Gigabit Ethernet, IPv4, UDP and ARP Protocol Stacks for Spartan3, Virtex-4 and Virtex-5 FPGAs

14 January 2008

Document reference : picotech-ethernet-140108-0

Picotech Ltd

www.picotech-ltd.com
info@picotech-ltd.com
Summary

This note describes the following Picotech firmware:

- 100/1000 Mbps full-duplex Ethernet Media Access Controller (EMAC)
- Ethernet Multiplexer
- Internet Protocol Version 4 (IPv4) protocol stack
- User Datagram Protocol (UDP) stack
- Address Resolution Protocol (ARP) stack

The firmware is implemented entirely in VHDL and has been verified on Xilinx Virtex4 and Spartan3E FPGAs using a range of 10/100/1000 Ethernet PHYs from Broadcom, Marvell and SMSC.

The firmware permits Spartan and Virtex FPGAs to transmit and receive UDP/IP datagrams over 100 and 1000Mbps Ethernet at the full bandwidth of the respective links. The firmware allows a configurable number of Ethernet, IP and UDP endpoints to be multiplexed onto a common Ethernet interface (PHY). The firmware implements the Address Resolution Protocol (ARP). The note also describes the implementation of an Ethernet frame interface to the IBM CoreConnect On-chip Peripheral Bus (OPB), enabling Microblaze (Spartan, Virtex4 LX, FX, Virtex5 LXT, SXT, FPGAs) and PowerPC (Virtex4 FX, Virtex5 FXT) FPGA microprocessors to communicate via TCP/IP simultaneously with the high bandwidth UDP/IP firmware.

Overview

Table 1 overleaf shows current Xilinx IP for Ethernet connectivity. Clearly there are several deficiencies:

- There is no standalone Ethernet MAC (one not requiring a PowerPC or MicroBlaze) for 10/100Mbps operation with an MII interface applicable to Spartan devices and the LX and SX members of the Virtex4 and 5 families (those without hard Ethernet MAC resources).
- There are no cores providing higher level networking protocol stacks, such as the Internet Protocol (IPv4), User Datagram Protocol (UDP) or Address Resolution Protocol (ARP).
- There are no cores providing packet switching for Ethernet frames – i.e. for Ethernet switches at layer 2, the Data Link layer, of the OSI model.

Typically higher level protocols such as the Internet Protocol (IPv4), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are implemented in software on the PowerPC or MicroBlaze 32bit microprocessors. To this end, Xilinx provide the Xilinx micro-kernel (xil_kernel) with the Light-Weight IP (lwip) TCP/IP stack. Alternatively Green Hill's Integrity, VxWorks, or one of the many flavours of Linux, such as uCLinux, Linux 2.4, 2.6, MontaVista etc. may be employed with the Linux TCP/IP stack to provide high level networking protocols.

For high bandwidth streaming applications, there are several disadvantages to this approach:

- Smaller members of the Spartan, Virtex4 and Virtex5 families do not have sufficient logic resource available to host a microprocessor, bus and Ethernet MAC peripheral with DMA capability suitable for high bandwidth streaming. The light-weight Ethernet MAC peripherals, such as xps_ethernetlite, which do fit in the smaller devices do not have DMA capability and in practice give sustainable UDP/IP or TCP/IP throughputs of only a few hundred kilobytes per second.
- Even for the larger members of the respective FPGA families with scatter/gather DMA capable Ethernet MAC peripherals, such as plb_ethernet and plb_gemac, the microprocessor forms a bottleneck for streaming. This is due in part to pre-pending protocol headers and calculating IP and UDP checksums, but also because the majority of operating systems require a double copy
of network packets: firstly between the peripheral and the kernel, and secondly between the kernel and user space.

