



# UDP/IP 100 Mbit Transmitter/Receiver

- High Speed Communication Core

September, 2005

Product Specification

## RealFast Intellectual Property

Vasteras Technology Park

Kopparbergsvagen 8  
S – 722 13 Vasteras  
Sweden

Email: stefan.sjoholm@realfast.se  
Phone: +46 (0)21 – 470 20 25  
Fax: +46 (0)21 – 470 21 25  
URL: www.realfast.se

## Features

- Send and receive UDP-packets (User Datagram Protocol) over Ethernet
- Send and receive broadcast messages
- Full duplex
- Replies to echo ping-requests
- ARP-table with 4 entries
- Check incoming/outgoing packets against CRC errors
- Checks UDP checksum on received packet
- Send 256 byte wide packets
- Programmable IP/MAC address
- Supports transfer rates at 10/100 Mbit/s
- Fully synchronous implementation, which uses block- RAM for storage
- Compatible with standard Ethernet transceivers and MII (Media Independent Interface)

## 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 highly suitable for sending large amount of data (210 byte) in one packet.

## General Description

The UDP/IP core sends and receives Ethernet packets to a 10/100Mbit 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 v 6.0	
Support	
Support provided by [RealFast AB]	

The core has a unique MAC-address (Media Access Control) and an IP-address (Internet Protocol) 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 (Local Area 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 256 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 (Address Resolution Protocol) will be sent to get the MAC-address.

The core will respond to ARP requests that are targeted to the core's IP-address.

Family	Example Device	Fmax (MHz)	Slices	IOB	GCLK	Design Tools
Spartan-IIIE	XC2S300E-6	60	1065	200	3	ISE 6.3i

Table 1: Example Implementation Statistics

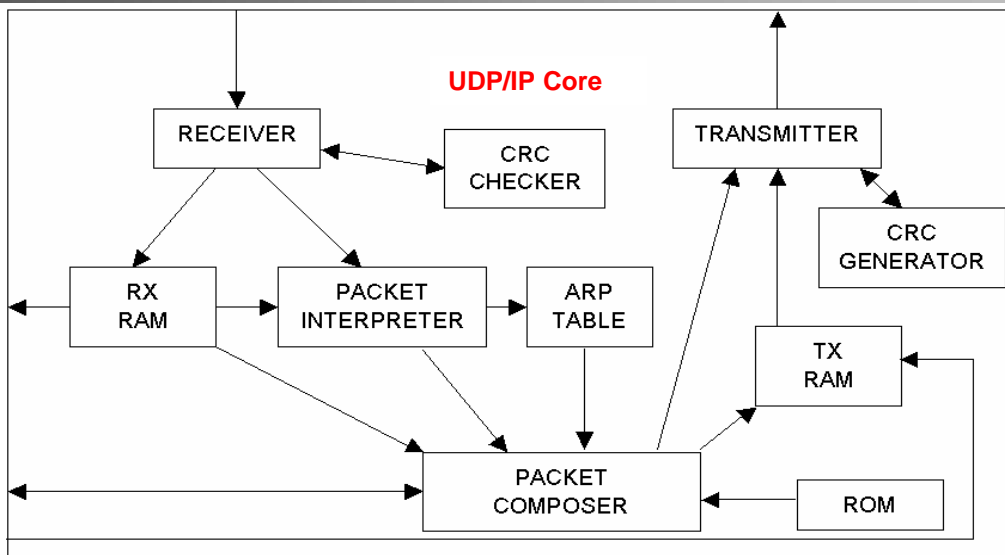


Figure 1: UDP/IP Core Block Diagram

The purpose of this packet is for other Ethernet nodes to get the core MAC-address.

The IP-core will also send ARP-requests by itself either manually or automatically. The core sends two ARP-requests with 1 sec of timeout delay each before an ARP-response timeout occur.

ICMP (Internet Control Message Protocol) 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 .

### Functional Description

As shown above and explained below, the UDP/IP core includes six major blocks: Receiver, Transmitter, CRC checker, CRC generator, packet interpreter and packet composer. 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 256 bytes will be rejected. A message that doesn't match the core or the broadcast MAC address will not be stored.

#### 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-packet it will flag to the application that a new UDP-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

Depening one which packet that is to 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.

#### Transmitter

This block reads data from TX-RAM, puts out the transmit packet to the nibble-wide databus and sets control signals. At the beginning, 16 nibbles 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.

#### CRC Generator

While the packet is sent, CRC is calculated, one byte at a time. It uses the CRC32 polynomial for Ethernet. The polynomial 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 512\*8 bits wide. The RX-RAM stores the incoming message and the TX-RAM holds the transmit message.

### 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 a FIFO. The oldest data in the table will be replaced when a new entry is needed.

### Core Modifications

The size of the UDP-packet can be extended. Echo request/reply functionality can be excluded. Other types of packets can be supported. CRC 32 checking on incoming packets can be omitted

### Core Assumptions

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

- Packets bigger than 256 bytes will not be received.
- The core will not work for 1Gbit Ethernet.
- 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 IP header checksum will not be checked.
- 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...  
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 Spartan 2 with a Broadcom transceiver, where the communication with several PC:s has been verified.

### Design Services

RealFast also offers core integration, core customisation and other design services.

### Ordering Information

This product is available from RealFast Intellectual Property, under terms of the SignOnce IP License. See [www.realfast.se](http://www.realfast.se) for pricing or contact RealFast for additional information about this product.

**RealFast Intellectual Property**  
Kopparbergsvagen 8  
S – 722 13 Vasteras  
Sweden

Email: stefan.sjoholm@realfast.se  
Phone: +46 (0)21 – 470 20 25  
Fax: +46 (0)21 – 470 21 25  
URL: www.realfast.se

RealFast Intellectual Property cores are purchased under a Licence Agreement, copies of which are available on request. RealFast Intellectual Property 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](http://www.xilinx.com)

## UDP/IP I/O Signals

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

Signal	Direction	Description
reset_n	Input	Master Reset (Asynchronous)
phy_reset_n	Output	Transceiver master reset
rxdata(3:0)	Input	Transceiver receiver databus
rx_clk	Input	Receiver master clock from transceiver
rx_dv	Input	Receiver data valid strobe from transceiver
tx_clk	Input	Transmitter master clock from transceiver
txdata(3:0)	Input	Transceiver transmitter databus
tx_en	Output	Transmit data enable to transceiver
tx_er	Output	Transmit error line
rx_crc_mismatch	Output	Activated when a CRC error detected on RX
led	Output	Toggles for every new UDP-packet rec.
udp_pkt_rec	Output	new UDP packet received
udp_pkt_acq	Input	packet received acknowledge
udp_data_rec(7:0)	Output	Databus for RX-RAM
udp_data_tra(7:0)	Input	Databus for TX-RAM
udp_addr_rec(7:0)	Input	Addressbus for RX-RAM memory
udp_addr_tra(7:0)	Input	Addressbus for TX-RAM memory
ram_wr_str	Input	TX-RAM write strobe
udp_tra_req	Input	UDP transmit request
udp_tra_acq	Output	UDP transmit acknowledge
udp_length(7:0)	Input	UDP transmit packet length
arp_send_req	Input	Manual ARP req initiaing
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_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
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	UDP message target port part 1
target_port_2(7:0)	Input	UDP message target port part 2
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

**Table 2: UDP/IP I/O Signals**