 32 or 64.

This error occurs if C_IPIF_DWIDTH is set to anything other than 32 or 64. This version of the core does not support a Bus2IP_Data width of anything but 32 or 64 bits.

Buffer Descriptor Update During Packet Scatter Gather Operation

At times it may be necessary to update a Buffer Descriptor chain while the DMA Controller is actively processing the Buffer Descriptors. Two types of active Buffer Descriptor updates are possible. The first is referred to as Active Buffer Descriptor Additions and the second is Active Buffer Descriptor Modification.

Active Buffer Descriptor Detains is the process of adding Buffer Descriptors to an active chain without stopping the chain. See <RD Red>"Active Buffer Descriptor Additions" on page 41.

Active Buffer Descriptor Modification is the process of simply moving the stop point in a buffer descriptor chain without first stopping the chain. This method works well for Buffer Descriptors arranged in a loop configuration. See <RD Red>"Active Buffer Descriptor Chain Modification" on page 45.

The following sections discuss both methods for Buffer Descriptor updates and illustrates the interaction between the software application and hardware.

Active Buffer Descriptor Additions

Active Buffer Descriptor Additions is the process of adding Buffer Descriptors to an already active Buffer Descriptor chain without the software application halting the chain before doing the additions.

Inserting or Appending A Buffer Descriptor Chain

The DMA Controller has been designed such that the software application can insert or append new Buffer Descriptor chains into an already active chain with minimal effort on the part of the software application. The following is a list of steps the software application must take in order to insert or append a new BD chain into an active BD chain.

These steps assume that the new BD chain is already created in remote memory and that the DMA Controller is already actively processing the original BD chain.

1. Set the Scatter Gather disable bit in the Software Control Register to a 1. (Set SWCR.SGD = 1). By setting SGD=1 the software application places a temporary pause point at the end of the currently active packet. When the hardware reaches a BD with DMACR.LAST = '1' on TX or reaches the end of a receive packet the hardware will stop processing buffer descriptors and pause to allow software to complete the BD Chain update.
2. Read the BDA Register in the DMA controller to verify which packet the DMA Controller is currently processing. Buffer descriptors cannot be modified within the same packet that hardware

is currently processing. They can only be modified on BD packets that exist in the chain after the current active packet. In other words if the DMA Controller is already past the point at which you will be inserting the new BD chain then it is too late to insert the BD chain at that location.

3. Update the BDA pointer of the last BD in the packet to which you wish to insert the new BD chain, with the address of the first BD in the new BD chain.
4. Channel Register BDA Update **(Special care must be taken at this step)**

   If the new BD chain is being attached to the end of a packet that is currently active then the BDA Channel Register in the DMA controller will need to be updated with the address of the first BD of the new BD chain.

   To do this the software application MUST wait for the DMA SG to halt (i.e. SWCR.SGE = 0 or receipt of SGDA interrupt).

   If the new BD chain is being inserted or appended to a packet that has not and is not currently being processed by the DMA Controller then do **NOT** update the BDA Channel Register.
5. Re-enable Scatter Gather processing
   a. If SG Automatic Restart is enabled (SWCR.DSGAR=0) then set the Scatter Gather disable bit in the Software Control Register to a 0, (Set SWCR.SGD = 0) to allow the DMA Controller to continue SG processing.
   b. If SG Automatic Restart is disabled (SWCR.DSGAR=1) then first verify that SG process as halted either by receiving SGDA interrupt or by polling SWCR looking for SWCR.SGE=0. Once SG processing has halted then it may be restarted by establishing the conditions SWCR.SGE=1, SWCR.SGD=0, and SWCR.SGS=0 (See <RD Red>"Stopping and Starting SG Operation" on page 12)

At this point the new BD chain will be inserted or appended to the active BD chain.

Buffer Descriptor Chain Insert Example

As an example, assume the software application has set up the following Buffer Descriptor Chain, BD0 to BD9 as can be seen in Figure 9. Assume, also, that the software application has started DMA SG operation by writing the address of BD 0 to the BDA Register in the DMA Controller, set SWCR.SGE = 1 and set SWCR.SGD = 0. The DMA SG Controller will fetch BD 0 and BD 1 and begin processing BD 0.

