universitätbonn

DHP emulator

5th International Workshop on DEPFET Detectors and Applications

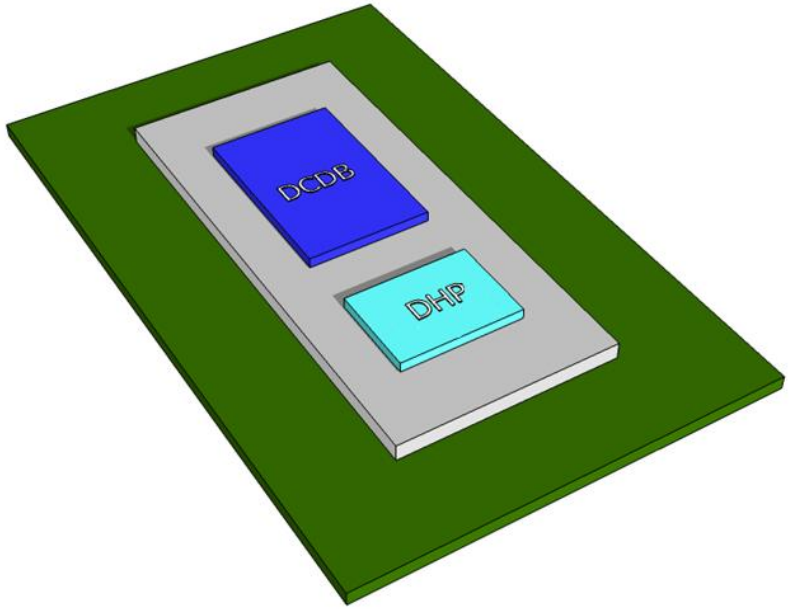
from 29 September 2010 to 01 October 2010

Mikhail Lemarenko

What is it?

DHP 0.1 chip

- First prototype of the Data Handling Processor chip



DHP emulator

- Processing unit implemented together with DCD-B readout firmware
- Exists as HDL code realized in FPGA (Virtex-4)

What is it?

universitätbonn

It is <u>NOT</u>

- FPGA replication of the DHP 0.1
- Alternative to the software simulation

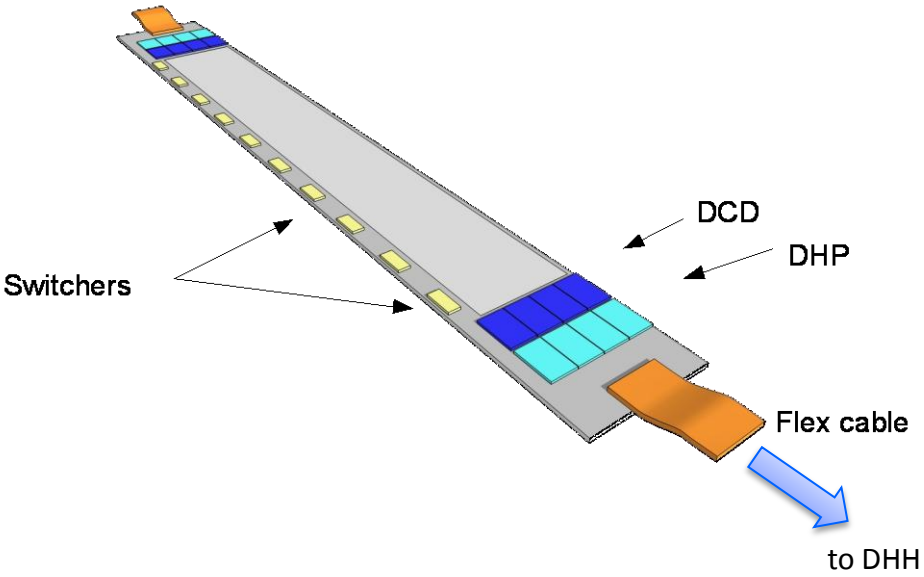
…but it is about :

- Having a fast DCD-B readout system
- Checking the optimal DHP data processing configuration(what we think it is) on the real data
- Having a similar readout conditions as in the final experiment
- What is feasible in FPGA is also likely to be easily done in the real chip

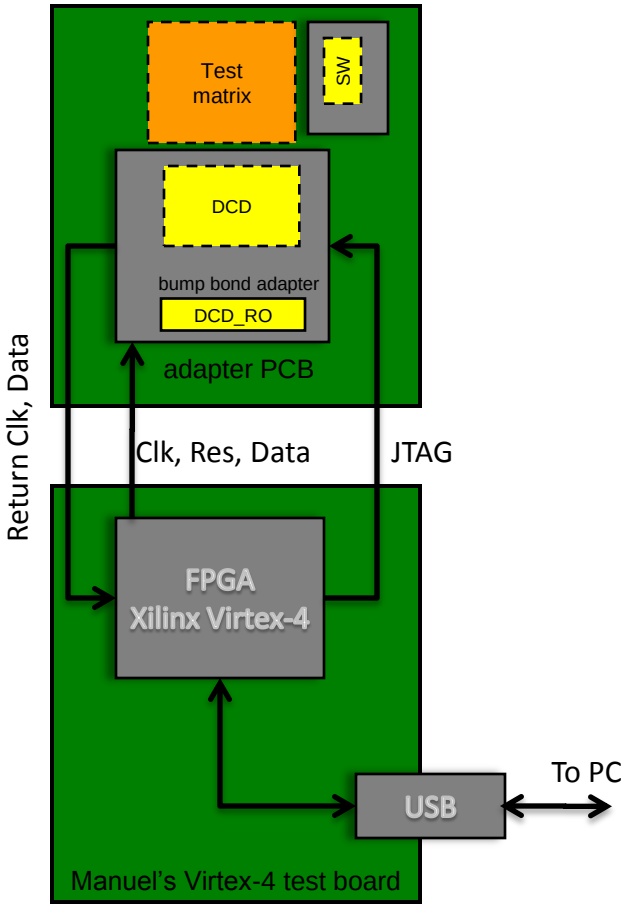
Where

- DEPFET module

(Still a long way to go)

