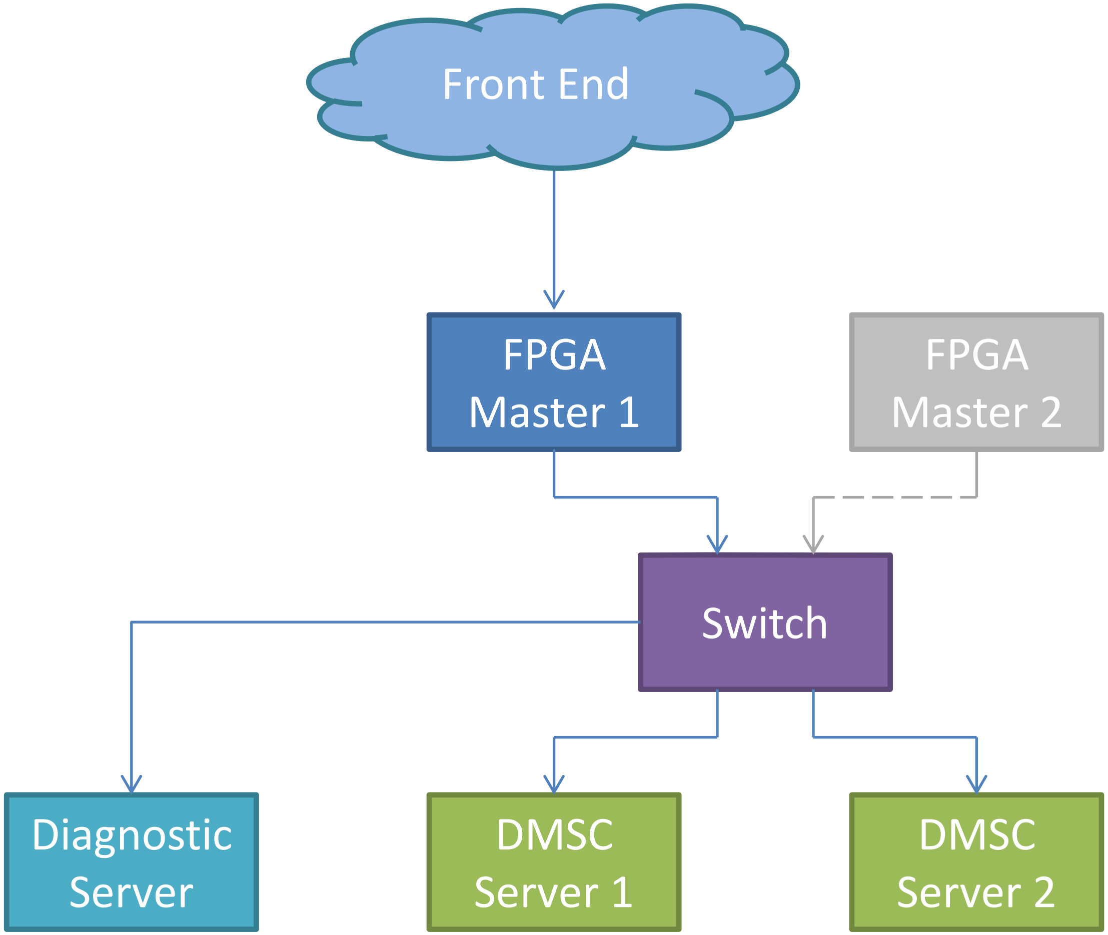
EUROPEAN
SPALLATION
SOURCE

High Rate Packet Generator

Second DMSC Detector Group Readout Meeting
21st February 2018
Steven Alcock (DG)

