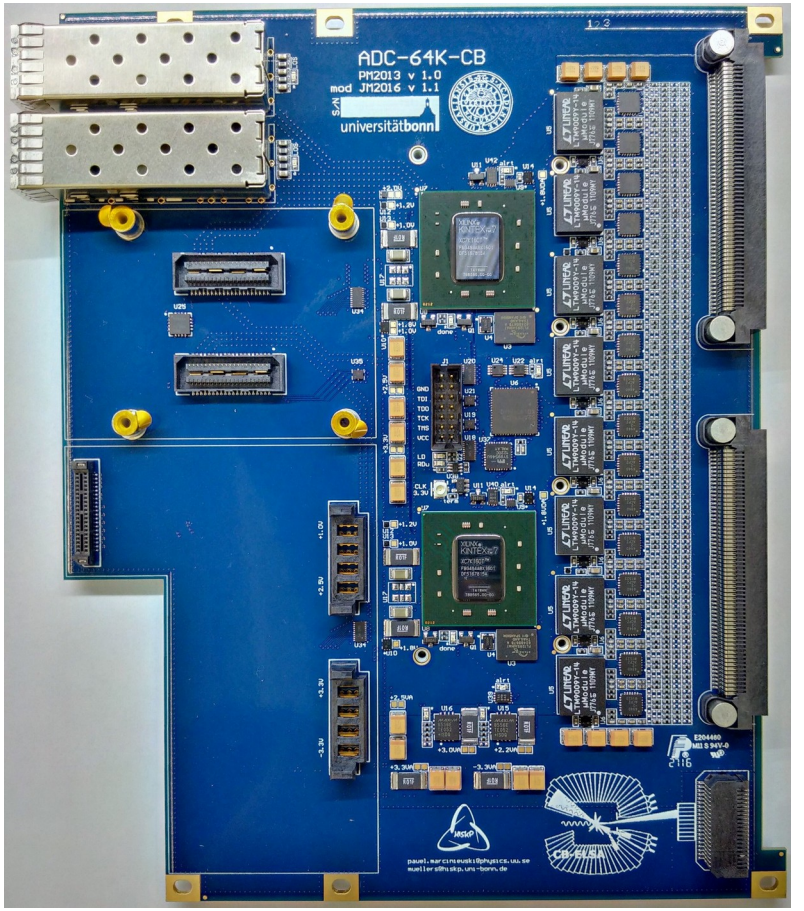
Ethernet-based Firmware for Lab-use of PANDA-EMC-SADC

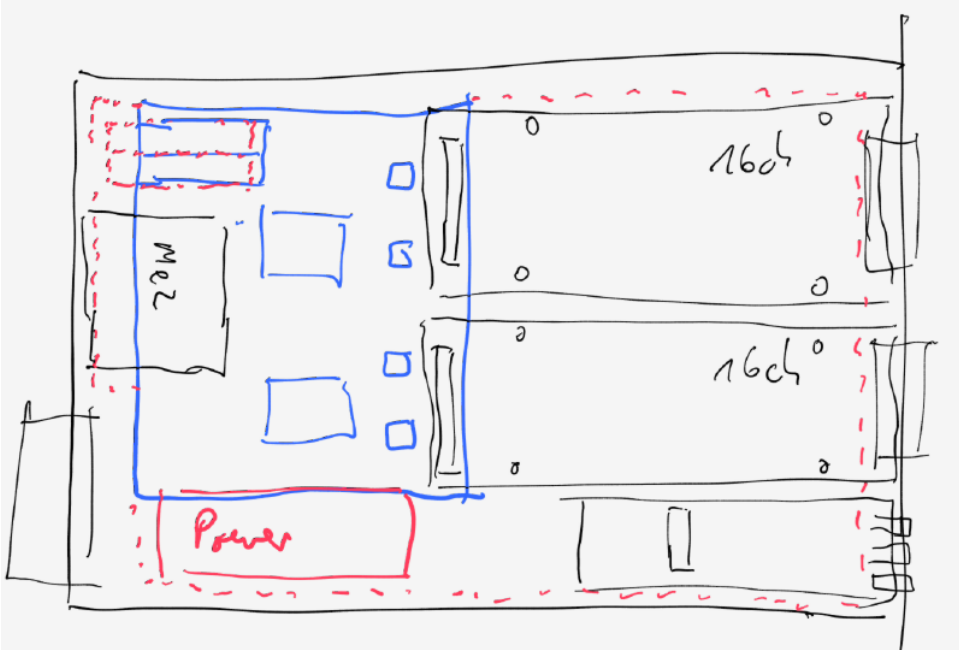
Johannes Müllers
Helmholtz-Institut für Strahlen- und Kernphysik
Universität Bonn



Contributions to CB-SADC project:
Timo Poller
Christoph Schmidt
Jan Schultes
Ulrike Thoma
Georg Urff

and Pawel Marciniewski (Uppsala)

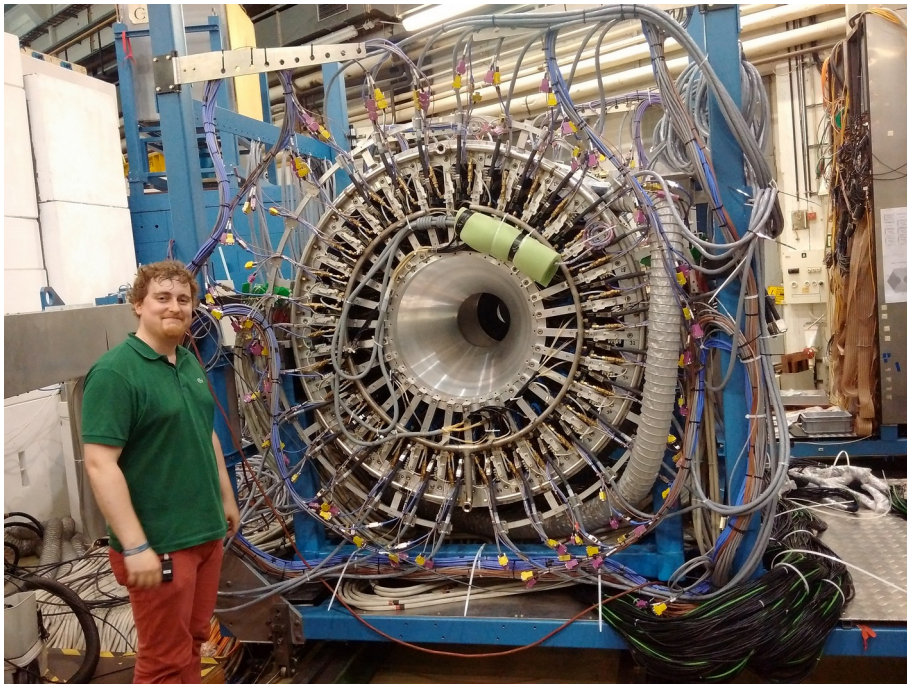
Introduction: CB-SADC



Artist: Christoph Schmidt

CB-SADC - History

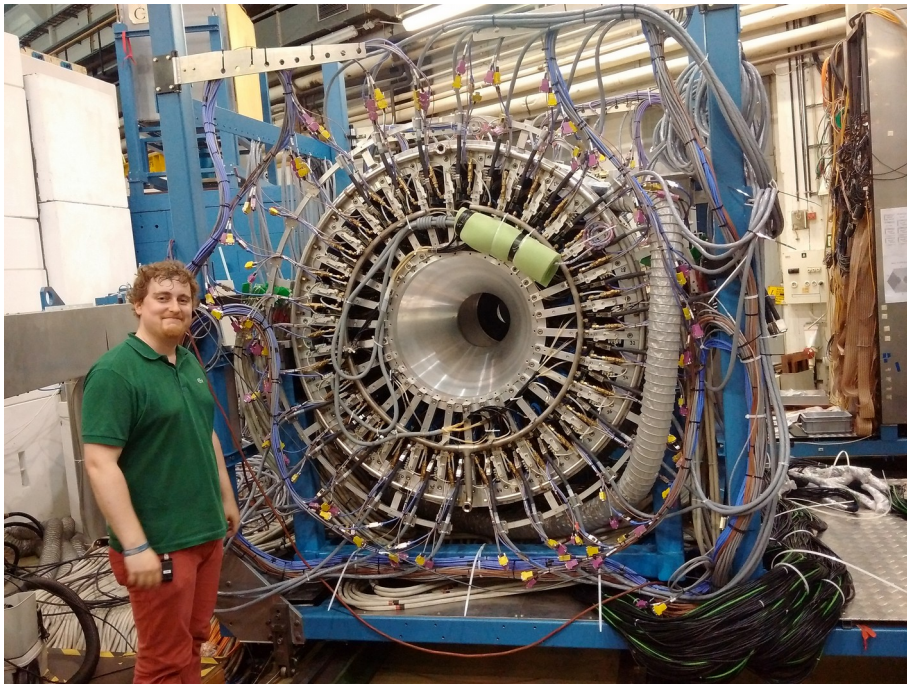
- CBELSA/TAPS experiment in Bonn (Baryon spectroscopy)
- EM Calorimeter ("Crystal Barrel") with Fastbus QDC
  → Limitation 2kHz, no pile-up correction, run-based pedestal



CB-SADC - History

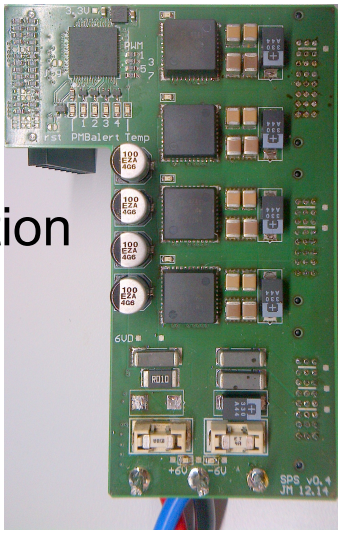
- CBELSA/TAPS experiment in Bonn (Baryon spectroscopy)
- EM Calorimeter ("Crystal Barrel") with Fastbus QDC
  → Limitation 2kHz, no pile-up correction, run-based pedestal
- 2014 (spring): First test with PM‘s 16-ch prototype
- 2014 (autumn): Customization of PM‘s 64-ch prototype
- 2015: First prototype (1 pc)
- 2016: Second prototype (4 pc)

