

Using FPGAs to Generate Gigabit Ethernet Data Transfers and Studies of the Network Performance of DAQ Protocols

Dave Bailey, Richard Hughes-Jones, Marc Kelly
The University of Manchester, UK

Abstract— FPGA devices have become common in data acquisition (DAQ) systems for High Energy Particle Physics Experiments. The next generation of DAQ systems will use FPGAs with commercial networking components. The use of FPGAs to generate gigabit-Ethernet data streams has been investigated using a Virtex 4 development system to generate raw Ethernet packets over both fibre and copper links. Details of the firmware developed to drive the links are presented. Throughput and packet loss over standard Ethernet networks to PCs have been measured using different request-response protocols. Sequential request and group request DAQ data collection protocols have been implemented and initial scaling tests are reported.

Index Terms—, e-VLBI, CBR, TCP/IP, UDP/IP Gigabit Ethernet.

I. INTRODUCTION

Experiments at the International Linear Collider (ILC) [1] require a new generation of data acquisition systems that are cost-effective and maximize the use of commercially available components. Using standard networking protocols and hardware will ensure compatibility between different components of the detectors, whilst allowing for seamless incremental upgrades to individual systems. The R&D for the ILC experiments is expected to continue until the end of the decade.

Experiments at the ILC will take data without the use of a hardware trigger. Consequently, to minimize dead time, all experimental data must be read from the detector in the 200 ms gaps between beam bunch trains. This semi-continuous data stream must be routed to DAQ computers, currently assumed to be PCs, processed and then sent to offline storage. The aim is to study the options for a general solution, but for discussion we have taken the CALICE [2] ECAL detector as an example. Figure 1 shows a typical DAQ system envisaged for the ILC detectors. The front-end electronics are connected to concentrators over custom, high-speed links. These concentrators then feed the data over a network to compute nodes where the data are processed and stored. These concentrators may do some re-formatting or zero suppression. Whilst the use of some application-specific devices is inevitable, it is highly desirable to use commercial networking devices and protocols as much as possible. One possible option

would be the use of 1 and 10 Gigabit Ethernet. In particular, since much of the front-end electronics will be controlled by FPGAs, it is vital to assess the suitability of FPGAs for directly driving network traffic and to optimize the performance of the hardware and protocols used to send the data over the network.

As indicated in Figure 1, without some form of traffic shaping between the concentrators and the destination PCs, there would be the classic bottleneck problem on the egress of the Ethernet switch, with data queuing for transmission to the procession node and the possibility of packet loss.

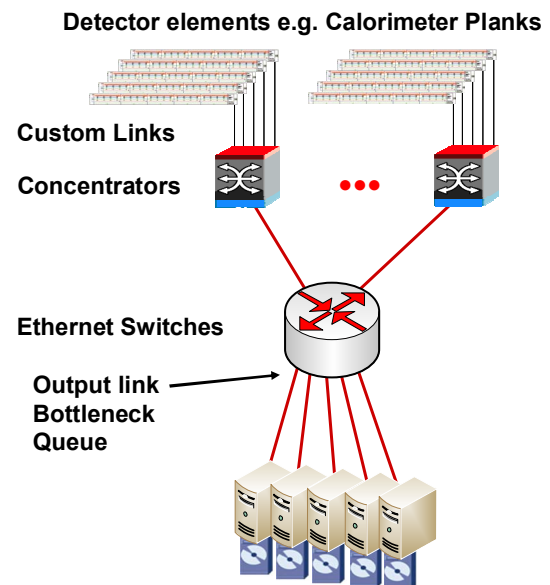


Figure 1. The Architecture of the readout taking the ECAL as an example.

II. THE HARDWARE TEST PLATFORM

A Xilinx [4] Virtex 4 FPGA has been used to directly generate Ethernet network traffic. The system used is a versatile commercial development board produced by PLDA [3]. This comprises a Xilinx Virtex4FX60 series FPGA, mounted on an eight lane PCI Express expansion card. On the board, in addition to the FPGA there are DDR2 memory, two SFP

cages, two HSSDC connectors and a variety of other user configurable IO. This FPGA is a large and complex device; housing two complete PowerPC CPUs, a large amount of internal memory and sixteen high-speed serial communication modules known as RocketIO Multi-Gigabit Transceivers (MGTs).

