Hardware UDP

High Performance Simultaneous Data Acquisition

Summary and reference document for details relating to D-TACQ support for Hardware UDP on ACQ2106 Carrier via SFP Port D on MGT482-SFP.

Revision History

Revision	Date	Author(s)	Description
1.0	26/05/2022	SR	Created
2.0	06/07/2022	SR	New diagrams. More prose. Added HUDP to Host section
3.0	15/07/2022	SR	Section better specifying packet structure and max limits
4.0	18/01/2023	SR	Some formatting fix-ups

Contents

1 Low Latency Demo Setup

1.1 HW

- AI Box : acq2106_189
- AO Box : acq2106_274

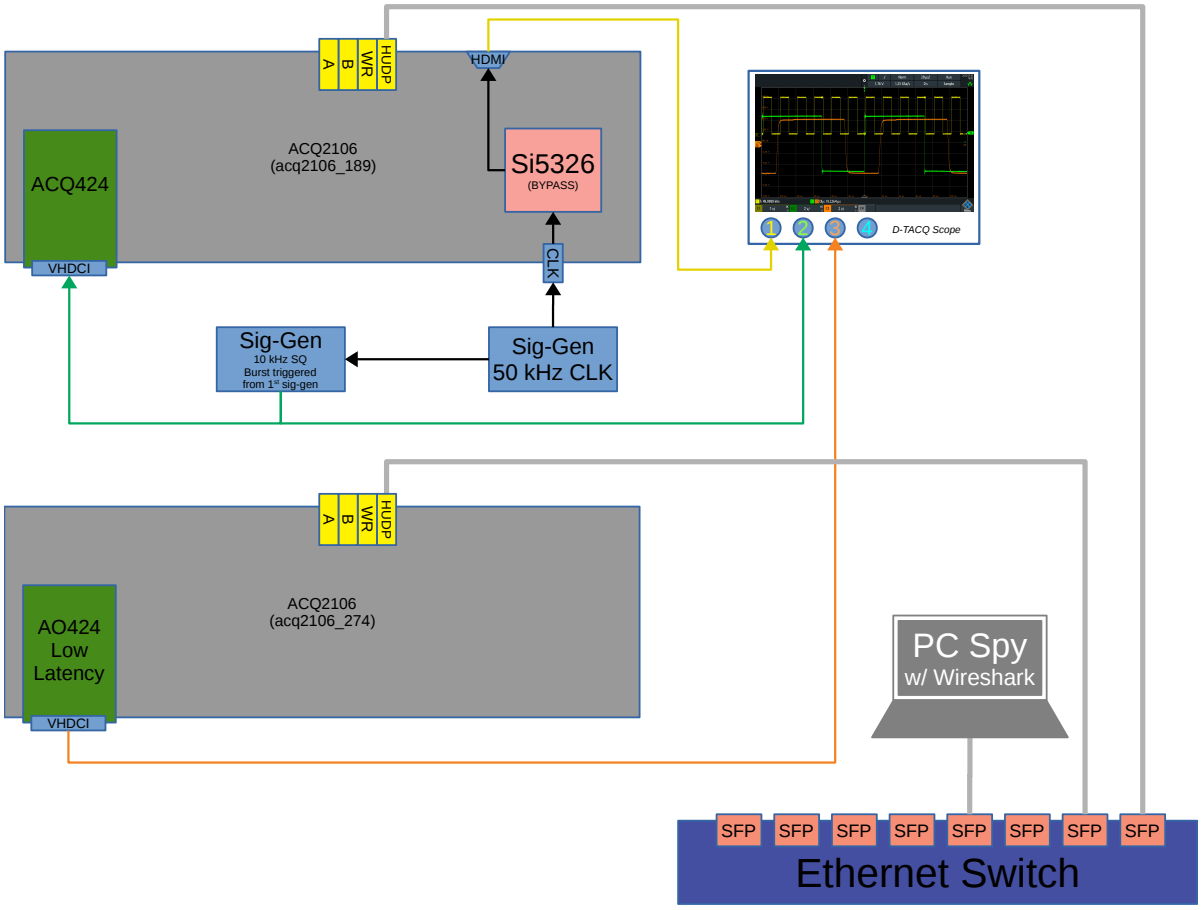

Figure 1: HUDP Demo Hardware Configuration

We set the AI box clock to bypass so that there is no phase delay between the clock from the signal generator and the clock which the ADC sees. The AO is in Low Latency mode (update as soon as you have data) so the clock on the AO box is less important.

