

## COM-5402SOFT IP/TCP SERVER/UDP/ARP/PING STACK for GbE VHDL SOURCE CODE OVERVIEW

### Overview

Gigabit-speed IP protocols like TCP/IP and UDP/IP can demand a high level of computation on processors. The trend has been to move the implementation of these fast but highly repetitive tasks to a TCP offload engine (TOE) to free the application processor from frequent interrupts.

The COM-5402SOFT is a generic Internet protocol stack (including the [VHDL source code](#)) designed to support 1Gbps throughputs on low-cost FPGAs. It is designed to achieve the maximal throughput theoretically possible for a given medium.

The following protocols are implemented in modular VHDL components: TCP server, UDP frames, ARP and PING. Ancillary components are also included for streaming, test signal generation and bit error rate measurement.

These components can be instantiated as needed for the application. For a TCP-IP server application (waiting for connections from clients), one must instantiate *packet\_parsing.vhd*, *arp.vhd*, *tcp\_server.vhd*, *tcp\_tx.vhd* and *txp\_rxbuf.vhd*. The maximum number of concurrent TCP connections can be adjusted prior to VHDL synthesis depending on the available FPGA resources.

Wireshark Libpcap network capture files can be used as receiver input for simulation purposes.

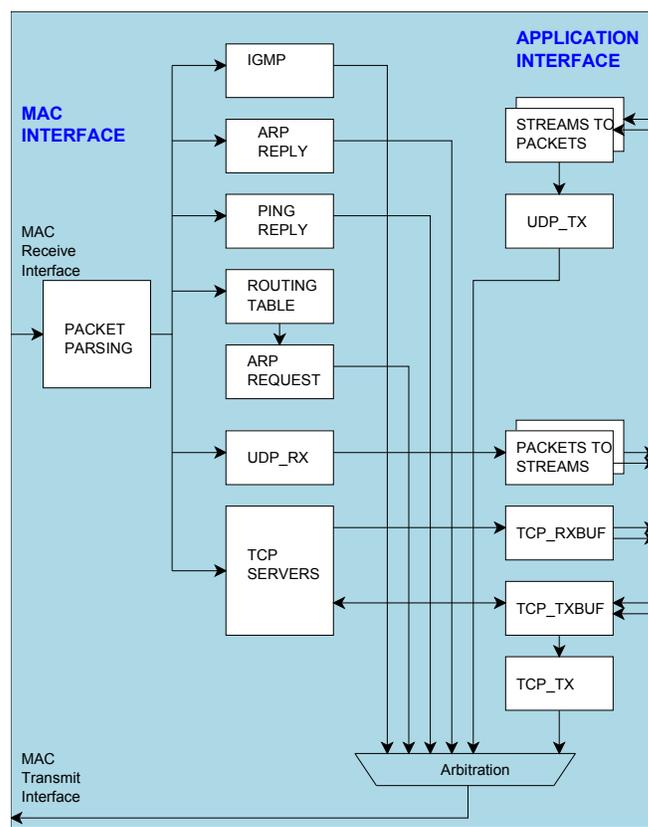
The code is written specifically for IEEE 802.3 Ethernet packet encapsulation (RFC 894), IPv4 protocols.

The code interfaces seamlessly with the COM-5401SOFT Tri-mode 10/100/1000 Mbps Ethernet MAC for the MAC / PHY layers implementation. Code to interface with the Xilinx Tri-Mode Ethernet MAC (TEMAC) is also included. More generally, the MAC interface is

generic and simple enough to interface with any Ethernet MAC component with minimum glue logic.

The component's very efficient implementation makes it suitable for multiple concurrent TCP and UDP streams instantiations within a small FPGA.

### Block Diagram



### Target Hardware

The code is written in generic VHDL so that it can be ported to a variety of FPGAs capable of running at 125 MHz or above. It does not use any Xilinx primitive.