Another feature of the Virtex4 FPGA is the hardware embedded Media Access Controller (MAC) modules. When combined with the RocketIO these provide an almost complete gigabit Ethernet solution without the need for any external support logic. The only hardware required is a physical network interface, provided in this case by a standard Small Form-factor Pluggable (SFP) module.

In order to provide the high precision reference clock to drive the RocketIO block, the board must be plugged into a PCI Express socket. A standard 100Mhz reference clock is provided by the host motherboard, multiplied on board by 5/4 and cleaned up by an external clock driver to provide the necessary 125Mhz that drives most of the FPGA logic. A secondary 50Mhz clock is obtained from a crystal on the board. This is used to drive some of the control logic.

III. THE FIRMWARE DESIGN

A. Overview

Figure 2 shows a block diagram of the firmware used on the FPGA is divided into four sections:

- Ethernet Interface – consisting of the embedded MAC, RocketIO and simple control logic.
- Packet buffers and arbitration logic used for routing packets for input and output.
- Embedded CPU – a firmware implementation used to provide a simple mechanism to control the MAC and packet state machines.
- Packet state machines. This is the high-speed packet generation logic used for Ethernet testing and the request-response system.

B. The Ethernet Interface

Internally the Ethernet interface is packaged into a re-usable block containing the RocketIO physical layer along with the MAC. This block handles the 8B10B byte encoding and clock regeneration from the incoming data stream together with the various negotiation stages necessary for setting up the link. The protocol used is 1000Base-X, which matches that used by standard fibre Ethernet. When driving fibre SFP modules this is passed straight over the link, however when using copper SFP modules it is translated from the 1000Base-X connection from the FPGA to 1000Base-TX to the remote host. This allows the FPGA design to be independent of the actual network fabric and work over both fibre and copper without changes. All connections are rate locked to 1000Mbit by virtue of the 1000Base-X protocol; however the hardware does

contain the ability to run at 10/100Mbit when using copper SFP's and the SGMII protocol.

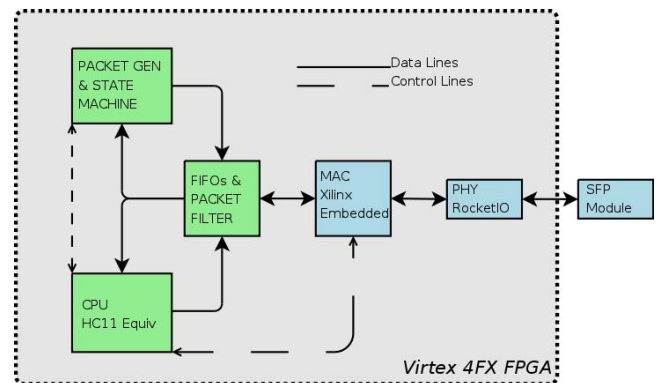


Figure 2. Overview of the design
The blue blocks show the hardware cores and the Green blocks illustrate the user written code.

C. The Packet Buffers.

The interface between the MAC layer and the FPGA fabric is based on packets of data. On reception of a packet, the MAC generates a “Start-of-Packet” signal followed by a stream of 8 bit data at 125Mhz. A “Good” or “Bad” packet signal indicates the end of the stream. A bad packet is one that has been corrupted and has failed the 32bit CRC. Packet transmission is similar; the FPGA presents the MAC with a “Start-of-packet”, data and then finally the “End-of-packet”. The Ethernet CRC is generated by the MAC and so the user logic does not need to add it to the data stream. The nature of the MAC requires that once a packet starts it is processed at full speed with no delays, therefore a buffering mechanism is used to allow processing of the packet to be asynchronous to its movement over the network. Transmit (TX) and receive (RX) buffers are based on example designs provided by Xilinx. These buffers transfer complete packets to and from the MAC. The design combines these buffers with logic that allows the RX stream to be split, dependant on Ethernet packet type, into various channels. One channel is processed by the Embedded CPU, whilst another is processed by the State Machine(s). This filtering is configurable dynamically by the CPU. The TX stream is handled in a similar manner; the output from both CPU and state machine are multiplexed into the MAC in a controlled ratio. This is to allow the CPU to transmit packets out onto the network, but to give the State Machine priority when transmitting. A mix ratio of 1:1 to 1:2¹⁶ is possible.

