CT trigger experience from TERZINA

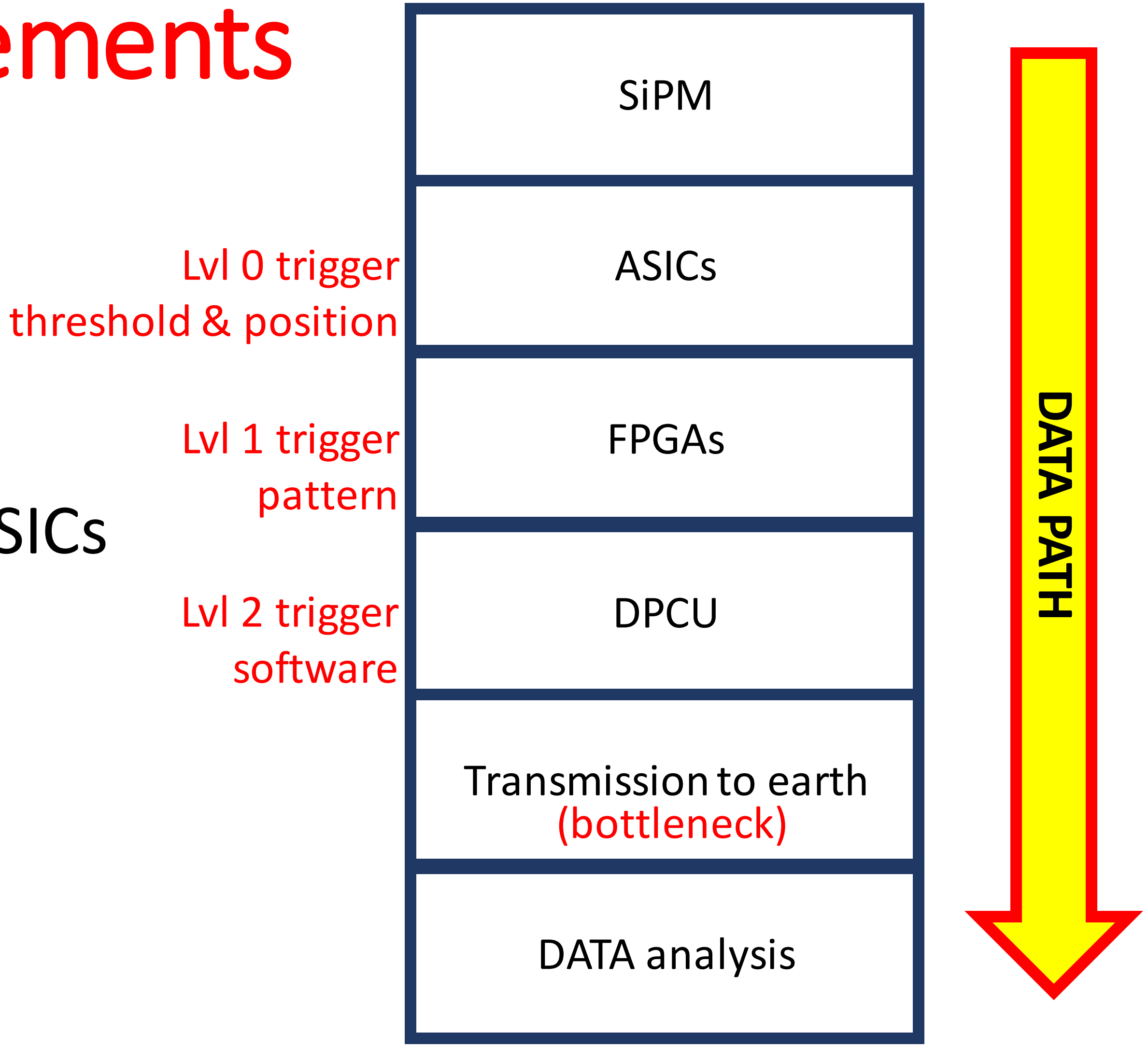
Stefan Cristi Zugravel
stefan.zugravel@to.infn.it