Bring the clock back out of HDMI from the 2106 before displaying on scope. This tells us for certain that the Si5326 is successfully in bypass mode and has the added benefit of showing the user where the sample clock lies in relation the their excite signal.

The signal generator which is providing the clock signal, is also driving the External Trigger input of a secondary function generator. This secondary function generator outputs a square wave burst on the rising edge of the clock (not on every edge, burst period is longer than clock period).

We can then control the phase of the burst signal relative to the rising edge of the clock to account for the aperture delay and slew rate of ACQ424 input.

See figures in Section 8 for a detailed look at the oscilloscope traces and latency measurements.

1.2 HUDP Core Configuration

D-TACQ have provided a handy Python interface to help when configuring the HUDP Core.

```
[dt100@eigg-fs acq400_hapi]$ ./user_apps/acq2106/hudp_setup.py -h
usage: hudp_setup.py [-h] [--netmask NETMASK] [--tx_ip TX_IP] [--rx_ip RX_IP]
                     [--gw GW] [--port PORT] [--run0 RUN0] [--play0 PLAY0]
                     [--broadcast BROADCAST] [--disco DISCO] [--spp SPP]
                     txuut rxuut

hdup_setup

positional arguments:
  txuut                 transmit uut
  rxuut                 transmit uut

optional arguments:
  -h, --help            show this help message and exit
  --netmask NETMASK     netmask (default: 255.255.255.0)
  --tx_ip TX_IP         rx ip address (default: 10.12.198.128)
  --rx_ip RX_IP         tx ip address (default: 10.12.198.129)
  --gw GW               gateway (default: 10.12.198.1)
  --port PORT           port (default: 53676)
  --run0 RUN0           set tx sites+spad (default: 1 1,16,0)
  --play0 PLAY0         set rx sites+spad (default: 1 16)
  --broadcast BROADCAST broadcast the data (default: 0)
  --disco DISCO         enable discontinuity check at index x (default: None)
  --spp SPP             samples per packet (default: 1)
```

For our example above (Figure 1), the configuration is as follows.

We execute a `sync_role` command to place the AI box's clock synthesiser in bypass.

```
./user_apps/acq400/sync_role.py --fin=50k --fclk=50k --si5326_bypass 1 --toprole=fpmaster,strg acq2106_189
```

Then we set up a TX-RX pair between box acq2106_189 and acq2106_274. Here we use the broadcast address .255 so that the PC which is also in the network can easily spy on the packets as they fly past.

```
./user_apps/acq2106/hudp_setup.py --tx_ip 10.12.198.128 --rx_ip 10.12.198.255 --run0='1 1,16,0' --play0='1 16' \
--broadcast=1 --disco=16 acq2106_189 acq2106_274
```