<table>
<thead>
<tr>
<th>Core</th>
<th>Speed</th>
<th>Interface</th>
<th>Bus</th>
<th>DMA</th>
<th>MII Management</th>
<th>Approximate Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>opb_ethernetlite</td>
<td>10/100</td>
<td>MII</td>
<td>OPB</td>
<td>no</td>
<td>no</td>
<td>Supplied with EDK</td>
</tr>
<tr>
<td>xps_ethernetlite</td>
<td>10/100</td>
<td>MII</td>
<td>PLB</td>
<td>no</td>
<td>no</td>
<td>Supplied with EDK</td>
</tr>
<tr>
<td>opb_ethernet</td>
<td>10/100</td>
<td>MII</td>
<td>OPB</td>
<td>yes</td>
<td>yes</td>
<td>$5K</td>
</tr>
<tr>
<td>plb_ethernet</td>
<td>10/100</td>
<td>MII</td>
<td>PLB</td>
<td>yes</td>
<td>yes</td>
<td>$5K</td>
</tr>
<tr>
<td>plb_gemac</td>
<td>1000</td>
<td>GMII</td>
<td>PLB</td>
<td>yes</td>
<td>yes</td>
<td>$14K</td>
</tr>
<tr>
<td>xps_ll_temac</td>
<td>10/100/1000</td>
<td>MII/GMII/RGMII</td>
<td>PLB/Client i/f</td>
<td>yes</td>
<td>Requires hard EMAC</td>
<td></td>
</tr>
<tr>
<td>plb_temac</td>
<td>10/100/1000</td>
<td>MII/GMII/SGMII</td>
<td>PLB</td>
<td>yes</td>
<td>yes</td>
<td>Requires hard EMAC</td>
</tr>
<tr>
<td>gemac</td>
<td>1000</td>
<td>GMII/RGMII/SGMII</td>
<td>Client i/f</td>
<td>-</td>
<td>$14K</td>
<td></td>
</tr>
<tr>
<td>temac</td>
<td>10/100/1000</td>
<td>MII/GMII/RGMII</td>
<td>Client i/f</td>
<td>-</td>
<td>$18K</td>
<td></td>
</tr>
<tr>
<td>xgmac</td>
<td>10000</td>
<td>XGMII</td>
<td>Client i/f</td>
<td>-</td>
<td>$7K</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Xilinx Ethernet IP

To fill this gap Picotech has developed firmware in VHDL for a 100/1000 Mbps Ethernet MAC with MII and GMII interfaces, and parametrized Ethernet MUX, Internet Protocol (IPv4), Address Resolution Protocol (ARP) and User Datagram Protocol (UDP) cores.

**Typical Use Cases**

*High Bandwidth Low Latency Streaming over UDP/IP with Concurrent Low Bandwidth High Reliability TCP/IP for Control/Status Messaging*

Figure 1 overleaf shows a typical use case for the Picotech firmware described in this note. In this example, the firmware provides three MAC/IP endpoints, two with full-duplex four port UDP/IP endpoints, and one with a single TCP/IP endpoint.

High bandwidth, low latency data is streamed over the UDP/IP ports. This part is implemented wholly in the FPGA fabric. With no processor bottleneck the full Ethernet bandwidth (Gigabit or 100Mbps) may be used. Address resolution is provided by the ARP entity.

Picotech UDP links are well suited for transporting high bandwidth data, such as video or sonar acoustic data, high bandwidth non-acoustic data such as INS data, or implementing 'virtual COM ports' – transporting legacy RS232/422 data over Ethernet transparently.

Low bandwidth, high reliability control and status messages - for example power supply control, status monitoring - are sent via the TCP/IP endpoint, using the PowerPC or MicroBlaze microprocessor's TCP/IP stack on the Xilinx micro-kernel or Linux 2.4 or 2.6 kernels. The PowerPC or MicroBlaze microprocessors may run high level application software implemented in C, C++ or Java to implement Web Servers, Telnet clients etc.
High Bandwidth Real-Time Data Acquisition using UDP/IP on Low End FPGA

Figure 2 below illustrates another use case for the firmware on a small Virtex or Spartan device. In this scenario, a system is required to stream high bandwidth, low latency data in real-time using a connectionless best-effort data path. TCP/IP is not required. In this instance the firmware is configured for sixteen UDP/IP bidirectional links at one IP Addresses. Address resolution is again provided by the ARP entity.
In common with the Xilinx Ethernet IP, the Picotech cores do not adopt a store and forward approach, but rather use a buffer-less design. Consequently the Picotech cores only require FPGA logic resources: no block RAM or buffering is used within the protocol stacks. This enables even the smaller Spartan devices to run a low latency, maximum throughput, real-time UDP/IP stack with ARP, wholly implemented within the FPGA fabric.

By adopting a common interface, the Ethernet and Protocol Stack firmware may be connected in many ways: for example in systems not requiring an Ethernet MUX, the IPv4 entity may be connected directly to an Ethernet MAC. Furthermore because the client side interface of the Picotech Ethernet MAC matches that of the Xilinx GEMAC and hard EMAC wrapper, this enables drop-in replacement of the MAC on those devices with a hard EMAC (Virtex4 FX, Virtex5 LXT, SXT, FXT).

