

High speed debug with ngscopeclient

Andrew D. Zonenberg, Ph.D
Lead Developer, ngscopeclient



About Me

- Ph.D RPI 2015, OS/SoC security emphasis
- Embedded / semiconductor security by day
- Assorted OSS/OSH projects by night
- Lead dev of libscopehal, ngscopeclient, and ancestors

What is ngscopeclient?



- **GPU accelerated** (Vulkan) T&M UI
- Runs on Windows, Linux, and even MacOS
- Drivers for products from 25+ vendors
- Online (remote control) or offline analysis capable
- **Free and open source** (3-clause BSD license)
  - AI-free, hand crafted by and for human experts
- Concept: “GNU Radio for T&M”

Why ngscopeclient?

- Speed
- Extensibility
- Interoperability

Speed



Scopes are too slow!

- I don’t mean bandwidth or sample rate
- Protocol decodes, math, measurements, etc degrade WFM/s
- More WFM/s means
  - Better developer experience
  - Faster iteration
  - Less dead time, easier to find rare events
- No name-and-shame. ***This is an industry wide problem***

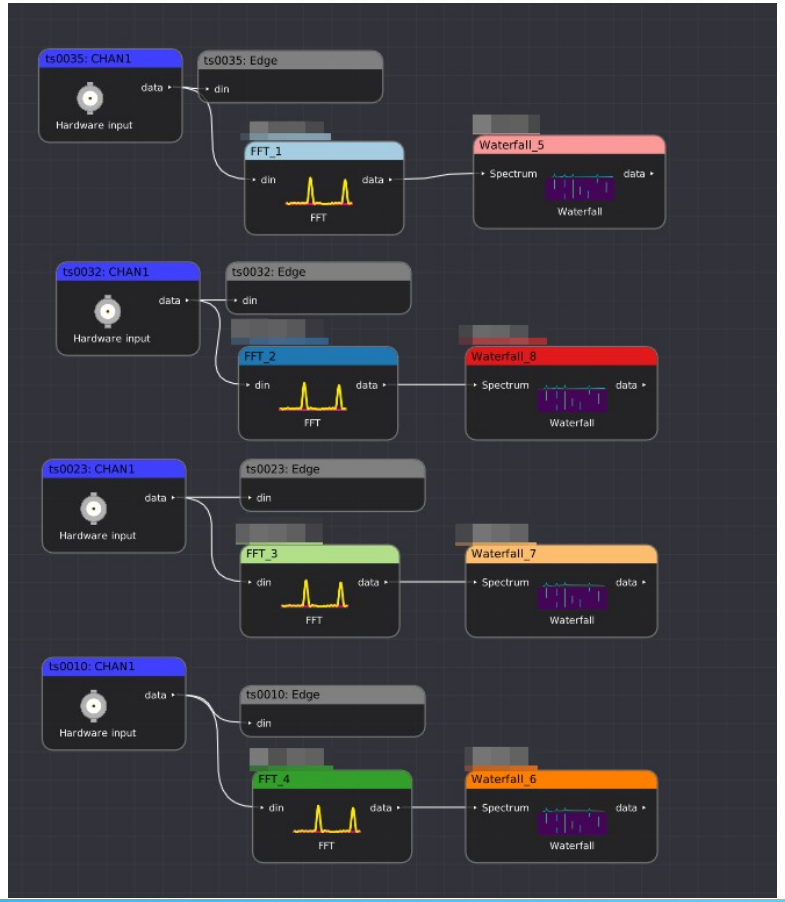


Scopes are too slow!

- Vendors today spec WFM/s in a strange way
  - Minimum memory depth
  - Sequence mode
  - Bound by trigger re-arm time
- Very few use cases care about this spec
- Users care about overall responsiveness with actual workloads
  - How many FFTs, eye patterns, 8b10b decodes, etc per unit time
  - At realistic memory depths, not e.g. 1K points

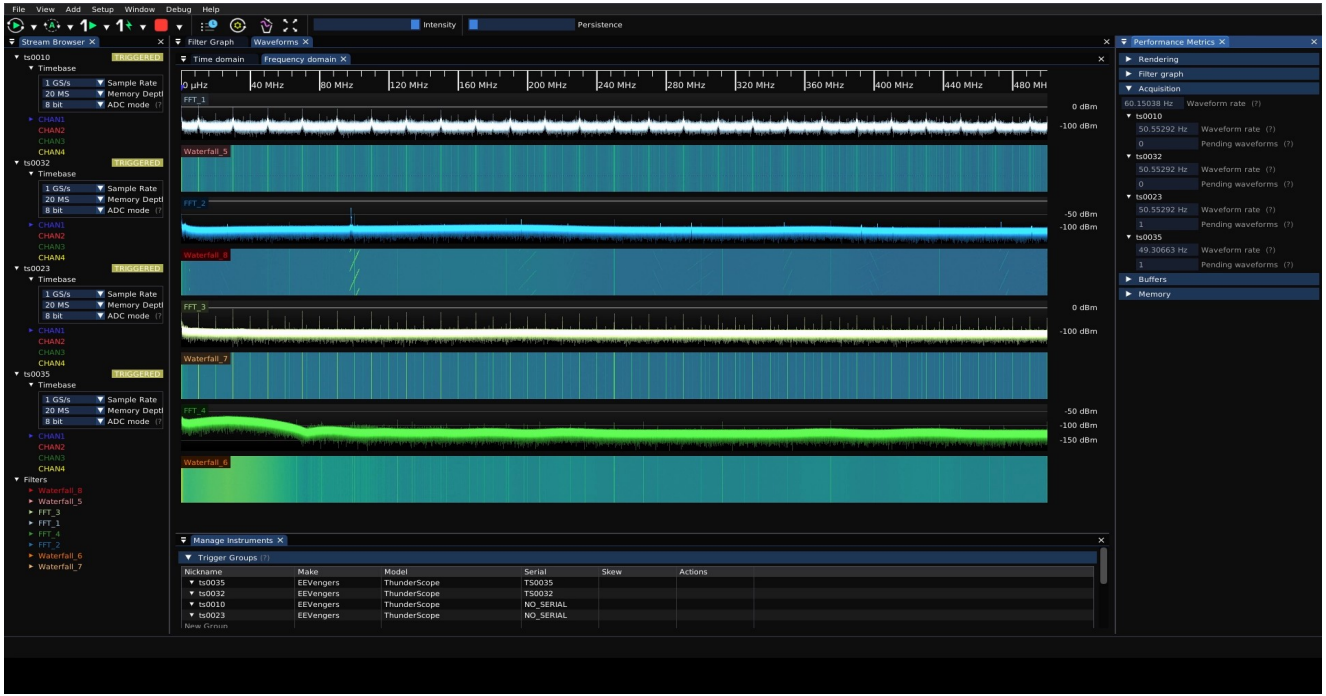
Challenge for scope vendors

- Any scope in your portfolio
- Here’s the setup
  - 4 channels, 20M points
  - 1 GS/s
  - FFT (Blackman window)
  - Waterfall
- How fast can you update?



