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 (CHANNELS) and the burst length (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)

A write access defines the length of the transfer (in bytes), and starts the transfer. A read access returns the DMA channel status. The relevant values for the status are:

Cnannel Status	Value
DMA_IDLE	0
DMA_SUCCESS	1
DMA_READ_ERROR	2
DMA_WRITE_ERROR	3
DMA_BUSY	>3

- **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.

In order to support various protection mechanisms, each channel akes 4K bytes in the address space. The segment size is 32 K bytes, and the segment associated to this peripheral must be aligned on a 32K bytes boundary. Only 8 address bits are decoded :

- The five bits ADDRESS[4:0] define the target register.
- The three bits ADDRESS[14:12] 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.