DMA controller  
Rev 1.0



# *Document changing history*

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Details | Author |
| 20 sept 2017 | Rev 1.0 | Newly created | Duong Nguyen |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[*Document changing history* 1](#_Toc494964724)

[Table of Contents 2](#_Toc494964725)

[1. Overview 3](#_Toc494964726)

[2. Design Features. 3](#_Toc494964727)

[3. Functional Block Diagram. 4](#_Toc494964728)

[3.1 IO description. 4](#_Toc494964729)

[3.2 APB Module. 6](#_Toc494964730)

[3.3 Channels Arbiter Module. 6](#_Toc494964731)

[3.4 DMA Handshaking. 8](#_Toc494964732)

[3.5 DMA Engine Module. 9](#_Toc494964733)

[3.6 DMA Descriptor Look Up Module. 12](#_Toc494964734)

[4. Configuration and status registers. 14](#_Toc494964735)

[4.1 Main Configuration & Status Register (0x000). 15](#_Toc494964736)

[4.2 Main DMA status register (0x004). 15](#_Toc494964737)

[4.3 Channel 0 Control register (c0cr, 0x010). 15](#_Toc494964738)

[4.4 Channel 0 status register (c0sr, 0x014). 16](#_Toc494964739)

[4.5 Channel 0 transfer size register (c0tsr, 0x018). 17](#_Toc494964740)

[4.6 Channel 0 current descriptor pointer register (c0cdpr, 0x01C). 17](#_Toc494964741)

[4.7 Channel 0 source address register (c0sar, 0x020). 17](#_Toc494964742)

[4.8 Channel 0 destination address register (c0dar, 0x024). 18](#_Toc494964743)

[5. Programming Sequence. 18](#_Toc494964744)

# 1. Overview

The **Direct memory access** (**DMA**) is a feature of computer systems that allows certain hardware subsystems to access main system [memory](https://en.wikipedia.org/wiki/Computer_storage) ([Random-access memory](https://en.wikipedia.org/wiki/Random-access_memory)), independent of the [central processing unit](https://en.wikipedia.org/wiki/Central_processing_unit) (CPU).

Without DMA, when the CPU is using [programmed input/output](https://en.wikipedia.org/wiki/Programmed_input/output), it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU first initiates the transfer, then it does other operations while the transfer is in progress, and it finally receives an [interrupt](https://en.wikipedia.org/wiki/Interrupt) from the DMA controller when the operation is done. This feature is useful at any time that the CPU cannot keep up with the rate of data transfer, or when the CPU needs to perform useful work while waiting for a relatively slow I/O data transfer. Many hardware systems use DMA, including [disk drive](https://en.wikipedia.org/wiki/Disk_drive) controllers, [graphics cards](https://en.wikipedia.org/wiki/Graphics_card), [network cards](https://en.wikipedia.org/wiki/Network_card) and [sound cards](https://en.wikipedia.org/wiki/Sound_card). DMA is also used for intra-chip data transfer in [multi-core processors](https://en.wikipedia.org/wiki/Multi-core_processor). Computers that have DMA channels can transfer data to and from devices with much less CPU overhead than computers without DMA channels. Similarly, a processing element inside a multi-core processor can transfer data to and from its local memory without occupying its processor time, allowing computation and data transfer to proceed in parallel.

DMA can also be used for "memory to memory" copying or moving of data within memory. DMA can offload expensive memory operations, such as large copies or [scatter-gather](https://en.wikipedia.org/wiki/Vectored_I/O) operations, from the CPU to a dedicated DMA engine.

# 2. Design Features.

- Up to 4 DMA Channels.

- 4 priority levels.

- Fixed-address and incrementing address support.

- APB bus for configuration.

- AHB-lite bus for data transfer.

- Circular Buffer Support.

- Scatter-gather DMA support (Linked list descriptor).

# 3. Functional Block Diagram.



Figure 1-DMA functional block diagram.