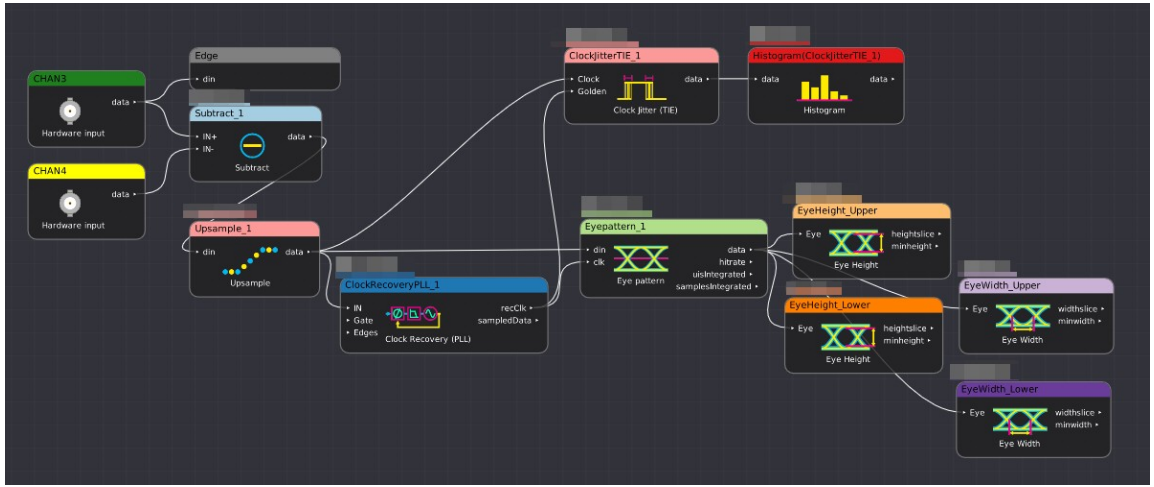
I’ll go first

- 4 ThunderScopes
- 50 WFM/s
- 4 Gsps real time
- 200x 20M point FFT/sec



Let’s try something else

- Differential PAM-3 / MLT-3
- Your choice of line rate
- 10M point memory depth
- CDR PLL, any loop filter
- Eye pattern
- Eye width + height measurements
- TIE track + histogram



