FPGA COMMUNICATIONS BASED ON GIGABIT ETHERNET

L.R. Doolittle, C. Serrano, LBNL, Berkeley, CA 94720, USA

Abstract

The use of Field Programmable Gate Arrays (FPGAs) in accelerators is widespread due to their flexibility, performance, and affordability. Whether they are used for fast feedback systems, data acquisition, fast communications using custom protocols, or any other application, there is a need for the end-user and the global control software to access FPGA features using a commodity computer. The choice of communication standards that can be used to interface to a FPGA board is wide, however there is one that stands out for its maturity, basis in standards, performance, and hardware support: Gigabit Ethernet. In the context of accelerators it is desirable to have highly reliable, portable, and flexible solutions. We have therefore developed a chip- and board-independent FPGA design which implements the Gigabit Ethernet (GbE) standard. Our design has been configured for use with multiple projects, supports full line-rate traffic, and communicates with any other device implementing the same well-established protocol, easily supported by any modern workstation or controls computer.

INTRODUCTION

The FPGA communication system based in GbE is the result of two success stories. First FPGAs, whose capabilities have improved dramatically in the last few years in terms of size, performance, and flexibility. This tremendous improvement has enabled the range of applications of FPGA solutions to explode, increasing the volume of production, and therefore reducing the price of every new generation of FPGA family appearing on the market. The second success story corresponds to Ethernet (and GbE as the highest performance flavor to see adoption in today's commodity hardware). The key to the success of Ethernet resides in its simplicity and its associated side effects [1]. Implementation is simple in hardware and provides larger data rates than most computers can process these days. The evolution of the Ethernet protocol since its creation has accompanied that of computers since its creation in 1970s, and has been the Local Area Network (LAN) protocol by excellence ever since (with the market of Ethernet switches alone amounting to $16 billion in 2010).

In accelerators, FPGAs have introduced a great deal of flexibility and performance into machine fast controls and operations. They are extensively used to implement highly specialized tasks where simplicity, low production cost, and reliability are big assets. Even if simple and reliable, these systems need means for communications with commodity computers for configuration, data acquisition and diagnostics via Global Controls such as EPICS and TANGO. The nature of highly specialized hardware, combined with the simplicity of the GbE protocol make the implementation of GbE in FPGAs our preferred way of communication with the Global Controls.

STANDARD SELECTION

Once computers are taken away from highly specialized hardware in accelerators, a number of hardware standards other than GbE are available to enable communication between FPGAs and Global Controls, such as: PCI, USB, CAN, VME, VXI, cPCI, PCIe, IEEE-488 (HP-IB), IEEE-1392 (Firewire), SATA (or eSATA), etc. Longevity is a usual requirement for hardware installed in accelerators, and the chosen communication standard should last at least as long as the actual hardware. While CAN, VME, IEEE-488 and cPCI are indeed standards, they do not achieve as much performance/simplicity ratio as that provided by GbE. PCI hardware will undoubtedly be present for a long time, however active development seems to have shifted from the original parallel standards to the high-speed serial PCIe version.

USB can be amazingly attractive for small projects. Throughput is high, as is the availability for hardware components for both sides of the link. On the other hand, both software and hardware models are very much desktop-based. Cable distance is limited, and the standard is young and unstable enough that software support can be quirky. The LBNL experience running USB-based instrumentation affirms that it should be avoided for production accelerator installations. Ethernet on the other hand accounts for maturity, stability, high availability of hardware, high performance and ease of integration with highly specialized hardware in accelerators.

OVERVIEW

In addition to GbE, our implementation includes IP, UDP and ARP at full line speed. UDP seems a better match to real-time communications than TCP since point-to-point dedicated networks are normally available in accelerator deployments, and is much better suited to the finite resources of an FPGA. The additional complexity added by supporting UDP over raw Ethernet is largely rewarded by the extensive UDP support on the host side (BSD sockets API, and its descendants). Raw Ethernet support is not as usual, and programs manipulating raw Ethernet packets typically require elevated levels of privileges, removing one layer of security.

Fig. 1 shows a block diagram of an implementation example using the FPGA Ethernet module. The core func-