## **3.1 IO description.**

|  |  |  |
| --- | --- | --- |
| **Name** | **I/O** | **Function** |
| rst | I | Active high Reset. |
| clk | I | AHB lite clock. |
| APB Interface | | |
| psel\_i | I | DMA select. |
| pwrite\_i | I | 1’b1 : Write transfer.  1’b0 : Read transfer. |
| penable\_i | I | APB enable. |
| paddr\_i[31:0] | I | APB Address. |
| pwdata [31:0] | I | Write data. |
| pstrb\_i[3:0] | I | Byte strobe. |
| hprot\_m0\_i[3:0] | I | AHB-Lite protection control for master 0. Do not use these signals. |
| pprot\_i[3:0] | I | Don’t use. |
| prdata\_o[31:0] | O | Read data/. |
| pslverr\_o | O | Transfer is OK or not. |
| pready\_o |  | Data ready. |
| Linked list descriptor interface (AHB master). | | |
| hrdata\_i[31:0] | I | AHB-Lite read data from slave. |
| hready\_i | I | AHB-Lite ready signal from slave. |
| hresp\_i | I | AHB-Lite transfer response from slave. |
| hsel\_o | O | AHB-Lite slave select. |
| haddr\_o[31:0] | O | AHB-Lite address bus . |
| hsize\_o[2:0] | O | AHB-Lite transfer size . |
| htrans\_o[1:0] | O | AHB-Lite transfer type. |
| hwrite\_o | O | AHB-Lite write indication. |
| hwdata\_o[31:0] | O | AHB-Lite write data. |
| hmastlock\_o | O | AHB-Lite master locked sequence indication. |
| hburst\_o[2:0] | O | AHB-Lite burst type indication. |
| Read/write data interface (AHB master). | | |
| hrdata\_i[31:0] | I | AHB-Lite read data from slave. |
| hready\_i | I | AHB-Lite ready signal from slave. |
| hresp\_i | I | AHB-Lite transfer response from slave. |
| hsel\_o | O | AHB-Lite slave select. |
| haddr\_o[31:0] | O | AHB-Lite address bus . |
| hsize\_o[2:0] | O | AHB-Lite transfer size . |
| htrans\_o[1:0] | O | AHB-Lite transfer type. |
| hwrite\_o | O | AHB-Lite write indication. |
| hwdata\_o[31:0] | O | AHB-Lite write data. |
| hmastlock\_o | O | AHB-Lite master locked sequence indication. |
| hburst\_o[2:0] | O | AHB-Lite burst type indication. |
| Handshaking Interface | | |
| dreq\_i[7:0] | I | DMA request signals from peripherals. |
| dack\_o[7:0] | O | DMA acknowledge signals. |

## **3.2 APB Module.**

APB bus slave interface is used for register access. It supports 32 bit data word.

## **3.3 Channels Arbiter Module.**

The prioritizing arbiter will select the next channel to process, based first on priority, and secondarily, if all priorities are equal, in a round robin way. Each channel has a 4-bit priority value associated with it. A value of 0 is lowest priority, and 7 is highest priority.

Channels with the same priority are processed in a round robin way.

