



SVT-Layer0 Pixels Readout Architectures

Filippo Maria Giorgi - INFN Bologna



XV SuperB General Meeting
Caltech, CA – USA, Dec. 14-17 2010



- New Matrix Scan Logic (respect to SPX0)
- Triggered Architecture
- Integration achievements
- Optimizations
- Simulations
- 2011 Submissions
- Conclusions

Summary

Respect to previous submission SuperPX0:

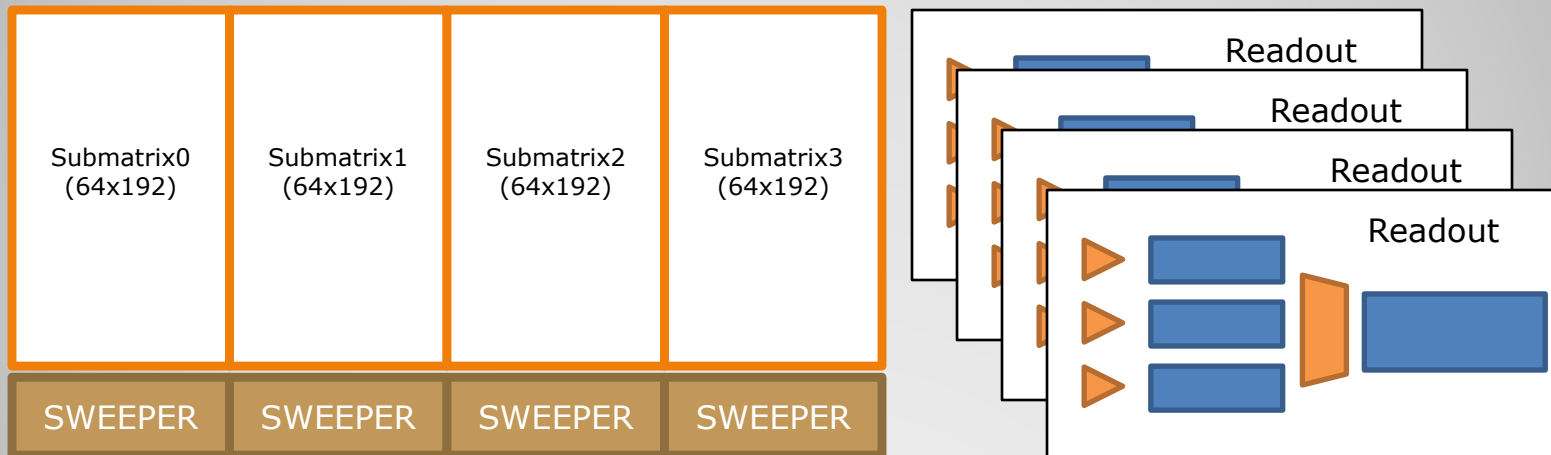
- **Dense in-pixel digital logic** (Time labeling, arbitrary TS comparator for time ordered readout)
- **NO Macro-Pixels** → no FREEZING required → **much less dead area**
- **NO *Scan Buffer*** (saving RO area) but still **time ordered matrix scans**
- **Smaller BC periods allowed** → **better time resolution**
- **Polyvalent Triggered & Data-push architecture**

New Matrix Scan Logic

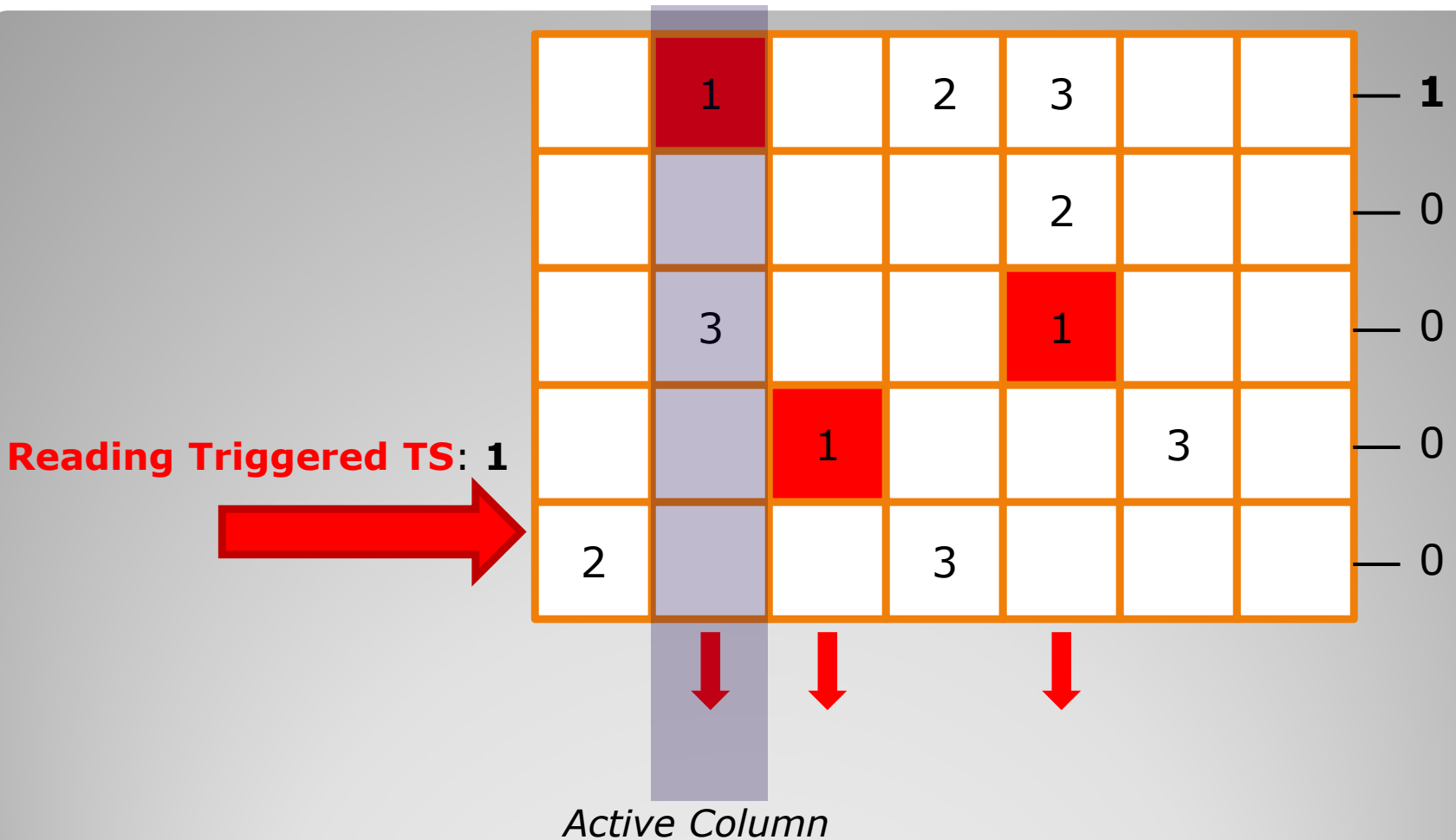
Data-push architecture for pixels on Layer0 requires a lot of output bandwidth. ($100\text{MHz/cm}^2 \times \sim 20\text{bit} = \sim 2\text{ Gbps/cm}^2$)

Some modifications, involving the sweeper architecture only, make possible to exploit the **matrix itself as a hit buffer** for a triggered architecture.

We are evaluating if this solution is viable taking into account the expected maximum trigger latency ($\sim 6\ \mu\text{s}$) of SuperB.

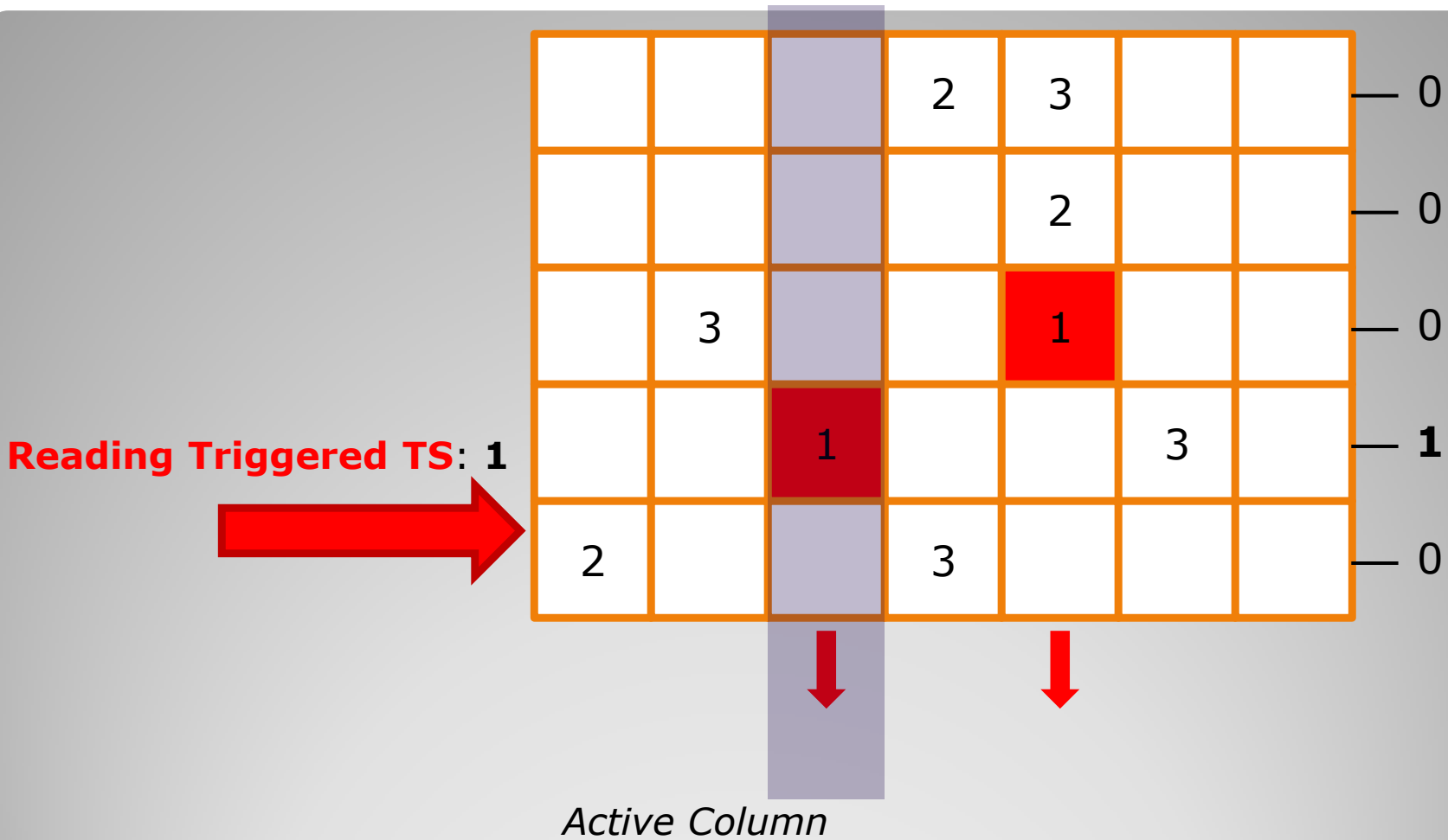


Triggered Architecture



Only triggered time stamps are requested and swept out.