D. The Embedded CPU

The embedded CPU is a VHDL model [5] based on, and code compatible with, the Freescale 68HC11 device. This is a standard 8bit design with the ability to address up to 64Kbytes of memory. A full GCC development environment exists for this device. Whilst code compatible with the HC11, the design is not an exact copy. The embedded hardware that exists on the

various Freescale models is not present and instead a variety of custom hardware has been placed into the design that interfaces to the MAC, state machine and TX/RX buffers. Parallel and serial interfaces also exist to provide debugging and simple control functions. Further blocks to provide HC11 compatible timers and interrupt functions are implemented as required by the design. The reason for choosing a fully soft core, and not the embedded PowerPCs, was to reserve those CPUs for use in data processing at a later development stage. The design uses the 50Mhz clock divided down to 12.5Mhz. The main function of the CPU is to configure the MAC on boot up, finalize the network auto-configuration and to provide a simple interface to control the main State Machine blocks. Communication is carried out using raw Ethernet packets, to remove the overhead of running an UDP/IP based protocol stack. The application header used is lightweight and allows for easy extension to handle additional functions as they are implemented.

E. The Packet State Machine

The State Machine block shown in Figure 2 is an easily extendable wrapper around the main testing code. In the current design there are two separate functions: a simple packet generator and a request-response system. The relation between the State Machines is shown in Figure 3. The packet

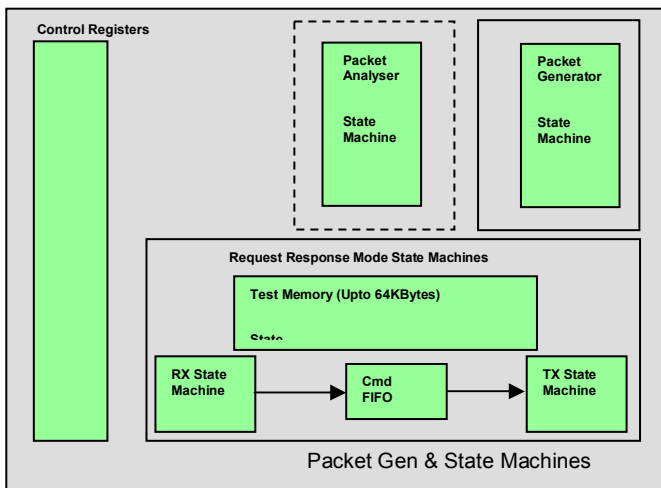


Figure 3. The relationship between the State machines.

analyser state machine is in development.

IV. DETAILS OF THE STATE MACHINES.

A. The Packet Generator State Machine

The packet generator implements registers that determine packet length, count, inter-packet delay and destinations. This allows the system to become a packet source for network testing purposes. Up to sixteen destinations can be programmed, with the system cycling through them on a packet by packet basis. The data can be either static or a simple random pattern derived from a seed value and a linear

feedback shift register. Packet size and count are included to allow remote systems to monitor lost packets. Up to 2^{32} packets can be generated, with a configurable inter-packet delay that is granular to 8ns. Currently the packet size is limited to a payload of 1500 bytes, the Ethernet standard, and what is considered valid data by the receiving stations. The logic is capable of generating 2^{16} byte long packets should that be required.

B. The Request-Response State Machine

The request response system is more complex and is designed to mimic a more realistic experimental setting where remote machines are requesting event data from the DAQ system. All requests contain a command e.g. read memory, or write memory and will receive a reply which may contain data or a suitable return code as appropriate. Internally implemented are 64Kbytes of RAM, an RX state machine, a TX state machine and a FIFO linking them, into which requests can be queued. The RX state machine deals with two issues; firstly the data packets that are used to pre-load the memory with data patterns, the write memory command, and secondly processing any other incoming request and queuing them in the FIFO. A schematic of the states and transition of the RX machine are shown in Figure 4. Each request has a checksum that is compared with the data to give some error protection. Requests consist of an offset in the memory and a data length. For memory read requests, this length need not be limited by Ethernet frame lengths as the system will automatically fragment the data to be returned into the correct maximum frame sizes.

