Chapter 8 DMA Controller (DMAC)

8.1 Overview

DMAC does not support TrustZone technology and work under non-secure state only.

DMAC is mainly used for data transfer of the following slaves: I2S0, SD/MMC, SDIO, eMMC,UART0, UART1, UART2, SPI0.

Following table shows the DMAC request mapping scheme.

Table 8-1DMAC Request Mapping Table

Req number	Source	Polarity
0	I2S_2ch_ tx	High level
1	I2S_2ch_ rx	High level
2	UART0 tx	High level
3	UART0 rx	High level
4	UART1 tx	High level
5	UART1 rx	High level
6	UART2 tx	High level
7	UART2 rx	High level
8	SPI tx	High level
9	SPI rx	High level
10	SD/MMC	High level
11	SDIO	High level
12	eMMC	High level
13	SPDIF	High level
14	I2S_8ch tx	High level
15	I2S_8ch rx	High level

DMACsupports the following features:

- Supports 15 peripheral request.
- Up to 64bits data size.
- 8 channel at the same time.
- Up to burst 16.
- 1 interrupts output and 1 abort output.
- Supports 32 MFIFO depth.

8.2 Block Diagram

Figure 8-1 shows the block diagram of DMAC.

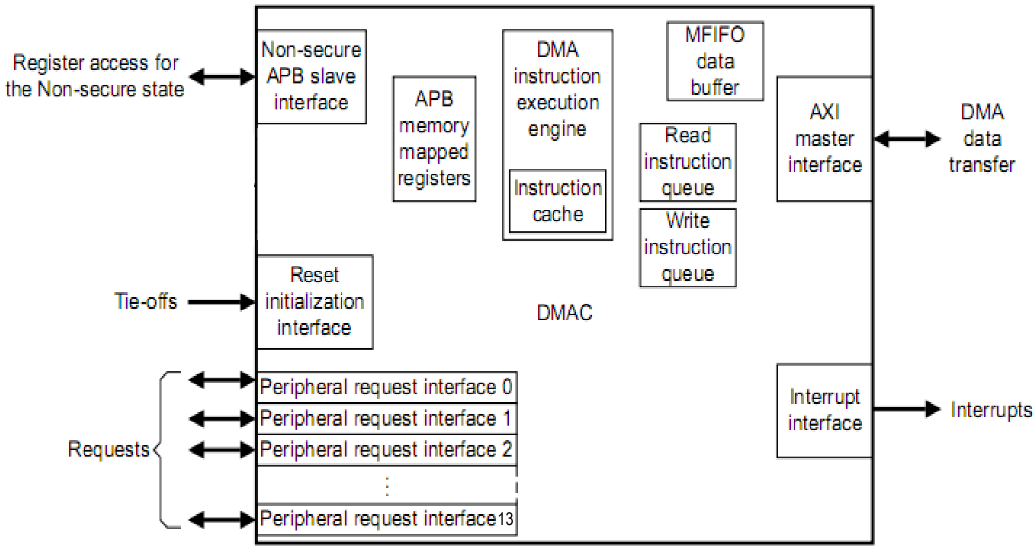

Fig. 8-1 Block diagram of dmac

8.3 Function Description

8.3.1 Introduction

The DMAC contains an instruction processing block that enables it to process program code that controls a DMA transfer. The program code is stored in a region of system memory that the DMAC accesses using its AXI interface. The DMAC stores instructions temporarily in a cache.

DMAC supports 8 channels,each channel capable of supporting a single concurrent thread of DMA operation. In addition, a single DMA manager thread exists, and you can use it to initialize the DMA channel threads. The DMAC executes up to one instruction for each AXI clock cycle. To ensure that it regularly executes each active thread, it alternates by processing the DMA manager thread and then a DMA channel thread. It uses a round-robin process when selecting the next active DMA channel thread to execute.

The DMAC uses variable-length instructions that consist of one to six bytes. It provides a separate Program Counter (PC) register for each DMA channel. When a thread requests an instruction from an address, the cache performs a look-up. If a cache hit occurs, then the cache immediately provides the data. Otherwise, the thread is stalled while the DMAC uses the AXI interface to perform a cache line fill. If an instruction is greater than 4 bytes, or spans the end of a cache line, the DMAC performs multiple cache accesses to fetch the instruction.

When a cache line fill is in progress, the DMAC enables other threads to access the cache, but if another cache miss occurs, this stalls the pipeline until the first line fill is complete.

When a DMA channel thread executes a load or store instruction, the DMAC adds the instruction to the relevant read or write queue. The DMAC uses these queues as an instruction storage buffer prior to it issuing the instructions on the AXI bus. The DMAC also contains a Multi First-In-First-Out (MFIFO) data buffer that it uses to store data that it reads, or writes, during a DMA transfer.

8.3.2 Operating states

Figure shows the operating states for the DMA manager thread and DMA channel threads.

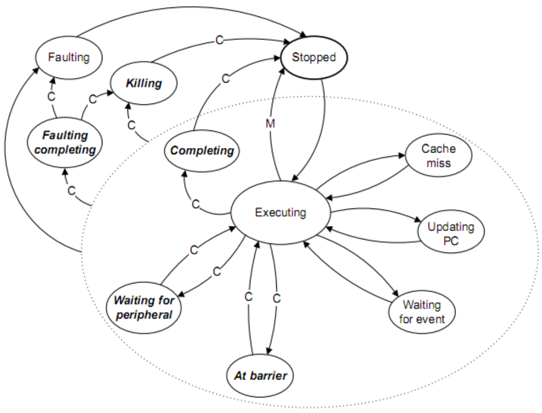

Fig. 8-2 DMAC operation states

Note:
arcs with no letter designator indicate state transitions for the DMA manager and DMA channel threads, otherwise use is restricted as follows:
C DMA channel threads only.
M DMA manager thread only.

After the DMAC exits from reset, it sets all DMA channel threads to the stopped state, and the status of boot_from_pc(tie-off interface of dmac) controls the DMA manager thread state:

boot_from_pc is LOW :DMA manager thread moves to the Stopped state.
boot_from_pc is HIGH :DMA manager thread moves to the Executing state.