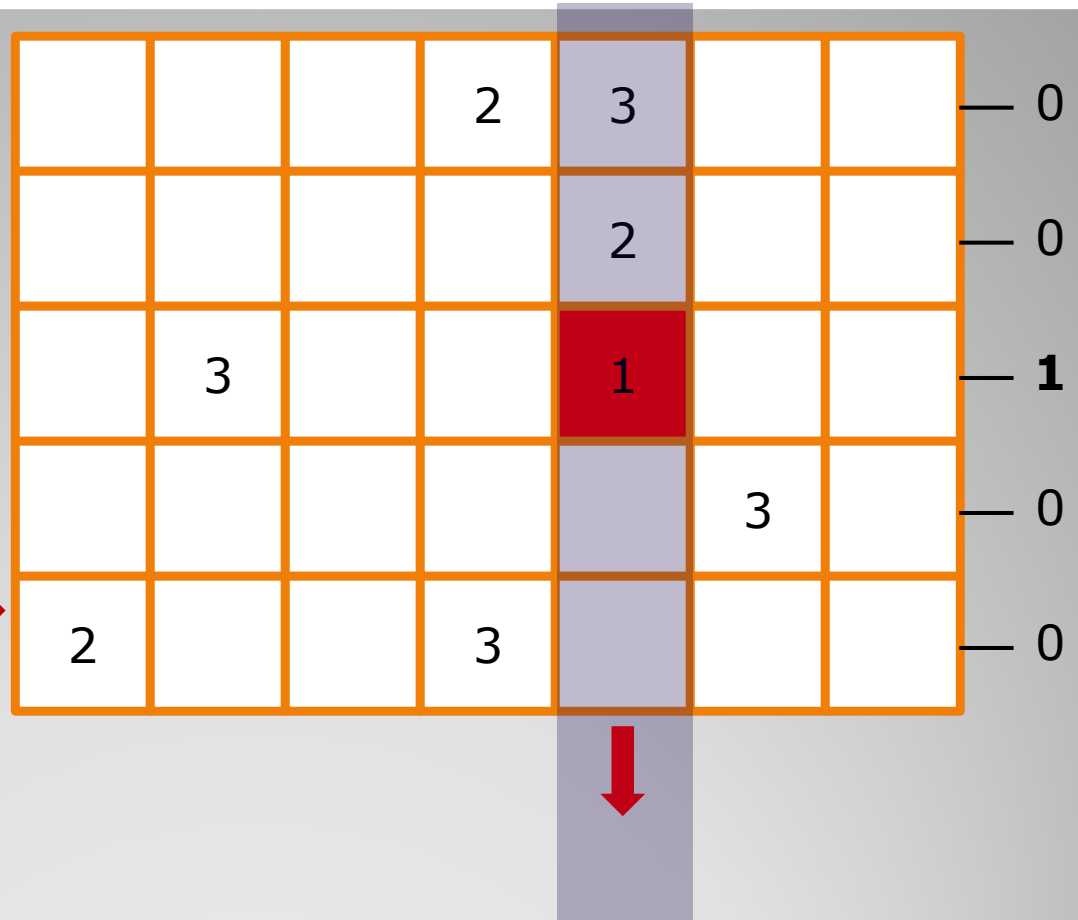
Triggered Architecture



Only triggered time stamps are requested and swept out.

Triggered Architecture

Reading Triggered TS: **1**

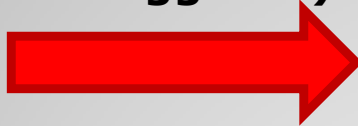


Active Column

Only triggered time stamps are requested and swept out.

Triggered Architecture

Request TS: 2
(was NOT Triggered)

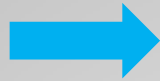


			2	3			0
					2		0
	3						0
						3	0
2				3			0

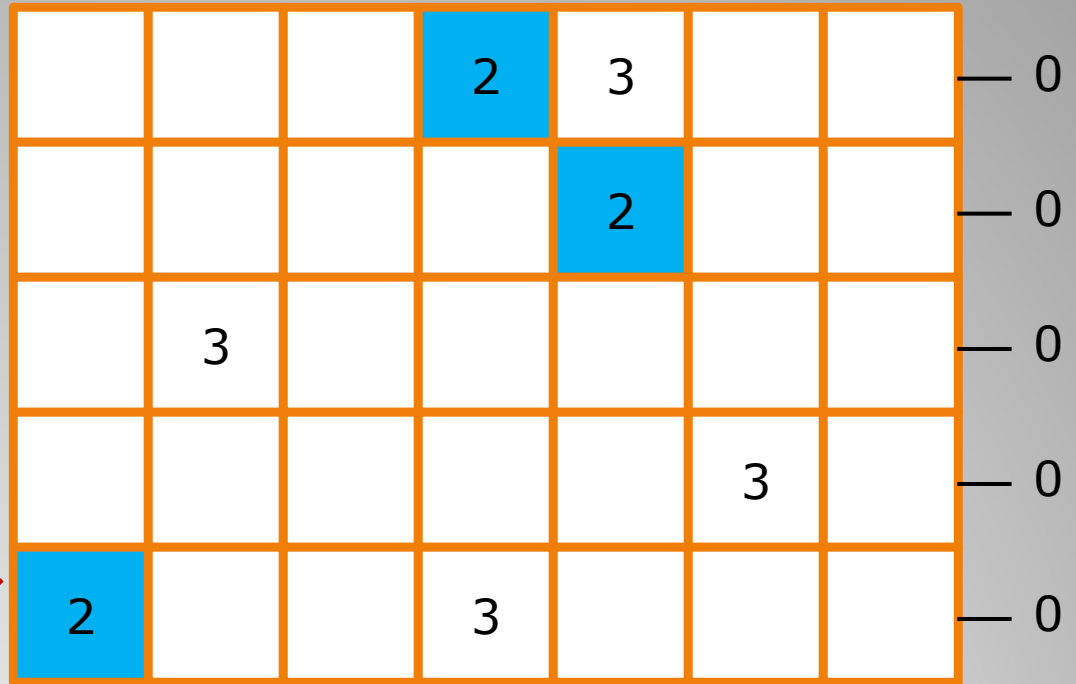
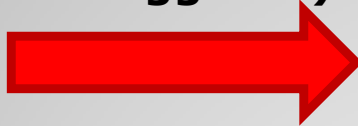
After the latency, a TS-dependent reset is asserted. Only corresponding pixels are reset

Triggered Architecture

TS dependent Reset



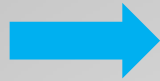
Request TS: 2
(was NOT Triggered)



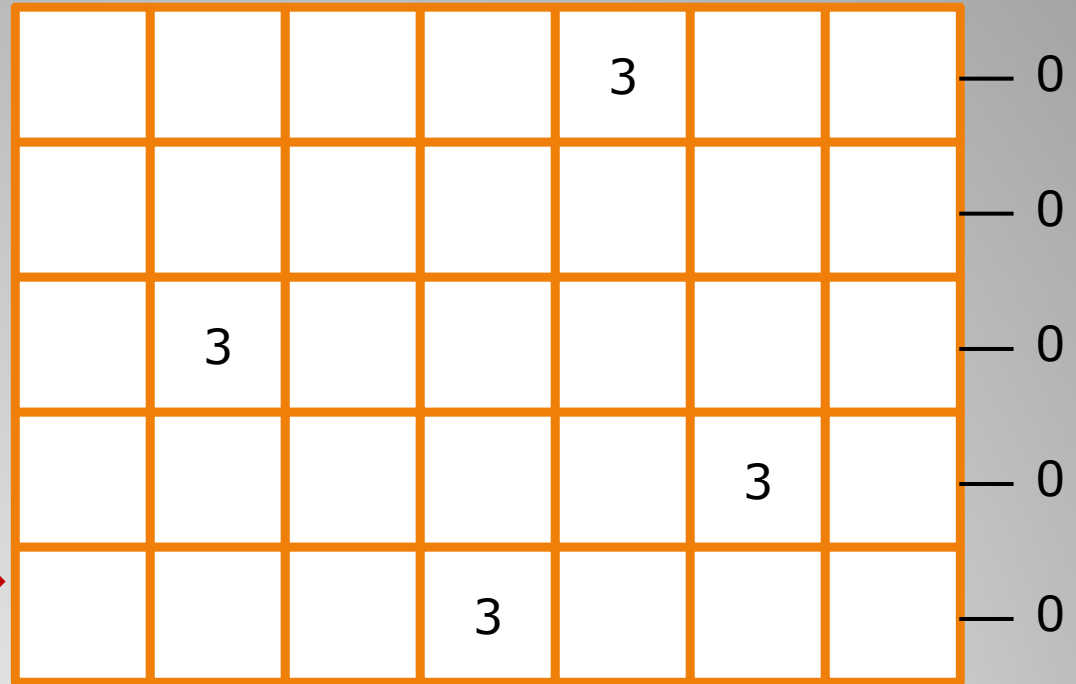
After the latency, a TS-dependent reset is asserted. Only corresponding pixels are reset

Triggered Architecture

TS dependent Reset



Request TS: 2
(was NOT Triggered)



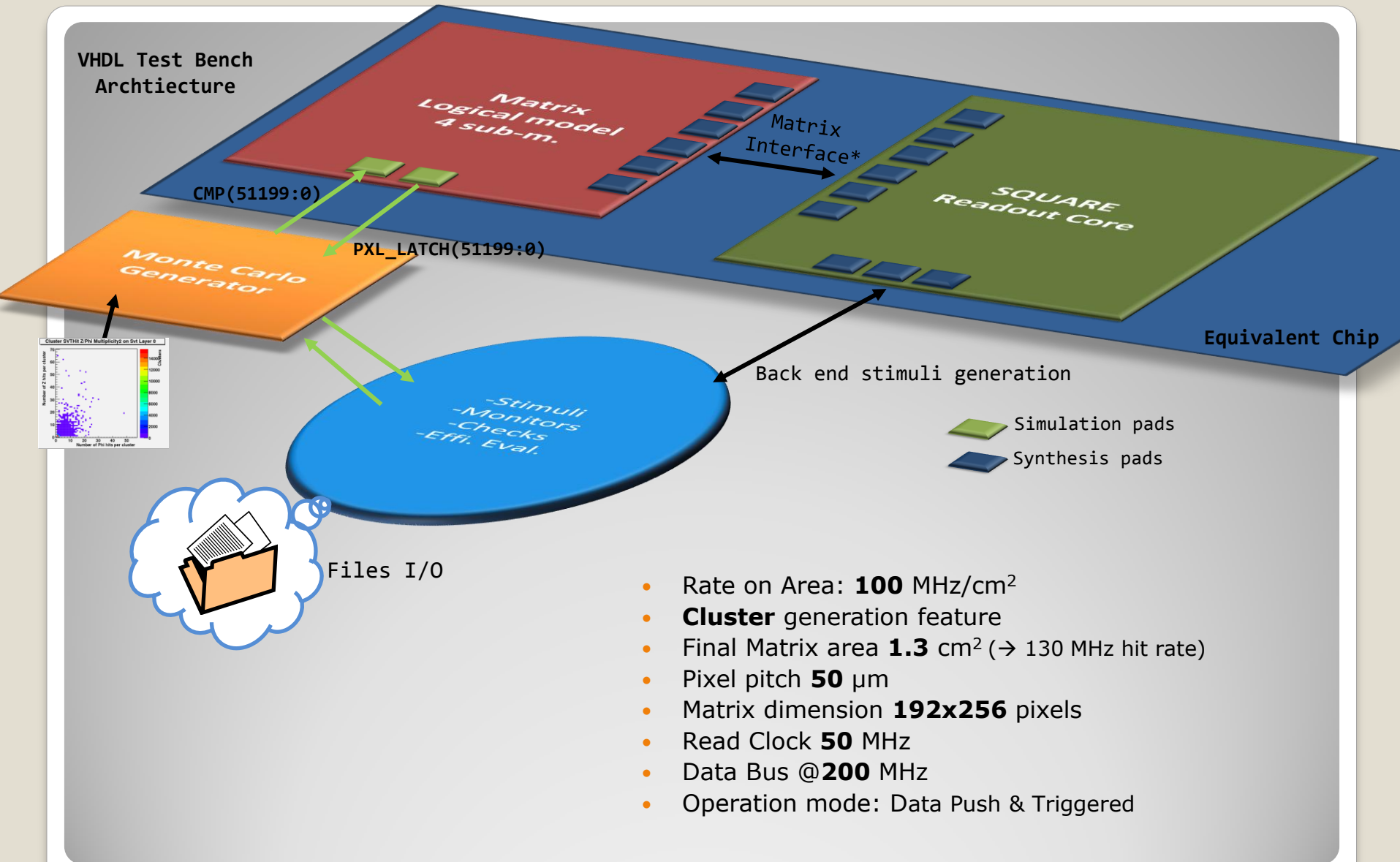
After the latency, a TS-dependent reset is asserted. Only corresponding pixels are reset