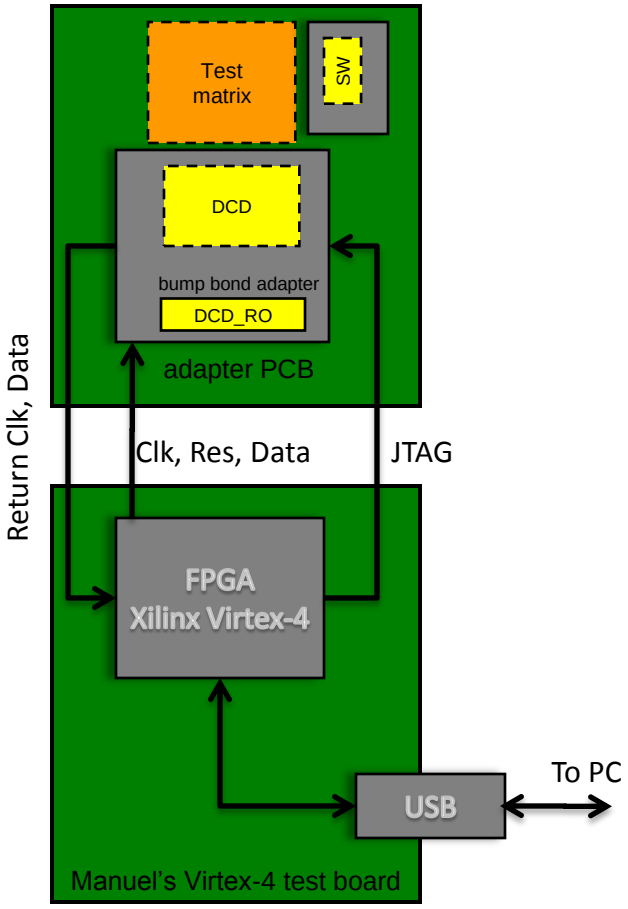


- DCD-B Test system

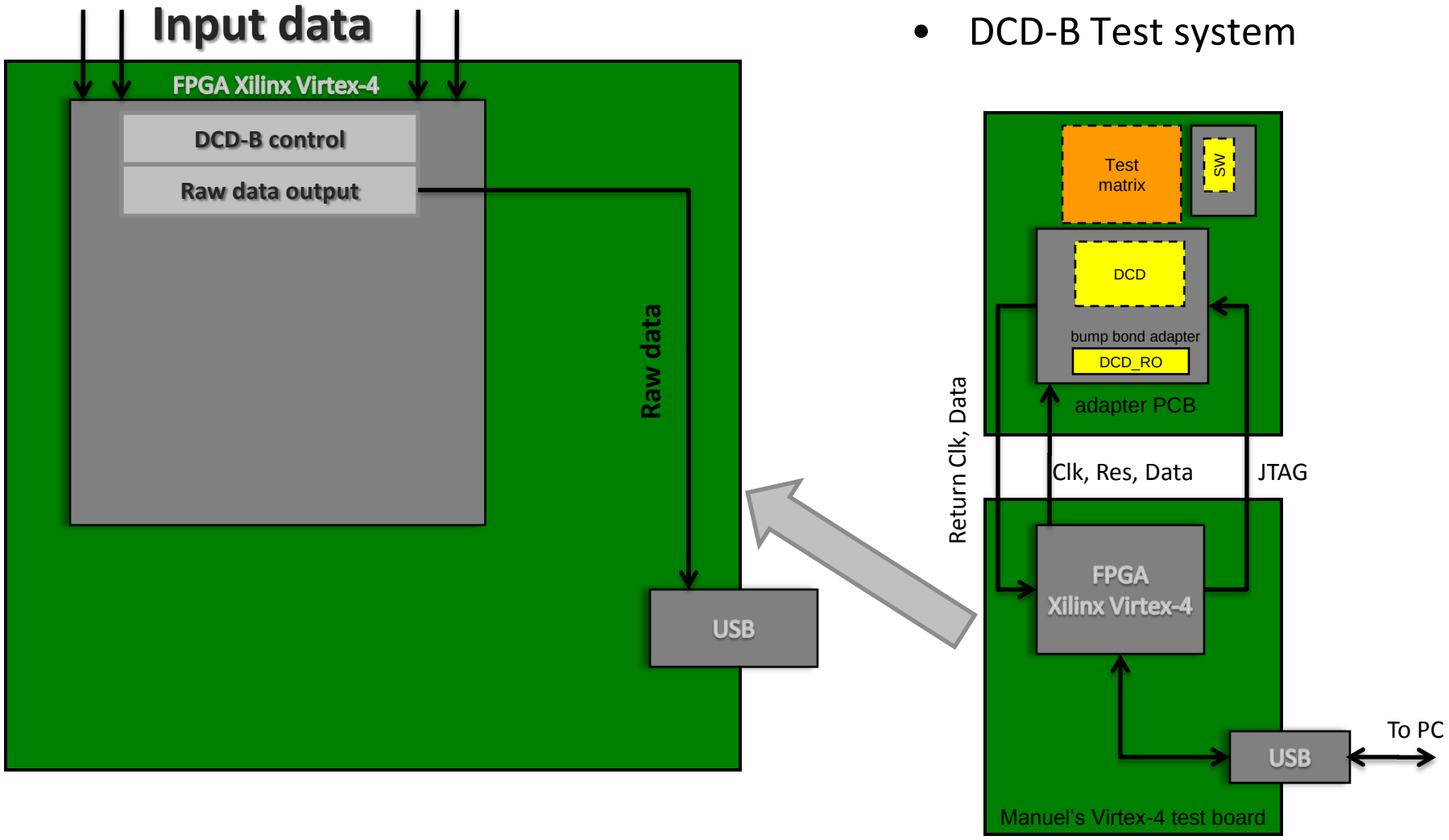


DEPFET test system

- DCD-B Test system

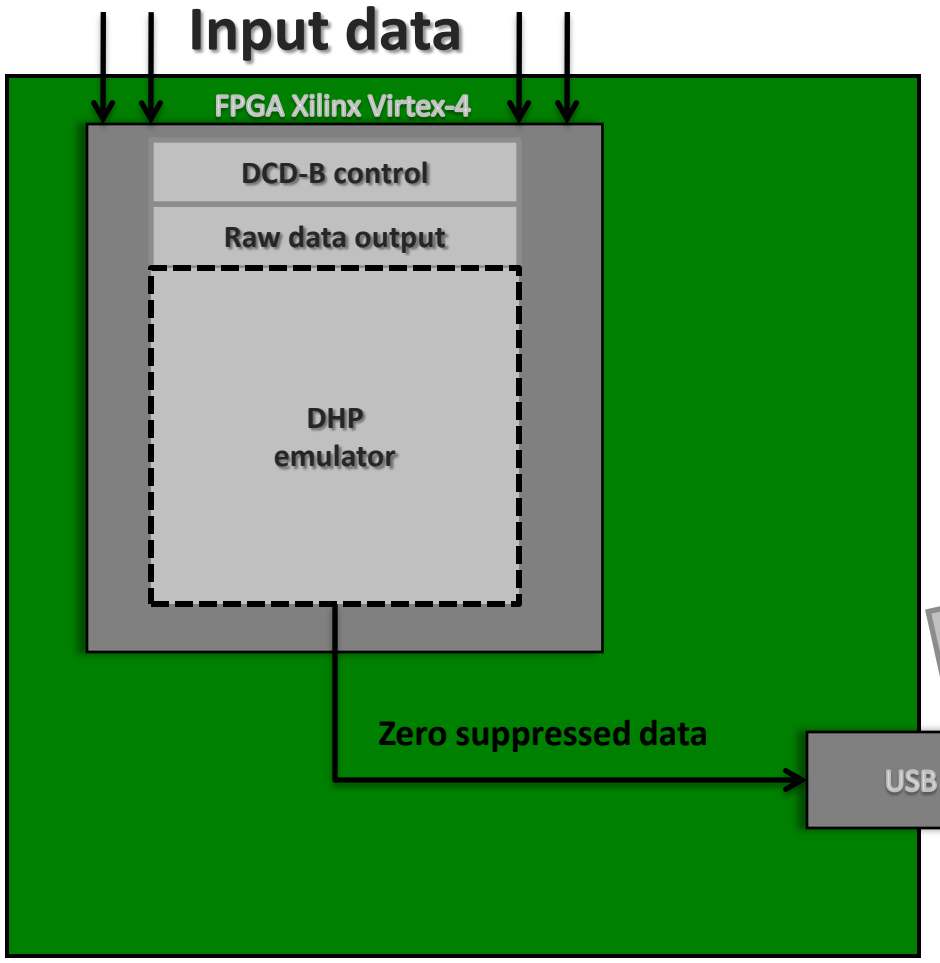


DEPFET test system

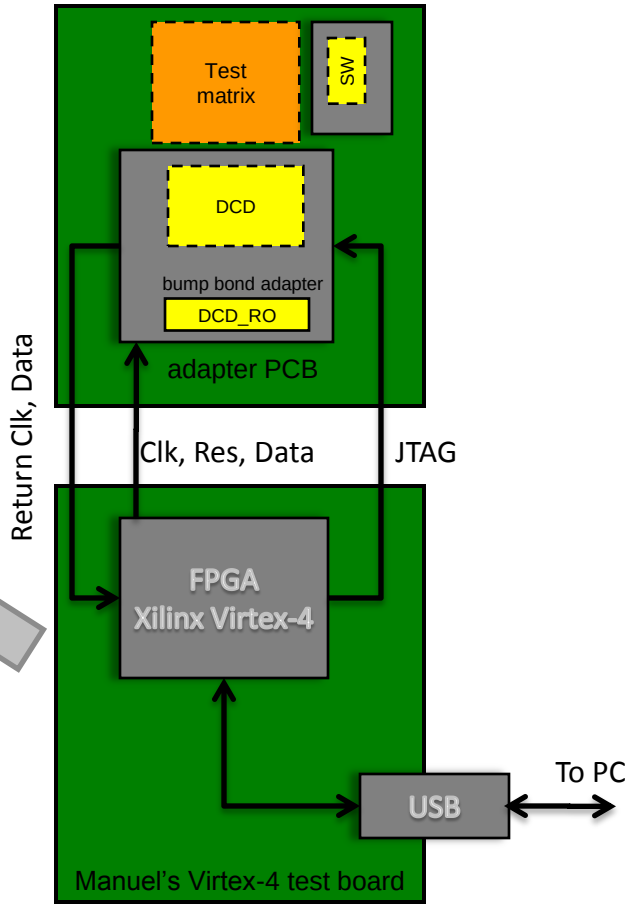


- DCD-B Test system

DEPFET test system



- DCD-B Test system

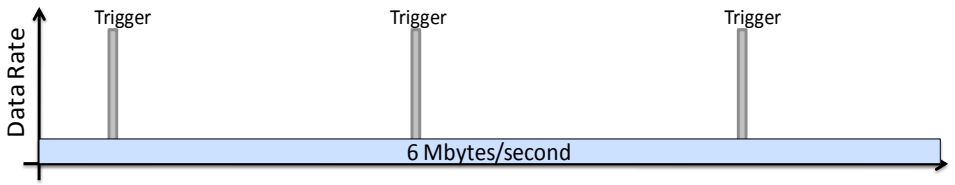


Comparative characteristics