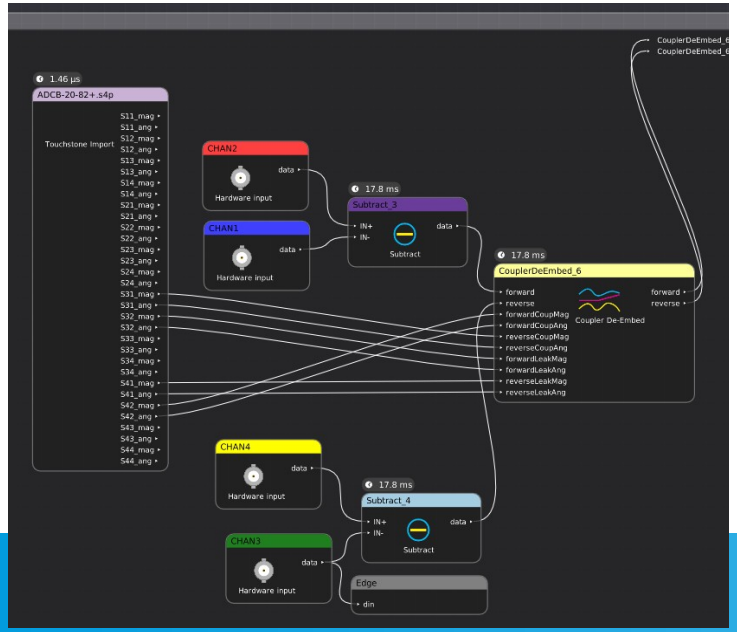
Real time again

- ThunderScope
- 50 WFM/s
- 2x 500 Msps



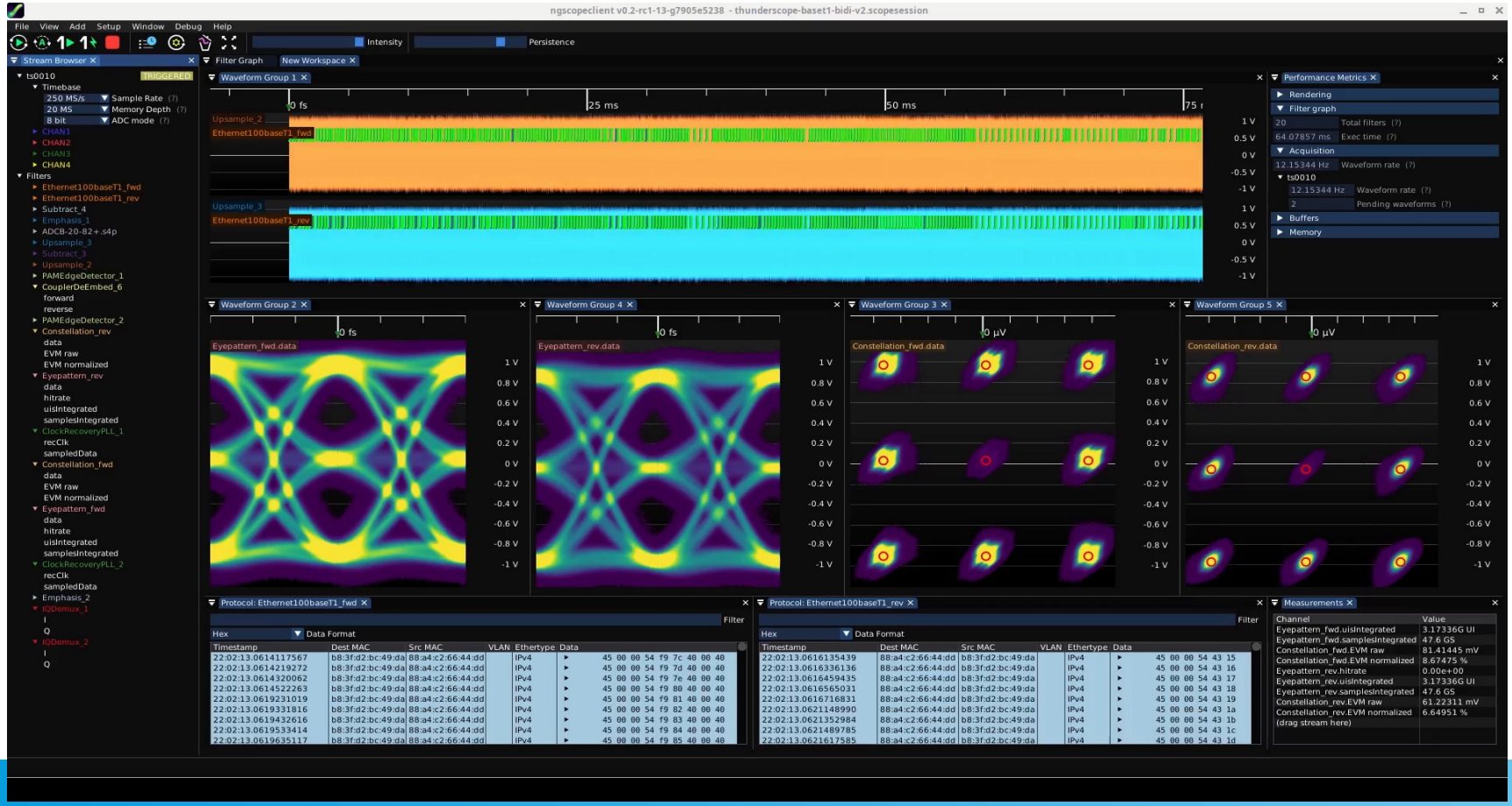
Let’s turn it up a notch

- 100baseT1 automotive Ethernet
- 250 Msps on 4 channels, 20M point memory depth
- De-embed coupler, eye pattern, constellation, Ethernet decode





Still (barely) real time





What's the secret?

- Teraflops of shader cores on the GPU
- If you don't use them you're missing out

Render Config	
Shading Units:	4096
TMUs:	256
ROPs:	128
Compute Units:	64
Matrix Cores:	128
RT Cores:	64
L0 Cache:	32 KB per WGP
L2 Cache:	8 MB
L3 Cache:	64 MB

Theoretical Performance	
Pixel Rate:	373.8 GPixel/s
Texture Rate:	747.5 GTexel/s
FP16 (half):	95.68 TFLOPS (2:1)
FP32 (float):	47.84 TFLOPS
FP64 (double):	1,495.0 GFLOPS (1:32)

https://www.techpowerup.com/gpu-specs/radeon-ai-pro-r9700.c4290

```
54  #define Y_SIZE 64
55  layout(local_size_x=8, local_size_y=Y_SIZE, local_size_z=1) in;
56
57  void main()
58  {
59      //Divide work up into blocks
60      uint numThreads = gl_NumWorkGroups.x * gl_WorkGroupSize.x;
61      uint nthread = gl_GlobalInvocationID.x;
62      uint samplesPerThread = size / numThreads;
63
64      //Find our thread's block
65      uint istart = nthread * samplesPerThread;
66      uint iend = istart + samplesPerThread;
67      if(nthread == numThreads - 1)
68          iend = size;
69
70      float delta = nmax - nmin;
71      float fbins = float(bins);
72      for(uint i=istart+1+gl_LocalInvocationID.y; i < iend; i += Y_SIZE)
73      {
74          float fbin = (pin[i] - nmin) / delta;
75
76          uint bin = uint(floor(fbin * fbins));
77          if(fbin < 0)
78              bin = 0;
79          else
80              bin = min(bin, bins - 1);
81
82          atomicAdd(pout[bin], 1);
83      }
84  }
```



PC streaming B/W is a limiting factor

- Headless scopes are in the lead
- Actual achieved performance (not theoretical):
  - ThunderScope: 8 Gb/s (1 Gsps @ 8 bit) ***real time, full ADC rate***
  - PicoScope 6000E: 2.7 Gb/s (170 Msps @ 12 bit padded to 16)

Next-gen hardware will be much faster

- ThunderScope Pro (preliminary):
  - 38.4 Gb/s (2x 1.6 Gsps 12 bit) ***real time, full ADC rate***
  - 48 Gb/s (4x 1 Gsps 12 bit) ***limited by PCIe gen4 x4***
  - 51.2 Gb/s (4x 1.6 Gsps 8 bit) ***real time, full ADC rate, low precision***

PC streaming B/W is a limiting factor

- Benchtop scopes all < 1 Gbps, most << 1 Gbps
  - Best I’ve seen is 829 Mbps (51 Msps @ 16 bit)
- Some $50k+ scopes barely break 100 Mbps to PC
  - SCPI API architecture and design matter a LOT
  - Vendors - talk to me if you want advice on performance / API design



PC streaming B/W is a limiting factor

- Both O(1) and O(n) scaling terms matter
- Many scopes from budget vendors have decent O(n) scaling
  - Deep memory isn’t *that* much slower than shallow
  - Doesn’t help if you take a few hundred ms to download **any** data



PC streaming B/W is a limiting factor

- Streaming B/W is a hard number to find
  - Usually need to measure yourself
  - Pico is the only vendor I see making **any** claims in datasheets
- If you’re shopping for scopes and care about this, good luck
- **I want vendors to advertise and compete on this spec!**



