Prevas AB
Legeringsgatan 18
SE-721 03 Västerås
Sweden

Phone: +46 (0)21 – 360 19 00
Fax: +46 (0)21 – 360 19 29
Email: jonas.lindstedt@prevas.se
URL: www.prevas.se

Features

- Send and receive UDP-packets over Ethernet
- Supports transfer rates at 10/100/1000Mbit/s
- Send and receive multicast messages
- Replies to echo ping-requests
- Supports reception of Pause Frames to temporarily stop all traffic from the Core
- ARP-table with 4 entries(Can be extended upon request)
- Check incoming/outgoing packets against CRC errors
- Checks UDP checksum on received packet
- Send 1518 byte wide packets
- Programmable IP/MAC/Default Gateway address
- Fully synchronous implementation, which uses block- RAM for storage
- Compatible with standard Ethernet transceivers
  using GMII interface, RGMII optional
- TCP-Channel

Applications

The core is suitable for implementing high-speed communication with one or more PC:s or other udp_ip cores when a UART or parallel port is not fast enough. It is also great for sending large amount of data in one packet.

General Description

The UDP/IP core sends and receives Ethernet packets to a 10/100/1000Mbit transceiver.

CORE Facts

Provided with Core	
Documentation	User's Reference Manual
Design File Formats	EDIF netlist; VHDL Source RTL (available at extra cost)
Constraints Files	Udp_ip.ucf
Verification	VHDL test bench
Instantiation templates	VHDL
Reference designs & application notes	UDP-packet loopback program, with PC-interface
Additional Items	None
Simulation Tool Used	
Modelsim v6.3	
Support	
Support provided by [Prevas AB]	

The core has a unique MAC-address and an IP-address that can be changed at any time.

The main purpose of the core is to transfer UDP-packets to one or more target on a LAN-network.

When UDP packets are sent to the core's IP-address, it will be received and stored in RAM. Status flags are then activated to signal to the application layer that a new packet has been received. A received packet with faulty CRC or with a packet size larger than 1518 bytes will not be processed. Check of incoming UDP checksum can be enabled through a generic constant. The core can also be configured to reject packets that are larger than a specified length.

New UDP-packets to be transmitted is written to the UDP/IP core RAM memory. When the application layer has finished writing data it will activate the send request flag and the packet will be transmitted to the target. If the TX UDP-packet IP-address is not in the ARP table, an automatic ARP-request will be sent to get the MAC-address.

The core will respond to ARP requests that are targeted to the core's IP-address. The purpose of this packet is for other Ethernet nodes to get the core's MAC-address. The IP-core will also send ARP-requests by itself either manually or automatically.

Family	Example Device	Fmax (MHz)	Slices	BRAM	GCLK	Design Tools
Spartan-3E	XC3S1600E-5	85/125	1363	5	2	ISE 10.i

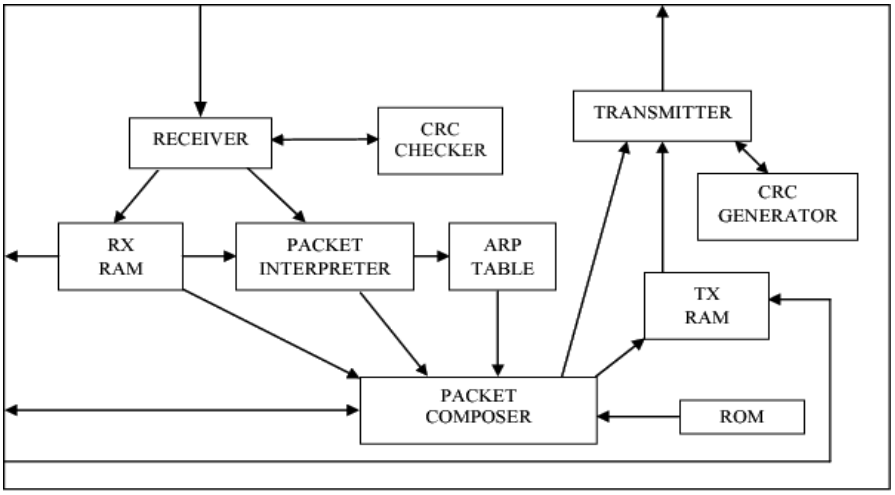

Figure 1. Block scheme over UDP IP core.

The core sends two ARP-requests with 1 sec of timeout delay each before an ARP-response timeout occur. ICMP echo reply messages from other PC:s running in an windows environment, will be responded with an echo reply message. The ARP-table has four entries. It will store IP addresses together with MAC addresses.

To inhibit transmission of data frames for a specified period of time, the IP-core supports reception of pause frames which stops all traffic from the core. After reception of a pause frame, the core will be suspended until the specified pause time has expired. Reception of a pause frame does not affect the transmission of a frame that has already been submitted by the application layer.

Available transfer mode is 10/100/1000Mbit.

