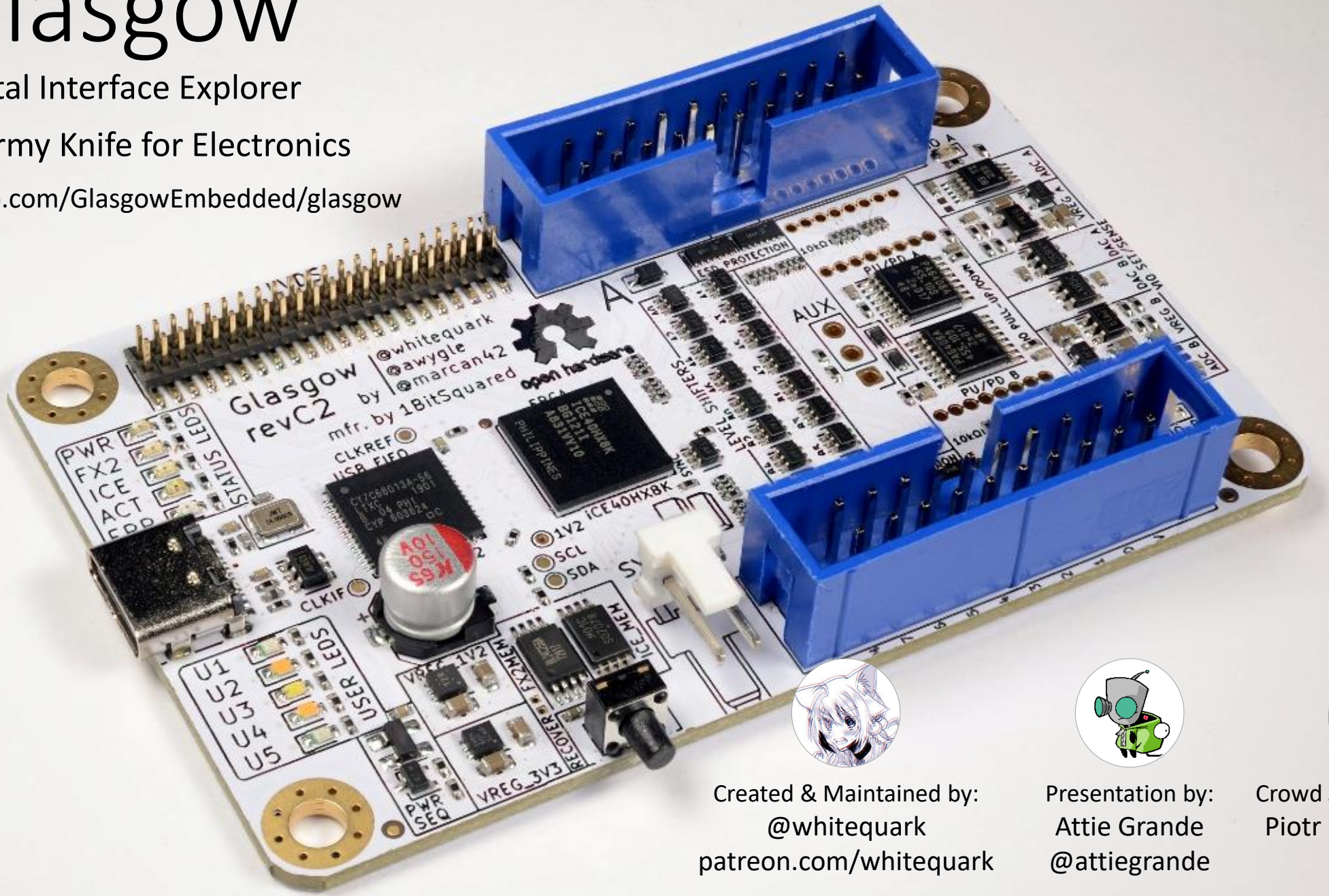
Glasgow

Digital Interface Explorer

Scots Army Knife for Electronics

https://github.com/GlasgowEmbedded/glasgow



Created & Maintained by:
@whitequark
patreon.com/whitequark

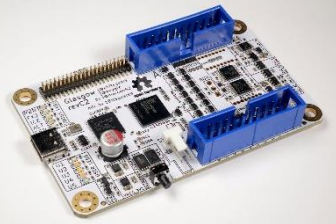
Presentation by:
Attie Grande
@attiegrande

Crowd Supply & DFM by:
Piotr Esden-Tempski
@esden

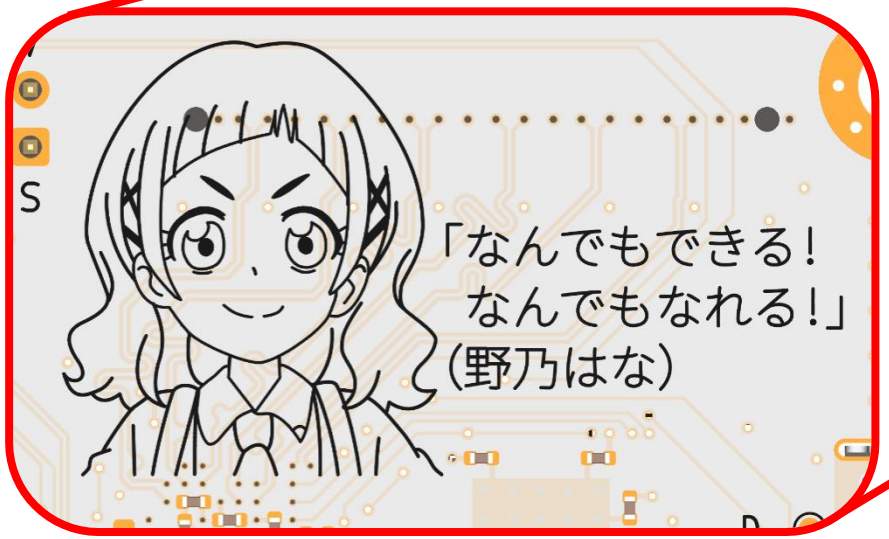
Overview

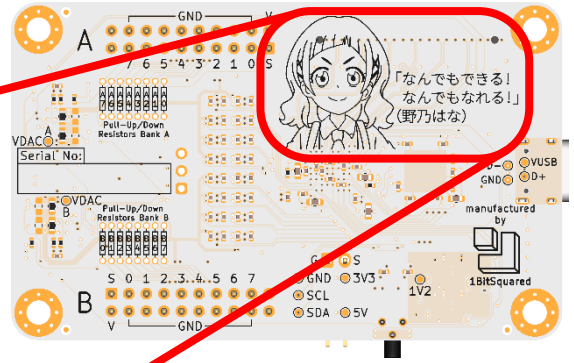
- What's the Concept, *"What can I do?"*
- About the I/O
- Interfaces Between Software and Gateware
- Anatomy of an Applet
- Deep Dive: Data Path & Buffer Management
- Future Plans
- Crowd Supply Campaign & DFM (Piotr / @esden)
- Questions!

What's the Concept?

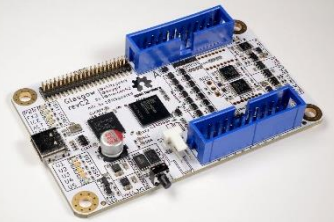


- Nono Hana from Pretty Cure
- Quote can be read as
  " [Glasgow] can do anything!
  [Glasgow] can be anything! "

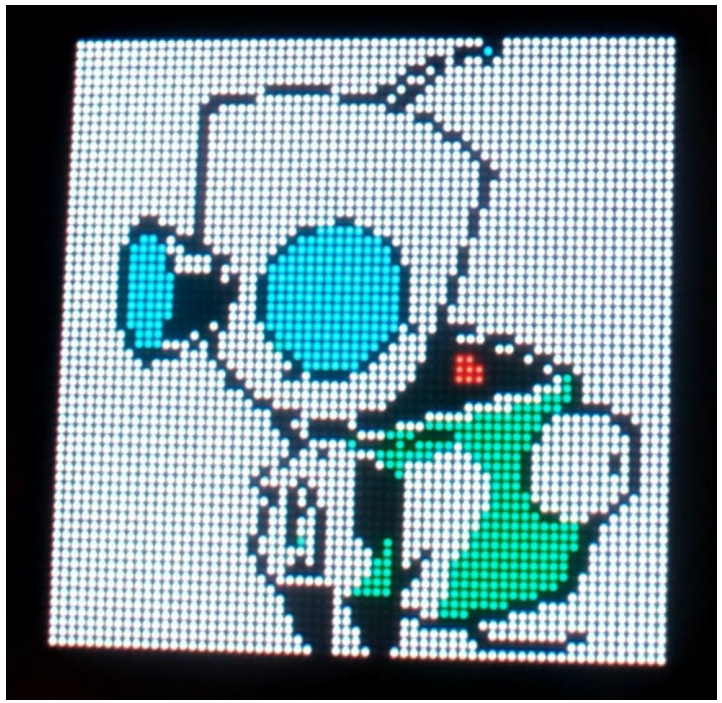




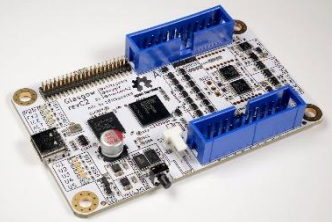
What's the Concept?



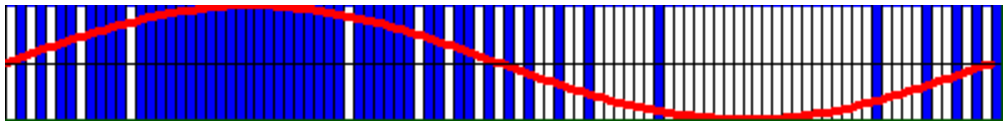
- FPGA permits real time signalling and I/O
- e.g: WS2812 LEDs (right)
- e.g: HUB75 with 64x64 framebuffer (below, WIP)

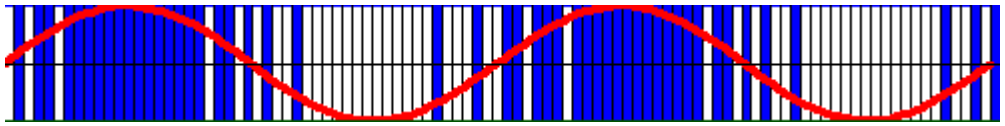


What's the Concept?

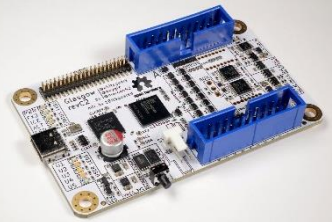


- Connect a digital output directly to a speaker
- Socket / FIFO interface between the two
- Sigma Delta DAC in FPGA
- "High" quality audio output
- Sounds surprisingly good!
- Zero extra components required





What's the Concept?

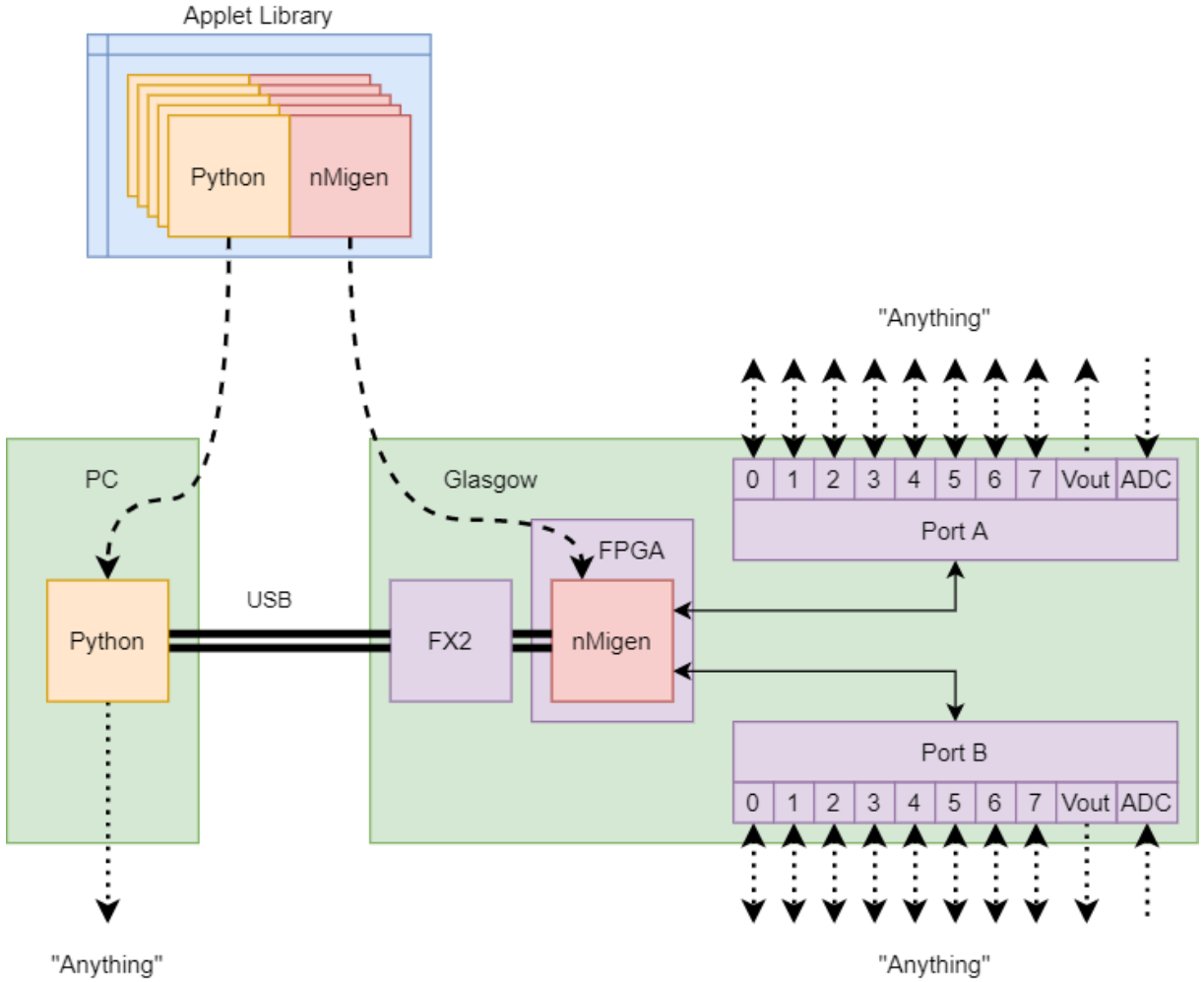


- JTAG / SPI / I2C interface
- VGA test pattern
- Parallel RGB capture
- Automatically detect JTAG pinout
- UART, e.g: unknown voltage and baudrate
- Many other applets already exist!

- Have an uncommon / proprietary device?
  - Connect it up and write an applet!
  - A PHY isn't a hard barrier
    (e.g: I'm working on a CAN add-on)

What's the Concept?

- Designed for simplicity and robustness
- Turnkey Setup
- No requirement to learn Python / nMigen
- Easily connect to many digital interfaces
- Applets written in Python and nMigen
  - Big library of existing applets
  - Python runs on the computer
  - nMigen runs on the FPGA
- Open Source FPGA toolchain
  - Very quick, you'll rarely wait for a build



What's the Concept?

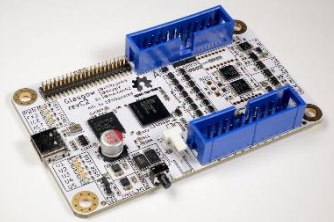
- SCD30 – CO2, Temperature and Humidity
- Getting started:
  1. Connect 4x wires
  2. Run the applet
  3. … get sensor readings

What's the Concept?

- SCD30 – CO2, Temperature and Humidity
- Getting started:
  1. Connect 4x wires
  2. Run the applet
  3. … get sensor readings
  4. … tweak the command line
     --> log to InfluxDB

influxdb

What's the Concept?