API Wishlist

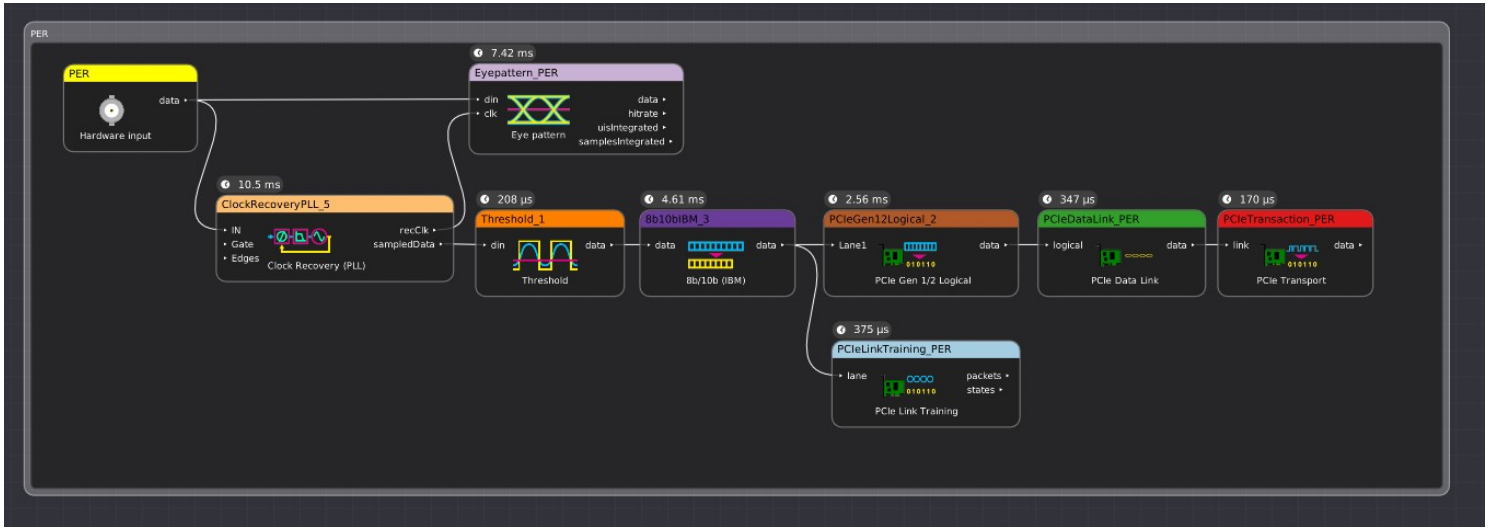
- Push notifications when user adjusts front panel controls
  - Current: cache clientside and need explicit flush
- Interlocked/async push of waveform data
  - Current: single shot trigger, poll for ready, read data, re-arm



Extensibility

Filter Graph Workflow

- Connect (typed) data streams between processing blocks
- Configure each block
- If you've used GNU Radio you'll find it familiar





Flexible Implementation

- Waveform data can live in RAM or VRAM, sync as needed
- Filter blocks can be C++ on CPU, shaders on GPU, or a mix
  - Write a slow version first, then accelerate as needed

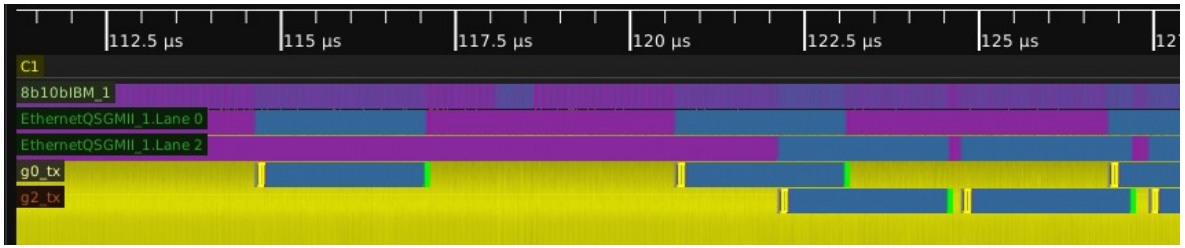


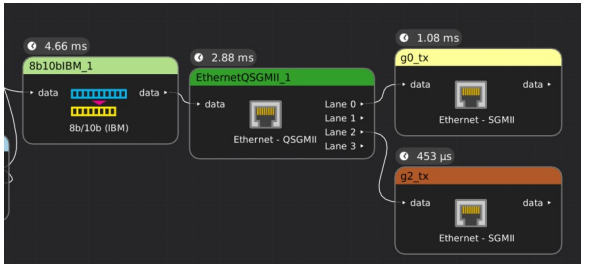
Debug Challenge

- FPGA based Ethernet switch
- QSGMII 10/100/1000 baseT PHY
  - Quad SGMII
  - 4 gigabit Ethernet lanes over one 5 Gbps SERDES
- How to debug and instrument this?
  - You cannot buy a QSGMII protocol decode **from any vendor**
- SOTA: dump waveforms from scope, or FPGA over JTAG
  - Write your own slow postprocessing in python etc

Easy Extensibility

- Built QSGMII filter block in a few hours, ~100 lines of C++
- Demux single 8B/10B stream to four sub-rate streams
- Seamless integration with existing 8B/10B and SGMII decodes







Plugin capability

- Instrument drivers, filters, transports can be plugins
- “Linux kernel” model for v0.x series - no ABI stability
  - Plugins can be distributed as blobs **but only for a specific commit**
  - Updating libscopehal or ngscopeclient requires a recompile
- Aiming for more stable API/ABI in v1.0 (no ETA)
  - Rapid development and refactoring still ongoing



