
# ProNoC

## Multi-channel Direct memory access
## ( Multi-channel DMA)

**Summary** This module is a generic, multi-channel, Wishbone bus (WB)-based Multi-channel Direct memory access (DMA). It has three WB interfaces: two WB master interfaces which is used for reading and writing data on the memory and another WB slave interface which is used for programing the DMA. The DMA can be configured with different number of independent channels. A channel is active when it has data to be read/wriiten in the memory. Each active channel can take the control of memory interface for certain period of time. When there are multiple active channels a round robin arbitration policy is used to define which active channel takes the control of memory interface. The winner channel is allowed to read/write data on the memory via WB master interface while both of the flowing conditions are met a) The target memory is available (i.e the write FIFO is not full/ the read FIFO is not empty) b) The transfered data size does not reach the maximum burst size.

**Notice 1** The Multi-channel DMA supports WB burst mode. Hence, to avoid performance degradation make sure to enable the burst mode when you are calling memory module in processing tile generator.

**Notice 2** The WB byte enable signal does not supported yet. Hence, the read/write memory addresses and transferring data sizes must be given in word size.

**Multi-channel DMA Block diagram:** Figure1 depicts the functional block diagram of Multi-channel DMA module.

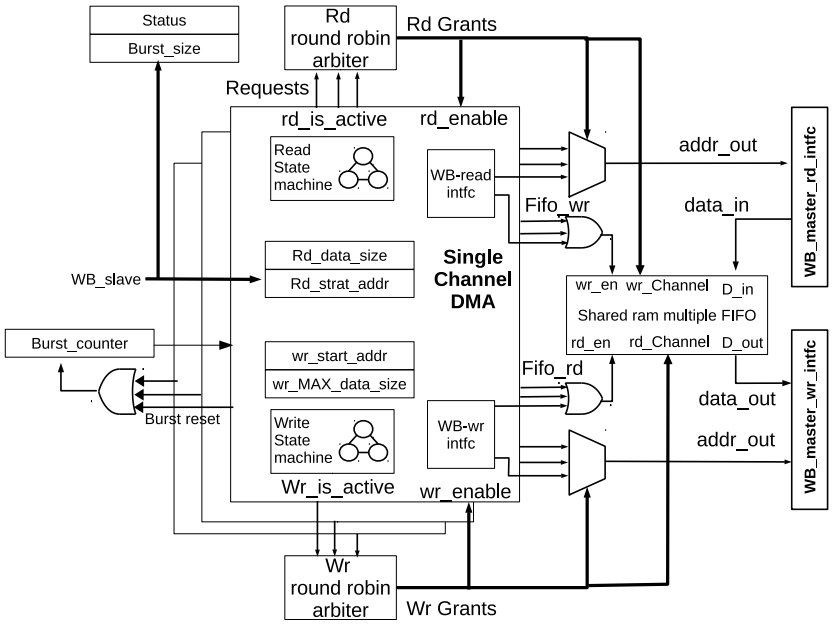


Figure 1: Multi-channel DMA functional block diagram

**Parameters**
**Description**

Table 1: Multi-channel DMA GUI Parameters.

| Name | Description | Permitted values | Default |
|---|---|---|---|
| CHANNEL | Number of DMA channels. In case there are multiple active DMA channels, Each time one single active DMA channel get access to the wishbone bus using round robin arbiter. The Wishbone bus is granted for the winter channel until its FIFO is not full and the number of sent data is smaller than the burst size. | $n \in \mathbb{N}$, $1 \leqslant n \leqslant 32$ | 1 |
| MAX-TRANSACTION-WIDTH | The width of maximum transaction size in words. The maximum data that can be sent via one DMA channel will be 2 power of MAX-DMA-TRANSACTION-WIDTH in words. | $n \in \mathbb{N}$, $2 \leqslant n \leqslant 32$ | 10 |
| MAX-BURST-SIZE | Maximum burst size in words. The wishbone bus will be released each time one burst is completed or when the internal FIFO becomes full. Then, the bus will be released for one clock cycle. In case, there are other active channels, another active channel will get access to the bus using round robin arbiter. This process will be continued until all desired data is transferred. | $n \in 2^m$, $1 \leqslant m \leqslant 11$ | 256 |
| FIFO-B | Channel FIFO size in words. All channels will share same FPGA block RAM. Hence, the total needed Block RAM words is the multiplication of channel num in channel FIFO size. | $n \in 2^m$, $1 \leqslant m \leqslant 11$ | 4 |
| Dw | The Wishbone bus data width in bits. | $n \in 4 \times m$, $8 \leqslant m \leqslant 64$ | 32 |

**Slave wishbone bus registers**

The Multi-channel DMA can be controlled using wishbone bus slave interface. Table 2 shows the Multi-channel internal registers. All Multi-channel registers are memory mapped.