## Device Utilization Summary

Device: Xilinx Spartan-6

	1 UDP tx, 1 UDP rx, ARP, Ping, routing table
Flip Flops	1938
LUTs	2815
RAMB16BWERs	3
DSP48A1s	0
GCLKs	2
DCMs/PLLs	0

	1 TCP server, ARP, Ping
Flip Flops	1745
LUTs	2822
RAMB16BWERs	3
DSP48A1s	0
GCLKs	2
DCMs/PLLs	0

	2 TCP servers, ARP, Ping
Flip Flops	2132
LUTs	3522
RAMB16BWERs	4
DSP48A1s	0
GCLKs	2
DCMs/PLLs	0

## Throughput Performance Examples

### Test setup1:

1 bidirectional connection between TCP server and TCP client over Gigabit Ethernet. 120 MHz FPGA processing clock. Measured sustained throughput: 452 Mbits/s concurrently in each direction.

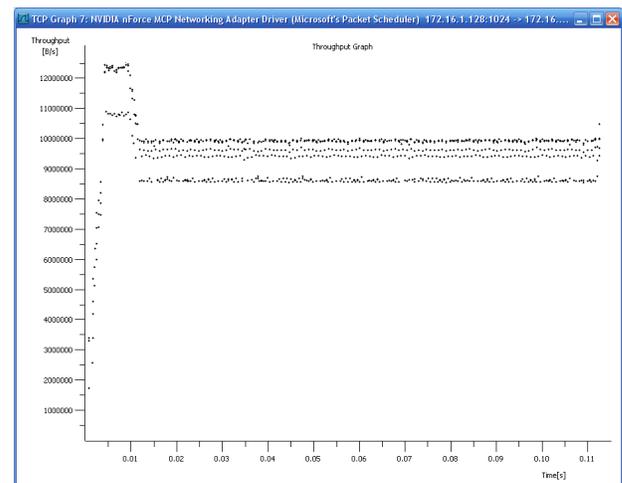
### Test setup2:

512-byte UDP packets sent point-to-point over a LAN cable. Xilinx Spartan-6 FPGA –2 speed grade.

Measured: 0 bit error, payload throughput 878.5 Mbits/s. This matches the theoretical throughput (accounting for Ethernet, IP and UDP overhead and slower (120 MHz) user clock). The maximum throughput for this UDP frame size is 915 Mbits/s when user clock is 125 MHz or above.

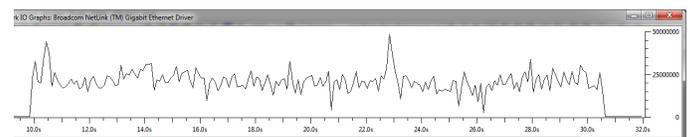
### Test setup 3:

TCP server transmit throughput on 100 Mbps LAN Wireshark measurement on receive PC. Average throughput 93 Mbps.



### Test setup 4:

TCP server sends 8Gbits to TCP Java client (see rxFileTCP in /sim directory) while Wireshark collects speed information. Point to point LAN connection from FPGA-based TCP server to PC.



Average throughput: 390.2 Mbits/s

## TCP Latency Performance Examples

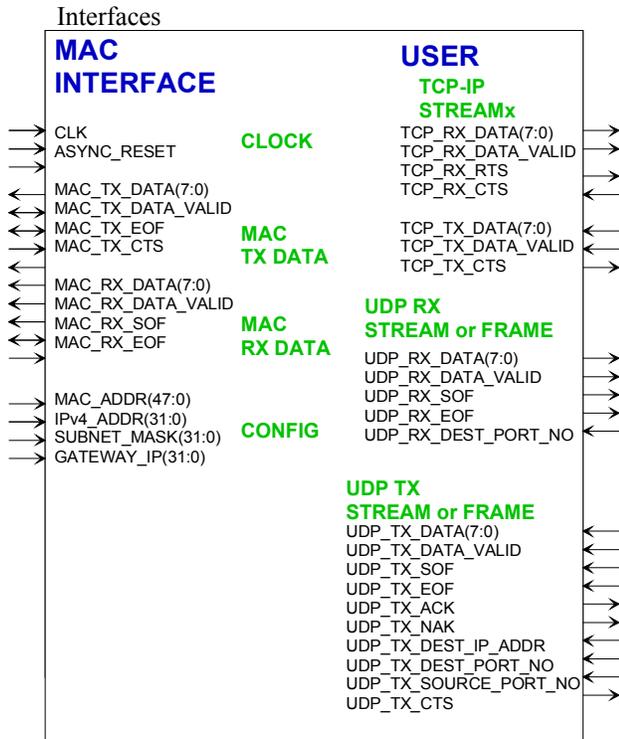
The transmit and receive latency depend on the frame length. For a maximum frame length of 1460 bytes, FPGA 125 MHz processing clock:

- Transmit latency (from the 1st byte of payload data input to the 1st byte of payload data output to the Ethernet MAC): 23.9 $\mu$ s
- Receive latency (from the 1st byte of Ethernet MAC input to the 1st byte of payload data output): 12.2 $\mu$ s

If latency is more important than throughput, the transmit segmentation threshold can be reduced to X payload bytes. In this more general case,

- Transmit latency (from the 1st byte of payload data input to the 1st byte of payload data output to the Ethernet MAC):  $0.5 + 2X/125 \mu\text{s}$

The receive latency (from the 1st byte of Ethernet MAC input to the 1st byte of payload data output):  $0.5 + X/125 \mu\text{s}$



## User Interface

This interface comprises three primary signal groups: MAC interface (direct connection to COM-5401SOFT MAC core or equivalent), TCP streams, UDP frames or UDP streams to/from the user application.

All signals are clock synchronous with a user-selected clock CLK (it does not have to be the same as the PHY clock). To guarantee a 1 Gbps throughput, a minimum 125 MHz clock speed is required.

The user interface is buffered by internal elastic buffers in both tx/rx directions.

## Configuration

The key configuration parameters are brought to the interface so that the user can change them dynamically at run-time. Other, more arcane, parameters are fixed at the time of VHDL synthesis.

### Pre-synthesis configuration parameters

The following configuration parameters are set prior to synthesis in the *com5402pkg.vhd* package or at the top level component *com5402.vhd*.

Configuration parameters in <i>com5402pkg.vhd</i>	Description
Maximum number of concurrent TCP streams	<b>NTCPSTREAMS_MAX</b>
Configuration parameters in <i>com5402.vhd</i>	Description
Transmit UDP enabled	<b>NUDPTX</b> '1' to enable, '0' to disable
Receive UDP enabled	<b>NUDPRX</b> '1' to enable, '0' to disable
IGMP enabled	<b>IGMP_EN</b> '1' to enable UDP multicast (which requires IGMP)
TCP streams	<b>NTCPSTREAMS</b> . Number of concurrent TCP streams instantiated for this component. Each additional TCP stream requires additional resources (RAM block, logic). Must be less than or equal to <b>NTCPSTREAMS_MAX</b>
Inactive input stream timeout	<b>TX_IDLE_TIMEOUT</b> When segmenting a transmit stream, a packet will be sent out with pending data if no new data was received within the specified timeout. Expressed as integer multiple of 4µs.
TCP keep-alive period	<b>TCP_KEEPALIVE_PERIOD</b> period in seconds for sending no data keepalive frames. "Typically TCP Keepalives are sent every 45 or 60 seconds on an idle TCP connection, and the connection is dropped after 3 sequential ACKs are missed"
TCP port numbers	Each TCP stream is identified by its 16-bit port number <b>TCP_LOCAL_PORTS(I)</b>

