>>> DAPHNE initial design considerations for the FEB
>>> High Performance data acquisition for ARAPUCAS in Dune experiment

Name: Manuel Arroyave and Javier Castaño
Date: May 29, 2019

>>> Content

1. Requirements

2. Mu2e FEB - Context

3. Alternative Design

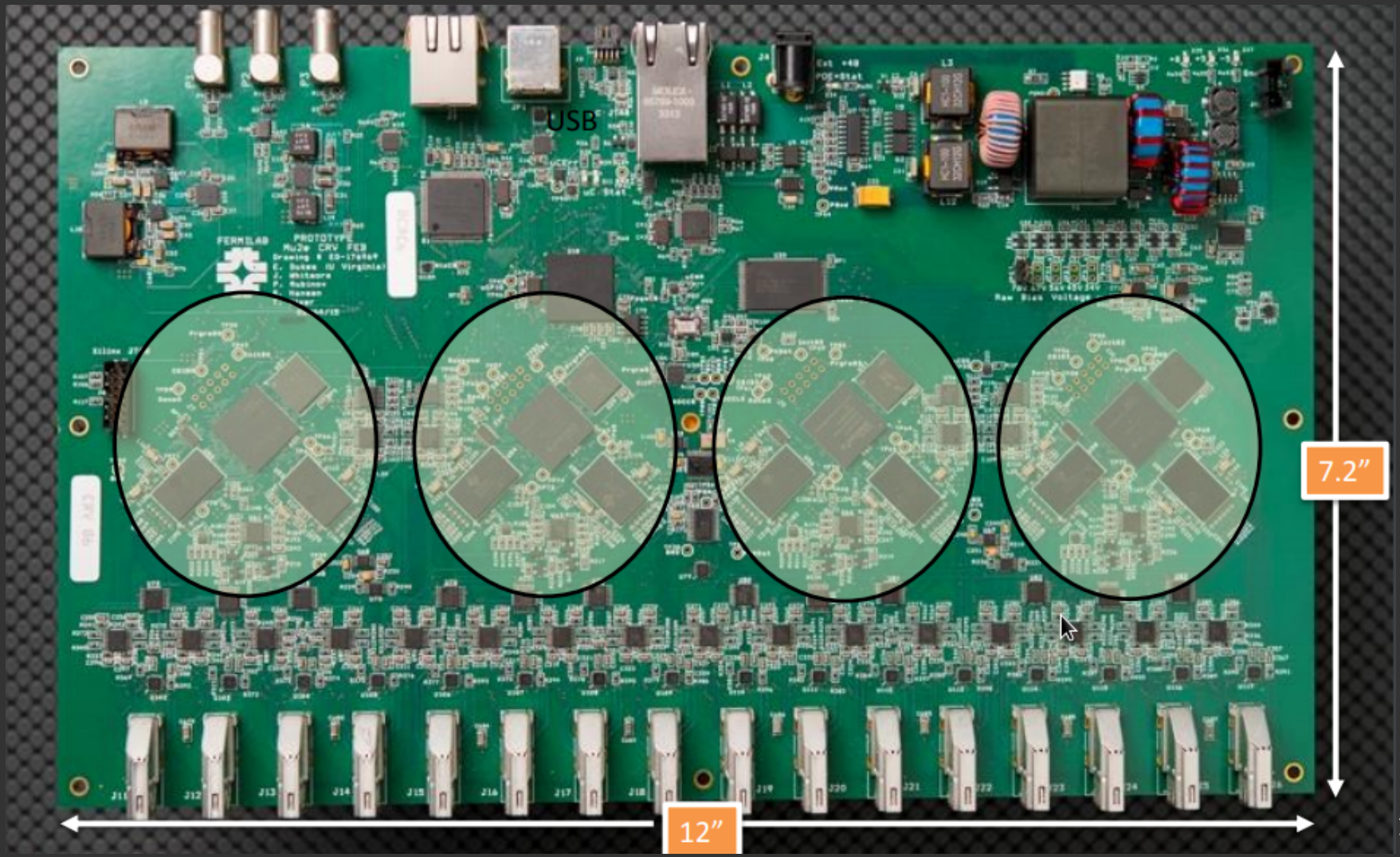
4. LiteX, Migen, Yosys...