Triggered Architecture

- All the readout architecture is coded in synthesizable VHDL. Now, **also the triggered extraction** feature of the sweeper.
 - Efficiency evaluations conducted with Monte Carlo hit extraction.
- **Barrels and Sweeper code optimization** for higher speed & lower synthesis time.
 - Full architecture (with optimized components) entirely **re-integrated** and **re-simulated** (final matrix dimensions 192x256)
 - Full architecture synthesized in Synopsys environment. Benefits are presented.

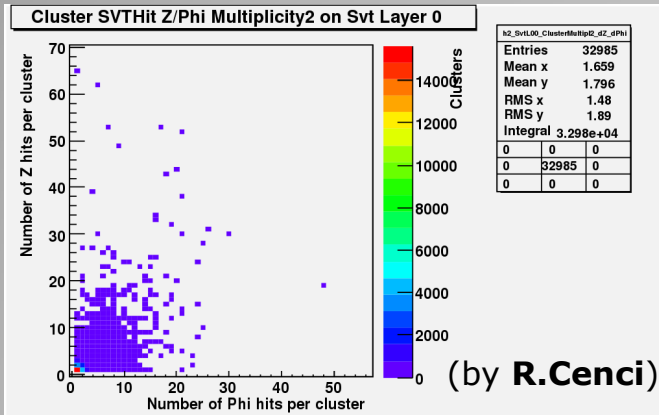
Integration achievements

VHDL Test Bench Architecture

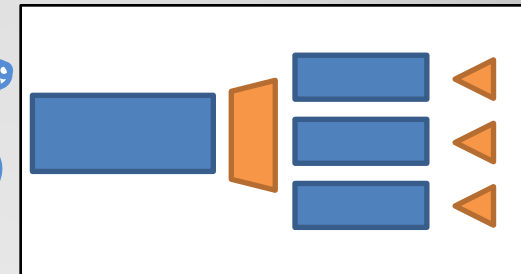
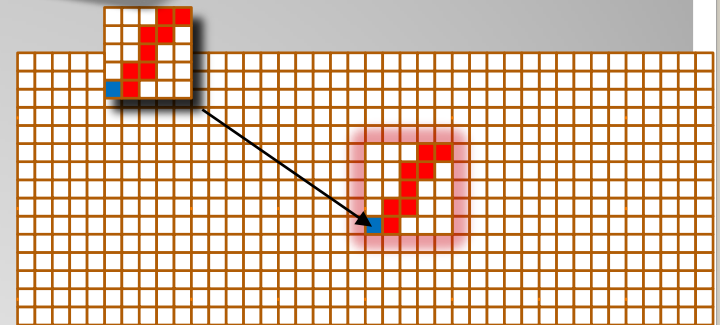


Simulations overview

Input Cluster spread dist. From Physics



MC hit extraction

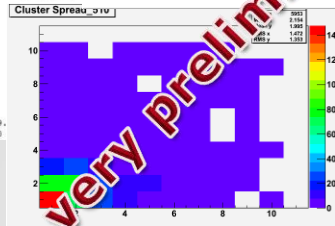
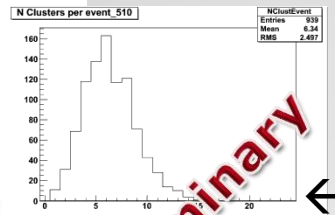


Architecture Simulation Run

```

SIM_NUMBER          520
Debug MC
Core Version         4.5c
----- MATRIX DIMENSIONS -----
N_XSUBMAT           4
N_YSUBMAT           1
SUBMAT_X_SIZE       50
SUBMAT_Y_SIZE       256
pxl_pitch (mm) =    0.050
Matrix area (mm2) = 32.000 x 4 x 1 = 128.000
----- READOUT PARAMETERS -----
Zone size =         8
B2 depth =          32
B1 depth =          128
----- READOUT CONFIGURATIONS -----
calib mode =        0
Clock ratio fast_clk/Bclk = 4
Funnel operating mode = 0
Rate range =        3
Phase select =      3
Fout timeout =      256
Data push / Triggered mode = Data push
----- READOUT FLAG ($ sim. ended) -----
Funnel err =        0 0 0 0
Funnel OUT err =    2
B1 full flags (latched) = 0 0 0 0
B1 mean fillings = 1474 1737 808 1943
B2 full flags (latched) = (0 0 0 0) (0 0 0 0)
----- SIMULATION PARAMETERS -----
Rdclk (MHz) =        50.000
fast_clk (MHz) =     200.000
BCO period (us) =    0.150
Generation Time Threshold (%) = 99.800000
Generation Granularity (test_clock) (ms) = 1.000
Sim duration (us) =  3000.000
----- SIMULATION RESULTS -----
Number of words sent (TS + hit words) = 359061
TS read at output =  78932
(B1_words - T2readout)/TOT_LATCHED_COUNTER = 0.725
Total already hit efficiency = 203 / 386688 = 99.
Total readout efficiency = 386485 / 386485 = 100.000
    
```

Reports, efficiency, analysis...



