VciChbufDma

1) Functional Description

This component is a multi-channels DMA controller supporting chained buffers. It can be used to move a data streams (such as network packets or video stream) from one set of chained buffers (SRC chbuf) to another set of chained buffers (DST chbuf), with minimal software activity.

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.

The state of each buffer must be defined in a variable called **status**. Each ?status? occupies 64 bytes, but only the LSB bit is useful (1 if the buffer is full, 0 if it is empty). A buffer and its status physical addresses must be 64 bytes aligned and must have the same extension (identical bits[43:32]).

A **chbuf descriptor** is a circular array of **buffer descriptors**. Each buffer descriptor occupies 64 bits:

- The 12 MSB bits contain the common extension of the buffer address and the buffer status address
- The 26 following bits contain the bits [31:6] of the buffer address
- The 26 LSB bits contain the bits [31:6] of the buffer status address

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

This DMA controller implements two modes 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 SRC and DST chbuf descriptors and status are scanned with a round robin priority. The first full SRC buffer found is read, and the first empty DST buffer found is written. This mode is activated when the CHBUF_PERIOD value is zero (default value).

A long as no error is reported, each channel FSM does not stop moving buffers, until it is reset by software.

> There is one IRQ per channel, that is activated each time a buffer has been sucessfully moved from source chbuf to destination chbuf, or when

an address error has been reported. This IRQ is acknowledged by a read command to the channel status register.

This component supports both 32 bits and 64 bits VCI RDATA & WDATA fields, and supports VCI addresses up to 64 bits. In order to support multiple simultaneous transactions, the channel index is transmitted in the VCI TRDID field.

The transfer between a SRC and a DST buffer is divided into several bursts, and more precisely into series of several pipelined bursts. In a series of pipelined bursts, all the read requests are sent successively and the responses are stored in a local fifo (one fifo per channel). Then the successive write commands are sent to the DST buffer.

The number of channels, the max burst length and the number of stages in the pipeline are constructor parameters:

- The number of channels (simultaneous transfers) cannot be larger than 8.
- The max burst length (in bytes) must be a power of 2 no larger than 64, and is typically equal to the system cache line width.
- The number of pipelined bursts cannot be larger than 4 (default parameter).

The total internal storage capacity for transferred data is (channels * pipelined_bursts * burst_max_length) bytes.

Each channel [k] has 10 memory-mapped 32 bits registers:

- CHBUF_RUN[k] (write-only) : channel running modes (see below)
- CHBUF_STATUS[k] (read-only) : channel status (see below)
- CHBUF_SRC_DESC[k] (read/write) : SRC chbuf descriptor 32 LSB bits physical address
- CHBUF_DST_DESC[k] (read/write) : DST chbuf descriptor 32 LSB bits physical address
- CHBUF_SRC_NBUFS[k] (read/write) : SRC chbuf number of buffers
- CHBUF_DST_NBUFS[k] (read/write) : DST chbuf number of buffers,
- CHBUF_BUF_SIZE[k] (read/write) : buffer size for both source & destination
- CHBUF_PERIOD[k] (read/write) : number of cycles between two status polling
- CHBUF_SRC_EXT[k] (read/write) : SRC chbuf descriptor 32 MSB bits physical address
- CHBUF_DST_EXT[k] (read/write) : DST chbuf descriptor 32 MSB bits physical address

For extensibility issues, you should access the DMA using globally-defined offsets, and you should include file `soclib/chbuf_dma.h` in your software. In order to support virtualisation mechanisms, for each channel, the channel addressable registers takes 4K bytes in the address space. The following address bits are decoded .

- The 5 bits ADDRESS[4:0] define the target register.
- The 3 bits ADDRESS[14:12] define the selected channel.

For each channel, various running modes can be set by writing in the CHBUF_RUN register:

Running Mode	value	
MODE_IDLE	0	soft reset request
MODE_NORMAL	1	Both SRC & DST buffers status are checked
MODE_NO_SRC_SYNC	2	SRC buffer status is not checked
MODE_NO_DST_SYNC	4	DST buffer status is not checked

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

Channel Status	value	
CHANNEL_IDLE	0	channel not running
CHANNEL_SRC_DESC_ERROR	1	bus error accessing SRC CHBUF descriptor
CHANNEL_DST_DESC_ERROR	2	bus error accessing DST CHBUF descriptor
CHANNEL_SRC_STATUS_ERROR	3	bus error accessing SRC BUF status
CHANNEL_DST_STATUS_ERROR	4	bus error accessing SRC BUF status
CHANNEL_DATA_ERROR	5	bus error accessing SRC or DST CHBUF data
CHANNEL_BUSY	>5	channel running

There is one private IRQ line for each channel, that is only used for bus error signalling, and is activated when channel[k] enters an error state. The channel can be reset by writing a null value in register CHBUF_RUN[k], forcing channel[k] to IDLE state.

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 uint32_t burst_max_length,   //  Max number of bytes transferred in a burst
     const uint32_t channels,   // Number of channels
     const uint32_t pipelined_bursts );  // Number of pipelined bursts
```

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.