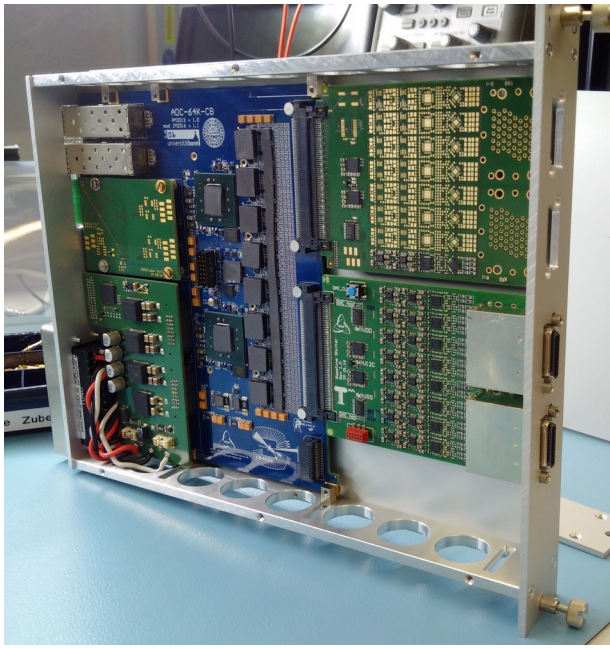
Students so far on the project:
1 PhD, 2 Master, 1 Bachelor

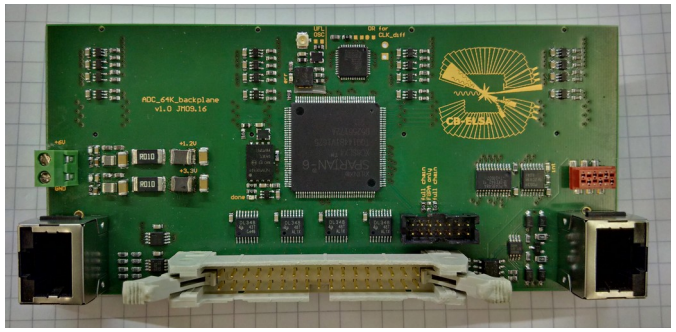


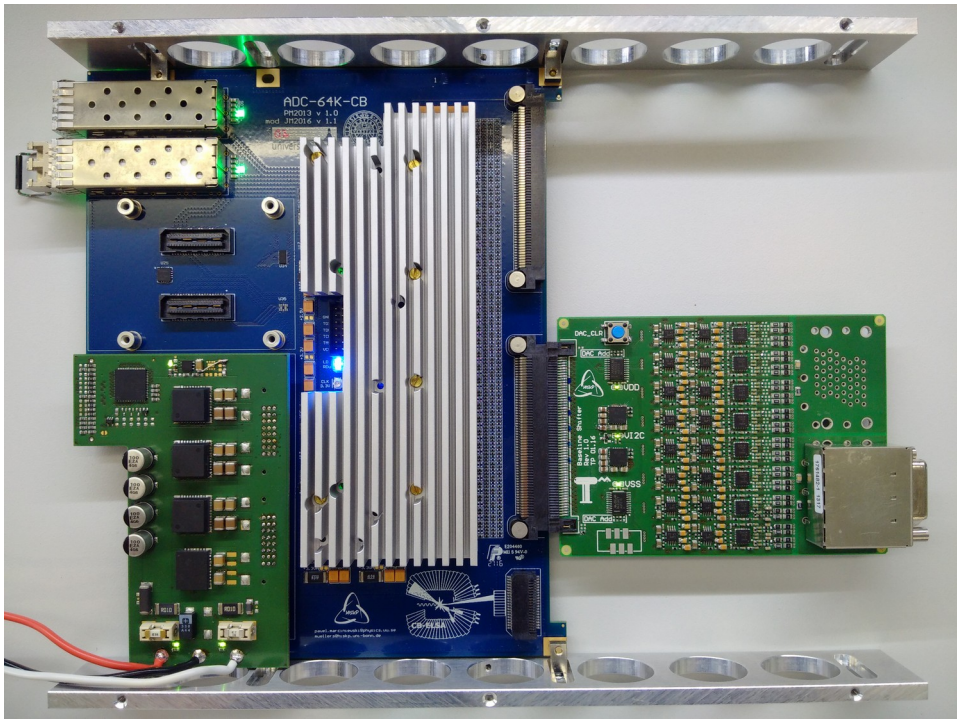
CB-SADC vs. PANDA-SADC

- NIM Formfactor
- Pluggable analog front-end
  - I²C adjustable pole-zero compensation
  - I²C adjustable baseline shifter
  - Shaping (~1µs)
- Pluggable Power-Supply
- Backplane
  → Trigger, Slowcontrol, Clock, JTAG









BUT: core nearly untouched!

CB-SADC Backend

- 2 NIM Crates with 12 SADCs each
  → 1536 channels (CB: 1320 + Energy sums)
- 1x Gigabit Ethernet (Copper/Fiber)
  per 32 (or 64) channels
- COTS switch with 1..4 uplinks to Saver
- Sampling mode*: 2.7GB/s per kHz
  Feature ext. mode: 23MB/s per kHz
  → Sampling only for pile-up events
- Central Trigger → Push architecture
- Event building: Offline

(*) 1024x16bit samples per channel

Firmware Development

hackread.com

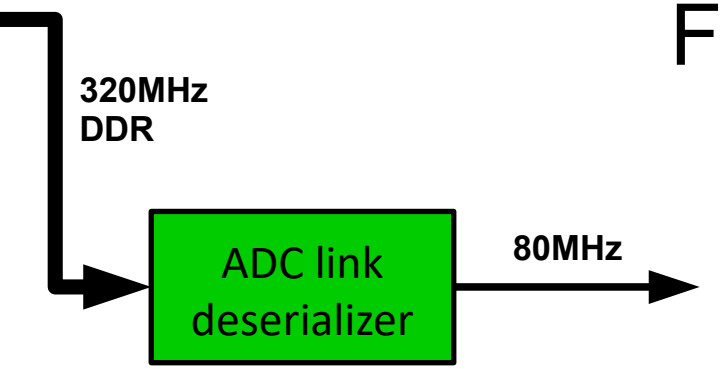
Firmware - History

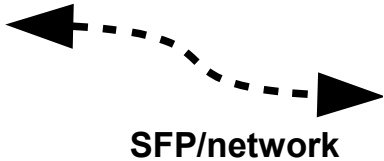
- 2014 (16-ch Virtex 5 prototype)
  - PM's firmware for lab test (debugging with JTAG)
  - Add feature extraction (baseline, integral, CFD)
  - Add network: Virtex 5 embedded MAC core

Firmware - History

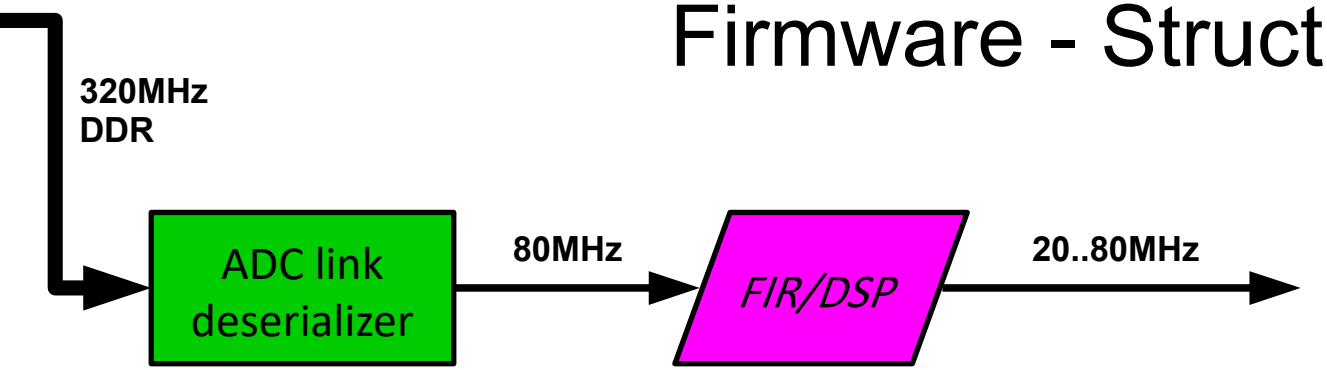
- 2014 (16-ch Virtex 5 prototype)
  - PM's firmware for lab test (debugging with JTAG)
  - Add feature extraction (baseline, integral, CFD)
  - Add network: Virtex 5 embedded MAC core
- 2014..2016 (64-ch Kintex 7 prototype)
  - PM's firmware for lab test (debugging with JTAG)
  - Add feature extraction (baseline, integral, max, CFD)
  - Moved from ISE to Vivado ("Teufel mit dem Beelzebub austreiben")
  - Add network:
    - Kintex 7 PCS/PMA (free Xilinx IP)
    - Kintex 7 TEMAC (commercial Xilinx IP → **bought project license**)
    - 1G eth UDP / IP Stack (opencores.org, BSD license)
  - Add FIR filter (decimation, low-pass)