Case study: Encrypted SAE J1939

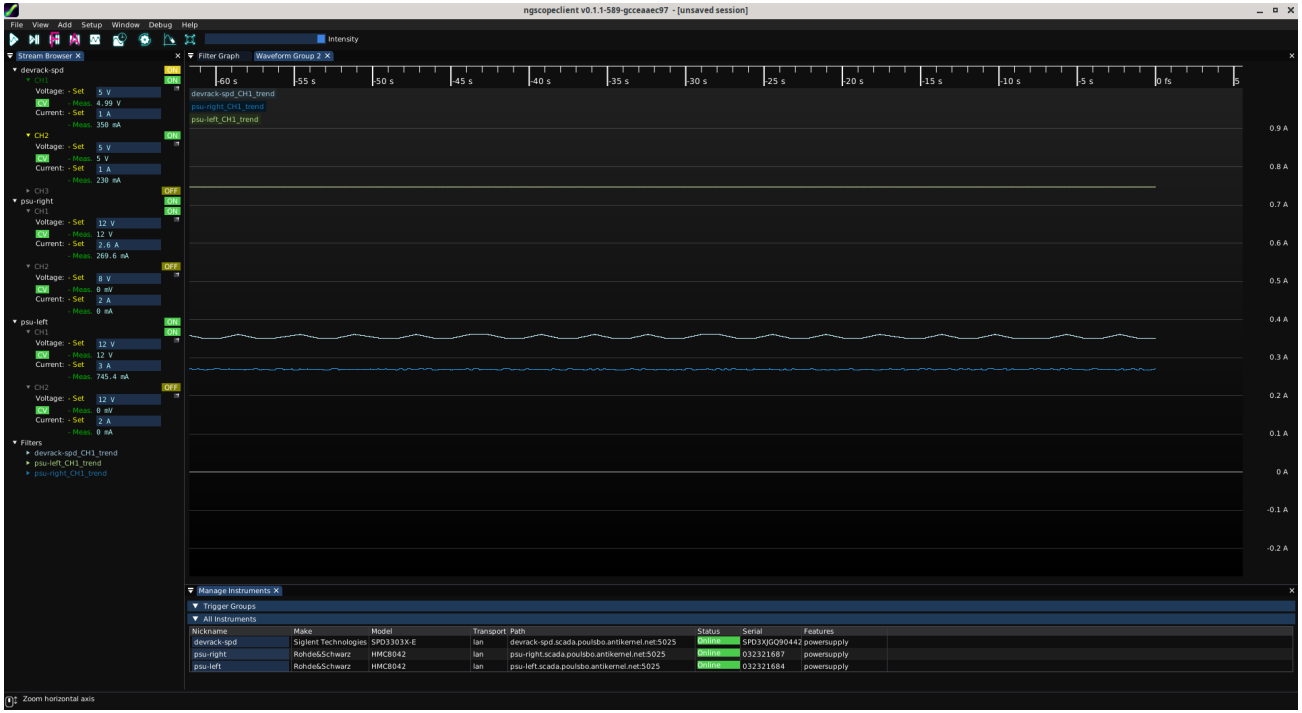
- Security assessment of heavy vehicle
- J1939 messages were encrypted
- Reverse engineered firmware, extracted keys
- Wrote decryption filter that outputs cleartext J1939 PDUs
- Building filter as plugin meant no need to fork upstream
  - Easy to make internal library of proprietary decodes
  - Can still run upstream ngscopeclient + contribute fixes back



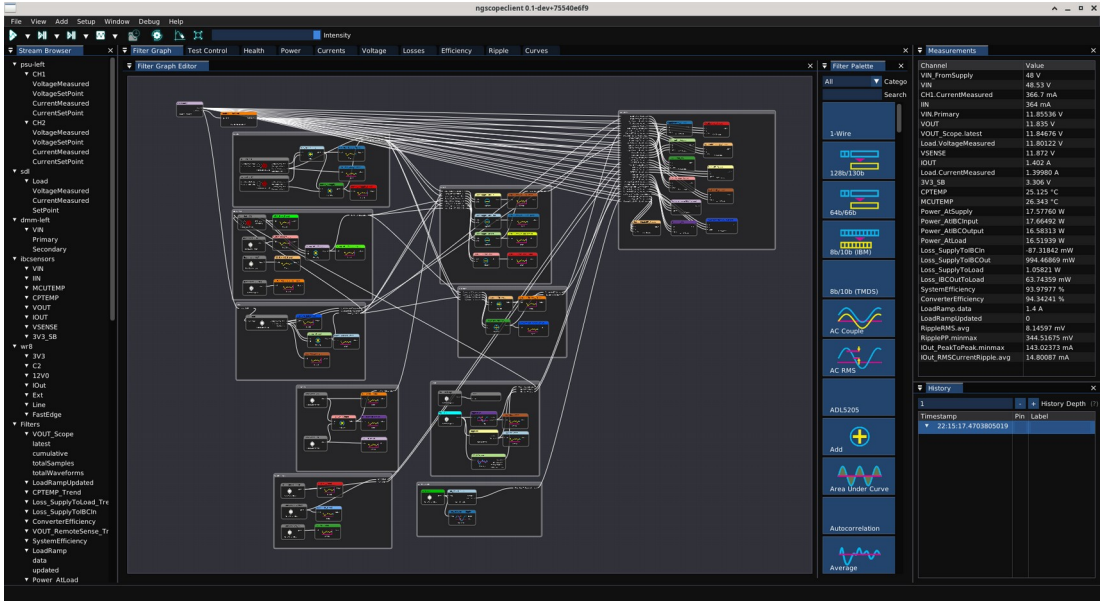
Interoperability

Multi Device Integration

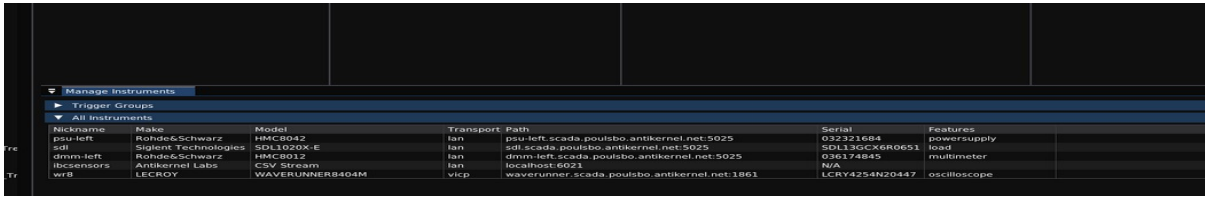
- Simple example: Graphing load from many PSUs