Two types of additions are wanted, the first being an insert of new Buffer Descriptors and the second being an appending of Buffer Descriptors to the end of an active chain. From a software perspective the steps for both cases are the same.

For the first type of chain update consider the insertion of Buffer Descriptors 3A,3B, and 3C into the chain as shown in Figure 9, the software application would walk through steps <RD Red>[1] to <RD Red>[5]. From the DMA Controller's perspective several situations or cases can occur. The following sections describe these cases and illustrate how the hardware handles each.

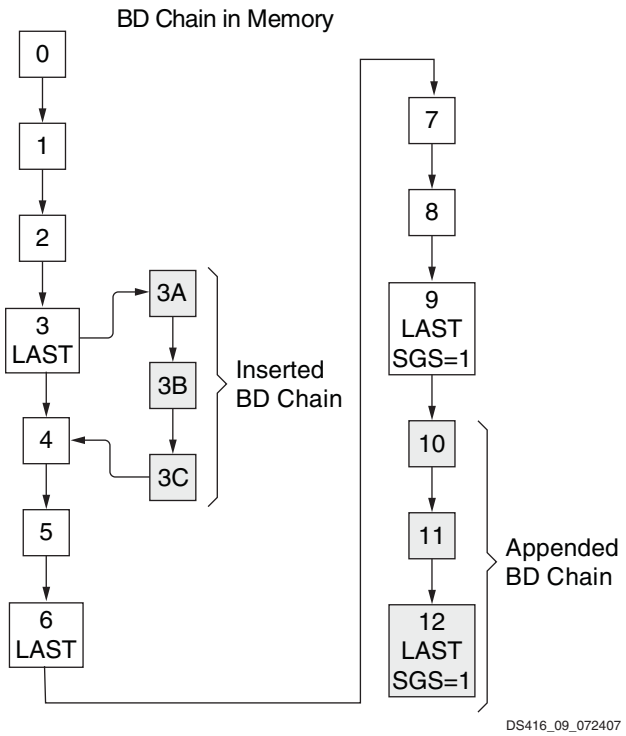

Figure 9: **Buffer Descriptor Chain Insert**

CASE 1 - BD Insert During Process of a Single BD

One possible case that could occur, is software sets SWCR.SGD=1, does all updates, and then sets SWCR.SGD=0 before the DMA SG Controller has completed processing of the current Buffer Descriptor. Figure 10 shows this case. The DMA SG Controller has already processed BD 0, is currently processing BD 1. The blue lines (horizontal bold lines) mark out a window indicating the time in which the software application has set SWCR.SGD = 1, performed its updates, and then resets SWCR.SGD = 0. In this situation the DMA Controller is not even aware that an update occurred.

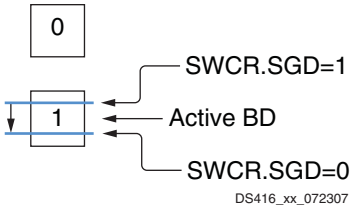

Figure 10: **BD Insert During Process of a Single BD**

CASE 2 - BD Insert During Processing of Multiple BD's

Another case is where software sets SWCR.SGD = 1 during the time the DMA Controller is processing BD0 and the DMA Controller completes BD0, begins processing BD1 before the software application can clear SWCR.SGD to 0. From the hardware perspective this is the same case as CASE1. Because processing is not at the end of the packet, fetches and dma transfers are uninterrupted. It is due to these

cases that it is important that the software application only update bd's on packets that are currently not being processed. Figure 11shows this case.

