A Real-Time Data Reduction and Processing Firmware for SRS

Raffaele Giordano

Physics Dept. - University of Napoli “Federico II”
and INFN Sezione di Napoli, Italy

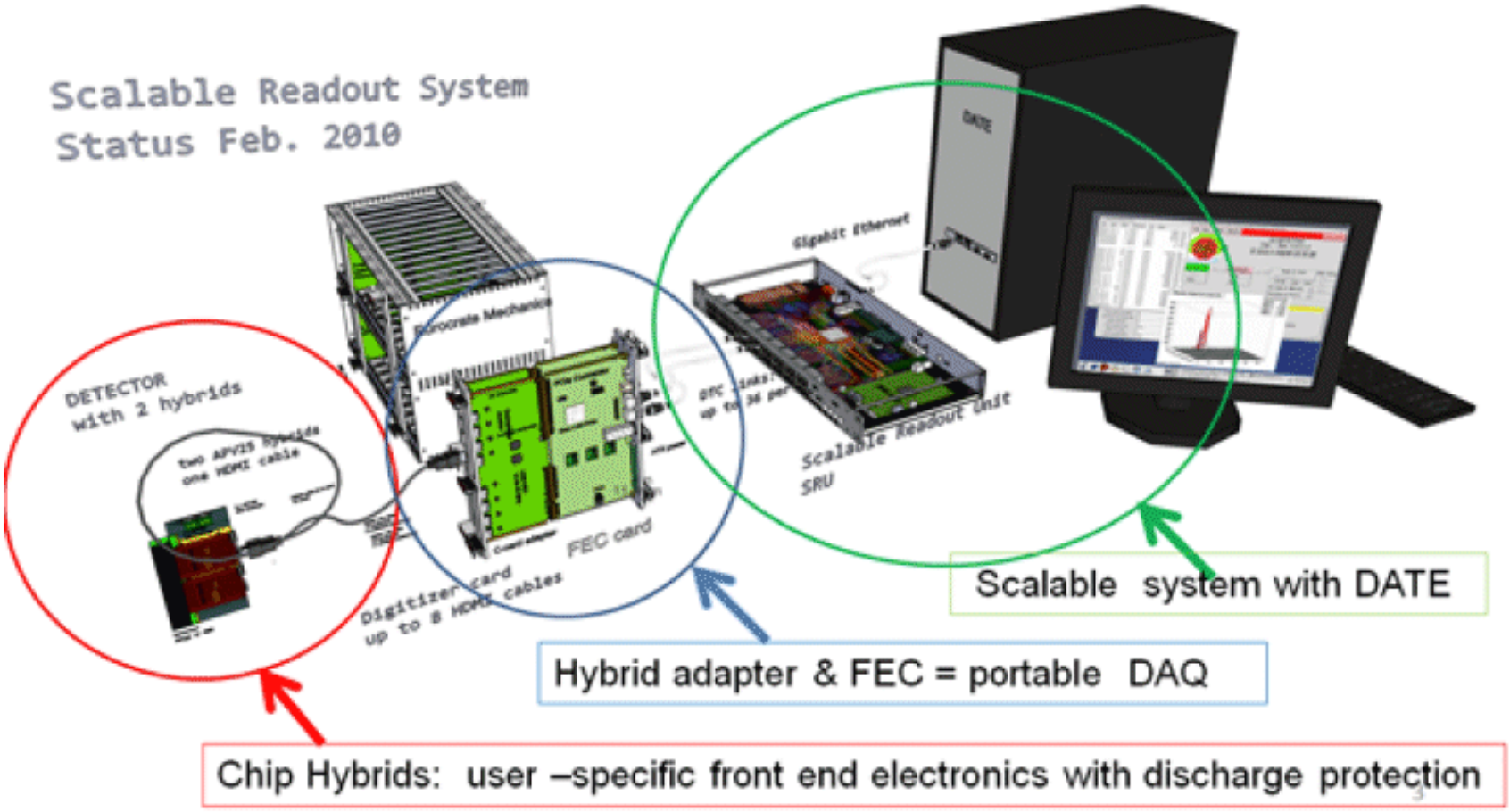
email: rgiordano@na.infn.it

Outline

- Scalable Readout System physical view
- APV data parsing
- APV-FPGA relative clock phase adjustment
- Pedestal correction (static)
- Pedestal correction (dynamic)
- Conclusions and future work

