Precise Latency Comparison Module for the NetFPGA

Adwait Gupte
Algo-Logic Systems
Palo Alto, CA USA
Email: adwait@algo-logic.com

John W. Lockwood
Algo-Logic Systems
Palo Alto, CA, USA
Email: jwlockwd@algo-logic.com

***Abstract*—In many mission critical real-time networked systems, such as those used in financial institutions, incoming data (such as market feeds) is brought in on redundant links. These links are generally provided by separate providers for maximum redundancy. Although the data on both of these links is expected to be the same, there are delays and packet losses that can be different. In this paper we describe a NetFPGA module which can accurately measure these delays to help the institutions evaluate the quality of service provided to them by their vendors.**

I. Introduction

When a packet is transmitted, the total time it takes to reach its destination depends upon several factors such as:

1) The path to destination
2) The propagation delay along the cable/fiber
3) The processing delays added by routers, switches and processing systems along the path

Thus, if two copies of a packet are transmitted through two different networks at the same time, they will not arrive at the destination at exactly the same time. Even if both the networks have similar bandwidth, loading, equipment, cable distances and other characteristics, the packets will randomly arrive at different times. On identical networks you would expect to see a latency difference which has a random distribution with a mean of 0 and a variance depending on the jitter of the systems along the path.

Market feed data received in financial institutions is exceedingly time critical. A delay of hundreds of nanoseconds can cause profits or losses of large amounts of money. As a result, financial institutions are extremely sensitive to latencies of the market feed networks.

Financial institutions typically pay to receive at least two copies of market feed data from two different network providers. One reason is to avoid a single point of failure. In case the network infrastructure of either of the providers goes down, the institution can still continue to receive data from the other network. Another reason to user multiple feeds is to lower latency.

If the services provided by both the network providers are comparable, the average latency between packets on these links would be an average of 0. However, if one of the providers has less latency than the other, data on that network will generally arrive earlier. Such latencies between providers, if they exist, are important to measure and characterize.

In this paper, we present a module which can identify the identical packets arriving on the two links and accurately calculate the latency between the links. The module also provides a software component which reads the latency information from the hardware and plots a histogram of arrival time differences.

Although we use the financial institutions' use of this module as a base use case in this paper, in general this module can be used in any domain. Additionally, the module can also be used to accurately measure the latency added by an appliance/networking gear by connecting the two ports of the NetFPGA to feeds going through and bypassing the appliance in questions. The module allows the granularity of the measurements to be changed on the fly to allow for use in networks that have small or large differences in latency.

This circuit was implemented as a contributed module for the NetFPGA[5][6][7], an open source FPGA based networking platform. The NetFPGA provides the base infrastructure for creating hardware accelerators in a standard PCs. It has a modular interface to the infrastructure to allow custom components to be offloaded in hardware.

The rest of the paper is organized as follows:Section II describes other tools currently available for network analysis. Section III describes the architecture of the latency module. Section IV describes the test setup used to validate the module. Section V presents the findings of our validations tests and Section VI concludes the paper.

II. Related Work

Network measurement tools can be divided broadly into two types: 1)Those meant for general purpose network measurement and 2) Those meant specifically for the financial domain. In this section we discuss some tools falling in to each of these categories and contrast their features against those offered by our module.

As networks become more important to the core businesses of companies, more and more sophisticated network management tools are becoming available. Several general purpose network monitoring tools [1] [2] [3] are currently available in the market. NetTracking DB [1] is a tool which keeps a track of the ARP table entries etc. in the routers on the network. It can also generate usage statistics and identify configuration errors in the network. Scriptlogic [2] is a more

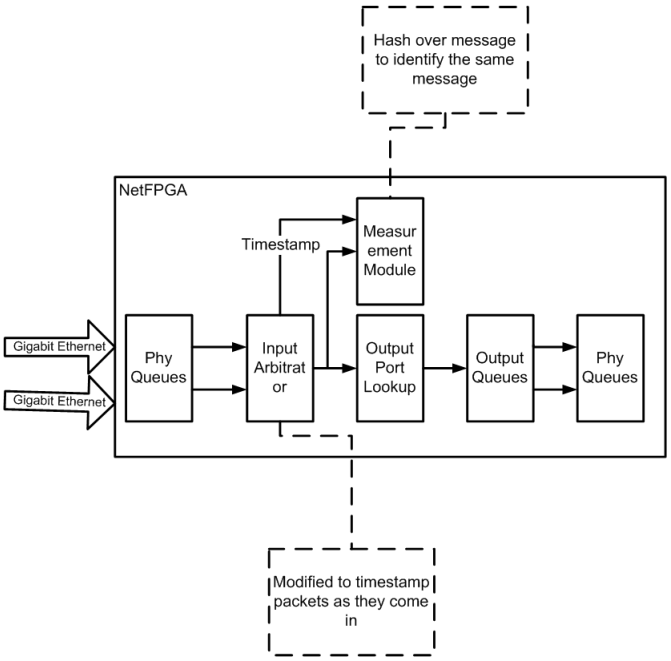


Fig. 1. The measurement module taps into the messages being passed through the pipeline, without adding any latency or interfering with the messages. Thus the measurement is completely passive.