>>> What we need to solve?

1. 40 Channel FEB
2. 80 MSPS (12ns/event)
3. Big data output in ADC's stage
   * 51Gb/s (or 6.4 GB/s) for 16 bit
   * 44.8Gb (or 5.6 GB/s) for 14 bit
4. Fast data digitization and multiplexing
5. Ethernet Gigabit ports 1Gb/s (or 125/MB/s)
6. Fast Time prototype.

* Using the highest capacity of 1 ETH Gb port we are able to stream $\approx$ 1/50th piece of total data produced by Max rate SPS ADC's.

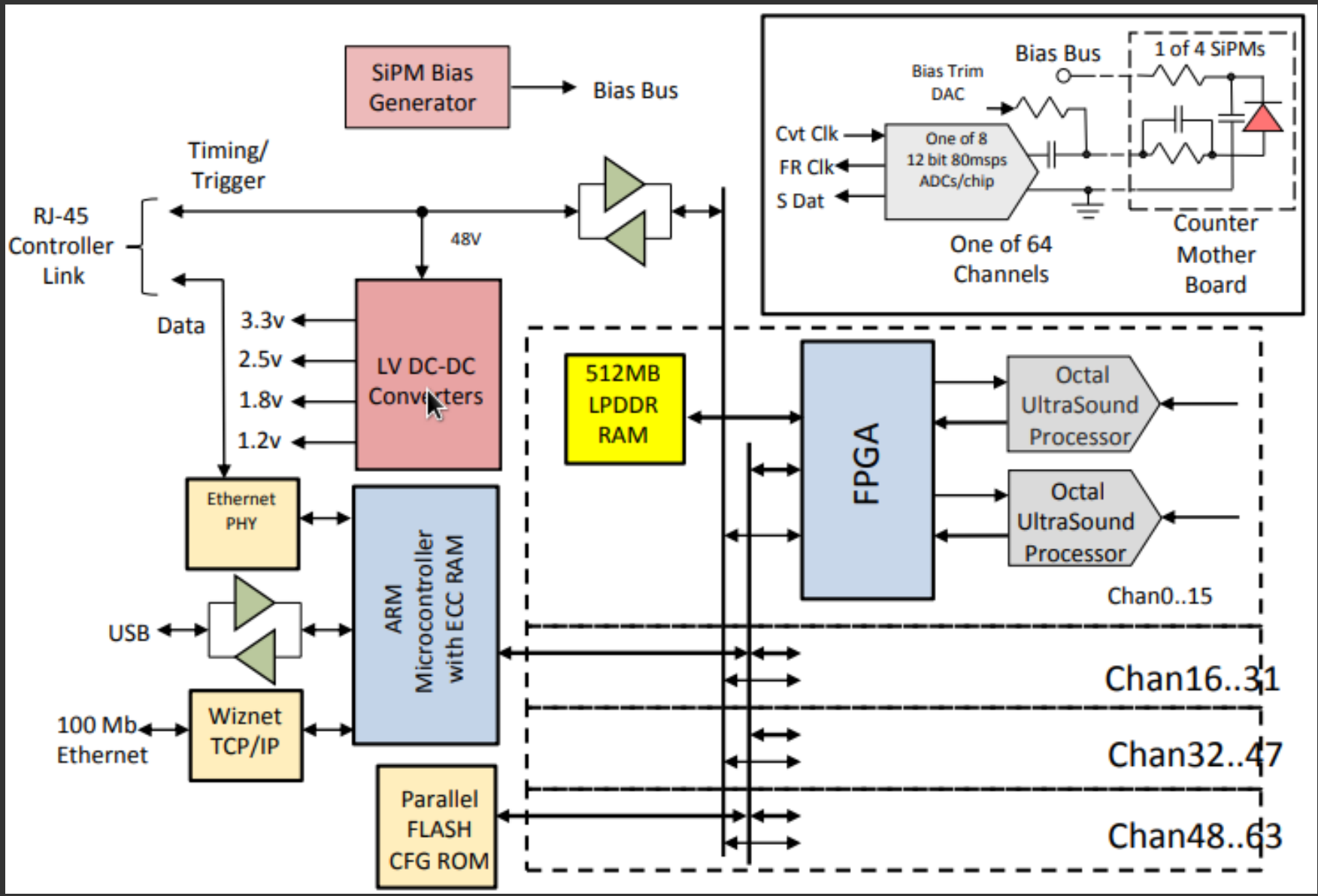
>>> Mu2e FEB



1

[1] Rubinov, 2015

>>> Mu2e FEB



[2]

[2] Rubinov, 2015

>>> Traditional SoC Architecture

1. 4 Spartan-6 (25KLUT's)
2. 2GB Ram
3. ZYNQ SoC privative CPU
4. Cortex-A9 ($\approx$ 650MHz/s $\times 2$).
5. VHDL specific modules.
6. Make use of specific privative buses
7. Just one Ethernet Gigabit port
8. + Self made Cores for ADC's and MUX

* Integrating self created cores propitiate abnormal behaviour of the system, bugs in the bus interface and malfunction of the dedicated core.
* Slow development. Lack of formal verification. Low Performance.

>>> Traditional SoC Architecture = Multiple Problems

1. No Framework for FPGA debugging Code
2. Code doesn't support different FPGA suppliers
3. Self topbench debugger that only works (partially) for a single FPGA board.
4. Very hard to get High Performance (it serialises).
5. Always dependent of privative IPCores. (RAM, Ethernet, Cordic, MUX...)
6. You can not get rid of the bad use of LUT's. (Vivado keeps high density buses even when it does not use it at all)

* 5 Years ago it was the ONLY way.
* ¿Is there any other way?

