

COM-5402SOFT IP/TCP/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.

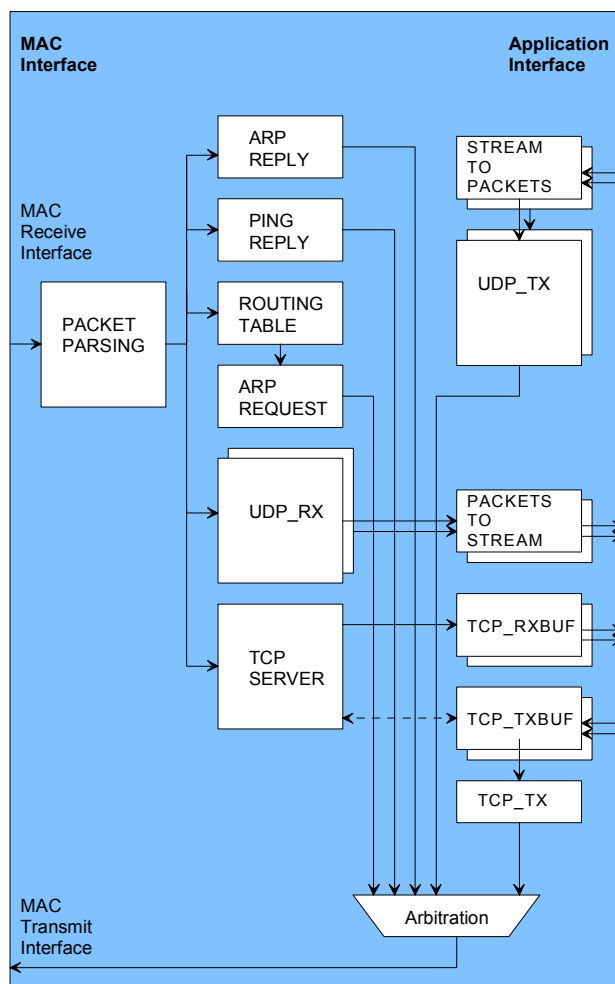
The code is written specifically for IEEE 802.3 Ethernet packet encapsulation (RFC 894), IPv4 protocols.

The code interfaces seamlessly with the COM-5401SOFT Tri-mode 10/100/1000 Mbps Ethernet MAC for the MAC / PHY layers implementation. However, 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. For example, it can be configured for 2 PHYs, 2 TCP servers and 2 UDP streams in a small Spartan-6 XC6SLX16 FPGA..

Block Diagram



Target Hardware

The code is written in generic VHDL so that it can be ported to a variety of FPGAs. The code was developed and tested on a Xilinx Spartan-6 FPGA.

It can be easily ported to any Xilinx Virtex-5, Virtex-6, Spartan-6 FPGAs and other FPGAs capable of running at 125 MHz or above.

Device Utilization Summary

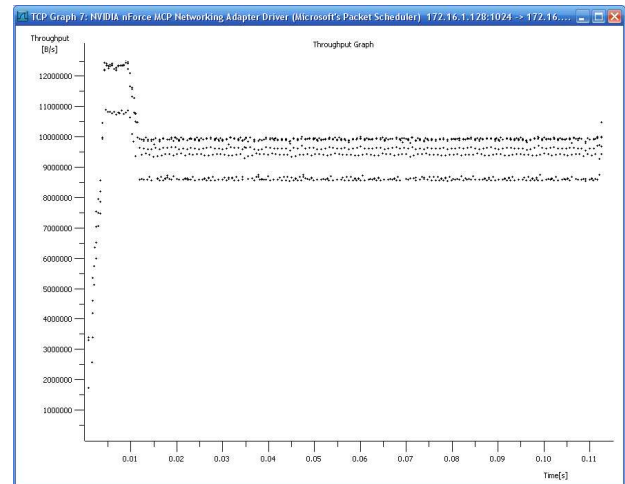
Device: Xilinx Spartan-6

	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

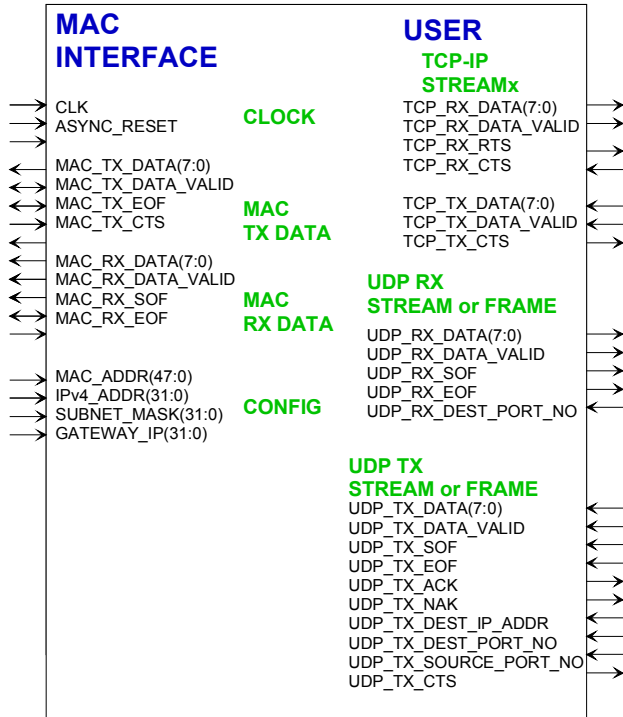
	2 TCP servers, ARP, Ping
Flip Flops	2132
LUTs	3522
RAMB16BWERs	4
DSP48A1s	0
GCLKs	2
DCMs/PLLs	0

Performance Examples



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

Interfaces



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
Number of concurrent TCP streams	NTCPSTREAMS. Each additional TCP stream requires additional resources (RAM block, logic).
Number of transmit UDP ports enabled	NUDPTX
Number of receive UDP ports enabled	NUDPRX
Configuration parameters in <i>com5402.vhd</i>	Description
TCP port numbers	Each TCP stream is identified by its 16-bit port number TCP_LOCAL_PORTS(I)
Transmit UDP port numbers	Each UDP stream is identified by its 16-bit port number UDP_TX_LOCAL_PORTS(I)
Receive UDP port numbers	Each UDP stream is identified by its 16-bit port number UDP_RX_LOCAL_PORTS(I) Receive and transmit UDP streams can use identical or different ports at the user's discretion.
Elastic buffer size	Expressed as an integer number NBUFS of 16Kbits RAM blocks. NBUFS is restricted to 1,2,4 or 8.