advanced tool which allows the real time monitoring of the network. It also provides traffic analysis capabilities. Similarly Lan Cope [3] provides traffic analysis and similar services. All of these tools provide a general overview of the state of the network and are useful in pinpointing errors within the network. They could also be used to measure the latency added by particular networking devices in the network. However, these tools cannot identify the same packets across different networks and therefore meausre latency differencves between redundant feeds which is the function of the described module.

The second type of the networking tools are domain specific tools which can be used to measure latencies. Endace [4] sells a domain specific network monitoring tool. The Correlix tools offered by Endace is a financial domain specific latency measurement tool. It can track packets across the network from order to execution to identify the latencies of various parts of the network. It also has the capability to identify packets having a causal relation while measuring these latencies. The full featured tool offers latency measurement features compared to module. But our module is easy to use and available as an open source tool.

III. Architecture

Figures 1 and 2 show how module fits in the NetFPGA pipeline and the architecture of the module itself. We discuss these and other architectural issues in detail in the following subsections.

A. Hardware

When measuring any quantity, it is important to avoid altering the quantity by the act of measurement itself. Figure 1 shows where the measurement module fits in to the pipeline of the NetFPGA. Since the module taps in to the data being passed between the input arbiter and the subsequent modules, it doesn't add any latency to the packets. Thus, the measurements made by the module are completely non invasive.

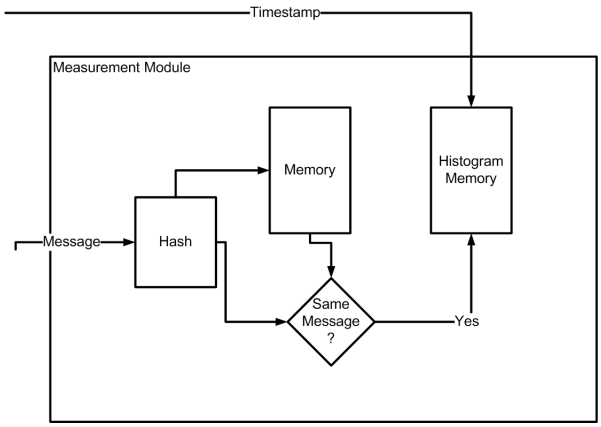


Fig. 2. The figure shows the architecture of the measurement module, described in Section III.

Figure 2 shows a detailed view of the architecture of the measurement module. As a message begins streaming into the module, a 64-bit hash is calculated over the entire message. At the end of the message, the lower 10 bits of the hash are used as a pointer into the memory and the upper 54 bits along with a 32 bit time stamp are stored in to the memory.

If the memory address that a packet hashes to is already in use, and if the part of the hash stored in the memory matches with the current hash, this indicates that the same message has been seen on the other link before. The current time stamp and the time stamp stored in the memory are then used to calculate the latency seen between the links on this particular message. This latency is logged in the memory. The entry made in the memory for the previous packet is invalidated.

If the memory address that a packet hashes to is in use but the part of the hash stored in the memory doesn't match the current match, a hash collision has occurred. This means that two messages which are different happen to have hashed to the same location. In such a case, the current message is simply ignored. Although this means that a data point is lost, since there are a large number of message flowing through the system at any given time, a few lost data points makes no difference to the quality of measurements taken by the module.

To guard against indefinitely waiting for packets which are lost on a link, a time out has been built into the circuit. An entry made in the memory ages out after a certain amount of time. The data points corresponding to such packets are simply not included in the histogram.

One of the problems we faced in getting an accurate time measurement on the NetFPGA was that even if the messages were received at the same time on the ethernet ports, the input arbiter artificially serialized them. This made the timestamp ins measurement module inaccurate. In order to overcome this problem, we modified the input arbiter to timestamp the

messages at that stage itself. This allows message time stamps to be accurate to the clock cycle (@ 125 MHz, ie: accurate to 8 ns) rather than after serialization of the packets which would add error..

B. Software

The hardware in the NetFPGA measures the latencies seen on each of the feeds. These are made available to the software via the register interface provided by the NetFPGA platform.

The software provided with the project, sequentially reads these registers and creates a data file understood by Gnuplot. GNU plot is then used to plot a histogram from the captured data. Figures 6 and 7 show such graphs obtained from GNU plot.