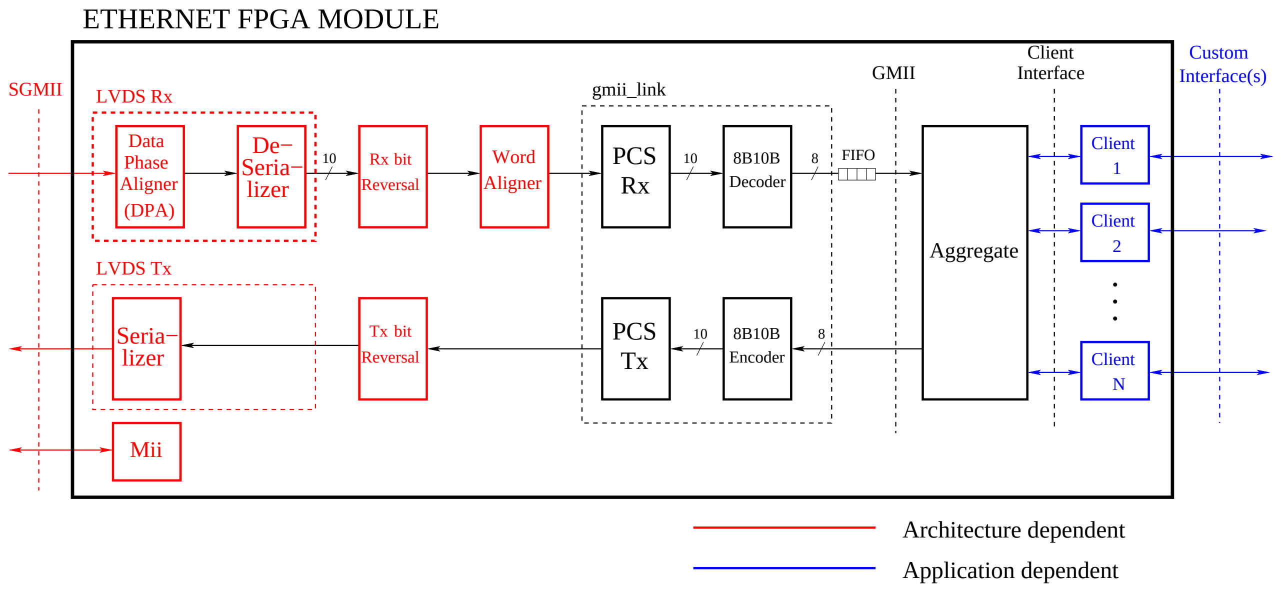

Figure 1: Top level block diagram. Example where the architecture dependent modules correspond to the implementation on an Altera Stratix-IV EP4SGX230KF40C2ES FPGA.

tionality is implemented in the Aggregate module, which has been custom-designed using Verilog HDL. A clean separation between this module and rest of the design allows for flexibility and portability, benefiting from a well defined medium-independent interface in the GbE side (GMII in the example shown in Fig. 1, which is defined as part of the GbE standard), and another well defined client interface for data exchange with the FPGA fabric. Details on the core design and interfaces are given next.

CORE DESIGN

The core design described here includes the architecture/implementation independent modules, shown in black in Fig. 1. This logic is independent of the targeted FPGA (as well as any peripherals), the particular application of the communication, and implements layers 2, 3, and 4 of the OSI model.

The FPGA Ethernet module has compile-time configurable IP and MAC addresses, and the current implementation assumes that the host side uses a single UDP port number for communication. Aggregate implements the upper level logic to (de)encapsulate UDP packets. It extracts the UDP packet length in the length field and uses it to generate the necessary control signals needed to interact with the client modules in the FPGA fabric. The host has the possibility of communicating with any of the n clients (in blue in Fig. 1), where each client is assigned a UDP port number. More details on the clients are given later.

The PCS Rx/Tx modules implement the 802.1x PCS standard, building a block which interfaces the Ethernet framer to Tx PMA (Physical Medium Attachment). It performs preamble generation, inserting idle patterns, 802.1x link negotiation and all the low-level signaling (excluding 8B10B coding, which is implemented in the 8B10B Encoder/Decoder). The 32-bit CRC of the Ethernet frame, and the length of the IP and UDP packets are checked for correctness. If any errors are detected, the whole Ethernet frame is dropped and the clients stay idle. Details on the clients are given next.

INTERFACES

Client Interface (FPGA side)

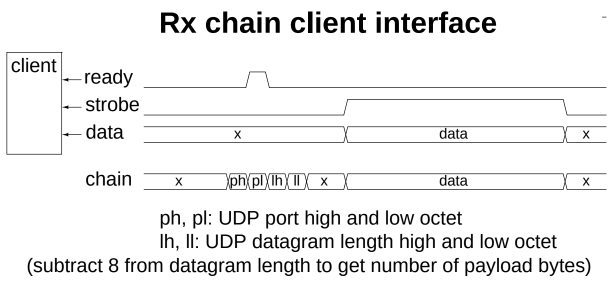
Rx chain client interface
client
ready
strobe
data
x
data
x
chain
x
ph
pl
lh
ll
x
data
x
ph, pl: UDP port high and low octet
lh, ll: UDP datagram length high and low octet
(subtract 8 from datagram length to get number of payload bytes)

Figure 2: Rx chain in the client interface shown in Fig. 1.

The Aggregate module provides the data contained in the UDP packet payload to the client modules (application dependent modules shown in blue in Fig. 1), where the distinction of each client is done using the UDP port number. These clients exist to provide the flexibility of implementing different custom protocols adapted to the application needs. This level of communication relies on data transaction from levels 1 through 4 in the OSI model, and custom interfaces to interact with the rest of the fabric in the FPGA. The different client instantiations share a common interface with the Aggregate module (labeled “client interface” in Fig. 1), and each implement a custom interface with the rest of the fabric in the FPGA (labeled "Custom Interface(s)" in Fig. 1).