1.3 CSS OPI

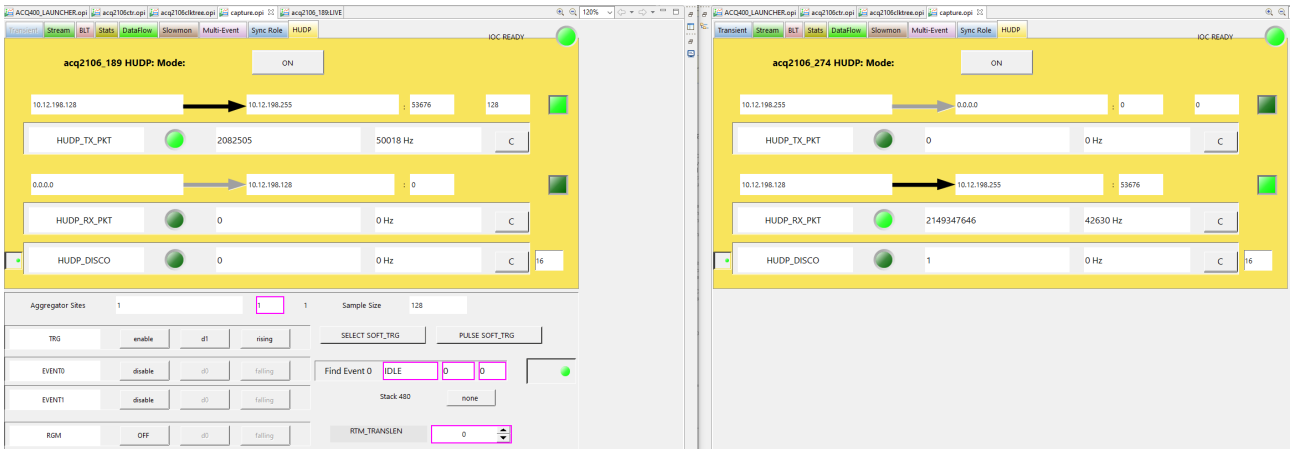

Figure 2: HUDP OPIs : A TX-only box on the left and an RX-only box on the right

1.4 Discontinuity Counter

The discontinuity (disco) counter uses the embedded sample counter (SPAD0) contained within every sample/-packet to verify that we have had no skips or lost packets throughout the course of a run.
Check that the latest received sample count is equal to the previously decoded sample count plus 1. Note, this

will fire once at the start of every shot (when our previous sample count is assumed to be zero and the first received sample count will also be zero.) It will also fire spuriously on the rollover of the 32-bit counter (once a day at 50 kHz). We could enhance the logic to account for these exceptions.

2 Clock and Excite Signal Phase Relationship

Here we demonstrate the effect of sampling the analogue rising edge at different points in the low-to-high transition. Even though the signal generator produces a very sharp rising edge, a combination of the bandwidth and slew rate limitations of the front end of the ADC mezzanine, ”flatten out” the rising edge somewhat.

- With the sample clock almost coincident with the analogue rising edge (far left) the ADC only manages to capture the very beginning of the rising edge, resulting in a very low amplitude update on the DAC.
- Pulling the excite signal forward in time (middle section) means that the sample clock is in effect, delayed, and thus samples further up the slope of the rising edge. This results in a higher amplitude update from the DAC.
- With the rising edge now having almost completed its low-to-high transition before the sample clock arrives, the ADC samples much further up the slope and the output from the DACs is almost full scale.

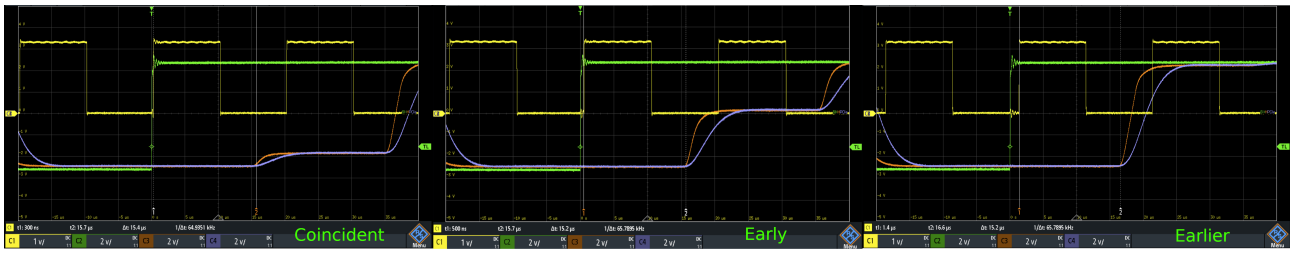

Figure 3: The result of the sample clock sampling the rising edge at various points during the signal slew

3 HUDP - Internal workings

Here we have a more detailed diagram which helps visualise data flows, both within an individual box employing the HUDP core, and in a larger system utilising HUDP as the backbone for data transfer.