universitätbonn

Raw Data DCD-RO output

- Frame-per-frame-wise operation equal to 4104bytes of data for a 64x64 currently connected matrix
- Best possible estimated Frame rate ~ 1.5kHz

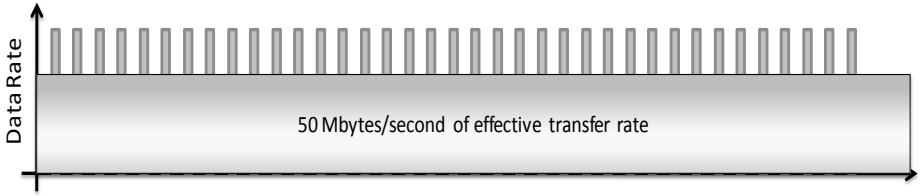


DHP emulator (Zero suppression mode)

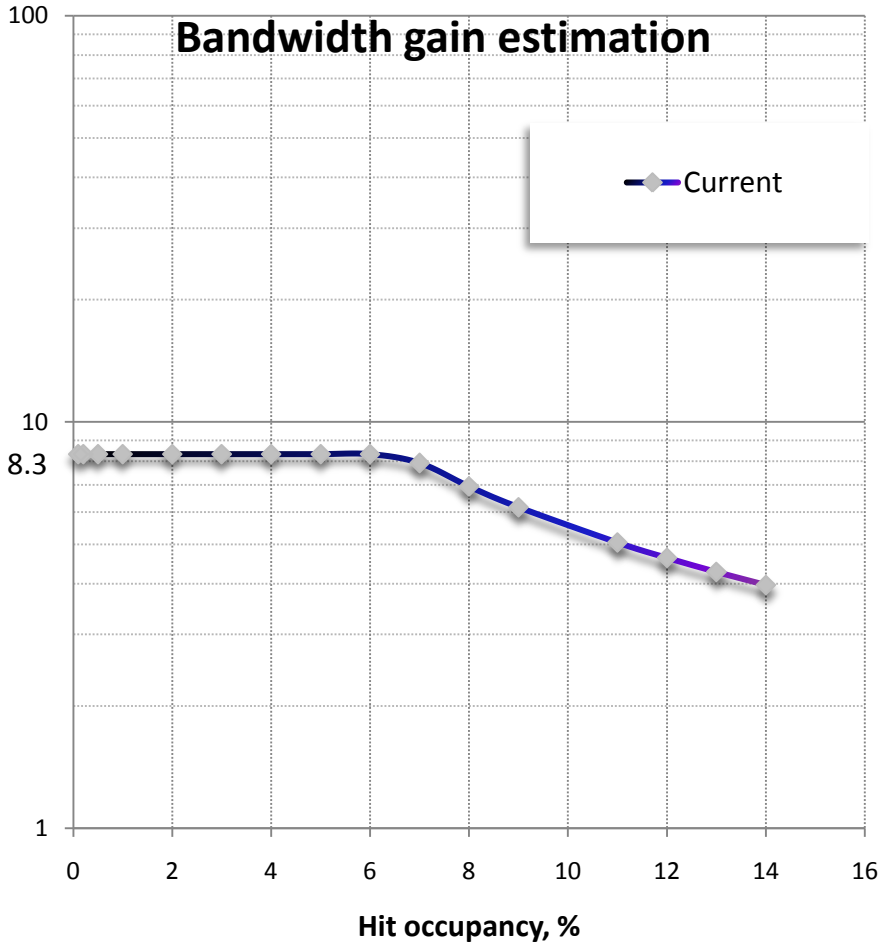
- Only zero suppressed data are sent out using format:

$$Out = [Trig_i][V_1X_1Y_1][V_2X_2Y_2]...[Trig_(i+1)]...$$

- Tested current frame rate ~12.5kHz (@ ~1% occupancy)

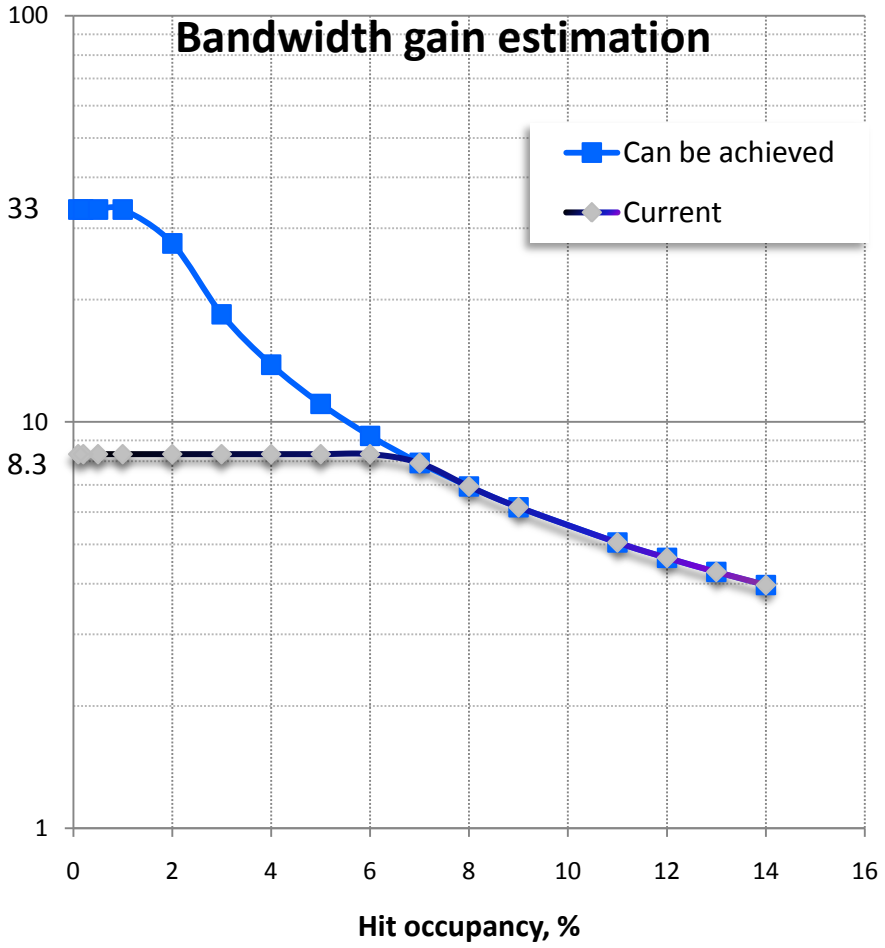


Equivalent transfer bandwidth with zero suppression.



- How faster do we transmit data using zero suppression

Equivalent transfer bandwidth with zero suppression.

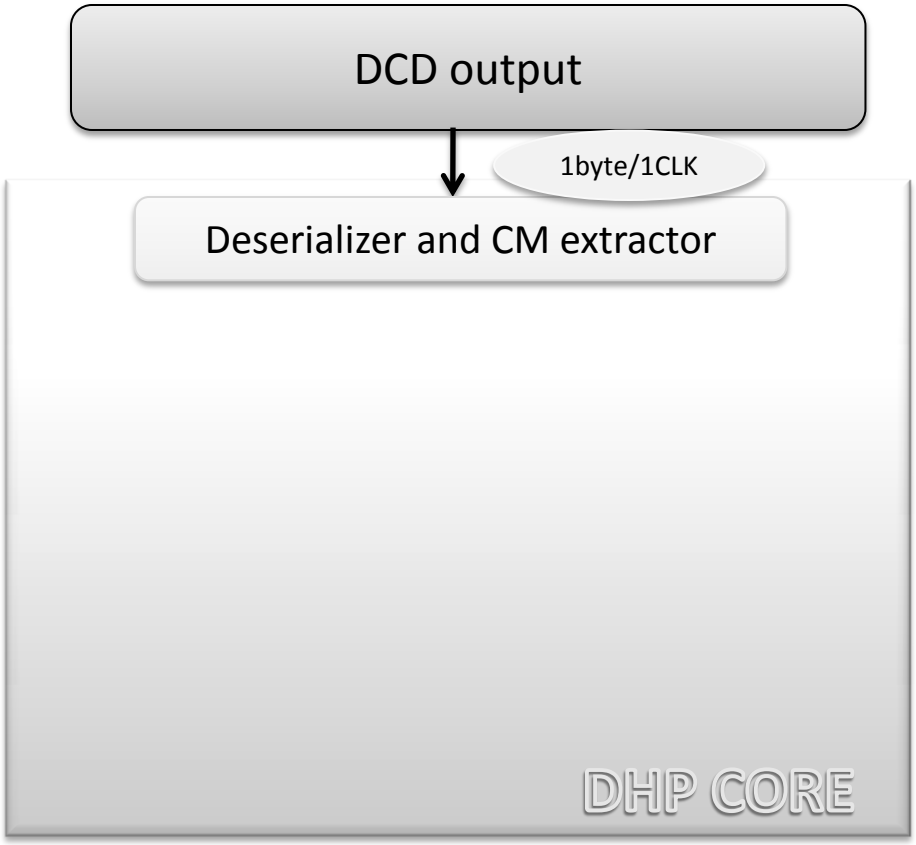


- How faster do we transmit data using zero suppression

- Still can be faster by a factor of 4 or even better

  (bottleneck optimization)