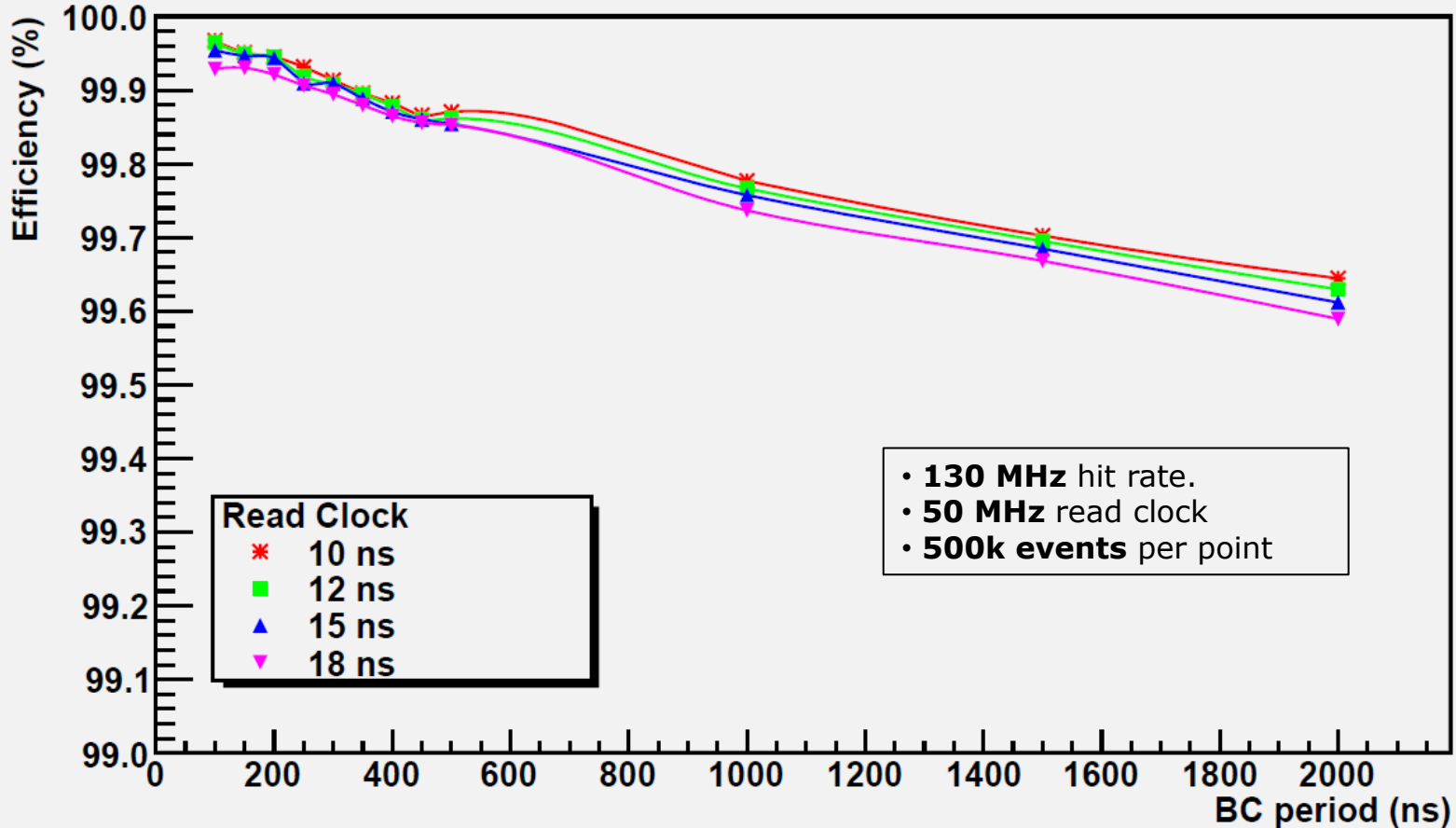
very preliminary



MC tuned to obtain 100 MHz/cm² **hit** rate on area

Simulation Results

Efficiency vs BC clock

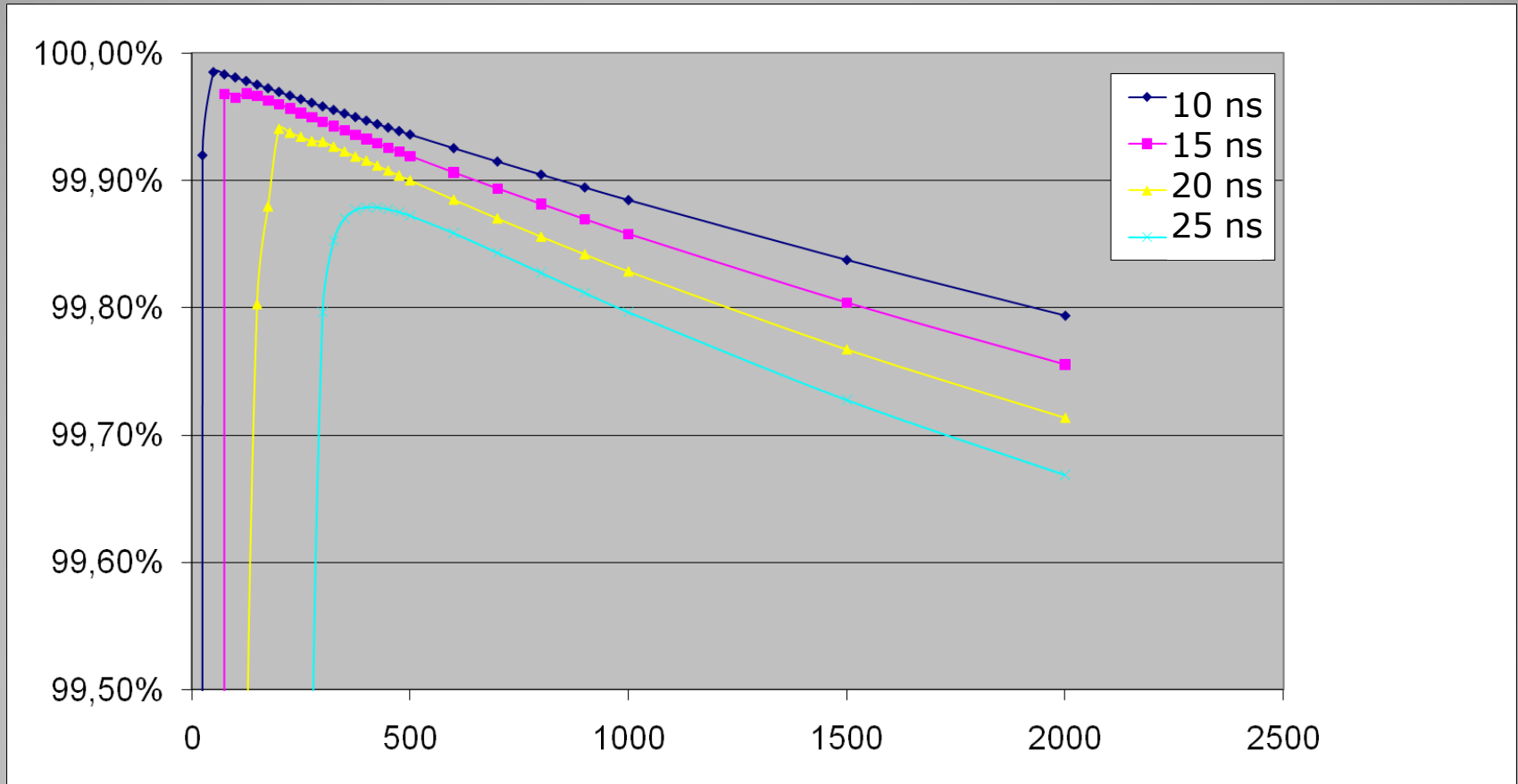


NOT taken into account:

- sensor efficiency (assumed 100%)
- pixel reset dead time (assumed few ns)

Simulation results DATA PUSH

Expected efficiency combinatorial evaluations



NOT taken into account:

- sensor efficiency (assumed 100%)
- pixel reset dead time (assumed few ns)

Analitic expectation DATA PUSH

Data push Efficiency results:

SuperPX1 DATA PUSH arch. BC (ns)

%	200	250	300
66.7	99.93	99.92	99.92
55.6	99.93	99.92	99.91
50	99.92	99.92	99.90

RD
(MHz)

NOT taken into account:

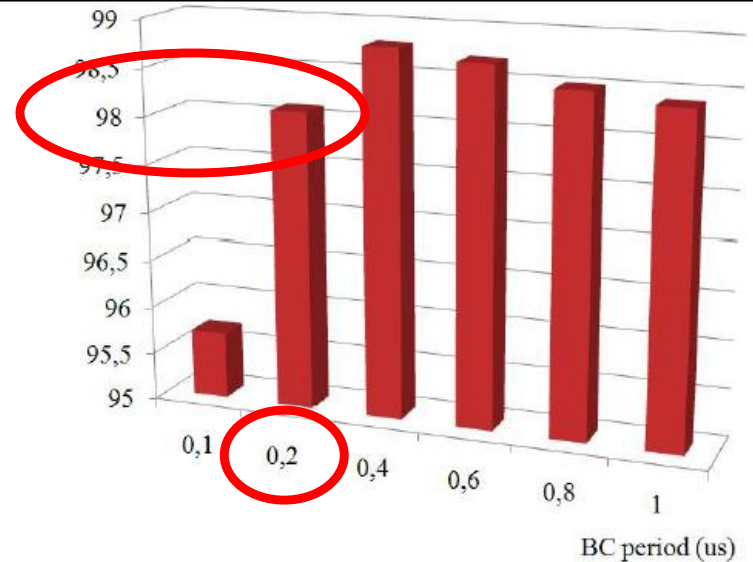
- sensor efficiency (assumed 100%)
- pixel reset dead time (assumed few ns)

Readout de-queuing efficiency 100% (no barrel overflows)

- **Hit check** results: **100 % match**.
- Fast_clock **4 x** RDclk (output bus frequency)

Compare with *SuperPX0* data push arch.

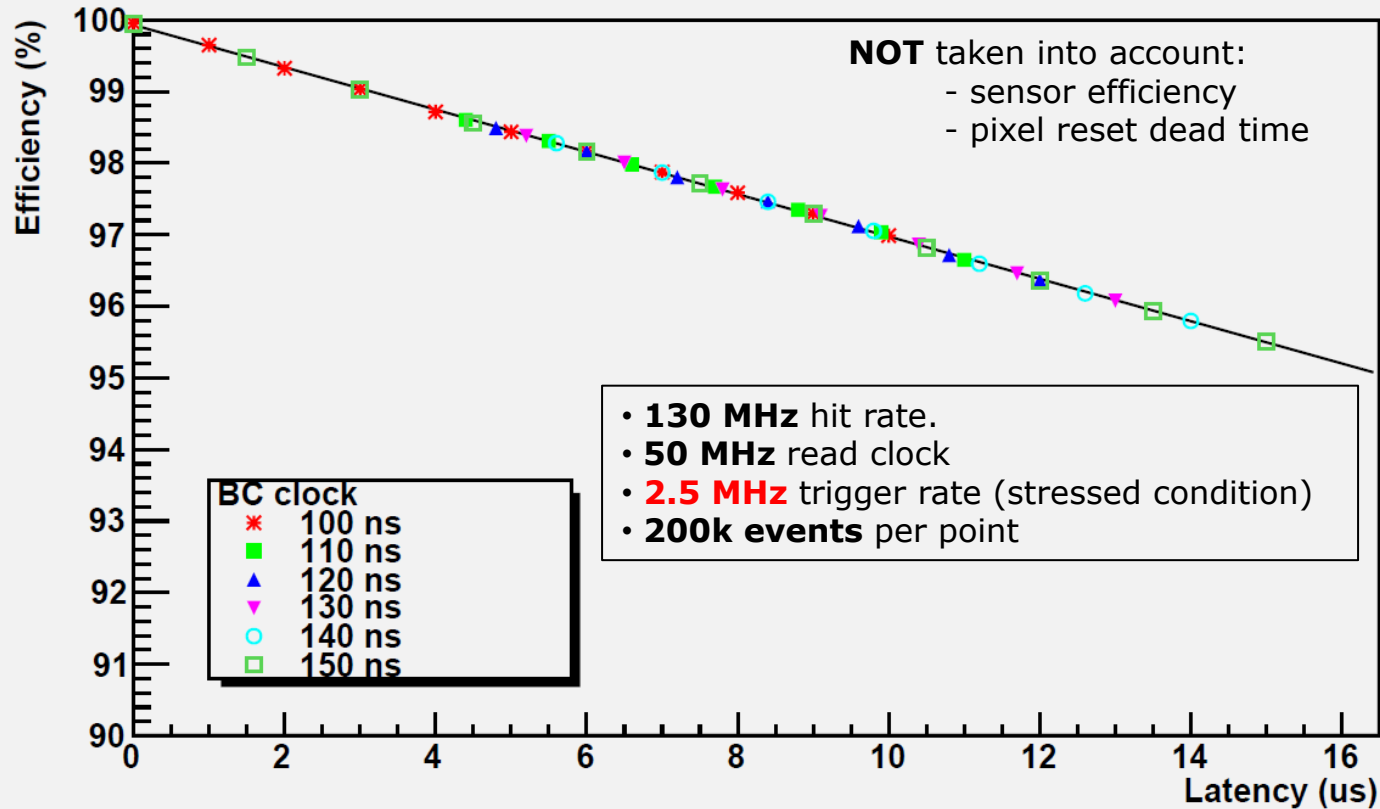
SUPERPX0 - RDclk **66.67 MHz** - Fast_clk **200 MHz (3x)**



efficiency results from similar simulations of *SuperPX0* readout

SuperPX0 comparison

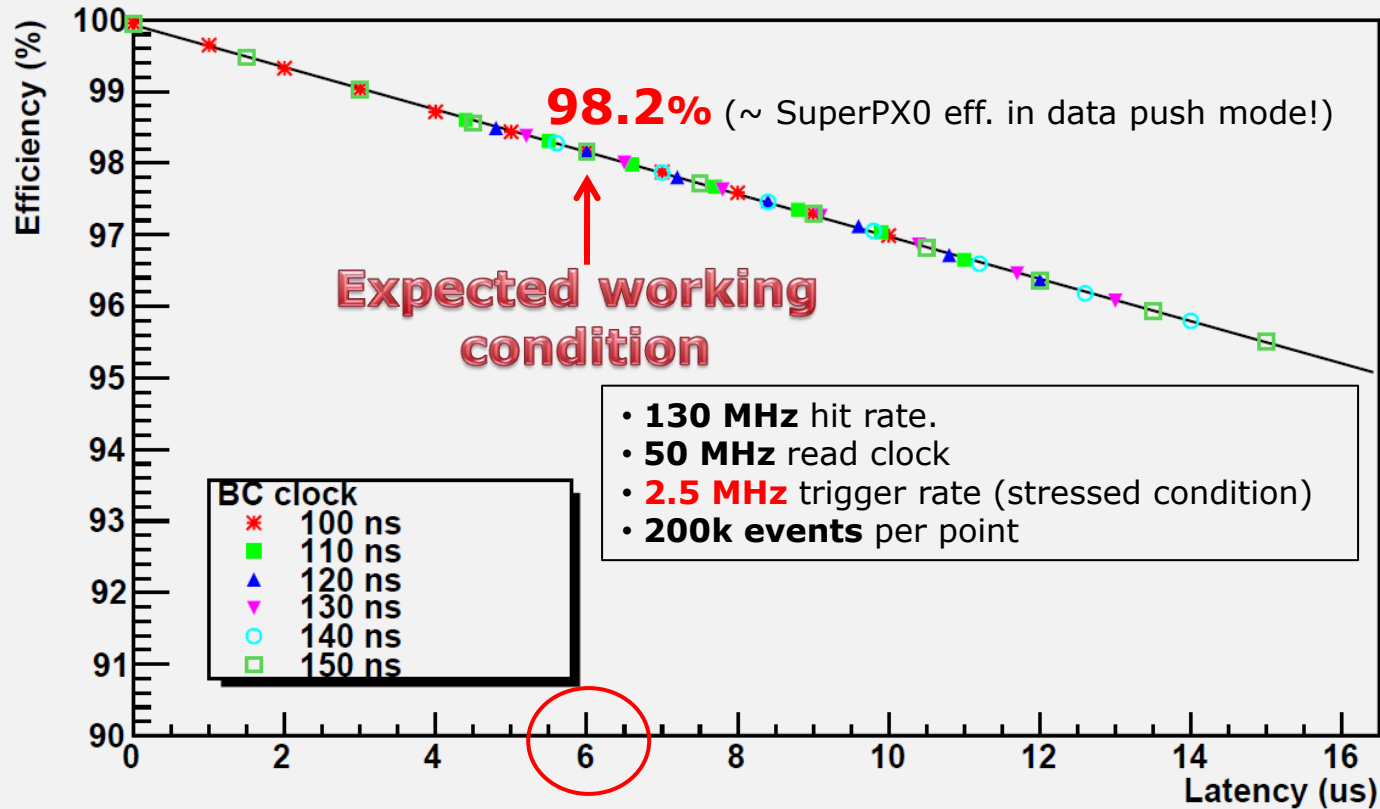
Efficiency vs Latency



- Smooth decrease of efficiency in function of trigger latency.
- Almost **no dependency** of efficiency **on BC period** (in this region)
- Linear fit slope: **-0.3 %/us**.

