VciMultiNic

1) Functional Description

The VciMultiNic component, is a multi-channels, GMII compliant, network controller for Gigabit Ethernet network.

It can support a throughput of 1 Gigabit/s, as long as the system clock frequency is larger or equal to the GMII clock frequency (ie 125 MHz).

Each channel defines a different destination MAC address, and each channel uses private RX and TX buffers. The number of channels is a constructor parameter, and cannot be larger than 8.

On the VCI side, this component makes the assumption that the VCI RDATA & WDATA fields have 32 bits.

Regarding the GMII physical interface, this simulation model supports three modes of operation, defined by a constructor parameter:

- **NIC_MODE_FILE**: Both the RX packets stream an the TX packets stream are read/written from/to dedicated files "nic_rx_file.txt" and "nic_tx_dile.txt", stored in the same directory as the top.cpp file.
- **NIC_MODE_SYNTHESIS**: The TX packet stream is still written to the "nic_tx_file.txt" file, but the RX packet stream is synthesised. The packet length (between 64 and 1538 bytes) and the source MAC address (8 possible values) are pseudo-random numbers.
- **NIC_MODE_TAP**: The TX and RX packet streams are send and received to and from the physical network controller of the workstation running the simulation.

It is a VCI target with no DMA capability : All data transfers must be performed by software, or by the external VciChbufDma component.

The data transfer unit between software and the NIC is a fixed-size **container**, containing an integer number of packets. A container is a 4 Kbytes buffer (1024 words of 32 bits).

The first 34 words define the fixed-format container header :

word0	NB_WORDS	NB_PACKETS
word1	PLEN[0]	PLEN[1]
...	...	...
word33	PLEN[64]	PLEN[65]

- NB_PACKETS is the actual number of packets in the container.
- NB_WORDS is the number of useful words in the container.
- PLEN[i] is the number of bytes for packet[i].

The packets are stored in the (1024 - 34) following words, The max number of packets in a container is 66 packets, and the packets are word-aligned.

Each channel contains two RX containers and two TX containers. The two RX containers are organized as a **RX_CHBUF**, and the two TX containers are organized as a **TX_CHBUF**.

A *CHBUF* is a set of M chained buffers, plus a *CHBUF_descriptor*. A *CHBUF_descriptor* is an array of M *buffer_descriptors*. A *buffer descriptor* contains the buffer base address PADDR (in the physical address space), and the buffer status STS (boolean : true => full / false => empty).

In order to simplify the L2/L3 cache coherence, each buffer descriptor is actually implemented as a 64 bytes record, even if only the first 8 bytes contain useful information(PADDR : bits[47:0]) / STS : bit 63).

This *CHBUF* organization allows the TX or RX buffers to be accessed by the VciChbufDma engine.

2) Addressable registers and buffers

In a virtualized environment each channel segment will be mapped in the address space of a different virtual machine. Each channel takes a segment of 32 Kbytes in the address space, to simplify the address decoding, but only 20K bytes are used.

- The first 4 Kbytes contain the RX_0 container data
- The next 4 Kbytes contain the RX_1 container data
- The next 4 Kbytes contain the TX_0 container data
- The next 4 Kbytes contain the TX_1 container data
- The next 4 Kbytes contain the channel addressable registers:

NIC_RX_DESC_LO_0	RX_0 descriptor low word	read/write
NIC_RX_DESC_HI_0	RX_0 descriptor high word	read/write
NIC_RX_DESC_LO_1	RX_1 descriptor low word	read/write
NIC_RX_DESC_HI_1	RX_1 descriptor high word	read/write
NIC_TX_DESC_LO_0	TX_0 descriptor low word	read/write
NIC_TX_DESC_HI_0	TX_0 descriptor high word	read/write
NIC_TX_DESC_LO_1	TX_1 descriptor low word	read/write
NIC_TX_DESC_HI_1	TX_1 descriptor high word	read/write
NIC_MAC_4	MAC address 32 LSB bits	read_only
NIC_MAC_2	MAC address 16 MSB bits	read_only
NIC_RX_RUN	RX channel activated	write_only
NIC_TX_RUN	TX channel activated	write_only

On top of the channels segments is the hypervisor segment, taking 4 Kbytes, and containing the global configuration registers: (all read/write). In a virtualized environment, the corresponding page should not be mapped in the virtual machines address spaces, as it should not accessed by the virtual machines.

