

VciMasterNic

1) Functional Description

The VciMasterNic component, is a GMII compliant, network controller for Gigabit Ethernet network, with a built-in DMA capability.

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).

To improve the throughput, this component supports up to 4 channels, indexed by the source IP address for the received (RX) packets, and indexed by the destination IP address for the sent (TX) packets. The actual number of channels is an hardware parameter that cannot be larger than 8. 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 and the TX packets stream are read/written from/to dedicated files "nic_rx_file.txt" and "nic_tx_file.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 sent and received to and from the physical network controller of the workstation running the simulation.

The packet length can have any value, from 64 to 1542 bytes.

The minimal data transfer unit between software and the NIC is a 4K bytes **container**, containing an integer number of variable size packets. The max number of packets in a container is 66 packets.

The received packets (RX) and the sent packets (TX) are stored in two memory mapped software FIFOs, implemented as chained buffers. Each slot in these FIFOs is a 4 Kbytes container. The number of containers, defining the queue depth, is a software defined parameter.

The container format is defined below:

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.

For the DMA engines, a container has only two states (full or empty), defined by a single bit, called the container "status". To access both the container status, and the data contained in the container, the DMA engines use two physical addresses, that are packed in a 64 bits *container descriptor*:

- desc[25:0] contain bits[31:6] of the "full" status physical address.
- desc[51:26] contain bits[31:6] of the "buffer" physical address.
- desc[63:52] contain the common 12 physical address extension bits.

Inside the NIC controller, each channel implements a 2 slots chained buffer (two containers) for RX, and another 2 slots chained buffer (two containers) for TX. For each channel, the build-in RX_DMA engine moves the RX containers from the internal 2 slots chained buffer to the external chained buffer implementing the RX queue in memory. Another build-in TX-DMA engine moves the TX containers from the external chained buffer implementing the TX queue in memory, to the internal TX 2 slots chained buffer.

To improve the throughput for one specific channel, the DMA engines use *pipelined bursts*: The burst length cannot be larger than 64 bytes, but each channel send 4 pipelined VCI transactions to mask the round-trip latency. Therefore, the NIC controller can control up to 32 parallel VCI transactions (4 channels * 4 bursts * 2 directions). The CMD/RSP matching uses both the VCI TRDID and PKTID fields:

- the channel index is sent in TRDID[3:2]
- the burst index is sent in TRDID[1:0]
- the is_rx bit is sent in SRCID

2) Addressable registers and buffers

The addressable registers can be split in two classes: *global* registers, and *channel* registers.

2.1) global registers

These registers are used for global NIC configuration or status, and are not linked to a specific channel.

NIC_G_CHANNELS Read Only Returns actual number of channels
NIC_G_BC_ENABLE Read/Write Enable Broadcast if non zero
NIC_G_PERIOD Read/Write container status polling period
NIC_G_MAC_4 Read/Write MAC address 32 LSB bits
NIC_G_MAC_2 Read/Write MAC address 16 MSB bits
NIC_G_NPKT_RESET Write Only reset all packets counters

NIC_G_NPKT_RX_G2S_RECEIVED = 10, *number of packets received on GMII RX port*
NIC_G_NPKT_RX_G2S_DISCARDED = 11, *number of RX packets discarded by RX_G2S FSM*

NIC_G_NPKT_RX_DES_SUCCESS = 12, *number of RX packets transmitted by RX_DES FSM*
NIC_G_NPKT_RX_DES_TOO_SMALL = 13, *number of discarded too small RX packets (<60B)*
NIC_G_NPKT_RX_DES_TOO_BIG = 14, *number of discarded too big RX packets (>1514B)*
NIC_G_NPKT_RX_DES_MFIFO_FULL = 15, *number of discarded RX packets if fifo full*
NIC_G_NPKT_RX_DES_CRC_FAIL = 16, *number of discarded RX packets if CRC32 failure*

NIC_G_NPKT_RX_DISPATCH_RECEIVED = 17, *number of packets received by RX_DISPATCH FSM*
NIC_G_NPKT_RX_DISPATCH_BROADCAST = 18, *number of broadcast RX packets received*
NIC_G_NPKT_RX_DISPATCH_DST_FAIL = 19, *number of discarded RX packets if DST MAC not found*
NIC_G_NPKT_RX_DISPATCH_CH_FULL = 20, *number of*

discarded RX packets if channel full

NIC_G_NPKT_TX_DISPATCH_RECEIVED = 41, *number of packets received by TX_DISPATCH FSM* NIC_G_NPKT_TX_DISPATCH_TOO_SMALL = 42, *number of discarded too small TX packets (<60B)* NIC_G_NPKT_TX_DISPATCH_TOO_BIG = 43, *number of discarded too big TX packets (>1514B)* NIC_G_NPKT_TX_DISPATCH_TRANSMIT = 44, *number of transmit TX packets*

2.2) Channel registers

These registers are replicated

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 be 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 for 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 transmitted 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_master_nic.sd

```
Uses( 'vci_master_nic' )
```

4) CABA Implementation

CABA sources

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

CABA Constructor parameters

```
VciMultiNic(  
    sc_module_name name,    // Component Name  
    const soclib::common::IntTab &tgtd, // 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,        // GII physical interface modeling  
    const uint32_t inter_frame_gap); // delay between two packets
```

CABA Ports

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

4) TLM-DT implementation

The TLM-DT implementation is not available yet.