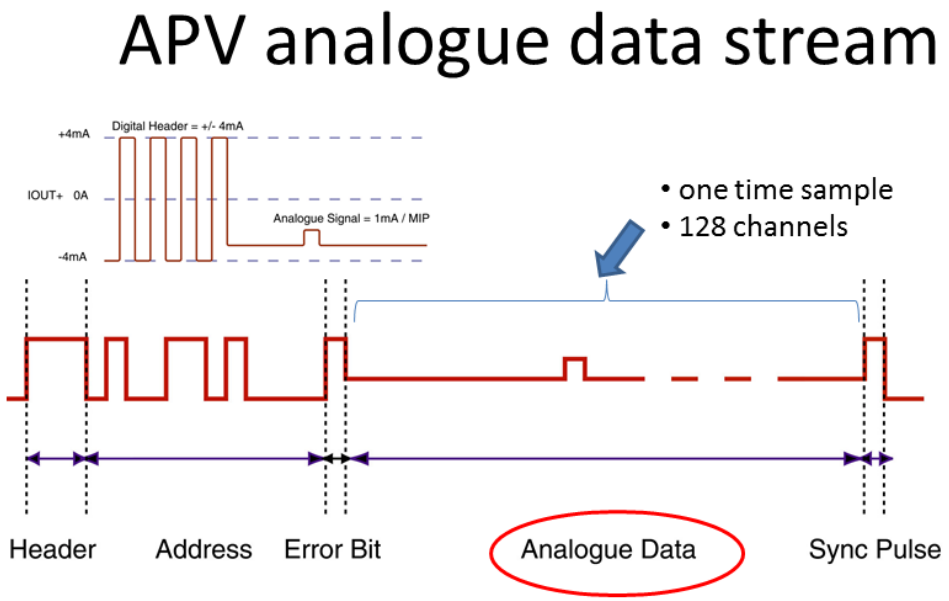
SRS Physical View

From SRS website



The APV Data Stream

From Sorin Martoiu's presentation

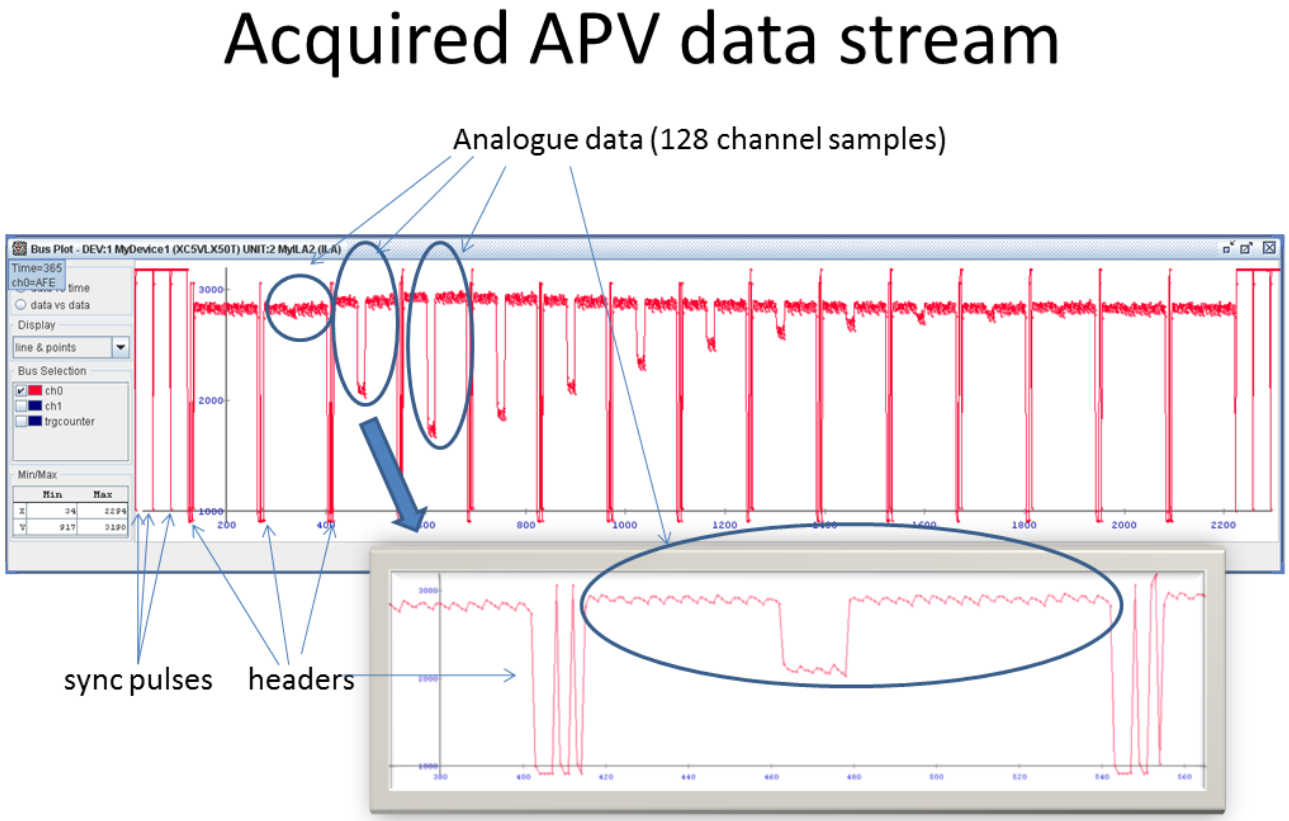


The output from the APV25-S1 chip is in differential current form in the range +/- 4mA. When there is no data to read out, the output of the chip is at the logic 0 level, with synchronisation pulses every 70 clock cycles (in 20 MHz mode) or 35 clock cycles (in 40Mhz mode). When an event is triggered, the chip waits until the start of the next (70 or 35) clock cycle period before any data is output. A data set is then made up of four parts: a digital header, a digital address, an error bit, and an analogue data

- Current driven signal (±400 mA)
- Stream with digital and analog info
- Digital: 3-bit header, 8-bit address,1-bit error flag
- Analog: 128 samples (1 time sample for 128 channels)
- When there is no data, the APV send synchronization pulses

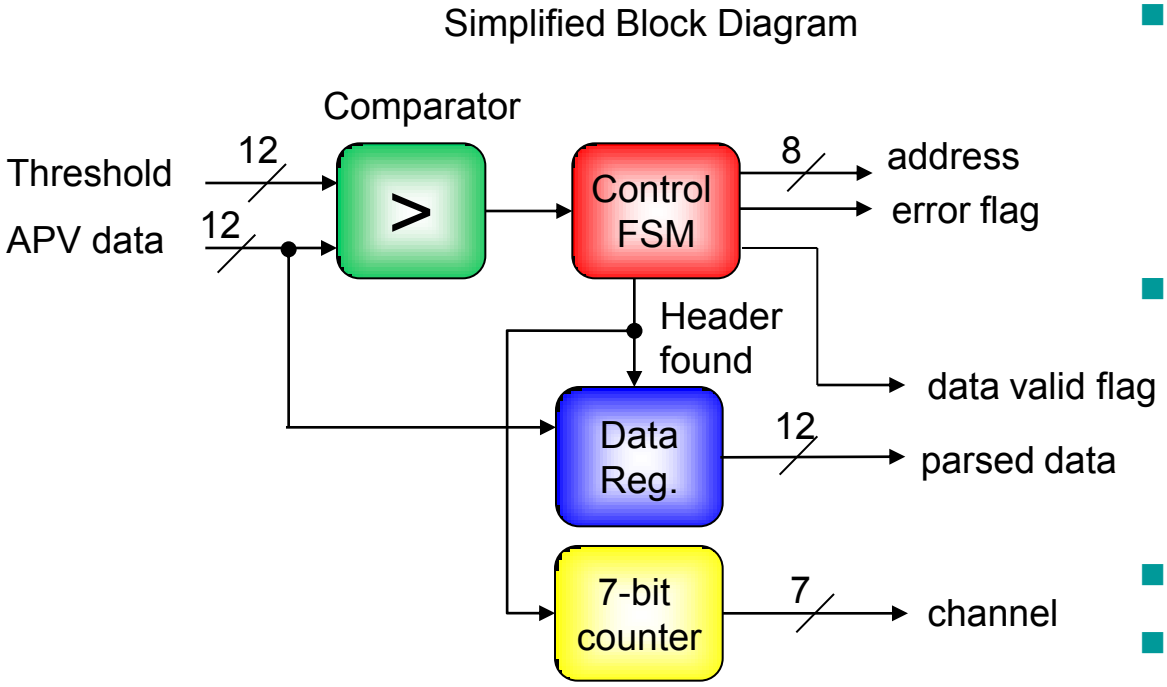
12-bit Digitized APV Data Stream

From Sorin Martoiu's presentation



- The whole stream is digitized on the hybrid-adapter board: 12 bits at 40 MHz (digital signal is inverted with respect to analog)
- In the present implementation, all the stream (even synch pulses) is transferred over Ethernet to the DAQ PC
- Need to transfer only the 128-channel data and skip synch pulses