The module exposes 1024 registers to the software for transferring the frequency distribution of the latencies seen between the two links. At the highest level of accuracy, these registers can correspond to steps of 8 ns in latency. However, in some applications, the maximum latencies might be much larger and accuracy to 8 ns might not be required. The software allows users to set the accuracy on the fly from 8 ns steps up to 16 ms steps, a factor of 2000x.

C. Memory Organization

The module exposes 1024 32 bit registers to the software. Each of these registers corresponds to the count of messages experiencing delays within a certain range. Depending on the range of delays expected, the accuracy of these buckets can be varied.

The register file is divided into two parts. The upper half corresponds to the counts of packets being delivered first on one link and the lower half corresponds to the packets being delivered first on the other link.

For example, to be accurate to 8 ns, there would be 1024 registers each corresponding to 8 ns delays on either of the links. Thus the maximum possible delay between the two links could be a maximum of (512 * 8 = 4096 ns).

The software plots the upper half of the registers as negative values on the X axis while the lower half are plotted as positive values. The Y axis corresponds to the counts while the X axis corresponds to the delay in ns.

D. Parameters

Depending on the domain in which this module is used, some details of the implementation might need to be adjusted. The following are the parameterizable properties of the module:

1) Size of Bucket: There are 1024 software accessible registers or "buckets" available by default. A number of readings within a certain range are combined into one bucket. The size of the bucket decides how accurate the readings obtained from the module are. eg: For a bucket size of 16, packets which are delayed by 1-16 ns will be combined into a single bucket. Depending on the accuracy required, one might want to change this bucket size. This has been made a run-time parameter so that when taking measurements, the accuracy can be adjusted on the fly according the the characteristics of the network under test.

2) Number of Buckets: By default there are 1024 buckets. This number can be increased if required by allocating more memory to it. The number of buckets and the size of each bucket together decide the accuracy and the maximum value of difference between the two links.

3) Hash Function Input: The default design hashes over the entire packet including headers. Our module performs an XOR of each 64 bit word of the packet. In the end the upper and lower 32 bit values of the hash are XORed again to form the 32 bit hash. However, in some cases it might be necessary to ignore certain fields such as the MAC address fields etc. This parameter can be used to decide which position in the packet to begin the hashing.

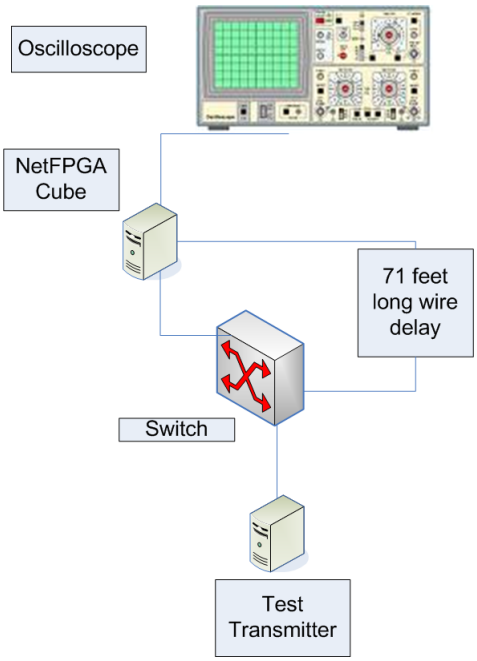


Fig. 3. The test setup used for the module.

IV. Test Setup

Figure 3 shows the setup used to test the module. One port of the NetFPGA was connected to the Gigabit Ethernet switch through a short CAT6 cable while the other was connected through a CAT6 cable 71 feet longer. This was done to be able to add a deterministic delay to one of the ports when measuring the latency.

The RGMII PHY used on the NetFPGA has signals "rgmii_(0-3)_rx_ctl" which go high when data is being transmitted/received on the corresponding LAN ports. These signals were connected to the debug bus IO pins on the NetFPGA and then viewed on an oscilloscope. This allowed us to independently measure the difference between the start times of the messages on the LAN ports to compare against the values measured by our module. Figure 4 shows the readings from the oscilloscope.

V. Results

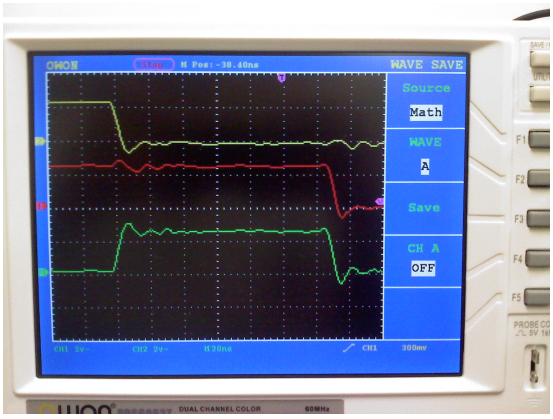
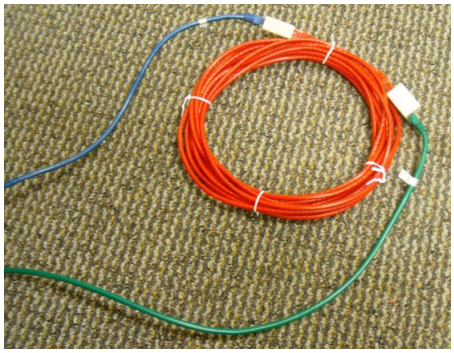