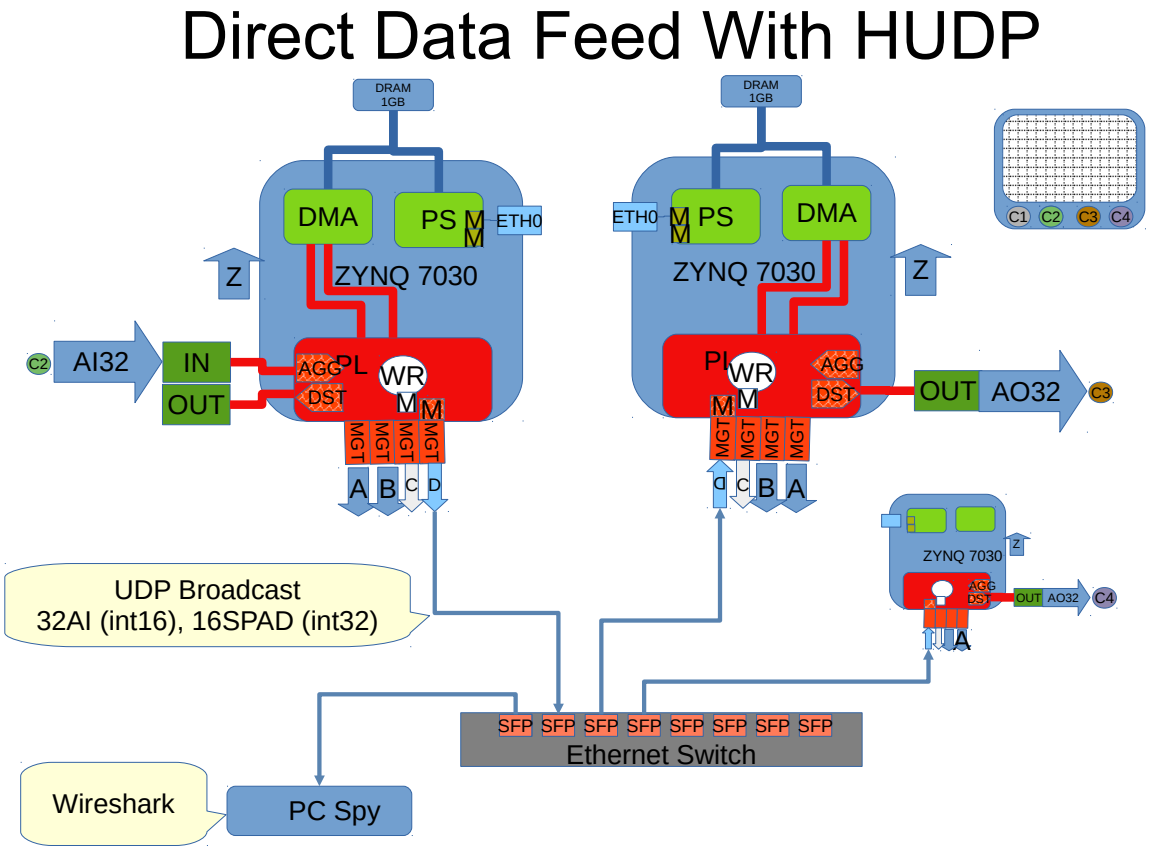

Figure 4: More detailed example of HUDP dataflows

This helps to visualise the entire datapath. Beginning at C2 AI32 we see the data flow into the site, through the Aggregator, down to the HUDP core and associated Mulit-Gigabit Transceiver (MGT) port.
We push packets out on SFP, through the Ethernet switch and back to the RX box. These packets hit the HUDP core on the AO box, data is stripped and passed down to the Distributor which is responsible for sharing out the data to selected sites. From here we carry on to the AO32 site and eventually back out into the analogue domain to be sampled by the oscilloscope (C3).

