VciChbufDma

1) Functional Description

This component is a multi-channels DMA controller supporting chained buffers. It can be used to move a stream from one set of chained buffers (SRC chbuf) to another set of chained buffers (DST chbuf), without involving software.

All buffers must have the same length, and must be aligned on a 32 bits word boundary. The buffer length must be the same for the SRC chbuf and for the DST chbuf.

A **chbuf descriptor** is a circular array of **buffer descriptors**. Each buffer descriptor occupies 64 bytes, but only the first 8 bytes (64 bits) contain useful information:

- The 48 LSB bits contain the buffer physical address
- The MSB bit 63 defines the buffer state (empty if 0)

The chbuf descriptor" base address must be a multiple of 64 bytes.

This DMA controller implements two mode to scan the SRC and DST chbufs:

- **IN_ORDER_FIFO**: Both the source chained buffers and the destination chained buffers are accessed in strict order, as defined by the SRC and DST chbuf descriptors. The access is blocking until the expected buffer is available. If the buffer is not available, the delay before retry is defined by the software addressable register CHBUF_PERIOD. This register must be non zero to activate this mode.
- **OUT_OF_ORDER**: The arc and DST chbuf descriptors are scanned. The first full SRC buffer found is read, and the first empty DST buffer found is written, with a round robin priority. This mode is activated when the CHBUF_PERIOD value is zero (default value).

This component supports both 32 bits and 64 bits VCI RDATA & WDATA fields.

The number of channels and the max burst size are constructor parameters:

- The number of channels (simultaneous transfers) cannot be larger than 8.
- The burst length (in bytes) must be a power of 2 no larger than 64, and is typically equal to the system cache line width.

Each channel k has 5 memory-mapped registers :

- **CHBUF_SRC[k]The chbuf descriptor address (CHBUF_DESC), and the number of chained**

buffers (CHBUF_NBUFS), as well as the elementary buffer size (BUF_SIZE) are software parameters that must be written in addressable registers *when launching a transfer between two chbufs.* - The elementary buffer size and all buffers base addresses must be multiple *of 4 bytes. If the source and destination buffers are not aligned on a burst* boundary, the DMA controler split the burst in two VCI transactions. In order to support various protection mechanisms, for each channel, *the channel addressable registers takes 4K bytes in the address space.* Only 8 address bits are decoded . - *The 5 bits ADDRESS[4:Ø] define the target register (see chbuf_dma.h)* - The 3 bits ADDRESS[14:12] define the selected channel.

For each channel, the relevant values for the channel status are:

- CHANNEL_IDLE : channel not running
- CHANNEL_SRC_DESC_ERROR : bus error accessing SRC CHBUF descriptor
- CHANNEL_DST_DESC_ERROR : bus error accessing DST CHBUF descriptor
- CHANNEL_SRC_DATA_ERROR : bus error accessing SRC CHBUF data
- CHANNEL_DST_DATA_ERROR : bus error accessing DST CHBUF data
- CHANNEL_BUSY : channel running

There is one private IRQ line for each channel, that is only used for bus error signaling, and is activated when channel[k] enters n error state. The channel can be reset by writing a nul value in register CHBUF_RUN[k], focing channel[k] to IDLE state. In order to support multiple simultaneous transactions, the channel *index is transmited in the VCI TRDID field. 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.

- **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	2
DMA_SUCCESS	0
DMA_READ_ERROR	1
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_chbuf_dma/caba/metadata/vci_chbuf_dn

See SoclibCc/VciParameters

```
Uses( 'vci_chbuf_dma' )
```

3) CABA Implementation

CABA sources

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

CABA Constructor parameters

```
VciChbufDma(
     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_max_length,   //  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.