Simplified Readout Electronics Architecture



Steven Alcock, Detector Group, 21st February 2018

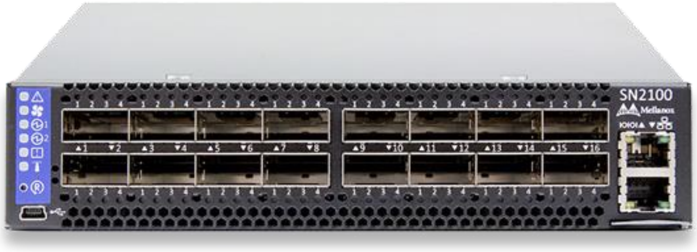
100 G Detector Readout Demonstrator

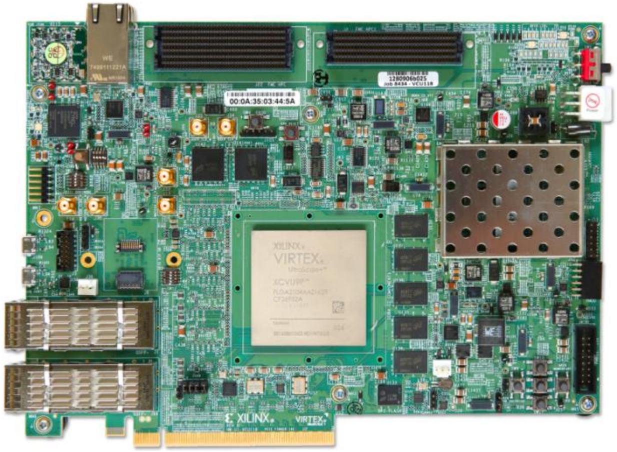
- Provides the DMSC with high-rate packets in a format representative of the final system.
- Allows data packing format to be agreed.
- Proves key interfaces.

DETECTOR GROUP ZERO-X 100G DATA GENERATOR
ESS
DETECTOR GROUP READOUT
Science & Technology Facilities Council