TCP keep-alive period	<b>TCP_KEEPALIVE_PERIOD</b> period in seconds for sending no data keepalive frames. "Typically TCP Keepalives are sent every 45 or 60 seconds on an idle TCP connection, and the connection is dropped after 3 sequential ACKs are missed"
Elastic buffer size	Customized I/O elastic buffer sizes for various TCP and UDP components.  Expressed number of address bits in Byte-wide RAM blocks. For example <b>ADDR_WIDTH</b> => 12 defines a 4KB buffer.  Also expressed as an integer number NBUFS of 16Kbits RAM blocks. NBUFS is typically restricted to 1,2,4 or 8 (see code comments).  Defined as generic parameters in the following components: <i>tcp_rxbufndemux2.vhd, tcp_txbuf.vhd, udp_tx, stream_2_packets.vhd.</i>

<i>Configuration parameters in arp_cache2.vhd</i>	<i>Description</i>
Routing table refresh period <b>REFRESH_PERIOD(19:0)</b>	Refresh period for this routing table. Expressed as an integer multiple of 100ms. Default value is 3000 (5 minutes).
<i>Configuration parameters in stream_2_packets.vhd</i>	<i>Description</i>
Maximum packet size when segmenting a stream to packets <b>MAX_PACKET_SIZE</b>	When segmenting a transmit stream, a packet will be sent out as soon as <b>MAX_PACKET_SIZE</b> bytes are collected. The recommended size is 512 bytes for a low overhead.
Retransmission timer <b>TX_RETRY_TIMEOUT</b>	A re-transmission attempt will be made periodically until routing information is available and the transmit path to the MAC is available. The retry period is expressed as an integer multiple of 4μs.

## Run-time configuration parameters

The user can set and modify the following controls at run-time. All controls are synchronous with the user-supplied global CLK.

<i>Run-time configuration</i>	<i>Description</i>
MAC address <b>MAC_ADDR(47:0)</b>	This network node 48-bit MAC address. The user is responsible for selecting a unique 'hardware' address for each instantiation.  Natural bit order: enter x0123456789ab for the MAC address 01:23:45:67:89:ab It is essential that this input matches the MAC address used by the MAC/PHY.
IPv4 address <b>IPv4_ADDR(31:0)</b>	Local IP address. 4 bytes for IPv4. Byte order: (MSB)192.68.1.30(LSB)
Multicast IP address <b>MULTICAST_IP_ADDR(31:0)</b>	to receive UDP multicast messages. One multicast address only 0.0.0.0 to signify that IP multicasting is not supported here.
Subnet Mask <b>SUBNET_MASK(31:0)</b>	Subnet mask to assess whether an IP address is local (LAN) or remote (WAN) Byte order: (MSB)255.255.255.0(LSB)
Gateway IP address <b>GATEWAY_IP(31:0)</b>	One gateway through which packets with a WAN destination are directed. Byte order: (MSB)192.68.1.1(LSB)
Transmit UDP destination IP address <b>UDP_TX_DEST_IP_ADDR</b>	The UDP destination IP address can be modified dynamically on a frame-by-frame basis.
Transmit UDP destination port number <b>UDP_TX_DEST_PORT_NO</b>	The UDP destination port number can be modified dynamically on a frame-by-frame basis.
Transmit UDP source port number <b>UDP_TX_SOURCE_PORT_NO</b>	The UDP source port number can be modified dynamically on a frame-by-frame basis.
Check UDP port number <b>CHECK_UDP_RX_DEST_PORT_NO</b>	check the received UDP frame destination port number matches <b>UDP_RX_DEST_PORT_NO</b> (1) or ignore it (0) In the latter case, the application is responsible for checking

	destination ports.
Receive UDP port number UDP_RX_DEST_PORT_NO	Local UDP port listening for incoming UDP frames. Receive and transmit UDP streams can use identical or different ports at the user's discretion.

### Limitations

This software does not support the following:

- IEEE 802.3/802.2 encapsulation, RFC 1042, only the most common Ethernet encapsulation.

Only one gateway is supported at any given time.

