New Ideas for including the Pixels in the L1 trigger for HL-LHC at CMS: joint effort FNAL and CNRS-INFIERI

David Christian

Fermilab

October 18, 2016

Context – High Rate Pixel Detectors

- Hybrid detectors
  - Sensor and Read Out Chip (ROC) are separately optimized using very different silicon process technologies.
  - Sensor and ROC are joined using “bump bonds” and “flip chip assembly.” Typically 4 – 8 ROCs are bonded to a single sensor.
  - ROC contains
    - Pixel Unit Cells; 1 per sensor pixel, with bump bond to a sensor pixel. Organized in columns (or double columns).
    - Peripheral Logic, including “end of column” logic, chip-level logic, and wire bond pads for power, I/O to DAQ, etc.

Hit Storage

- Hits are stored in the ROC. A small fraction are read out to DAQ in response to a trigger; most are discarded.
- Two approaches to hit storage:
  1. Send all hits to periphery as they occur (“column drain”).
  2. Store hits in Pixel array until trigger; only transfer hits to periphery if they will be read out.
  - (1) minimizes logic in PUC, allows smaller pixels and lower power consumption (shared ADCs in periphery).
  - (2) maximizes rate capability; rate limitation given by trigger rate & event size rather than by event rate & event size.

Trigger

- Events are “time stamped” using “beam crossing number” (“BCO” or “BX”).
- Trigger occurs either after a fixed latency (number of BX clock cycles) or as a request to read out a specific BX.

Use of Pixels in the Trigger?

- Rate near the beam is much too high to read out all data from every BX.
- “Region of Interest” (ROI) read out is possible.
- If trigger is accomplished by matching BX number and ROI = number of ROCs, then ROI read out and normal read out could be the same thing from the point of view of a ROC (but DAQ would have to save event fragment for readout after normal trigger).
- We have begun to develop an architecture to support ROI smaller than a ROC.
  - Trigger = trigger type + BX.
  - ROC saves data for later read out unless trigger type = full.

Content Addressable Memory (CAM)

- Trigger latency of 10 μs = 400 BX cycles (20 μs = 800)
- CAM is key to quickly find match to trigger BX.
  - Normal memory: fetch(*number*) = contents of memory address *number*.
  - CAM: fetch(*number*) = address of memory element with content = *number*.

Indirect Addressing

- Even closest to the beams at HL-LHC the hit rate per pixel is very low.
  - The number of hits needed to be stored in a group of pixels is << the latency in BX.
  - Area in the pixel array can be saved by storing the BX at the end of column and storing a pointer to the EOC BX in the hit pixel (or group of pixels).
    - If the list in the EOC of BX numbers corresponding to hits is a CAM, then the pointer is a CAM address.

Proposed Architecture

- Hits are stored locally in the pixel array, each with the address of an EOC CAM holding a BX.
- Hits are read out from the pixel array to the EOC in response to a match between a CAM-stored BX and the requested BX.
- For ROI triggers, only data with (column,row) in requested range is read out; full chip data is retained in EOC for possible later read out.
- Data held for possible later read out (in the pixel array or in the EOC) is dropped after it reaches a specified age (# of BX cycles).

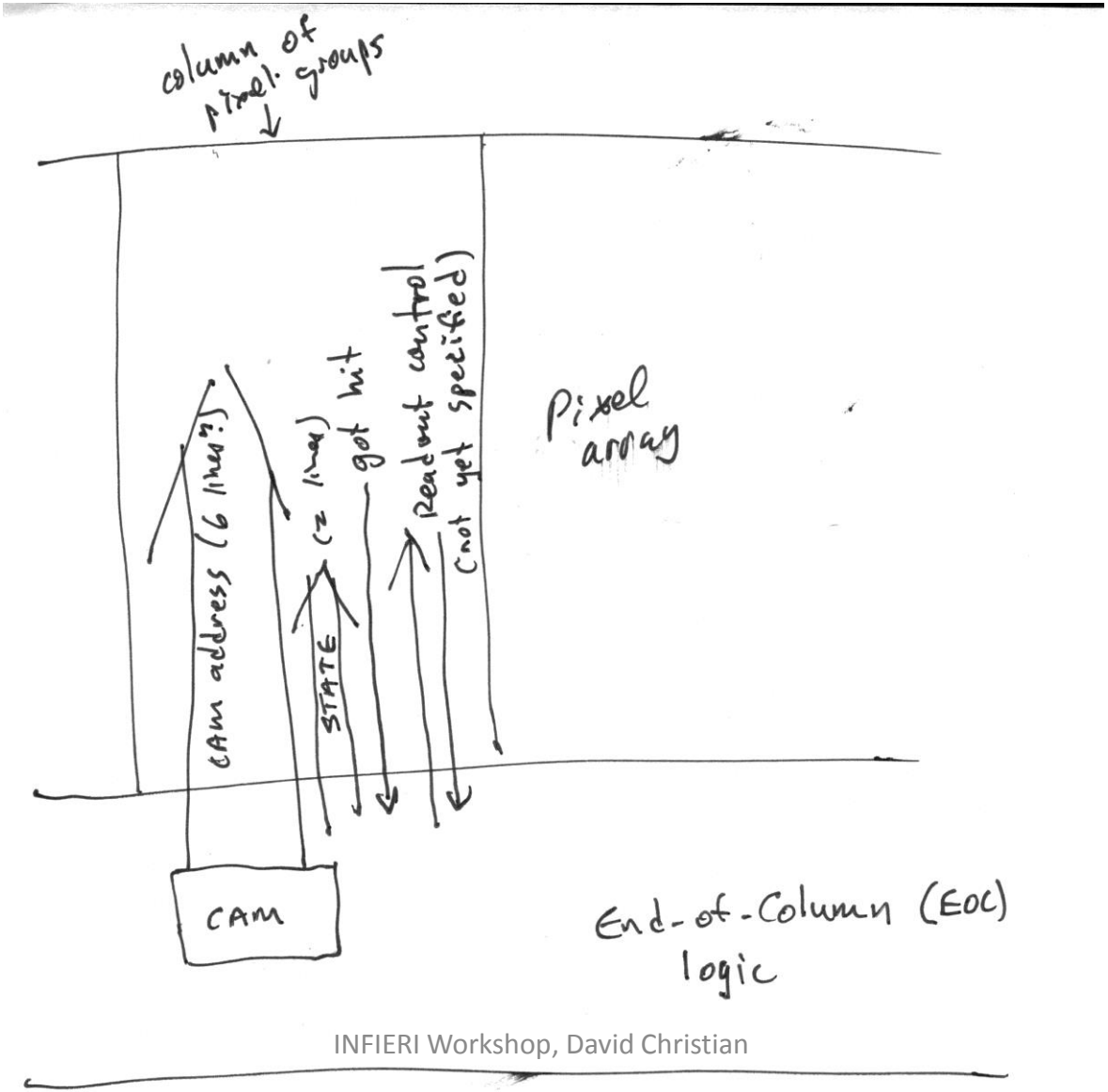
Nothing New under the Sun (Hardly)