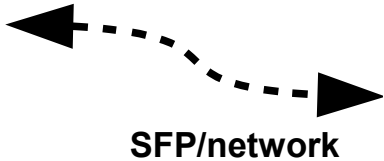
Firmware - Structure



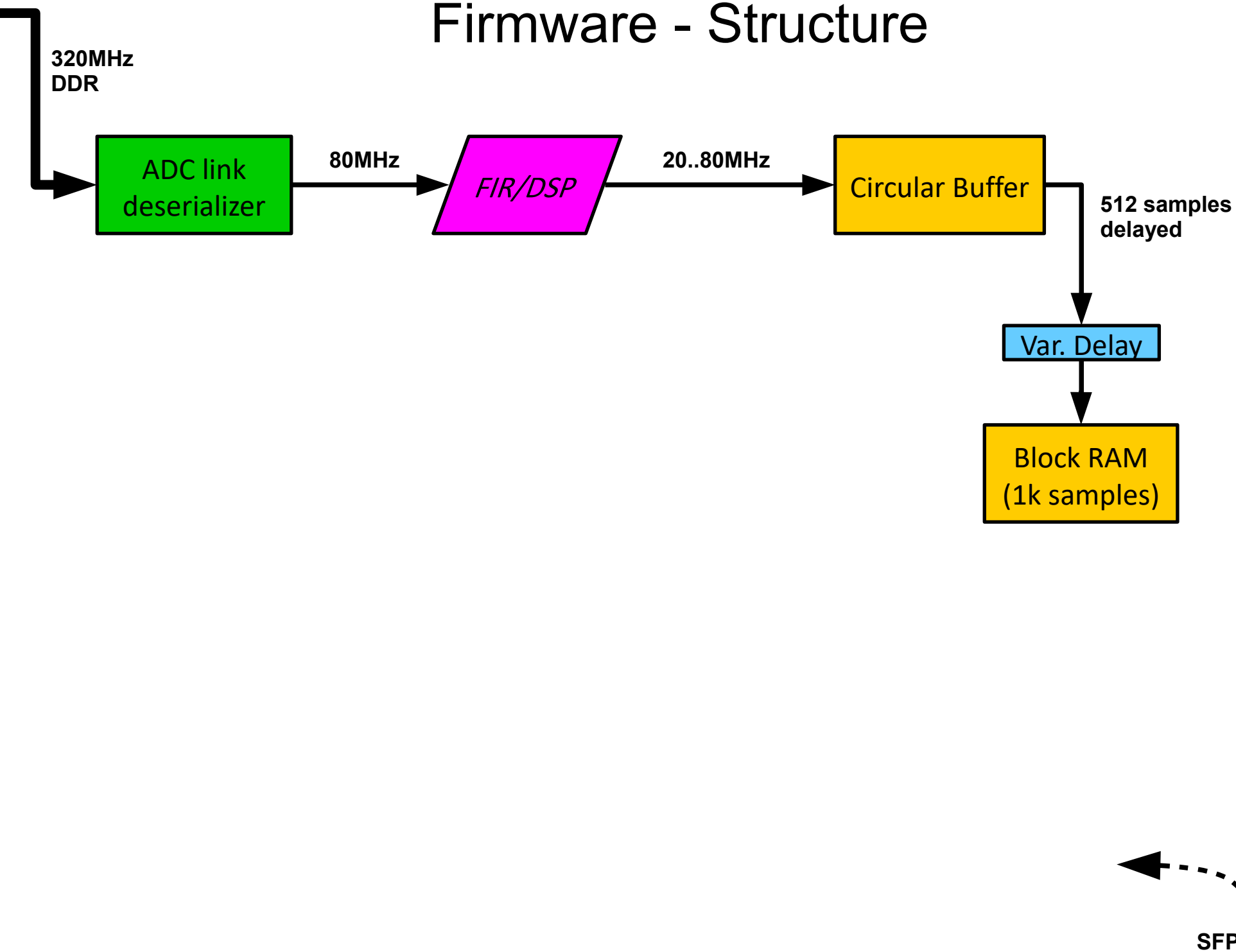


Firmware - Structure

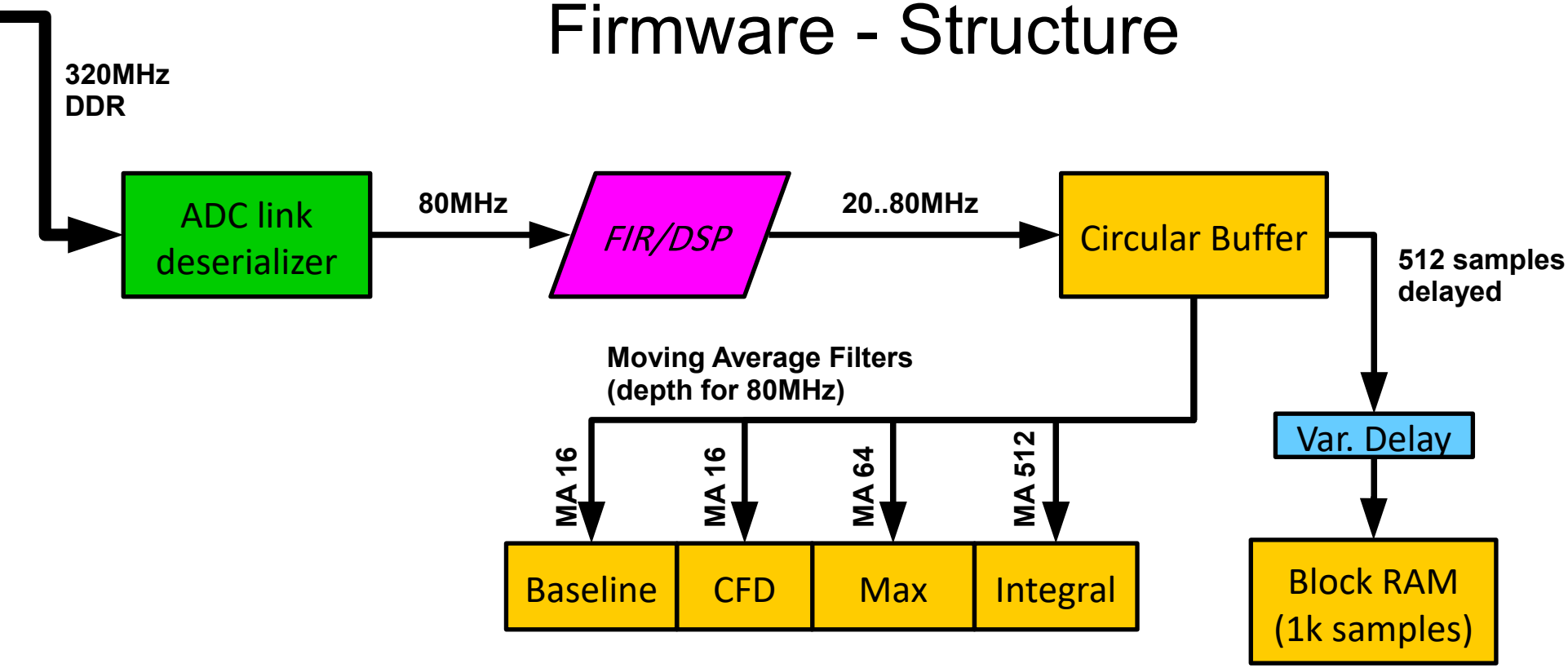


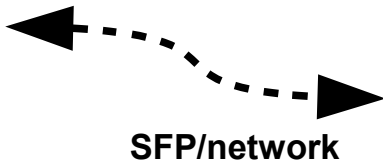


Firmware - Structure

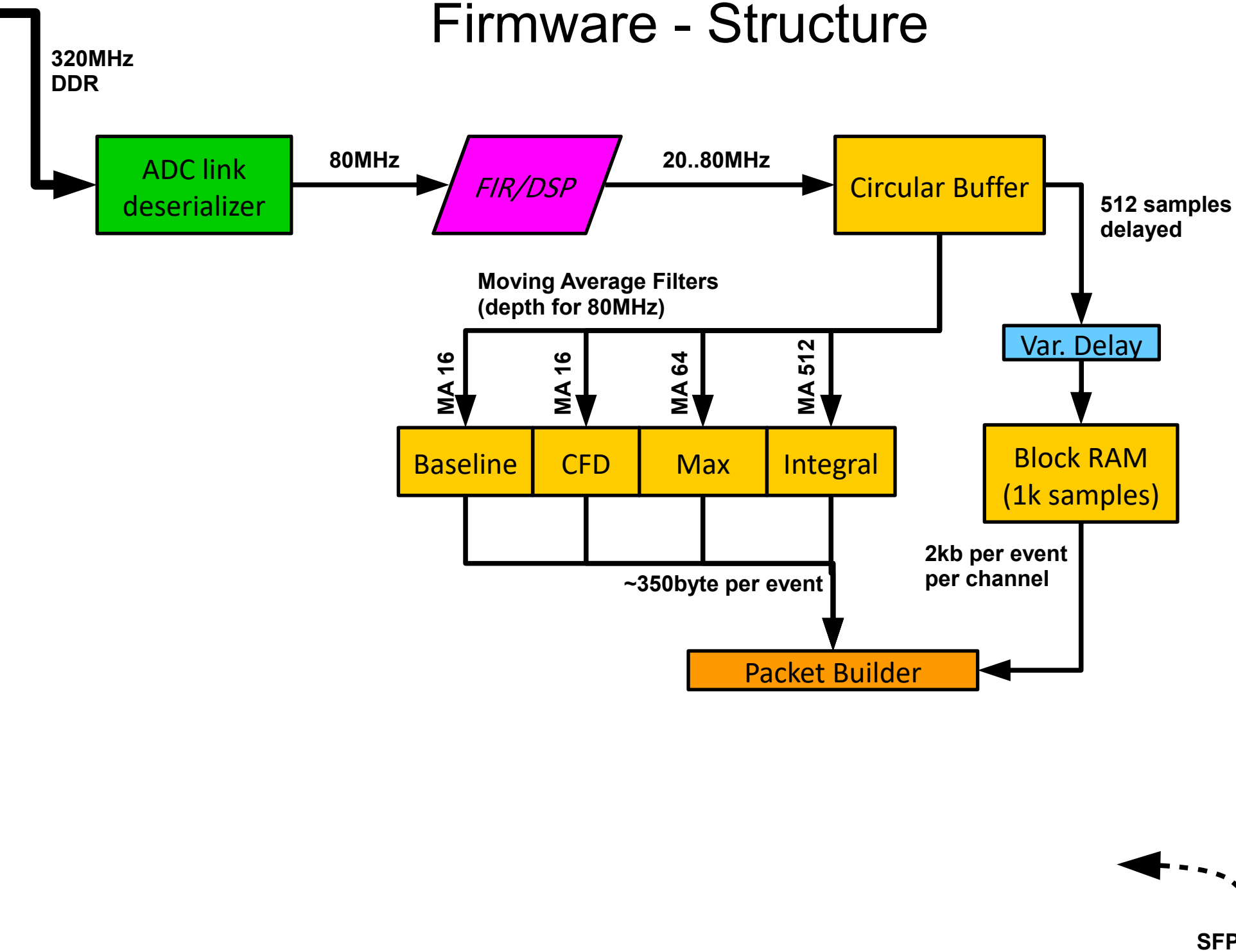


Firmware - Structure

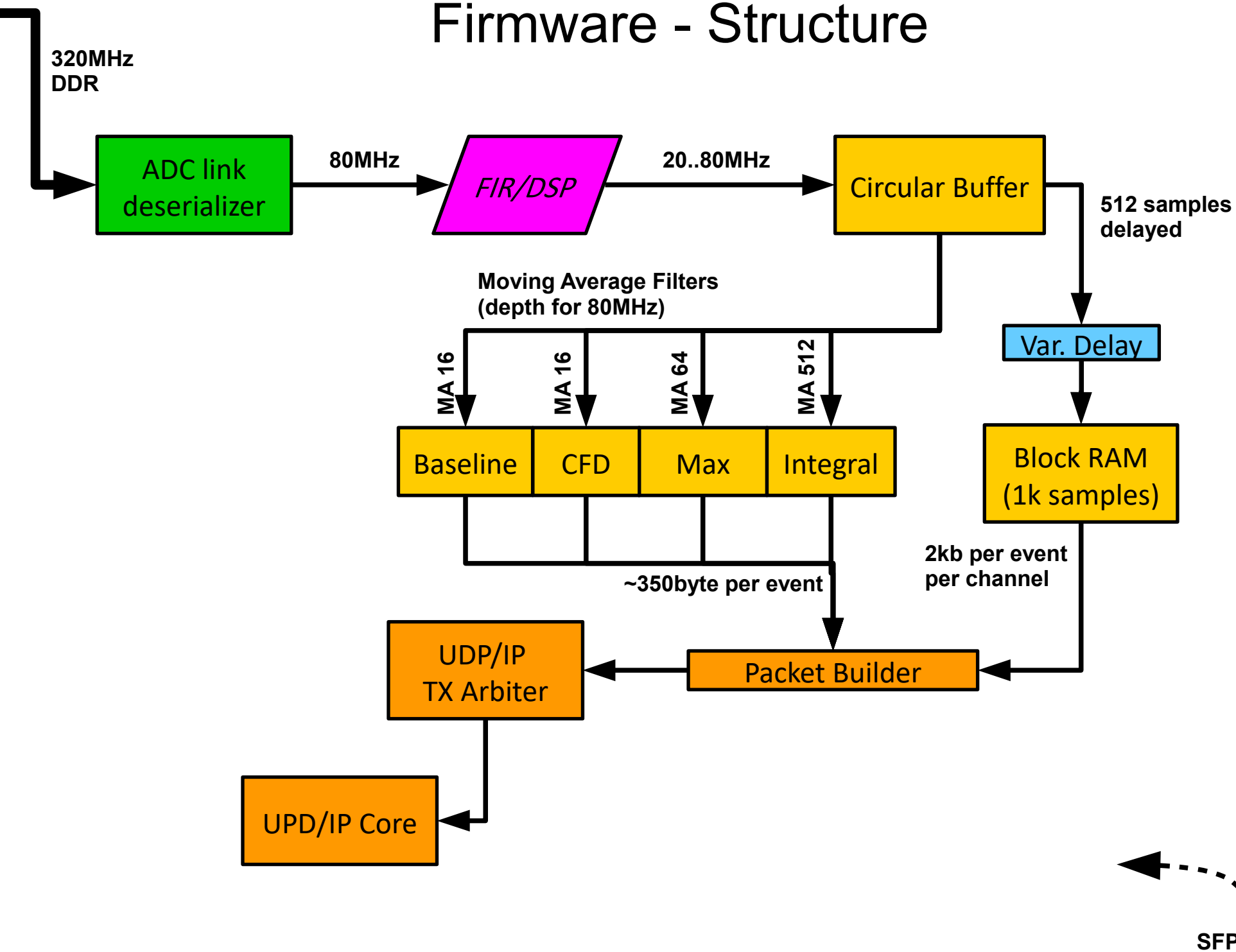




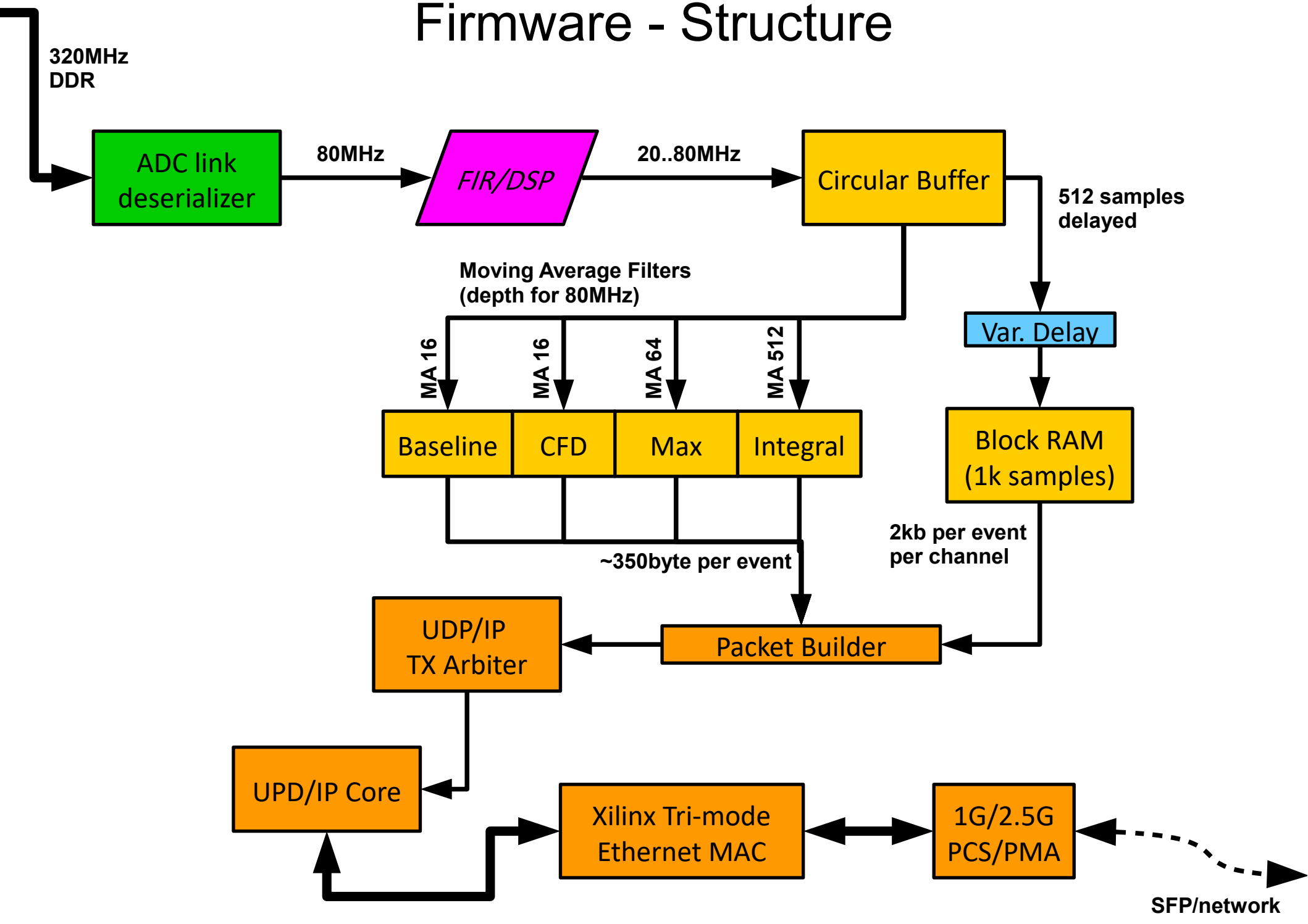
