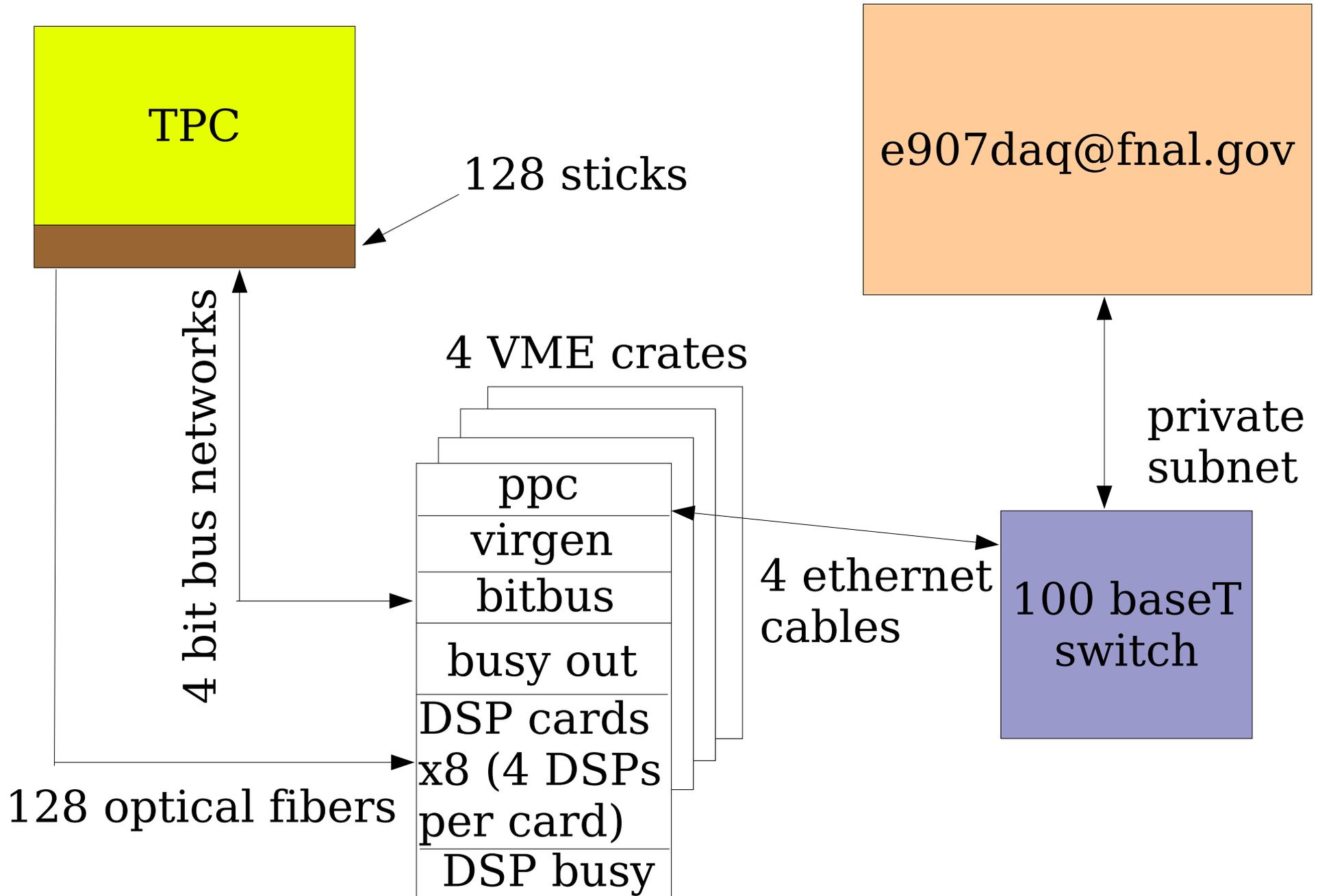


EOS TPC Electronics and Online Data Processing for Mipp

Outline

- Overview of the Electronics
- Timing
- Data processing on the stick
- Data processing on the DSP
- Data processing on the PPC
- Auto-inventory
- Pedestals
- Thresholds/ compression algorithm

Overview



Timing

Two basic time scales:

drift time of e in gas, 100ns/ sample in parallel on pads (16 μ s)

conversion time in ADC 520ns/sample in series (10ms)

Absolute Y position:

The y position of a track is based on a known drift velocity and the time the trigger occurred. The bucket clock is not in phase with a random trigger, and therefore the phase of the clock is measured when a trigger is fired. Beamppc process called tpctrigger measures this.

Stick

I assume that Peter has covered most of the details of the hardware of the stick

Key items (what they effect) in no particular order:

- discrete shaper amp per channel(DSP processing)
- multiplexed AMU and ADC (pedestal)
- TAXI is one way data transmission from the stick(data flow)
- BitBus is bidirectional (stick programming)
- FPGAs and BitBus CPU are used to set/control stick features
- AMU clock set in hardware, no software control(#bucket)
- TPC pad plane sets stick number (auto-inventory)
 - Full 7 bit address is sent over TAXI
 - 5 bit subset is used to set BitBus network address