Simulation results, TRIGGERED

Efficiency vs Latency



- Smooth decrease of efficiency in function of trigger latency.
- Almost **no dependency** of efficiency **on BC period** (in this region)
- Linear fit slope: **-0.3 %/us**.

Simulation results, TRIGGERED

Bandwidth usage estimated by simulations

data bus: 20 bit @ 200 MHz bus → 4 Gbps max throughput.

•**Data push mode**

- BC = 100 ns (10 MHz)

- Rate = 100 MHz/cm²

mean bandwidth usage of 2.6 Gbps

~22% bandwidth saving thanks to zone clusterization algorithm and time bundling of hits. (**respect to APSEL 4D** standard *xyt* hit word encoding)

•**Triggered mode**

- BC = 100 ns (10 MHz)

- Rate = 100 MHz/cm²

- Trigger Rate = **2.5 MHz** (largely overestimated, 1 trig. every 4 BC)

mean bandwidth usage of 660 Mbps

(corresponding to **~40 Mbps** for a standard **150 kHz trigger rate**).

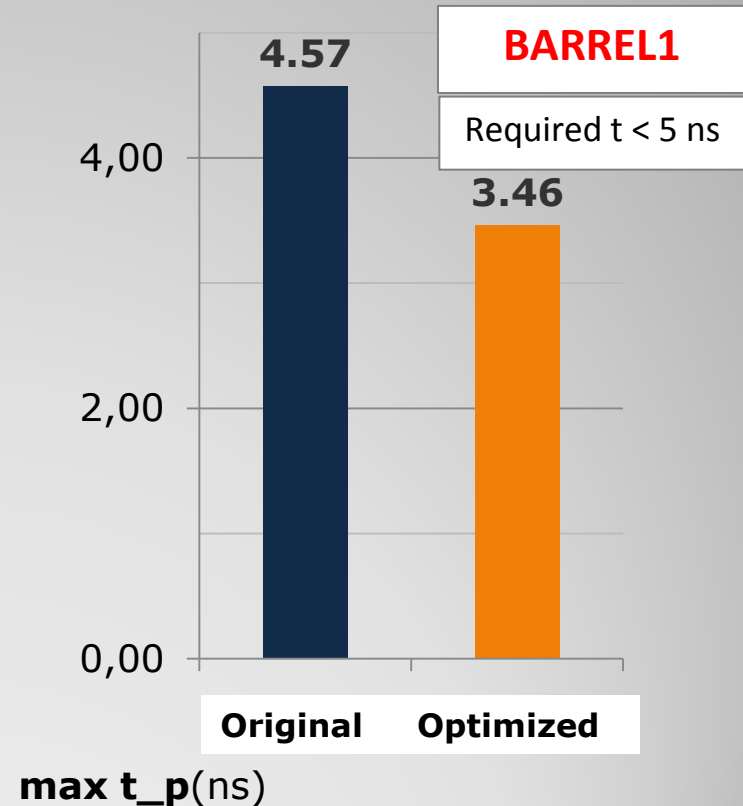
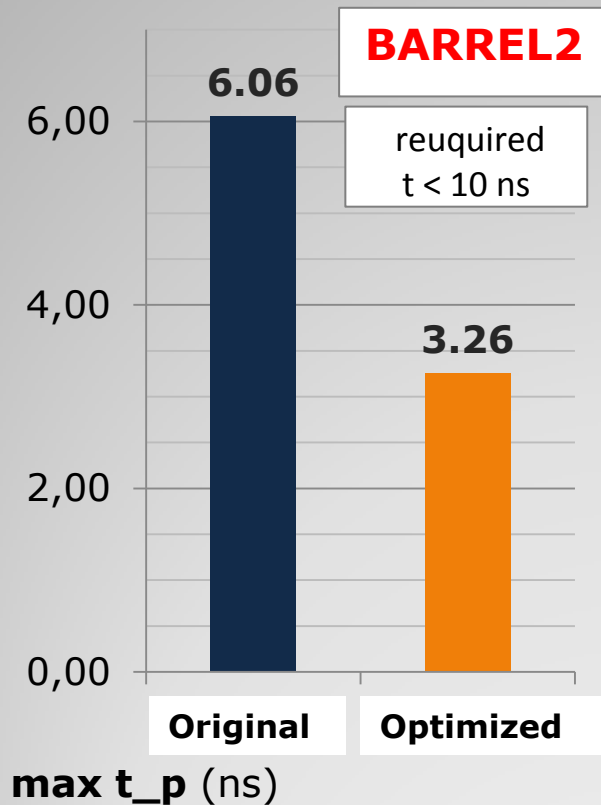
Simulation results: BANDWIDTH

- **Barrel and Sweeper were described in high-level VHDL code.**
 - Synthesis slow
 - Generated net-list was not optimized → improvable speed performances
- **Thesis on code optimization,** to be discussed this week in Bologna.
 - Barrels and Sweeper **rewritten almost at hardware level.**
 - Evident performance improvements are reported by the Synopsys Design Compiler tool.

Optimizations

Barrels speed optimization

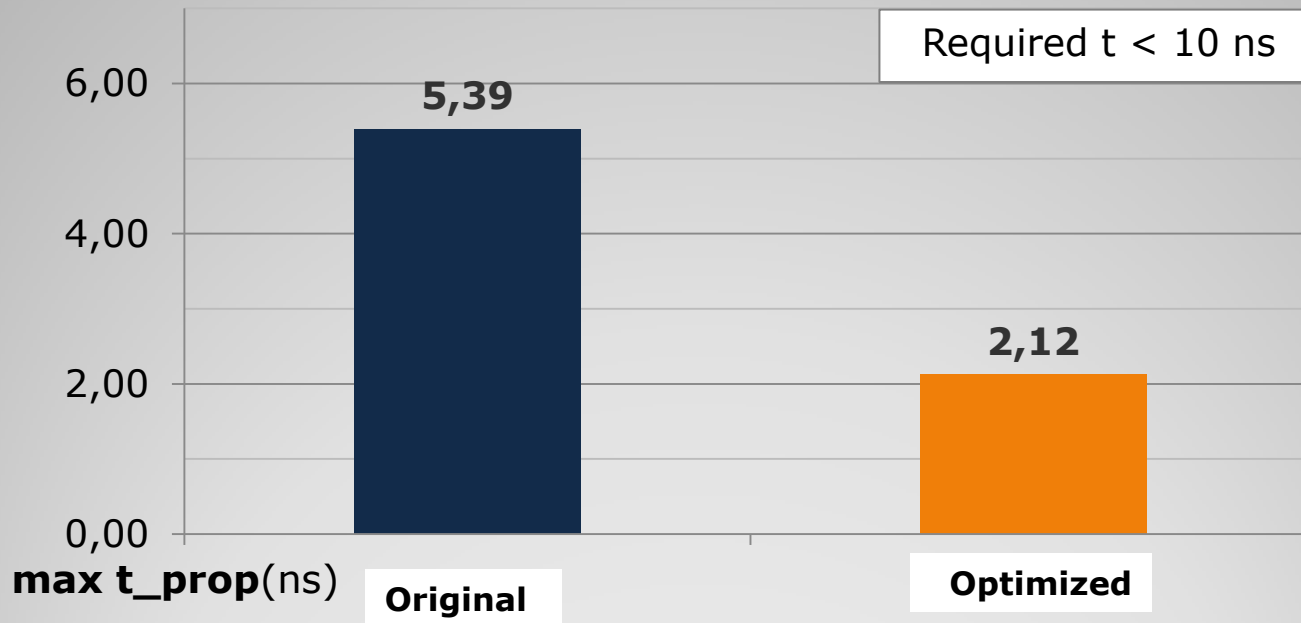
Worse reg. to reg. signal propagation time