Firmware - Structure



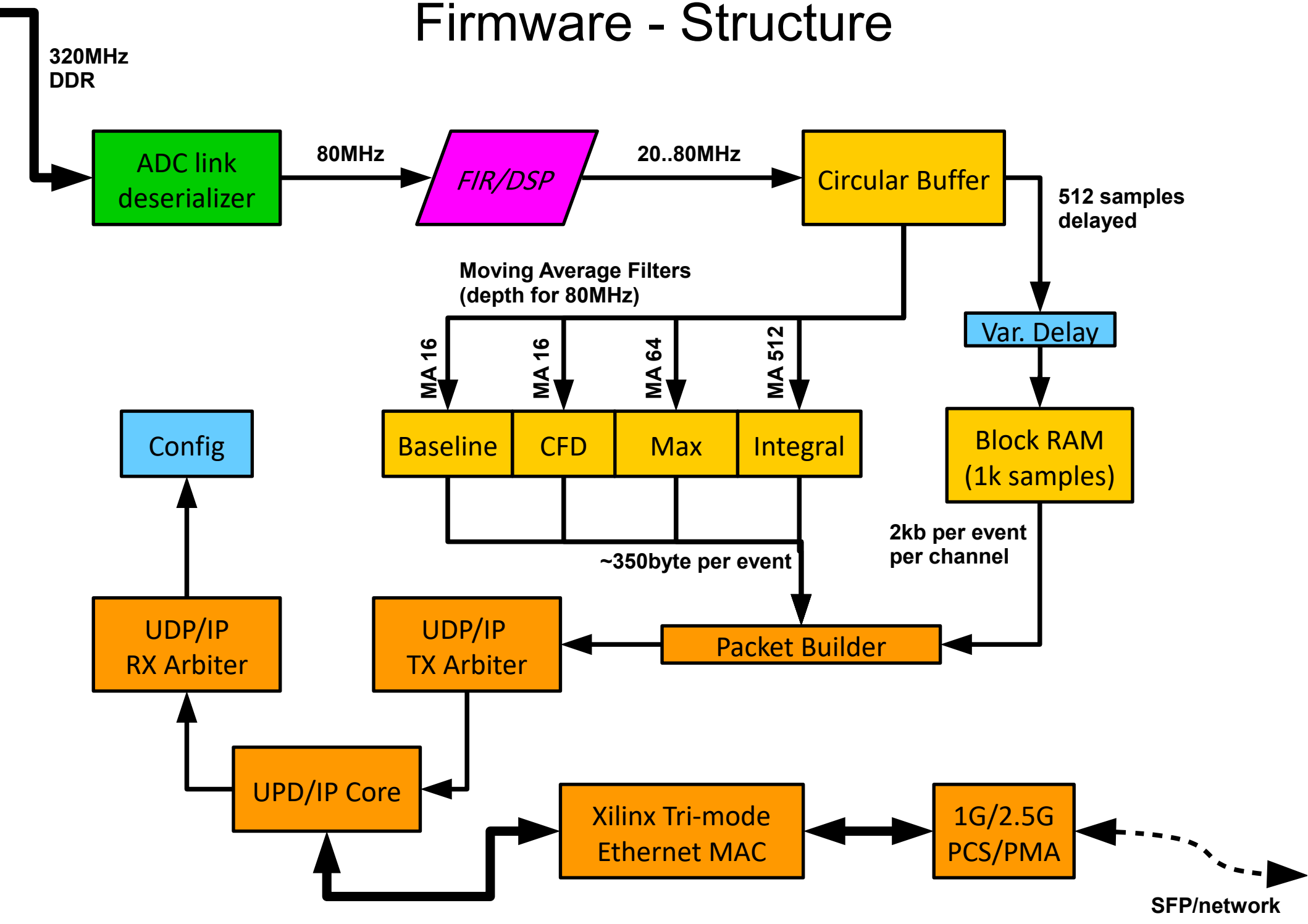
Firmware - Structure



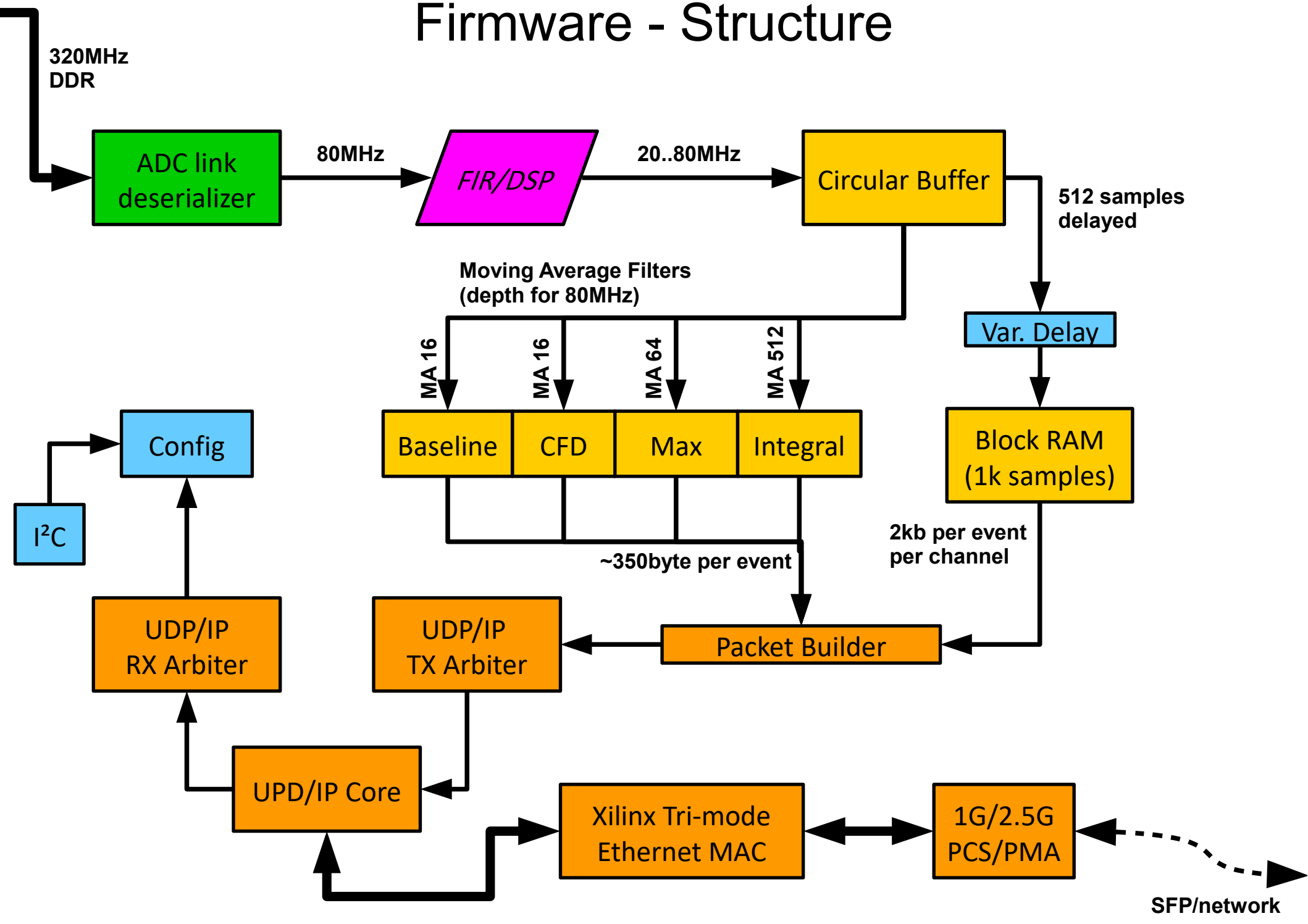
Firmware - Structure



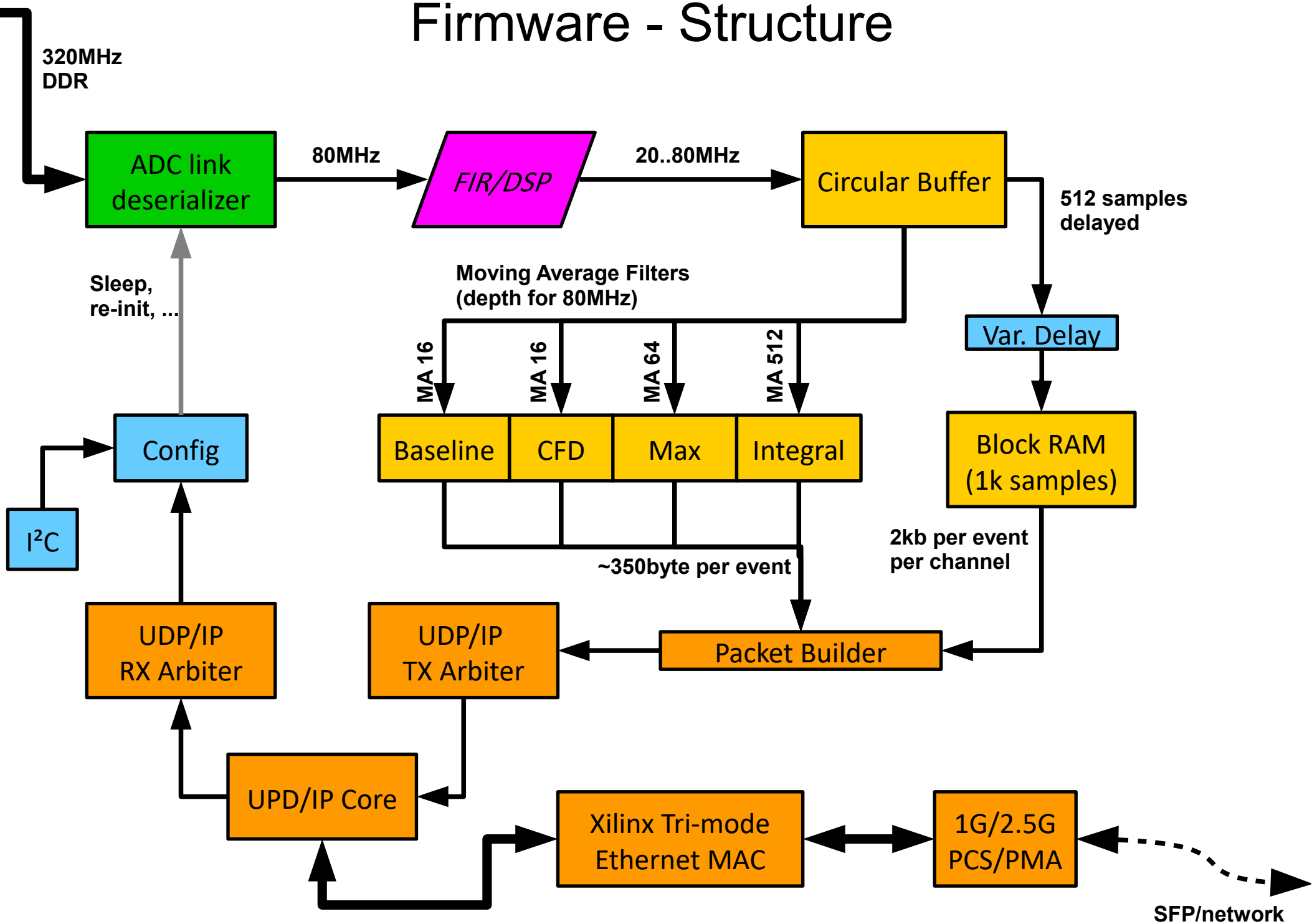
Firmware - Structure



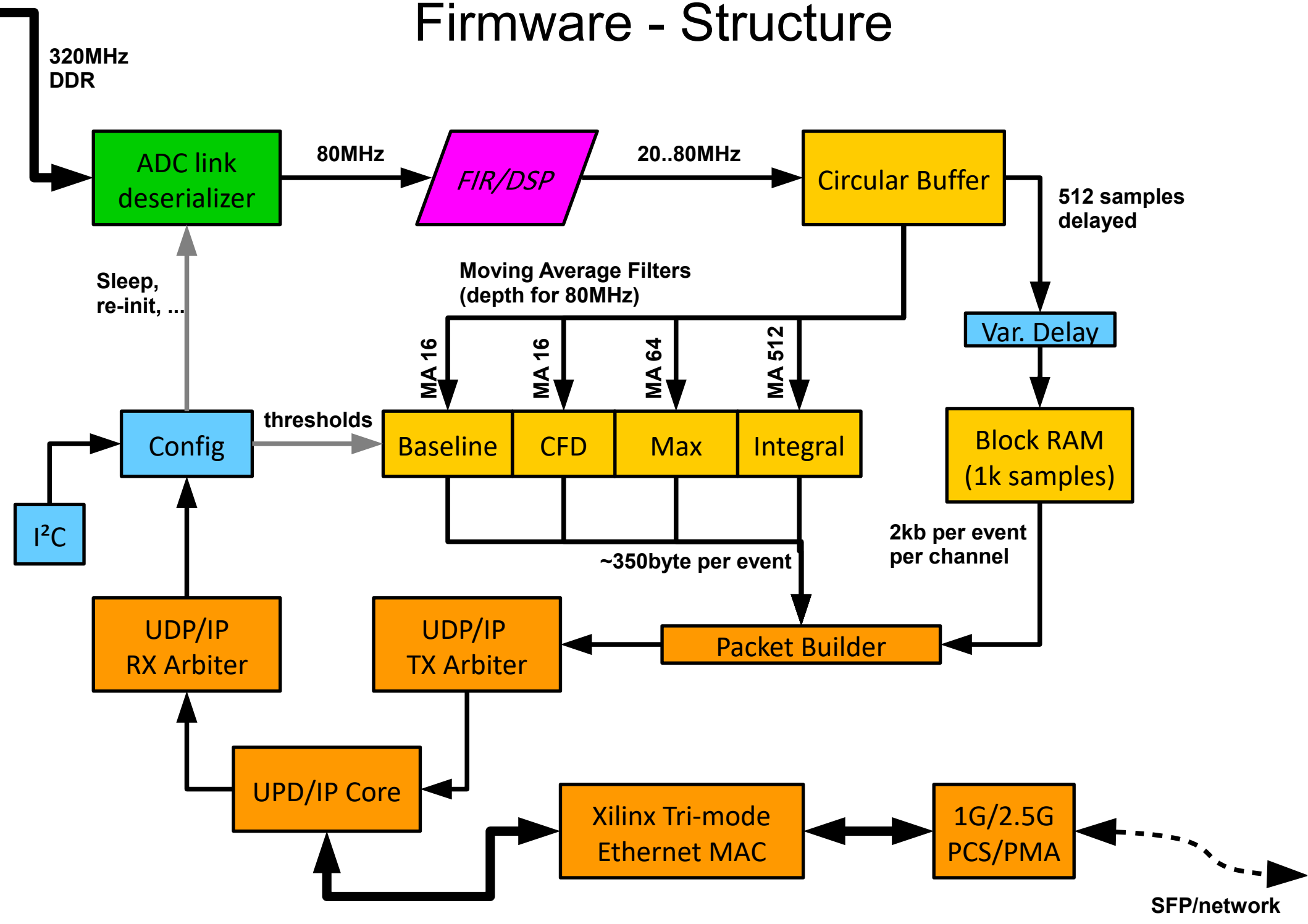
Firmware - Structure



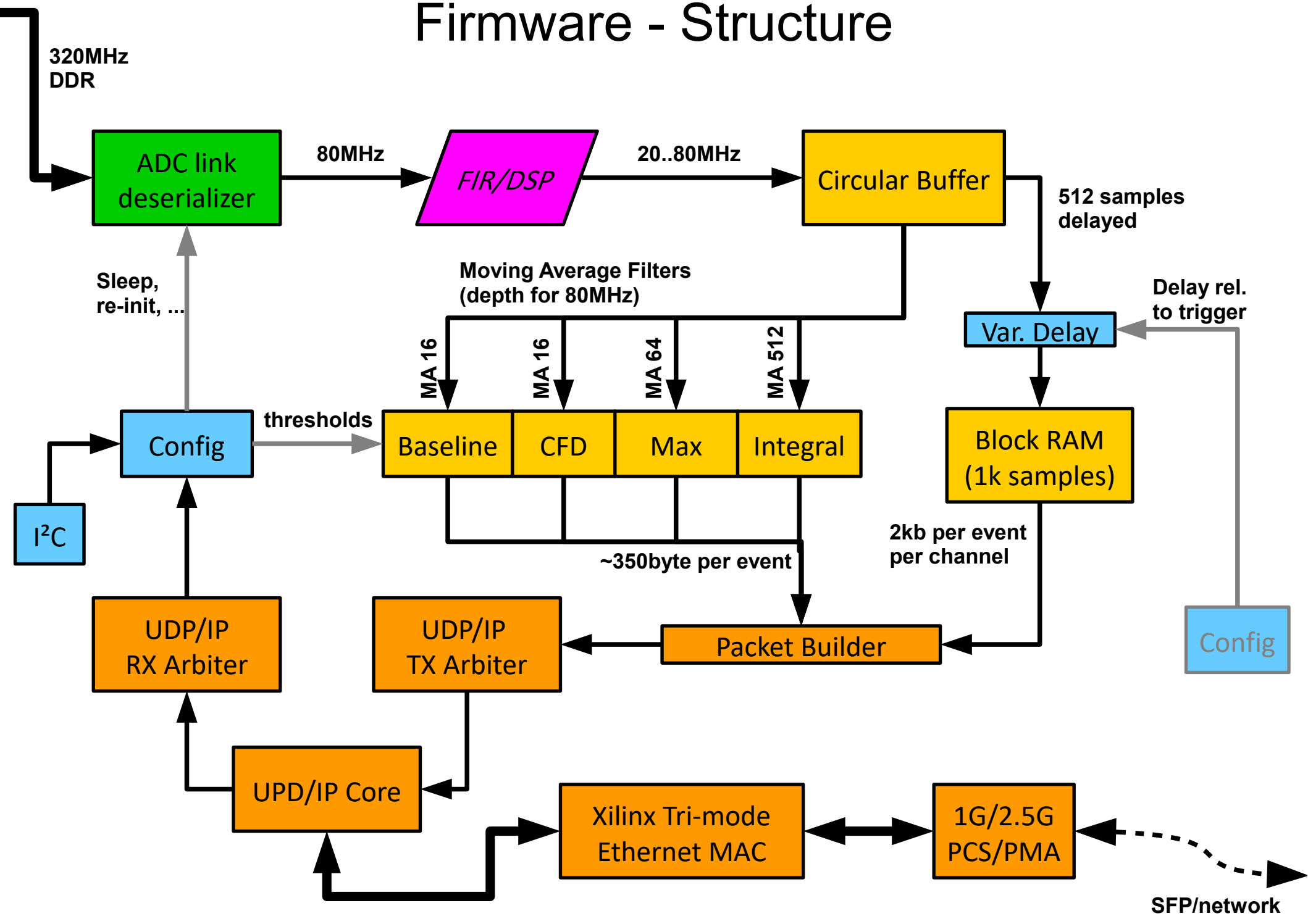
Firmware - Structure



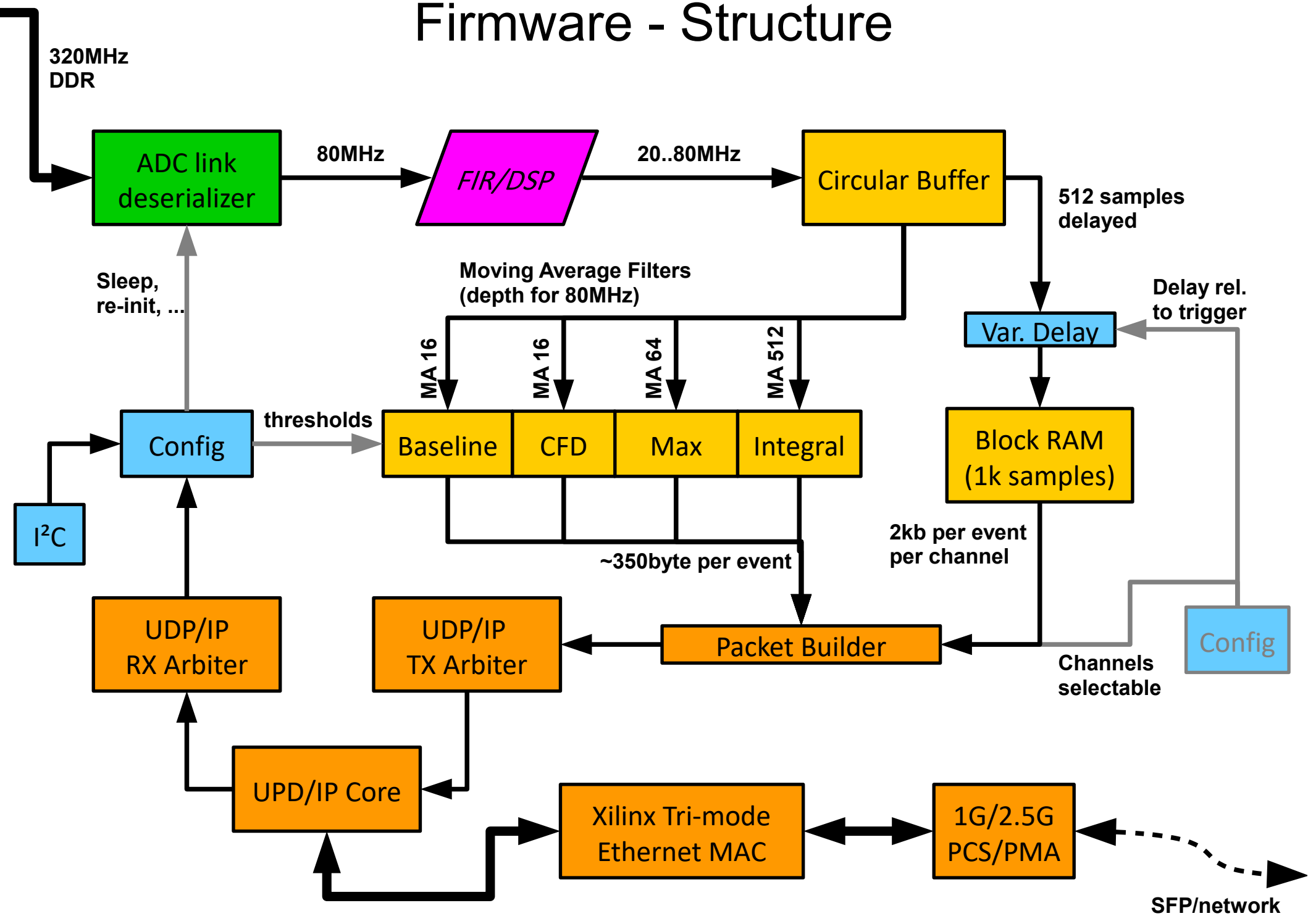
Firmware - Structure