>>> Bottleneck

$$25MB/s \sim 200b/s \sim 1/5ETH_{BW}$$

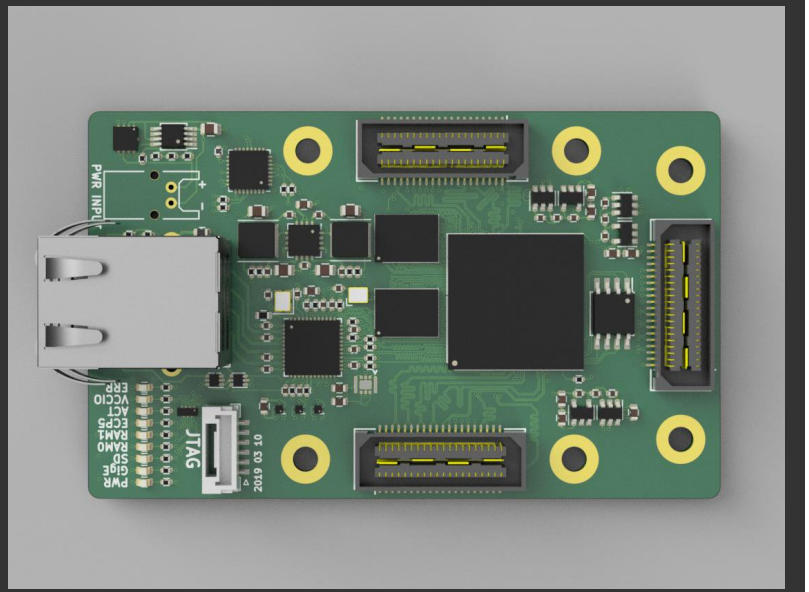
$$200b/42Gb = 4 \cdot 10^{-6}\%$$

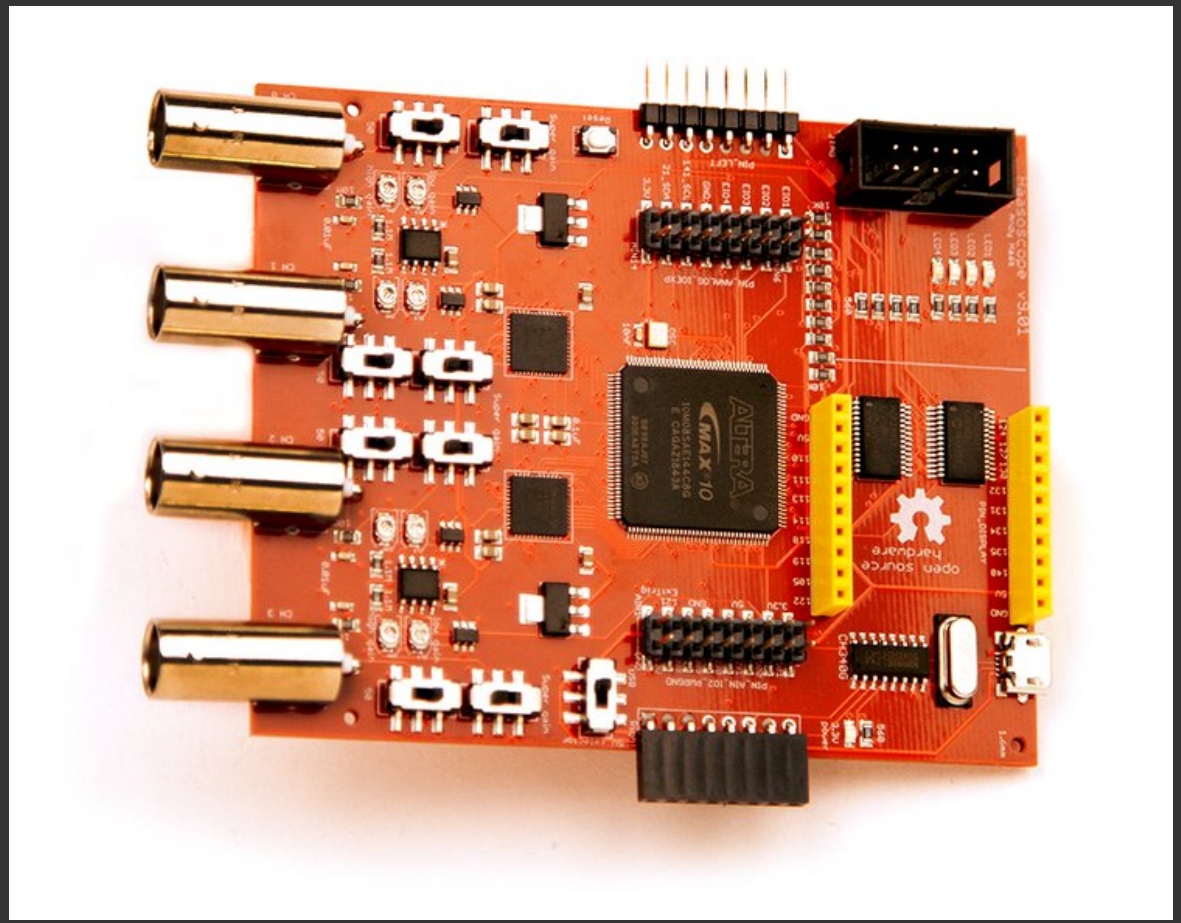
>>> Critical Route

1. ADC's need 6 tick's for 16 bit transaction.
2. At 400MHz it's 2.5 ns $\times$ 6 = 15 ns.
3. As one measure has in average 50 ticks, by Nyquist:
   1 sample $\approx$ 100 ticks or 1.5 μs $=$ 8Kb/40 channel
4. Very simple examination expends 12 ticks for the full 40 curves, enough to let recover the PD.
5. 80 ticks for MUX and send to ETH.
6. 42Gb/s ADC data Output.
7. 2 Full Ethernet give us $\approx 5\%$ per board.