### Software Licensing

The COM-5402SOFT is supplied under the following key licensing terms:

1. A nonexclusive, nontransferable license to use the VHDL source code internally, and
2. An unlimited, royalty-free, nonexclusive transferable license to make and use products incorporating the licensed materials, solely in bitstream format, on a worldwide basis.

The complete VHDL/IP Software License Agreement can be downloaded from <http://www.comblock.com/download/softwarelicense.pdf>

## Configuration Management

The current software revision is 007x.

Directory	Contents
/doc	Specifications, user manual, implementation documents when applicable.
/src	.vhd source code, .constraint file, .pkg packages. One component per file.
/sim	Testbenches. Please note that end-to-end TCP simulation requires both TCP server and TCP client code (such as COM-5403SOFT)
/project1	Xilinx Vivado 2017 project Xilinx Vivado 2019 project
/use_example	Examples of interface with the Ethernet MAC (COM-5401SOFT and Xilinx TEMAC)

### VHDL development environment

The VHDL software was developed using the following development environment:

- (a) Xilinx Vivado 2019.2 for synthesis, place and route and VHDL simulation
- (b) Xilinx ISE 14.7 for synthesis, place and route

The entire project fits easily within a Xilinx Artix7-100T. Therefore, the ISE project can be processed using the free Xilinx WebPack tools.

### Ready-to-use Hardware

Use examples are available to run on the following Comblock hardware modules:

- COM-1800 FPGA + GbE LAN + DDR3 SODIMM SOCKET + ARM + NAND development platform  
<http://www.comblock.com/com1800.html>

All hardware schematics are available at <https://comblock.com/download.html>

## Top-Level VHDL hierarchy

- COM5402(Behavioral) (com5402.vhd) (15)
  - Inst\_TIMER\_4US : TIMER\_4US(Behavioral) (tim
  - Inst\_PACKET\_PARSING : PACKET\_PARSING(E
  - Inst\_ARP : ARP(Behavioral) (arp.vhd)
  - Inst\_PING : PING(Behavioral) (ping.vhd) (1)
    - BRAM\_DP2\_001 : BRAM\_DP2(Behavioral) |
  - WHOIS2\_X.WHOIS2\_001 : WHOIS2(Behavioral
  - ARP\_CACHE2\_X.ARP\_CACHE2\_001 : ARP\_CA
    - BRAM\_DP2\_001 : BRAM\_DP2(Behavioral) |
  - IGMP\_QUERY\_001x.IGMP\_QUERY\_001 : IGMP
  - IGMP\_QUERY\_001x.IGMP\_REPORT\_001 : IGM
  - Inst\_UDP2SERIAL : UDP2SERIAL(Behavioral) (
  - UDP\_RX\_X.UDP\_RX\_001 : UDP\_RX(Behaviors
  - UDP\_TX\_NZ.UDP\_TX\_001 : UDP\_TX(Behavior:
    - RAMB\_16\_S9\_S9\_Y[0].BRAM\_DP2\_001 : B
  - TCP\_SERVER\_X.TCP\_SERVER\_001 : TCP\_SE
  - TCP\_SERVER\_X.Inst\_TCP\_TX : TCP\_TX(Beha
  - TCP\_SERVER\_X.Inst\_TCP\_RXBUFNDOMUX2 :
    - BRAM\_DP2\_X[0].BRAM\_DP2\_001 : BRAM\_
    - BRAM\_DP2\_X[1].BRAM\_DP2\_001 : BRAM\_
  - TCP\_SERVER\_X.Inst\_TCP\_TXBUF : TCP\_TXBU
    - OB\_001 : BRAM\_DP2(Behavioral) (bram\_d|
    - BRAM\_DP2\_X[0].BRAM\_DP2\_001 : BRAM\_
    - BRAM\_DP2\_X[1].BRAM\_DP2\_001 : BRAM\_

The code is stored with one, and only one, component per file.

The root entity (highlighted above) is *COM5402.vhd*. It contains instantiations of the IP protocols and a transmit arbitration mechanism to select the next packet to send to the MAC/PHY.