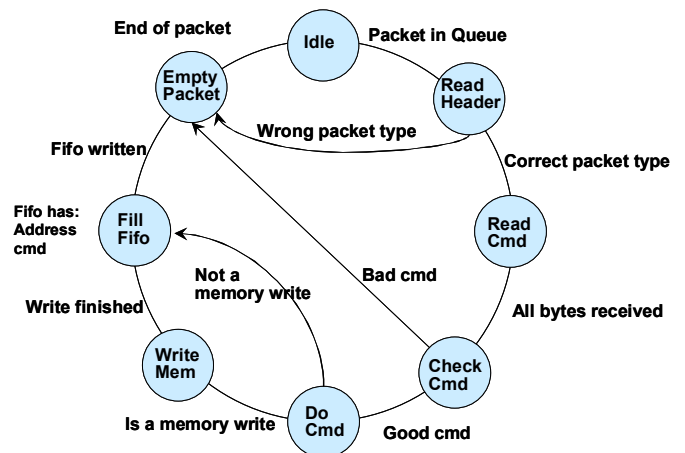


Figure 4. The receive state machine

The TX state machine, shown in Figure 5, loops over the FIFO, processing the data requests. Each reply is constructed and the required data is appended using logic to take into account requests that would span more than one Ethernet packet length. Checksumming is carried out on the whole packet data following the Ethernet header using a standard 16-bit one's complement of the one's complement sum of the data.

This is similar to the standard TCP/UDP/IP one's complement checksum method defined in RFC1071 [6]. The checksum is then appended as the last 2 bytes of the packet during transmission.

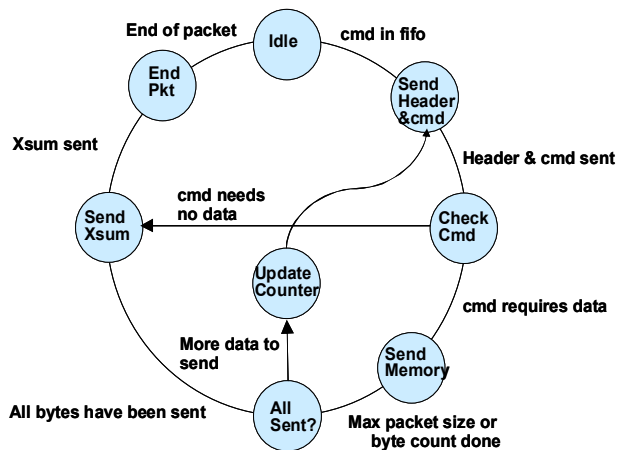


Figure 5. The transmit state machine

V. NETWORK PERFORMANCE OF THE FPGA SYSTEM

A. FPGA Request-Response Performance

The ability of the FPGA to generate responses was tested by sending requests from a PC to the FPGA asking it to return a packet of a specified length. The PC and the FPGA were connected to a Cisco 7600 with gigabit Ethernet. Interrupt coalescence was disabled on the PC for these tests. Figure 6 shows a plot of the round trip latency as a function of the packet size. The plot is expected to be smooth and linear with the slope representing the sum of the inverse data transfer rates for each step of the end-to-end path [7]. The measured slope is 0.018 μ s/byte, which is in good agreement with the expected slope of 0.0182 μ s/byte calculated from the various memory, PCI, Ethernet and FPGA transfer rates. The intercept of 19.7 μ s gives the sum of the propagation delays in the hardware components and the end system processing times. The step in latency at about 1300 bytes is due to a second packet being sent.

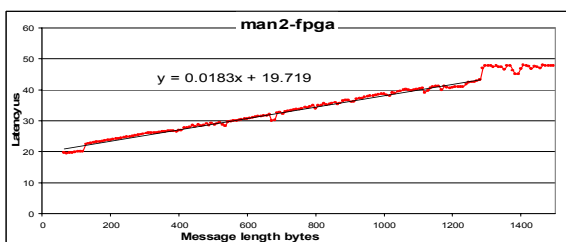


Figure 6. Latency as a function of time for the Request-Response protocol.

The FPGA was then configured to generate streams of 1 million packets. Histograms of the spacing between packets received by the PC are shown in the upper portion of Figure 7 using a log scale and indicate very good performance, FWHM 1.5 μ s. The graphs on the left hand side used a packet spacing of 12 μ s which represents operation at line speed. The right hand side had a spacing of 25 μ s which is equivalent to ~500 Mbit/s. In both cases there are some secondary peaks with a separation of about 5 μ s but these are over 100 times smaller.

The lower plots in Figure 7 show packet loss events during these tests and indicate that groups of up to 6 packets may be lost. This was unexpected and more that that observed when using UDP/IP frames between two PCs under similar conditions [8]. Further investigations are underway.