>>> Architecture

34

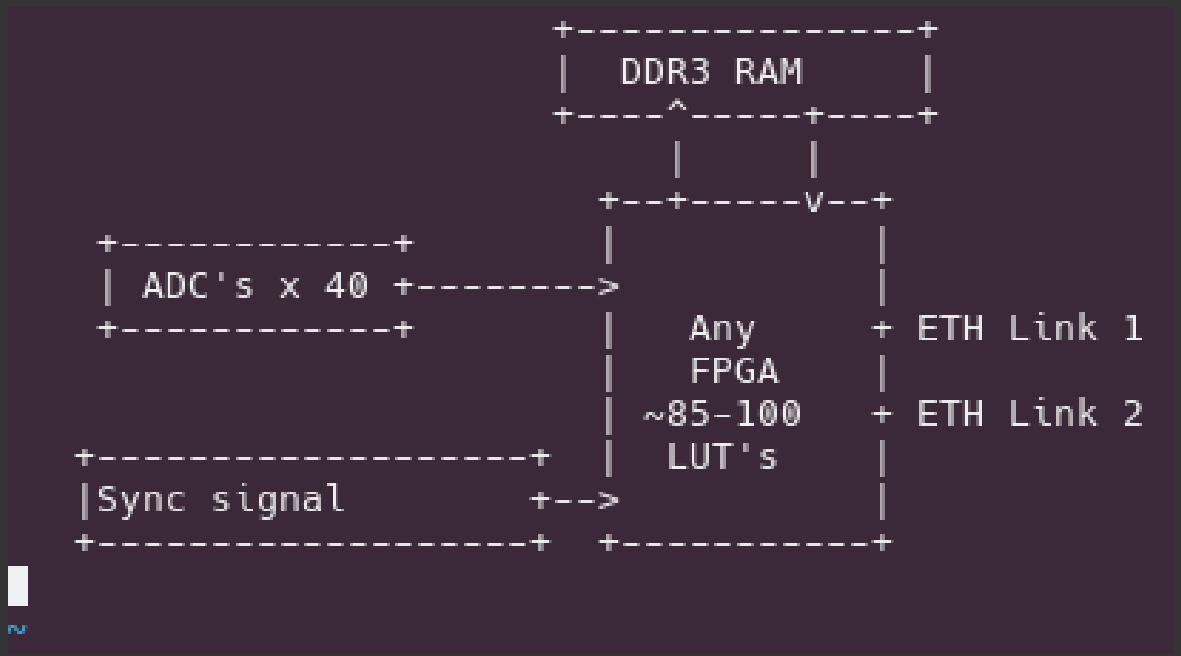




[3] https://github.com/gregdavill/ButterStick
[4] https://github.com/drandyhaas/Haasoscope