8.4 Register Description

8.4.1 Register summary

Name	Offset	Size	Reset Value	Description
DMAC_DSR	0x0000	W	0x0	DMA Status Register.
DMAC_DPC	0x0004	W	0x0	DMA Program Counter Register.
-	-	-	-	reserved
DMAC_INTEN	0x0020	W	0x0	Interrupt Enable Register
DMAC_EVENT_RIS	0x0024	W	0x0	Event Status Register.
DMAC_INTMIS	0x0028	W	0x0	Interrupt Status Register
DMAC_INTCLR	0x002C	W	0x0	Interrupt Clear Register
DMAC_FSRD	0x0030	W	0x0	Fault Status DMA Manager Register.
DMAC_FSRC	0x0034	W	0x0	Fault Status DMA Channel Register.
DMAC_FTRD	0x0038	W	0x0	Fault Type DMA Manager Register.
-	-	-	-	reserved

DMAC_FTR0	0x0040	W	0x0	Fault type for DMA Channel 0
DMAC_FTR1	0x0044	W	0x0	Fault type for DMA Channel 1
DMAC_FTR2	0x0048	W	0x0	Fault type for DMA Channel 2
DMAC_FTR3	0x004C	W	0x0	Fault type for DMA Channel 3
DMAC_FTR4	0x0050	W	0x0	Fault type for DMA Channel 4
DMAC_FTR5	0x0054	W	0x0	Fault type for DMA Channel 5
DMAC_FTR6	0x0058	W	0x0	Fault type for DMA Channel 6
DMAC_FTR7	0x005C	W	0x0	Fault type for DMA Channel 7
-	-	-	-	reserved
DMAC_CSR0	0x0100	W	0x0	Channel Status for DMA Channel 0
DMAC_CSR1	0x0108	W	0x0	Channel Status for DMA Channel 1
DMAC_CSR2	0x0110	W	0x0	Channel Status for DMA Channel 2
DMAC_CSR3	0x0118	W	0x0	Channel Status for DMA Channel 3
DMAC_CSR4	0x0120	W	0x0	Channel Status for DMA Channel 4
DMAC_CSR5	0x0128	W	0x0	Channel Status for DMA Channel 5
DMAC_CSR6	0x0130	W	0x0	Channel Status for DMA Channel 6
DMAC_CSR7	0x0138	W	0x0	Channel Status for DMA Channel 7
DMAC_CPC0	0x0104	W	0x0	Channel PC for DMA Channel 0
DMAC_CPC1	0x010c	W	0x0	Channel PC for DMA Channel 1
DMAC_CPC2	0x0114	W	0x0	Channel PC for DMA Channel 2
DMAC_CPC3	0x011c	W	0x0	Channel PC for DMA Channel 3
DMAC_CPC4	0x0124	W	0x0	Channel PC for DMA Channel 4
DMAC_CPC5	0x012c	W	0x0	Channel PC for DMA Channel 5
DMAC_CPC6	0x0134	W	0x0	Channel PC for DMA Channel 6
DMAC_CPC7	0x013c	W	0x0	Channel PC for DMA Channel 7
DMAC_SAR0	0x0400	W	0x0	Source Address for DMA Channel 0
DMAC_SAR1	0x0420	W	0x0	Source Address for DMA Channel 1
DMAC_SAR2	0x0440	W	0x0	Source Address for DMA Channel 2
DMAC_SAR3	0x0460	W	0x0	Source Address for DMA Channel 3
DMAC_SAR4	0x0480	W	0x0	Source Address for DMA Channel 4
DMAC_SAR5	0x04A0	W	0x0	Source Address for DMA Channel 5
DMAC_SAR6	0x04C0	W	0x0	Source Address for DMA Channel 6
DMAC_SAR7	0x04E0	W	0x0	Source Address for DMA Channel 7
DMAC_DAR0	0x0404	W	0x0	Dest Address for DMAChannel 0
DMAC_DAR1	0x0424	W	0x0	Dest Address for DMAChannel 1

DMAC_DAR2	0x0444	W	0x0	Dest Address for DMAChannel 2
DMAC_DAR3	0x0464	W	0x0	Dest Address for DMAChannel 3
DMAC_DAR4	0x0484	W	0x0	Dest Address for DMAChannel 4
DMAC_DAR5	0x04A4	W	0x0	Dest Address for DMAChannel 5
DMAC_DAR6	0x04C4	W	0x0	Dest Address for DMAChannel 6
DMAC_DAR7	0x04E4	W	0x0	Dest Address for DMAChannel 7
DMAC_CCR0	0x0408	W	0x0	Channel Control for DMA Channel 0
DMAC_CCR1	0x0428	W	0x0	Channel Control for DMA Channel 1
DMAC_CCR2	0x0448	W	0x0	Channel Control for DMA Channel 2
DMAC_CCR3	0x0468	W	0x0	Channel Control for DMA Channel 3
DMAC_CCR4	0x0488	W	0x0	Channel Control for DMA Channel 4
DMAC_CCR5	0x04a8	W	0x0	Channel Control for DMA Channel 5
DMAC_CCR6	0x04c8	W	0x0	Channel Control for DMA Channel 6
DMAC_CCR7	0x04e8	W	0x0	Channel Control for DMA Channel 7
DMAC_LC0_0	0x040C	W	0x0	Loop Counter 0 for DMA Channel 0
DMAC_LC0_1	0x042C	W	0x0	Loop Counter 0 for DMA Channel 1
DMAC_LC0_2	0x044C	W	0x0	Loop Counter 0 for DMA Channel 2
DMAC_LC0_3	0x046C	W	0x0	Loop Counter 0 for DMA Channel 3
DMAC_LC0_4	0x048C	W	0x0	Loop Counter 0 for DMA Channel 4
DMAC_LC0_5	0x04AC	W	0x0	Loop Counter 0 for DMA Channel 5
DMAC_LC0_6	0x04CC	W	0x0	Loop Counter 0 for DMA Channel 6
DMAC_LC0_7	0x04EC	W	0x0	Loop Counter 0 for DMA Channel 7
DMAC_LC1_0	0x0410	W	0x0	Loop Counter 1 for DMA Channel 0
DMAC_LC1_1	0x0430	W	0x0	Loop Counter 1 for DMA Channel 1
DMAC_LC1_2	0x0450	W	0x0	Loop Counter 1 for DMA Channel 2
DMAC_LC1_3	0x0470	W	0x0	Loop Counter 1 for DMA Channel 3
DMAC_LC1_4	0x0490	W	0x0	Loop Counter 1 for DMA