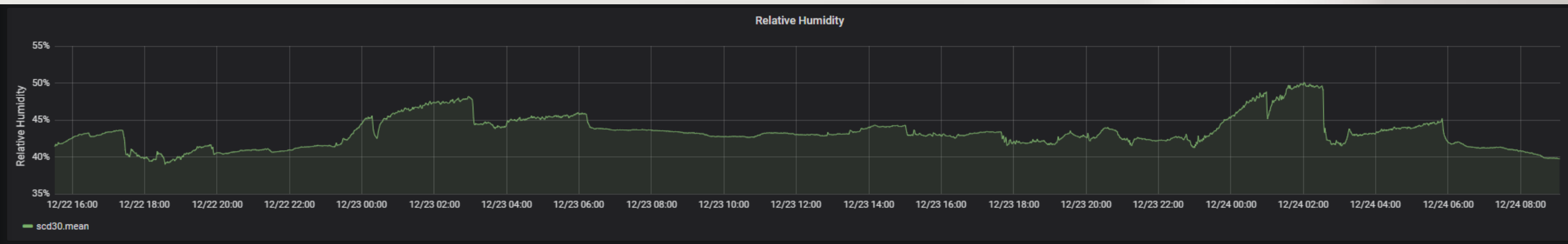


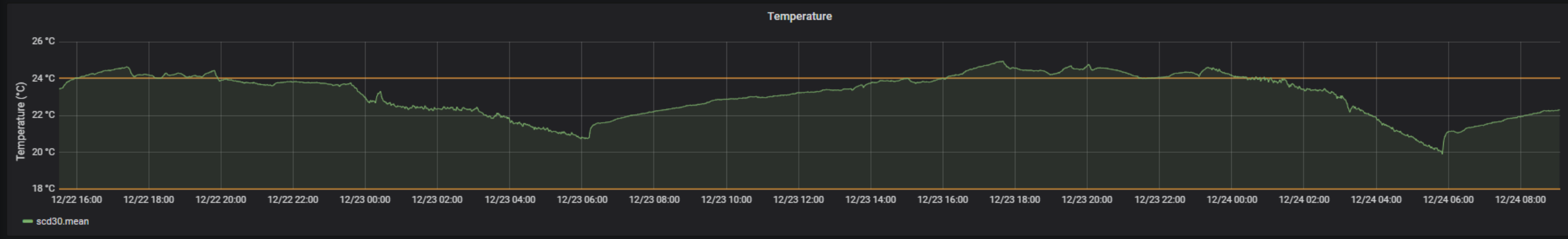
- SCD30 – CO2, Temperature and Humidity
- Getting started:
  1. Connect 4x wires
  2. Run the applet
  3. … get sensor readings
  4. … tweak the command line
     --> log to InfluxDB
  5. … connect to Grafana …

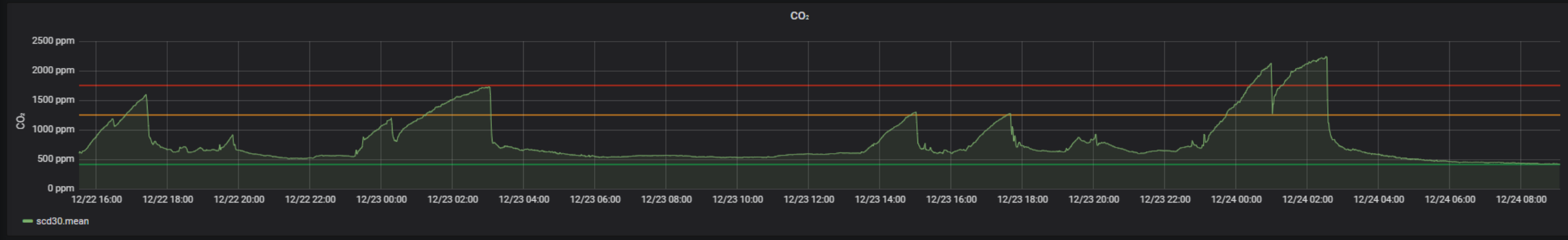
Grafana

influxdb

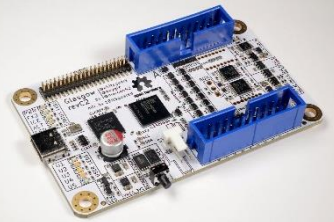
What’s the Concept?



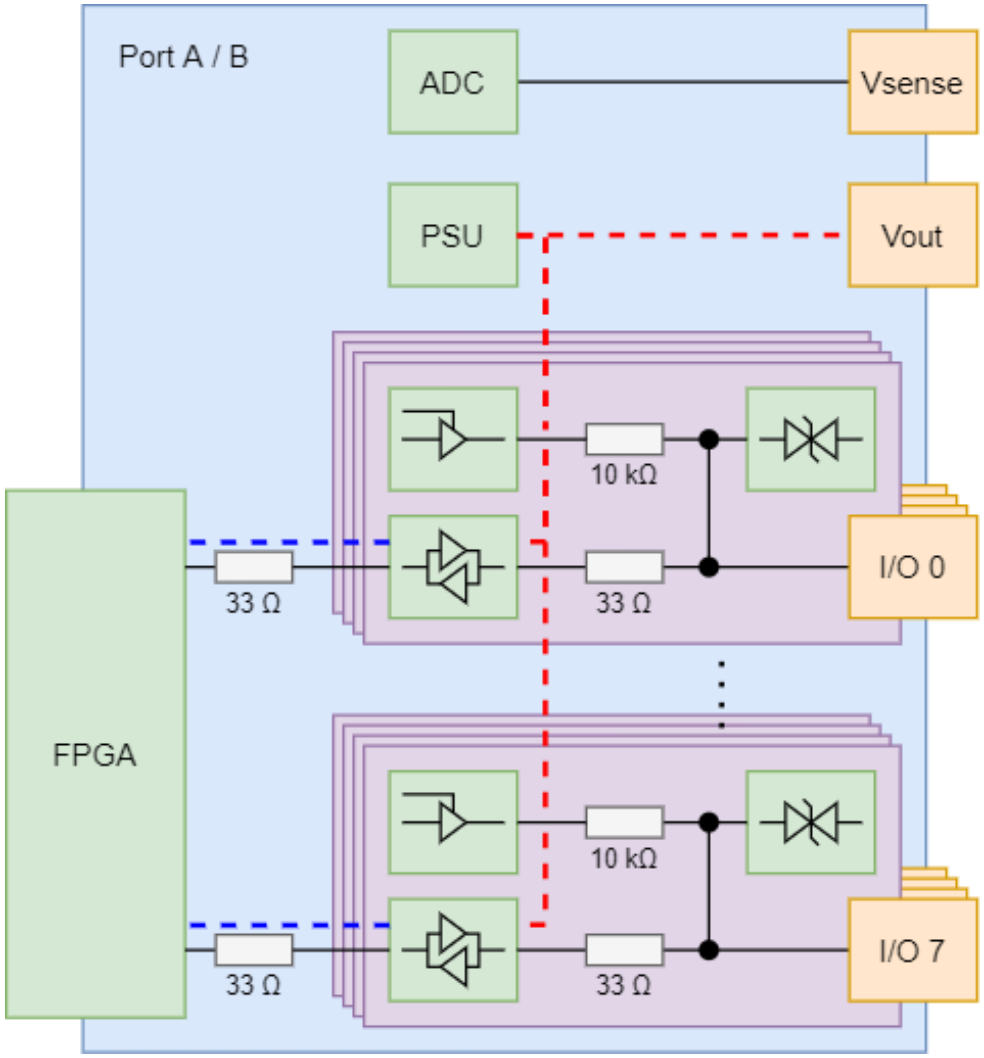




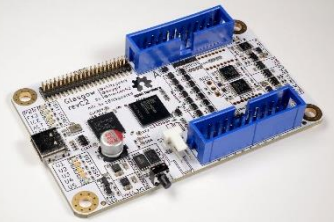
About the I/O



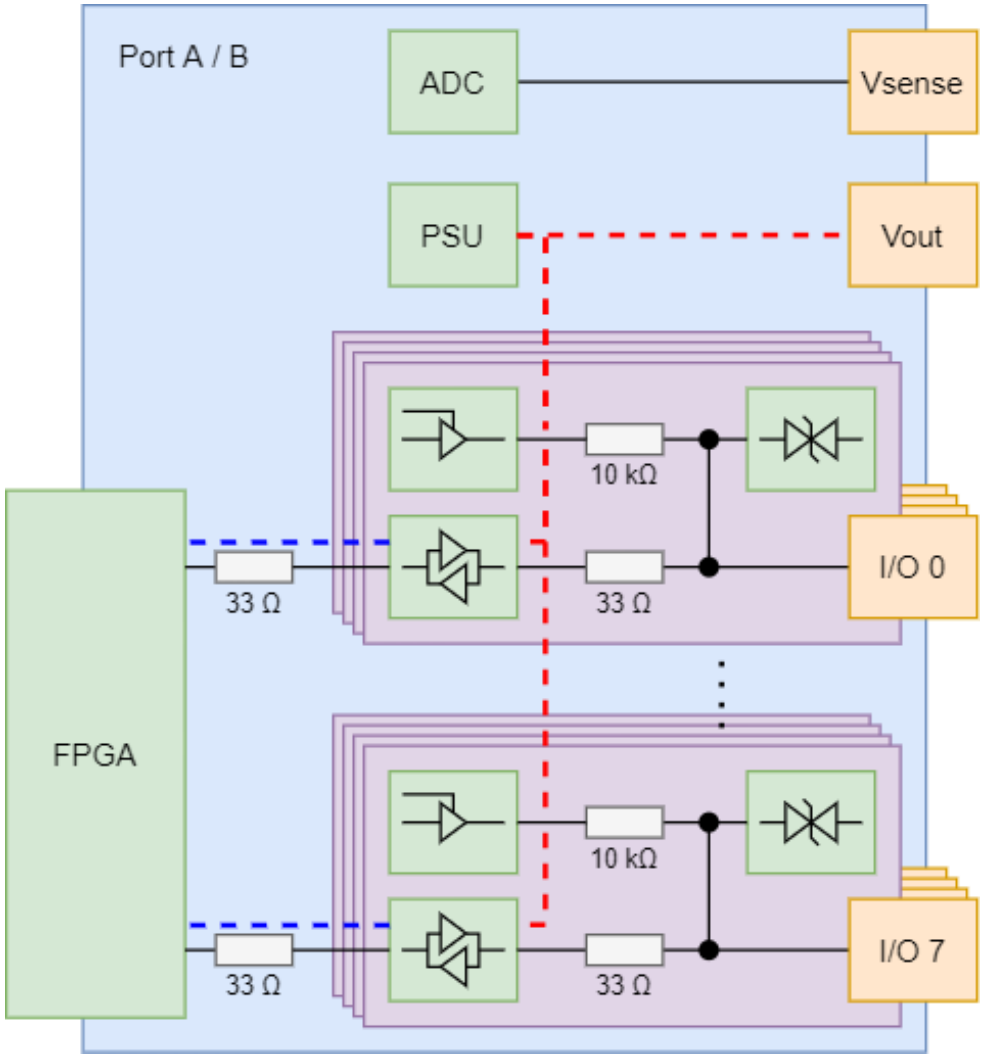
- 16x digital I/O pins, in 2x ports
- Can power & interface with many things *<u>without</u>* additional circuitry
- Care-free hookup
- Converts a hardware problem into a software problem
- You should never wonder:
  - *“do I trust Glasgow right now?...”*



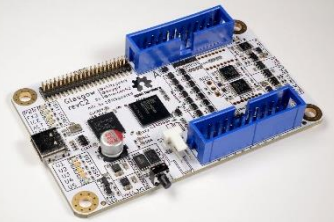
About the I/O



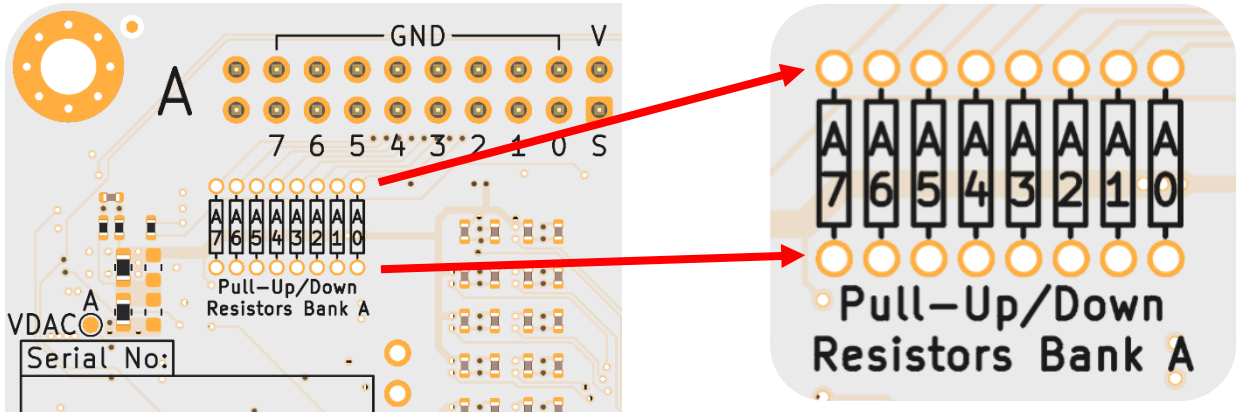
- Each pin has:
  - I/O buffers / level shifters
  - Bi-direction (*not* auto-dir)
  - ~100 MHz signalling
  - Individual 10 kΩ pull up / down
  - ESD protection
  - Infinite short circuit
  - Can do things like Open Drain!

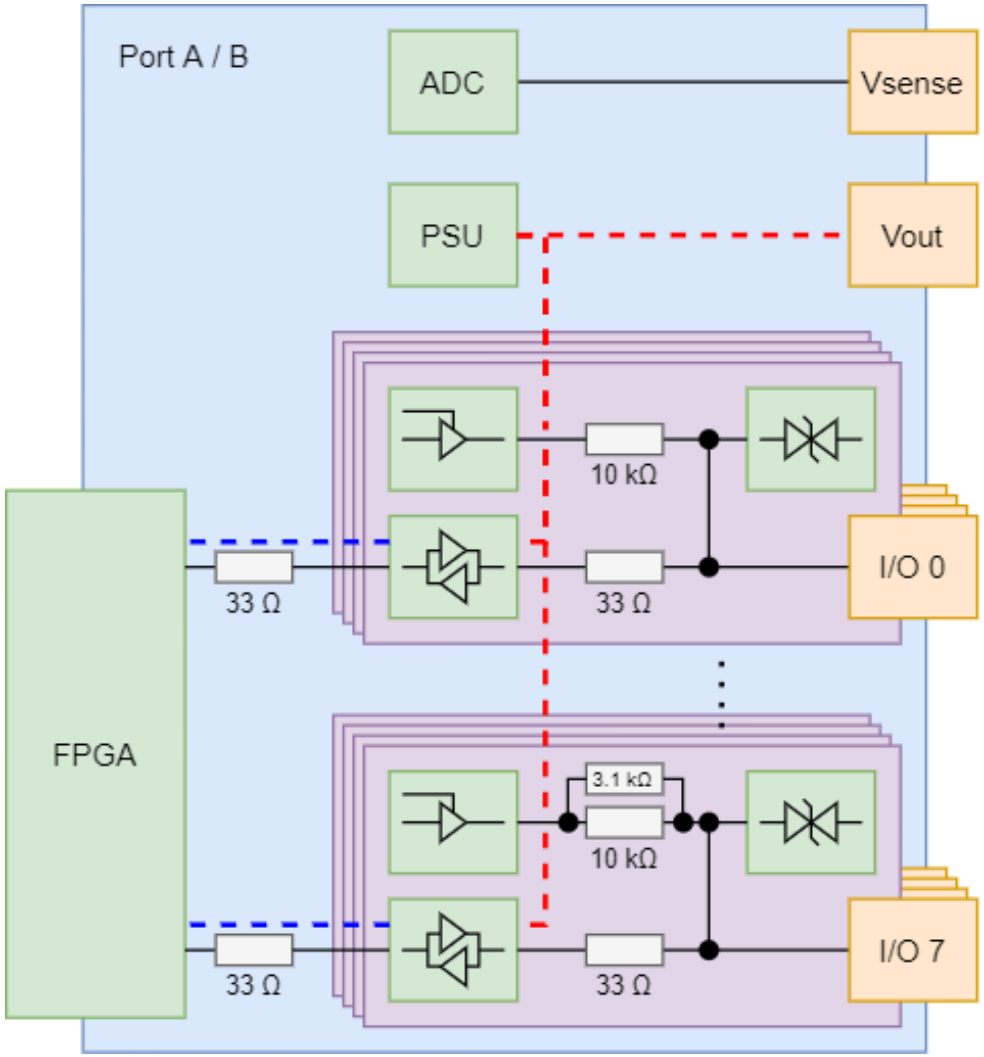


About the I/O

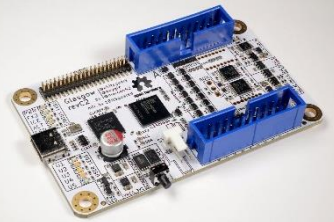


- Each pin has:
  - Onboard pull resistors permit generic termination
  - Can add easily another through-hole resistor in parallel using vias