Functional Description

As shown above and explained below, the UDP/IP core includes six major blocks: Receiver, Transmitter, CRC checker, CRC generator, packet interpreter, packet composer and lan control. Above that, the memory blocks and ARP-table is included. These blocks are also described below.

Receiver

This block handles the incoming messages. Data arrives in nibble-packets. It gradually stores the message in RX-RAM. A message larger than 1518 bytes will be rejected, i.e the core does not support jumbo frames. A message that doesn't match the core or the broadcast MAC address will not be stored. The exception is the globally assigned 48-bit multicast address 01-80-C2-00-00-01 which is used for MAC control frames such as a Pause frame. The receiver has two modes: 10/100Mbit uses nibble wide databus and 2,5/25MHz clk or 1000Mbit which uses a byte wide bus at 125MHz.

CRC Checker

32 bit CRC is calculated on incoming packets. If it doesn't match with the received CRC, the packet is thrown away and the CRC error flag is set.

Packet Interpreter

This block looks at certain fields in the received packet and decides if it is a supported packet or not. If it is a supported packet the block initiates the sending of a response packet. If it is an UDP/TCP-packet it will flag to the application that a new UDP/TCP-packet has arrived. The IP/MAC address will be saved in the ARP-table when the core receives an ARP-response message.

Packet Composer

From here packets are put together. Four types of packets can be sent.

- ARP-requests
- ARP-responses
- ICMP echo-replies
- UDPs
- TCPs

Depending one which packet that will be sent, different actions are taken. Parts of each message is read from a ROM and copied to TX-RAM. After that, other packet components like ip/mac-addresses, checksums, lengths, identifier etc are added to the message. UDP messages will be put together when the UDP transmit request signal has been set from the superior application layer. When TCP transmit request signal is set the packet composer adds ip/mac-addresses to the packet and lets the transmitter block transmit the packet.

Transmitter

This block reads data from TX-RAM, puts out the transmit packet on to the databus and sets control signals. At the beginning, 8 bytes of preamble are sent, where the last one is a start-of-frame nibble. At the end of each packet 4 bytes of CRC checksum is sent. The transmitter has two modes: 10/100Mbit uses nibble wide databus and 2,5/25MHz clk or 1000Mbit which uses a byte wide bus at 125MHz.

CRC Generator

While the packet is sent, CRC is calculated, one byte at a time. It uses the CRC32 polynom for Ethernet. The polynom looks like this:

x^32 + x^26 + x^23 + x^22 + x^16 + x^12 + x^11 + x^10 + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1.

RAM/ROM

The base of all messages is stored in a block ROM. Two block-RAMs are also included. Each block-RAM is 2048*16 bits and can hold two packets at the same time. The RX-RAM stores up to two incoming messages, i.e a new packet can be received while interpreting an already received packet. The TX-RAM works in a similar fashion and holds the transmit messages, where a new TX packet may be composed and written to the RAM at the same time as an already composed and stored packet is being transmitted.

ARP-table

The ARP-table stores IP/MAC addresses. There are four entries in the table. This means the core can send UDP packets to four different targets without doing any extra ARP-requests. The table works as a FIFO. The oldest data in the table will be replaced when a new entry is needed.

UDP/IP I/O Signals

The signal names of the UDP core are shown in Table 2.

Signal	Direction	Description
clk	Input	System Clock, 125 Mhz
reset_n	Input	Master Reset (Asynchronous)
phy_reset_n	Output	Transceiver master reset
rxdata(7:0)	Input	Transceiver receiver databus
rx_clk	Input	Receive clock line from PHY
rx_dv	Input	Receiver data valid strobe from transceiver
rx_er	Input	Phy receive error signal
tx_clk	Input	Transmit clock from PHY
gtx_clk	Output	Transmit clock to transceiver
txdata(7:0)	Output	Transceiver transmitter databus
tx_en	Output	Transmit data enable to transceiver
tx_er	Output	Transmit error line
udp_pkt_rec	Output	New UDP packet received
udp_pkt_acq	Input	packet received acknowledge
udp_data_rec(15:0)	Output	Databus for RX-RAM
udp_data_tra(15:0)	Input	Databus for TX-RAM
udp_addr_rec(9:0)	Input	Addressbus for RX-RAM memory
udp_addr_tra(9:0)	Input	Addressbus for TX-RAM memory
tcp_pkt_rec	Output	New TCP packet received
tcp_pkt_ack	Input	TCP packet received acknowledge
tcp_tra_req	Input	TCP transmit request
tcp_tra_ack	Output	TCP transmit acknowledge
ram_wr_str	Input	TX-RAM write strobe
udp_tra_req	Input	UDP transmit request
udp_tra_acq	Output	UDP transmit acknowledge
udp_length(10:0)	Input	UDP transmit packet length
arp_send_req	Input	Manual ARP req inititaing
arp_send_ack	Output	Manual ARP req ack
target_ip_address_1(7:0)	Input	Target IP address 1
target_ip_address_2(7:0)	Input	Target IP address 2
target_ip_address_3(7:0)	Input	Target IP address 3
target_ip_address_4(7:0)	Input	Target IP address 4
core_ip_address_1(7:0)	Input	UDP_IP core IP address 1
core_ip_address_2(7:0)	Input	UDP_IP core IP address 2
core_ip_address_3(7:0)	Input	UDP_IP core IP address 3