Firmware - Structure



Firmware - Structure



Firmware - Agenda

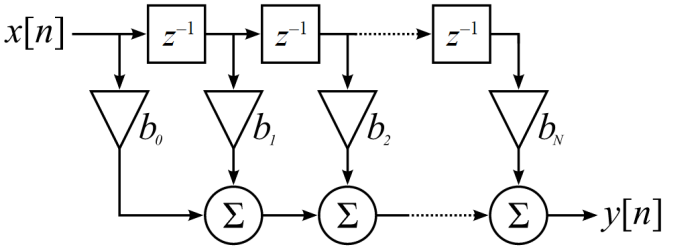
- Investigate FIR filters
- Switch from 1000Base-SX to 10/100/1000Base-T
  → keep both
- Implement Event-Buffering → no bare push-architecture
- Identification of SADC / automatic MAC and/or IP:
  use *FUSE_DNA* and/or *FUSE_USER*

Firmware - Agenda

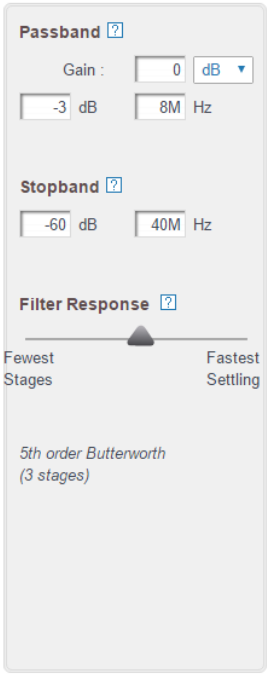