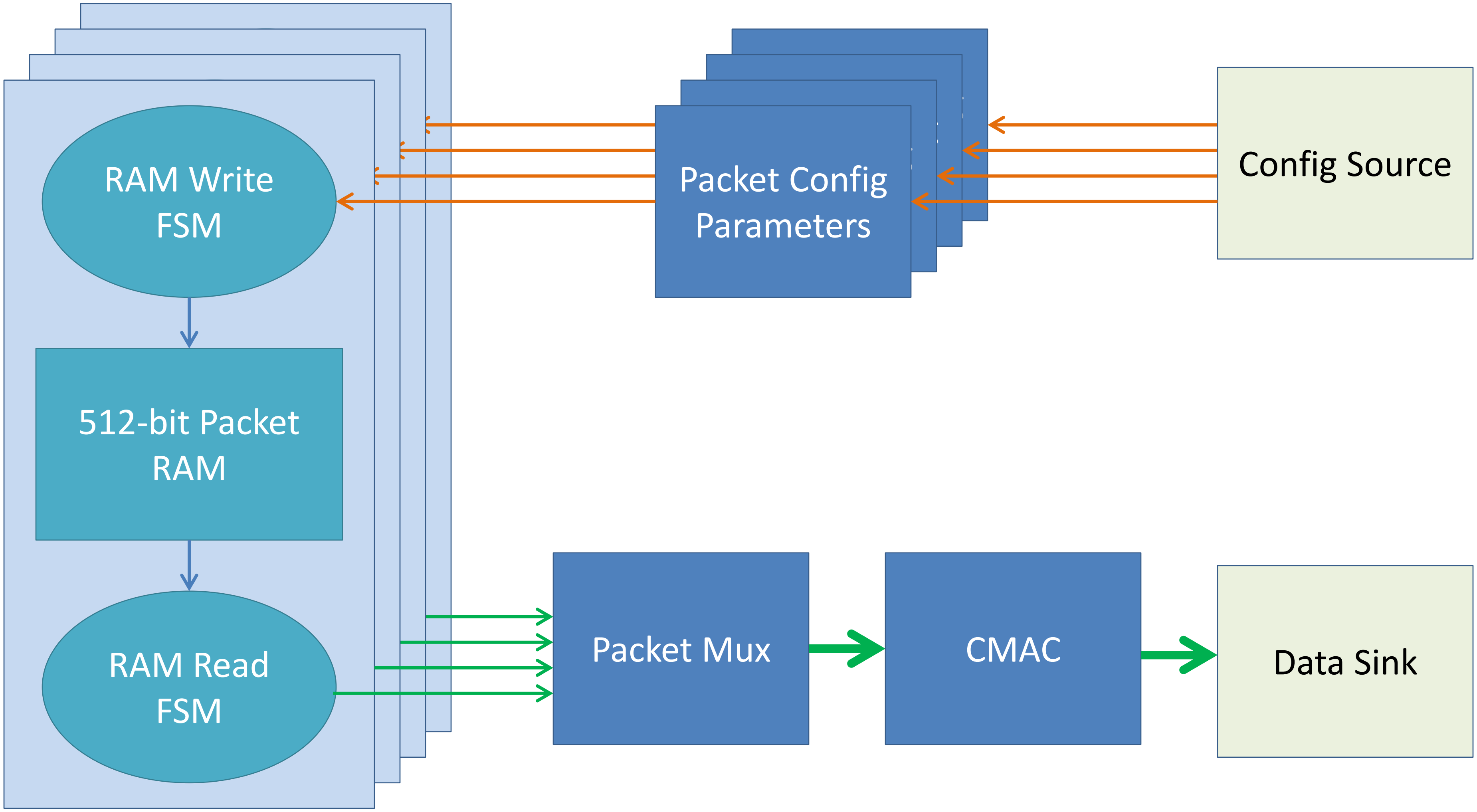
Key Features

- Four packet engines are independently configurable at run-time.
- Aggregate output data rate is configurable at run-time.
- Aggregate number of packets is configurable at run-time.
- Mellanox Switch can output data as 10/25/40/56/100 G. This means, for example, the full 100 G input could be split across 10 x 10 G outputs.