Structural schematic

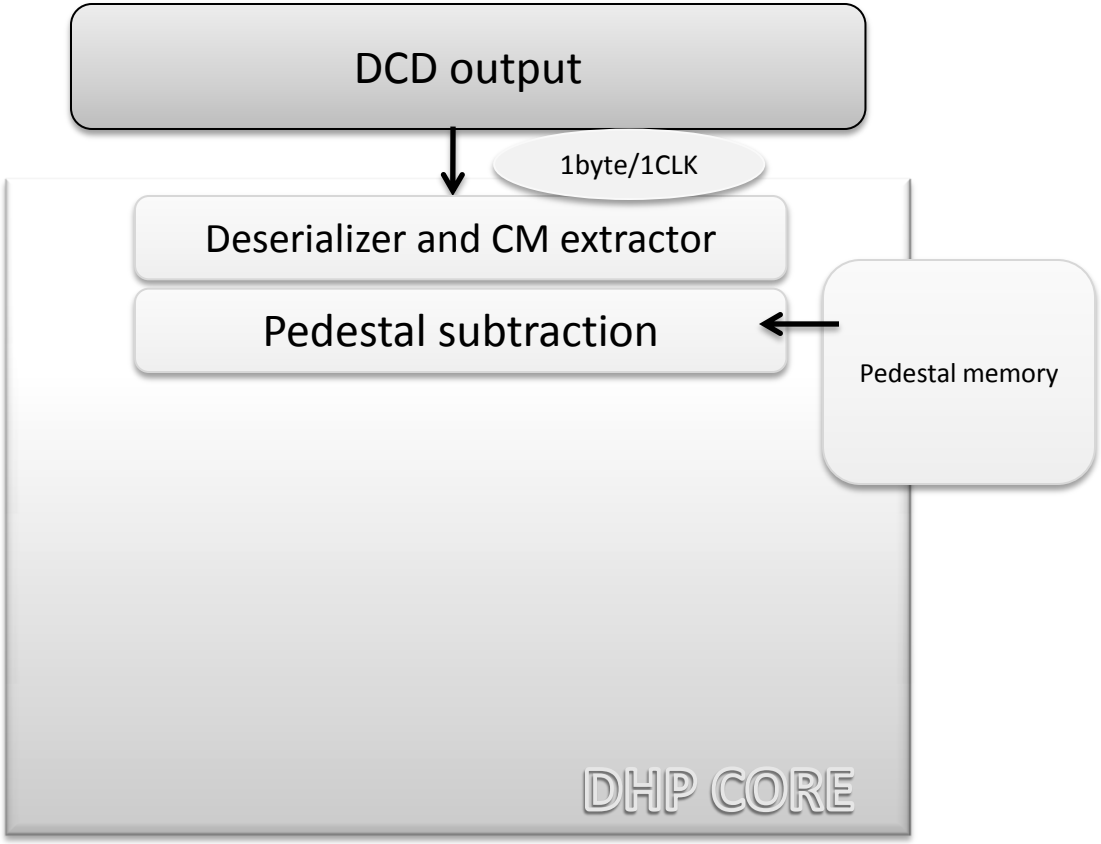


- CM subtraction and all other intermediate operations are done using 16bits precision to avoid introducing additional digitization error.
- Pedestals are stored in 16bits precision values, enough to do the dynamic pedestals update of averaging up to:

$$P_{N+1} = \frac{255P_N + I_N}{256}$$

(see DHP simulation talk)

Structural schematic

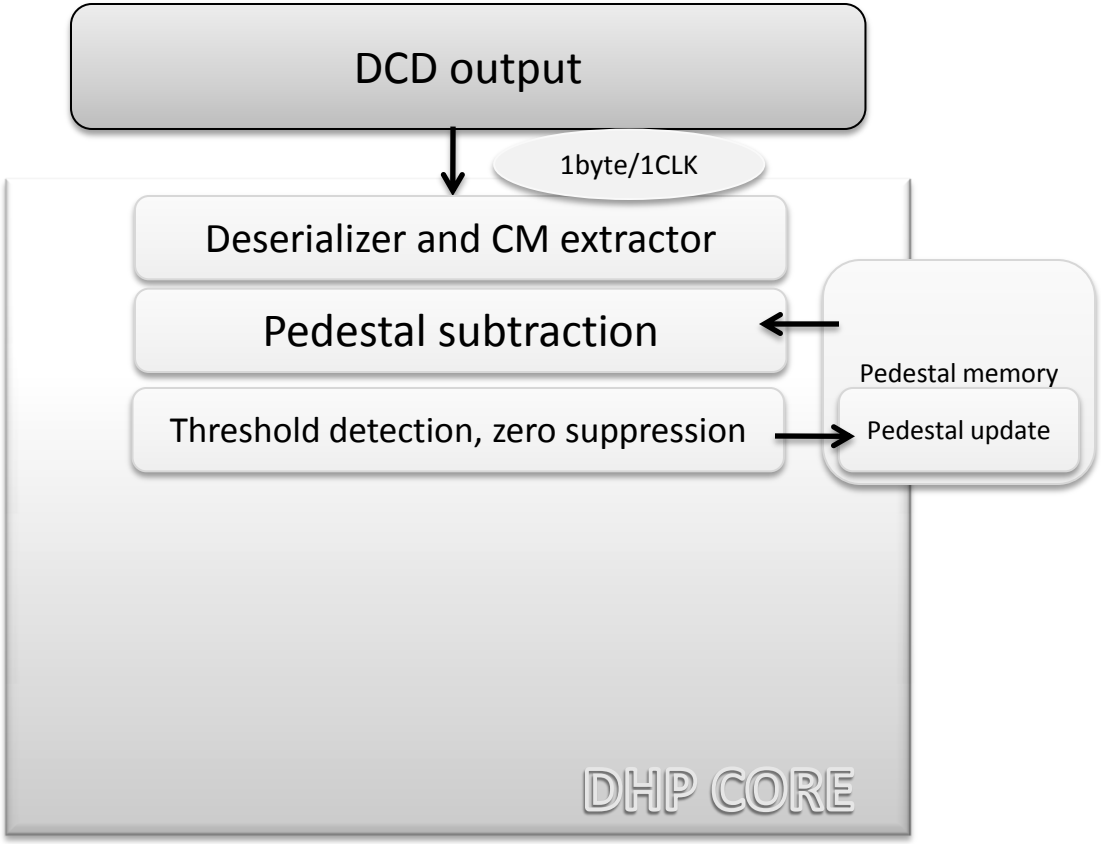


- CM subtraction and all other intermediate operations are done using 16bits precision to avoid introducing additional digitization error.
- Pedestals are stored in 16bits precision values, enough to do the dynamic pedestals update of averaging up to:

$$P_{N+1} = \frac{255P_N + I_N}{256}$$

(see DHP simulation talk)