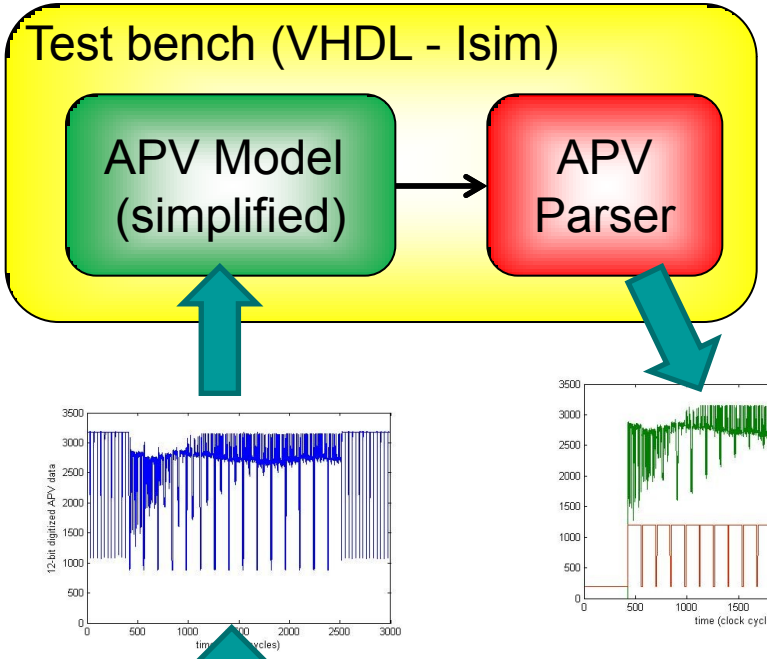
APV Data Parser

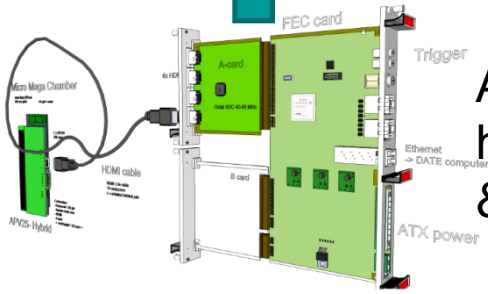


- A module for parsing the APV digital info (header, address, error bit) and to output the actual data only has been designed
- Input:
  - 12-bit digitized APV stream
  - 12-bit threshold for discriminating the digital info in the APV stream
- Output:
  - 12-bit APV data & qualifier (data valid flag)
  - channel number
  - APV Address & Error bit
- Fully pipelined
- Backward compatibility: need for a transparent mode?

- When there is not actual data from the APV (i.e. synch pulses) the data valid flag is kept low and parsed data bus is not updated

Simulation Strategy: APV Model



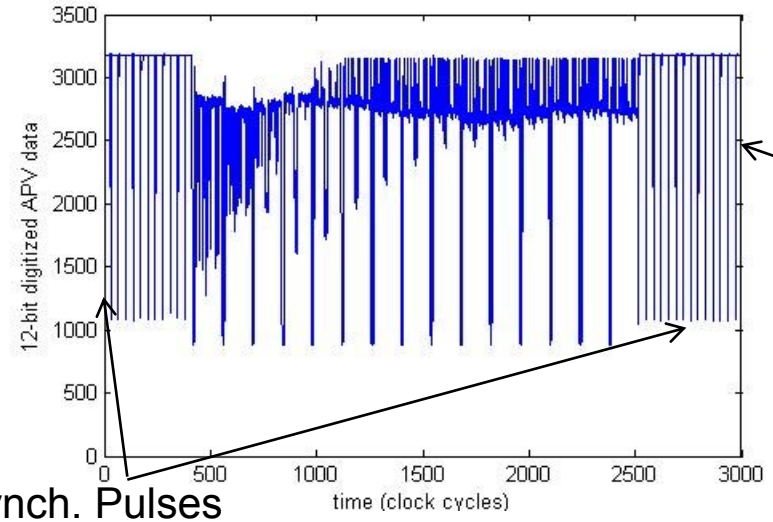


APV hybrid, hybrid-adapter & FEC card

- Library of Chipscope acquisition files used as benchmark
- VHDL model reads «real» APV streams acquired via Chipscope
- This approach makes it easier to debug the code
- After successful verification of the VHDL code => we tested the parser in the field