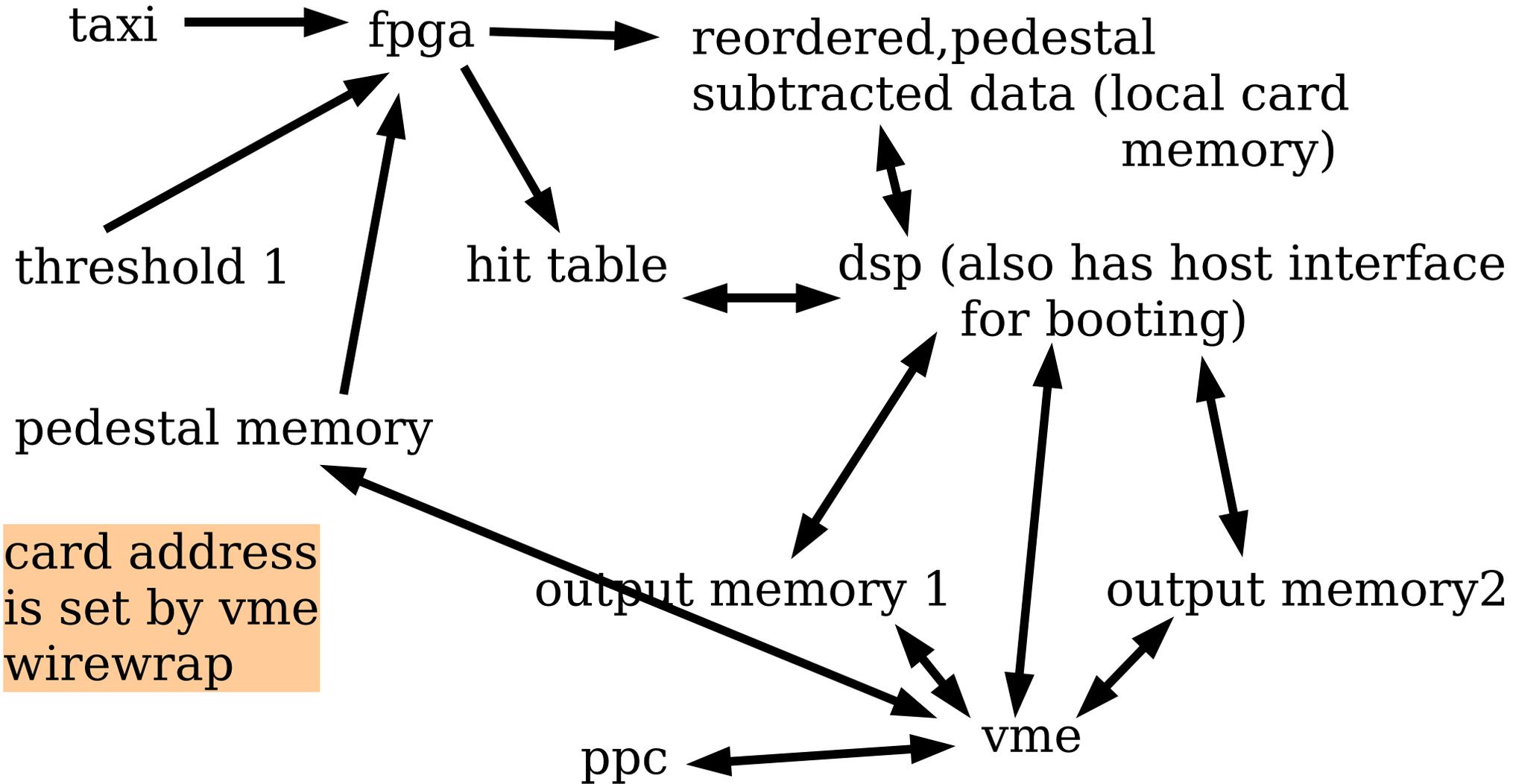
Controllable Stick Features through Bitbus:

- 4 modes: normal, pulser, off, memory dump
- Number of time buckets: 128,160,192,256
- software trigger
- other: pulser mask, memory for taxi dump

Receiver Card

fpga process as
adc converts on stick

No bitbus, that is a separate controller
Taxi is the input, vme memory is output



Motorola DSP56001

Processes the FPGA data with a 24bit CPU
few hundred lines of assembly, in cvs if interested

DSP starts processing after FPGA signals finished
DSP fills output buffer and signals ppc data is ready

3 selectable processing functions:

- full copy
- compressed data
- test readout

full copy:

- writes header
- writes hit table
- writes all adc values

compressed data:

- writes header
- writes only adc over threhsold2

PPC

linux operating system

C++ code

PPC controls the system timing after the dsp

PPC moves data from receiver card to buffer memory

data set to e907daq though ethernet on ppc board

Data is **not** processed further, only formed into packets and set to event builder

The event builder just puts the packets into an event and puts this on disk. Access to the data in the file is generally done though MippIO.

Auto-inventory

This is only to point out:

There is a coordinate transform
at MippIo

How we know the position of a
given pad in the data

<file:///home/mheffner/Desktop/tpc.gif>

DSPs identified by
custom VME (wirewrap)

Cards identified by pad
plane etch

PPC searches Bitbus and VME
for all sticks and DSPs

PPC software triggers all sticks
and compares bitbus address
to taxi information

Connection map is generated

Pedestals

voxel by voxel, because of the multiplexed data path

Process of Generation:

readout uncompressed data (setup the DAQ for pulser trigger and uncompressed)

Read raw data file and generate summary (MakeSummary.cc)

Find bad pads, $rms > 3$ (FindBadPads.cc)

Add in bad pads not found automatically, usually notices in the event display

Use average as pedestal and generate file (PedestalCreate.cc)

Note:

Bad Pads masked with pedestal set to 0xffff

Problems with signed subtraction --> 0x80 offset

Written to data stream

Thresholds/Algorithm

FPGA threshold

one per stick

current plan, all set to same value

generates pad hit table

algo-- if any bucket is $>$ FPGA thresh

set bit in hit table

DSP threshold

expand hit table to nearest neighbors

search all hit pads at DSP threshold

write only buckets $>$ DSP threshold to

output buffer