2023/12/20, Rome

Istituto Nazionale di Fisica Nucleare

TERZINA trigger requirements

- Efficient rejection of background
- Minimal dead time
- High configurability
- Trigger coming from two adjacent ASICs



SiPM

Lvl 0 trigger
threshold & position

ASICs

Lvl 1 trigger
pattern

FPGAs

Lvl 2 trigger
software

DPCU

Transmission to earth
(bottleneck)

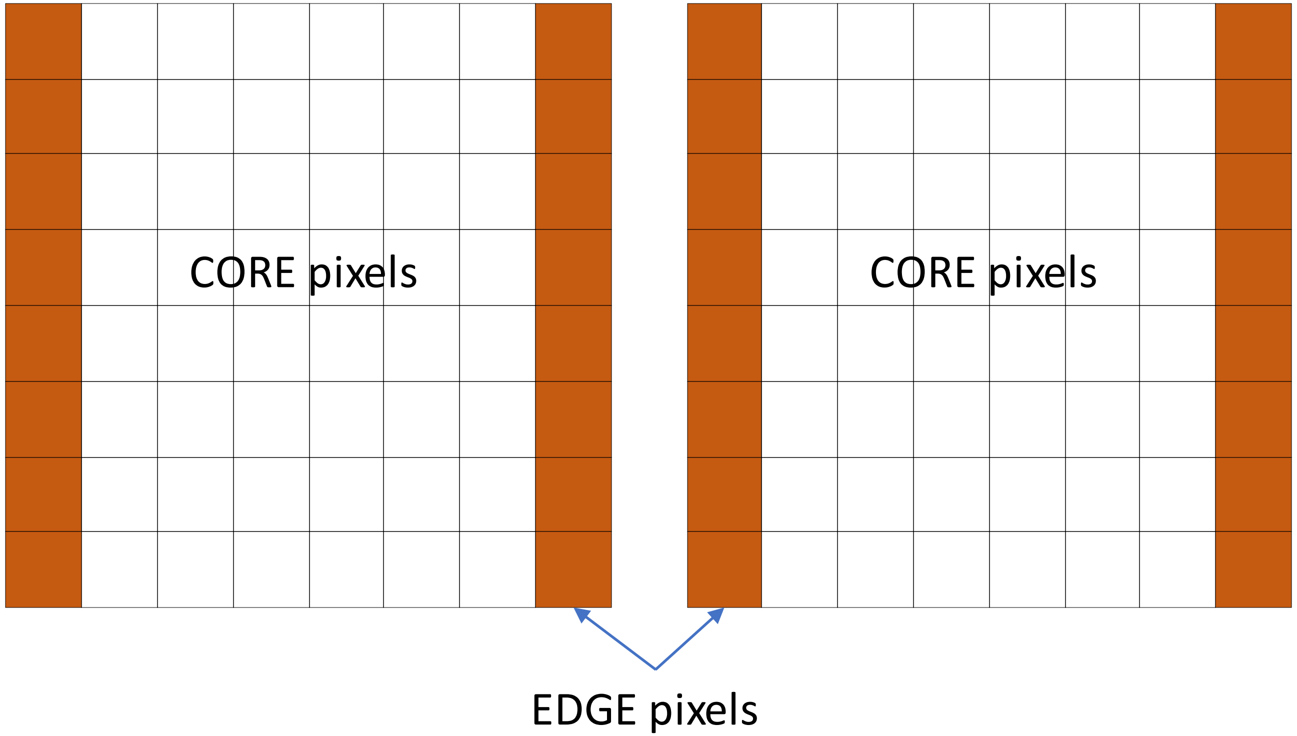
DATA analysis

DATA PATH

3 level trigger system

Level 0 trigger -> ASIC (threshold and hitmap)