Optimizations

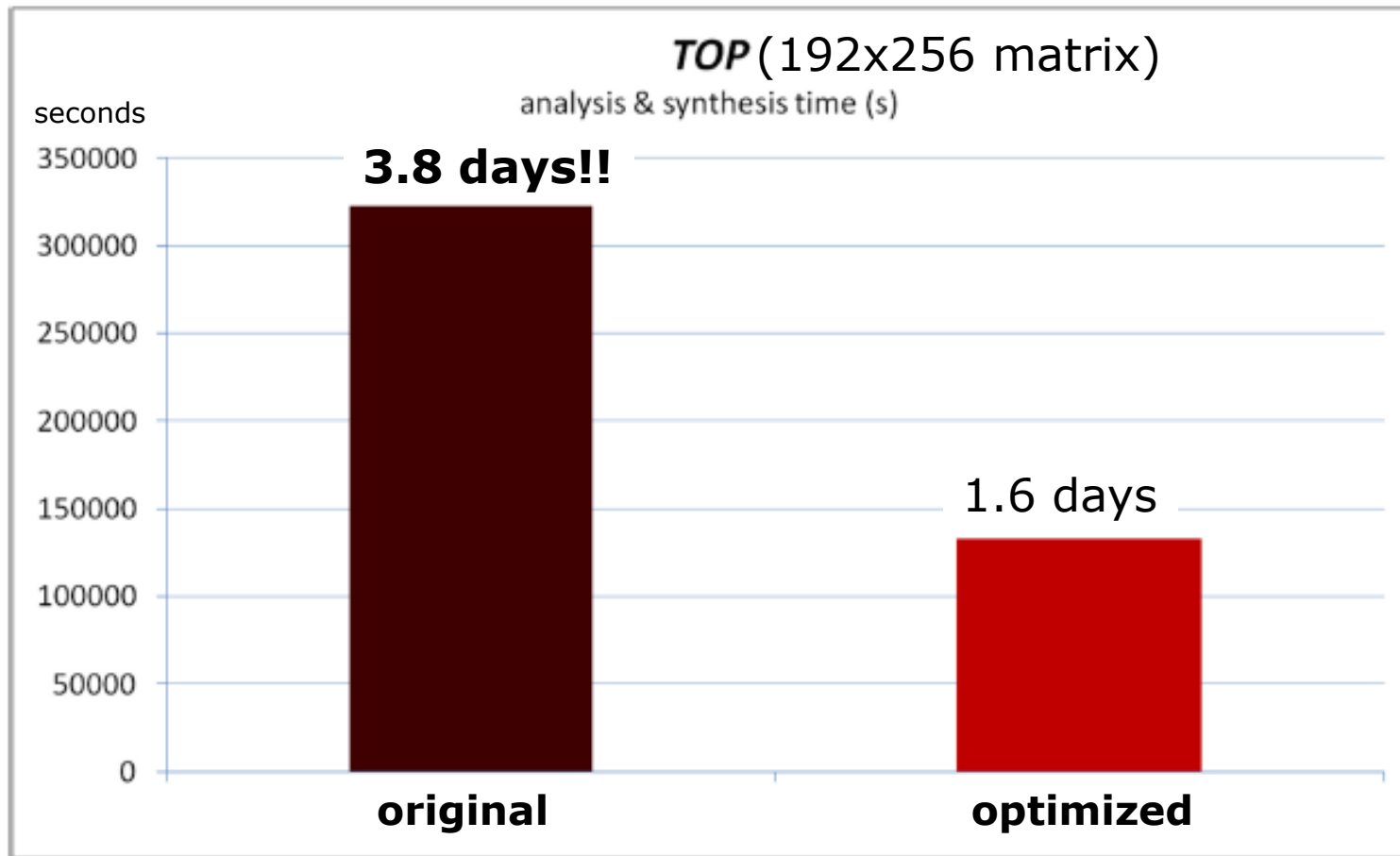
Sweeper speed optimization

Worse reg. to reg. signal propagation time



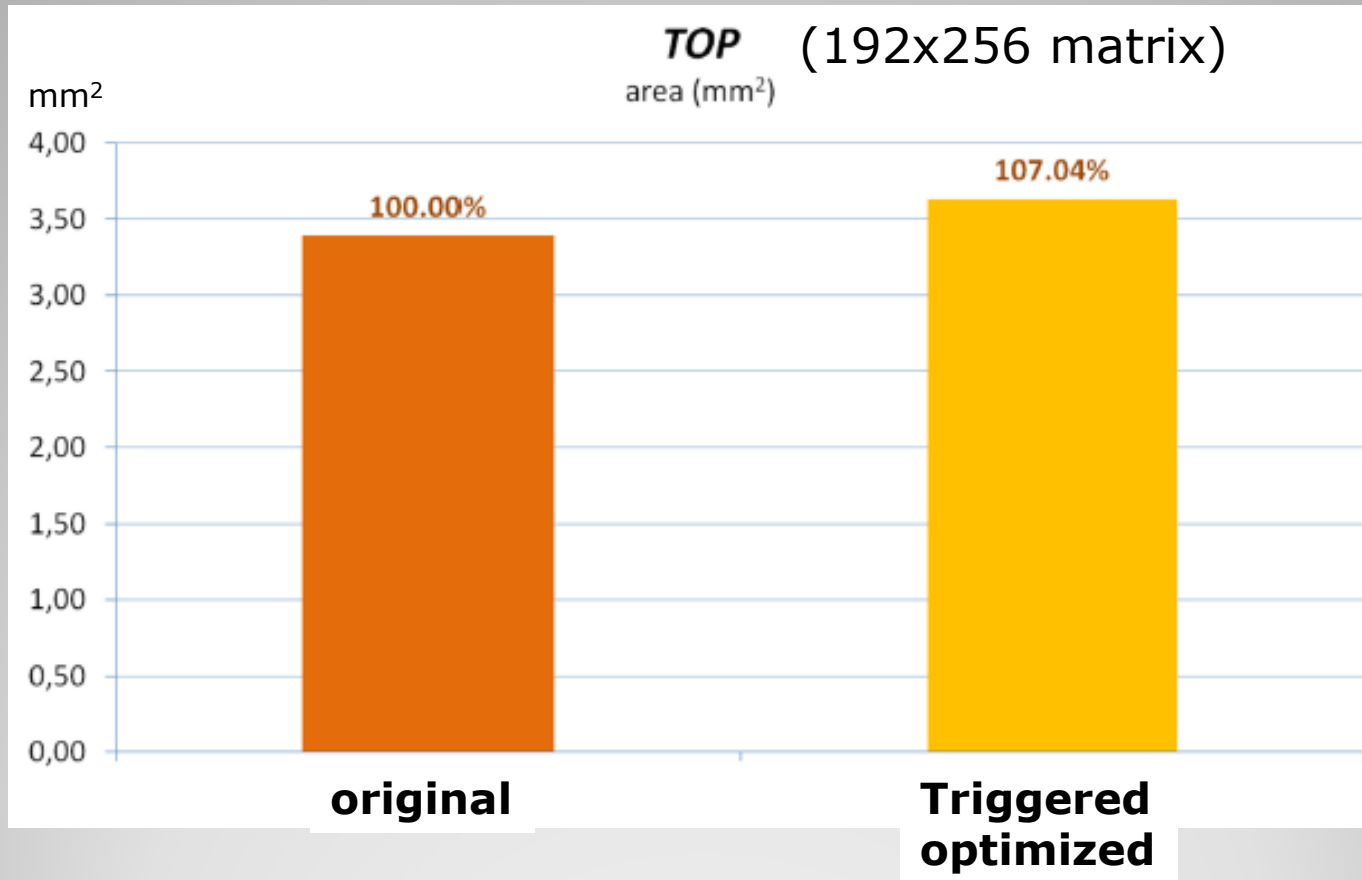
Optimizations

Full chip synthesis time optimization



Optimizations

Full chip cells area



Total cells area comparison

- **SuperPX1** hybrid 3D
 - Matrix 32x128
 - 2 sub-matrices 16x128
 - 4 sparsifiers
 - 8 zones for each sparsifier
 - zone width: 4 pixels

- **APSEL-VI** MAPS 3D
 - Matrix 96x128 (96x96)
 - 2 sub-matrices 48x128 (48x96)
 - 4 (3) sparsifiers
 - 8 zones for each sparsifier
 - zone width: 4 pixels

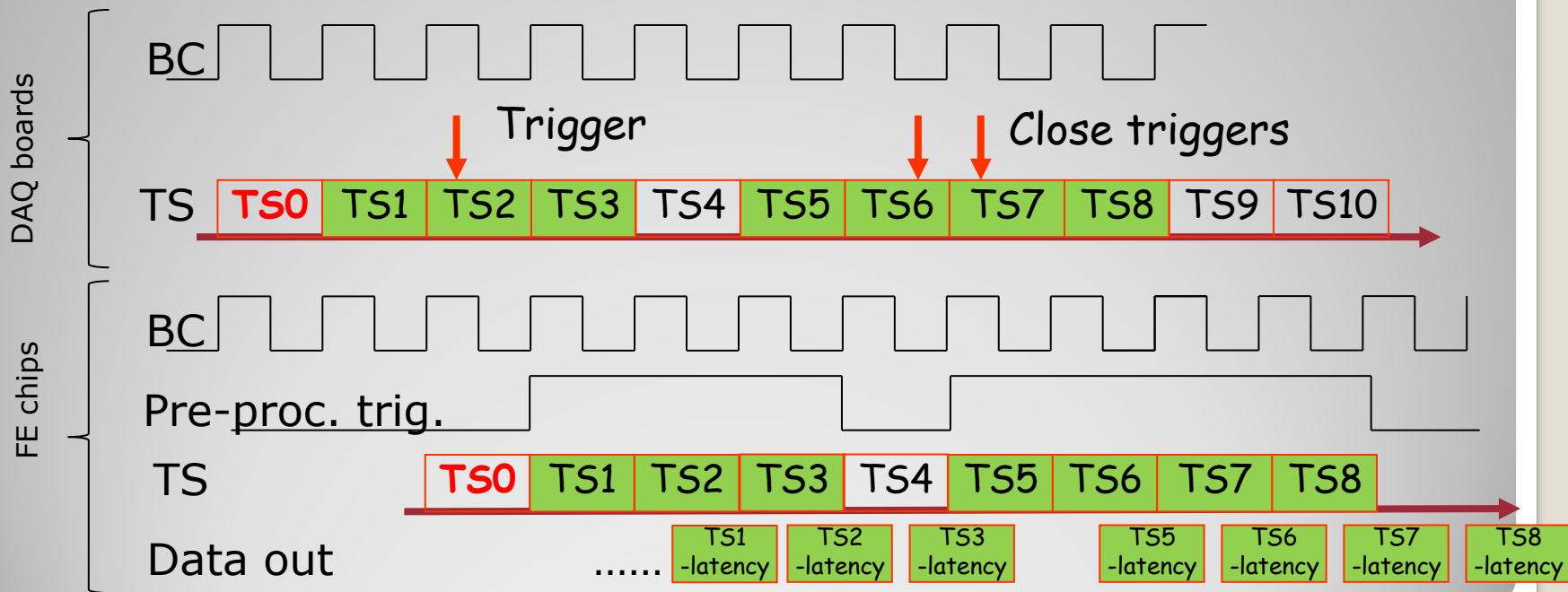
Submissions 2011

- Triggered architecture successfully implemented and simulated.
 - **98.2 %** readout efficiency at **6 us trigger latency**.
 - **BC period** down to **60 ns**.
 - **No BC dependency of efficiency** in the foreseen triggered working conditions.
- Optimizations lead to **faster readout circuits** and **faster synthesis time**.
- Total area after new features and optimization → **only 7% larger**.
- Next step: architecture tailoring for SuperPX1 and APSEL_VI

Conclusions

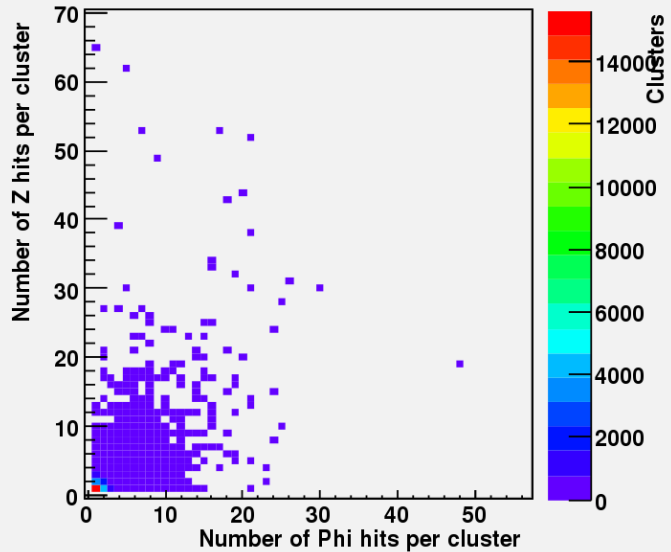
BACKUP

- DAQ boards responsible for trigger handling
- Pre-processed trigger sent to Front-end electronics.
 - Simpler on-chip trigger logic
 - Re-configurable logic on DAQ boards
- One-wire trigger to FE chips.
- Trigger latency configured on FE chips at start-up.
- Chip trigger signal synchronous to BC clock.