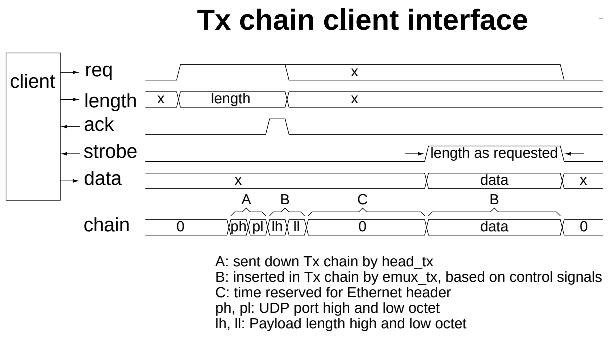
Tx chain client interface
client
req
length
ack
strobe
data
chain
x
length
length as requested
data
A
B
C
0
ph
pl
lh
ll
A: sent down Tx chain by head_tx
B: inserted in Tx chain by emux_tx, based on control signals
C: time reserved for Ethernet header
ph, pl: UDP port high and low octet
lh, ll: Payload length high and low octet

Figure 3: Tx chain in the client interface shown in Fig. 1.

Figs. 2 and 3 show the Rx and Tx flow chain for the client interface, respectively (seen from the clients' perspective). The Rx chain follows a simple strobe and data transmission, where the strobe is high if the data is valid, and the UDP packet it was sent into contained the UDP port associated with the client. A ready signal is provided a fixed number of clock cycles before the strobe, which can be used by the client module to anticipate when it will receive valid data.

The Tx chain follows a similar transmission scheme (strobe and data), preceded by a transmission request for a scheduler to assign when each client transmits the data onto the Ethernet link. When the client is ready to send data, it sets the request signal along with the length of the data to be transmitted. Once the scheduler has established the turn of the client to send the data, it sends an acknowledge to the client, that will then receive a strobe for as many clock cycles as previously specified in the length field.

The design is modular, and each UDP clients is handled by a different Verilog module. In addition to that, the project package includes means to automatically generate Verilog code for the Aggregate module, including automatic replication of client handling logic, and the proper module prototype to connect to as many clients as needed.

On-board Local Bus to UDP Gateway

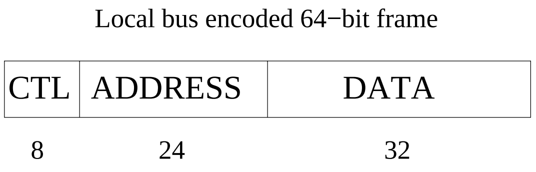

Figure 4: Local bus 64-bit frame.

In the general case, the Ethernet module provides a link between UDP/IP, and a series of customized interfaces, There are as many clients as protocols are encoded in the UDP packet payloads, and each client is assigned a UDP port number to be accessed from the host side. One example of client is what we call the Local Bus to UDP Gateway, where the access to FPGA registers is provided by a very simplified on-board data, address, strobe type local bus.

The master of the transaction is the software side, which can send series of register read and write commands to the FPGA by concatenating the 64-bit frame pattern (shown in Fig. 4) into the UDP payload. One of the 8 control bits is used to indicate the read/write command. If a read command is found, the FPGA ignores the data field and responds using the same 64-bit pattern, where the control and address field remain unchanged, and the data field is filled with the corresponding register value. On the other hand, if a write command is found, the data field is written onto the corresponding register, and the same frame is sent back to the host (which can use it as an acknowledgement). This scheme described above has an intrinsic bandwidth limit for the FPGA, since it only sends data upon request.

Host Interface (GbE link side)

The Ethernet protocol has increasingly added support for different data rates and physical media, and different FPGAs implement the physical layer (including serializer and deserializer mechanisms) differently. It was therefore indicated to provide a medium-independent interface to the core of the design in order to preserve generality and portability. A GMII interface (defined as part of the Ethernet standard) is provided to connect the FPGA core logic and the architecture specific modules (in red in Fig. 1).

The package described here also includes means to attach a live simulation of the synthesizable Verilog to a Linux tun/tap interface, where the GbE link can be abstracted and third party software can be used to exchange UDP packages with the FPGA logic, which has been extremely helpful in the debug process and can be used for early development of software without the need of actual hardware.

CONCLUSIONS

The Ethernet module has been used in various projects under different inter-lab collaborations. The package contains (not counting various instantiation of hardware primitives) purely synthesizable Verilog. There is no need for off-chip memory and synthesizes using 930 logic cells, and 3 block RAMs on a Xilinx Spartan-6 XC6SLX45T (which accounts for 1.7% and 2.6% of the available resources respectively). The physical layer has been demonstrated functional using GMII, Xilinx Virtex-5 MGT, and Altera Stratix-IV LVDS. The project is continuously evolving as new needs arise, and physical layer support plans or currently ongoing testing include RGMII (double-date-rate GMII), Spartan-6, and Virtex-6.

REFERENCES

[1] Rich Seifert, "Gigabit Ethernet: Technology and Applications for High Speed LANs", Addison-Wesley, 1998