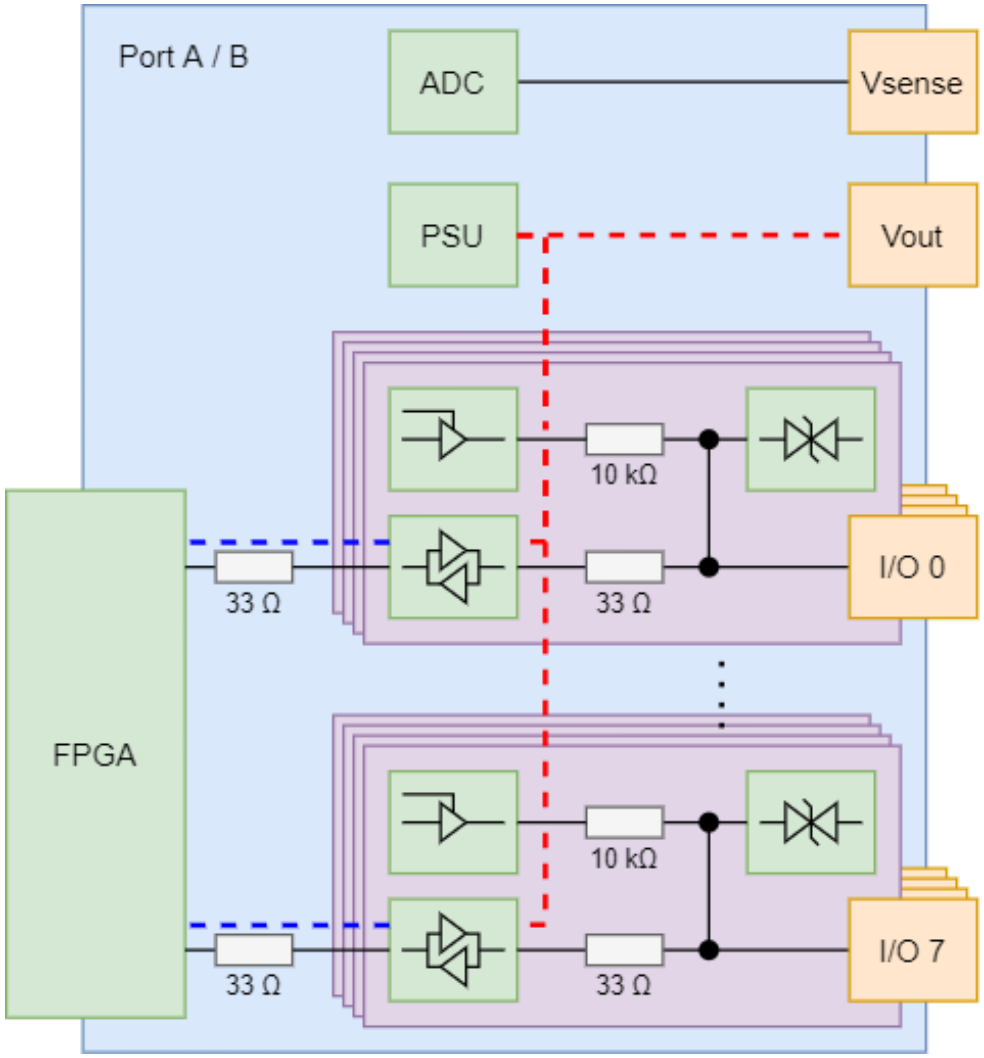




About the I/O

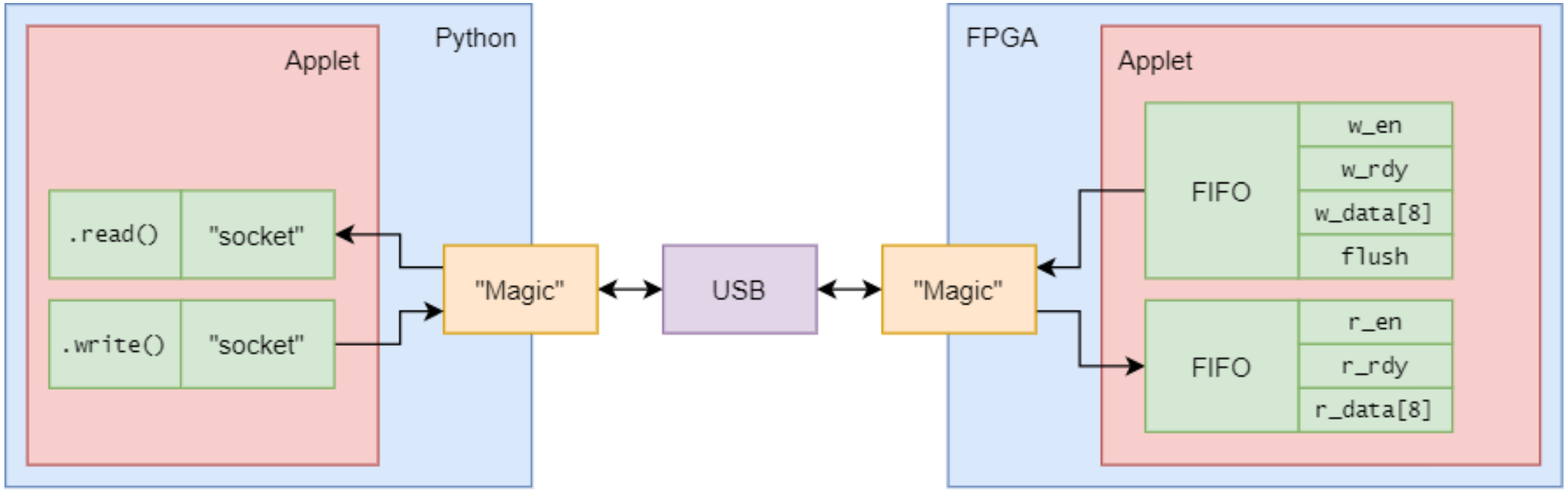


- Each port has:
  - Independent 1.8v - 5v power supplies
    - Upto ~150 mA
    - Infinite short circuit
  - All I/Os are translated to this voltage
  - Voltage sense, and monitor (36v Max)
  - Voltage “mirror”
  - Current sense, and trip / limit



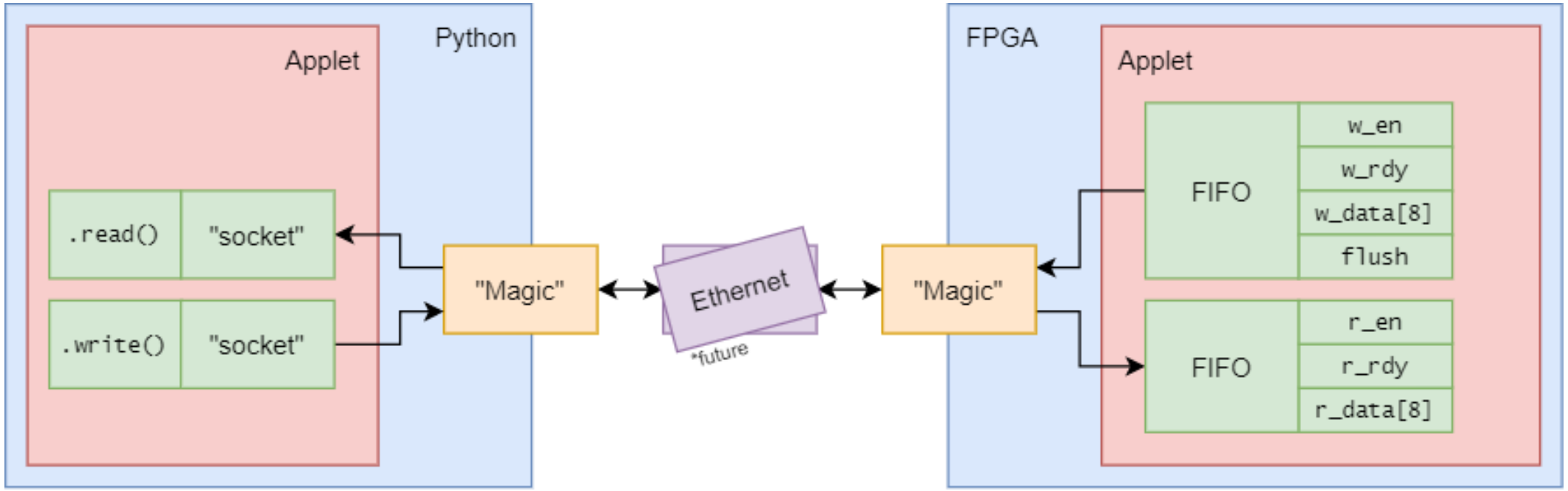
About the Interfaces – Socket / FIFO

- Socket-like interface in Python
- FIFO interface in nMigen
- Very simple to use, but if you're after performance, here be dragons (TBC…)



About the Interfaces – Socket / FIFO

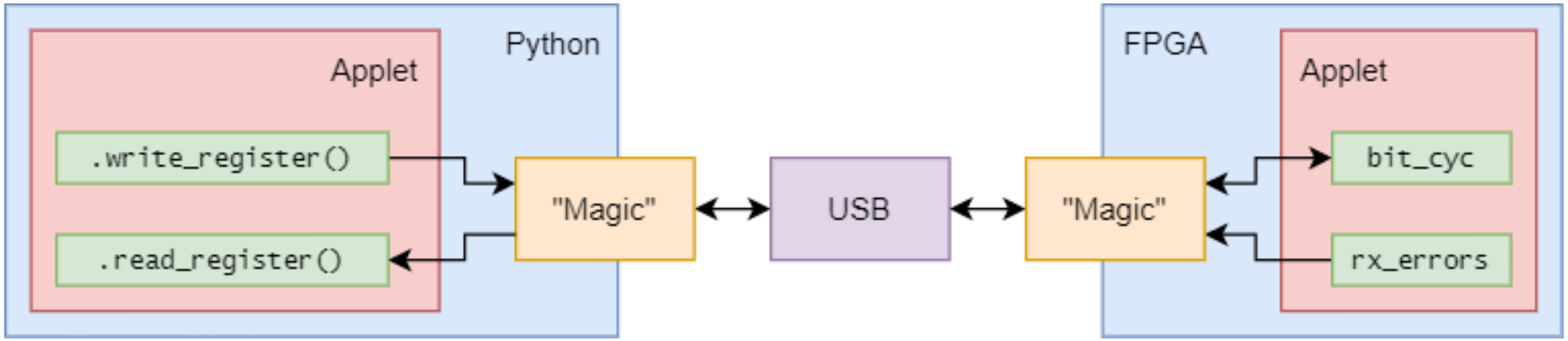
- Ethernet is in the *future* plans (not yet)
- The protocol should adapt easily



About the Interfaces – Registers

- Registers for configuration and status
  - Read-Write i.e: only Python can write
  - Read-Only i.e: only nMigen can write
- Just like peripheral registers in an MCU

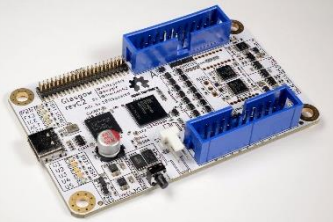
	Python		nMigen	
	Read	Write	Read	Write
Read-Write	✓	✓	✓	✗
Read-Only	✓	✗	✓	✓



Anatomy of an Applet

- Subclass of `GlasgowApplet` class, consists of three phases / parts:
  - Build
    - Gather the command line arguments associated to the build
    - Build the gateware (can be significantly different based on command line args!)
    - Changes trigger a rebuild of the gateware, which is then cached
  - Run
    - Gather the command line arguments associated with the instance / execution
    - Start up the applet in both Python and nMigen
    - Changes do not trigger a rebuild of the gateware, and are thus very fast to tweak
  - Interact
    - Gather the command line arguments associated with the usage / user
    - Make use of the exposed interfaces!
    - Changes to not trigger a rebuild of the gateware

Anatomy of an Applet



- Subclass of `GlasgowApplet` class, consists of three phases / parts:

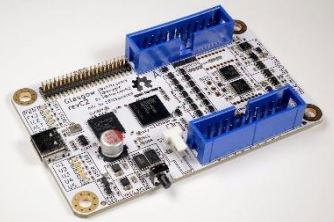
Rebuild?

- Build
  - Gather the command line arguments associated to the build
  - Build the gateware (can be significantly different based on command line args!)
  - Changes trigger a rebuild of the gateware, which is then cached

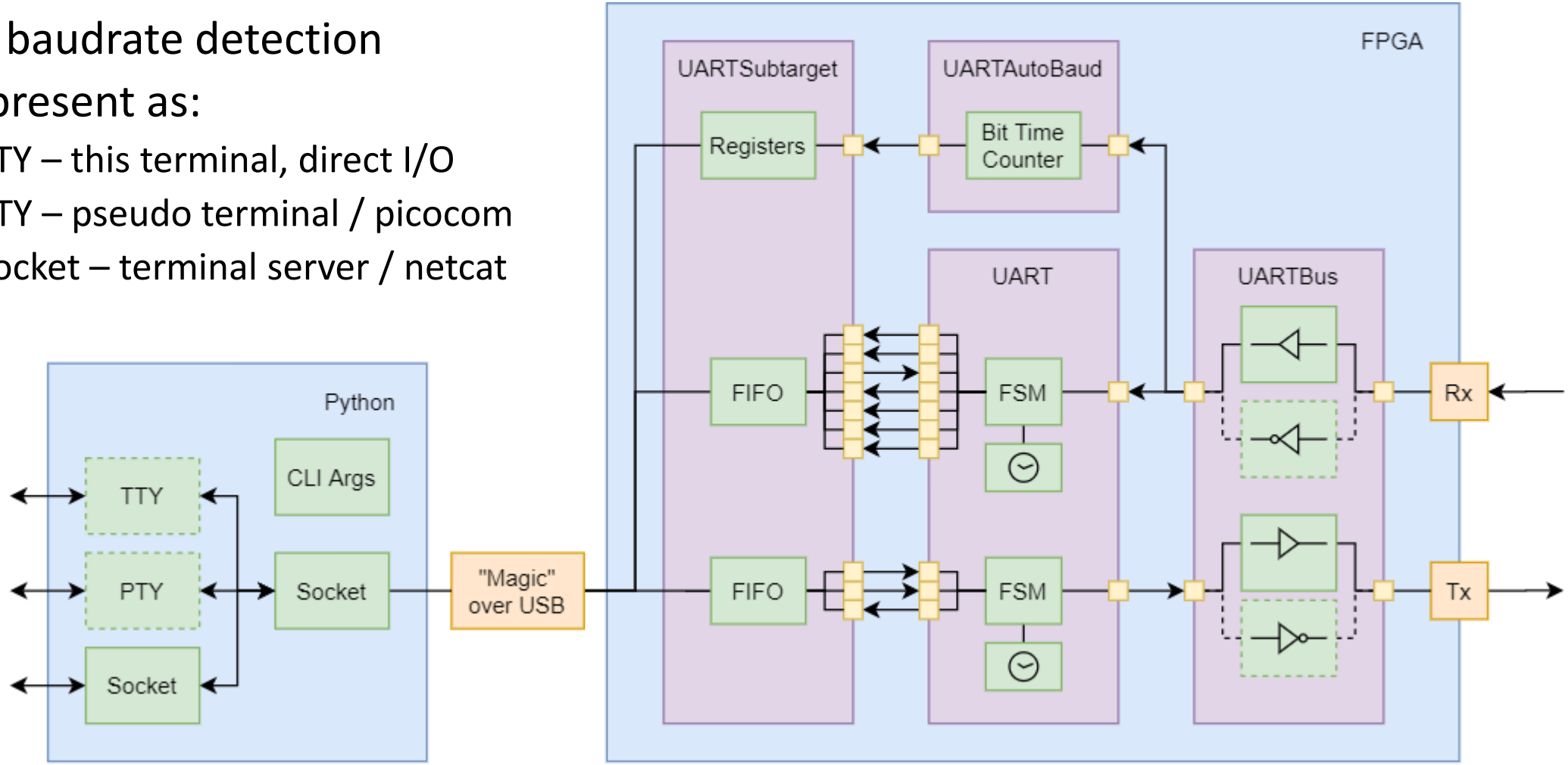
- Run
  - Gather the command line arguments associated with the instance / execution
  - Start up the applet in both Python and nMigen
  - Changes do not trigger a rebuild of the gateware, and are thus very fast to tweak

- Interact
  - Gather the command line arguments associated with the usage / user
  - Make use of the exposed interfaces!
  - Changes to not trigger a rebuild of the gateware

Anatomy of an Applet

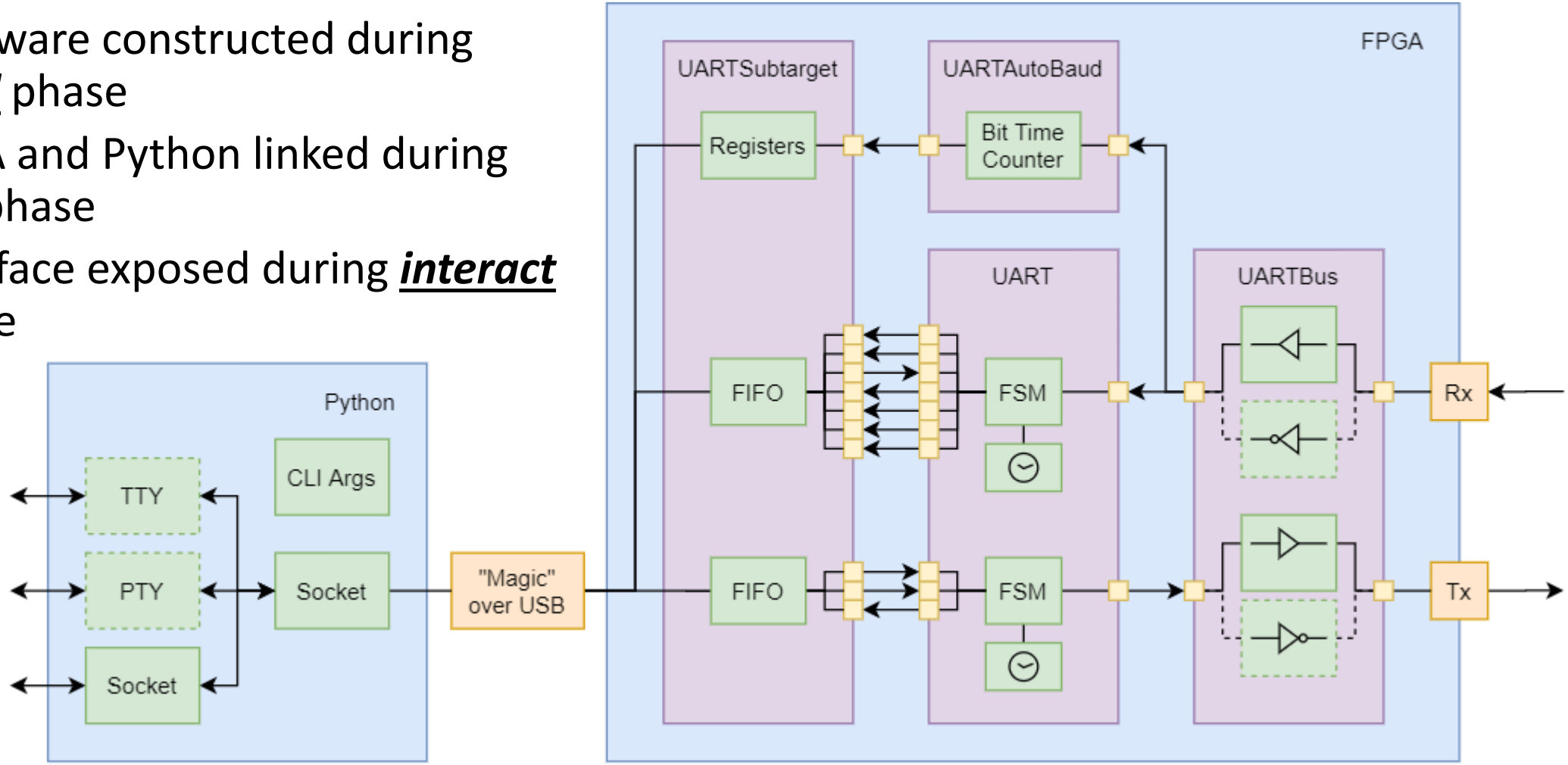


- Example: UART
  - Auto baudrate detection
  - Can present as:
    - TTY – this terminal, direct I/O
    - PTY – pseudo terminal / picocom
    - Socket – terminal server / netcat