4 HUDP to Host

4.1 Programs

4.1.1 tcpdump

```
[dt100@naboo ~]$ sudo tcpdump -i enp5s0 -n udp port 53676 -X -c 1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp5s0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:55:41.732280 IP 10.12.196.2.53676 > 10.12.196.227.53676: UDP, length 1248
        0x0000:  4500 04fc 0000 4000 4011 98f3 0a0c c402  E.....@.@.......
        0x0010:  0a0c c4e3 d1ac d1ac 04e8 7912 202b 9897  ..........y..+..
        0x0020:  2125 0400 226a 0400 2381 fbff 2499 ffff  !%.."j..#...$...
```

4.1.2 netcat

Beware firewalls! Check your firewall settings. It may be set to block UDP traffic outwith known ports.

```
[dt100@eigg-fs ch_data]$ nc -ulv 10.12.196.15 53676 | pv > shot_data
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on 10.12.196.15:53676
Ncat: Connection from 10.12.196.2.
58.1MiB 0:00:08 [   7.68 MB/s] [                          <=>  ]
```

4.1.3 isramp

Looks for the sample counter ramp in the data and checks for discontinuities (i.e. lost packets).

```
[dt100@eigg-fs ch_data]$ nc -ulv 10.12.196.15 53676 | pv | isramp -m 104 -c 96
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Listening on 10.12.196.15:53676
Ncat: Connection from 10.12.196.2.
```

`-m` = Number of LWords in sample
`-c` = Sample counter in column 96, counting from 0

The isramp program can be found at :
https://github.com/D-TACQ/AFHBA404/blob/master/FUNCTIONAL_TESTS/isramp.c.

4.2 Worked Example including multiple samples per packet

```
./user_apps/acq2106/hudp_setup.py --tx_ip 10.12.196.2 --rx_ip 10.12.196.227 --gw 10.12.196.1 \
--run0='1,2,3,4 1,8,0' --spp 3 acq2106_178 none
```

Capture from sites 1, 2, 3 & 4 with a SPAD length of 8.
acq2106 local IP = 10.12.196.2, transmitting to 10.12.196.227
Packing 3 samples into a each packet (samples per packet (spp)).

```
acq2106_178> get.site 0 aggregator
reg=0x003e8079 sites=1,2,3,4 threshold=16384 DATA_MOVER_EN=on spad=1,8,0
acq2106_178> get.site 0 NCHAN
104
acq2106_178> get.site 0 ssb
416
acq2106_178> get.site 10 tx_spp
3
acq2106_178> get.site 10 tx_calc_pkt_sz
1248
```

```
nc -ulv 10.12.196.227 53676 | pv > shot_data
```

5 Data rates to host - Testing

5.1 To the fibre port on Naboo

Can this interface cope with more packets per second than a vanilla copper RJ45 on the mobo?

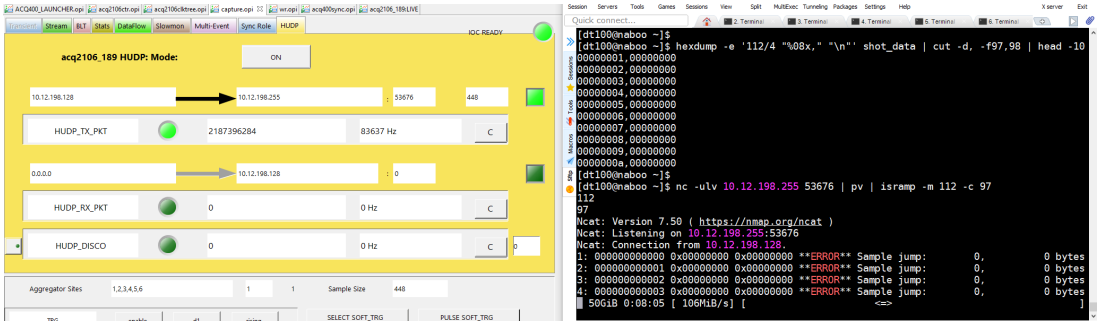

Figure 5: 250 kHz, 112 MB/s

$$250\ \text{kHz} \times 448 = 112\ \text{MB/s}$$

$$\frac{250\ \text{kHz} \times 448}{2^{20}} = 106.8\ \text{MiB/s @ 3 spp = 83,333 packets per second}$$