<i>Configuration parameters in arp_cache2.vhd</i>	<i>Description</i>
Routing table refresh period REFRESH_PERIOD(19:0)	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 MAX_PACKET_SIZE	When segmenting a transmit stream, a packet will be sent out as soon as MAX_PACKET_SIZE bytes are collected. The recommended size is 512 bytes for a low overhead.
Inactive input stream timeout TX_IDLE_TIMEOUT	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.
Retransmission timer TX_RETRY_TIMEOUT	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 MAC_ADDR(47:0)	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 IPv4_ADDR(31:0)	Local IP address. 4 bytes for IPv4. Byte order: (MSB)192.68.1.30(LSB)
Subnet Mask	Subnet mask to assess whether

SUBNET_MASK(31:0)	an IP address is local (LAN) or remote (WAN) Byte order: (MSB)255.255.255.0(LSB)
Gateway IP address GATEWAY_IP(31:0)	One gateway through which packets with a WAN destination are directed. Byte order: (MSB)192.68.1.1(LSB)

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>

Reference documents

[1] ComBlock COM-1600 FPGA + ARM + USB2.0+ DDR2 + NAND development platform
www.comblock.com/com1600.html

[2] ComBlock COM-5401 4-Port 10/100/1000 Mbps Ethernet Transceivers
www.comblock.com/com5401.html

Configuration Management

The current software revision is 1.

[a] VHDL source code in directory:
com-5402\src\

[b] Xilinx ISE project file:
com-5402\com-5402_ISE131.xise

[c] .ucf constraint file example when used on the COM-1600 FPGA platform:
com-5402\src\COM5402.ucf

[d] test benches in directory:
com-5402\sim\

[e] use example, .ngc for Spartan-6 and instantiation template are in directory:
com-5402\use_example

[f] Test components (pseudo random binary sequence generator, bit error rate measurement, stream to packets segmentation, etc) are in directory
com-5402\use_example\src

VHDL development environment

The VHDL software was developed using the following development environment:

- Xilinx ISE 13.1 with XST as synthesis tool
- Xilinx ISE Isim as VHDL simulation tool

Ready-to-use Hardware

The binary component (.ngc) is freely available for use on the following Comblock hardware modules:

- COM-1600 FPGA + ARM + DDR2 + NAND + USB2 development platform
- COM-1500 FPGA + ARM + DDR2 SODIMM + NAND + USB2 development platform
- COM-5401 4-Port 10/100/1000 Mbps Ethernet Transceivers

See the following code templates:
comblock.com/download.html#COM1600template

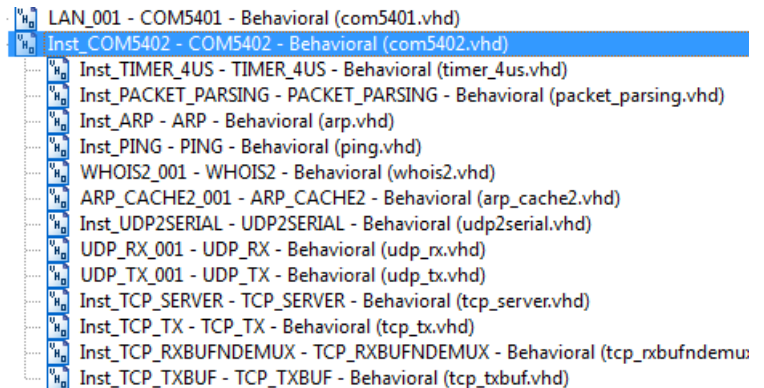
All hardware schematics are available in this CD.

Xilinx-specific code

The VHDL source code is written in generic VHDL with few Xilinx primitives. No Xilinx CORE is used. The Xilinx primitives are:

- IBUF
- IBUFG
- BUFG (global clocks)
- RAM block: RAMB16_S9_S9

Top-Level VHDL hierarchy



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 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_RXBUFNDEMUX.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.

- *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.
- *LFSR11P.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.

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 code is written to run at 125 MHz on a Xilinx Spartan-6 –2 speed grade with 2 concurrent TCP streams instantiated.

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).

ComBlock Compatibility List

FPGA development platform
COM-1600 FPGA + ARM + DDR2 + USB2 + NAND development platform
COM-1500 FPGA + DDR2 SODIMM socket + ARM development platform
Network adapter
COM-5401 4-port 10/100/1000 Mbps Ethernet Transceivers
Software
COM-5401SOFT Tri-mode 10/100/1000 Mbps Ethernet MAC. VHDL source code.
COM-5402SOFT IP/UDP/TCP/ARP/PING stack. VHDL source code.

ComBlock Ordering Information

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

Contact Information

MSS • 18221-A Flower Hill Way •
Gaithersburg, Maryland 20879 • U.S.A.
Telephone: (240) 631-1111
Facsimile: (240) 631-1676
E-mail: info@comblock.com