- Investigate FIR filters
- Switch from 1000Base-SX to 10/100/1000Base-T
  → keep both
- Implement Event-Buffering → no bare push-architecture
- Identification of SADC / automatic MAC and/or IP:
  use *FUSE_DNA* and/or *FUSE_USER*
- **Refactor Firmware**
  - AXI-Stream everywhere
  - Buffer-Tapping / MA more flexible
  - Make everything nicer / readable / ...

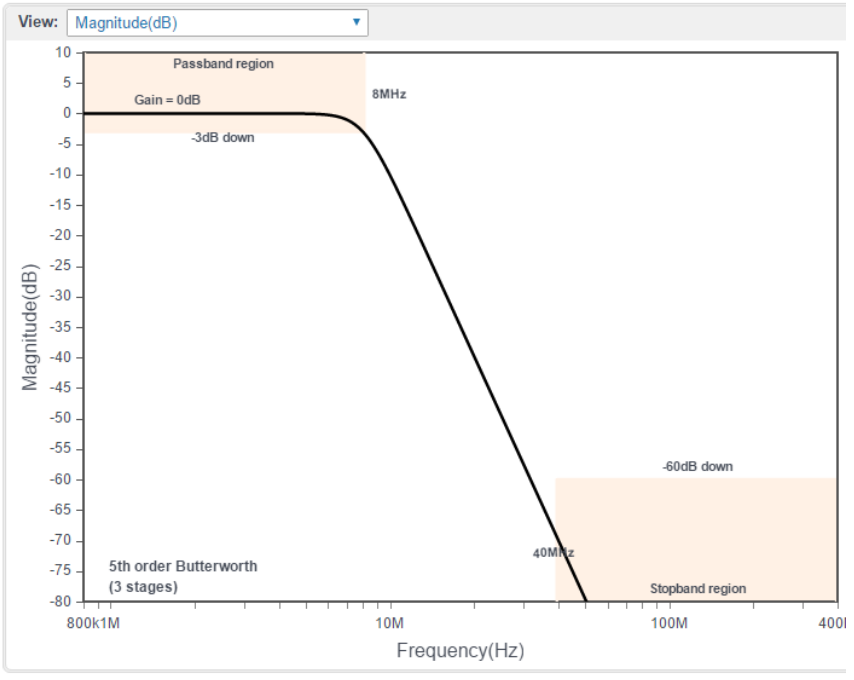