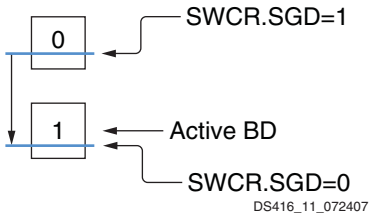

Figure 11: **Update Spanning Two BDs**

CASE 3 - BD Update Extending Beyond End of Packet

Case 3, shown in Figure 12, illustrates when the software update starts when the DMA Controller is process the last BD in a packet and extends beyond when the DMA Controller has completed process the BD. In this case the DMA Controller will halt processing of the BD chain until the software application clears SWCR.SGD to 0. At this point the hardware will fetch the new BD, BD3A, pointed to by the address loaded in the Channel Register BDA.

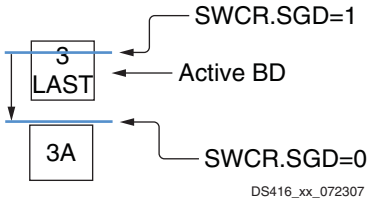

Figure 12: **Update Extending Beyond the End of a Packet**

CASE 4 - BD Append At End of Packet

For the following cases, the user wants to append BDs 10 through 12 onto the end of an active BD chain. The special cases occur when the appending takes place when the DMA Controller is already processing the last BD of a packet. In Case 4, see Figure 13, the software sets SWCR.SGD = 1, performs its updates, and then sets SWCR.SGD = 0, before the DMA Controller finishes processing of BD9. For this case the DMA process never stops. As soon as SWCR.SGD is set back to 0, DMACR.SGS is cleared to 0 and when the DMA Controller finishes BD9, it will continue to process the chain by fetching BD10 as pointed to by the Channel Register BDA.

In this case, the DMA process never stops. As soon as SGD-0, SGS is set to 0 and process continues processing the appended BDs pointed out by the Channel Register BDA.

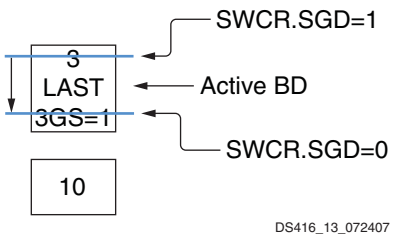

Figure 13: **BD Append At End of Packet**

CASE 5 - BD Append Extending Beyond End of Packet

Case 5, shown in Figure 14, illustrates the case where the software update extends in time beyond the last of a packet. The DMA Controller will halt, because SWCR.SGD is set to 1, and wait until the software application has completed its updates. When the software application sets SWCR.SGD = 0, if SWCR.DSGAR=0, then DMACR.SGS bit will be cleared, SWCR.SGE will be set to 1 and processing will begin again with the DMA Controller fetching BD10 as pointed to by the Channel Register BDA.

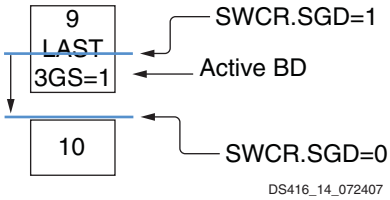

Figure 14: **BD Append Extending Beyond End of Packet**

Active Buffer Descriptor Chain Modification

Active Buffer Descriptor Chain Modification is the process of simply modifying the chain. With this method no new buffer descriptors are added. This method works well for buffer descriptors configured into a loop. Consider a situation where you have 14 buffer descriptors arranged in a loop, Figure 15, and you want to move the stop point from Buffer Descriptor 9, BD9, to Buffer Descriptor 13, BD13, without stopping the chain. The steps to perform such an operation are listed below.

These steps assume that the new BD chain is already created in remote memory and the DMA Controller is already actively processing the original BD chain.