48V IBC characterization

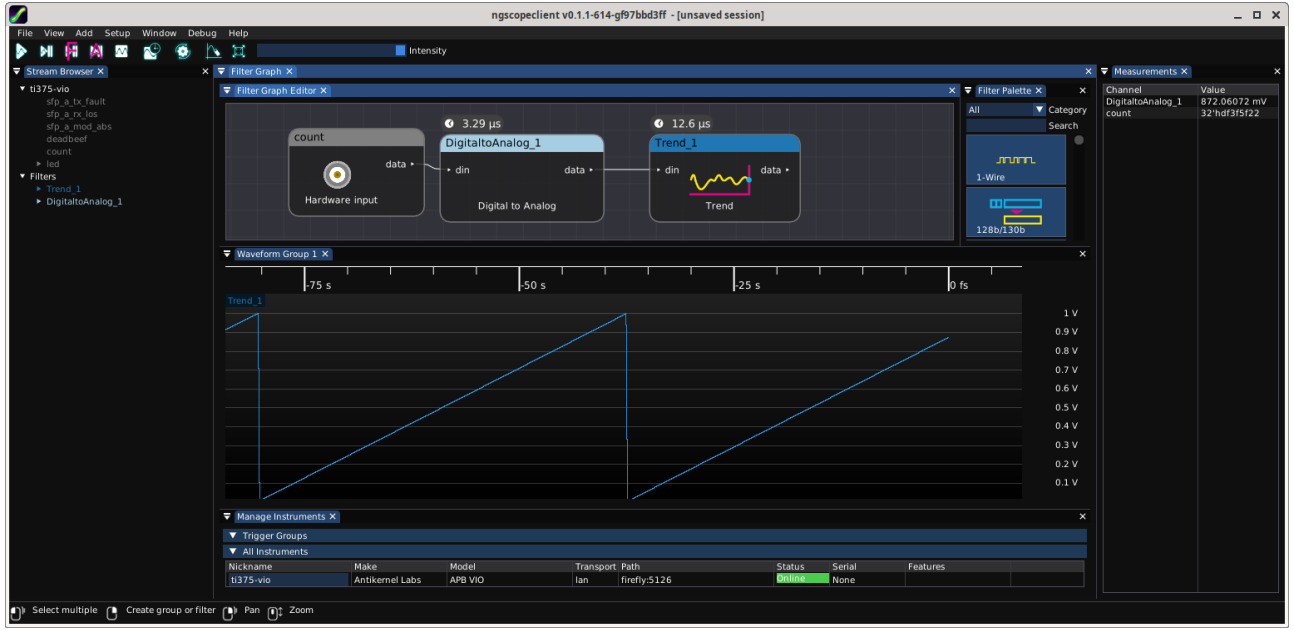






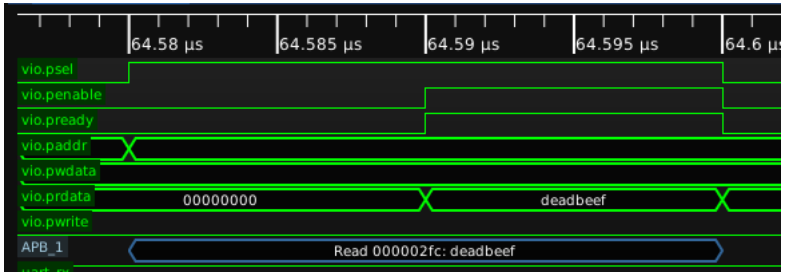
Oddball instruments

- FPGA virtual I/O IP



Internal FPGA debug

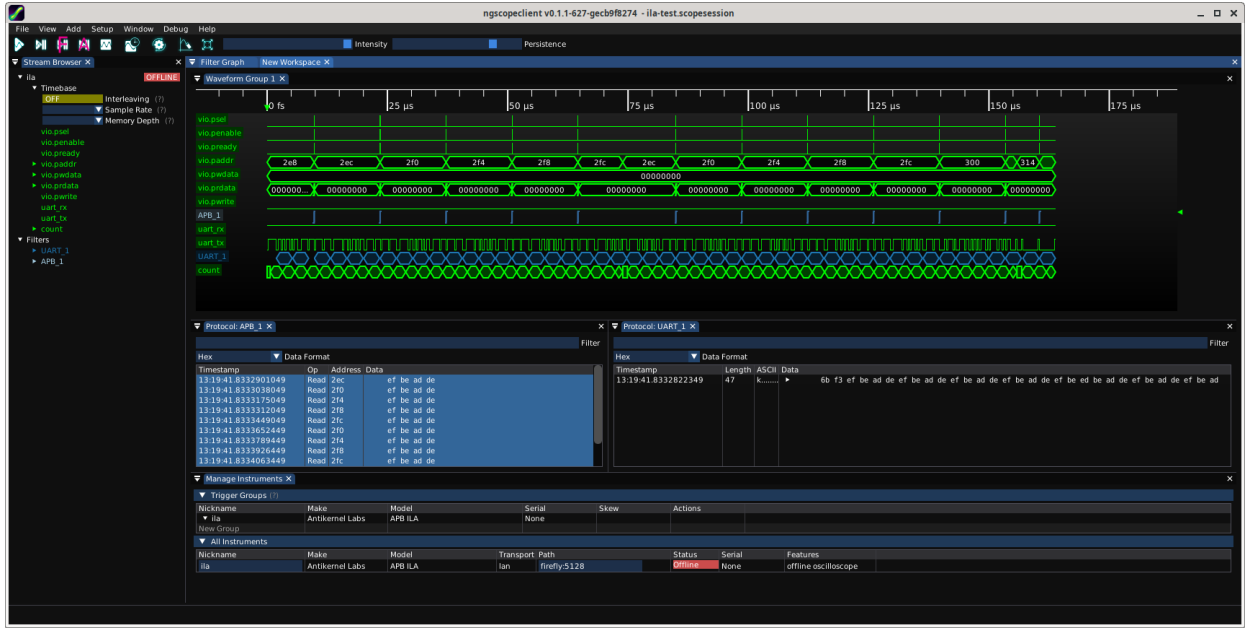
- Open FPGA ILA IP
- Decode for AMBA APB