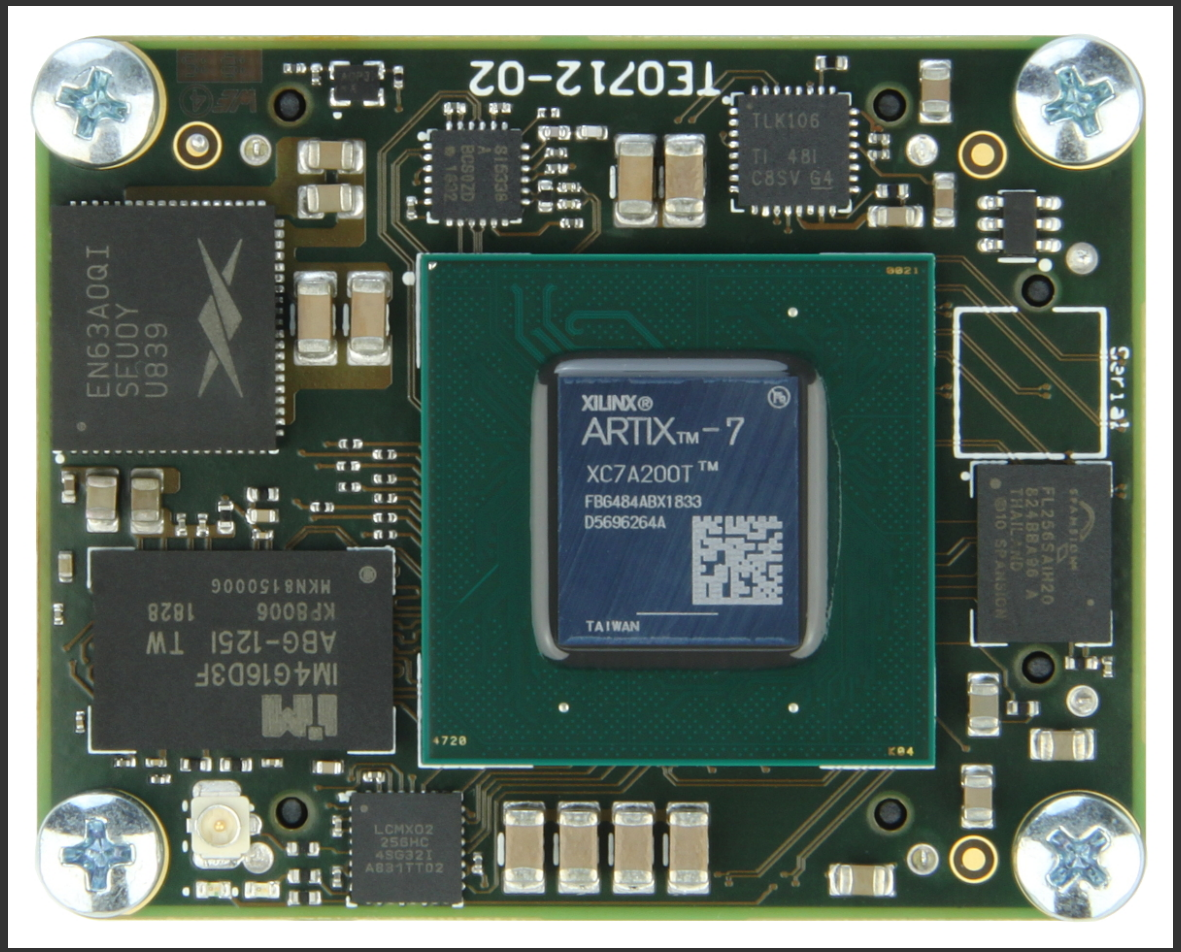
>>> Architecture

1. 1 FPGA $200K$ LUT's or 2 $85K$ LUT's.
2. 2 Ethernet Gb ports.
3. 4 HyperRAM x 64 MB.
4. Same Ultrasound AFE5807 ADC' s.



>>> Architecture

1. 1 FPGA $200K$ LUT's
2. 2 Ethernet Gb ports.
3. 1 GB RAM.
4. 260 pin count for 5 ADC's and ethernet output.



>>> Pin counting

Total pin count = 418 / 40 channels

* 5 × ADC' × 40 pin/each = 200
* 1 × ADC's General × 26/all = 26
* 4 × 64 MB hyperRAM × 24 pin /each = 96
* 2 × ETH Transceiver× 48/each = 96
* Bias Generator for DAC?