Testing the APV Parser in the Field

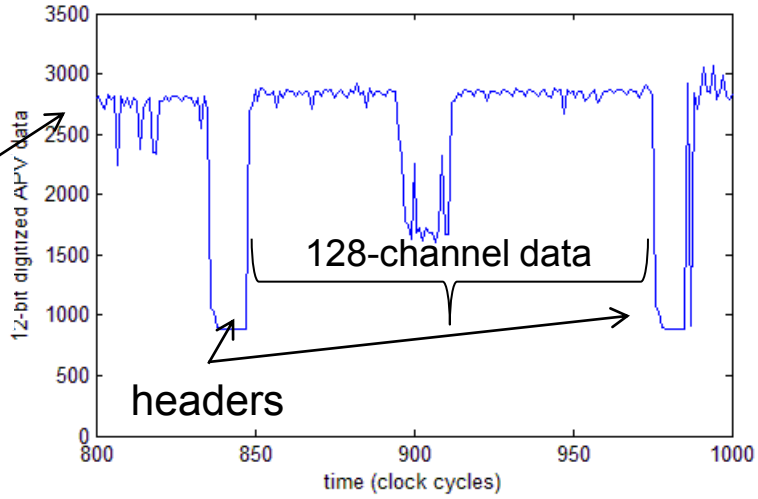
Burst of time samples from APV



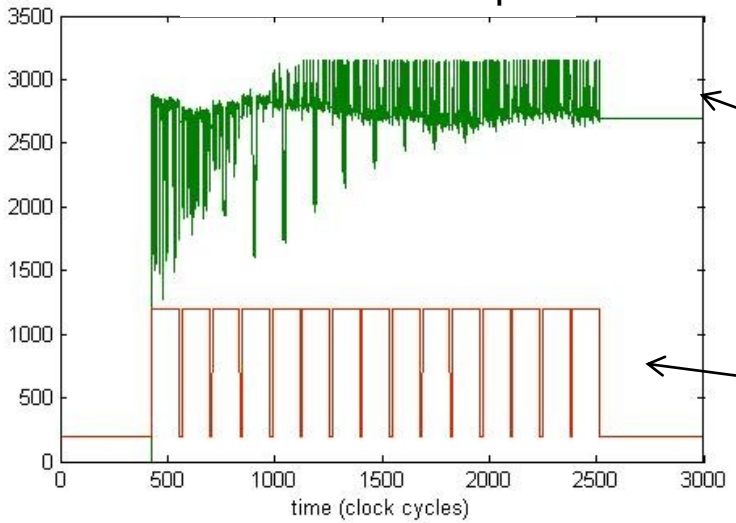
Synch. Pulses

12-bit digitized APV stream

Close-up: 1 time sample from APV (128-ch)