Firmware - FIR Filter

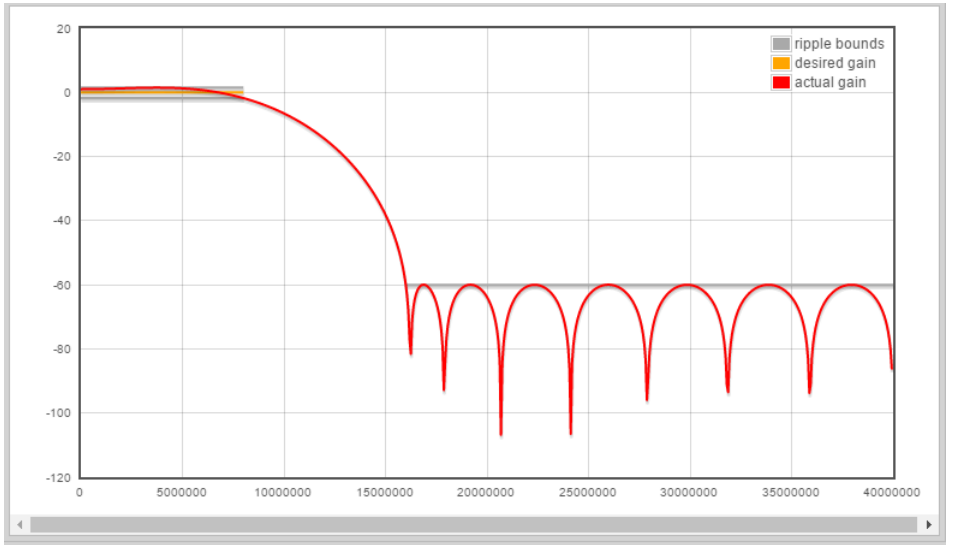
- Finite Impulse Response
- Kintex 7: 600 DSP cores (18/channel)

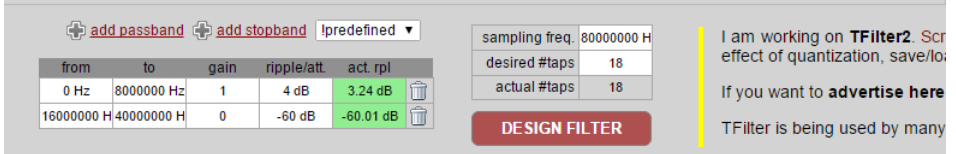


	Analog Filter	FIR Filter
Bode plot	Passband: flat Passband: -x dB/Oct.	Passband: flat + ripple Transition band: steep Stoppband: flat + ripple
-60dB@ 16MHz	5th/10th order Butterworth	18DSP slices/ch