Table 2: Multi-channel DMA memory map registers. Note that $n$ in address range is the channel number where $0 \leq n <$ CHANNEL.

| addr[7:0] | Register Name | Description |
|---|---|---|
| 0 | DMA-STATUS-WB-ADDR | DMA status register |
| 1 | BURST-SIZE-WB-ADDR | The DMA burst size in words |
| $(8n+2)$ | DATA-SIZE-WB-ADDR | The size of data to be read/written in bytes |
| $(8n+3)$ | RD-STRT-WB-ADDR | The start address of data to be sent in bytes |
| $(8n+4)$ | WR-STRT-WB-ADDR | The destination start address in byte |

**DMA-STATUS-WB-ADDR**

DMA status register includes three $n$-bit and two $n_w$=$\log_2(n)$-bit status variables where $n$ indicates the number of DMA channels. The $n$-bit variables are in one-hot format where each bit represents its corresponding channel. i.e bit 0 represent channel number 0 and so on.

| Bit | $3n$-1 ... $2n$ | $2n$-1 ... $n$ | $n$-1 ... 0 |
|---|---|---|---|
| | channel-rd-is-busy | channel-wr-is-busy | channel-is-active |
| Read/Write | R | R | R |
| Initial Value | 0 | 0 | 0 |

| Bit | $3n+2n_w$-1 ... $3n+n_w$ | $3n+n_w$-1 ... $3n$ |
|---|---|---|
| | rd-enable-binary | wr-enable-binary |
| Read/Write | R | R |
| Initial Value | 0 | 0 |

**channel-is-active:** The asserted bit in this register indicates which DMA channel is active. (Has a data to be read or written).

**channel-wr-is-busy:** The asserted bit in this register indicates which DMA channel has data to be written.

**receive-vc-is-busy:** The asserted bit in this register indicates which DMA channel has data to be read from memory.

**wr-enable-binary:** In any given clock cycle only one of the active DMAs can have access to the WB write interface. The value of this register shows the binary number of this DMA channel.

**rd-enable-binary:** In any given clock cycle only one of the active DMAs can have access to the WB read interface. The value of this register shows the binary number of this DMA channel.

**Note** Several DMA channels can be in busy (active) state at the same time. However, at each specific time at most two channels (one in read mode and another in write mode) can be enabled and have access to WB master interfaces. The read and write mode can be enabled for one active channel or two different active channels at the same time.

**BURST-SIZE-WB-ADDR** The read/write DMA burst size register. When the numbers of continuous read/written words in words becomes equal to the burst size, the enabled channel (the channel which has access to read or write WB interface) is disabled for one clock cycle. This channel will assert a request to the read/write round robin arbiter for accessing to the WB interface in the next clock cycle again, if it has not yet completed its task. Disabling channels allows the WB master interface to be accessed by other active master interfaces including other active channels which have data to be transferred more frequently. Setting smaller values prevents the enabled channel with large data size to hold the control of WB interface for a long period of time. The size of this register is defined by setting MAX-BURST-SIZE parameter in Verilog file. Setting a new burst size is only allowed when there is not any active channel in the DMA.

| Bit | log2(MAX-BURST-SIZE)-1 ... 0 |
|---|---|
| | BURST-SIZE-WB-ADDR |
| Read/Write | W |
| Initial Value | All valid bits set as one |

**DATA-SIZE-WB-ADDR** This register holds the size of data to be read/written in bytes. Each DMA channel has its own data-size register. The size of this register is defined using MAX-TRANSACTION-WIDTH Verilog parameter. Writing on this register is only permitted when its respective channel is in its ideal state (it is not busy).

| Bit | MAX-TRANSACTION-WIDTH-1 ... 0 |
|---|---|
| | DATA-SIZE-WB-ADDR |
| Read/Write | W |
| Initial Value | 0 |

**RD-STRT-WB-ADDR** This is the address pointer to the start location of the data to be read from the memory. The address must be given in byte (not words). The size of this register is equal to wishbone bus data width (Dw). Writing on this register is only permitted when its respective channel is in its ideal state (it is not busy).

| Bit | Dw-1 ... 0 |
|---|---|
| | RD-STRT-WB-ADDR |
| Read/Write | W |
| Initial Value | 0 |

**WR-STRT-WB-ADDR**

This is the address pointer to the start location of the data to be written in the memory. The address must be given in byte (not words). The size of this register is equal to wishbone bus data width (Dw). Writing on this register is only permitted when its respective channel is in its ideal state (it is not busy).

**Note**

The DMA's channel becomes active by written on this register. Hence, this register must be the last channel's register to be set by the user.

| Bit | Dw-1 | 0 |
|---|---|---|
| | WR-STRT-WB-ADDR | |
| Read/Write | W | |
| Initial Value | 0 | |

**DMA C header file**

```
#define dma_base_addr  0X93000000

// DMA internal register address definition
#define dma_STATUS_REG   (*((volatile unsigned int *) (dma_base_addr)))
#define dma_BURST_SIZE_ADDR_REG  (*((volatile unsigned int *) (
    dma_base_addr + 4)))
#define dma_DATA_SIZE_ADDR_REG(channel)  (*((volatile unsigned int *) (
    dma_base_addr + 8 +(channel<<5))))
#define dma_RD_START_ADDR_REG(channel)   (*((volatile unsigned int *) (
    dma_base_addr + 12 +(channel<<5))))
#define dma_WR_START_ADDR_REG(channel)  (*((volatile unsigned int *) (
    dma_base_addr + 16 +(channel<<5))))


// DMA function definition
#define dma_channel_is_busy(channel) ( (dma_STATUS_REG >> channel) & 0x1)


void dma_initial (unsigned int burst_size) {
     dma_BURST_SIZE_ADDR_REG  =  burst_size;
}


void dma_transfer (unsigned int channel, unsigned int read_start_addr,
    unsigned int data_size, unsigned int write_start_addr){
    // wait until DMA  channel is busy
    while ( dma_channel_is_busy(channel));
    dma_RD_START_ADDR_REG(channel)  = read_start_addr;
    dma_DATA_SIZE_ADDR_REG(channel)  =  data_size;
    dma_WR_START_ADDR_REG(channel)  = write_start_addr;
}
```