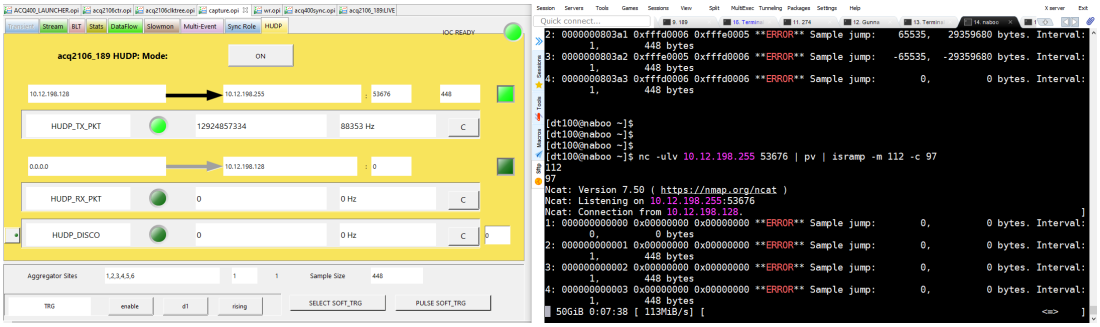

Figure 6: 265 kHz, 118 MB/s

$$265\ \text{kHz} \times 448 = 118.72\ \text{MB/s}$$

$$\frac{265\ \text{kHz} \times 448}{2^{20}} = 113.22\ \text{MiB/s @ 3 spp = 88,333 packets per second}$$

5.2 To the copper port on Naboo

The copper test is going through a busier switch. No material difference in packet rate handling observed between the two flavours of interface. Further testing would be useful.

6 Max Specifications

Frame Section	Size (Bytes)	Comment	Size (Bytes)
Inter-Frame Gap (IFG) (96 ns)	12	$(125\ \text{MB/s} \times 96\ \text{ns} = 12\text{B})$	12
MAC Preamble	8	including Start of Frame Delimiter (SFD)	8
MAC Destination Address	6		6
MAC Source Address	6		6
EtherType (or Length)	2		2
IPv4 Header UDP Header User Data (CALC_PKT_SIZE)	20 8 1472	Payload MTU 1500B	20 8 1472
Frame Check Sequence (FCS)	4		4
Total Frame Size	1518	Total w/ overheads (includes IFG and Preamble)	1538

Table 1: Breakdown of an Ethernet Frame from HUDP Core

Gigabit Ethernet raw line rate = 1.25 Gb/s,
accounting for 8B/10B encoding leaves us with 1 Gb/s,
this equates to 125 MB/s.

So we can transfer 125 million bytes per second, and our maximum frame size is 1538 bytes. This gives us a maximum packet rate of :

$$\frac{125 \times 10^6}{1538} \approx 81,274 \text{ frames per second}$$

This means that our maximum rate for transferring (useful) user data is as follows :

$$\begin{aligned} 81,274 \times 1472\text{B} &= 119.635\ \text{MB/s} \\ &\approx 114\ \text{MiB/s} \end{aligned}$$

From the above we can see that our total Ethernet overhead is equal to :

$$1538 - 1472 = 66\text{B}$$

To calculate our max packet rate for any sample size, and hence our maximum sample rate in a low latency system, we would use the following equation :

$$\frac{125 \times 10^6}{(\text{Sample Size} + 66)} = \text{Max. packets per second} \tag{1}$$

6.1 Worked Examples

6.1.1 Small Sample - Low Latency

Let's look at an example with small sample sizes. 1 ACQ424 card with a Scratchpad of 8 LWords.

$$\text{1 ACQ424 = 32CH} \times 2\text{B} = 64\text{B, 8 SPAD LWords = 8} \times 4\text{B} = 32\text{B}$$

$$\text{Total Sample Size = } = 64 + 32 = 96\text{B}$$

Our 96B sample size means that we can manage :

$$\frac{125 \times 10^6}{(96 + 66)} = 771,604 \text{ packets per second, (74 MB/s)}$$