Trigger handling

Cluster SVTHit Z/Phi Multiplicity2 on Svt Layer 0



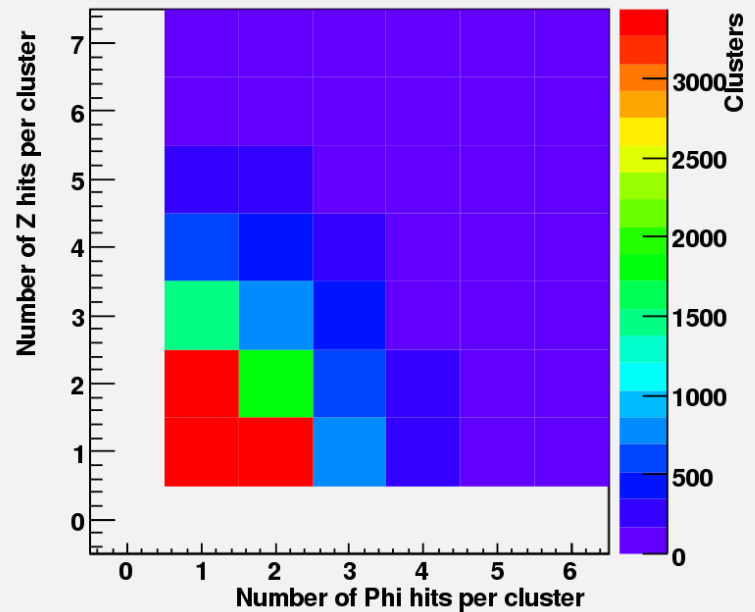
h2_SvtL00_ClusterMulti2_dZ_dPhi		
Entries	32985	
Mean x	1.659	
Mean y	1.796	
RMS x	1.48	
RMS y	1.89	
Integral	3.298e+04	
0	0	0
0	32985	0
0	0	0

(by R.Cenci)

Implemented cluster spread distribution zeta/phi from physics simulations

%	1	2	3	4	phi
1	47,28	12,7	2,5	0,7	...
2	10,6	5,8	1,8	0,6	
3	4,3	2,3	1,1	0,4	
4	2,0	1,2	0,6	0,3	
zeta					

Cluster SVTHit Z/Phi Multiplicity2 on Svt Layer 0

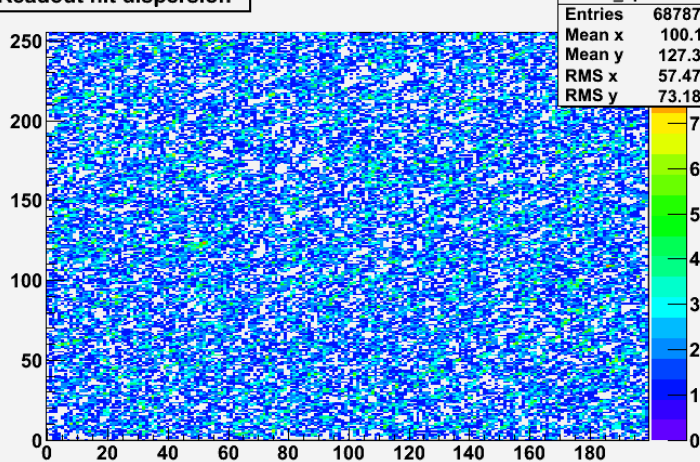


h2_SvtL00_ClusterMulti2_dZ_dPhi		
Entries	32985	
Mean x	1.507	
Mean y	1.635	
RMS x	0.883	
RMS y	1.093	
Integral	3.221e+04	
0	249	172
0	32208	356
0	0	0

Zoom/Equalize

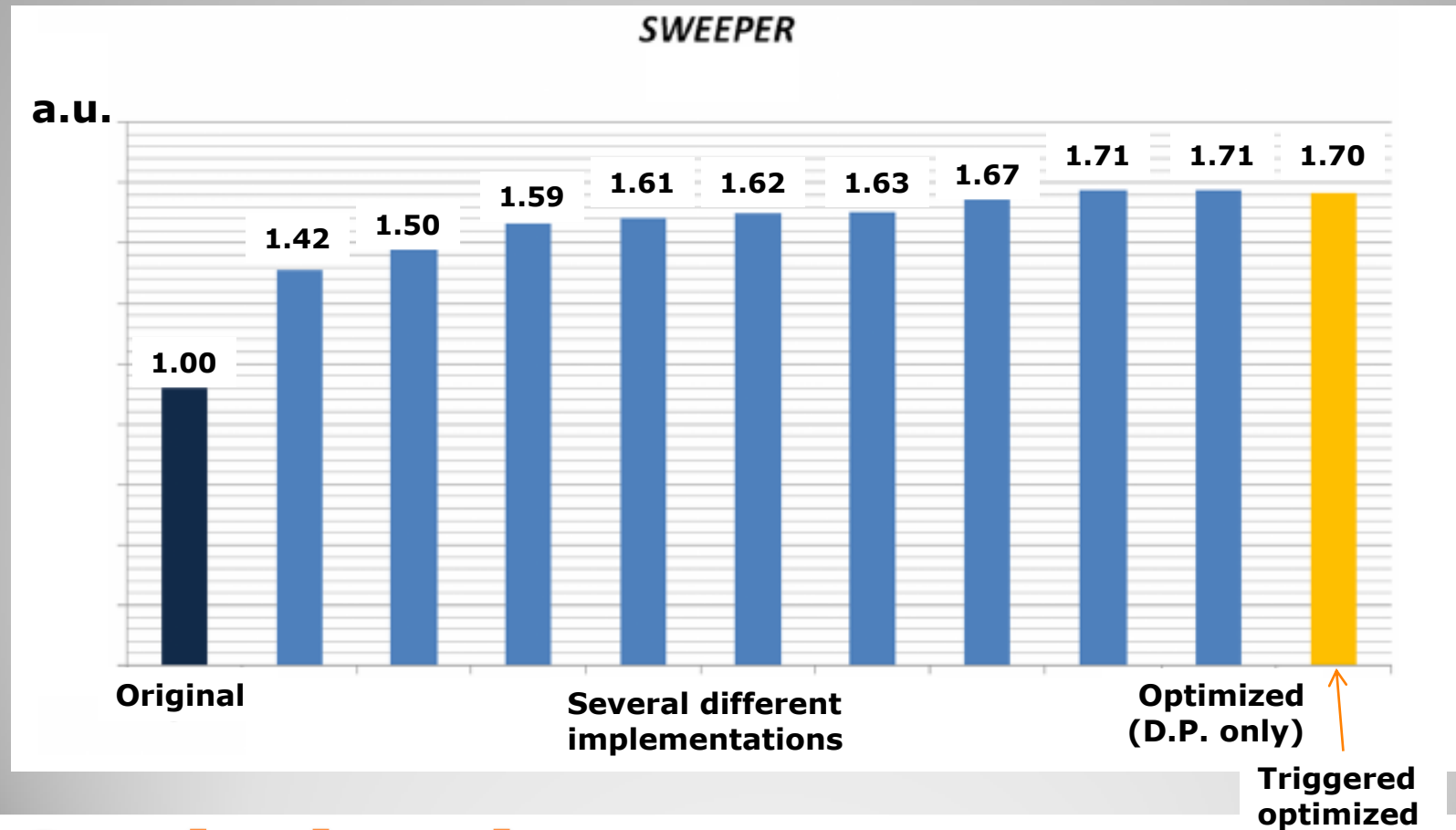
Simulated clustered hit distrib.

Readout hit dispersion



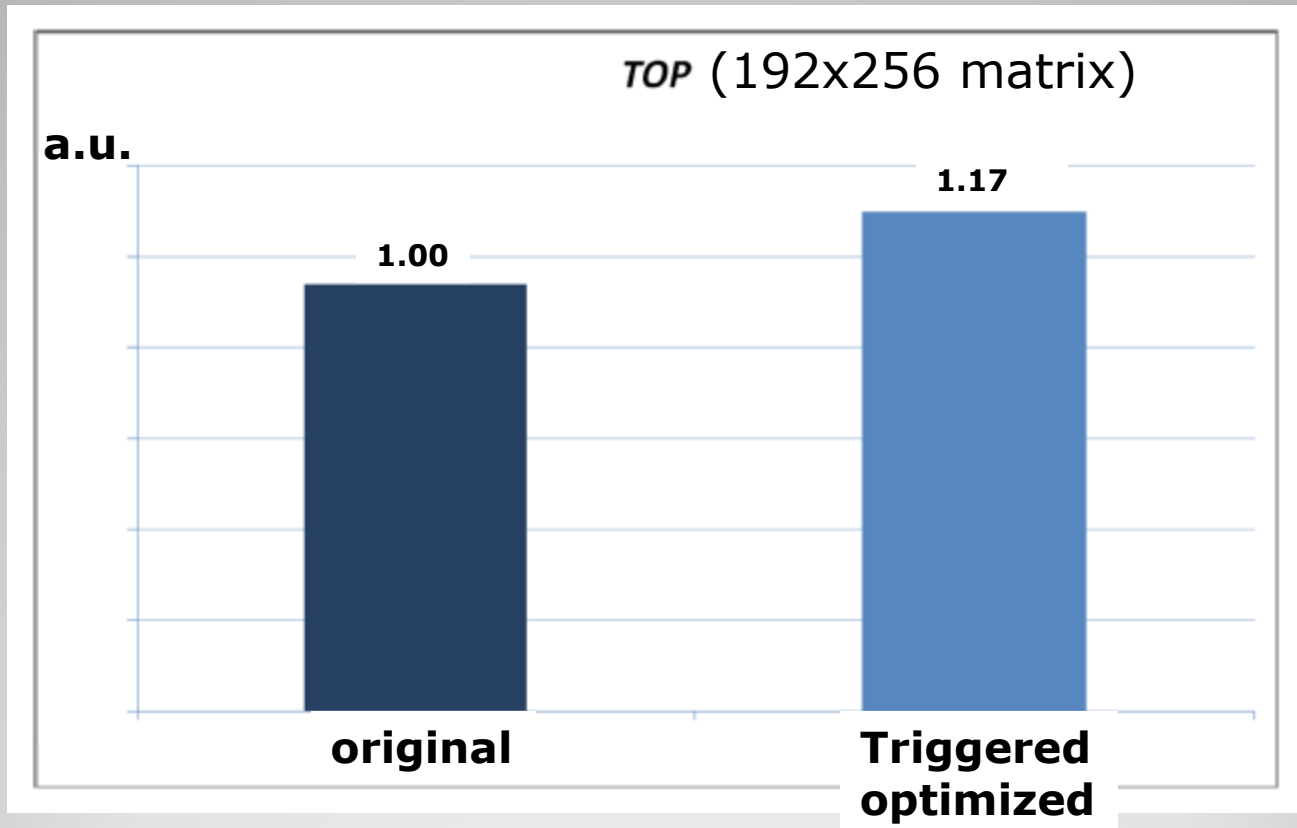
out_spread	
Entries	68787
Mean x	100.1
Mean y	127.3
RMS x	57.47
RMS y	73.18