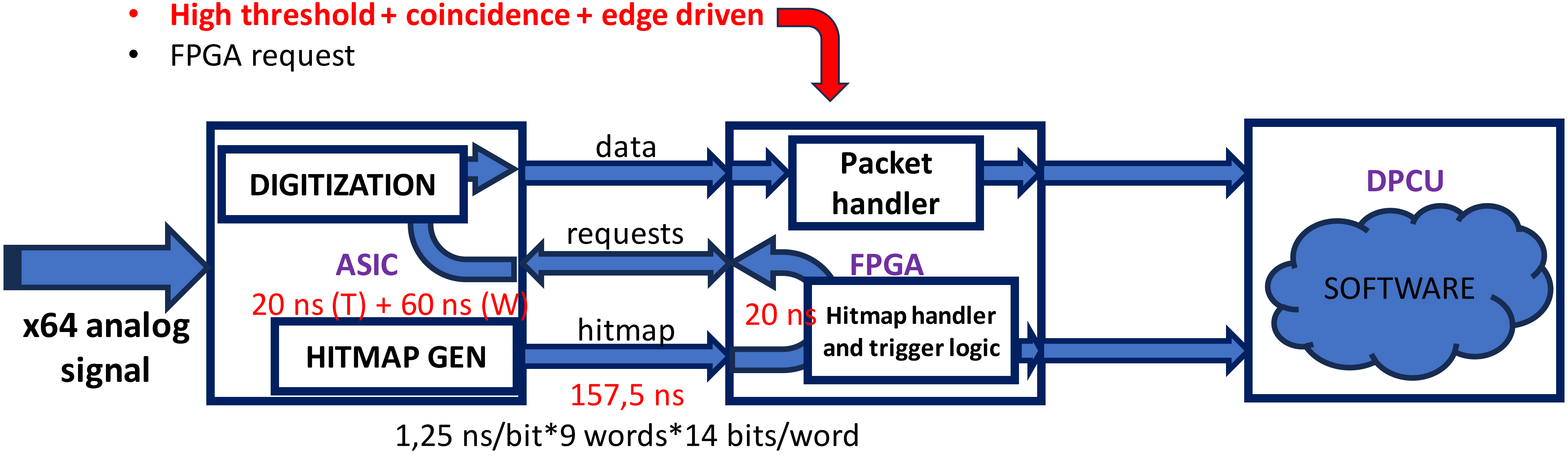
- Each ASIC can readout a 64 pixel 8*8 matrix
- Each pixel has two configurable thresholds (namely high and low).
- Any pixel can be configured as EDGE pixel, in picture the TERZINA configuration.



Hitmap generation and timing

The ASIC has three possible operating modes:

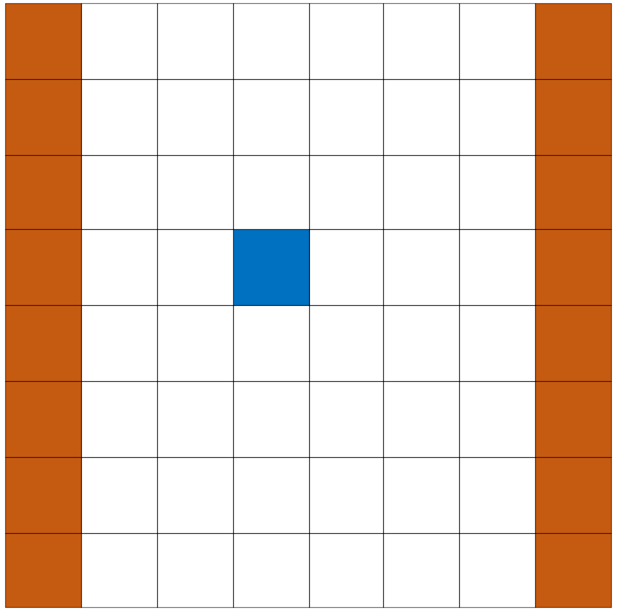
- Programmable time window
- **High threshold + coincidence + edge driven**
- FPGA request



- Each hitmap remains stored in the ASIC for T + 60 ns with T programmable (~20 ns)
- The ASIC waits 80 ns for the FPGA response (in the 32 cells configuration)
- Hitmap dead time ~20+60+157,5+20 ns -> ~260 ns from the first pixel over threshold to the confirmation to take data