6.1.2 Multiple Samples per Packet - High Throughput

6 ACQ424 cards with a Scratchpad of 16 LWords and 3 samples per packet.

$$6\ \text{ACQ424} = 6 \times 32\text{CH} \times 2\text{B} = 384\text{B},\ 16\ \text{SPAD LWords} = 16 \times 4\text{B} = 64\text{B}$$

$$\text{Total Sample Size} = = 3 \times (384 + 64) = 1344\text{B}$$

Our 96B sample size means that we can manage :

$$\frac{125 \times 10^6}{(1344 + 66)} = 88,652 \text{ packets per second, (119 MB/s)}$$

7 Monitoring Packets with Wireshark

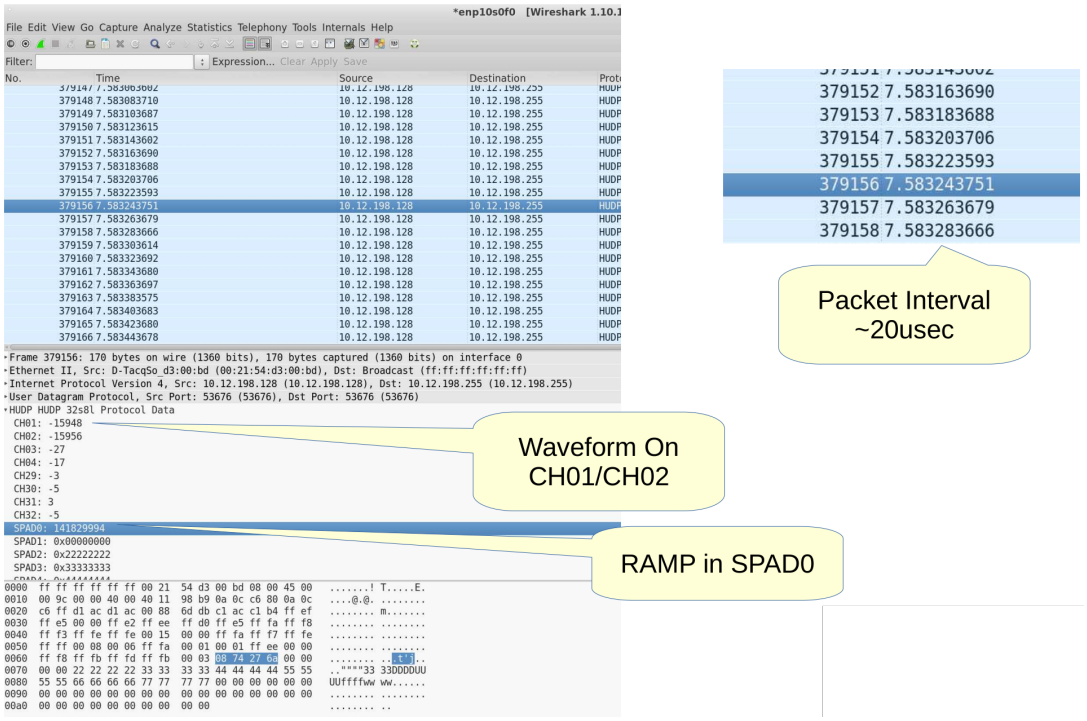

Figure 7: Monitoring packets with Wireshark plugin

https://github.com/D-TACQ/CUSTOM_WRPG/blob/master/CARE/dot.wireshark.plugins.hudp32s_8l.lua

8 Latency Measurements

8.1 Point-to-Point

Best possible result. AI box straight to AO box. Very stable. 12.6 µs. We should repeat this test using a 2 MHz ACQ425 (smallest possible aperture delay) and an AO424-LLC in 16CH mode to minimise both input and output delay as well as the smallest possible packet. This could then be compared to the Aurora latency record of 5.1 µs from the D-TACQ LLC White Paper.