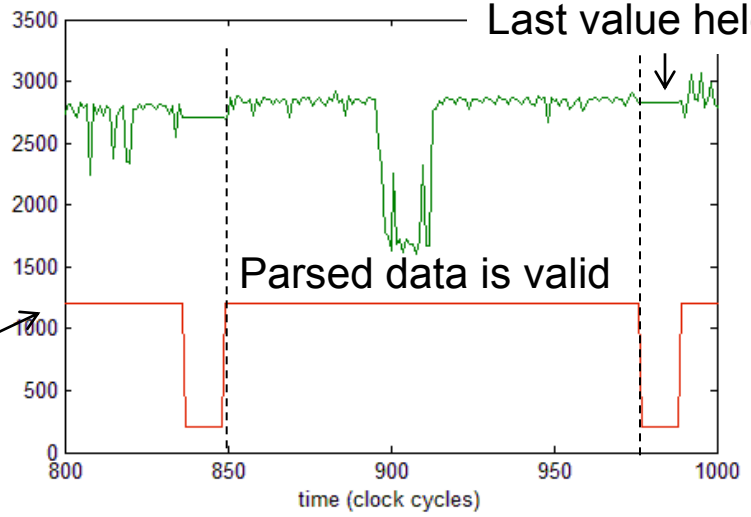
APV Parser output



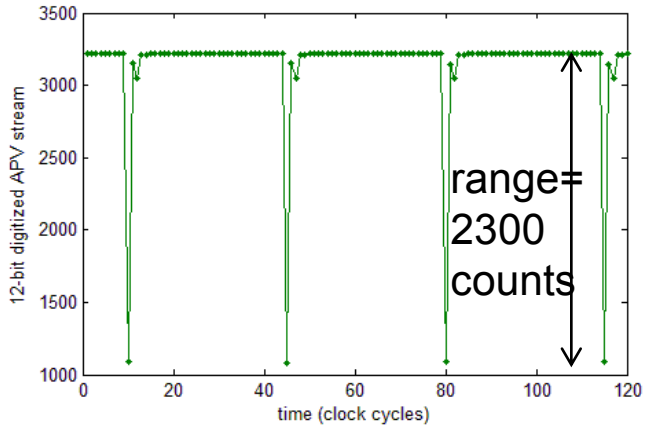
Parsed data

Data valid Flag

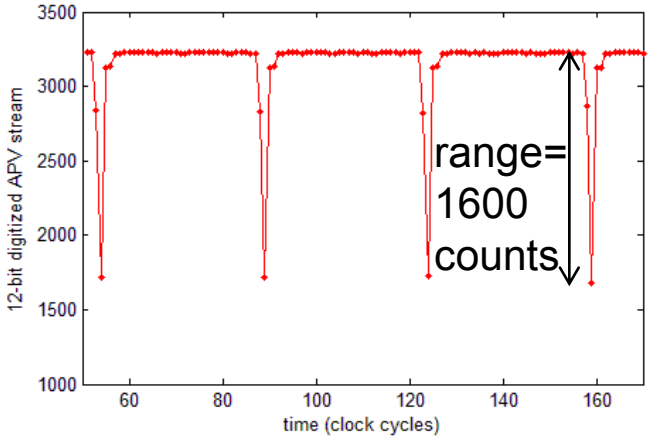
Last value held



APV/TPLL Clock Phase Adjustment



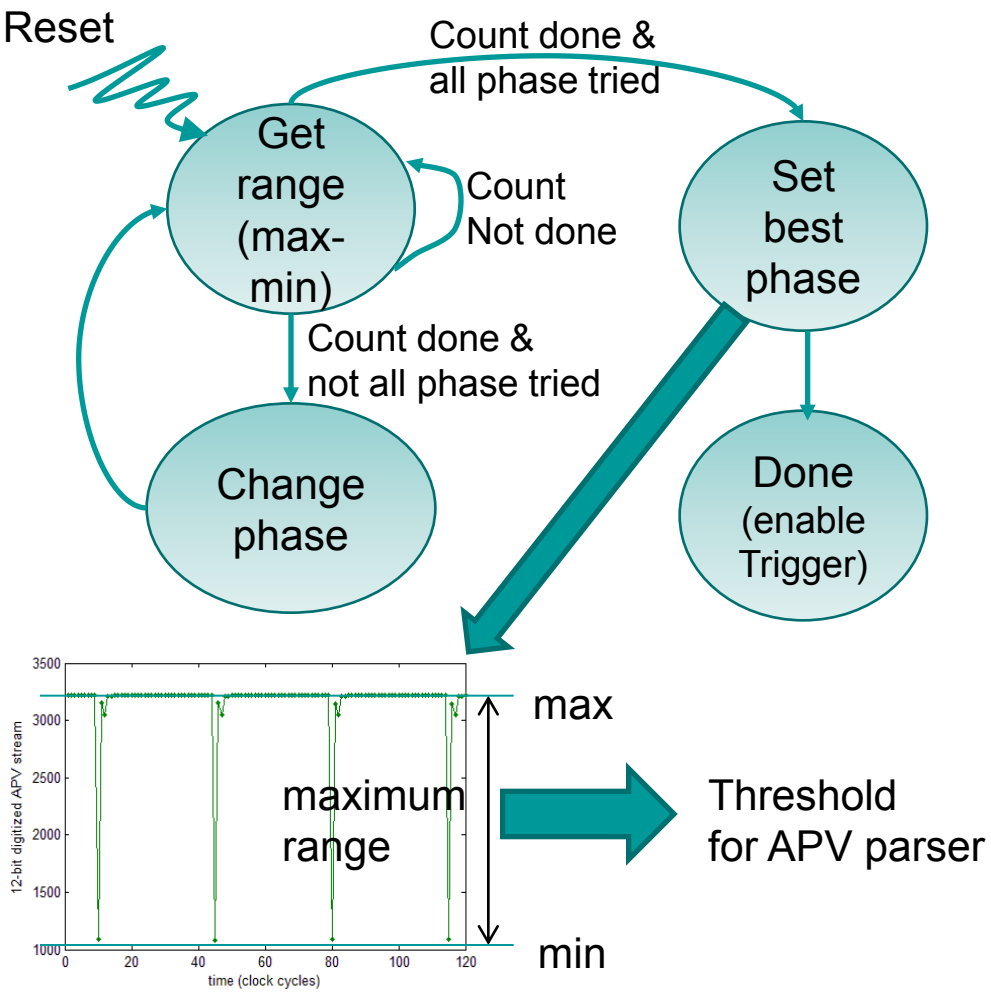
Correct clock edge sampling



Wrong clock edge sampling

- If APV sampling phase is not suitable with respect to ADC clock phase: signal sampled while it is not stable => poor digitization
- Need for adjusting the APV clock phase (i.e. the phase of the TPLL clock on the hybrid)
- The best phase is the one having the highest dynamics

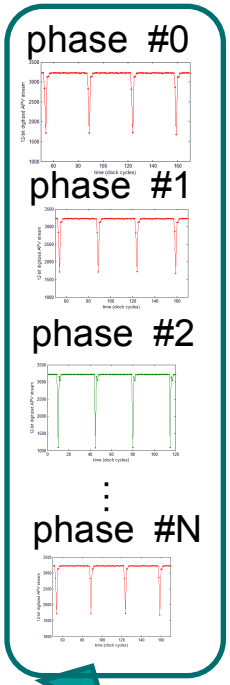
TPLL Phase Auto-Adjuster

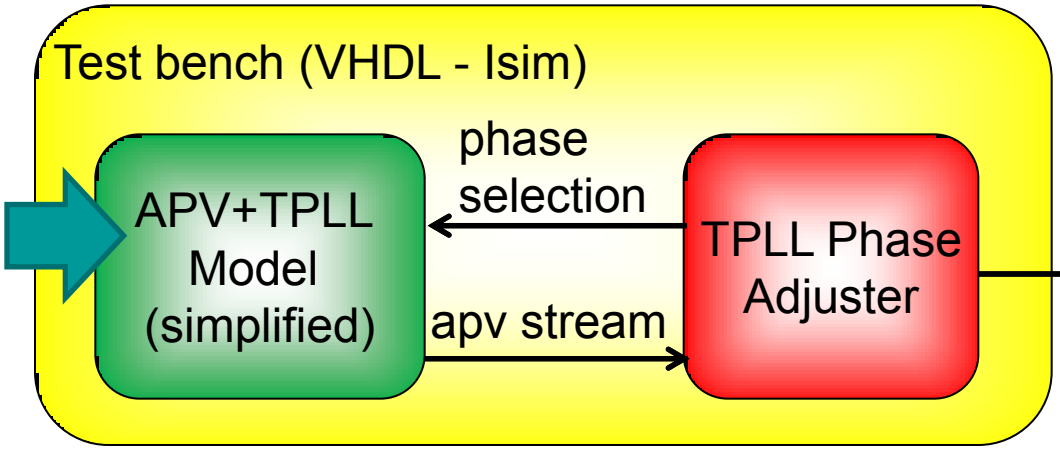


- Based on exhaustive search of best suitable phase (24 possible phases)
- Trigger inhibited during the search, the phase adjuster only receives synch pulses
- For each phase the min-to-max range (in ADC counts) is measured during a pre-defined time window (e.g. 1024 clock cycles)
- The phase having the maximum range is selected after the search
- The adjuster also uses the range info to set a threshold value for the APV parser

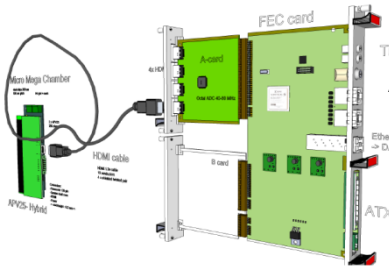
Simulation Strategy: APV+TPLL Model

APV streams acquired with different TPLL phases





Chipscope



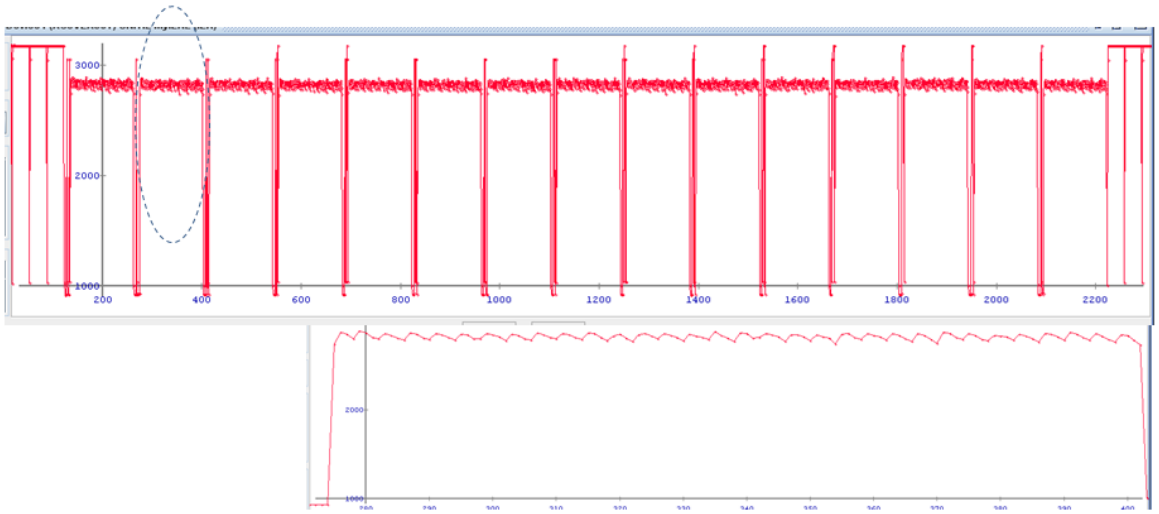
APV hybrid, C-card & FEC card

- Designed a model of the APV+TPLL for simulating the phase selection
- The model uses a set of APV streams acquired with different TPLL phases to emulate the effect of a phase change on the APV stream
- The phase adjuster has been successfully tested in simulation