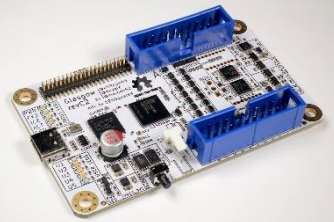


Anatomy of an Applet

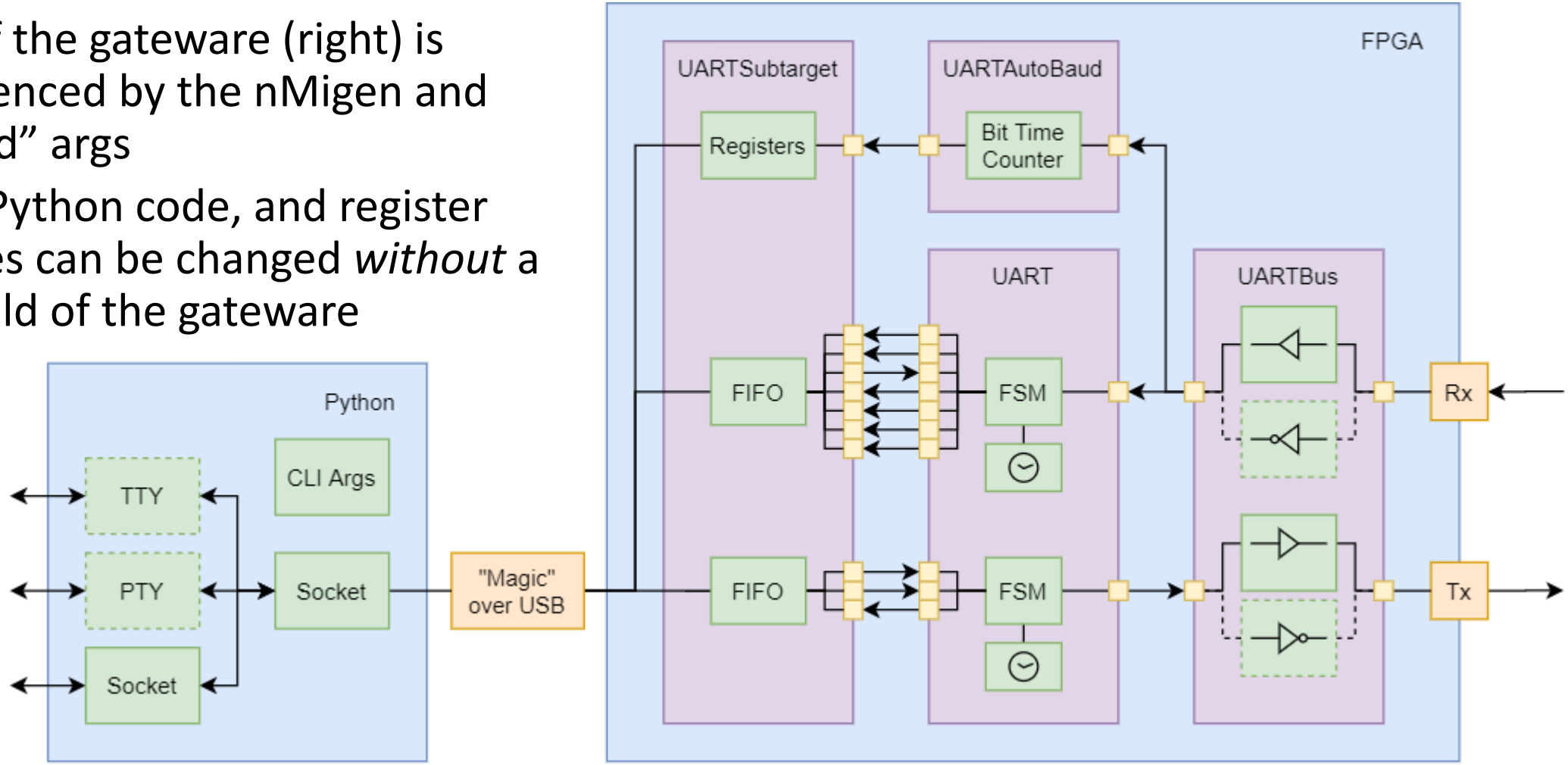
- Example: UART
  - Gateware constructed during ***build*** phase
  - FPGA and Python linked during ***run*** phase
  - Interface exposed during ***interact*** phase



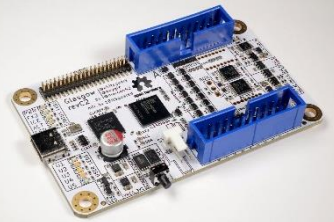
Anatomy of an Applet



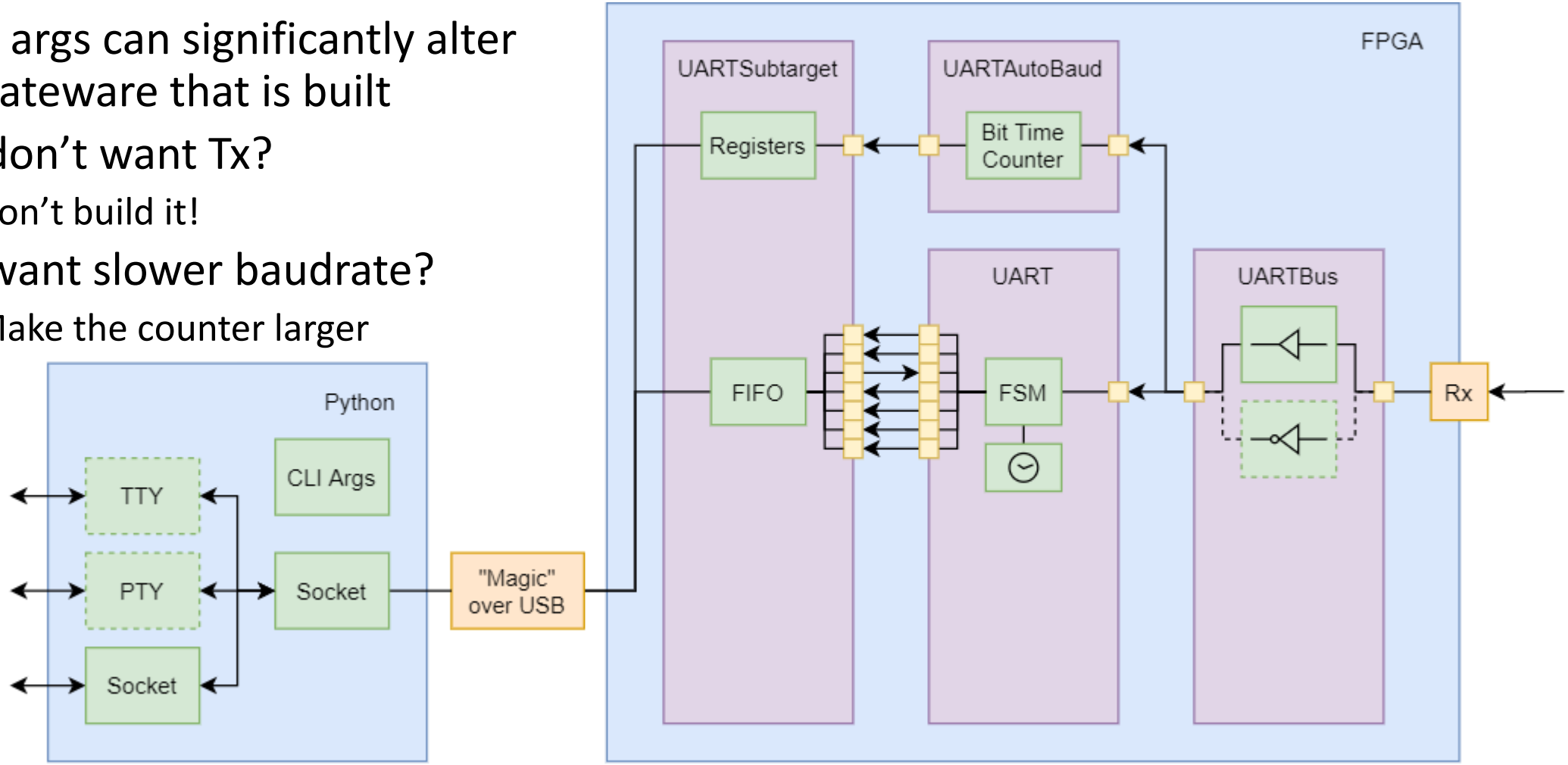
- Example: UART
  - All of the gateware (right) is influenced by the nMigen and “build” args
  - The Python code, and register values can be changed *without* a rebuild of the gateware



Anatomy of an Applet



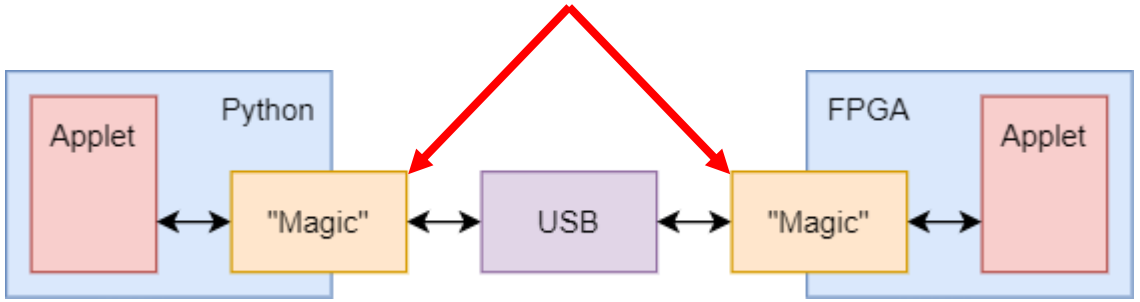
- Example: UART
  - Build args can significantly alter the gateware that is built
  - e.g: don't want Tx?
    - Don't build it!
  - e.g: want slower baudrate?
    - Make the counter larger



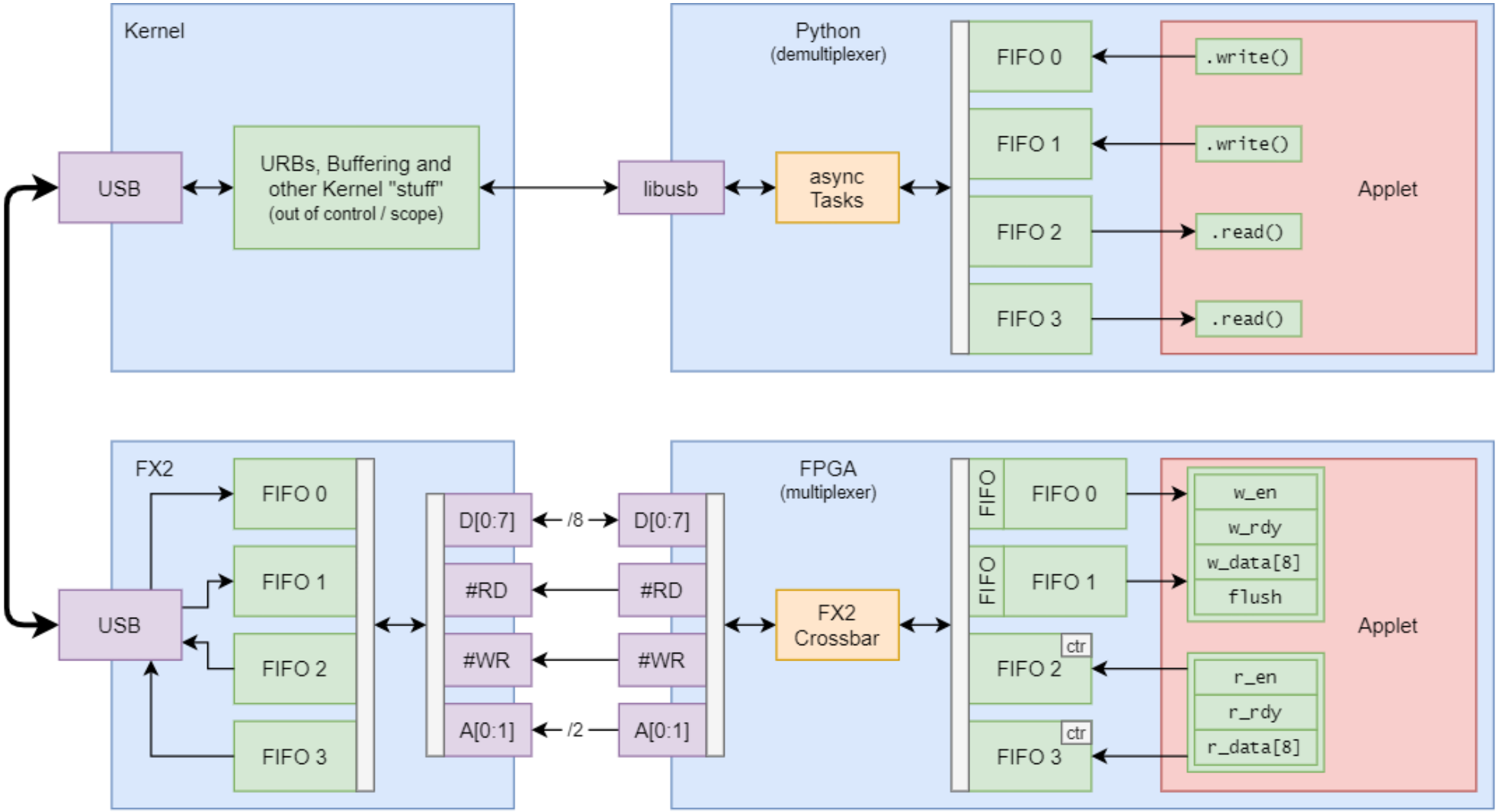
Deep Dive: Byte Stream Data Path

- Simple / low bandwidth applications can use in ignorance
- To keep the throughput up requires careful management…
  - A lot of the problems are taken care of for you by the infrastructure
  - One (ish?) edge case is still a hidden trap for users
    - Non-obvious / deep technical explanation

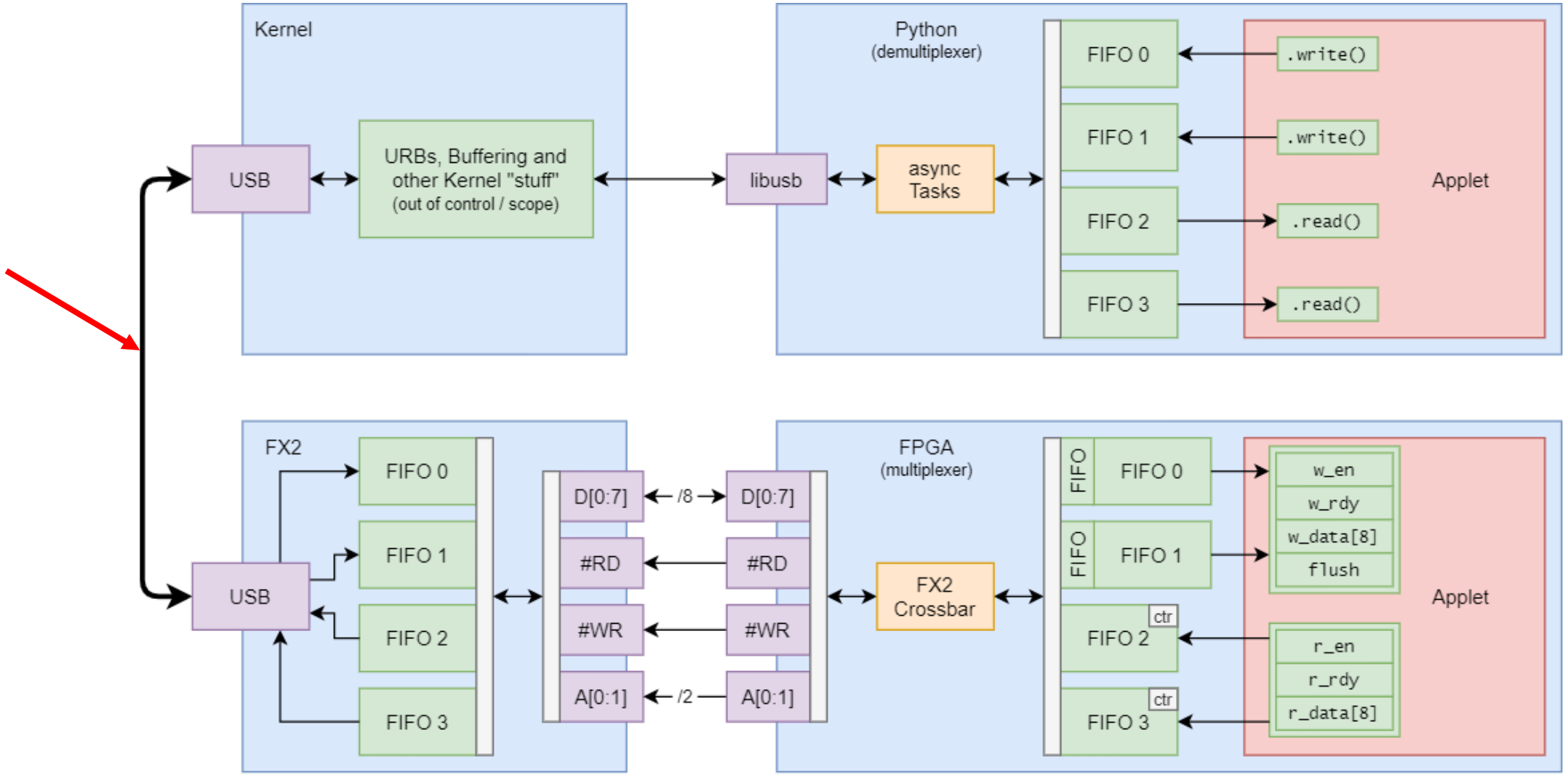
- Let's expand on that "Magic" block either side of USB…