- I recently discovered that essentially all of this idea is contained in a 1992 paper by Michael Wright, Jacques Millaud, and David Nygren:

"A Pixel Unit-Cell Targeting 16ns Resolution and Radiation Hardness in a Column Read-Out Particle Vertex Detector"

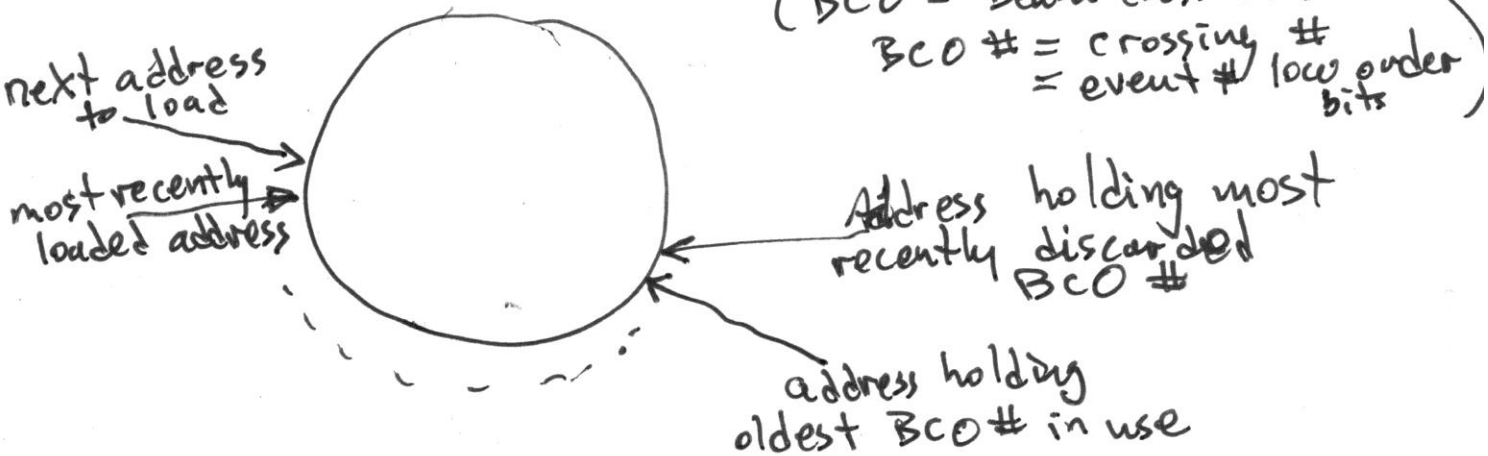
- Presented at the Third International Conference on Advanced Technology and Particle Physics, Como, Italy, June 22-26, 1992.

Backup Slides



End-of-Column (EOC)
logic

organize CAM addresses like a ring buffer
(BCO = Beam Cross Over
BCO # = crossing # = event # low order bits)



EOC CAM – Pixel Group Communication

State	EOC CAM Management	Pixel Group Action
1=Latch BCO#	Load current BCO# into next CAM Address, broadcast next CAM address to pixel array, check CAM for match with trigger request.	IF hit, latch CAM address, associate with hit data, and assert GOT HIT… Else do nothing.
2=Read Out	If trigger, broadcast trigger CAM address (found in state above). Check CAM for match with timed out BCO #. IF GOT HIT, increment next address to load (check for CAM full, ie next address to load = most recently discarded address); raise flag if CAM full.	IF CAM address match, move hit & CAM address to read out logic… Else do nothing.
3=Reset	If match with timed out BCO (found in state above), broadcast CAM address holding timed out BCO #(check that matching address = oldest; error if not) and increment oldest address in use. Lower CAM full flag if raised.	If CAM address match, reset storage (forget hit) associated with oldest CAM address. Release GOT HIT if asserted.