>>> FPGAs

ECP5 FPGA ≈ 70 USD $\times$ 2

* 85KLUT
* 100 - 400 MHz
* 381 pin/each

ARTY XC7A200T ≈ 200 USD $\times$ 1

* 200KLUT
* 100 - 400 MHz
* 500 pin
* same LUT/price ratio for all ARTIX family

>>> FEB Material Cost

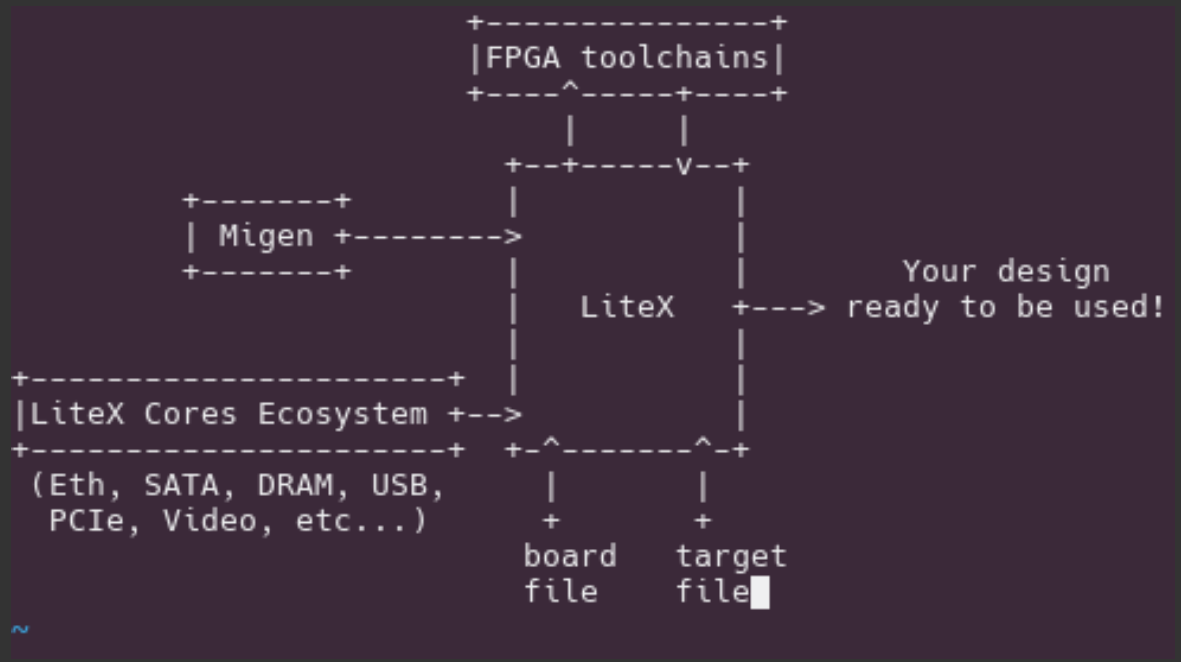
Less than 1000 USD / Board

* $\times 1$ FPGA 250 USD
* $\times 1$ DDR4 RAM 10 USD
* $\times 5$ ADC's 72 USD
* $\times 1$ PCB 200 USD
* $\times n$ others 100 USD

>>> Migen - Litex

```
 __   _ __      _  __
/ /  (_) /____ | |/_/
/ /__/ / __/ -_)>  <
/____/_/\__/\__/_/|_|
     Migen inside

Build your hardware, easily!
Copyright 2012-2019 / EnjoyDigital
```



>>> Migen - LiteX

HDL library based in Python.
Fast Prototype!

1. Total control of logic. (formal methods)
2. Easy migration to any FPGA
3. Test benches are self made but cores are reusable and logic applies for any FPGA.
4. Very easy to get High Performance.
5. You can generate just what you need (very optimized use of FPGA capability)

>>> Migen - LiteX

HDL library based in Python.
Fast Prototype!

1. Total control of logic. (formal methods)
2. Easy migration to any FPGA
3. Test benches are self made but cores are reusable and logic applies for any FPGA.
4. Very easy to get High Performance.
5. You can generate just what you need (very optimized use of FPGA capability)

   * 5 Years of development. Full use of hardware capabilities.
   * We can make use of formal verification.