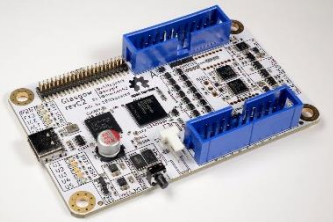
Deep Dive: Byte Stream Data Path

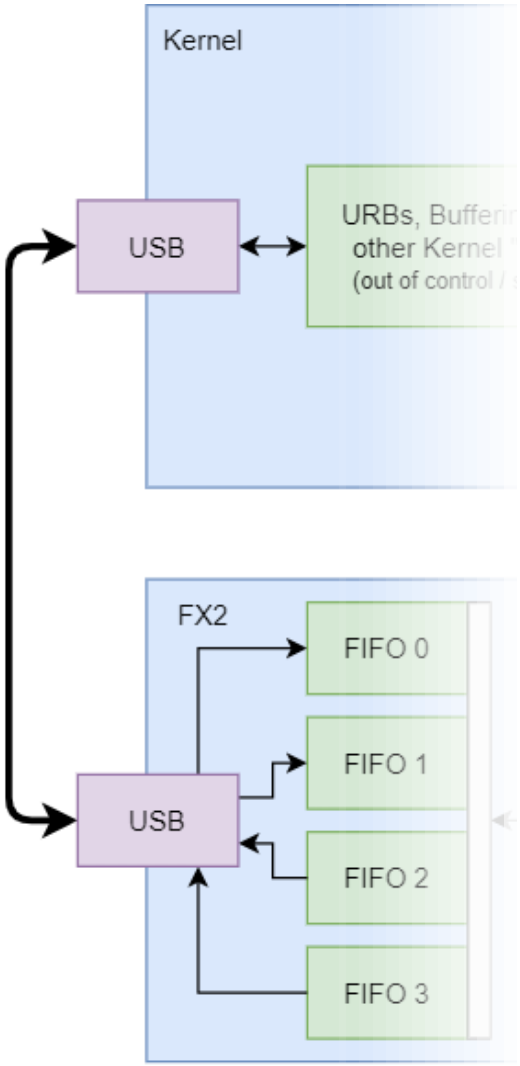


Deep Dive: Byte Stream Data Path



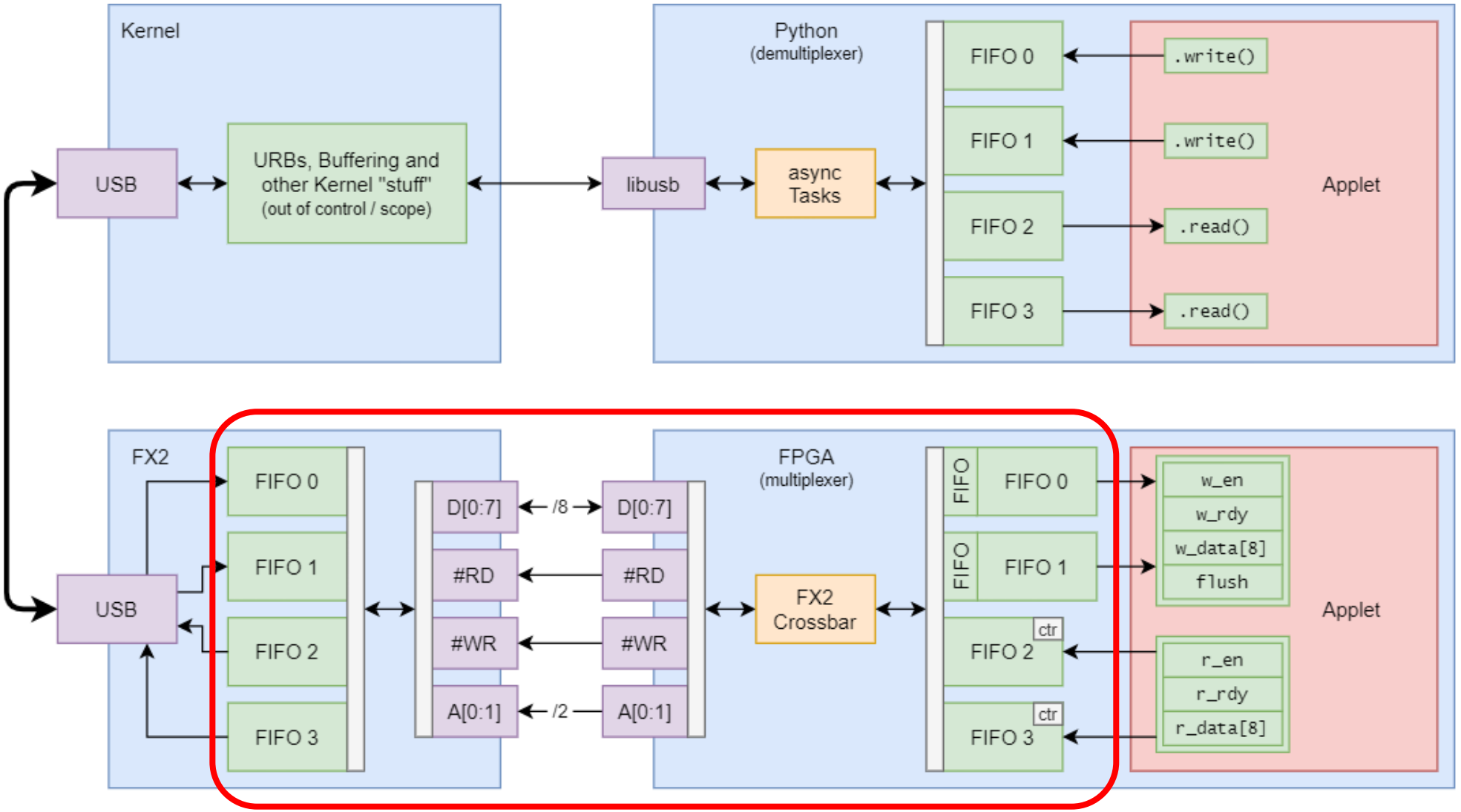
Deep Dive: Byte Stream Data Path





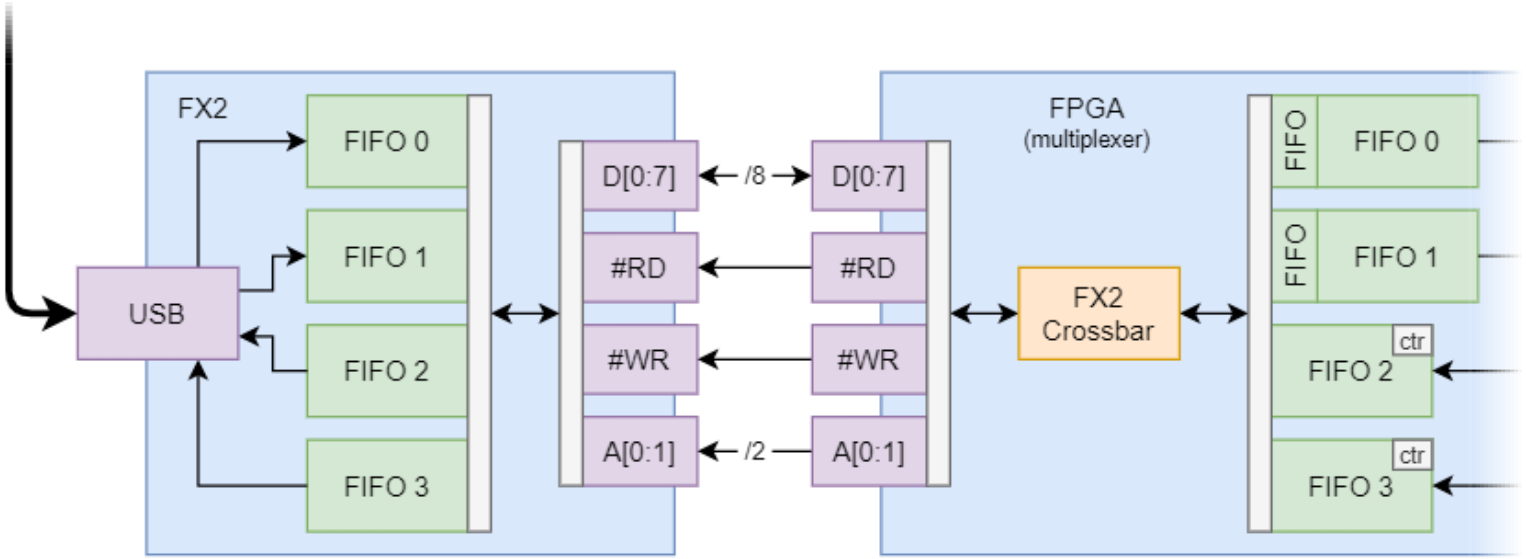
- USB packet size, and polling!
- If you have a lot to say, don't send a partial packet
  - This is effectively saying "*I'm done!*"
  - Host won't ask again for a while
  - Can make FIFO(s) overflow
- If host asks for a packet, and you have a lot to say, *try* to give a *full packet!*
  - This is effectively saying "*I have more to say!*"
  - Host will probably ask again more quickly
- FX2 and Glasgow FIFOs are fairly small

Deep Dive: Byte Stream Data Path



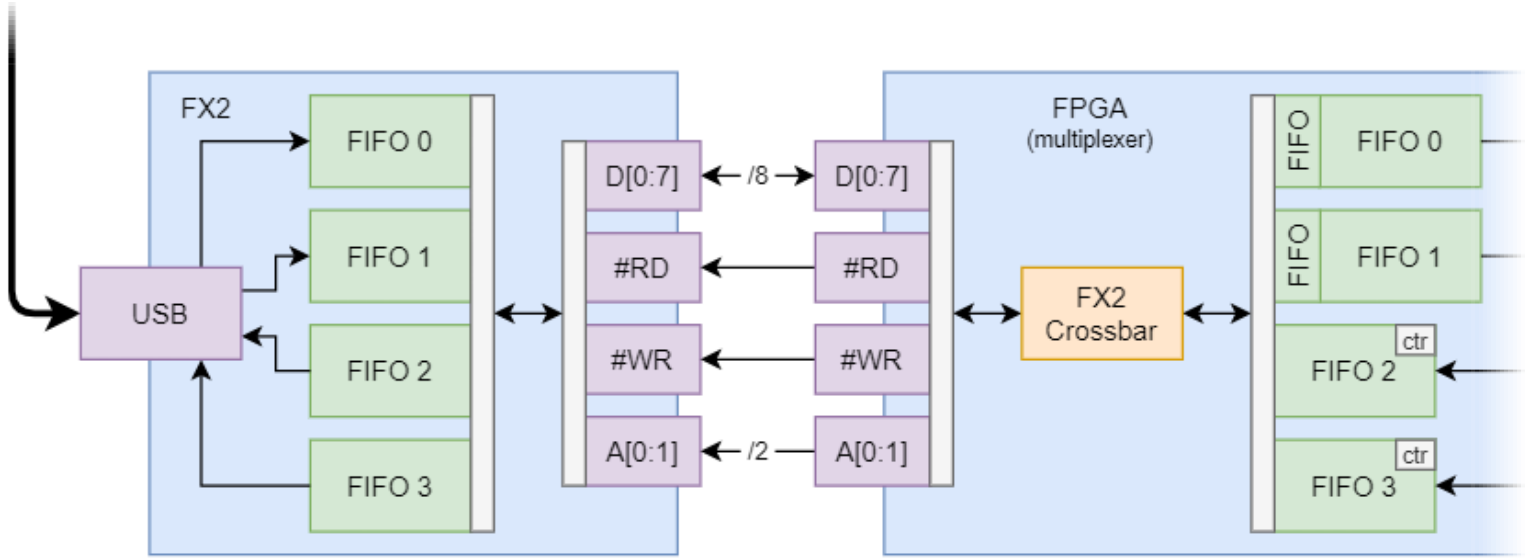
Deep Dive: Byte Stream Data Path

- Massively complex area, user doesn't have to know about it!
- See: `gateware/fx2_crossbar.py` (3x screens of explanation from @whitequark!)
- FX2 has 4x FIFOs, FPGA has up to 4x FIFOs (to match)
- The crossbar coordinates transfers between these FIFOs



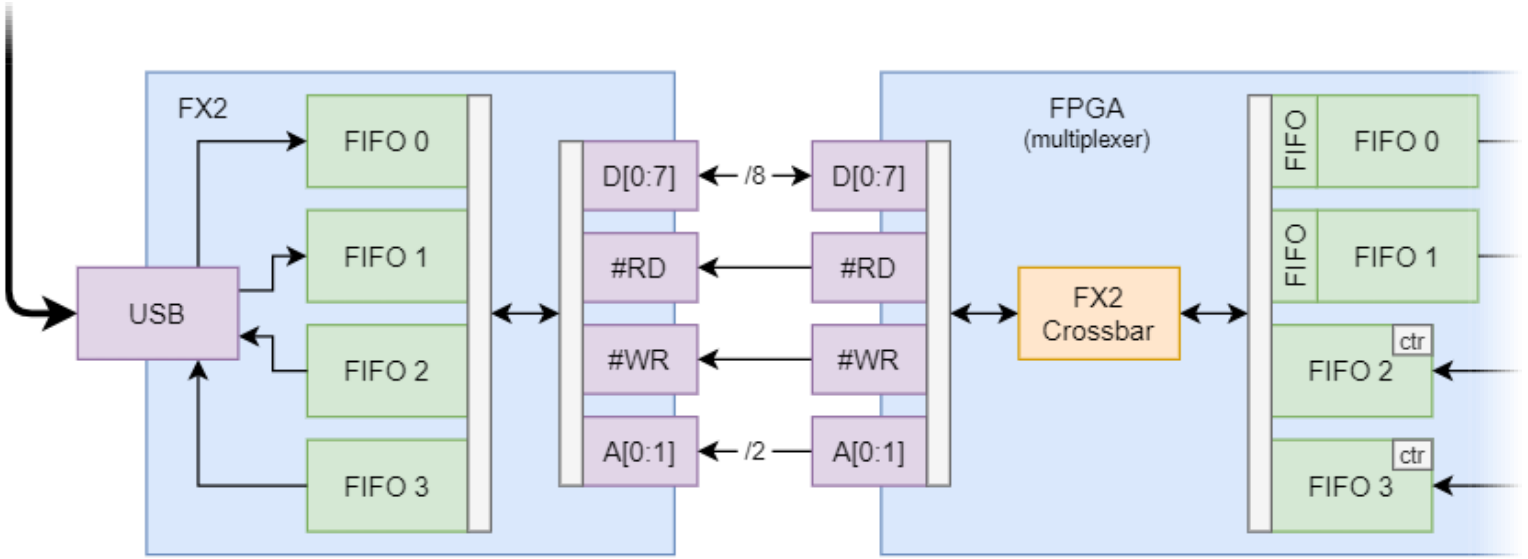
Deep Dive: Byte Stream Data Path

- FX2 configured for synchronous transfer, as clock follower
- 2-bit address to select the desired FX2 FIFO
- FPGA is in control



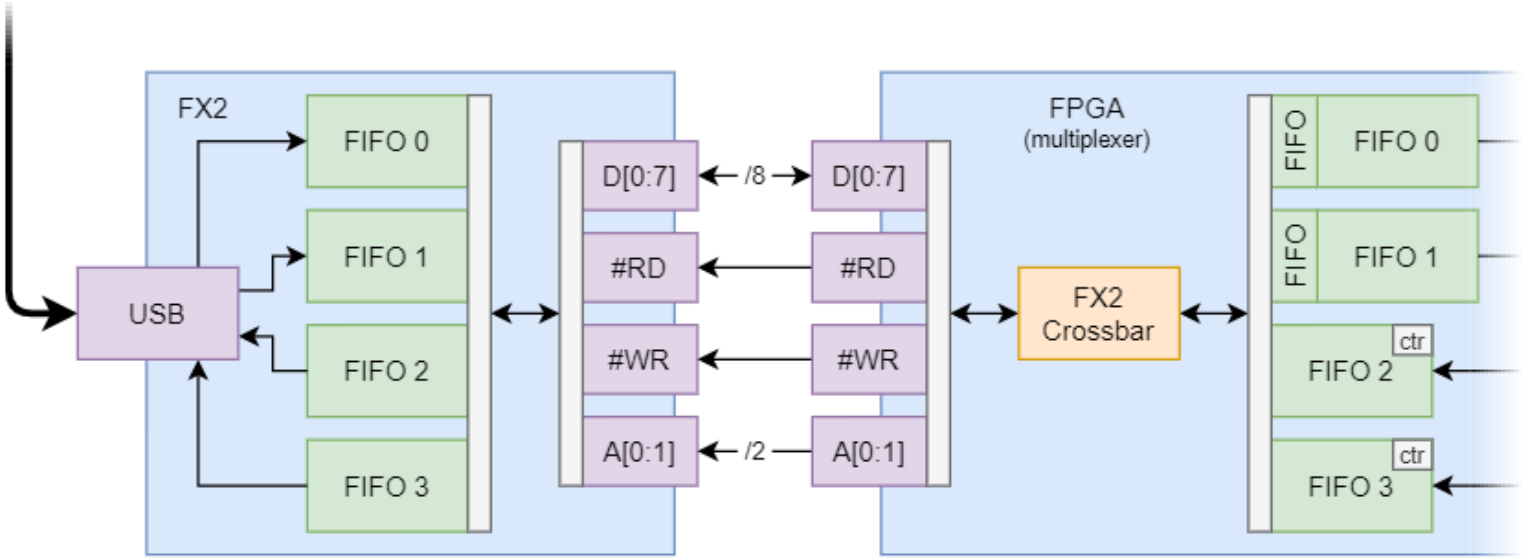
Deep Dive: Byte Stream Data Path

- I/O signals must be buffered, which adds pipelining
- When FPGA writes to FX2 FIFO, *"full"* flag will change late!
- FX2 signals not valid until long after the input capture of the FPGA!
- Feedback nightmare – is the FX2 FIFO full? is the FPGA FIFO empty?