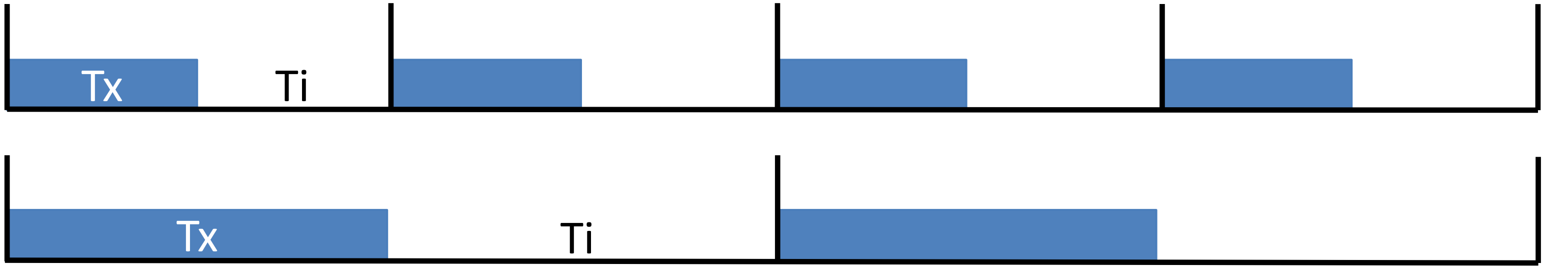


FPGA Architecture



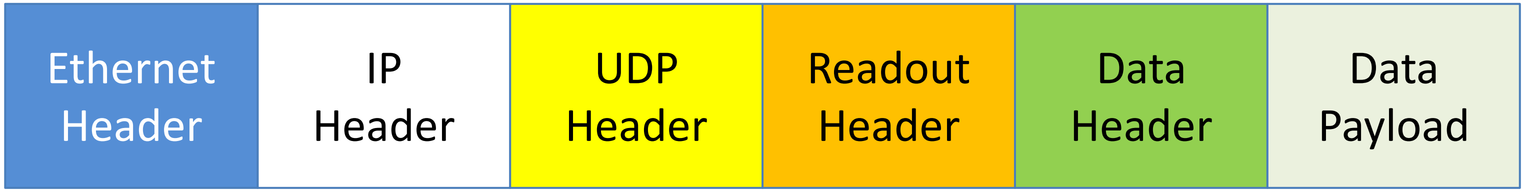
Steven Alcock, Detector Group, 21st February 2018

Variable Packet Rate Design

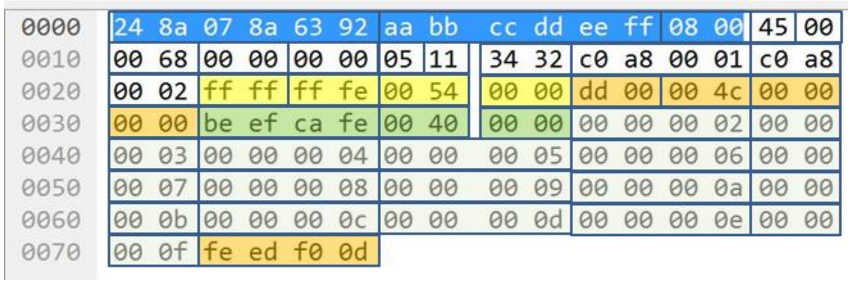


- Data rate is ratio of time spent transmitting at 100 G (Tx) to time spend idle (Ti).
- Tx = {Packet Length in bits} / {100 G Transmission Rate} = L/100e9
- Ti = {Number of Idle Clock Cycles} / {Processing Clock Speed} = G/322.265625e6
- Therefore the new time taken to transmit one packet becomes {Tx + Ti}
- The new rate becomes R = {L/(Tx + Ti)} = {L/(L/100e9 + G/322.265625e6)}
- Rearranging gives G = {322.265625*(L/Re5)*(100-R)} where G is in Gbps

Packing Format Example 1

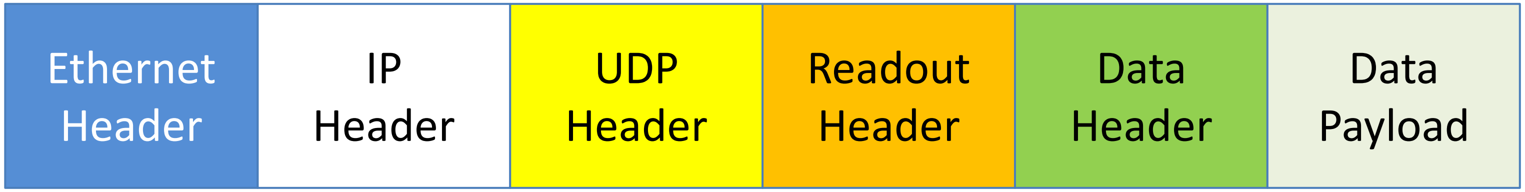


Ethernet Header | IP Header | UDP Header | Readout Header | Data Header | Data Payload



```
0000  24 8a 07 8a 63 92 aa bb  cc dd ee ff 08 00 45 00
0010  00 68 00 00 00 00 05 11  34 32 c0 a8 00 01 c0 a8
0020  00 02 ff ff ff fe 00 54  00 00 dd 00 00 4c 00 00
0030  00 00 be ef ca fe 00 40  00 00 00 00 00 02 00 00
0040  00 03 00 00 00 04 00 00  00 05 00 00 00 06 00 00
0050  00 07 00 00 00 08 00 00  00 09 00 00 00 0a 00 00
0060  00 0b 00 00 00 0c 00 00  00 0d 00 00 00 0e 00 00
0070  00 0f fe ed f0 0d
```