Pedestal (Static)

From Sorin Martoiu's presentation

Pedestal correction



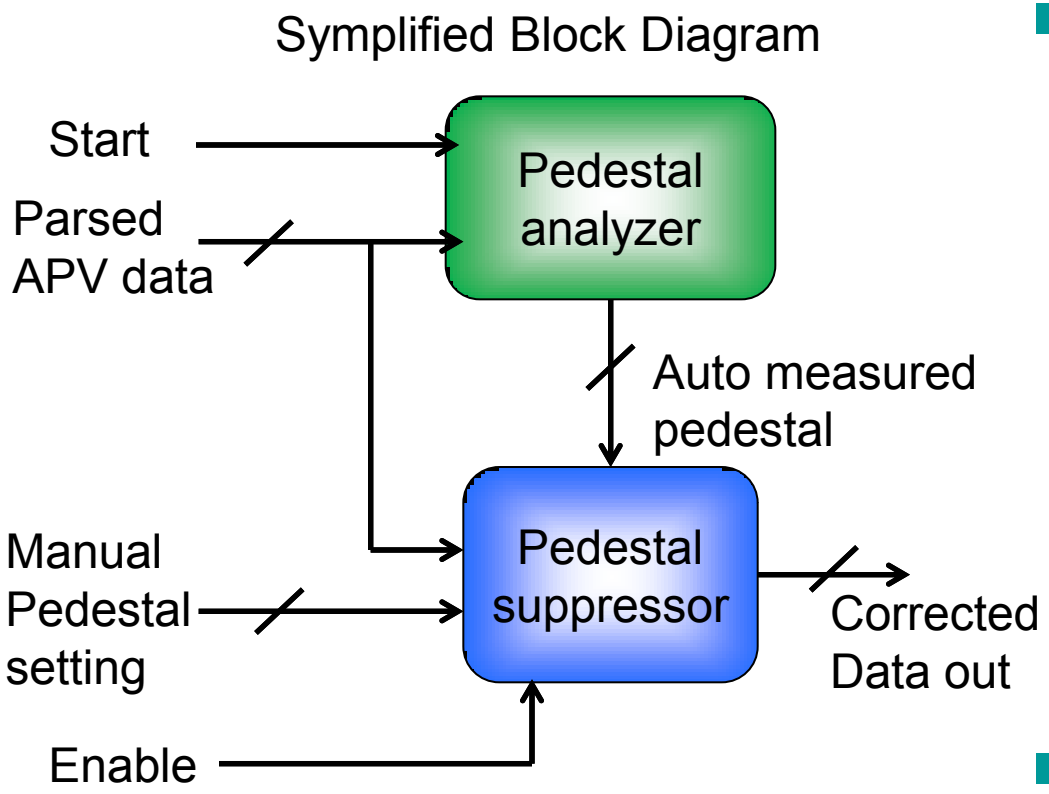
- high variations observed
  - temperature dependence?
  - power supply dependece?
- further investigation
- periodic on-line pedestal measure (?)

- Pedestal variations on the time scale of minutes (or longer)
- Due to
  - Temperature?
  - Power supply?
- Need periodic on-line pedestal measurement and correction

Chip or Channel ?

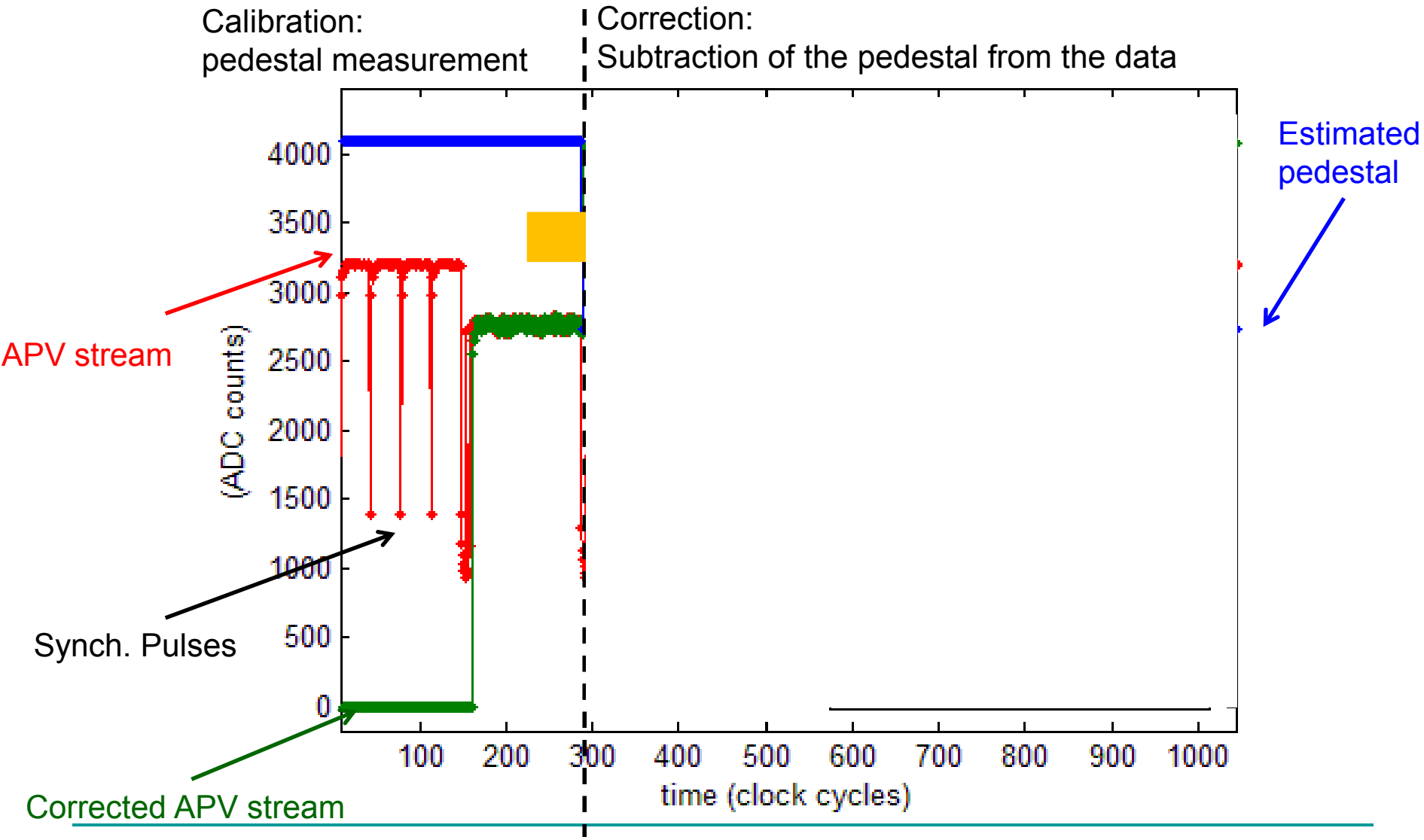
- Measure and subtract the same pedestal for all the channels? Or each channel has a different pedestal?
- We could perform an analysis on representative benchmark data (again to be acquired via chipscope) to evaluate the difference of the two methods
- Let us suppose this difference is negligible: we calculate one pedestal for the whole chip (i.e. all the channels)