				Channel 4
DMAC_LC1_5	0x04B0	W	0x0	Loop Counter 1 for DMA Channel 5
DMAC_LC1_6	0x04D0	W	0x0	Loop Counter 1 for DMA Channel 6
DMAC_LC1_7	0x04F0	W	0x0	Loop Counter 1 for DMA Channel 7
-	-	-	-	reserved
DMAC_DBGST DMAC_ATUS	0x0D00	W	0x0	Debug Status Register.
DMAC_DBGCMD	0x0D04	W	0x0	Debug Command Register.
DMAC_DBGINST 0	0x0D08	W	0x0	Debug Instruction-0 Register.
DMAC_DBGINST 1	0x0D0C	W	0x0	Debug Instruction-1 Register.
DMAC_CR0	0x0E00	W		Configuration Register 0.
DMAC_CR1	0x0E04	W		Configuration Register 1.
DMAC_CR2	0x0E08	W		Configuration Register 2.
DMAC_CR3	0x0E0C	W		Configuration Register 3.
DMAC_CR4	0x0E10	W		Configuration Register 4.
DMAC_CRDn	0x0E14	W		Configuration Register Dn.
DMAC_WD	0X0E80	W		Watchdog Register

Notes:
*Size: **B** – Byte (8 bits) access, **HW** – Half WORD (16 bits) access, **W** –WORD (32 bits) access*

8.4.2 Detail Register Description

DMAC_DSR
Address:Operational Base+0x0
DMA Manager Status Register

Bit	Attr	Reset Value	Description
31:10	-	-	Reserved
9	R	0x0	Provides the security status of the DMA manager thread: 0 = DMA manager operates in the Secure state 1 = DMA manager operates in the Non-secure state.
8:4	R	0x0	When the DMA manager thread executes a DMAWFE instruction, it waits for the following event to occur: b00000 = event[0] b00001 = event[1] b00010 = event[2] … b11111 = event[31].
3:0	R	0x0	The operating state of the DMA manager: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event

	R		b0101-b1110 = reserved b1111 = Faulting.

DMAC_DPC
Address:Operational Base+0x4
DMA Program Counter Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Program counter for the DMA manager thread

DMAC_INTEN
Address:Operational Base+0x20
Interrupt Enable Register

Bit	Attr	Reset Value	Description
31:0	RW	0x0	Program the appropriate bit to control how the DMAC responds when it executes DMASEV: Bit [N] = 0 If the DMAC executes DMASEV for the event-interrupt resource N then the DMAC signals event N to all of the threads. Set bit [N] to 0 if your system design does not use irq[N] to signal an interrupt request. Bit [N] = 1 If the DMAC executes DMASEV for the event-interrupt resource N then the DMAC sets irq[N] HIGH. Set bit [N] to 1 if your system designer requires irq[N] to signal an interrupt request.

DMAC_EVENT_RIS
Address:Operational Base+0x24
Event-Interrupt Raw Status Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Returns the status of the event-interrupt resources: Bit [N] = 0 Event N is inactive or irq[N] is LOW. Bit [N] = 1 Event N is active or irq[N] is HIGH.

DMAC_INTMIS
Address:Operational Base+0x28
Interrupt Status Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the status of the interrupts that are active in the DMAC: Bit [N] = 0 Interrupt N is inactive and therefore irq[N] is LOW. Bit [N] = 1 Interrupt N is active and therefore irq[N] is HIGH

DMAC_INTCLR
Address:Operational Base+0x2c
Interrupt Clear Register

Bit	Attr	Reset Value	Description
31:0	W	0x0	Controls the clearing of the irq outputs: Bit [N] = 0 The status of irq[N] does not

			change. Bit [N] = 1 The DMAC sets irq[N] LOW if the INTEN Register programs the DMAC to signal an interrupt. Otherwise, the status of irq[N] does not change.

DMAC_FSRD
Address:Operational Base+0x30
Fault Status DMA Manager Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the fault status of the DMA manager. Read as: 0 = the DMA manager thread is not in the Faulting state 1 = the DMA manager thread is in the Faulting state.

DMAC_FSRC
Address:Operational Base+0x34
Fault Status DMA Channel Register

Bit	Attr	Reset Value	Description
31:0	R	0x0	Each bit provides the fault status of the corresponding channel. Read as: Bit [N] = 0 No fault is present on DMA channel N. Bit [N] = 1 DMA channel N is in the Faulting or Faulting completing state.

DMAC_FTRD
Address:Operational Base+0x38
Fault Type DMA Manager Register

Bit	Attr	Reset Value	Description
31	-	-	reserved
30	R	0x0	If the DMA manager aborts, this bit indicates if the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface.
29:17	-	-	reserved
16	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA manager performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response
15:6	-	-	reserved
5	R	0x0	Indicates if the DMA manager was attempting to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = DMA manager has appropriate security to execute DMAWFE or DMASEV 1 = a DMA manager thread in the Non-secure state attempted to execute either:

			• DMAWFE to wait for a secure event • DMASEV to create a secure event or secure interrupt
4	R	0x0	ndicates if the DMA manager was attempting to execute DMAGO with inappropriate security permissions: 0 = DMA manager has appropriate security to execute DMAGO 1 = a DMA manager thread in the Non-secure state attempted to execute DMAGO to create a DMA channel operating in the Secure state.
3:2	-	-	reserved
1	R	0x0	Indicates if the DMA manager was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand.
0	R	0x0	Indicates if the DMA manager was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction.

DMAC_FTR0~DMAC_FTR7
Address:Operational Base+0x40
Operational Base+0x44
Operational Base+0x48
Operational Base+0x4C
Operational Base+0x50
Operational Base+0x54
Operational Base+0x58
Operational Base+0x5C
Fault Type DMA Channel Register

Bit	Attr	Reset Value	Description
31	R	0x0	Indicates if the DMA channel has locked-up because of resource starvation: 0 = DMA channel has adequate resources 1 = DMA channel has locked-up because of insufficient resources. This fault is an imprecise abort
30	R	0x0	If the DMA channel aborts, this bit indicates if the erroneous instruction was read from the system memory or from the debug interface: 0 = instruction that generated an abort was read from system memory 1 = instruction that generated an abort was read from the debug interface. This fault is an imprecise abort but the bit is only valid when a precise abort occurs.
29:19	-	-	reserved
18	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs a data read: 0 = OKAY response