Structural schematic

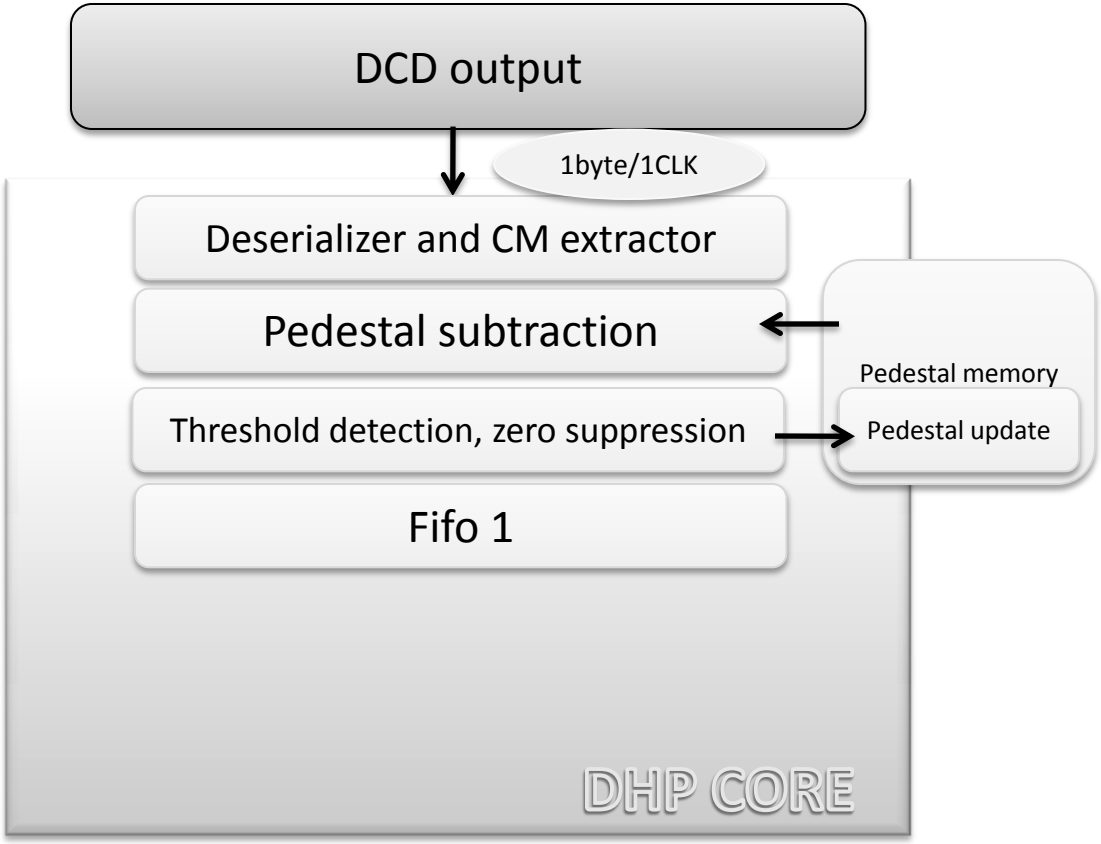


- CM subtraction and all other intermediate operations are done using 16bits precision to avoid introducing additional digitization error.
- Pedestals are stored in 16bits precision values, enough to do the dynamic pedestals update of averaging up to:

$$P_{N+1} = \frac{255P_N + I_N}{256}$$

(see DHP simulation talk)

Structural schematic

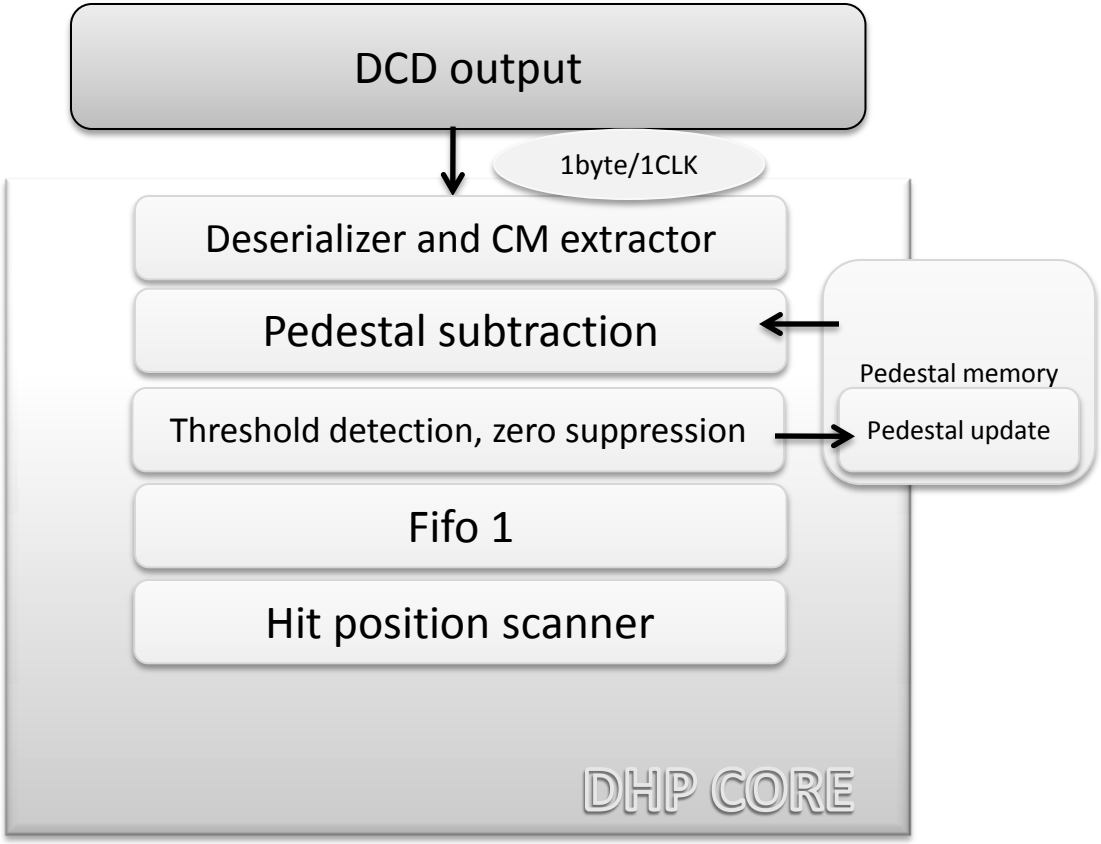


- CM subtraction and all other intermediate operations are done using 16bits precision to avoid introducing additional digitization error.
- Pedestals are stored in 16bits precision values, enough to do the dynamic pedestals update of averaging up to:

$$P_{N+1} = \frac{255P_N + I_N}{256}$$

(see DHP simulation talk)