Pedestal Measurer & Remover



- Two pedestal setting mode
  - Automatic: acquire the pedestal from the stream (parsed APV data)
  - Manual setting of the pedestal via a dedicated port
- Enable/Disable pedestal removal
- Fully pipelined

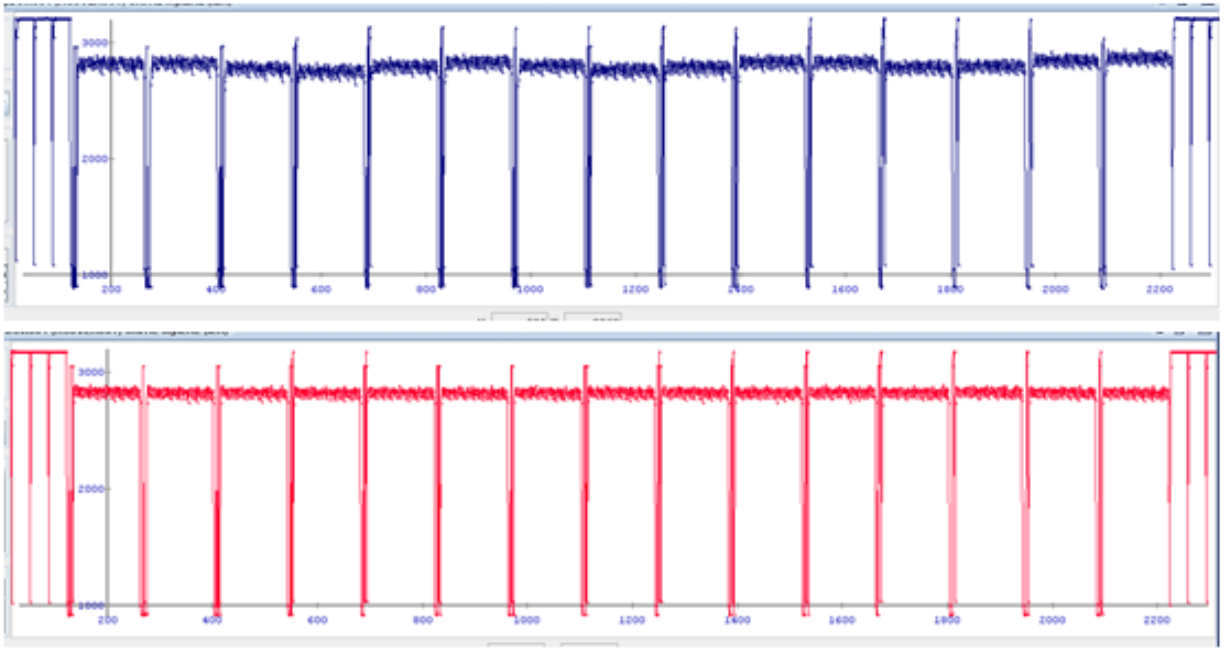
Simulation Results



Pedestal (Dynamic)

From Sorin Martoiu's presentation

Common-mode correction



- spurious common-mode noise

- Currently performed as a 2-step calculation off-line (ask Sorin for details on the algorithm)

Dynamic Pedestal On-line Correction

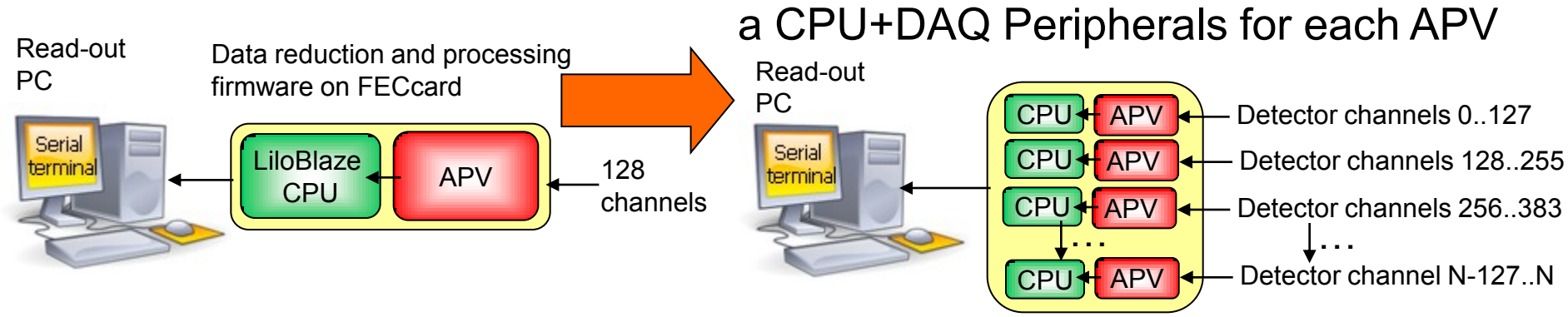
- Correction on each sample depends on subsequent (future) samples
- Can use the present approach also on-line, but need memory for this
  - For instance with one V5 Block-ram (36kbits) configured as a FIFO
    - we can cope with a burst of 20 time samples from APV (20x128 12-bit digitized samples)
    - minimum acceptable spacing between APV data bursts dependent on processing time for correcting the pedestal => need to be estimated

Embedded Processor ?