			1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort
17	R	0x0	Indicates the AXI response that the DMAC receives on the BRESP bus, after the DMA channel thread performs a data write: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is an imprecise abort.
16	R	0x0	Indicates the AXI response that the DMAC receives on the RRESP bus, after the DMA channel thread performs an instruction fetch: 0 = OKAY response 1 = EXOKAY, SLVERR, or DECERR response. This fault is a precise abort.
15:14	-	-	reserved
13	R	0x0	Indicates if the MFIFO did not contain the data to enable the DMAC to perform the DMAST: 0 = MFIFO contains all the data to enable the DMAST to complete 1 = previous DMALDs have not put enough data in the MFIFO to enable the DMAST to complete. This fault is a precise abort.
12	R	0x0	Indicates if the MFIFO prevented the DMA channel thread from executing DMALD or DMAST. Depending on the instruction: DMALD 0 = MFIFO contains sufficient space 1 = MFIFO is too small to hold the data that DMALD requires. DMAST 0 = MFIFO contains sufficient data 1 = MFIFO is too small to store the data to enable DMAST to complete. This fault is an imprecise abort
11:8	-	-	reserved
7	R	0x0	Indicates if a DMA channel thread, in the Non-secure state, attempts to program the CCRn Register to perform a secure read or secure write: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to perform a secure read or secure write. This fault is a precise abort
6	R	0x0	Indicates if a DMA channel thread, in the Non-secure state, attempts to execute DMAWFP, DMALDP, DMASTP, or DMAFLUSHP with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: • DMAWFP to wait for a secure peripheral • DMALDP or DMASTP to notify a secure peripheral

			• DMAFLUSHP to flush a secure peripheral. This fault is a precise abort.
5	R	0x0	Indicates if the DMA channel thread attempts to execute DMAWFE or DMASEV with inappropriate security permissions: 0 = a DMA channel thread in the Non-secure state is not violating the security permissions 1 = a DMA channel thread in the Non-secure state attempted to execute either: • DMAWFE to wait for a secure event • DMASEV to create a secure event or secure interrupt. This fault is a precise abort.
4:2	-	-	reserved
1	R	0x0	Indicates if the DMA channel thread was attempting to execute an instruction operand that was not valid for the configuration of the DMAC: 0 = valid operand 1 = invalid operand. This fault is a precise abort.
0	R	0x0	Indicates if the DMA channel thread was attempting to execute an undefined instruction: 0 = defined instruction 1 = undefined instruction. This fault is a precise abort

DMAC_CSR0~DMAC_CSR7
Address:Operational Base+0x100
Operational Base+0x108
Operational Base+0x110
Operational Base+0x118
Operational Base+0x120
Operational Base+0x128
Operational Base+0x130
Operational Base+0x138
Channel Status Registers

Bit	Attr	Reset Value	Description
31:22	-	-	reserved
21	R	0x0	The channel non-secure bit provides the security of the DMA channel: 0 = DMA channel operates in the Secure state 1 = DMA channel operates in the Non-secure state
20:16	-	-	reserved
15	R	0x0	When the DMA channel thread executes DMAWFP this bit indicates if the periph operand was set: 0 = DMAWFP executed with the periph operand not set 1 = DMAWFP executed with the periph operand set
14	R	0x0	When the DMA channel thread executes DMAWFP this bit indicates if the burst or single

			operand were set: 0 = DMAWFP executed with the single operand set 1 = DMAWFP executed with the burst operand set.
13:9	-	-	reserved
8:4	R	0x0	If the DMA channel is in the Waiting for event state or the Waiting for peripheral state then these bits indicate the event or peripheral number that the channel is waiting for: b00000 = DMA channel is waiting for event, or peripheral, 0 b00001 = DMA channel is waiting for event, or peripheral, 1 b00010 = DMA channel is waiting for event, or peripheral, 2 . . . b11111 = DMA channel is waiting for event, or peripheral, 31
3:0	R	0x0	The channel status encoding is: b0000 = Stopped b0001 = Executing b0010 = Cache miss b0011 = Updating PC b0100 = Waiting for event b0101 = At barrier b0110 = reserved b0111 = Waiting for peripheral b1000 = Killing b1001 = Completing b1010-b1101 = reserved b1110 = Faulting completing b1111 = Faulting

DMAC_CPC0~DMAC_CPC7
Address:Operational Base+0x104
Operational Base+0x10C
Operational Base+0x114
Operational Base+0x11c
Operational Base+0x124
Operational Base+0x12C
Operational Base+0x134
Operational Base+0x13C
Channel Program Counter Registers

Bit	Attr	Reset Value	Description
31:0	R	0x0	Program counter for the DMA channel n thread

DMAC_SAR0~DMAC_SAR7
Address:Operational Base+0x400
Operational Base+0x420
Operational Base+0x440
Operational Base+0x460

Operational Base+0x480
Operational Base+0x4A0
Operational Base+0x4C0
Operational Base+0x4E0

Source Address Registers

Bit	Attr	Reset Value	Description
31:0	R	0x0	Address of the source data for DMA channel n

DMAC_DAR0~DMAC_DAR7

Address:Operational Base+0x404
Operational Base+0x424
Operational Base+0x444
Operational Base+0x464
Operational Base+0x484
Operational Base+0x4A4
Operational Base+0x4C4
Operational Base+0x4E4

DestinationAddress Registers

Bit	Attr	Reset Value	Description
31:0	R	0x0	Address of the Destinationdata for DMA channel n

DMAC_CCR0~DMAC_CCR7

Address:Operational Base+0x408
Operational Base+0x428
Operational Base+0x448
Operational Base+0x468
Operational Base+0x488
Operational Base+0x4A8
Operational Base+0x4C8
Operational Base+0x4E8

Channel Control Registers

Bit	Attr	Reset Value	Description
31:28	-	-	reserved
27:25	R	0x0	Programs the state of AWCACHE[3,1:0]a when the DMAC writes the destination data. Bit [27] 0 = AWCACHE[3] is LOW 1 = AWCACHE[3] is HIGH. Bit [26] 0 = AWCACHE[1] is LOW 1 = AWCACHE[1] is HIGH. Bit [25] 0 = AWCACHE[0] is LOW 1 = AWCACHE[0] is HIGH
24:22	R	0x0	Programs the state of AWPROT[2:0]a when the DMAC writes the destination data. Bit [24] 0 = AWPROT[2] is LOW 1 = AWPROT[2] is HIGH. Bit [23] 0 = AWPROT[1] is LOW 1 = AWPROT[1] is HIGH. Bit [22] 0 = AWPROT[0] is LOW 1 = AWPROT[0] is HIGH
21:18	R	0x0	For each burst, these bits program the number of data transfers that the DMAC performs when it writes the destination data: b0000 = 1 data transfer