1. Set the Scatter Gather disable bit in the Software Control Register to a 1. (Set SWCR.SGD = 1). It should be noted that at this point the DMA Controller will be in one of two states. If the DMA controller reaches a stop point the DMA controller will pause operations and SWCR.SGE = '0'. If the DMA Controller does not reach a stop point the DMA Controller will continue processing BD's and SWCR.SGE = '1'. Stop points are defined as 'end of packets' i.e. BD with L=1.
2. Read the BDA register in the DMA controller to verify that the DMA Controller is processing a buffer descriptor in the packet prior to the packet to which you wish to modify. In other words if the DMA Controller is already past the point at which you will be modifying the BD's then, in a loop configuration, the DMA controller will not process the modified BD's until it comes around the loop again.
3. Modified the Buffer Descriptors in remote memory. For example, set SGS=0 in BD9 and set SGS=1 in BD13.
4. Re-enable Scatter Gather processing
    a. If SG Automatic Restart is enabled (SWCR.DSGAR=0) then set the Scatter Gather disable bit in the Software Control Register to a 0, (Set SWCR.SGD = 0) to allow the DMA Controller to continue SG processing.
    b. If SG Automatic Restart is disabled (SWCR.DSGAR=1) then first verify that SG process as halted either by receiving SGDA interrupt or by polling SWCR looking for SWCR.SGE=0. Once SG processing has halted then it may be restarted by establishing the conditions SWCR.SGE=1,

SWCR.SGD=0, and SWCR.SGS=0 (See <RD Red>"Stopping and Starting SG Operation" on page 12).

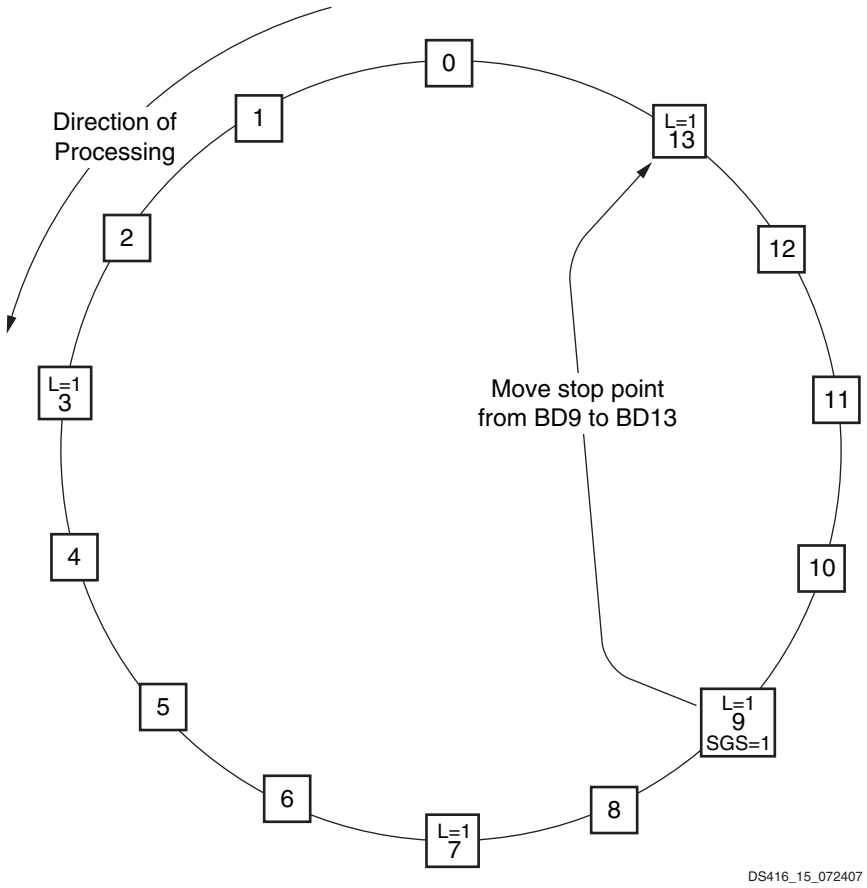

Figure 15: **Buffer Descriptor Chain - Loop Configuration**

Several cases may occur during the BD Chain modification. The following illustrates the interaction between hardware and software during modification.

CASE 1: Modifying BD Being Processed

This special cases occur when the modification takes place when the DMA Controller is already processing the last BD of a packet. The software sets SWCR.SGD = 1, performs its updates, and then sets SWCR.SGD = 0, before the DMA Controller finishes processing of BD9. For this case the DMA process never stops. As soon as SWCR.SGD is set back to 0, DMACR.SGS is cleared to 0 and when the DMA Controller finishes BD9, it will continue to process the chain by fetching BD10 as pointed to by the Channel Register BDA. If during the modification the software application read the software control register, both SGE and SGD will be set to a '1' indicating that the DMA Controller did not reach a stopping point.