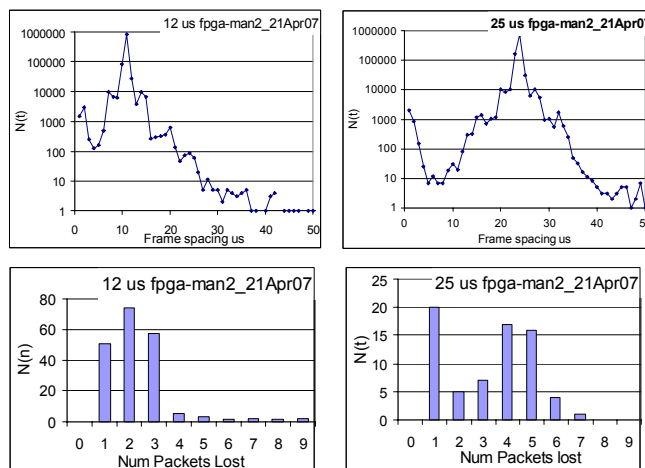


Figure 7. Plots of the packet jitter and packet loss. Left side operating at a packet spacing of 12 μ s, Right operation at 25 μ s. Top: Histograms of the latency Bottom: Corresponding packet loss distributions.

Figure 8 shows a linear relation, with a slope of almost 1, between the mean packet separation observed at the PC and that requested from the FPGA, indicating that the FPGA does indeed generate packets at the spacing requested.

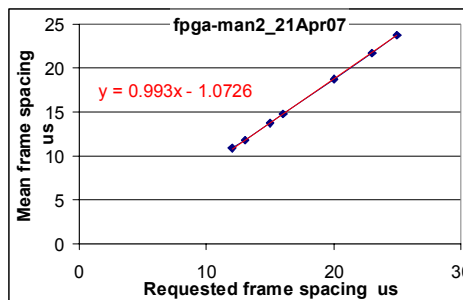


Figure 8. A plot of the mean packet separation versus that requested from the FPGA

VI. PERFORMANCE OF THE DAQ PROTOCOLS

Two methods of requesting data from the concentrators by the PC have been examined: sequential requests to each concentrator and grouped requests where several requests are sent out at once. These possible DAQ protocols are shown schematically in Figure 9, with the thin lines being the requests and the large areas representing the data movement. For these tests the requesting PC used a 10 Gigabit Ethernet connection to the Cisco switch and PC as well as FPGA systems were used as concentrator nodes. Figure 10 shows how the two protocols scale. The blue diamonds show the time taken to build an event using sequential requests increasing with the number of request-responses required. The red circles show the scaling of the grouped requests. In this case the collection time increases by 24.6 μs /node. From the network transit times alone one would expect about 13.3 μs /node, but this does not take any software overheads into account.

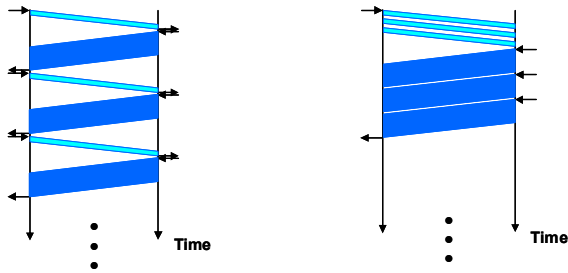


Figure 9. Diagrams of the DAQ protocols investigated.
Left: Sequential requests
Right: Grouped requests

Figure 11 shows histograms of the collection times when data is requested from 2, 3 and 4 nodes using sequential requests. Apart from the mean, the plots are similar with small secondary peak 5 μs later and a group about 25 μs later than the main peak.

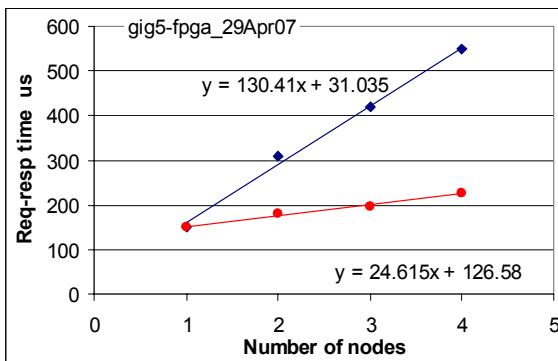


Figure 10. The time taken to build an event as a function of the number of nodes.
Blue diamonds: sequential requests,
Red circles: grouped requests

Figure 12 shows histograms of the collection times when data is requested from 2, 3 and 4 nodes using grouped requests. In this case the histograms are multi-nodal with the first two peaks separated by $\sim 7 \mu\text{s}$.