			b0001 = 2 data transfers b0010 = 3 data transfers . . . b1111 = 16 data transfers. The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size
17:15	R	0x0	For each beat within a burst, it programs the number of bytes that the DMAC writes to the destination: b000 = writes 1 byte per beat b001 = writes 2 bytes per beat b010 = writes 4 bytes per beat b011 = writes 8 bytes per beat b100 = writes 16 bytes per beat b101-b111 = reserved. The total number of bytes that the DMAC writes out of the MFIFO when it executes a DMAST instruction is the product of dst_burst_len and dst_burst_size.
14	R	0x0	Programs the burst type that the DMAC performs when it writes the destination data: 0 = Fixed-address burst. The DMAC signals AWBURST[0] LOW. 1 = Incrementing-address burst. The DMAC signals AWBURST[0] HIGH.
13:11	R	0x0	Set the bits to control the state of ARCACHE[2:0]a when the DMAC reads the source data. Bit [13] 0 = ARCACHE[2] is LOW 1 = ARCACHE[2] is HIGH. Bit [12] 0 = ARCACHE[1] is LOW 1 = ARCACHE[1] is HIGH. Bit [11] 0 = ARCACHE[0] is LOW 1 = ARCACHE[0] is HIGH.
10:8	R	0x0	Programs the state of ARPROT[2:0]a when the DMAC reads the source data. Bit [10] 0 = ARPROT[2] is LOW 1 = ARPROT[2] is HIGH. Bit [9] 0 = ARPROT[1] is LOW 1 = ARPROT[1] is HIGH. Bit [8] 0 = ARPROT[0] is LOW 1 = ARPROT[0] is HIGH.
7:4	R	0x0	For each burst, these bits program the number of data transfers that the DMAC performs when it reads the source data: b0000 = 1 data transfer b0001 = 2 data transfers b0010 = 3 data transfers . .

			. b1111 = 16 data transfers. The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size
3:1	R	0x0	For each beat within a burst, it programs the number of bytes that the DMAC reads from the source: b000 = reads 1 byte per beat b001 = reads 2 bytes per beat b010 = reads 4 bytes per beat b011 = reads 8 bytes per beat b100 = reads 16 bytes per beat b101-b111 = reserved. The total number of bytes that the DMAC reads into the MFIFO when it executes a DMALD instruction is the product of src_burst_len and src_burst_size
0	R	0x0	Programs the burst type that the DMAC performs when it reads the source data: 0 = Fixed-address burst. The DMAC signals ARBURST[0] LOW. 1 = Incrementing-address burst. The DMAC signals ARBURST[0] HIGH

DMAC_LC0_0~DMAC_LC0_7
Address:Operational Base+0x40c
Operational Base+0x42C
Operational Base+0x44C
Operational Base+0x46C
Operational Base+0x48C
Operational Base+0x4AC
Operational Base+0x4CC
Operational Base+0x4EC
Loop Counter 0 Registers

Bit	Attr	Reset Value	Description
31:8	-	-	reserved
7:0	R	0x0	Loop counter 0 iterations

DMAC_LC1_0~DMAC_LC1_7
Address:Operational Base+0x410
Operational Base+0x430
Operational Base+0x450
Operational Base+0x470
Operational Base+0x490
Operational Base+0x4B0
Operational Base+0x4D0
Operational Base+0x4F0

Loop Counter 1 Registers

Bit	Attr	Reset Value	Description
31:8	-	-	reserved

7:0	R	0x0	Loop counter 1 iterations

DMAC_DBGSTATUS
Address:Operational Base+0xD00
Debug Status Register

Bit	Attr	Reset Value	Description
31:2	-	-	reserved
1:0	R	0x0	The debug encoding is as follows: b00 = execute the instruction that the DBGINST [1:0] Registers contain b01 = reserved b10 = reserved b11 = reserved.

DMAC_DBGCMD
Address:Operational Base+0xD04
Debug Command Register

Bit	Attr	Reset Value	Description
31:2	-	-	reserved
1:0	W	0x0	The debug encoding is as follows: b00 = execute the instruction that the DBGINST [1:0] Registers contain b01 = reserved b10 = reserved b11 = reserved

DMAC_DBGINST0
Address:Operational Base+0xD08
Debug Instruction-0 Register

Bit	Attr	Reset Value	Description
31:24	W	0x0	Instruction byte 1
23:16	W	0x0	Instruction byte 0
17:11	-	-	reserved
10:8	W	0x0	DMA channel number: b000 = DMA channel 0 b001 = DMA channel 1 b010 = DMA channel 2 … b111 = DMA channel 7
7:1	-	-	reserved
0	W	0x0	The debug thread encoding is as follows: 0 = DMA manager thread 1 = DMA channel.

DMAC_DBGINST1
Address:Operational Base+0xD0C
Debug Instruction-1 Register

Bit	Attr	Reset Value	Description
31:24	W	0x0	Instruction byte 5
23:16	W	0x0	Instruction byte 4
15:8	W	0x0	Instruction byte 3
7:0	W	0x0	Instruction byte 2

DMAC_CR0
Address:Operational Base+0xE00
Configuration Register 0

Bit	Attr	Reset Value	Description
31:22	-	-	reserved
21:17	R	0x2	Number of interrupt outputs that the DMAC provides: b00000 = 1 interrupt output, irq[0] b00001 = 2 interrupt outputs, irq[1:0] b00010 = 3 interrupt outputs, irq[2:0] . . . b11111 = 32 interrupt outputs, irq[31:0].
16:12	R	0x7	Number of peripheral request interfaces that the DMAC provides: b00000 = 1 peripheral request interface b00001 = 2 peripheral request interfaces b00010 = 3 peripheral request interfaces . . . b11111 = 32 peripheral request interfaces.
11:7	-	-	reserved
6:4	R	0x5	Number of DMA channels that the DMAC supports: b000 = 1 DMA channel b001 = 2 DMA channels b010 = 3 DMA channels . . . b111 = 8 DMA channels.
3	-	-	reserved
2	R	0x0	Indicates the status of the boot_manager_ns signal when the DMAC exited from reset: 0 = boot_manager_ns was LOW 1 = boot_manager_ns was HIGH.
1	R	0x0	Indicates the status of the boot_from_pc signal when the DMAC exited from reset: 0 = boot_from_pc was LOW 1 = boot_from_pc was HIGH
0	R	0x1	Supports peripheral requests: 0 = the DMAC does not provide a peripheral request interface 1 = the DMAC provides the number of peripheral request interfaces that the num_periph_req field specifies.