In this case, shown in Figure 16, the DMA process never stops. As soon as SGD=0, SGS will be set to 0 and the process continues by processing the appended BDs pointed to by the Channel Register BDA.

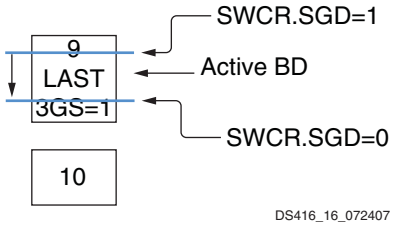

Figure 16: **BD Modified At End of Packet**

Case 2: BD Modification Extending Beyond End of Packet

Case 2, shown in Figure 17, illustrates the case where the software modification extends in time beyond the last of a packet. The DMA Controller will pause, because SWCR.SGD is set to 1, and wait until the software application has completed its updates. When the software application sets SWCR.SGD = 0, the DMACR.SGS bit will be cleared, SWCR.SGE will be set to 1 and processing will begin again with the DMA Controller fetching BD10 as pointed to by the Channel Register BDA. If during the modification the software application read the software control register, SGE would be set to a '0' and SGD would be set to a '1' indicating that the DMA Controller did reach a stopping point.

Because SGD=1, the DMA Controller will halt, even if SGS=1. As soon as SGD=0, SGS will be set to 0, SGE will be set to 1, and the process will start again, fetching the appended BD pointed to by the Channel Register BDA.

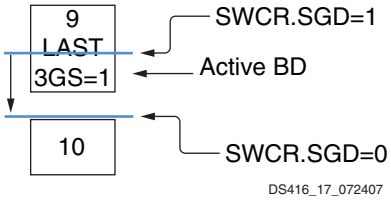

Figure 17: **BD Append Extending Beyond End of Packet**

CASE 3 - BD Modified During Processing of Multiple BD's

Another case, shown in Figure 18, is where software sets SWCR.SGD = 1 during the time the DMA Controller is processing BD0 and the DMA Controller completes BD0, begins processing BD1, before the software application can clear SWCR.SGD to 0. BD2 is flushed in case the software application modified BD2 and when the DMA Controller completes processing BD1, it will refetch BD2.

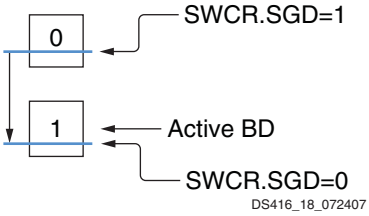

Figure 18: **Update Spanning Two BDs**

FPGA Design Application Hints

Single Entry in Unconstrained Array Parameters

As has been discussed previously, the DMA parameterization employes generics that are defined as unconstrained arrays, i.e. arrays whose size is left unbound at declaration and fixed later by the user. This is the underlying VHDL mechanism that allows the DMA to grow or shrink to the size required by the application. Generally, the size of the unconstrained array and its element values are fixed simultaneously by the user by assigning to the array a constant aggregate. The aggregate is simply a list of values enclosed in parentheses and separated by commas.

The list can take either of two forms, *positional association*, in which the values at indices in the array are populated by the aggregate elements as they appear, left to right, or *named association*, in which each value populates an index to which it is explicitly assigned by being proceeded by "INDX =>". Thus, the following positional and named aggregates are identical: (4,3,9) and (0=>4, 1=>3, 2=>9).

The user should be aware that for aggregates with a single element, positional association is unfortunately not allowed. The reason for this is that otherwise the syntax would be ambiguous with a parenthesized expression. Being aware of this VHDL restriction might save the user some aggravation should this usage case come up. The following example shows both the incorrect and the correct way to associate a single element to an unconstrained array.

C_DMA_CHAN_TYPE **=> (0);** --VHDL **positional association** NOT allowed because it would be ambiguous.