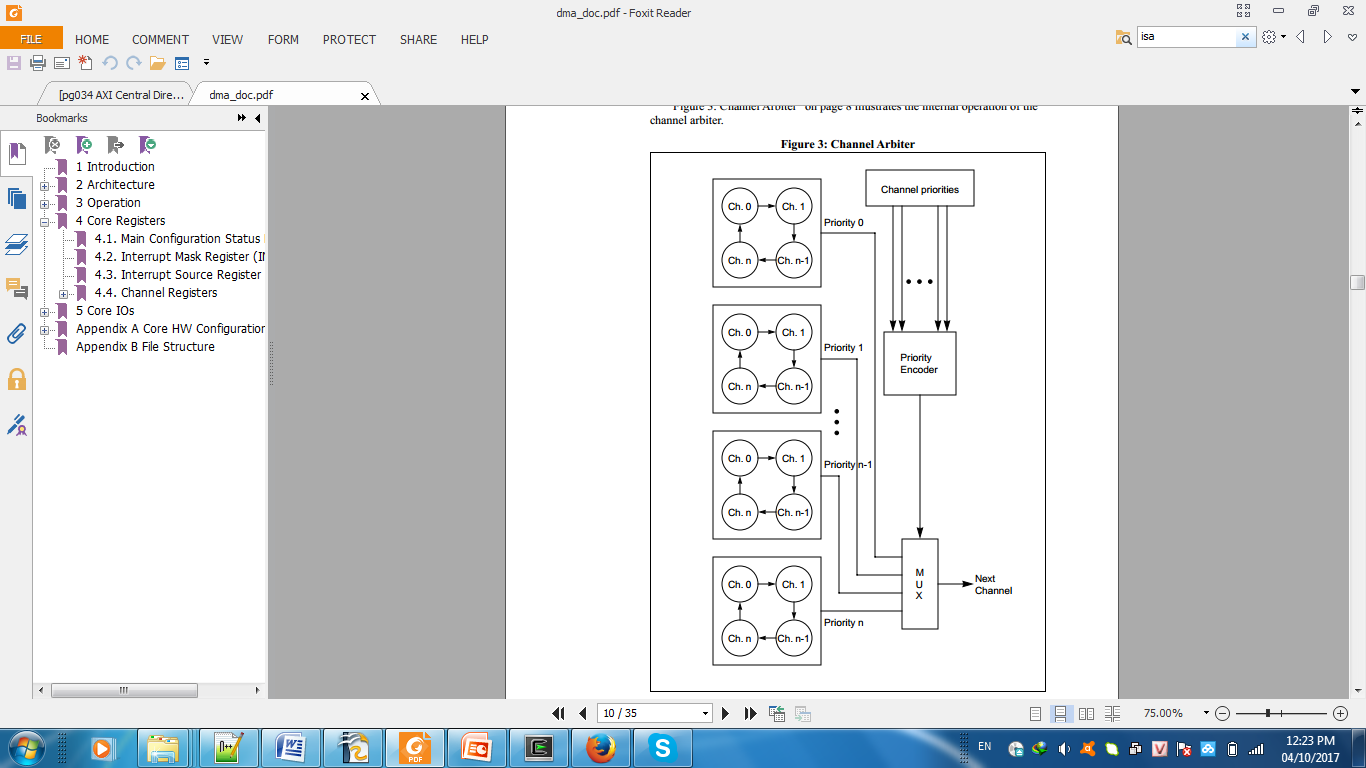


Figure 2-Channel arbiter illustration.

Figure 2 above is an example illustration of arbiter.



Figure 3-DMA channels arbiter implementation.

## **3.4 DMA Handshaking.**

When DMA channel operates in handshake mode, DMA wait for the perippheral’s DMA request signal asserted before starting transfer. After finishing all transfer, DMA will assert ACK signal and re-arbitrate to grant next channel.



Figure 4-Handshaking timing between DMA and peripheral.



Figure 4-DMA Handshaking implementation.

## **3.5 DMA Engine Module.**

Each DMA channel can be programmed to perform various transfer operations.



Figure 4-Simplified DMA operation flow.

**Normal Mode Operation :**

In this mode the DMA engine performs a block copy from one memory location to another. When the transfers are completed, the DMA engine will assert an interrupt (if enabled) and then go to serve next channel.

The software can also force the channel to stop by writing a one to the STOP bit in the channel register. In this case the DMA channel will immediately stop and indicate an error condition by setting the STOP\_ERR bit in the channel register and asserting an error interrupt (if enabled).

**Handshake Mode Operation :**

DMA channel and periperal handshake via REQ and ACK signals. DMA channel wait for the perippheral’s DMA request signal asserted before starting transfer. After finishing all transfer, DMA will assert ACK signal, generate interrupt and re-arbitrate to grant next channel.

**Linked List Mode Operation :**

Descriptor linked list is a mechanism that allows for automated data transfer scheduling through a pre-programmed instruction list of transfer descriptors (Linked list Descriptor Definition). This instruction list is programmed by software application into a  
memory-resident data structure that must be accessible by the Descriptor AHB interface. This list of instructions is organized into what is referred to as a transfer descriptor chain. Each descriptor has an address pointer to the next sequential descriptor to be processed. The last descriptor in the chain generally points back to the first descriptor in the chain but it is not required.



Figure 5-Linked list descriptor.