DMAC_CR1
Address:Operational Base+0xE04
Configuration Register 1

Bit	Attr	Reset Value	Description
31:8	-	-	reserved

7:4	R	0x5	[7:4] num_i-cache_lines Number of i-cache lines: b0000 = 1 i-cache line b0001 = 2 i-cache lines b0010 = 3 i-cache lines … b1111 = 16 i-cache lines.
3	-	-	reserved
2:0	R	0x7	The length of an i-cache line: b000-b001 = reserved b010 = 4 bytes b011 = 8 bytes b100 = 16 bytes b101 = 32 bytes b110-b111 = reserved

DMAC_CR2
Address:Operational Base+0xE08
Configuration Register 2

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the value of boot_addr[31:0] when the DMAC exited from reset

DMAC_CR3
Address:Operational Base+0xE0C
Configuration Register 3

Bit	Attr	Reset Value	Description
31:0	R	0x0	Provides the security state of an event-interrupt resource: Bit [N] = 0 Assigns event<N> or irq[N] to the Secure state. Bit [N] = 1 Assigns event<N> or irq[N] to the Non-secure state.

DMAC_CR4
Address:Operational Base+0xE10
Configuration Register 4

Bit	Attr	Reset Value	Description
31:0	R	0x6	Provides the security state of the peripheral request interfaces: Bit [N] = 0 Assigns peripheral request interface N to the Secure state. Bit [N] = 1 Assigns peripheral request interface N to the Non-secure state

DMAC_CRDn
Address:Operational Base+0xE14
DMA Configuration Register

Bit	Attr	Reset Value	Description
31:30	-	-	reserved
29:20	R	0x20	The number of lines that the data buffer contains: b000000000 = 1 line b000000001 = 2 lines

			… b111111111 = 1024 lines
19:16	R	0x9	The depth of the read queue: b0000 = 1 line b0001 = 2 lines . . . b1111 = 16 lines.
15	-	-	reserved
14:12	R	0x4	Read issuing capability that programs the number of outstanding read transactions: b000 = 1 b001 = 2 … b111 = 8
11:8	R	0x7	The depth of the write queue: b0000 = 1 line b0001 = 2 lines … b1111 = 16 lines.
7	-	-	reserved
6:4	R	0x3	Write issuing capability that programs the number of outstanding write transactions: b000 = 1 b001 = 2 … b111 = 8
3	-	-	reserved
2:0		0x3	The data bus width of the AXI interface: b000 = reserved b001 = reserved b010 = 32-bit b011 = 64-bit b100 = 128-bit b101-b111 = reserved.

DMAC_WD
Address:Operational Base+0xE80
DMA Watchdog Register

Bit	Attr	Reset Value	Description
31:1	-	-	reserved
0	RW	0x0	Controls how the DMAC responds when it detects a lock-up condition: 0 = the DMAC aborts all of the contributing DMA channels and sets irq_abort HIGH 1 = the DMAC sets irq_abort HIGH.

8.5 Timing Diagram

Following picture shows the relationship between dma_req and dma_ack.

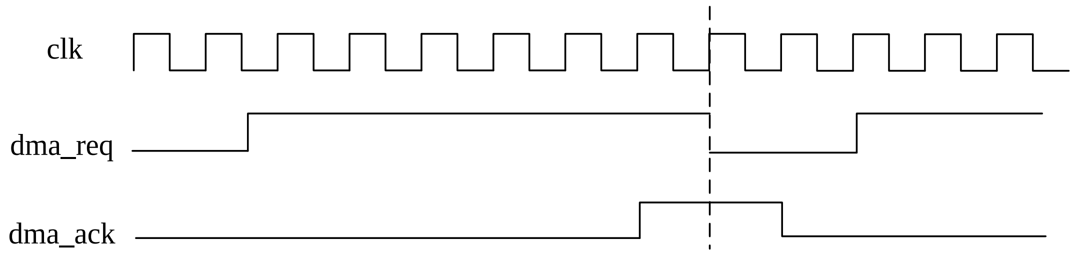

Fig. 8-3 DMAC0 request and acknowledge timing

8.6 Interface Description

DMAC has the following tie-off signals.It can be configured by GRF register.(Please refer to the chapter to find how to configure)

DMAC

interface	Reset value	Control source
boot_addr	0x0	GRF
boot_from_pc	0x0	GRF
boot_manager_ns	0x0	GRF
boot_irq_ns	0xf	GRF
boot_periph_ns	0xfffff	GRF

boot_addr

Configures the address location that contains the first instruction the DMAC executes, when it exits from reset.

boot_from_pc

Controls the location in which the DMAC executes its initial instruction, after it exits from reset:

0 = DMAC waits for an instruction from either APB interface
1 = DMA manager thread executes the instruction that is located at the address that

boot_manager_ns

When the DMAC exits from reset, this signal controls the security state of the DMA manager thread:

0 = assigns DMA manager to the Secure state
1 = assigns DMA manager to the Non-secure state.

boot_irq_ns

Controls the security state of an event-interrupt resource, when the DMAC exits from reset:

boot_irq_ns[x] is LOW
The DMAC assigns event<x> or irq[x] to the Secure state.
boot_irq_ns[x] is HIGH
The DMAC assigns event<x> or irq[x] to the Non-secure state.

boot_periph_ns

Controls the security state of a peripheral request interface, when the DMAC exits from reset:

boot_periph_ns[x] is LOW

The DMAC assigns peripheral request interface x to the Secure state.
boot_periph_ns[x] is HIGH
The DMAC assigns peripheral request interface x to the Non-secure state.

8.7 Application Notes

8.7.1 Using the APB slave interfaces

You must ensure that you use the appropriate APB interface, depending on the security state in which the boot_manager_ns initializes the DMAC to operate. For example, if the DMAC is in the secure state, you must issue the instruction using the secure APB interface, otherwise the DMAC ignores the instruction. You can use the secure APB interface, or the non-secure APB interface, to start or restart a DMA channel when the DMAC is in the Non-secure state.

The necessary steps to start a DMA channel thread using the debug instruction registers as following:

1. Create a program for the DMA channel.

2. Store the program in a region of system memory.

3. Poll the DBGSTATUS Register to ensure that debug is idle, that is, the dbgstatus bit is 0.

4. Write to the DBGINST0 Register and enter the:

• Instruction byte 0 encoding for DMAGO.
• Instruction byte 1 encoding for DMAGO.
• Debug thread bit to 0. This selects the DMA manager thread.

5. Write to the DBGINST1 Register with the DMAGO instruction byte [5:2] data, see Debug Instruction-1 Register o. You must set these four bytes to the address of the first instruction in the program, that was written to system memory in step 2.