The root also includes the following components:

- The *PACKET\_PARSING.vhd* component parses the received packets from the MAC and efficiently extracts key information relevant for multiple protocols. Parsing is done on the fly without storing data. Instantiated once.
- The *ARP.vhd* component detects ARP requests and assembles an ARP response

Ethernet packet for transmission to the MAC. Instantiated once.

- The *PING.vhd* component detects ICMP echo (ping) requests and assembles a ping echo Ethernet packet for transmission to the MAC. Instantiated once.
- The *WHOIS2.vhd* component generates an ARP request (broadcast) packet requesting that the target identified by its IP address responds with its MAC address.
- The *ARP\_CACHE2.vhd* component is a shared routing table that stores up to 128 IP addresses with their associated 48-bit MAC addresses and a 'freshness' timestamp. An arbitration circuit is used to arbitrate the routing request from multiple transmit instances. Instantiated once.
- The *IGMP\_QUERY.vhd* component detects a valid IGMP membership query and triggers a response when applicable.
- The *IGMP\_REPORT.vhd* component sends an IGMP membership report out to whom it may concern.
- The flexible *UDP\_TX.vhd* component encapsulates a data packet into a UDP frame addressed from any port to any port/IP destination. Instantiated once, irrespective of the number of source or destination UDP ports.
- The *UDP\_RX.vhd* component validates received UDP frames and extracts the data packet within. As the validation is performed on the fly (no storage) while received data is passing through, the validity confirmation is made available at the end of the packet. The calling application should therefore be able to 'backtrack' upon receiving an invalid packet. Instantiated once, irrespective of the number of UDP ports being listened to. Although this component is written for one port, it can very easily be modified to accommodate several ports (follow the PORT\_NO signal). Therefore, there is never any need to instantiate more than one component.
- The *TCP\_SERVER.vhd* component is the heart of the TCP protocol. It is written parametrically so as to support NTCPSTREAMS concurrent TCP

connections. It essentially handles the TCP state machine of a TCP server: initially listening for connection requests from remote TCP clients, establishing and tearing down the connections and managing flow control while the connections are established.

- The *TCP\_TX.vhd* component formats TCP tx frames, including all layers: TCP, IP, MAC/Ethernet. It is common to all concurrent streams.
- The *TCP\_TXBUF.vhd* component stores TCP tx payload data in individual elastic buffers, one for each transmit stream. The buffer size is configurable prior to synthesis as  $NBUFS * 16Kbits$  RAM blocks.
- The *TCP\_RXBUFNDEMUX2.vhd* component demultiplexes several TCP rx streams. It does not include any elastic buffer. It is expected that the user will instantiate elastic buffers if the application requires it. Data bytes are received in sequence without gaps or backtracking.

Additional components are also provided for use during system integration or tests. These components require easy manual editing to connect to the top level fabric.

- *DHCP\_CLIENT.vhd* to fetch an IP address dynamically from an external DHCP server.
- *DHCP\_SERVER.vhd* to dynamically lease IP addresses from a group of consecutive addresses when queried by DHCP clients.
- *STREAM\_2\_PACKETS.vhd* segments a continuous data stream into packets. The transmission is triggered by either the maximum packet size or a timeout waiting for fresh stream bytes.
- *PACKETS\_2\_STREAM.vhd* reassembles a data stream from received valid packets while discarding invalid packets. The packet's validity is assessed at the end of packet. It is designed to connect seamlessly with the *TCP\_RX.vhd* component.
- *LFSR1IP.vhd* generates a pseudo-random binary stream PRBS11 for use during throughput and bit error rate tests. It is capable of generating 1 Gbps (8 bit per clock @ 125 MHz).

- *BER2.vhd* synchronizes with a received data stream and counts bit errors. It is also capable of working at 1 Gbps.