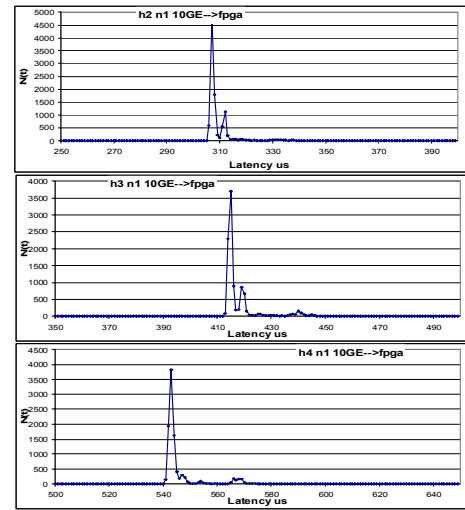


Figure 11. Histograms of the collection times for sequential requests.

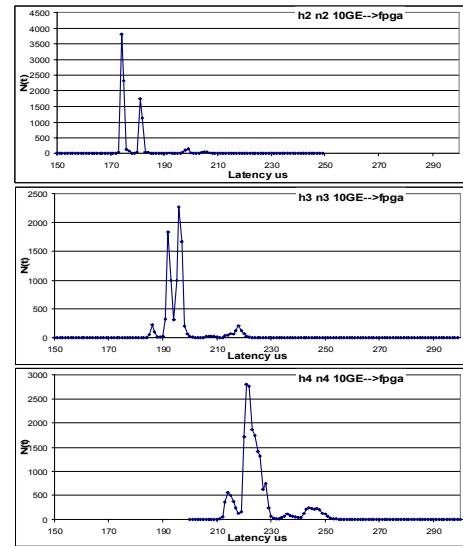


Figure 12. Histograms of the collection times of grouped requests.

VII. CONCLUSIONS

We have implemented a flexible FPGA design to generate Ethernet packets and perform various DAQ data collection protocols. This FPGA has been used as an Ethernet data source and sink to investigate network latency, throughput, and packet loss when communicating with a number of PCs through commercial switches. Our measurements of the network performance show that the FPGA can easily drive 1 Gigabit Ethernet at line rate and that the packet dynamics on the wire are those expected. The packet loss reported by the end system application is being investigated.

It appears that the request-response style of data collection can provide suitable traffic shaping, preventing classic bottleneck queue losses on the egress ports of the Ethernet switch. Further scaling tests and operation at 10 Gigabit are

being planned.

REFERENCES

- [1] *International Linear Collider (ILC) home page*, viewed May 2007, <http://www.linearcollider.org/cms/>
- [2] *The CALICE Collaboration Home Page*, viewed May 2007, <http://polywww.in2p3.fr/activites/physique/flc/collab.html>
- [3] *PLDApplications home page*, 13856 Aix-en-Provence France, viewed Jan 2007 <http://www.plda.com>
- [4] *Xilinx, Inc home page*, California USA, viewed Jan 2007, <http://www.xilinx.com>
- [5] *Home page Green Mountain Computer Systems Inc.*, Vermont USA, viewed Feb 2007, <http://www.gmvhdl.com>
- [6] R. Braden, D. Borman, C. Partridge, *Computing the Internet Checksum*, Request for Comments: 1071, September 1988 <http://www.ietf.org/rfc/rfc1071.txt>
- [7] R. Hughes-Jones and F. Saka, *Investigation of the Performance of 100Mbit and Gigabit Ethernet Components Using Raw Ethernet Frames*, Technical Report ATL-COM-DAQ-2000-014, Mar 2000. http://www.hep.man.ac.uk/~rich/atlas/atlas_net_note_draft5.pdf
- [8] Richard Hughes-Jones, Peter Clarke, Steven Dallison, *Performance of Gigabit and 10 Gigabit Ethernet NICs with Server Quality Motherboards*, Grid Edition of Future Generation Computer Systems Volume 21, Issue 4, April 2005 p 469-488