Register name	function	Reset value
NIC_G_VIS	bitfield / bit N = 0 -> channel N is disabled	all inactive
NIC_G_ON	NIC active if non zero (inactive at reset)	inactive
NIC_G_BC_ENABLE	boolean / broadcast enabled if true	disabled
NIC_G_TDM_ENABLE	boolean / enable TDM dor TX if true	disabled
NIC_G_TDM_PERIOD	value of TDM time slot	
NIC_G_PYPASS_ENABLE	boolean / enable bypass for TX if true	enabled
NIC_G_MAC_4[8]	default MAC address 32 LSB bits for channel[i]	
NIC_G_MAC_2[8]	default MAC address 16 LSB bits for channel[i]	

The Hypervisor segment contains also various event counters for statistics (read/write)

NIC_G_NPKT_RX_G2S_RECEIVED	number of packets received on GMII RX port
NIC_G_NPKT_RX_G2S_DISCARDED	number of RX packets discarded by RX_G2S FSM
NIC_G_NPKT_RX_DES_SUCCESS	number of RX packets transmited by RX_DES FSM
NIC_G_NPKT_RX_DES_TOO_SMALL	number of discarded too small RX packets
NIC_G_NPKT_RX_DES_TOO_BIG	number of discarded too big RX packets
NIC_G_NPKT_RX_DES_MFIFO_FULL	number of discarded RX packets for fifo full
NIC_G_NPKT_RX_DES_CRC_FAIL	number of discarded RX packets for checksum
NIC_G_NPKT_RX_DISPATCH_RECEIVED	number of packets received by RX_DISPATCH FSM
NIC_G_NPKT_RX_DISPATCH_BROADCAST	number of broadcast RX packets received
NIC_G_NPKT_RX_DISPATCH_DST_FAIL	number of discarded RX packets for DST MAC
NIC_G_NPKT_RX_DISPATCH_CH_FULL	number of discarded RX packets for channel full
NIC_G_NPKT_TX_DISPATCH_RECEIVED	number of packets received by TX_DISPATCH FSM
NIC_G_NPKT_TX_DISPATCH_TOO_SMALL	number of discarded too small TX packets
NIC_G_NPKT_TX_DISPATCH_TOO_BIG	number of discarded too big TX packets
NIC_G_NPKT_TX_DISPATCH_SRC_FAIL	number of discarded TX packets because SRC MAC
NIC_G_NPKT_TX_DISPATCH_BROADCAST	number of broadcast TX packets received
NIC_G_NPKT_TX_DISPATCH_BYPASS	number of bypassed TX->RX packets
NIC_G_NPKT_TX_DISPATCH_TRANSMIT	number of transmit TX packets

For extensibility issues, you should access all these registers using the globally-defined offsets in file

source:trunk/soclib/soclib/module/connectivity_component/vci_multi_nic/include/soclib/multi_nic.h?

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

3) Component definition & usage

source:trunk/soclib/soclib/module/connectivity_component/vci_multi_nic/caba/metadata/vci_multi_nic.sd?

```
Uses( 'vci_multi_nic' )
```

4) CABA Implementation

CABA sources

- interface : source:trunk/soclib/soclib/module/connectivity_component/vci_multi_nic/caba/source/include/vci_multi_nic.h?
- implementation : source:trunk/soclib/soclib/module/connectivity_component/vci_multi_nic/caba/source/src/vci_multi_nic.cpp?

CABA Constructor parameters

```
VciMultiNic(
     sc_module_name name,   //  Component Name
     const soclib::common::IntTab &tgtid,  // Target index
     const soclib::common::MappingTable &mt,   // MappingTable
     const size_t channels,   // Number of channels
     const uint32_t mac4,   //  MAC address 32 LSB bits
     const uint32_t mac2,   //  MAC address 16 MSB bits
     const int mode );    // GMII physical interface modeling
```

CABA Ports

- **p_resetn** : Global system reset
- **p_clk** : Global system clock
- **p_vci** : The VCI target port
- **p_rx_irq[k]** : As many RX output IRQ ports as the number of channels
- **p_tx_irq[k]** : As many TX output IRQ ports as the number of channels

4) TLM-DT implementation

The TLM-DT implementation is not available yet.