core_ip_address_4(7:0)	Input	UDP_IP core IP address 4
core_mac_address_0(7:0)	Input	UDP_IP core MAC address 0
core_mac_address_1(7:0)	Input	UDP_IP core MAC address 1
core_mac_address_2(7:0)	Input	UDP_IP core MAC address 2
core_mac_address_3(7:0)	Input	UDP_IP core MAC address 3
core_mac_address_4(7:0)	Input	UDP_IP core MAC address 4
core_mac_address_5(7:0)	Input	UDP_IP core MAC address 5
subnet_mask_1(7:0)	Input	Subnet mask part 1
subnet_mask_2(7:0)	Input	Subnet mask part 2
subnet_mask_3(7:0)	Input	Subnet mask part 3
subnet_mask_4(7:0)	Input	Subnet mask part 4
default_gateway_1(7:0)	Input	Default gateway part 1
default_gateway_2(7:0)	Input	Default gateway part 2
default_gateway_3(7:0)	Input	Default gateway part 3
default_gateway_4(7:0)	Input	Default gateway part 4
core_port_1(7:0)	Input	UDP_IP core port address part 1
core_port_2(7:0)	Input	UDP_IP core port address part 2
target_port_1(7.0)	Input	UPD message target port part 1
target_port_2(7.0)	Input	UPD message target port part 2
rx_error	Output	Faulty CRC on recived frame
overflow	Output	Missed packet, buffer overflow
incorrect_udp_length	Output	Set max UDP packet length error flag (option)
incorrect_udp_chksum	Output	Incorrect UDP checksum error flag (option)
arp_req_timeout	Output	ARP response timeout error flag
arp_req_resp_ack	Output	ARP response received
udp_max_lenght(10:0)	Input	Specify max udp rx length
multicast_en	Input	Enable multicast port
group_adress(31:0)	Input	Group address port
enable_1gb	Input	Toggles between 10/100 and 1000Mbit transfer speed
pause_timer_done	Output	Pause frame status. Low when paused.
pause_time(15:0)	Output	Indicates the remaining pause time (pause quanta) set by a MAC PAUSE Frame.

Table 2: UDP/IP I/O Signals

Core Modifications

Echo request/reply functionality can be excluded. Other types of packets can be supported, for example RARP. An RGMII bridge can be added. The number of ARP entries can be extended upon request. The core can be customised to work with other type of pld devices.

Core Assumptions

The UDP/IP core is following the Ethernet and IP standard but has the following simplifications:

- The core doesn't support jumbo frames
- The core does not cover the data link-layer, therefore an external transceiver is needed.
- Min packet length is 64 bytes, therefore shorter packets will be padded with extra bytes at the end.
- ICMP echo-reply messages will not be checked for correct ICMP checksum.
- The core will not fully support the Internet Protocol (IP). Fragmentation etc. is not supported. Most of the fields in the IP header are set to a default values, except for source/destination addresses and total length.
- The core does not support configuration through MDIO or configuration vectors, this is expected to be implemented outside the UDP/IP core.
- On ICMP echo-replies, IP header identifier field will not be increased.
- ICMP echo-reply messages will always contain 32 byte data. The data payload will start on hex 61 and increase to hex 77, then it start over again at hex 61 and goes up to

hex 69. These numbers represent ascii values starting with a, b, c…

Supported Devices

- Virtex-5
- Virtex-4
- Virtex-II Pro
- Virtex-II
- Virtex-E
- Spartan-3E
- Spartan-3

Verification Methods

The UDP/IP core's functionality has been extensively tested with a testbench and a large number of test patterns. It has also been implemented on a Xilinx Virtex-2 with a Broadcom 1Gbit transceiver, where the communication with several PC:s has been verified.

Design Services

Prevas AB also offers core integration, core customisation and other design services.

Ordering Information

This product is available from Prevs AB, under terms of the SignOnce IP License. See www.prevas.se for pricing or contact Prevas for additional information about this product.

Prevas AB
Legeringsgatan 18
SE – 721 03 Vasteras
Sweden

Phone: +46 (0)21 – 360 19 00
Fax: +46 (0)21 – 360 19 29
Email: jonas.lindstedt@prevas.se
URL: www.prevas.se

Prevas AB cores are purchased under a Licence Agreement, copies of which are available on request. Prevas AB retains the right to make changes to these specifications at any time, without notice. All trademarks, registered trademarks, or servicemarks are the property of their respective owners.

Related Information

Xilinx Programmable Logic

For information on Xilinx programmable logic or development system software, contact your local Xilinx sales office, or:

Xilinx, Inc. 2100 Logic Drive
San Jose, CA 95124
Phone:+1 408-559-7778
URL: www.xilinx.com