Sweeper speed optimization



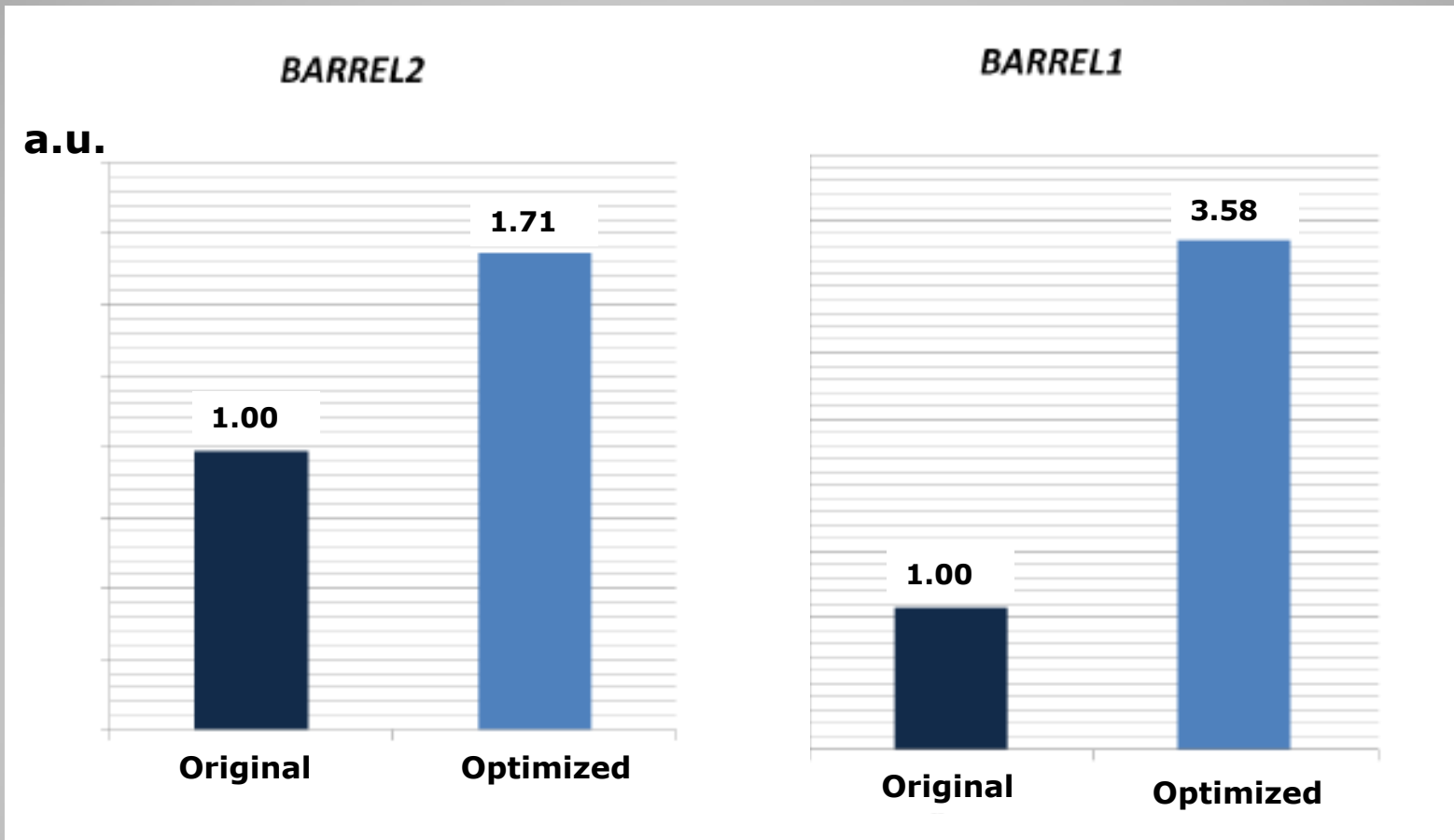
Optimizations

Full chip speed optimization



Optimizations

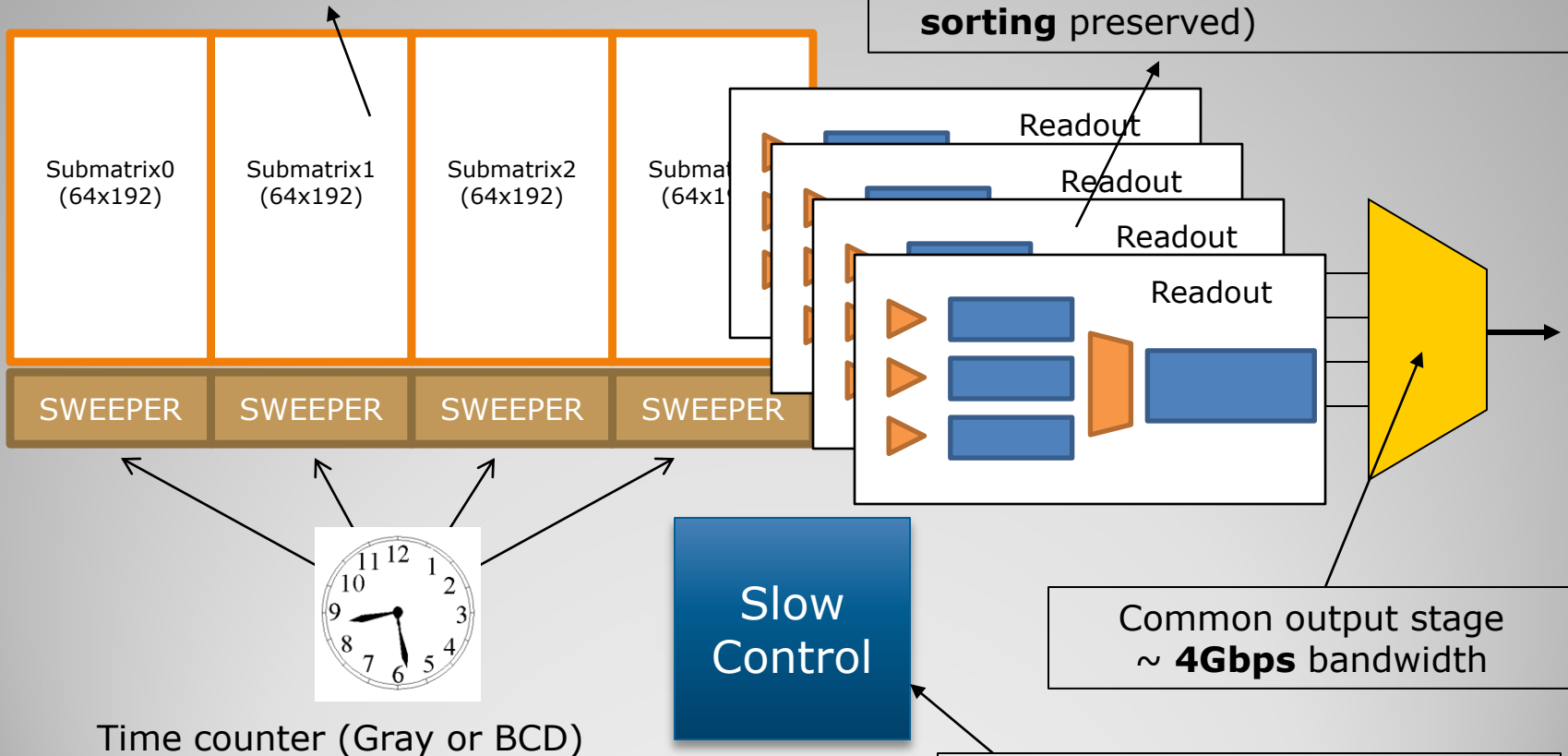
Barrels speed optimization



Optimizations

- Pixel organized into **4 sub-matrices**
- **Each** sub-matrix has an **independent** scan logic
- Increase horizontal parallelization
- The shorter the scans the greater the effi.

- One readout for **each** sub-matrix
- Vertical parallel sparsification (**one entire column** per clock cycle)
- Hit encoding
- Hit de-queuing system (**time sorting** preserved)



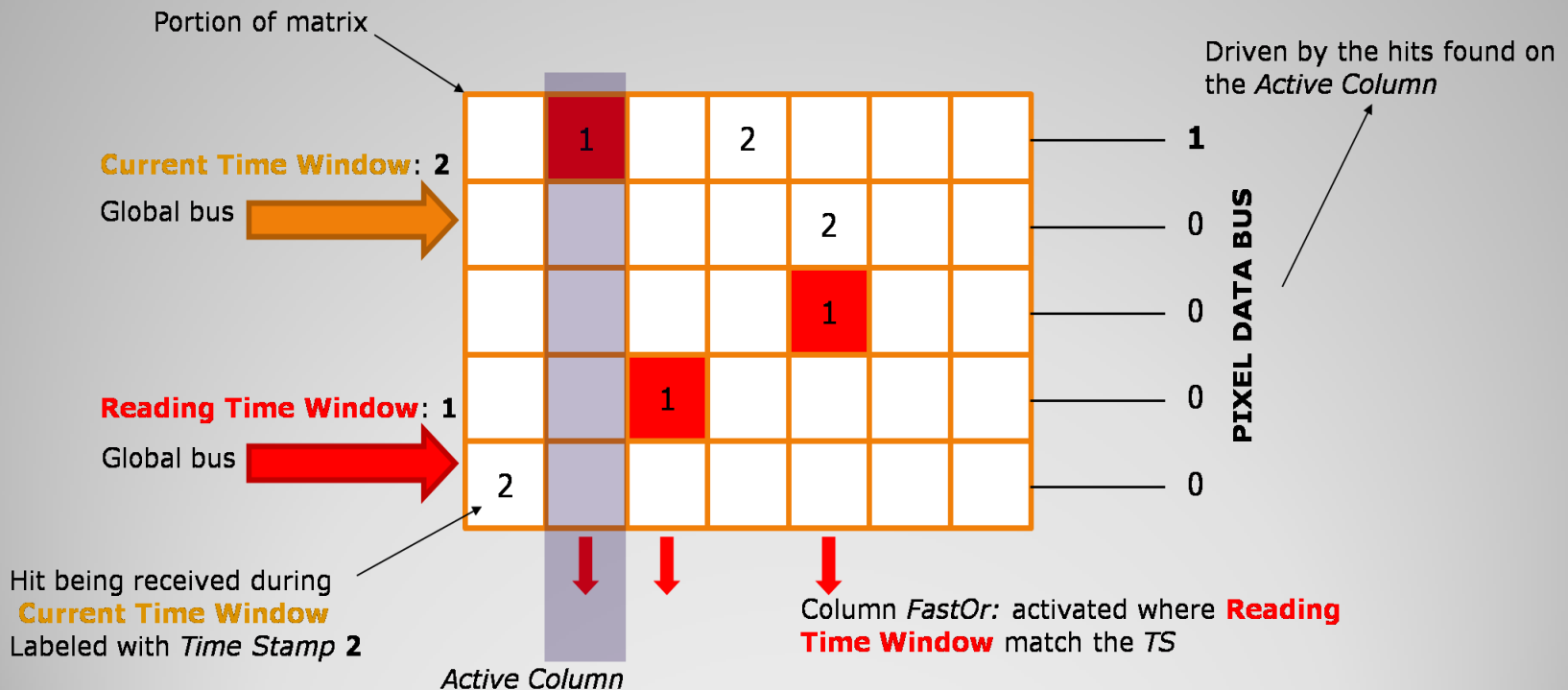
Readout Scheme

- I²C-like external interface:
- 2 pad per chip
 - 2 lines for entire module

EXAMPLE

During **Time Window 2** :

- Some pixels getting fired and labeled with *Time Stamp (TS) = 2*
- The readout queries the columns containing hits labeled with $TS=1$ (**Reading Time Window** → *FastOr* activation)
- The readout moves the *Active Column* over the columns with an active *FastOr*.



Matrix scan Logic example