Fig. 4. This image shows a zoomed in view of the time difference to be able to measure it. The resolution is 20 ns/division and so the total latency between the two ports is shown to be around 130 ns. The top lines are the activity signals on the ethernet port while the bottom line is the difference between the two.

(a) The router used for the experiment



(b) The 71 feet long wire used to add the delay

Fig. 5. Some more images showing the test setup.

The propagation delay of a cable is about 5.5 ns/meter. The delayed port was connected by a cable which was 71 feet (21.64 meters) longer than the other and therefore we expected to see a latency of about 119 ns (5.5*21.64). The readings we measured from the module instead indicated a latency of about 130 ns (seen in Figure 6).

In order to verify expected and actual results, we used the "rgmii_(0-3)_rx_ctl" pins explained in Section IV. As seen in Figure 4 the delay seen on the wire was approximately 130 ns as measured by our module.

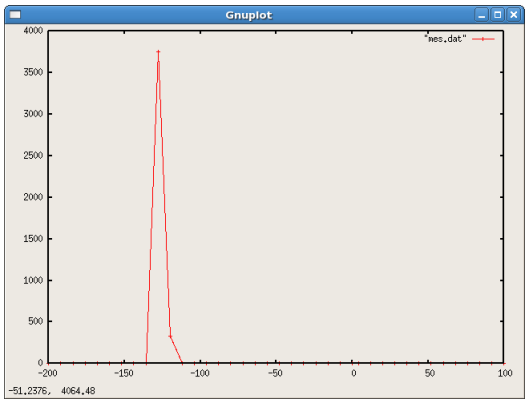


Fig. 6. The latency seen between two ports of the NetFPGA when a wire delay was added on one of the ports. The graph shows a latency of around 130 ns on one of the ports

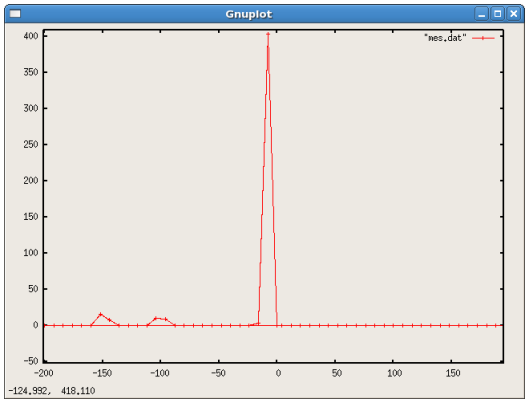


Fig. 7. The latency seen on the two ports of the NetFPGA when no wire delay is connected. This is simply the delay added by the switch when forwarding broadcast messages

To account for the excess delay over and above the propagation delay of the longer wire, we eliminated the longer wire and connected both ports of the NetFPGA directly to the ports on the switch. Figure 7 shows the measurements in this test case. The delay here is seen to be around 10 ns. Along with the expected propagation delay of the longer wire, this account for the approximately 130 ns delay measured by our module.

Thus we were able to verify the measurements of the module not only with out expectations but also from independent readings taken from the oscilloscope.

VI. Conclusion

We have provided the design and implemented a module to accurately measure the latencies between links transmitting the same data. The module in addition to the financial domain utilization outlined here, can also be used to measure the latencies added by a particular networking hardware. The source code can be downloaded at www.algo-logic.com/latency/download.htm

REFERENCES

[1] NetTrackingDB, *http://netdbtracking.sourceforge.net/*

[2] Script Logic, *http://www.scriptlogic.com/products/perspective/*

[3] Lan Cope, *http://www.lancope.com/*

[4] Endace, *http://www.endace.com/latency-measurement.html*

[5] G. Gibb, J. W. Lockwood, J. Naous, P. Hartke, and N. McKeown. Netfpga: An Open Platform for Teaching How to Build Gigabit-rate Network Switches and Routers., *In IEEE Transactions on Education, August 2008*

[6] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo. NetFPGA - An Open Platform for Gigabit Rate Network Switching and Routing., *In International Conference on Microelectronic Systems Education, 2007.*

[7] G. Watson, N. MxKeown, and M. Casado. NetFPGA - A Tool for Network Research and Education., *In 2nd Workshop on Architecture Research using FPGA Platforms (WARFP), Febuary 2006*