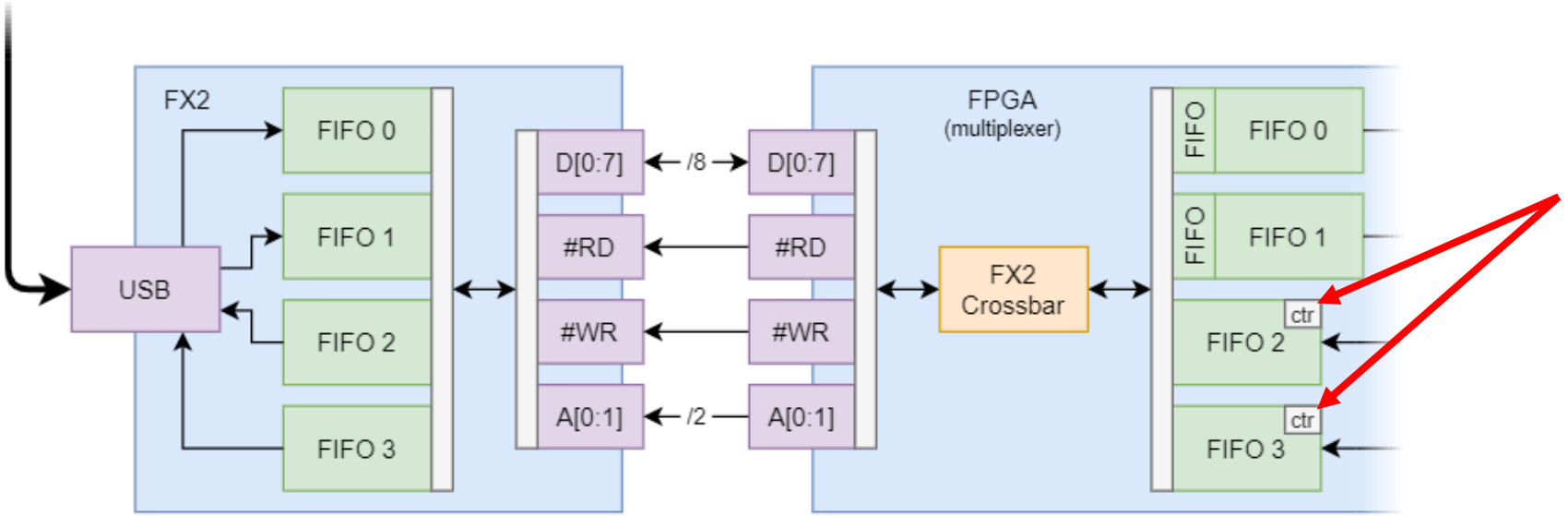
Deep Dive: Byte Stream Data Path

- Different solution for IN FIFO vs OUT FIFO



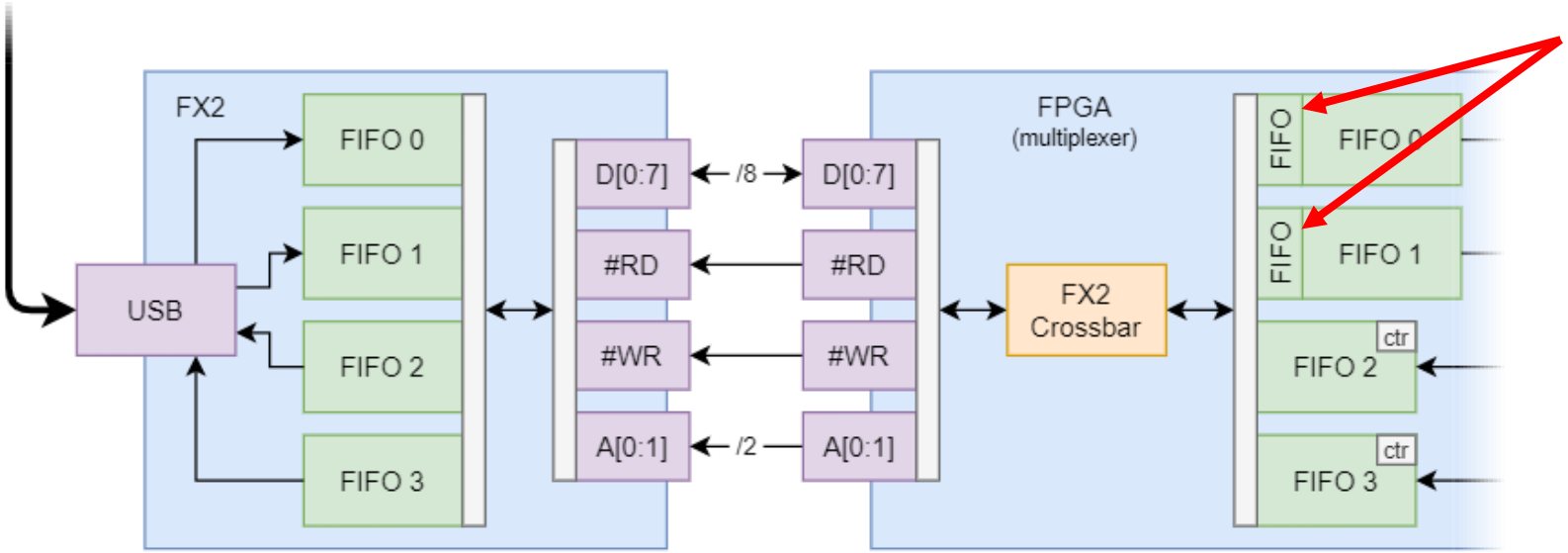
Deep Dive: Byte Stream Data Path

- Solution – IN FIFOs (FPGA to PC)
  - Track the FX2 FIFO level using a counter in the FPGA…
  - Gives us a virtual, but perfect *“full”* flag
  - Coordination for reset / FIFO purge, out-of-band



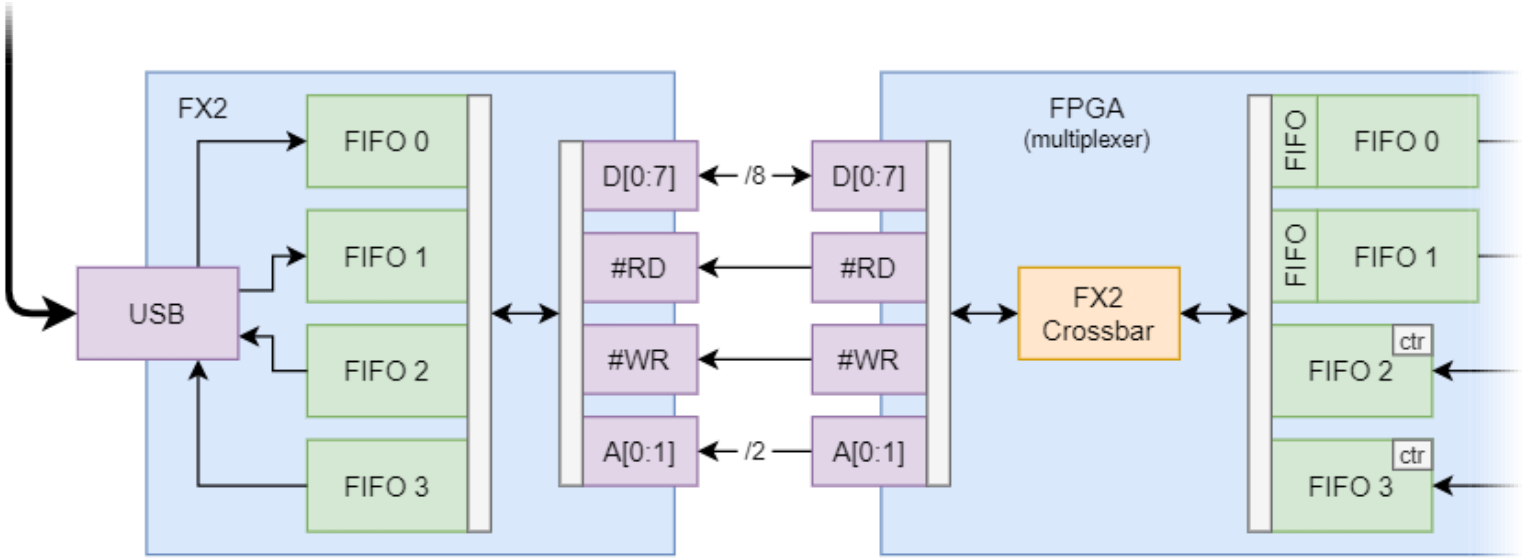
Deep Dive: Byte Stream Data Path

- Solution – OUT FIFOs (PC to FPGA)
  - Very small FIFO in front of the main FIFO
  - Absorbs any additional writes from the pipeline

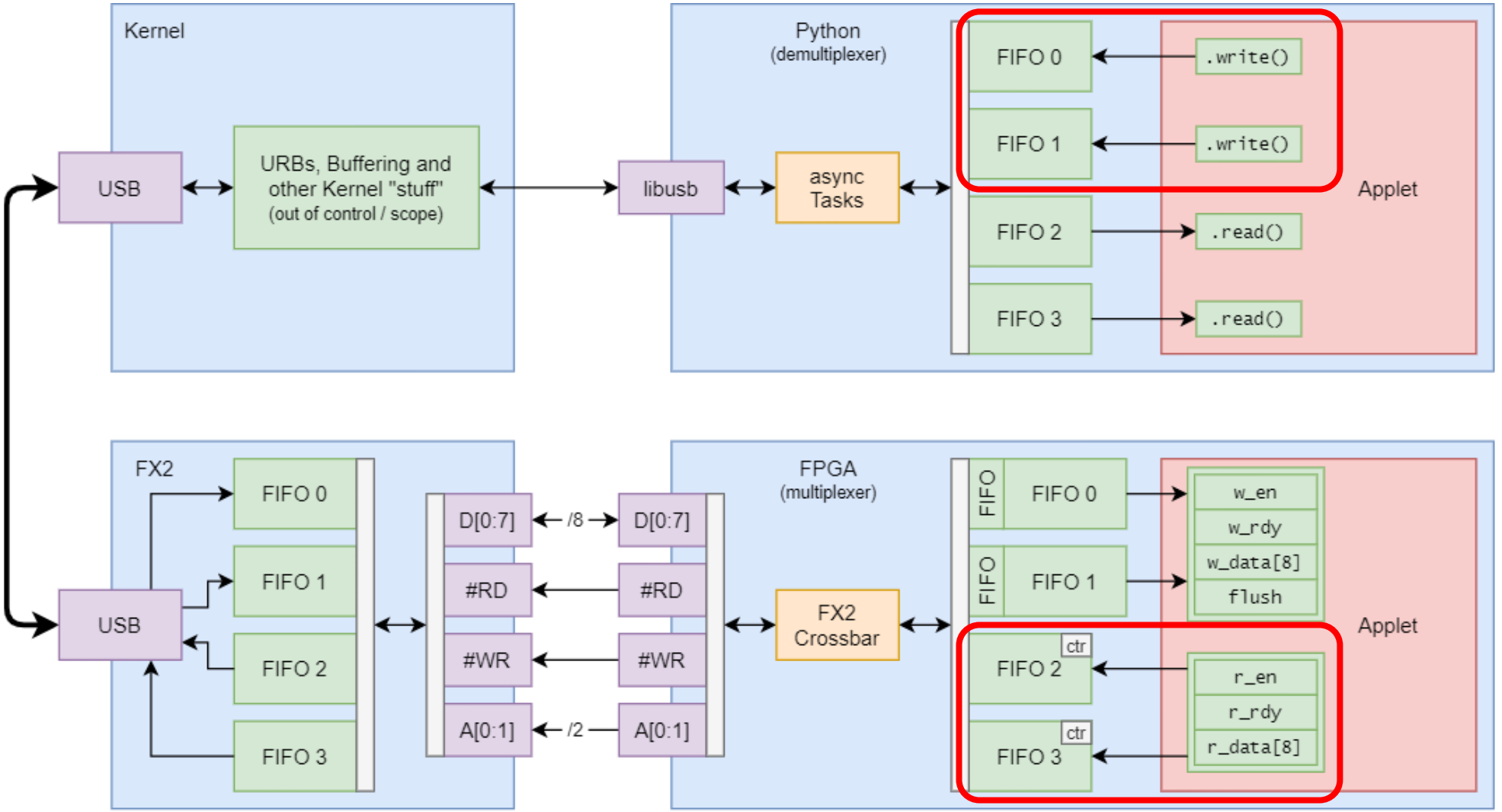


Deep Dive: Byte Stream Data Path

- USB is packet-oriented, FIFOs are byte-oriented
- For IN FIFOs, the FPGA is responsible for inserting packet boundaries
  - Short USB packets need to be forcibly flushed
  - ZLP generated if previous packet was full, but there is no more data

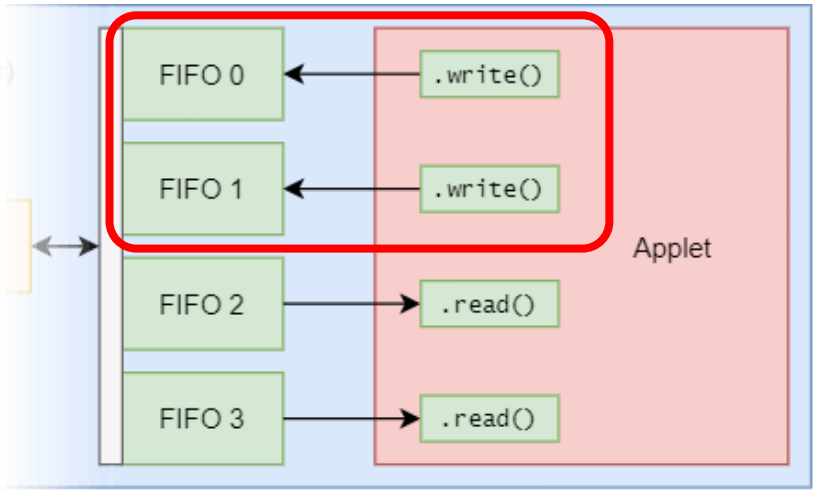


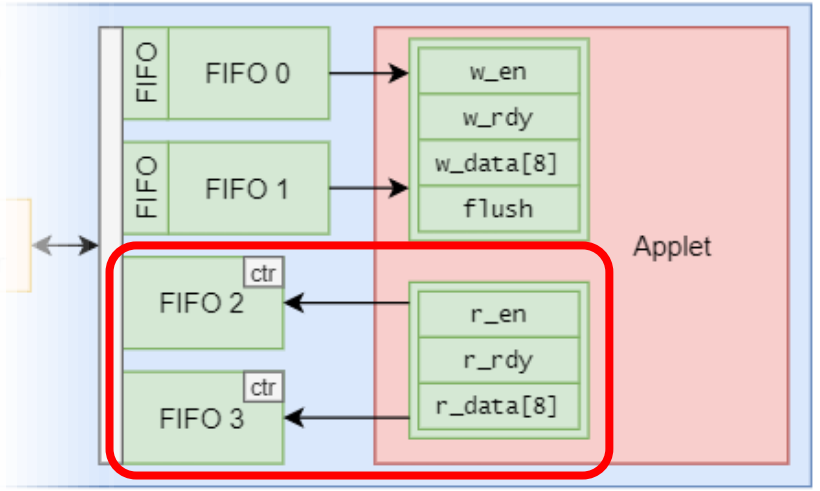
Deep Dive: Byte Stream Data Path



Deep Dive: Byte Stream Data Path

- In summary…
- As you may have gathered…
  - Flush timing is really important
- If you're low bandwidth
  - Just use the default, `auto_flush = True`
- If you're high bandwidth
  - You'll want to set `auto_flush = False`
  - Flush manually if / when necessary
- With careful configuration, Glasgow can achieve ~42 MiB/s over USB 2.0(!)





