Correlator Data Transport Benchmark Testing of FPGA/GPU Correlators using the HASHPIPE Software and 10Gbit Ethernet

Christopher Schollar **10/10/2012**

Hashpipe is code used to move data through an X engine running on a CPU and GPU. It handles collecting packets from multiple F engines through a NIC (Network Interface Card), re-ordering the data and converting it to a type appropriate for processing on the Graphics Card. It then sends data from the system RAM to the GPU RAM and calls the xGPU code which performs the X operation on the inputs provided. After correlation hashpipe copies the output of the CUDA X-engine (xGPU), re-orders/re-interprets the data and sends the data to a catcher machine which handles further accumulation, data aggregation and saves the data in a miriad file.

The Hashpipe code is the a derivative of GUPPI, software developed by NRAO (National Radio Astronomy Observatory) for FPGA/GPU pulsar instrumentation. It was then modified/adapted for by UC Berkeley for FPGA/GPU for the Green Bank VEGAS multibeam spectrometer . Most recently it has been further modified/adapted by UCB for use in FPGA/GPU correlators under the name HASHPIPE and this is the code I have been testing.

When correlating astronomical data in real time, the amount of samples correlated per second is directly related to the sky bandwidth and therefore we have a minimum required data rate that hashpipe can pass through a X-engine for any given configuration of number of inputs, number of channels and sky bandwidth correlated. Therefore we must be confident that hashpipe can handle the data rates we require for a given instrument and it is with this goal in mind that I have tested the throughput of hashpipe.

These tests were performed on a machine identical to those which have been sent to South Africa to be used as part of the new PAPER correlator. The vital statistcs are listed below.

1. Motherboard : Supermicro X8DTG-QF
2. CPU : 2x Intel Xeon E 5620 Quad-Core 2.40 Ghz
3. NIC : 2x 10GbE Intel SVR PCIE EXP*X9502*CX4 (Dual 10GbE NICs)
4. GPU : 2x Nvidia GeForce GTX 580

The NIC cards and Graphics cards where all plugged into PCIe2.0 x16 busses, each providing 8 GB/s transfer along the busses to the CPU, well above the maximum speed that the 10GbE NICs can capture packets.

In the tests I found that **hashpipe could handle 9.9 Gb/s** with **each instance** running **using one 10GbE port for each instance**. I tested this machine with 2 instances of hashpipe running simultaneously and got 9.9 Gb/s throughput on both instances, providing 19.8 Gb/s for 1 machine with the specifications above using two 10GbE ports (ie 1 port on each NIC card). Below I detail how the packets were generated. This was running the X-Engine with a configuration for 64 channels and 128 inputs (or 64 dual polarization antennas)

Packet Generator

In order to test the throughput of the X-Engines, I needed a reliable way to send data to the NIC cards where I could control the data rate and the packets were of the same format as those generated by the F-Engines in the PAPER correlator. In order to do this I took a packet Generator which had been developed by Gilbert Hsyu at UC Berkeley and modified it for my purposes. This packet generator runs

on a ROACH1 board and can output UDP packets with payloads from 8 Bytes to 9 Kbytes (or more, but the NIC cards can handle a max size of 9KB) and can control the data rate of the payload (ie. without taking into account the overhead of UDP headers). This ensures that hashpipe was transferring 9.9 Gb/s of real data.

The packets generated were all identical, except for the header's containing a Message Count, F engine ID and X engine ID. The packets generated were of the same format as would be generated by PAPERs ROACH based F Engine.