6. Writing zero to the DBGCMD Register. The DMAC starts the DMA channel thread and sets the dbgstatus bit to 1.

8.7.2 Security usage

When the DMAC exits from reset, the status of the configuration signals that tie-off signals which descripted in chapter 10.6.

DMA manager thread is in the secure state

If the DNS bit is 0, the DMA manager thread operates in the secure state and it only performs secure instruction fetches. When a DMA manager thread in the secure state processes:

DMAGO

It uses the status of the ns bit, to set the security state of the DMAchannel thread by writing to the CNS bit for that channel.

DMAWFE

It halts execution of the thread until the event occurs. When the event occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding INS bit.

DMASEV

It sets the corresponding bit in the INT_EVENT_RIS Register, irrespective of the security state of the corresponding INS bit.

DMA manager thread is in the Non-secure state

If the DNS bit is 1, the DMA manager thread operates in the Non-secure state, and it only performs non-secure instruction fetches. When a DMA manager thread in the Non-secure state processes:

DMAGO

The DMAC uses the status of the ns bit, to control if it starts a DMA channel thread. If:

ns = 0

The DMAC does not start a DMA channel thread and instead it:
1. Executes a NOP.
2. Sets the FSRD Register, see Fault Status DMA Manager
3. Sets the dmago_err bit in the FTRD Register, see Fault Type DMA Manager Register.
4. Moves the DMA manager to the Faulting state.

ns = 1

The DMAC starts a DMA channel thread in the Non-secure state and programs the CNS bit to be non-secure.

DMAWFE

The DMAC uses the status of the corresponding INS bit, in the CR3 Register, tocontrol if it waits for the event. If:

INS = 0

The event is in the Secure state. The DMAC:
1. Executes a NOP.
2. Sets the FSRD Register, see Fault Status DMA Manager Register.
3. Sets the mgr_evnt_err bit in the FTRD Register, see Fault TypeDMA Manager Register.
4. Moves the DMA manager to the Faulting state.

INS = 1

The event is in the Non-secure state. The DMAC halts execution of the thread and waits for the event to occur.

DMASEV

The DMAC uses the status of the corresponding INS bit, in the CR3Register, to control if it creates the event-interrupt. If:

INS = 0

The event-interrupt resource is in the Secure state. The DMAC:
1. Executes a NOP.
2. Sets the FSRD Register, see Fault Status DMA Manager Register.
3. Sets the mgr_evnt_err bit in the FTRD Register, see Fault Type DMA Manager Register.
4. Moves the DMA manager to the Faulting state.

INS = 1

The event-interrupt resource is in the Non-secure state. The DMACcreates the event-interrupt.

DMA channel thread is in the secure state

When the CNS bit is 0, the DMA channel thread is programmed to operate in the Secure state and it only performs secure instruction fetches.

When a DMA channel thread in the secure state processes the following instructions:

DMAWFE

The DMAC halts execution of the thread until the event occurs. When the event occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding INS bit, in the CR3 Register.

DMASEV

The DMAC creates the event-interrupt, irrespective of the security state of the corresponding INS bit, in the CR3 Register.

DMAWFP

The DMAC halts execution of the thread until the peripheral signals a DMA request. When this occurs, the DMAC continues execution of the thread, irrespective of the security state of the corresponding PNS bit, in the CR4 Register.

DMALDP, DMASTP

The DMAC sends a message to the peripheral to communicate that data transfer

is complete, irrespective of the security state of the corresponding PNS bit, in the CR4 Register.

DMAFLUSHP

The DMAC clears the state of the peripheral and sends a message to the peripheral to resend its level status, irrespective of the security state of the corresponding PNS bit, in the CR4 Register.

When a DMA channel thread is in the Secure state, it enables the DMAC to perform secure and non-secure AXI accesses

DMA channel thread is in the Non-secure state

When the CNS bit is 1, the DMA channel thread is programmed to operate in the Non-secure state and it only performs non-secure instruction fetches.

When a DMA channel thread in the Non-secure state processes the following instructions:

DMAWFE

The DMAC uses the status of the corresponding INS bit, in the CR3 Register, to control if it waits for the event. If:

INS = 0

The event is in the Secure state. The DMAC:
1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Register.
3. Sets the ch_evnt_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel to the Faulting completing state.

INS = 1

The event is in the Non-secure state. The DMAC halts execution of the thread and waits for the event to occur.

DMASEV

The DMAC uses the status of the corresponding INS bit, in the CR3 Register, to control if it creates the event. If:

INS = 0

The event-interrupt resource is in the Secure state. The DMAC:
1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Register.
3. Sets the ch_evnt_err bit in the FTRn Register, see Fault Type DMA Channel Registers .
4. Moves the DMA channel to the Faulting completing state.

INS = 1

The event-interrupt resource is in the Non-secure state. The DMAC creates the event-interrupt.

DMAWFP

The DMAC uses the status of the corresponding PNS bit, in the CR4 Register, to control if it waits for the peripheral to signal a request. If:

PNS = 0

The peripheral is in the Secure state. The DMAC:
1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Register.
3. Sets the ch_periph_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel to the Faulting completing state.

PNS = 1

The peripheral is in the Non-secure state. The DMAC halts execution of the thread and waits for the peripheral to signal a request.

DMALDP, DMASTP

The DMAC uses the status of the corresponding PNS bit, in the CR4 Register, to control if it sends an acknowledgement to the peripheral. If:

PNS = 0

The peripheral is in the Secure state. The DMAC:
1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Register.
3. Sets the ch_periph_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel to the Faulting completing state.

PNS = 1

The peripheral is in the Non-secure state. The DMAC sends a message to the peripheral to communicate when the data transfer is complete.

DMAFLUSHP

The DMAC uses the status of the corresponding PNS bit, in the CR4 Register, to control if it sends a flush request to the peripheral. If:

PNS = 0

The peripheral is in the Secure state. The DMAC:
1. Executes a NOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Registe.
3. Sets the ch_periph_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel to the Faulting completing state.

PNS = 1

The peripheral is in the Non-secure state. The DMAC clears the state of the peripheral and sends a message to the peripheral to resend its level status.

When a DMA channel thread is in the Non-secure state, and a DMAMOV CCR instruction attempts to program the channel to perform a secure AXI transaction, the DMAC:

1. Executes a DMANOP.
2. Sets the appropriate bit in the FSRC Register that corresponds to the DMA channel number. See Fault Status DMA Channel Registe.
3. Sets the ch_rdwr_err bit in the FTRn Register, see Fault Type DMA Channel Registers.
4. Moves the DMA channel thread to the Faulting completing state.