Packing Format Example 2



Ethernet Header

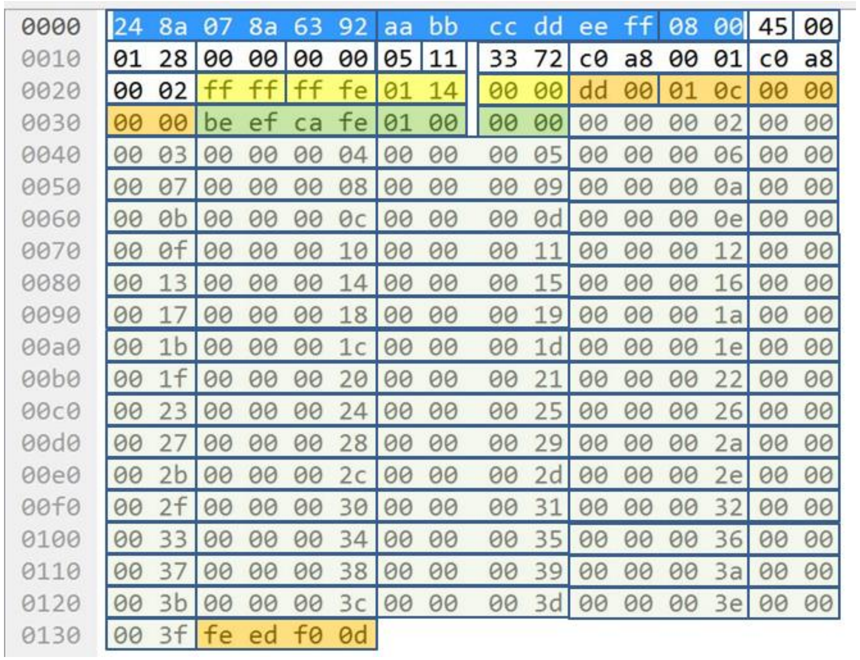
IP Header

UDP Header

Readout Header

Data Header

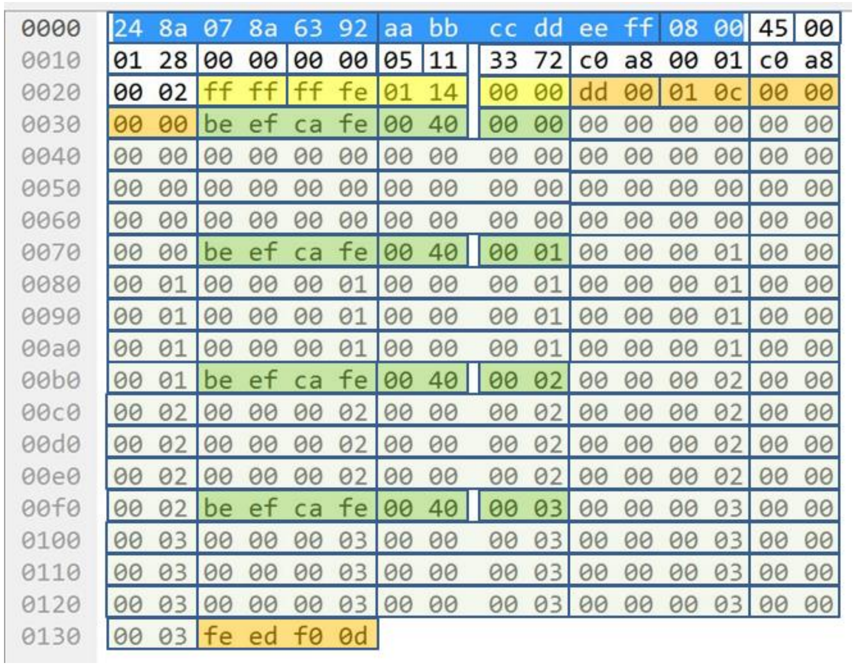
Data Payload



```
0000  24 8a 07 8a 63 92 aa bb  cc dd ee ff 08 00 45 00
0010  01 28 00 00 00 00 05 11  33 72 c0 a8 00 01 c0 a8
0020  00 02 ff ff ff fe 01 14  00 00 dd 00 01 0c 00 00
0030  00 00 be ef ca fe 01 00  00 00 00 00 00 02 00 00
0040  00 03 00 00 00 04 00 00  00 05 00 00 00 06 00 00
0050  00 07 00 00 00 08 00 00  00 09 00 00 00 0a 00 00
0060  00 0b 00 00 00 0c 00 00  00 0d 00 00 00 0e 00 00
0070  00 0f 00 00 00 10 00 00  00 11 00 00 00 12 00 00
0080  00 13 00 00 00 14 00 00  00 15 00 00 00 16 00 00
0090  00 17 00 00 00 18 00 00  00 19 00 00 00 1a 00 00
00a0  00 1b 00 00 00 1c 00 00  00 1d 00 00 00 1e 00 00
00b0  00 1f 00 00 00 20 00 00  00 21 00 00 00 22 00 00
00c0  00 23 00 00 00 24 00 00  00 25 00 00 00 26 00 00
00d0  00 27 00 00 00 28 00 00  00 29 00 00 00 2a 00 00
00e0  00 2b 00 00 00 2c 00 00  00 2d 00 00 00 2e 00 00
00f0  00 2f 00 00 00 30 00 00  00 31 00 00 00 32 00 00
0100  00 33 00 00 00 34 00 00  00 35 00 00 00 36 00 00
0110  00 37 00 00 00 38 00 00  00 39 00 00 00 3a 00 00
0120  00 3b 00 00 00 3c 00 00  00 3d 00 00 00 3e 00 00
0130  00 3f fe ed f0 0d
```