- A tiny FPGA-embedded processor (e.g. Xilinx PicoBlaze) could be used to run the dynamic pedestal correction algorithm on-line
- With respect to a standard FSM approach
  - Pros
    - flexibility: easy upgrade and/or modification of the program
    - debug info and statistics might be available on a serial port
  - Cons
    - higher resource occupation, but still moderate (just 26 slices in a V6)
    - Slower due to IO overhead, but could solve this by means of dedicated peripherals and direct memory access

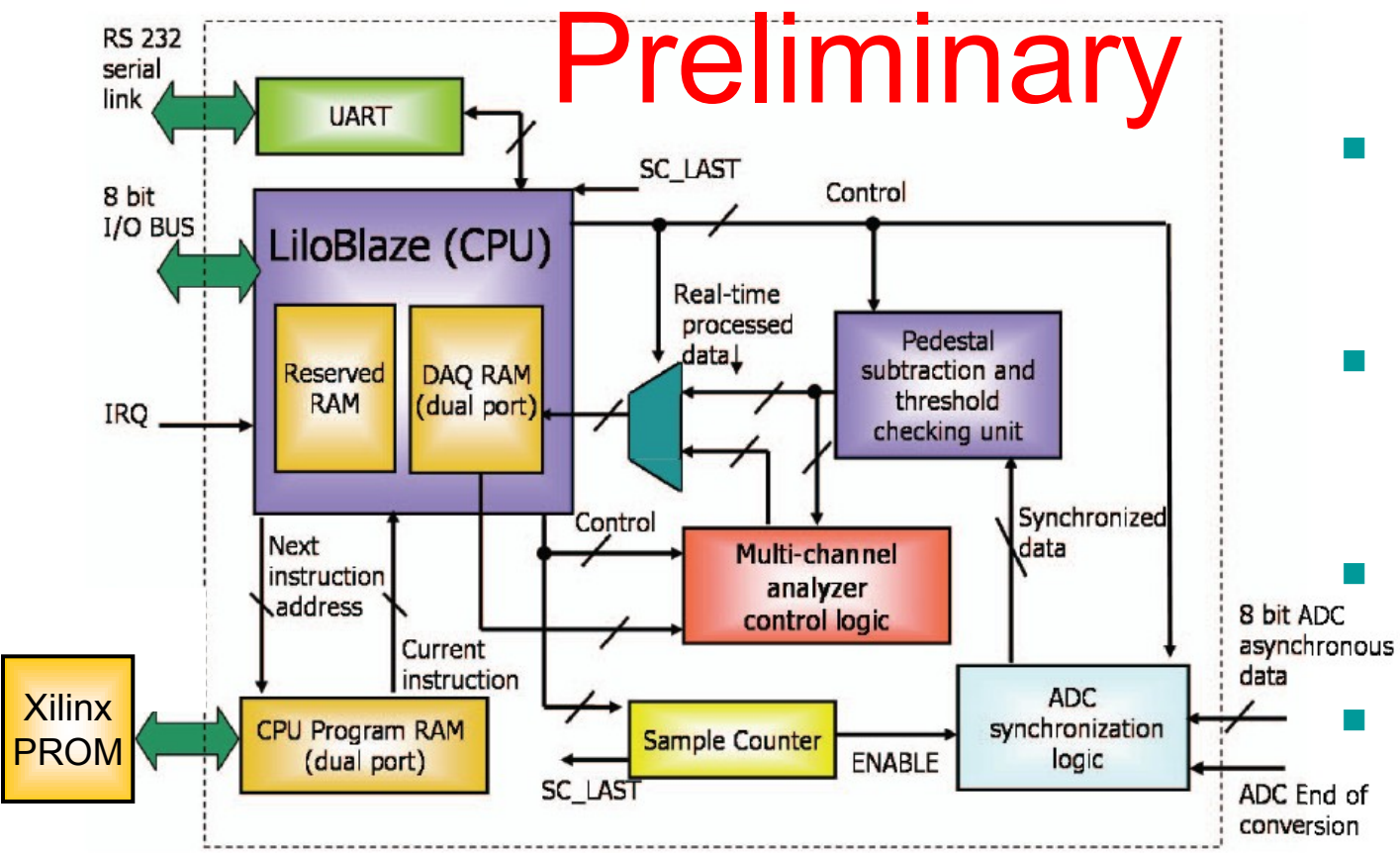
LiloBlaze: a Custom Microprocessor

- 8-bit RISC architecture
- optimized to acquire 8,12 or 16-bit data, most important features are:
  - Timing predictability: every instruction takes always 2 clock cycles
  - High working frequency (100 MHz in Xilinx V2-1000 FPGA)
  - Very low logic footprint for scalability (2k gates) =>



- Architecture compatible with PicoBlaze in order to inherit its software tools and documentation

LiloBlaze Close-up



- Custom modified PicoBlaze microprocessor
- Peripherals for real-time APV data processing
- Integrated 36kbit Block Ram for APV data
- Serial IO interface
- Parallel (16-bit) IO

Conclusions

- SRS data reduction and feature extraction modules presently under development in Napoli
- Features implemented and tested:
  - APV data parsing (successfully tested in the field)
- Features implemented and ready to be tested in the field:
  - Hybrid vs FPGA clock phase auto-adjustment (successfully verified in simulation)
  - Static pedestal estimate and correction (successfully verified in simulation)
- Under development:
  - Dynamic pedestal estimate and correction
  - Custom microprocessor for enhanced feature extraction

Aknowlegdement

- We wish to thank Hans and Sorin for all their help and support in setting up the requirements and in testing the data reduction firmware in the field

- Many thanks also to Givi and George (and again to Sorin) for providing the chipscope data as well as to Prof. de Asmundis and Marcin for helping out with the APV data format

Questions ?

Ask Raffaele!

Email: rgiordano@na.infn.it

Office: +39 081 67 92 57