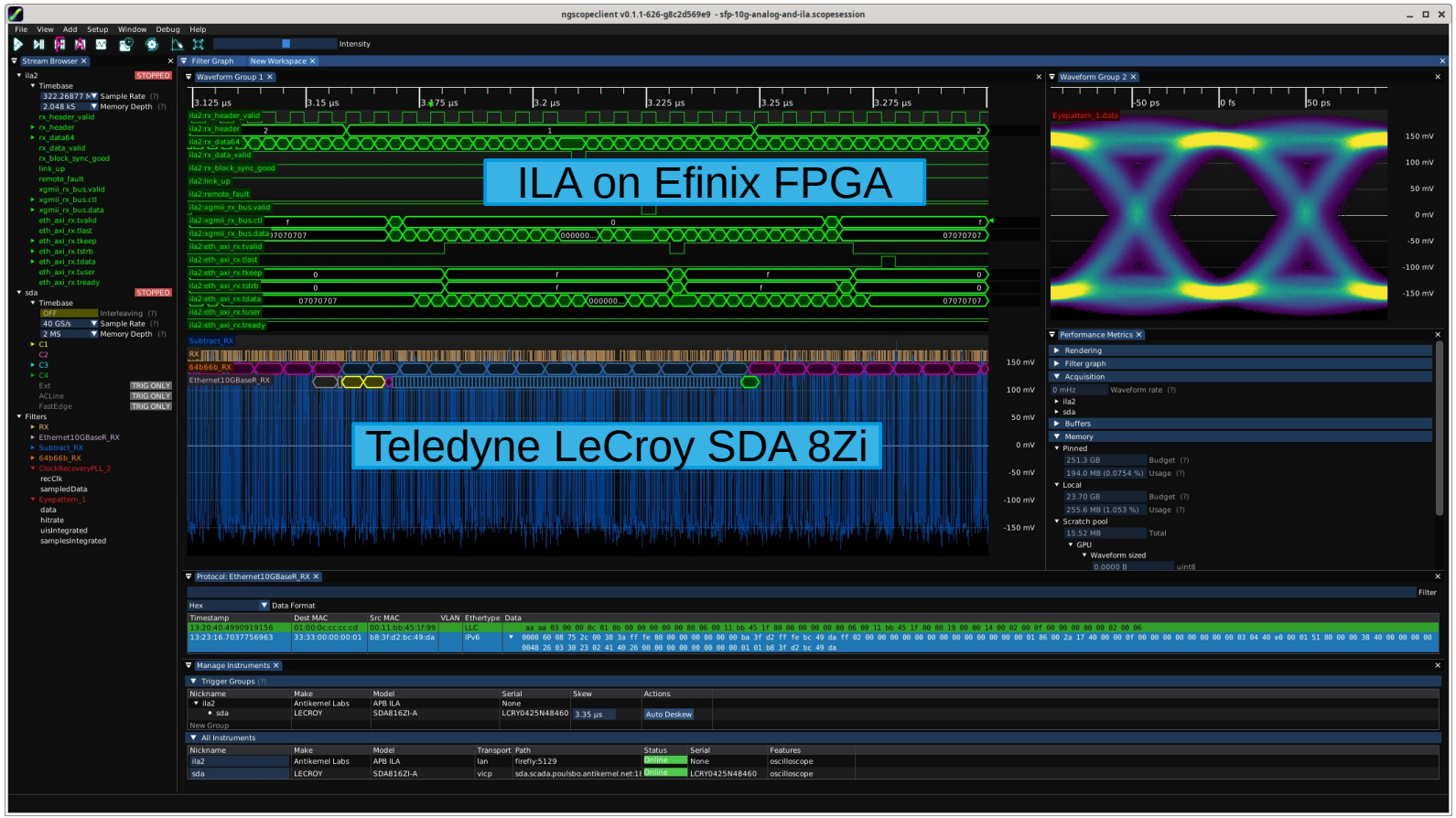
Protocol: APB_1

Hex | Data Format

Timestamp	Op	Address	Data
13:19:41.8333312049	Read	2f8	ef be ad de
13:19:41.8333449049	Read	2fc	ef be ad de
13:19:41.8333652449	Read	2f0	ef be ad de
13:19:41.8333789449	Read	2f4	ef be ad de
13:19:41.8333926449	Read	2f8	ef be ad de
13:19:41.8334063449	Read	2fc	ef be ad de
13:19:41.8334200449	Read	300	74 6e 75 6f
13:19:41.8334337449	Read	304	63 00 00 00
13:19:41.8334408049	Read	318	00 00 00 00

