# VciMultiDma

## 1) Functional Description

As the VciDma component, this component moves data from a source memory buffer to a destination memory buffer. It is both a target and an initiator.

- It is addressed as a target to be configured for a transfer.
- It is acting as an initiator to do the transfer.

The VciMultiDma component supports up to 8 simultaneous DMA transfers, corresponding to 8 independant DMA channels. As there is only one VCI initiator port, the general arbitration policy between the active channels is round-robin.

The number of channels and the max burst length are constructor parameters. The burst length parameter must be multiple of 4 bytes.

This component makes the assumption that the VCI RDATA & WDATA fiels have 32 bits. The source buffer base address, the destination buffer base address and the buffer length mus be multiple of 4 bytes. The buffer length is not constrained to be a multiple of the burst length.

Each channel has its own set of memory mapped registers, and for each channel a specific IRQ can be optionally asserted when transfer is completed.

Each channel k has 5 memory-mapped registers :

- **DMA_SRC[k]** (Read / Write)

It defines the physical address of the source buffer.

- **DMA_DST[k]** (Read / Write)

It defines the physical address of the destination buffer.

- **DMA_LEN[k]** (Read / Write)

Writing in this register defines the length of transfer (in bytes), and starts the transfer. This register gets back to 0 when transfer is finished. This register can be used to test the DMA coprocessor status.

- **DMA_RESET[k]** (Write-only)

Writing any value into this pseudo-register makes a clean re-initialisation of the DMA coprocessor: The on-going VCI transaction is completed before the coprocessor returns the IDLE state. This write access must be used by the software ISR to aknowledge the DMA IRQ.

- **DMA_IRQ_DISABLED[k]** (Read / Write)

A non zero value disables the IRQ line. The RESET value is zero.

The aligned segment size associated to this VCI target is 256 bytes, as only 8 address bits are decoded : the 5 LSB bits define the target register, and the 3 MSB bits define the channel index.

For extensibility issues, you should access the DMA using globally-defined offsets, and you should include file `soclib/dma.h` in your software, it defines `DMA_SRC`, `DMA_DST`, `DMA_LEN`, `DMA_RESET`, `DMA_IRQ_DISABLED`.

This hardware component checks for segmentation violation, and can be used as a default target.

## 2) Component definition & usage

source:trunk/soclib/soclib/module/infrastructure_component/dma_infrastructure/vci_multi_dma/caba/metadata/vci_multi_dm

See SoclibCc/VciParameters

```
Uses( 'vci_multi_dma' )
```

## 3) CABA Implementation

### CABA sources

- interface : source:trunk/soclib/soclib/module/infrastructure_component/dma_infrastructure/vci_multi_dma/caba/source/include/
- implementation : source:trunk/soclib/soclib/module/infrastructure_component/dma_infrastructure/vci_multi_dma/caba/source/src/vci_

### CABA Constructor parameters

```
VciDma(
     sc_module_name name,   //  Component Name
     const soclib::common::MappingTable &mt,   // MappingTable
     const soclib::common::IntTab &srcid,  // Initiator index
     const soclib::common::IntTab &tgtid,  // Target index
     const size_t burst_size,   //  Max number of bytes transfered in a burst
     const size_t channels );  // Number of channels
```

### CABA Ports

- **p_resetn** : Global system reset
- **p_clk** : Global system clock
- **p_vci_target** : The VCI target port
- **p_vci_initiator** : The VCI initiator port
- **p_irq[k]** : As many output IRQ ports as the number of channels

## 4) TLM-DT implementation

The TLM-DT implementation is not available yet.