When DMA channel operate in this mode, DMA engine requests Descriptor Look Up Engine to get the current descriptor from memory via AHB bus and send it to DMA engine. Then DMA is starting transfers based on descriptor definitions.

After all transfer are finished, DMA engine request Descriptor Look Up engine to update current descriptor in memory so that the software application can track the completion status of the transfer associated with current DMA channel.

If the EOL bit in the current decriptor is set, the DMA engine will stop after finishing current transfers, set the DONE bit in the channel register, and assert an interrupt, if enabled.



Figure 6-Main DMA engine implementation.

## **3.6 DMA Descriptor Look Up Module.**

The Descriptor Look up Engine fetches and updates DMA control transfer descriptors from system memory through the dedicated AHB Master interface. It allows descriptor prefetch and processing in parallel with ongoing DMA data transfer  
operations.



Figure 6-Descriptor look up implementation.

**Transfer Descriptor Definition.**

|  |  |  |
| --- | --- | --- |
| **Offset** | **Name** | **Adress** |
| 0x00 | Control\_status | Transfer control, status. |
| 0x04 | nxt\_des\_ptr | Next Descriptor Pointer. |
| 0x08 | sa | Source address. |
| 0x0C | da | Destination address. |

**Next Descriptor Pointer.**

|  |  |  |
| --- | --- | --- |
| **Bit** | **Name** | **Adress** |
| [31:4] | nxt\_ptr | Next Descriptor Pointer. This field is an address pointer to the first word of the next transfer descriptor. |
| [3:0] | reserved | These bits are reserved and fixed to zeros. |

**Source Address.**

|  |  |  |
| --- | --- | --- |
| **Bit** | **Name** | **Adress** |
| [31:0] | des\_sa | Source Address. This value specifies the starting address for data read operations for the associated DMA transfer. |

**Destination Address.**

|  |  |  |
| --- | --- | --- |
| **Bit** | **Name** | **Adress** |
| [31:0] | des\_da | Destination Address. This value specifies the starting address for data write operations for the associated DMA transfer. |

**Control and Status Word.**

|  |  |  |
| --- | --- | --- |
| **Bit** | **Name** | **Adress** |
| [31:18] | reserved |  |
| [17] | des\_err | 1 : Errors happened during transfer. |
| [16] | des\_done | 0 : Descriptor not completed. 1 : Descriptor completed. |
| [10:15] | reserved |  |
| [12] | eol | If set, indicates that this is the last descriptor in the list. |
| [11:0] | size | Total transfer size in word (32 bit). |

Prior to starting DMA linked list mode operations, the software application must set up a transfer descriptor chain. After the main DMA engine begins operations, it fetches, processes, and then updates the descriptors.

It stop fetching descriptor when EOL bit in control and status field is 1.

# 4. Configuration and status registers.

|  |  |  |
| --- | --- | --- |
| **No** | **Name** | **Adress** |
| 1 | DMA Configuration Register (dcr) | 0x00 |
| 2 | DMA status register (dsr) | 0x04 |
| Channel 0 Register | | |
| 3 | Channel 0 Control register (c0cr). | 0x10 |
| 4 | Channel 0 status register (c0sr). | 0x14 |
| 5 | Channel 0 transfer size register (c0tsr). | 0x18 |
| 6 | Channel 0 current descriptor pointer register (c0cdpr). | 0x1C |
| 7 | Channel 0 source address register (c0sar). | 0x20 |
| 8 | Channel 0 destination address register (c0dar). | 0x24 |
|  | Channel 1 Registers | Offset : 0x30 |
|  | Channel 2 Registers | Offset : 0x50 |
|  | Channel 3 Registers | Offset : 0x70 |

## **4.1 Main Configuration Register (0x000).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default Value** | **Description** |
| [31:5] | Reserved |  |  | Reserved |
| [4] | dma\_en | R/W | 0 | 1’b1 : Enable DMA engine.  Writing 1’b0 to this bit will pause DMA engine. |
| [3:0] | ch\_int\_en[3:0] | R/W | 0 | Writing 1’b1 to these bit will enable interrupt generation for each channel.  [0] : Interrupt enable for channel 0.  ………..  [3] : Interrupt enable for channel 3. |

