

VciDma Functional Description

This VCI component is both a target and an initiator.

- Addressing as a target allows to configure it for a transfer.
- Initiator will do the transfer

There is only one DMA context handled at a time

An IRQ is optionally asserted when transfer is finished.

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

Memory region layout

- DMA_SRC

The source of memory copy.

- DMA_DST

The destination of memory copy

- DMA_LEN

Length of transfer, in bytes. Writing to this register initiates the transfer, you should write to it after src and dst. This register gets back to 0 when transfer is finished.

- DMA_IRQ_ENABLED
 - ◆ Written to: A boolean enabling the IRQ line (0 is disabling)
 - ◆ Read from: A boolean indicating the completion of the transfer, (0 is completed)

Component usage

For extensibility issues, you should access the DMA using globally-defined offsets.

You should include file source:trunk/soclib/include/soclib/dma.h from your software, it defines DMA_SRC, DMA_DST, DMA_LEN, DMA_IRQ_ENABLED.

Sample code:

```
#include "soclib/dma.h"

static const volatile void* dma_address = 0xc0000000;

void * memcpy(void *dst, const void *src, const size_t len)
{
    volatile int *dma = ((int*)dma_address);

    soclib_io_set( dma, DMA_DST, dst );
    soclib_io_set( dma, DMA_SRC, src );
}
```

```

        soclib_io_set( dma, DMA_LEN, len );
        while( soclib_io_get( dma, DMA_LEN ) )
            ;
        return dst;
    }

```

(add -I/path/to/soclib/include to your compilation command-line)

Component definition

Available in source:trunk/soclib/desc/soclib/vci_dma.sd

Usage

VciDma has no other parameter than VCI ones, it may be used like others, see [SoclibCc/VciParameters](#)

```
Uses( 'vci_dma', **vci_parameters )
```

VciDma CABA Implementation

The caba implementation is in

- source:trunk/soclib/systemc/include/caba/target/vci_dma.h
- source:trunk/soclib/systemc/src/caba/target/vci_dma.cc

Template parameters:

- The VCI parameters

Constructor parameters

```

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

```

Ports

- sc_in<bool> **p_resetn** : Global system reset
- sc_in<bool> **p_clk** : Global system clock
- soclib::caba::VciTarget<vci_param> **p_vci_target** : The VCI target port
- soclib::caba::VciInitiator<vci_param> **p_vci_initiator** : The VCI initiator port
- sc_out<bool> **p_irq** : Interrupt port