Because the number of Ethernet and UDP ports is parametrized by VHDL generics, and as many IPv4 and UDP entities may be instantiated as there are Ethernet ports, so any number of Ethernet/IP/UDP endpoints may be created. Because of this, it is not possible to give a single figure for resource utilisation, however Table 2 below gives some examples.
<table>
<thead>
<tr>
<th>Core</th>
<th>BRAM</th>
<th>Mult18x18/DSP48</th>
<th>Slices</th>
<th>Performance $F_{\text{max}}$ (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100/1000 Mbps EMAC</td>
<td>0</td>
<td>0</td>
<td>258</td>
<td>205</td>
</tr>
<tr>
<td>Ethernet MUX (4 port)</td>
<td>0</td>
<td>0</td>
<td>243</td>
<td>188</td>
</tr>
<tr>
<td>IPv4</td>
<td>0</td>
<td>0</td>
<td>571</td>
<td>151</td>
</tr>
<tr>
<td>UDP (8 Port)</td>
<td>0</td>
<td>0</td>
<td>785</td>
<td>196</td>
</tr>
<tr>
<td>ARP</td>
<td>0</td>
<td>0</td>
<td>416</td>
<td>230</td>
</tr>
</tbody>
</table>

*Table 2: Picotech Ethernet and Protocol Stack core Resource Utilisation and Performance.*

The payload data rate achieved in practice using the Picotech firmware depends upon the protocol chosen, and the ratio of payload size to total packet size. For example if 1024 bytes of payload data were transported using UDP/IP over Ethernet this is how it would break down:

- 1024 bytes - payload
- 20 bytes - IPv4 header
- 8 bytes - UDP header
- 26 bytes - Ethernet header and trailer (preamble, MAC src/dst, len/type, fcs)
- 12 bytes - Ethernet inter-frame gap.

In this example 1090 bytes are required to transport the 1024 bytes of payload. Add to this the overhead of 8B/10B encoding and the payload data rate is approximately 75% of the Ethernet bandwidth in this example, i.e. 8.96MiB/s for 100Mbps Ethernet, or 89.6MiB/s for Gigabit Ethernet. Because the UDP/IP protocol stack is implemented entirely in firmware, this maximum theoretical rate is achievable – it isn’t limited by a software protocol stack running on a microprocessor.

For higher rates, UDP/IP datagrams may be transmitted with larger payloads, alternatively the payload may be transported in IP datagrams (saving the 8 bytes of UDP header) or raw Ethernet frames (saving the UDP and IP headers). The Ethernet inter-frame gap may also be reduced below the recommended 96 bits.

**100/1000 Mbps Full Duplex Ethernet MAC**

The Picotech Ethernet MAC conforms to the MII (100Mbps) and GMII (1000Mbps) interfaces specified by the IEEE 802.3-2005 standard - Sections 2 and 3 Reconciliation Sub layer (RS) and Media Independent Interface (MII), Gigabit Media Independent Interface (GMII). The Ethernet MAC also implements the MII Management Interface, enabling the PHYs basic and extended control and status registers to be read and written. This allows the PHYs speed, duplex and auto-negotiation parameters to be configured.

On the receive path the Picotech Ethernet MAC handles the Ethernet preamble, start of frame delimiter, source and destination addresses, length/type and data fields and calculates, checks and strips off the frame check sequence (32bit CRC). On the transmit path the Ethernet MAC prepends the Ethernet preamble, start of frame delimiter and calculates and appends the frame check sequence, in addition to padding frames smaller than the minimum Ethernet frame length.

The client side interface of the Picotech Ethernet MAC matches that of the Xilinx GEMAC and hard EMAC wrapper. This enables drop-in replacement of the MAC on those devices with a hard EMAC (Virtex4 FX, Virtex5 LXT, SXT, FXT).

---

1 MiB, mebi-byte, $2^{20}$ bytes.
Ethernet MUX

The Picotech Ethernet MUX is a layer 2 - Data Link layer - Ethernet frame multiplexer/demultiplexer. The Ethernet MUX is parametrized by VHDL generics enabling it to be configured with any number of ports. The MUX contains a MAC address table whose entries may be programmed dynamically via a host port.

Ethernet frames arriving from the Ethernet MAC interface on the left hand side of Figure 1 are routed to client ports if their destination addresses match those stored in the MAC address table for the client port. Frames arriving with destination addresses not matching any of those stored in the MAC address table are discarded.

Ethernet frames arriving from client ports on the right hand side of the diagram are multiplexed onto the Ethernet MAC. Flow control is achieved using the transmit data valid/acknowledge signals: clients ready to send Ethernet frames place the first byte of the frame on the client transmit tx_data bus and assert tx_data_valid. When ready to send, the Ethernet MUX strops tx_ack, whereupon the client transmits the remainder of the Ethernet frame whilst continuing to assert data valid. This is the same flow control mechanism adopted by the Picotech EMAC and Xilinx Gemac and Temac EMAC client interfaces.

The Ethernet MUX may be configured for round-robin or priority driven switching of client transmit Ethernet frames.

The Ethernet MUX's MAC address table may be dynamically programmed via a host interface. One of the entries for the client ports may of course be configured with the Ethernet broadcast address, in which case Ethernet broadcast frames will be directed to that port. This is used for example with the Picotech Address Resolution Protocol (ARP) stack to handle ARP requests and replies.