Structural schematic

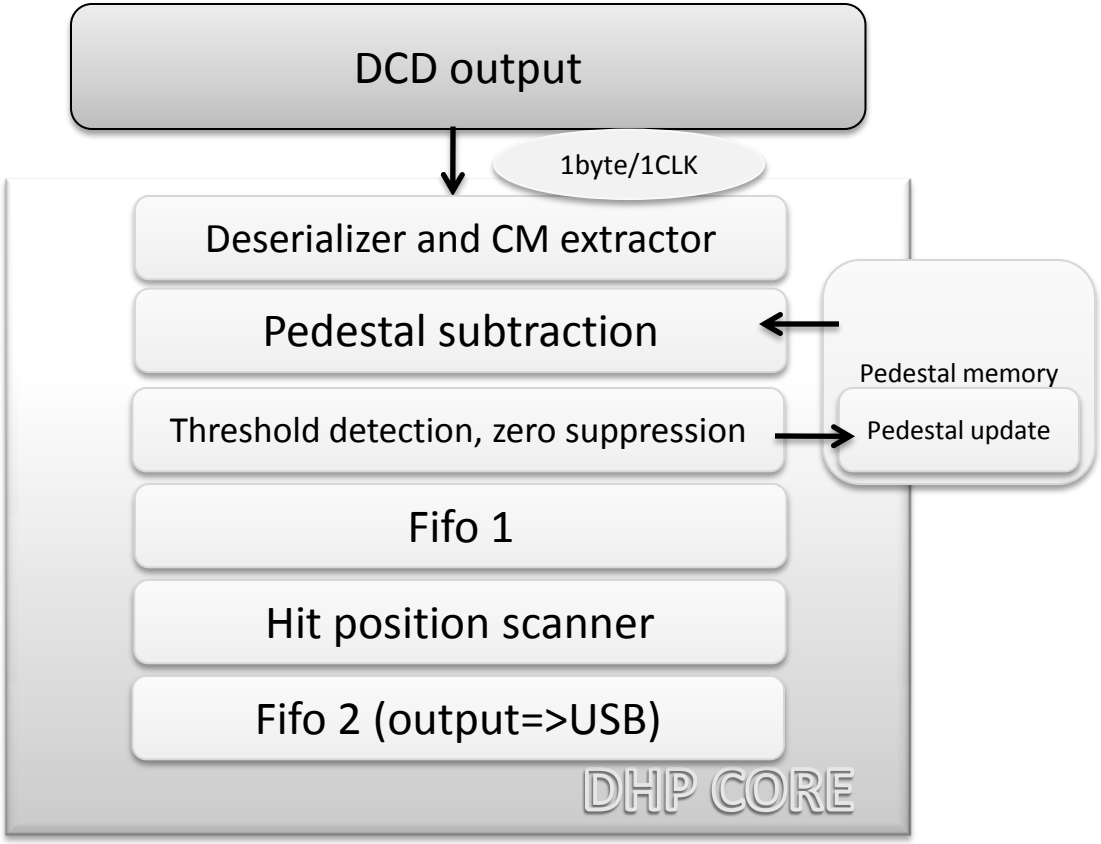


- CM subtraction and all other intermediate operations are done using 16bits precision to avoid introducing additional digitization error.
- Pedestals are stored in 16bits precision values, enough to do the dynamic pedestals update of averaging up to:

$$P_{N+1} = \frac{255P_N + I_N}{256}$$

(see DHP simulation talk)

Structural schematic

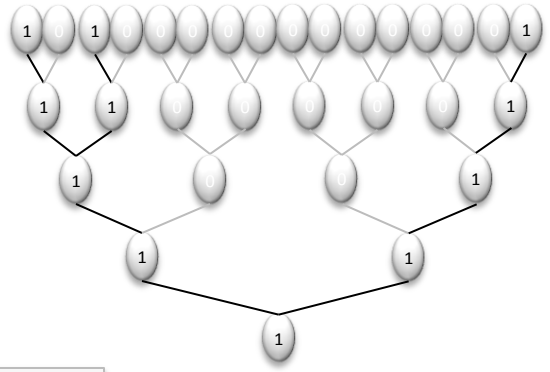


- CM subtraction and all other intermediate operations are done using 16bits precision to avoid introducing additional digitization error.
- Pedestals are stored in 16bits precision values, enough to do the dynamic pedestals update of averaging up to:

$$P_{N+1} = \frac{255P_N + I_N}{256}$$

(see DHP simulation talk)

Hit position scanner

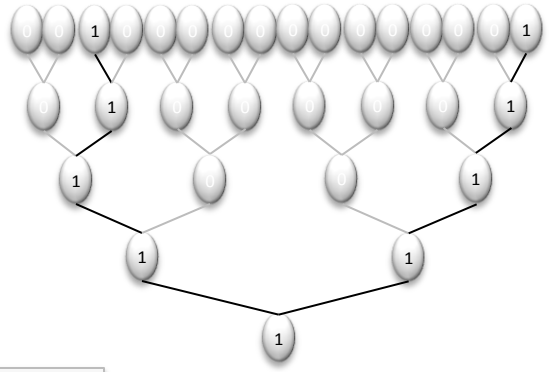


Left is 0
Right is 1

Hit found with path from bottom to top:
left-left-left-left ⇔ [0000] = position 0

- Although seemingly trivial task may be a serious bottleneck if not seriously taken in consideration
- Hit scanner is implemented using ‘binary tree’ technique, inspired from DHP predecessors (MEPHISTO, CURO etc…)
- Tree is updated asynchronously, giving one hit position per clock

Hit position scanner



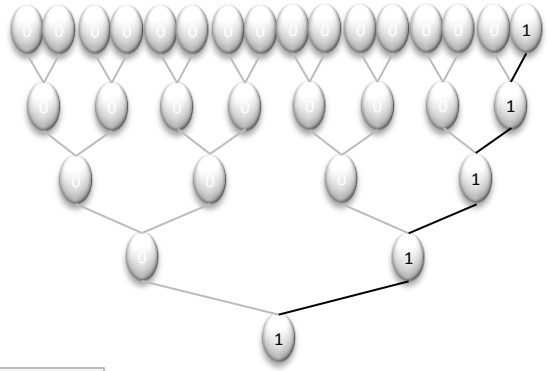
Left is 0
Right is 1

Hit found with path from bottom to top:
left-left-left-left ⇔ [0000] = position 0