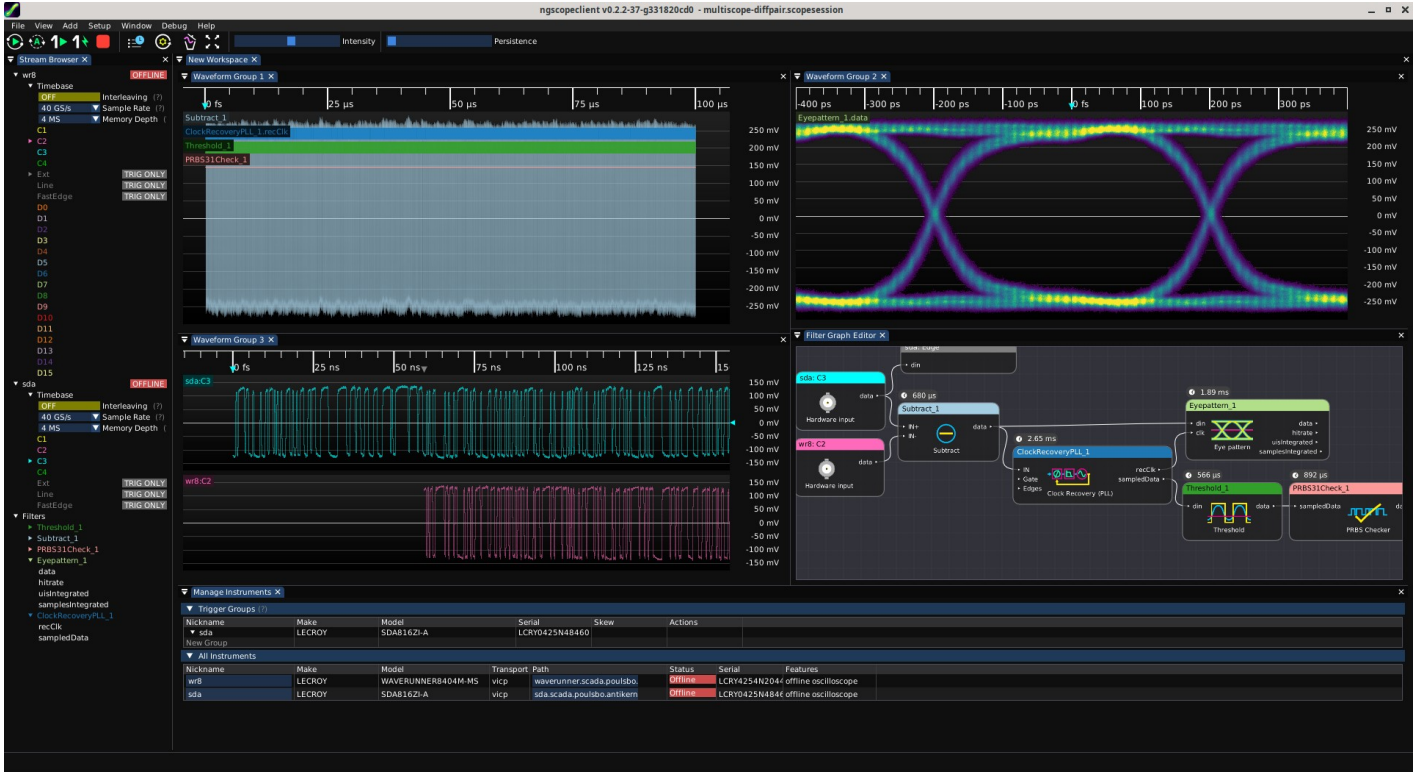


FPGA mixed-domain debug

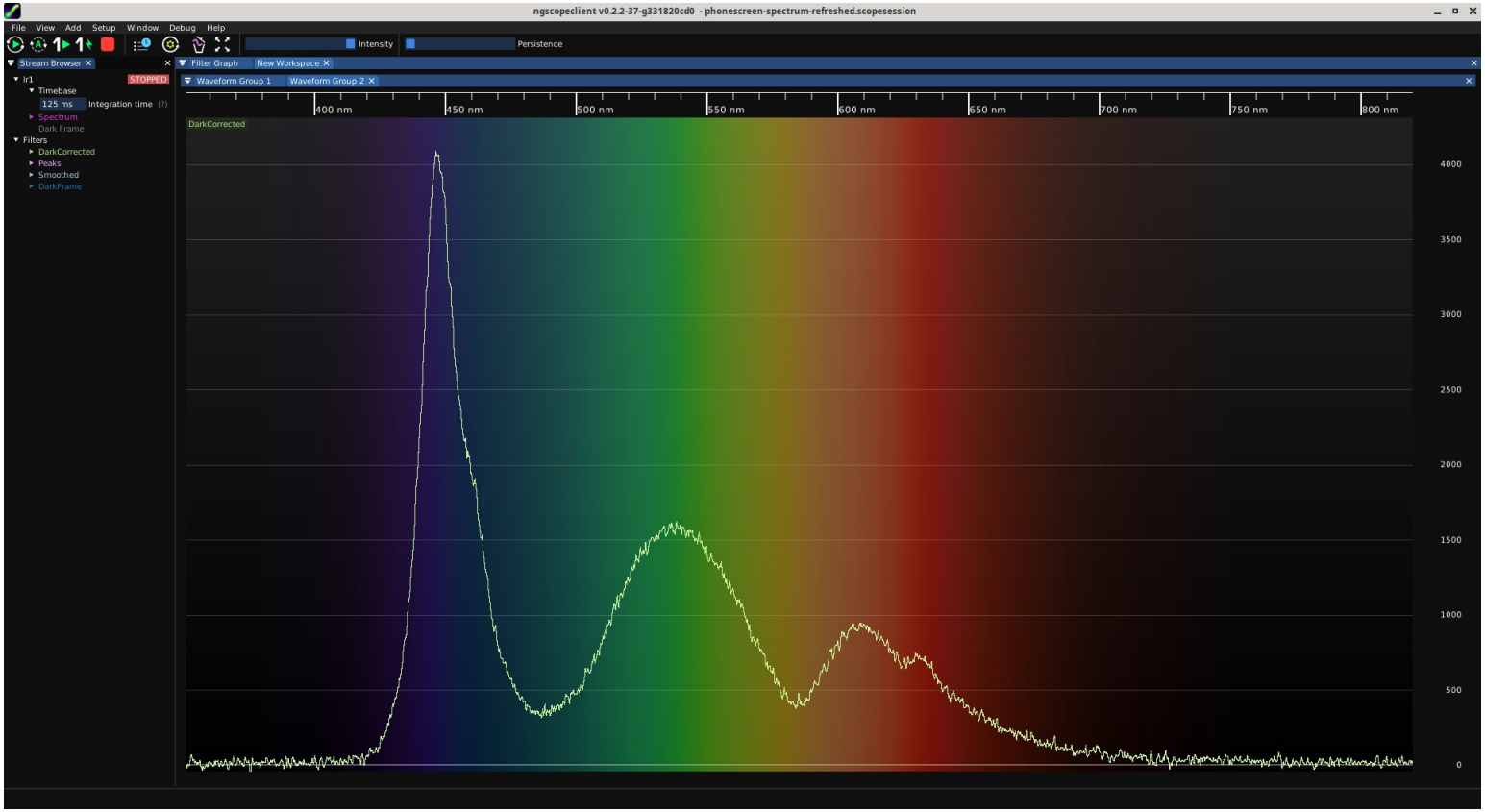


Automatic cross-trigger deskew

- Feed PRBS into both scopes
- Cross correlation
- Find peak



Weirder instruments work too





Acknowledgements

- Code contributions from 90+ people around the globe
- Hardware and financial support
  - EEVengers
  - Entropic Engineering
  - Pico Technology



Questions?