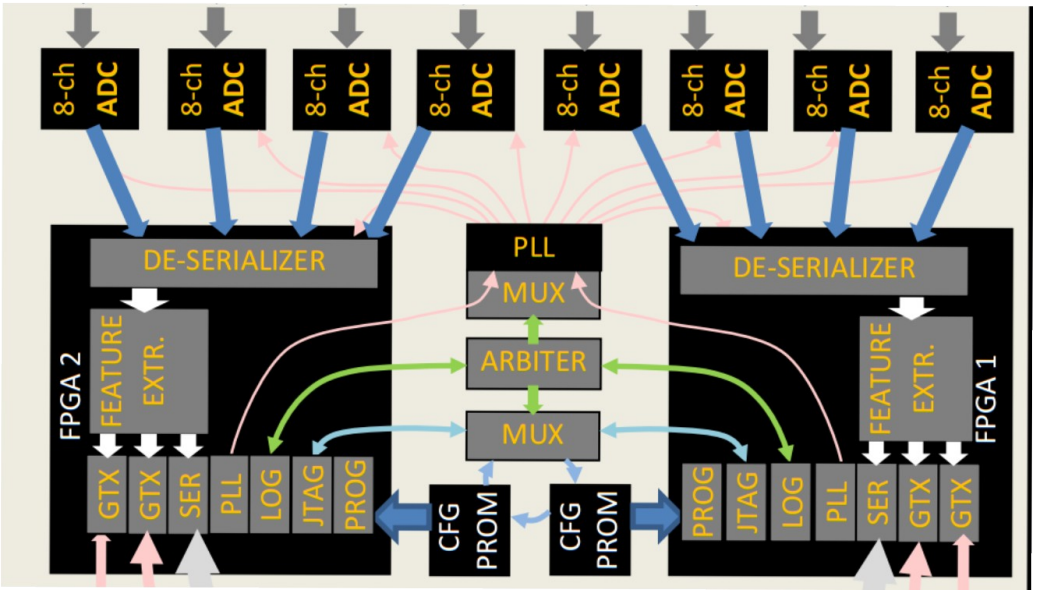


Compatibility with PANDA SADC

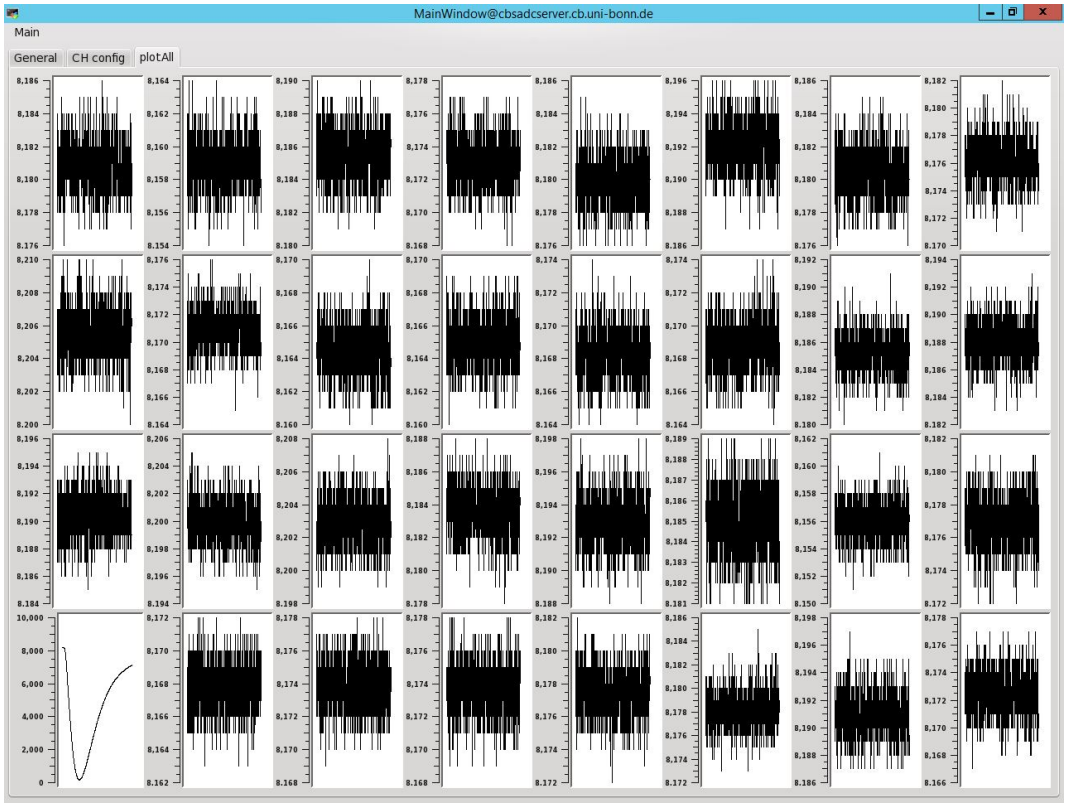
- FIR/DSP can easily be stripped from the design to keep full bandwidth for PANDA lab firmware

Compatibility with PANDA SADC

- FIR/DSP can easily be stripped from the design to keep full bandwidth for PANDA lab firmware
- Core design of SADC left nearly untouched except for arbitration circuit
  - not critical for lab test
  - Problems observed for ROM bootup (Luigi)
    → has to be investigated



QT Tool


MainWindow@cbsadcserver.cb.uni-bonn.de
Main
General
CH config
plotAll

C++/Qt SADC tool

- Framework: Qt and Qwt
- Communication with SADC over UDP/IP
- Easy handling of UDP datagrams
- Waveform Liveview
- Dump to disk

Partly recycled by

- Christoph Schmidt (CB-SADC LEVB)
- Matthias Steinke (Lab tests Bochum)

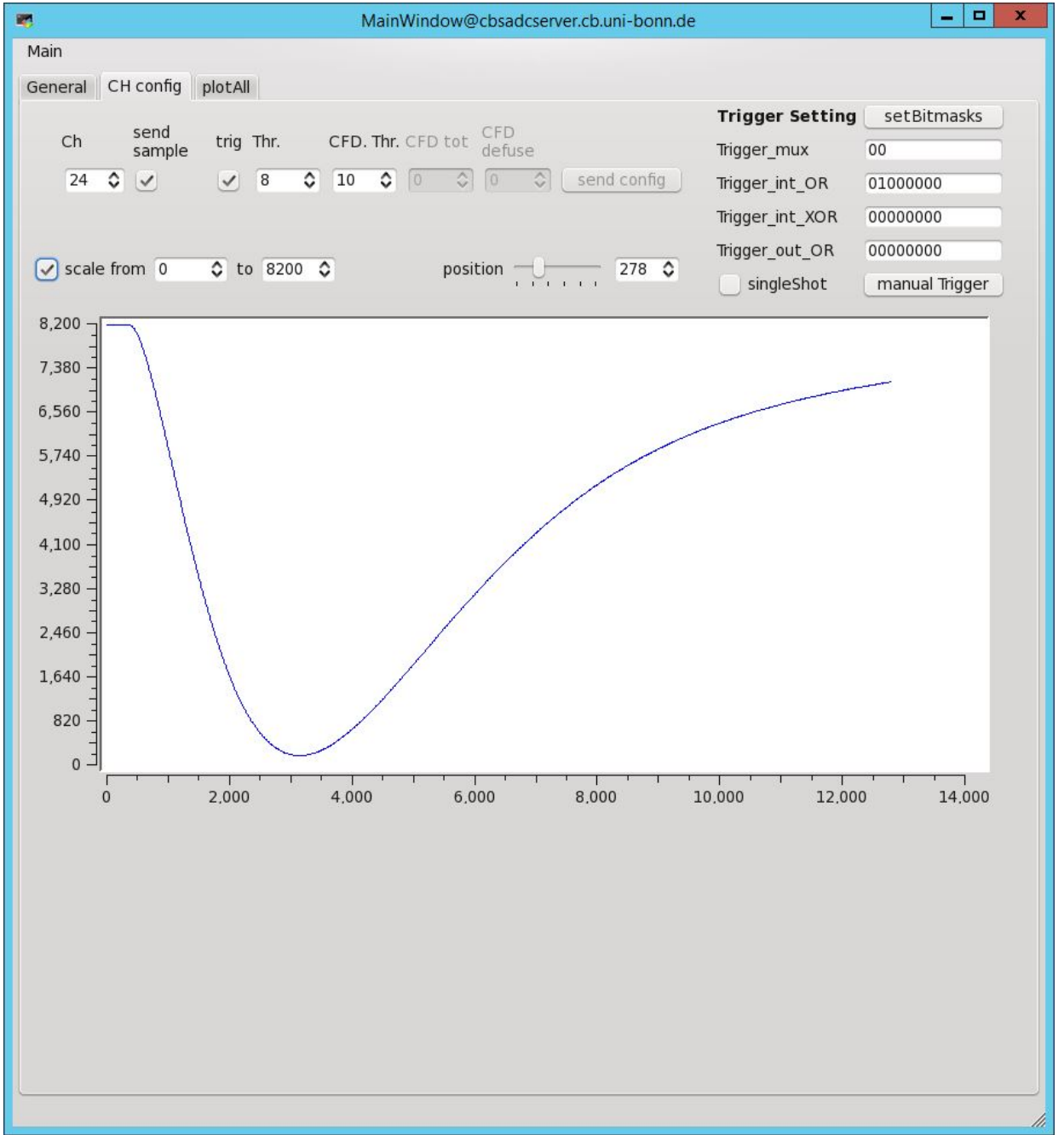
```
// EVENT BUILDER
class SADC_event_builder: public QObject
{
Q_OBJECT
  public:
  SADC_event_builder();
  ~SADC_event_builder();
  bool addFEPacket(const QByteArray & dataFE);
  bool addSamplePacket(const QByteArray & dataSP);
  vector<QByteArray> fullSample;
private:
  QByteArray featureExtraction;
  unsigned int previousEventID;
  unsigned int currentEventID;
  unsigned int currentExpectedSamples;
  bitset<NR_OF_CHANNELS * NR_OF_SAMPLE_PARTS> currentExpectedSampleParts;
signals:
  void eventComplete(int eventsComplete);
  void eventIncomplete(int eventsIncomplete);
  void samplesComplete();
};

// Wrapper for Feature Extraction packet
class SADC_FE_wrapper
{
public:
  SADC_FE_wrapper(const QByteArray & dataFE);
  ~SADC_FE_wrapper();
  unsigned int getCompilationDatecode() const;
  unsigned int getEventID() const;
  long double getTimestampTrigger() const;
  unsigned int getExpectedSamples() const;
  double getTimestampCFD(int channel) const;
  double getTimestampMax(int channel) const;
  double getBaseline(int channel) const;
  double getIntegral(int channel) const;
  double getMax(int channel) const;
private:
  const QByteArray& m_data;
};

// Wrapper for sample packet
class SADC_SP_wrapper
{
public:
  SADC_SP_wrapper(const QByteArray & dataSP);
  ~SADC_SP_wrapper();
  unsigned int getEventID() const;
  unsigned int getChannel() const;
  unsigned int getPart() const;
  const QByteArray extractSample() const;
private:
  const QByteArray& m_data;
};
```

C++/Qt SADC tool

- Inspection of single channel
- Inspection of all channels
- Configuration of internal or external triggers
- Waveform can be shifted w.r.t. trigger



C++/Qt SADC tool