Future Plans

- Rev C
  - Will always be supported
  - Does not compete with other revisions – it's a different tool
- Rev D
  - 4x ports, for 32x I/O pins
  - Addons from revC will be compatible
  - Planned – at least 2 years out
- Rev E
  - *Probably* USB 3.0 and/or Ethernet
  - *Probably* faster / low-voltage / differential interfaces (SYZYGY?)
  - Planned – no ETA

Production and Campaign

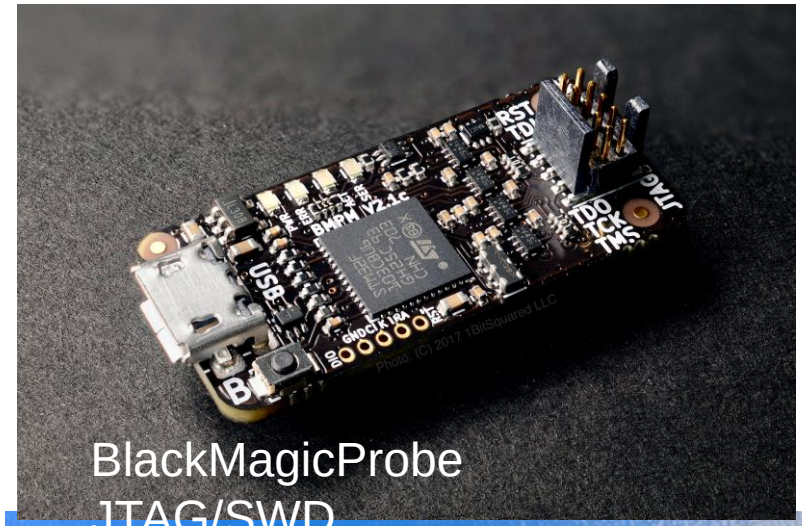
- Piotr Esden-Tempski 1BitSquared
- CrowdSupply Campaign (finished but pre-orders available)
- DesignForManufacture (DFM)
- All designed in KiCad
- Parts Availability
- Footprint new JEDEC → roundrect pads
- Added case, wire harness

Who am I

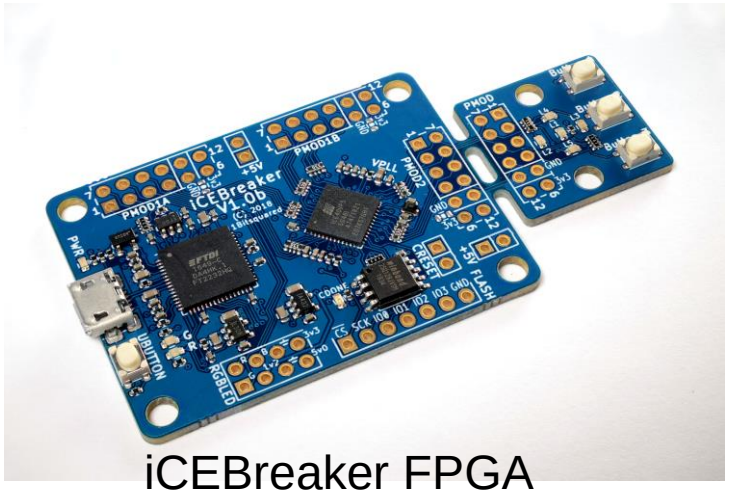
- Piotr Esden-Tempski
- Founder 1BitSquared
- Open-Source Hardware
- Development boards
- Debug Tools

1 BIT SQUARED®

1BitSquared

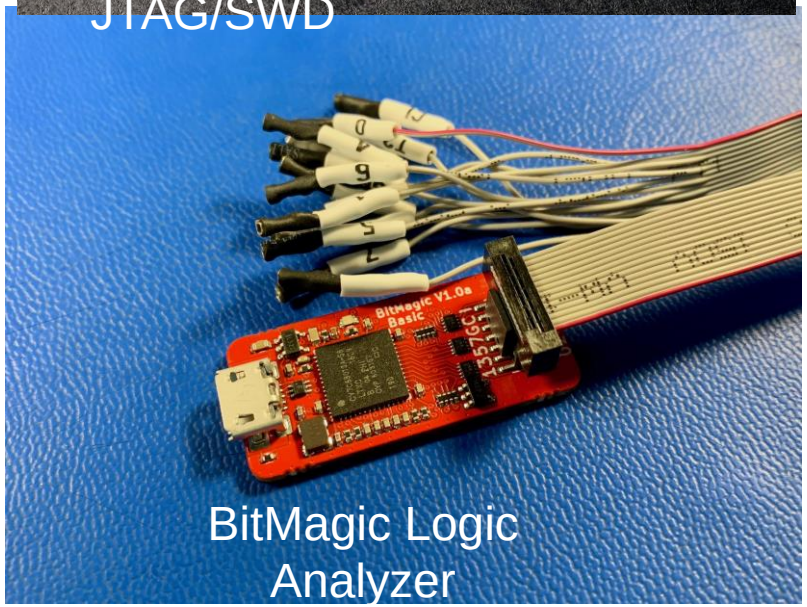


BlackMagicProbe
JTAG/SWD

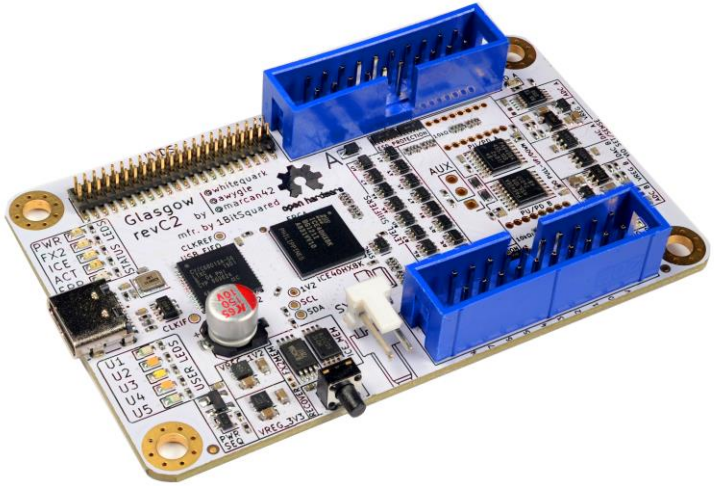


iCEBreaker FPGA

Some of the things we make at 1BitSquared



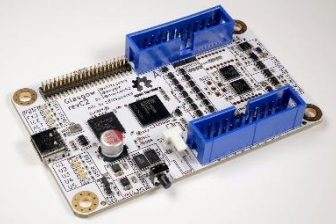
BitMagic Logic Analyzer



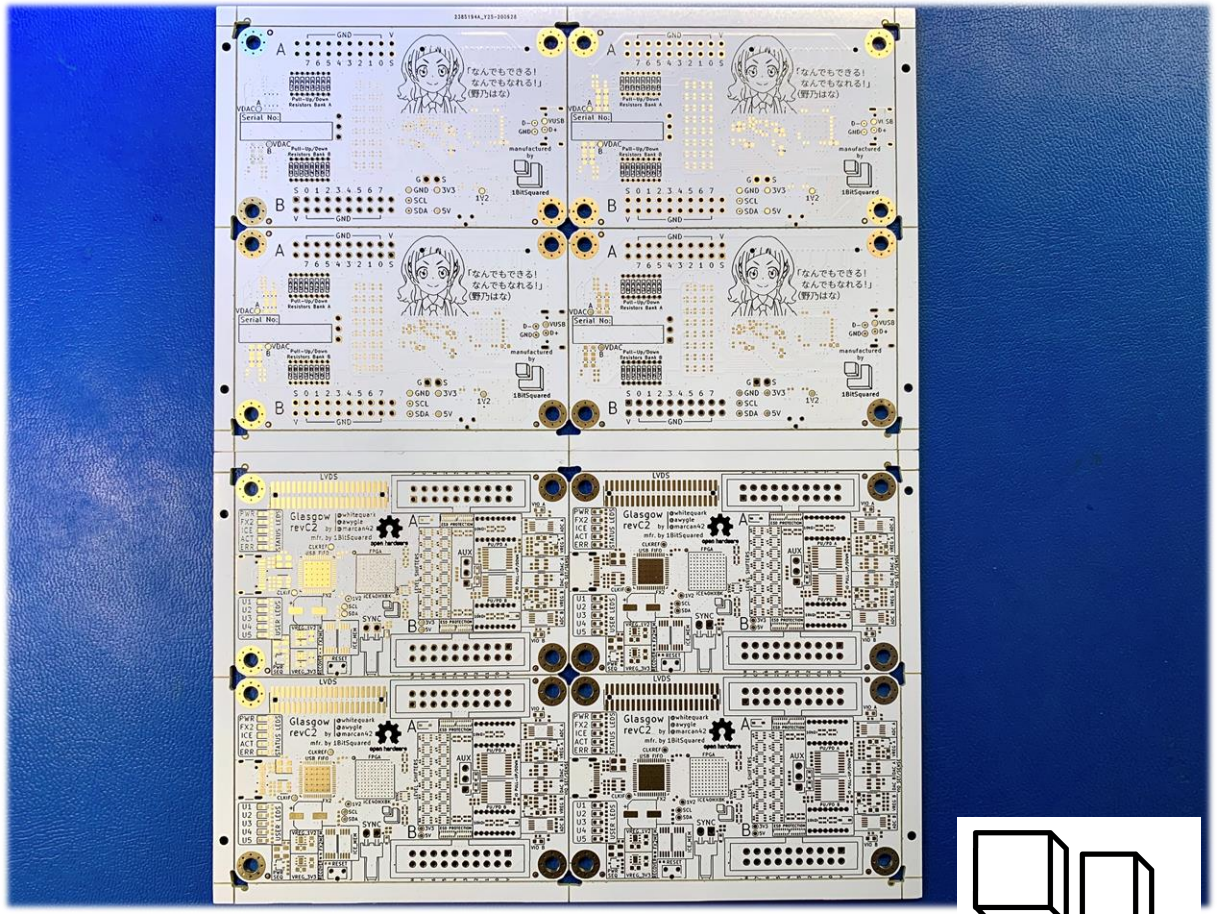
Glasgow
Digital Interface
Explorer

1 BIT SQUARED®

Batch Production

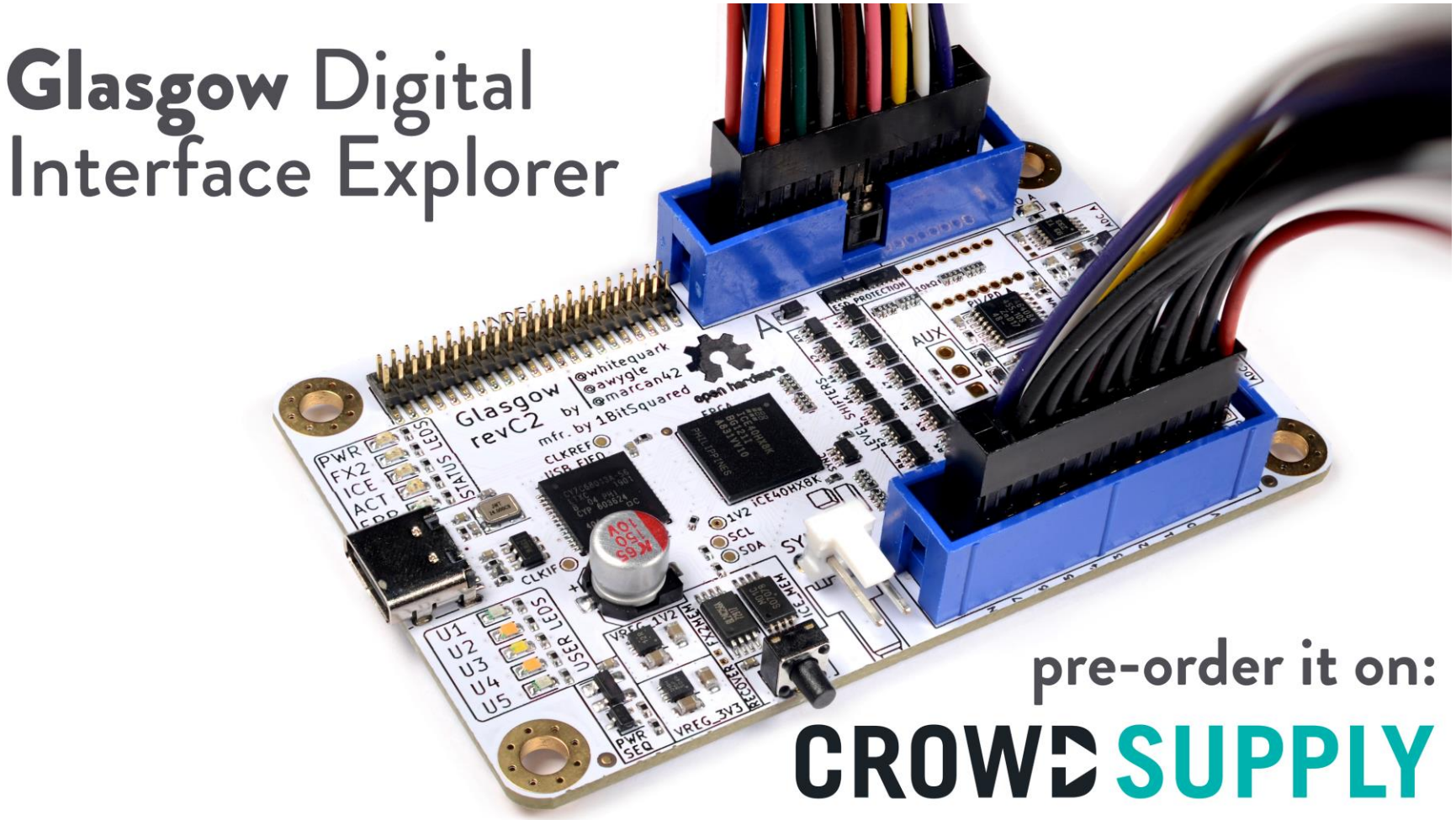


- On Whitequark's request
- Batch Production
- Lower Cost hardware
- Economies of Scale
- Design For Manufacture (DFM)
- Logistics
- Easier access to the hardware