Packing Format Example 3

Ethernet Header	IP Header	UDP Header	Readout Header	Data Header 0	Data Payload 0
				Data Header 1	Data Payload 1
				Data Header 2	Data Payload 2
				Data Header 3	Data Payload 3



```
0000  24 8a 07 8a 63 92 aa bb  cc dd ee ff 08 00 45 00
0010  01 28 00 00 00 00 05 11  33 72 c0 a8 00 01 c0 a8
0020  00 02 ff ff ff fe 01 14  00 00 dd 00 01 0c 00 00
0030  00 00 be ef ca fe 00 40  00 00 00 00 00 00 00 00
0040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0070  00 00 be ef ca fe 00 40  00 01 00 00 00 01 00 00
0080  00 01 00 00 00 01 00 00  00 01 00 00 00 01 00 00
0090  00 01 00 00 00 01 00 00  00 01 00 00 00 01 00 00
00a0  00 01 00 00 00 01 00 00  00 01 00 00 00 01 00 00
00b0  00 01 be ef ca fe 00 40  00 02 00 00 00 02 00 00
00c0  00 02 00 00 00 02 00 00  00 02 00 00 00 02 00 00
00d0  00 02 00 00 00 02 00 00  00 02 00 00 00 02 00 00
00e0  00 02 00 00 00 02 00 00  00 02 00 00 00 02 00 00
00f0  00 02 be ef ca fe 00 40  00 03 00 00 00 03 00 00
0100  00 03 00 00 00 03 00 00  00 03 00 00 00 03 00 00
0110  00 03 00 00 00 03 00 00  00 03 00 00 00 03 00 00
0120  00 03 00 00 00 03 00 00  00 03 00 00 00 03 00 00
0130  00 03 fe ed f0 0d
```

Enhancements Since IKON13

- Duplicated packet engine logic for second QSFP+ – can now egress 200 Gbps over two links.
- Rewrote entire configuration firmware and software – fixed UART firmware bug, and provided a much more flexible and readable Python interface.
- The throughput of each engine can now be independently configured.

What Next?

- To be discussed!