## VHDL simulation

A testbench (tb\*.vhd), located in the /sim directory, can be used to validate the source code through VHDL simulation. However, because of the interactive nature of the TCP and DHCP protocols, other IP cores are needed for such simulations:

- TCP end-to-end simulation is only possible when a TCP client (COM-5403SOFT or equivalent) are present.
- Simulation of DHCP dynamic assignment of IP addresses is only possible when a DHCP client (COM-5403SOFT or equivalent) are present.

Warning: Because of the COM-5402SOFT maturity, a few testbenches used during development are kept in the /sim directory 'just in case'. They may require minor editing to reflect the features added over time. Only the testbenches with recent timestamps are ready to use 'out-of-the box'.

## Clock / Timing

The software uses one synchronous clock CLK. The clock should be at least 125 MHz in order to take full advantage of the Gbit Ethernet speed. The code can operate properly at less than 125 MHz, albeit at reduced throughput.

The software uses one synchronous clock CLK. The clock should be at least 125 MHz in order to take full advantage of the Gbit Ethernet speed. The code can operate properly at less than 125 MHz, albeit at reduced throughput.

The code is written to run at 125 MHz on the following targets:

- Xilinx Spartan-6 –2 speed grade with 2 concurrent TCP streams instantiated.
- Any Xilinx 7-series FPGA/SoC, even at the lowest speed grade.



## **Libcap File Player**

Real network packets captured by the popular Wireshark LAN analyzer can be used as realistic stimulus for the COM-5402 software. The *tocom5402.vhd* test bench reads a libpcap-formatted file as captured by Wireshark and feeds it to the COM-5402 receive path. The input file must be named *input.cap* and be placed in the same directory as the ISE project.

The libpcap file format is described in <http://wiki.wireshark.org/Development/LibpcapFileFormat>

Note that Wireshark is sometimes unable to capture checksum fields when the PC operating system offloads the checksum computation to the network interface hardware. In order to still be allowed to simulate, set `SIMULATION := '1'` in the generic map section of the COM5402.vhd component. When doing so,

- (a) the IP header checksum is considered valid when read as `x"0000"`.
- (b) The TCP checksum computation is forced to a valid `0x0001`, irrespective of the 16-bit field captured by Wireshark.

## **Use Case#1: Two TCP servers, No UDP**

Configuration steps:

- (a) Instantiate one COM5402.vhd component for each MAC and Ethernet transceiver. Connect the MAC signals between the COM5402.vhd component and the MAC (COM5401.vhd for example).
- (b) Set the number of concurrent TCP streams in the com5402pkg.vhd package.  
constant NTCPSTREAMS: integer := 2;  
Likewise, define the number of UDP streams as zero.  
constant NUDPTX: integer := 0;  
constant NUDPRX: integer := 0;
- (c) Define the ports for each stream in com5402.vhd:  
TCP\_LOCAL\_PORTS(0) <= x"0400"; -- port 1024 for stream #0  
TCP\_LOCAL\_PORTS(1) <= x"0404"; -- port 1028 for stream #1
- (d) Enter the MAC/IP/Gateway addresses and subnet mask at the COM5402.vhd input.
- (e) Connect the two streams to the COM5402.vhd top level I/Os.  
-- Rx streams  
TCP\_RX\_DATA =>  
TCP\_RX\_DATA\_VALID =>  
TCP\_RX\_CTS =>  
-- Tx streams  
TCP\_TX\_DATA =>  
TCP\_TX\_DATA\_VALID =>  
TCP\_TX\_CTS =>

## **Use Case#2: UDP streaming (tx/rx), No TCP server**

Configuration steps:

- (a) Instantiate one COM5402.vhd component for each MAC and Ethernet transceiver. Connect the MAC signals between the COM5402.vhd component and the MAC (COM5401.vhd for example).
- (b) In the com5402pkg.vhd package set the following constants as follows:  
constant NTCPSTREAMS: integer := 0;  
constant NUDPTX: integer := 1;  
constant NUDPRX: integer := 1;

- (c) Define the UDP tx and rx local ports in com5402.vhd:  
 UDP\_RX\_DEST\_PORT\_NO <= x"0400";  
 UDP\_TX\_SOURCE\_PORT\_NO <= x"0404"; (can be changed for each tx packet)
- (d) Enter the MAC/IP/Gateway addresses and subnet mask at the COM5402.vhd input.
- (e) If the application calls for UDP streaming instead of raw UDP frames, instantiate the PACKETS\_2\_STREAM.vhd and STREAM\_2\_PACKETS.vhd components.