1 BIT SQUARED®

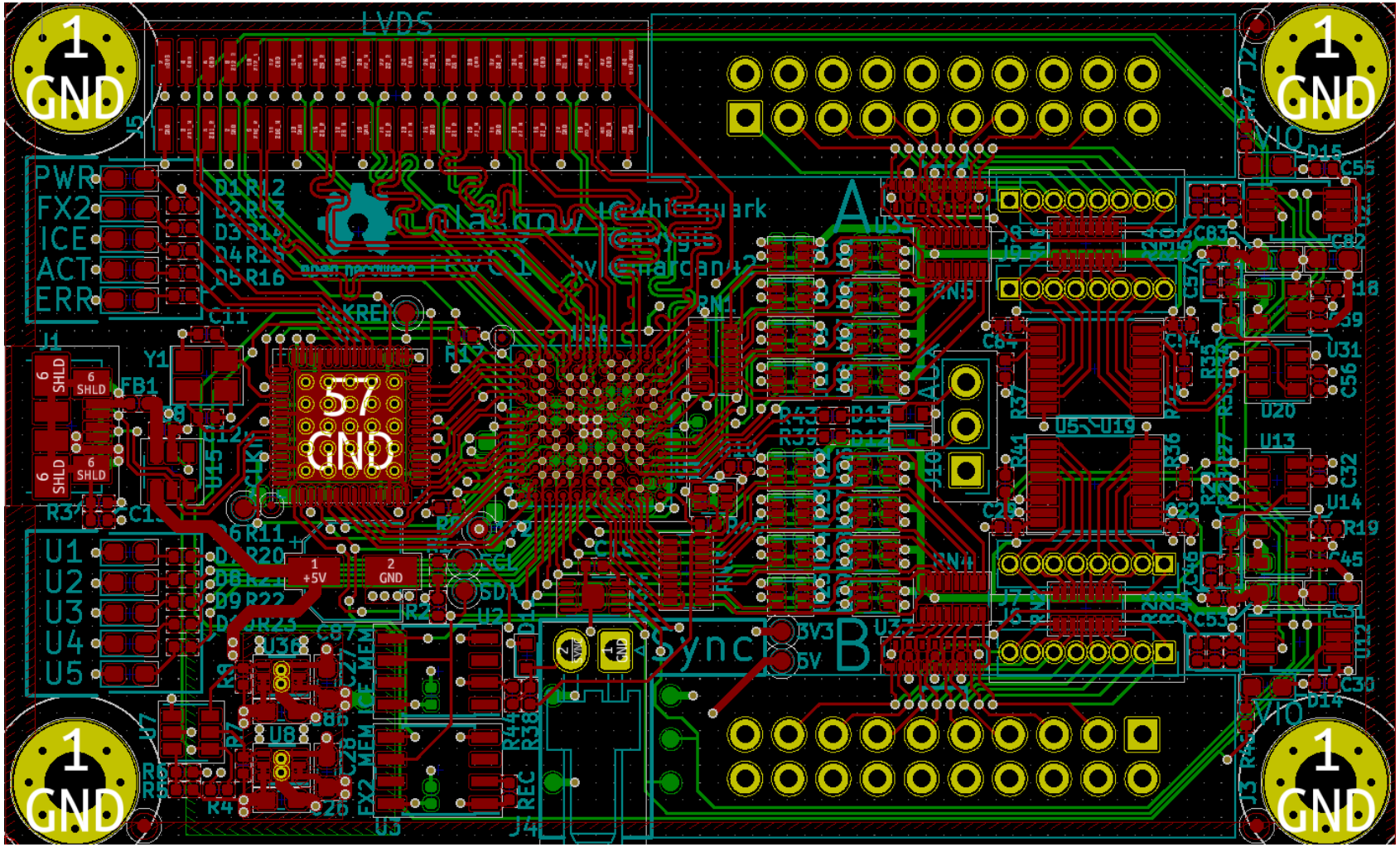
CrowdSupply Campaign



https://www.crowdsupply.com/1bitsquared/glasgow

1 BIT SQUARED®

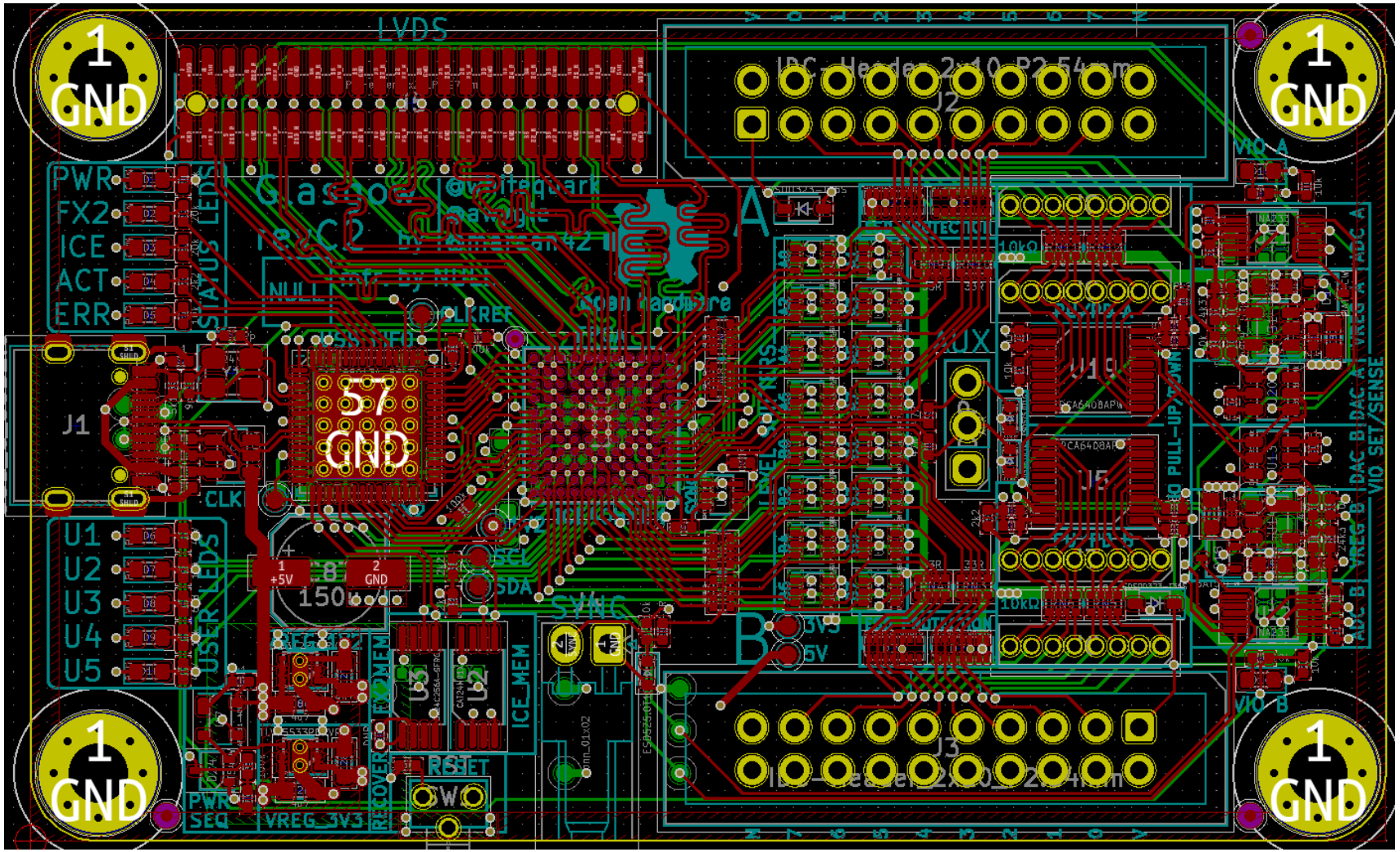
Design For Manufacture



revC0&1 Designed in KiCad Stable by Marcan

1 BIT SQUARED®

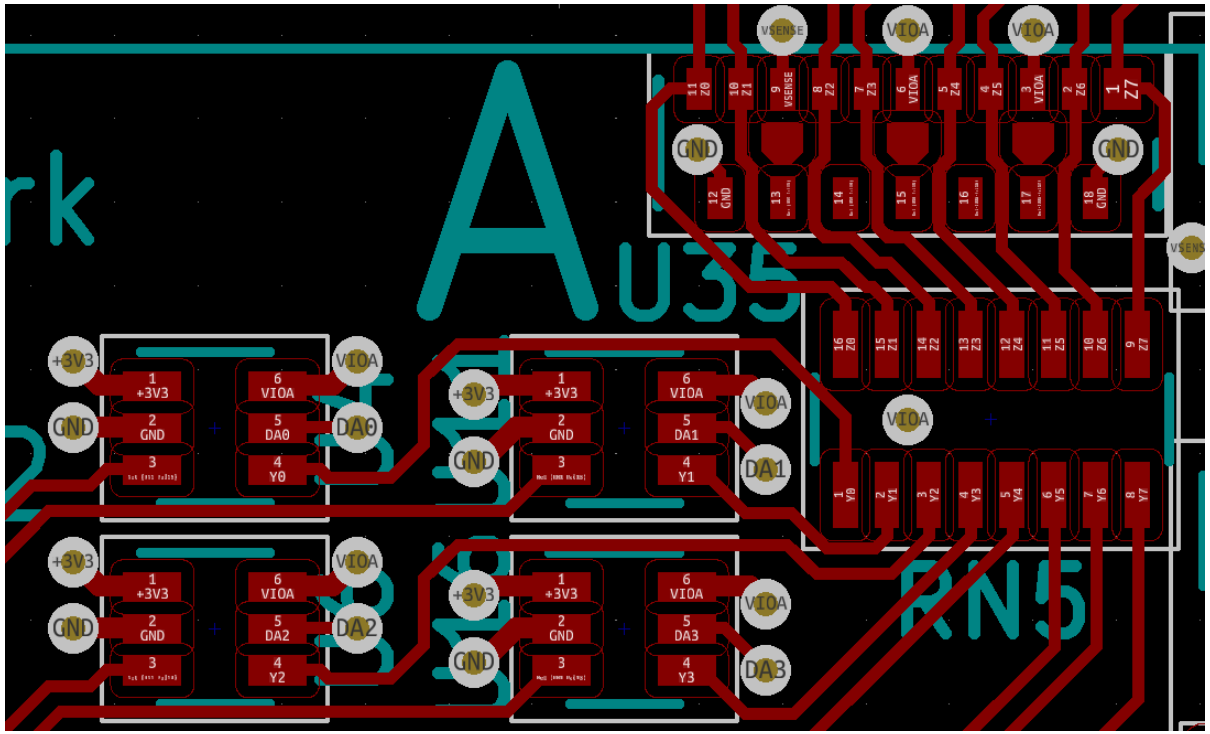
Design For Manufacture

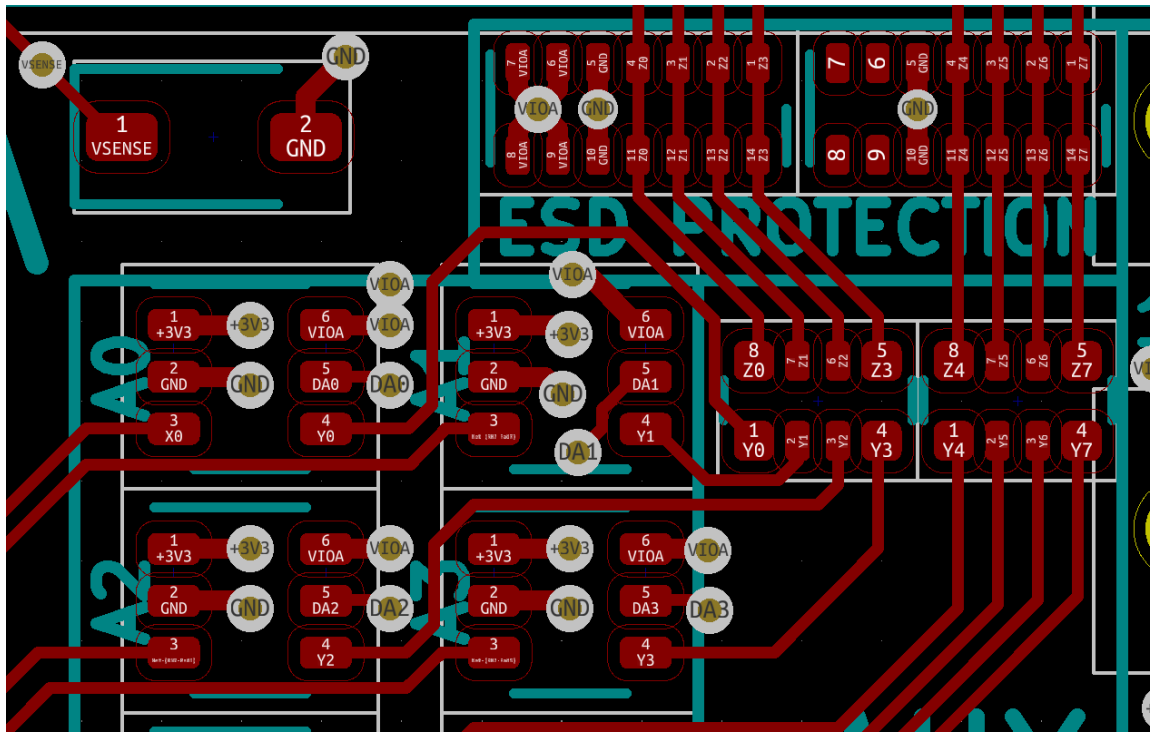


Some changes were needed for production.

1 BIT SQUARED®

Design For Manufacture





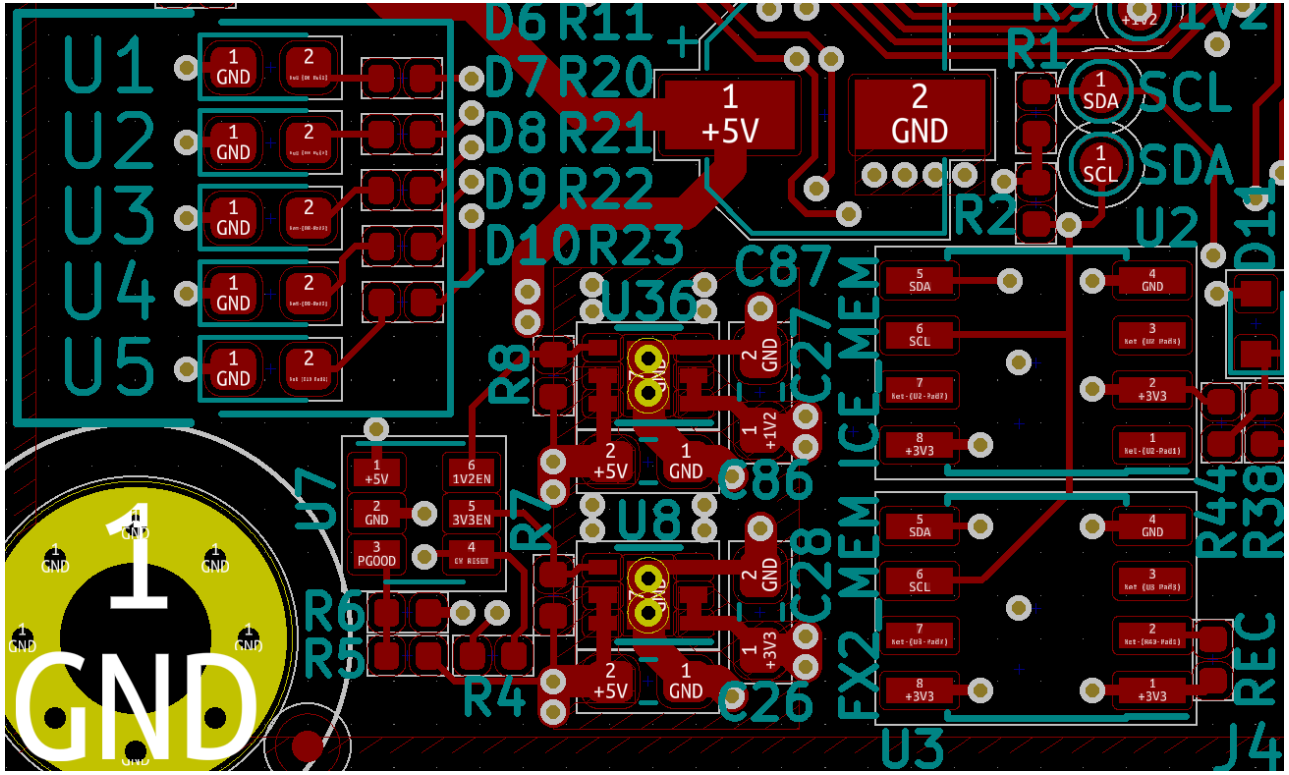
Roundrect pads

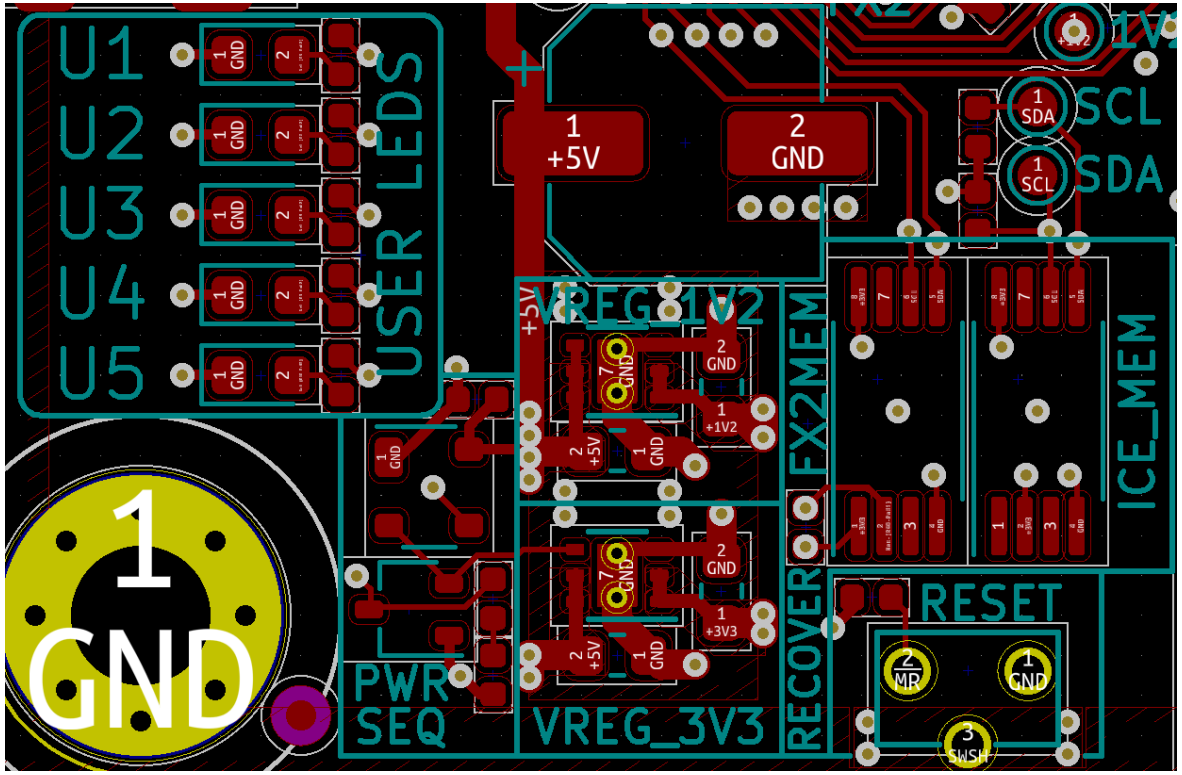
More common parts → ESD, Resistor Packs, Level shifters

Less error prone footprints → ESD Diode GND pads

1 BIT SQUARED®

Design For Manufacture





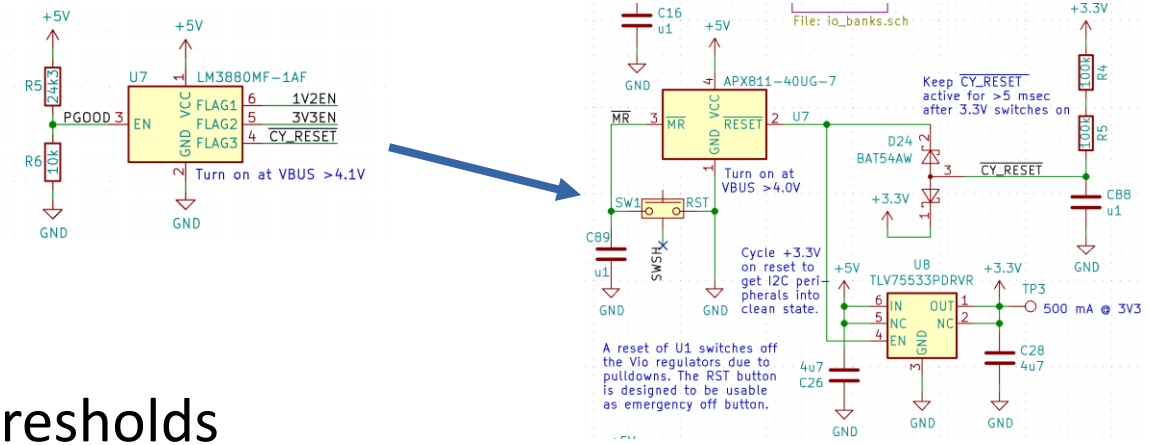
Silkscreen changes → User/Block legend vs Part designators

Added RESET/E-Stop Button → smaller EEPROM packages

1 BIT SQUARED®

Design improvements

- Power sequencing and reset



- Bank current monitoring with settable thresholds
  ADC081C021 → INA233

- Over-/Under-Voltage protection
  - Including specified hardware failure modes

1 BIT SQUARED®

Credits

- Whitequark → @whitequark
- Awygle → @awygle
- Marcan → @marcan42
- electronic_eel
- Many more in the #glasgow IRC channel

1 BIT SQUARED®

KiCad thoughts

- KiCad rocks!
- Collaboration through GitHub is decent but…
- Wish: dedicated proper KiCad diff tool
- Wish: web visualizer integrated into GitLab/GitHub
- Wish: more streamlined official library integration with footprint/3D model generators
- Wish: SVG visualization for documentation (PcbDraw)

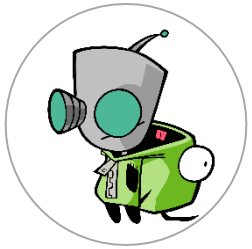
1 BIT SQUARED®

Any Questions?!

- Find us & chat: #glasgow (freenode.net, or 1BitSquared Discord)
- Sources: https://github.com/GlasgowEmbedded/glasgow
- Get one: https://www.crowdsupply.com/1bitsquared/glasgow
- Support whitequark: https://www.patreon.com/whitequark

@whitequark

@attiegrande



@esden