## **4.2 Main DMA status register (0x004).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default Value** | **Description** |
| [31:5] | Reserved |  |  | Reserved |
| [4] | dma\_busy | R | 0 | 1’b1 : DMA engine is busy. |
| [3:0] | ch\_int\_status[3:0] | R | 0 | [0] : Interrupt source of channel 0.  ………..  [3] : Interrupt source of channel 3. |

## **4.3 Channel 0 Control register (c0cr, 0x010).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default Value** | **Description** |
| [31:11] | Reserved |  |  | Reserved |
| [10] | sw\_reset | W |  | Writing 1 to reset DMA channel. |
| [9:8] | ch\_pri | R/W | 0 | Channel priority. 7 is highest priority. |
| [7:6] | mode | R/W | 0 | 00 : normal mode.  01 : Hardware handshake mode.  10 : Linked list mode.  11 : Reserved. |
| [5] | inc\_des | R/W | 0 | 0: Do not increment destination address. 1: Increment destination address. |
| [4] | inc\_src | R/W | 0 | 0: Do not increment source address. 1: Increment source address. |
| [3] | done\_int\_en | R/W | 0 | Complete Interrupt Enable. When set to 1, it allows the channle to generate an interrupt when the transfer is done. |
| [2] | stop\_err\_int\_en | R/W | 0 | Stop Error interrupt enable. When set to 1, it allows the channel to generate an Stop Error interrupt. |
| [1] | wtt\_err\_int\_en | R/W | 0 | Words to transfer error interrupt enable. When set to 1, it allows the channel to generate an interrupt if number of transferred words are zero. |
| [0] | ch\_en | R/W | 0 | Channel enabled.  1’b1 : Enable.  1’b0 : Disable. |

## **4.4 Channel 0 status register (c0sr, 0x014).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default Value** | **Description** |
| [31:5] | Reserved |  |  | Reserved |
| [3] | done\_irq | R | 0 | Interrupt on Complete. When set to 1, this bit indicates an interrupt event has been generated on completion of a DMA channel. If the corresponding enable bit is set (c0cr.done\_int\_en = 1), an interrupt out is generated. 0 = No Done Interrupt 1 = Done Interrupt active Writing a 1 to this bit to clear it. |
| [2] | stop\_err | R |  | Channel DMA is stop suddenly by sw reset or disable when the transfer is on going.  Writing a 1 to this bit to clear. |
| [1] | wtt\_err | R | 0 | Words to transfer error. (In case words to transfer is zero).  Writing a 1 to this bit to clear. |
| [0] | ch\_idle | R | 0 | 1’b1 : Idle  1’b0 : busy.  When set the bit indicates the programmed transfer has completed and the DMA channel is waiting for a new transfer to be programmed. |

## **4.5 Channel 0 transfer size register (c0tsr, 0x018).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default Value** | **Description** |
| [31:8] | Reserved |  |  | Reserved |
| [7:0] | size[7:0] | R/W | 0 | Number of word (32bit) to transfer. Maximum size is 255 words. |

## **4.6 Channel 0 current descriptor pointer register (c0cdpr, 0x01C).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default Value** | **Description** |
| [31:4] | cur\_ptr[25:0] | R | 0 | Current Descriptor Pointer. Indicates the pointer of the current descriptor being worked on. When the DMA Engine is running (c0sr.idle = 0), this register is updated by the Lookup Descriptor Engine to reflect the starting address of the current descriptor being executed.  The register should only be written by the software application when chanel is in idle state. |
| [3:0] | Reserved |  |  |  |

## **4.7 Channel 0 source address register (c0sar, 0x020).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default Value** | **Description** |
| [31:0] | sa[31:0] | R/W | 0 | Source Address Register. This register is used as the starting read address for DMA data transfers. The address value written must be aligned. The software application should only write to this register when the channel DMA is idle. |

## **4.8 Channel 0 destination address register (c0dar, 0x024).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bits** | **Name** | **Access** | **Default Value** | **Description** |
| [31:0] | da[31:0] | R/W | 0 | Destination Address Register. This register is used as the starting write address for DMA data transfers. The address value written must be aligned.  The software application should only write to this register when the channel DMA is idle. |

# 5. Programming Sequence.