Internet Protocol Version 4 (IPv4)

The Picotech Internet Protocol entity implements an IPv4 protocol stack, capable of transmitting and receiving IPv4 packets (RFC 791) encapsulated in Ethernet frames.

The entity is configured (optionally by ARP) with destination and source MAC and IP addresses. Ethernet frames received by the IPv4 entity, from an Ethernet MAC or switch, are parsed. If an Ethernet frame contains a valid IP packet with the correct source and destination addresses and IP header, the IP payload of that packet is forwarded to the client interface. The entity checks that the IP packet encapsulated within the Ethernet frame(s) has the correct MAC and IP source and destination addresses, length/type field, identification, protocol, the correct total length and a valid IPv4 header checksum matching that calculated across the received IP header.

IP payloads transmitted to the entity on the client interface are encapsulated within IP packets within Ethernet frames. The IPv4 entity prepends the Ethernet frame and IP packet headers containing the MAC and IP addresses programmed into the entity, calculates and inserts the IP header checksum, identification, protocol, time-to-live, flags, fragment offset, type-of-service and version fields.

Clients to the IPv4 entity need no knowledge of the structure of Ethernet frames or IPv4 packets. On the receive side the client is provided with a simple byte-wide data bus and data valid signal. Only the payload of received IP packets is delivered to the client. On the transmit side the client merely has to provide a byte-wide data bus, data valid signal and throttle sends using the acknowledge signal. The client only has to provide the payload of IP packets to the IPv4 entity.

User Datagram Protocol (UDP)
The Picotech User Datagram Protocol (UDP) entity implements the UDP protocol (RFC 768) and provides multiplexing and demultiplexing of a configurable number of UDP ports. The UDP entity is parametrized by VHDL generics enabling it to be configured with any number of ports. The UDP port number table may be dynamically programmed via a host interface.

For each client port the entity registers a UDP source and destination port number pair. UDP datagrams arriving from the IPv4 interface on the left hand side of Figure 1 are routed to client ports if their source and destination port number pairs match those stored in the UDP port number table for the client port. Datagrams arriving with source and destination port number pairs not matching any of those stored in the port number table are discarded. The UDP entity checks that the length of the UDP datagram received matches that stored in the UDP header.

UDP datagram payloads arriving from client ports on the right hand side of Figure 1 are multiplexed onto the IPv4 bus. The UDP entity may be configured for round-robin or priority driven switching of client UDP payloads.

Clients to the UDP entity need no knowledge of the structure of UDP datagrams. On the receive side the client is provided with a simple byte-wide data bus and data valid signal. Only the payload of received UDP datagrams is delivered to the client. On the transmit side the client merely has to provide a byte-wide data bus, data valid signal and throttle sends using the acknowledge signal. The client only has to provide the payload of UDP datagrams to the UDP entity.

**Address Resolution Protocol (ARP)**

The Picotech Address Resolution Protocol (ARP) entity implements the ARP protocol (RFC 826). The ARP entity is parametrized by VHDL generics enabling it to control any number of IPv4 endpoints. Typically the ARP entity is connected to a port on the Picotech Ethernet MUX configured for the broadcast Ethernet address of ff-ff-ff-ff-ff, and an array of buses connect ARP to the host control ports of IPv4 entities. Figure 3 below shows ARP configured for just two IPv4 endpoints.

The ARP entity receives and parses ARP requests in broadcast Ethernet frames. If an ARP request contains a Target Protocol Address (TPA) matching one of those IPv4 entities to which it is connected, the ARP entity responds with an ARP reply containing the corresponding Source Hardware Address (SHA) for that IPv4 entity. Furthermore the ARP entity programs the IPv4 entity addressed with the received Source Hardware Address (SHA) of the remote end point.
**IBM CoreConnect OPB Ethernet Peripheral: opb_picotech_ethernet**

The opb_picotech_ethernet peripheral conforms to the IBM CoreConnect On-chip Peripheral Bus (OPB) specification and allows MicroBlaze or PowerPC processors to send and receive Ethernet data. The peripheral may be connected directly to the Picotech Ethernet Media Access controller or via the Picotech Ethernet MUX allowing microprocessor high level networking traffic to share a single Ethernet PHY with high bandwidth traffic from the Picotech UDP/IP firmware.

Similarity to the Xilinx Ethernet IP on the OPB side allows the Picotech opb_picotech_ethernet peripheral to share Xilinx and Linux kernel device drivers with this IP. The Picotech IP core differs however on the Ethernet side by deferring Ethernet preamble and frame-check sequence calculation to the Picotech Ethernet MAC, and its conformance to the flow control mechanism adopted enables the peripheral to participate in the Picotech Ethernet framework.