C_DMA_CHAN_TYPE => (0=> 0); -- VHDL **named association** instead.

Design Implementation

Target Technology

This DMA SG Controller is targeted for the Virtex-II Pro and Virtex-4-FX devices

Device Utilization and Performance Benchmarks

The DMA benchmarks are shown in Table 14 for a Virtex-2P -7 FPGA.

Table 14: **DMA FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7)**

Parameter Values							Device Resources			
Number of Channels	C_DMA_CHAN_TYPE	C_INTR_COALESCE(3)	C_RD_HDR_SIZE(2) (Bytes)	C_RD_FTR_SIZE(2) (Bytes)	C_WR_HDR_SIZE(2) (Bytes)	C_WR_FTR_SIZE(2) (Bytes)	Slices	Slice Flip-Flops	4-input LUTs	fMAX_REG(1)
1	0	NA	NA	NA	NA	NA	185	216	176	173.4

Table 14: **DMA FPGA Performance and Resource Utilization Benchmarks (Virtex-II Pro -7)**

1	1	NA	NA	NA	NA	NA	361	417	383	146.4
1	2	0	16	8	NA	NA	551	640	569	126.6
1	2	1	16	8	NA	NA	590	682	635	139
2	0,0	NA	NA	NA	NA	NA	313	353	314	143.5
2	1,1	NA	NA	NA	NA	NA	623	693	708	129.7
2	2,3	1	0	0	0	8	1013	1060	1215	127.6
2	2,3	1	16	0	0	8	1075	1145	1274	130.4
2	2,3	0	16	8	16	8	995	1083	1163	128.0
2	2,3	1	16	8	16	8	1089	1160	1296	132.1

Notes:

1. Fmax represents the maximum frequency of the DMA service in a standalone configuration. The actual maximum frequency will depend on the entire system and may be greater or less than what is recorded in this table. Thus Fmax should be used purely as a reference and a rough measure of the relative affects various configurations have on the operating frequency.
2. Header and Footer sizes are only applicable for Scattter Gather Channels.
3. Interrupt Coalescing is only valid for Packet Scatter Gather Channels.

Specification Exceptions

None

DMA Signaling Exceptions

The following signals are not used by this version of DMA Controller, dma_sg_v2_01_a.

IPIC

- Bus2IP_Freeze

Buffer Descriptor Interface

- Bus2SG_CmdRd_RdCnt
- Bus2SG_CmdWr_WrCnt
- Bus2SG_CmdRd_Rearbitrate
- Bus2SG_CmdWr_Rearbotrate
- Bus2SG_CmdRd_rem
- SG2Bus_CmdRd_Lock
- SG2Bus_CmdWr_Lock

DMA Command Interface

- Bus2DMA_CmdRd_Rearbitrate
- Bus2DMA_CmdWr_Rearbitrate
- Bus2DMA_CmdRd_Length_is_Zero
- Bus2DMA_CmdWr_Length_is_Zero
- DMA2Bus_CmdRd_DRE_Push2FIFO
- DMA2Bus_CmdWr_DRE_Push2FIFO
- DMA2Bus_CmdRd_Lock

- DMA2Bus_CmdWr_Lock

Reference Documents

The following documents contain supplemental information that might be of interest.

- *IPSPEC001 Virtex-II Pro HDL Coding Guidelines*
- *IPSPEC002 Virtex-II Pro Standard Abbreviations*
- *IBM CoreConnect64-Bit On-Chip Peripheral Bus, Architectural Specification (v2.x)*
- *IBM CoreConnect64-Bit Processor Local Bus, Architectural Specification (v3.x)*
- *PLB Master with LocalLink V3 (DS459 v1.x)*
- *Xilinx LogiCORE LocalLink interface specification*

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
2/17/06	1.0	Initial Xilinx release
7/25/07	1.1	Converted to current DS template; updated images to graphic standard.
4/24/09	1.5	Updated supported tools. Removed unsupported devices: Virtex-II Pro.

Notice of Disclaimer