## **Components overview**

### **WHOIS2.VHD**

Before sending any IP packet, one must translate the destination IP address into a 48-bit MAC address. A look-up table (within *arp\_cache2.vhd*) is available for this purpose. Whenever there is no entry for the destination IP address in the look-up table, an ARP request is broadcasted to all asking for the recipient to respond with an ARP response. The main task of the *whois2.vhd* component is to assemble and send this ARP request.

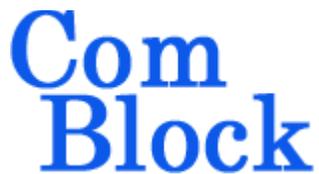
### **ARP\_CACHE2.VHD**

A block RAM is used as cache memory to store 128 MAC/IP/Timestamp records. Each record comprises (a) a 48-bit MAC address, (b) the associated 32-bit IPv4 address and (c) a timestamp when the information was last updated. The information is updated continuously based on received ARP responses and received IP packets. The component keeps track of the oldest record, which is the next record to be overwritten.

Whenever the application requests the MAC address for a given IP address (search key), this component searches the block RAM for a matching IP address key. If found, it returns the associated MAC address. If the search key is not found or is older than a refresh period, this component asks *whois2.vhd* to send an ARP request packet.

The code is optimized for fast access. Response time is between 0.1 and 1.33 us depending on the record location in memory.

This routing table is instantiated once and shared among multiple instances requiring routing services. An arbitration circuit is used to sequence routing requests from several transmit instances (for example several instantiations of the UDP\_TX component).



## COM-5402SOFT IP/TCP SERVER/UDP/ARP/PING STACK for GbE VHDL SOURCE CODE OVERVIEW

### **ComBlock Compatibility List**

<b>FPGA development platform</b>
<a href="#">COM-1800</a> FPGA (Xilinx Artix7-100T) + DDR3 SODIMM + GbE LAN development platform
<b>Software</b>
<a href="#">COM-5401SOFT</a> Tri-mode 10/100/1000 Mbps Ethernet MAC. VHDL source code.
Xilinx TEMAC, rev 9.0 or above
<a href="#">COM-5403SOFT</a> IP/UDP/TCP CLIENT/ARP/PING stack. VHDL source code.

### **Acronyms**

Acronym	Definition
BER	Bit Error Rate
BRAM	Dual-port Block RAM
CTS	Clear To Send, a flow-control signal allowing the data source to send data.
IP	Internet Protocol
LSb	Least Significant bit
MSb	Most Significant bit
PRBS-11	Pseudo-Random Binary Sequence, 2047-bit period
Rx	Receive
TCP	Transmission Control Protocol
Tx	Transmit
UDP	User Datagram Protocol

### **ComBlock Ordering Information**

COM-5402SOFT IP/TCP  
SERVER/UDP/ARP/PING PROTOCOL STACK,  
VHDL SOURCE CODE

ECCN: EAR99

MSS • 845 Quince Orchard Boulevard Ste N •  
Gaithersburg, Maryland 20878-1676 • U.S.A.  
Telephone: (240) 631-1111  
Facsimile: (240) 631-1676  
E-mail: [sales@comblock.com](mailto:sales@comblock.com)

MSS • 845 Quince Orchard Boulevard Ste N • Gaithersburg, Maryland 20878-1676 • U.S.A.  
Telephone: (240) 631-1111 Facsimile: (240) 631-1676 [www.ComBlock.com](http://www.ComBlock.com)

© MSS 2020 Issued 10/2/2020