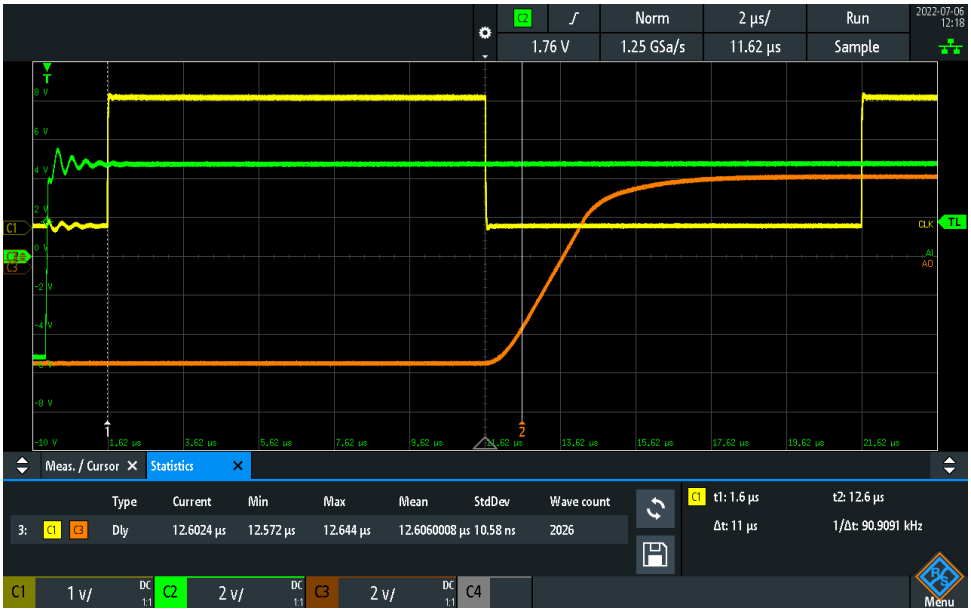

Figure 8: HUDP Point-to-Point Latency

8.2 Through Switch

Note the extra jitter incurred by the trip through the switch in addition to the further ≈ 5 ns of latency. 17.4 µs.

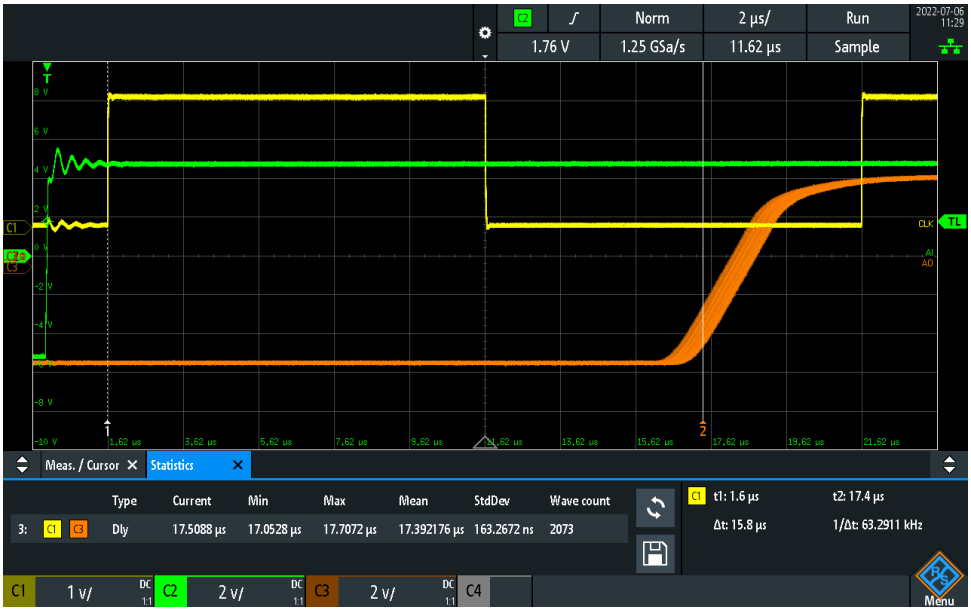

Figure 9: HUDP Latency through Switch