Hit found with path from bottom to top:
left-left-right-left ⇔ [0010] = position 2

- Although seemingly trivial task may be a serious bottleneck if not seriously taken in consideration
- Hit scanner is implemented using ‘binary tree’ technique, inspired from DHP predecessors (MEPHISTO, CURO etc…)
- Tree is updated asynchronously, giving one hit position per clock

Hit position scanner



Left is 0
Right is 1

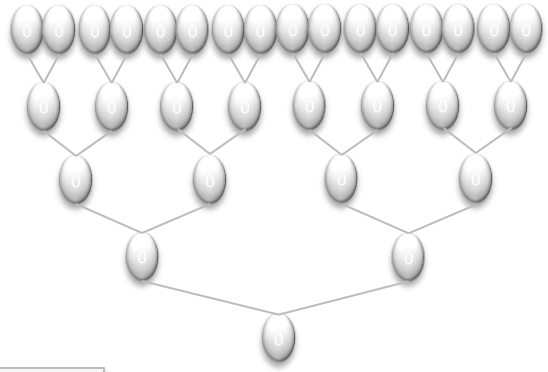
Hit found with path from bottom to top:
left-left-left-left ⇔ [0000] = position 0

Hit found with path from bottom to top:
left-left-right-left ⇔ [0010] = position 2

Hit found with path from bottom to top:
right-right-right-right ⇔ [1111] = position 15

- Although seemingly trivial task may be a serious bottleneck if not seriously taken in consideration
- Hit scanner is implemented using ‘binary tree’ technique, inspired from DHP predecessors (MEPHISTO, CURO etc…)
- Tree is updated asynchronously, giving one hit position per clock

Hit position scanner



Left is 0
Right is 1

Hit found with path from bottom to top:
left-left-left-left ⇔ [0000] = position 0

Hit found with path from bottom to top:
left-left-right-left ⇔ [0010] = position 2

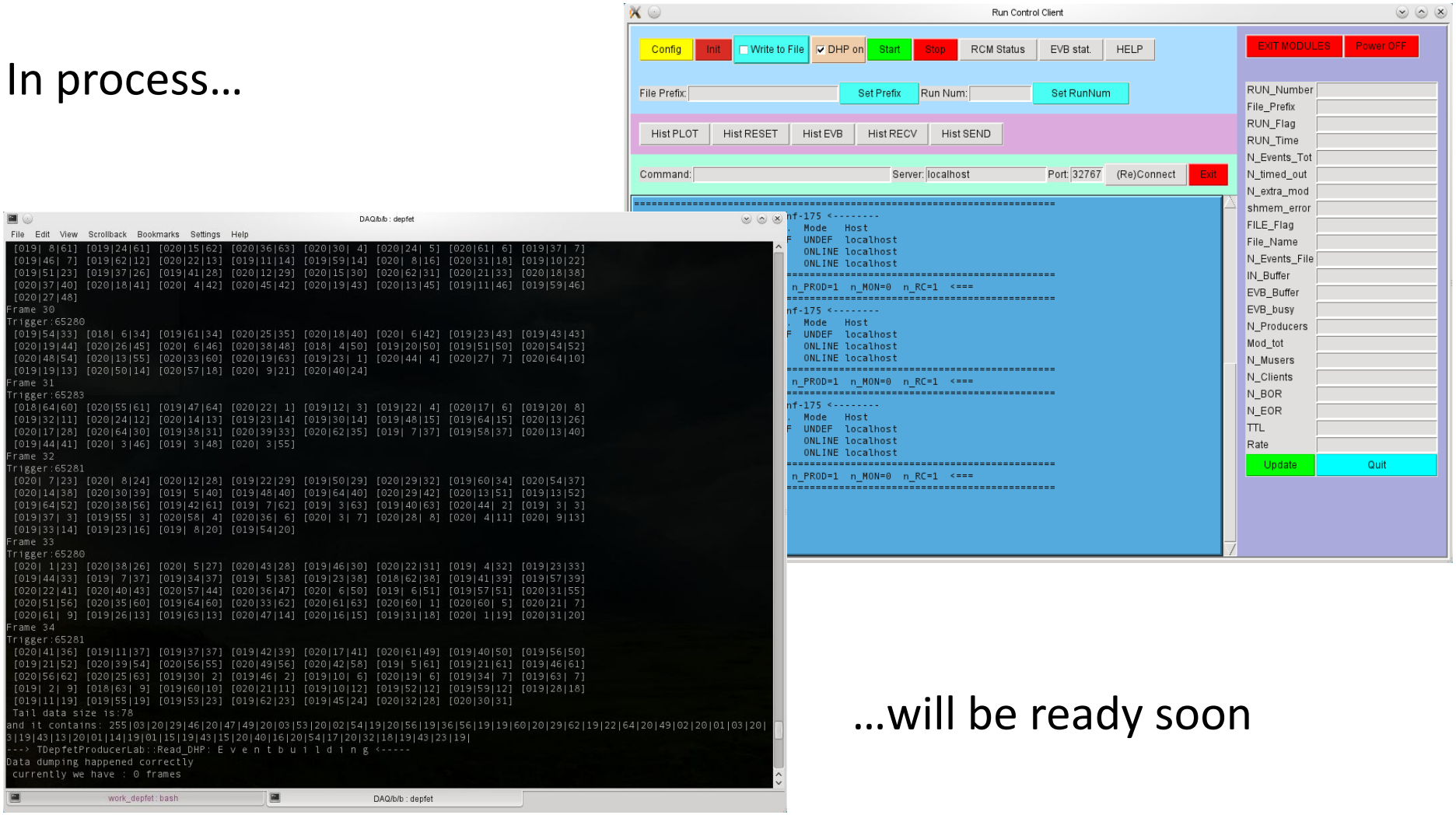
Hit found with path from bottom to top:
right-right-right-right ⇔ [1111] = position 15

All hits are found in 3 clock cycles instead of 16

- Although seemingly trivial task may be a serious bottleneck if not seriously taken in consideration
- Hit scanner is implemented using ‘binary tree’ technique, inspired from DHP predecessors (MEPHISTO, CURO etc…)
- Tree is updated asynchronously, giving one hit position per clock

Integration to the DAQ

In process…



…will be ready soon

What do we want to learn from it?

universitätbonn

- Although data processing is initially checked with the software simulation, everything have to be verified for its feasibility in hardware
- All features can be tested in real conditions
- DHP emulator can be used during the next test beam together with the regular DCD readout system
  - Read data with the zero suppressed mode
  - Faster or continuous readout allow us to check several DEPFET matrix properties impossible to be seen with the slow readout, such as complete clear check

(To be done)

- Clustering
  - Depends a lot on average cluster size
- Increase input bandwidth

universitätbonn

THANK YOU