8.7.3 Programming restrictions

Fixed unaligned bursts

The DMAC does not support fixed unaligned bursts. If you program the following conditions, the DMAC treats this as a programming error:

Unaligned read

• src_inc field is 0 in the CCRn Register
• the SARn Register contains an address that is not aligned to the size of data that the src_burst_size field contain

Unaligned write

• dst_inc field is 0 in the CCRn Register
• the DARn Register contains an address that is not aligned to the size of data that the dst_burst_size field contains

Endian swap size restrictions

If you program the endian_swap_size field in the CCRn Register, to enable a DMA channel to perform an endian swap then you must set the corresponding SARn Register and the corresponding DARn Register to contain an address that is aligned to the value that the endian_swap_size field contains.

Updating DMA channel control registers during a DMA cyclerestrictions

Prior to the DMAC executing a sequence of DMALD and DMAST instructions, the values you program in to the CCRn Register, SARn Register, and DARn Register control the data byte lane manipulation that the DMAC performs when it transfers the data from the source address to the destination address. You'd better not update these registers during a DMA cycle.

Resource sharing between DMA channels

DMA channel programs share the MFIFO data storage resource. You must not start a set ofconcurrently running DMA channel programs with a resource requirement that exceeds theconfigured size of the MFIFO. If you exceed this limit then the DMAC might lock up and generate a Watchdog abort.

8.7.4 Unaligned transfers may be corrupted

For a configuration with more than one channel, if any of channels 1 to 7 is performing transfers between certain types of misaligned source and destination addresses, then the output data may be corrupted by the action of channel 0.

Data corruption might occur if all of the following are true:

1. Two beats of AXI read data are received for one of channels 1 to 7.
2. Source and destination address alignments mean that each read data beat is split across two lines in the data buffer (see Splitting data, below).
3. There is one idle cycle between the two read data beats .
4. Channel 0 performs an operation that updates channel control information during this idle cycle (see Updates to channel control information, below)

Splitting data

Depending upon the programmed values for the DMA transfer, one beat of read data from the AXI interface mneed to be split across two lines in the internal data buffer. This occurs when the read data beat contains datbytes which will be written to addresses that wrap around at the AXI interface data width, so that these bytes could not be transferred by a single AXI write data beat of the full interface width.

Most applications of DMA-330 do not split data in this way, so are NOT vulnerable to data corruption from thisdefect.

The following cases are NOT vulnerable to data corruption because they do not split data:

• Byte lane offset between source and destination addresses is 0 When source and destination addresses have the same byte lane alignment, the offset is 0 and a wrap operation that splits data cannot occur.
• Byte lane offset between source and destination addresses is a multiple of source size

Source size in CCRn	Allowed offset between SARn and DARn
SS8	any offset allowed.
SS16	0,2,4,6,8,10,12,14
SS32	0,4,8,12
SS64	0,8

8.7.5 Interrupt shares between channel.

As the DMAC0 does not record which channel (or list of channels) have asserted an interrupt. So it will depend on your program and whether any of the visible information for that program can be used to determine progress, and help identify the interrupt source.

There are 4 likely information sources that can be used to determine the progress made by a program:

- Program counter (PC)
- Source address
- Destination address
- Loop counters (LC)

For example, a program might emit an interrupt each time that it iterates around a loop. In this case, the interrupt service routine (ISR) would need to store the loop value of each channel when it is called, and then compare against the new value when it is next called. A change in value would indicate that the program has progressed.

The ISR must be carefully written to ensure that no interrupts are lost. The sequence of operations is as follows:

1. Disable interrupts
2. Immediately clear the interrupt in DMA-330
3. Check the relevant registers for both channels to determine which must be serviced
4. Take appropriate action for the channels
5. Re-enable interrupts and exit ISR

8.7.6 Instruction sets

Table 8-2 DMAC Instruction sets

Mnemonic	Instruction	Thread usage: • M = DMA manager • C = DMA channel
DMAADDH	Add Halfword	C
DMAEND	End	M/C
DMAFLUSHP	Flush and notify Peripheral	C
DMAGO	Go	M
DMAKILL	Kill	C
DMALD	Load	C
DMALDP	Load Peripheral	C
DMALP	Loop	C
DMALPEND	Loop End	C
DMALPFE	Loop Forever	C
DMAMOV	Move	C
DMANOP	No operation	M/C
DMARMB	Read Memory Barrier	C
DMASEV	Send Event	M/C
DMAST	Store	C
DMASTP	Store and notify Peripheral	C
DMASTZ	Store Zero	C
DMAWFE	Wai t For Event M	M/C
DMAWFP	Wait For Peripheral	C
DMAWMB	Write Memory Barrier	C
DMAADNH	Add Negative Halfword	C

8.7.7 Assembler directives

In this document, only DMMADNH instruction is took as an example to show the way the instruction assembled. *For the other instructions , please refer to pl330_trm.pdf.*

DMAADNH

Add Negative Halfword adds an immediate negative 16-bit value to the SARn Register or DARn Register, for the DMA channel thread. This enables the DMAC to support 2D DMA operations, or reading or writing an area of memory in a different order to naturally incrementing addresses. See Source Address Registers and Destination Address Registers.

The immediate unsigned 16-bit value is one-extended to 32 bits, to create a value that is the two's complement representation of a negative number between -65536 and -1, before the DMAC adds it to the address using 32-bit addition. The DMAC discards the carry bit so that addresses wrap from 0xFFFFFFFF to 0x00000000. The net effect is to subtract between 65536 and 1 from the current value in the Source or Destination Address Register.

Following table shows the instruction encoding.

Imm[15:8]	Imm[7:0]	0	1	0	1	1	1	ra	0

Assembler syntax

DMAADNH <address_register>, <16-bit immediate>

where:
<address_register>
Selects the address register to use. It must be either:
SAR
SARn Register and sets ra to 0.
DAR
DARn Register and sets ra to 1.

<16-bit immediate>
The immediate value to be added to the <address_register>.

You should specify the 16-bit immediate as the number that is to be represented in the instruction encoding. For example, DMAADNH DAR, 0xFFF0 causes the value 0xFFFFFFF0 to be added to the current value of the Destination Address Register, effectively subtracting 16 from the DAR.

You can only use this instruction in a DMA channel thread.

8.7.8 MFIFO usage

For MFIFO usage , please refer to pl330_trm.pdf