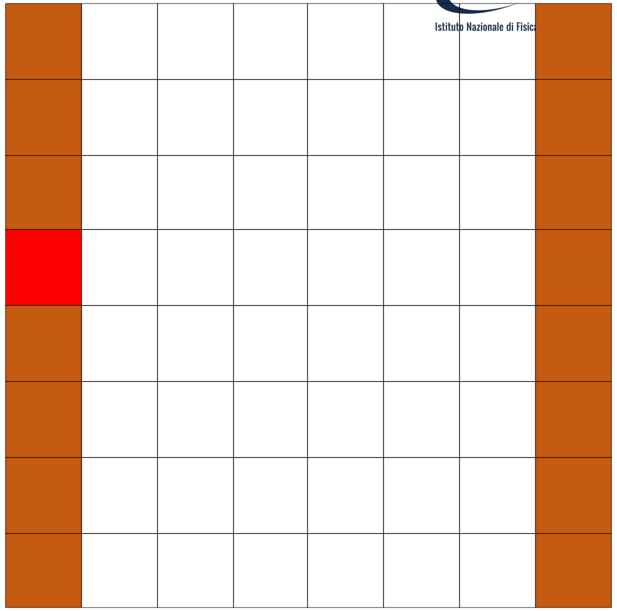
T+W ≤ (cells/2)*5 ns

Level 0 scenarios

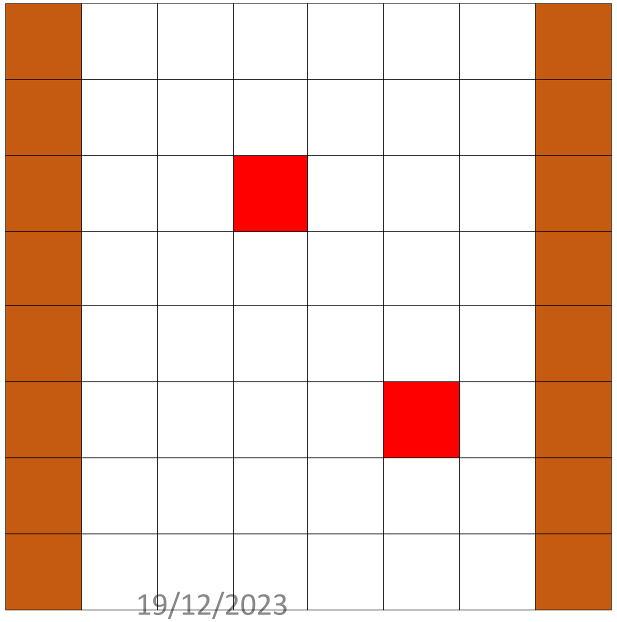


CASE 0:
HIGH threshold verified on a pixel (EDGE OR CORE).
Hitmap generation.

CASE 2:
LOW threshold verified on an EDGE pixel.
Hitmap generation.

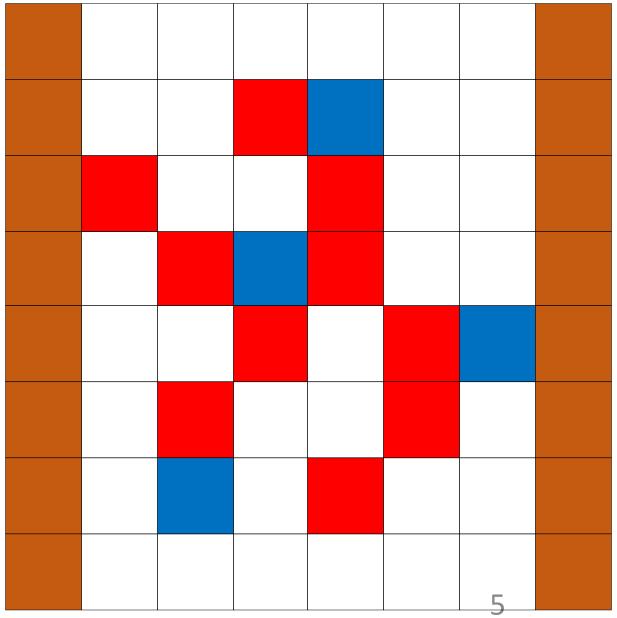


Programmable integration time T (~20 ns).



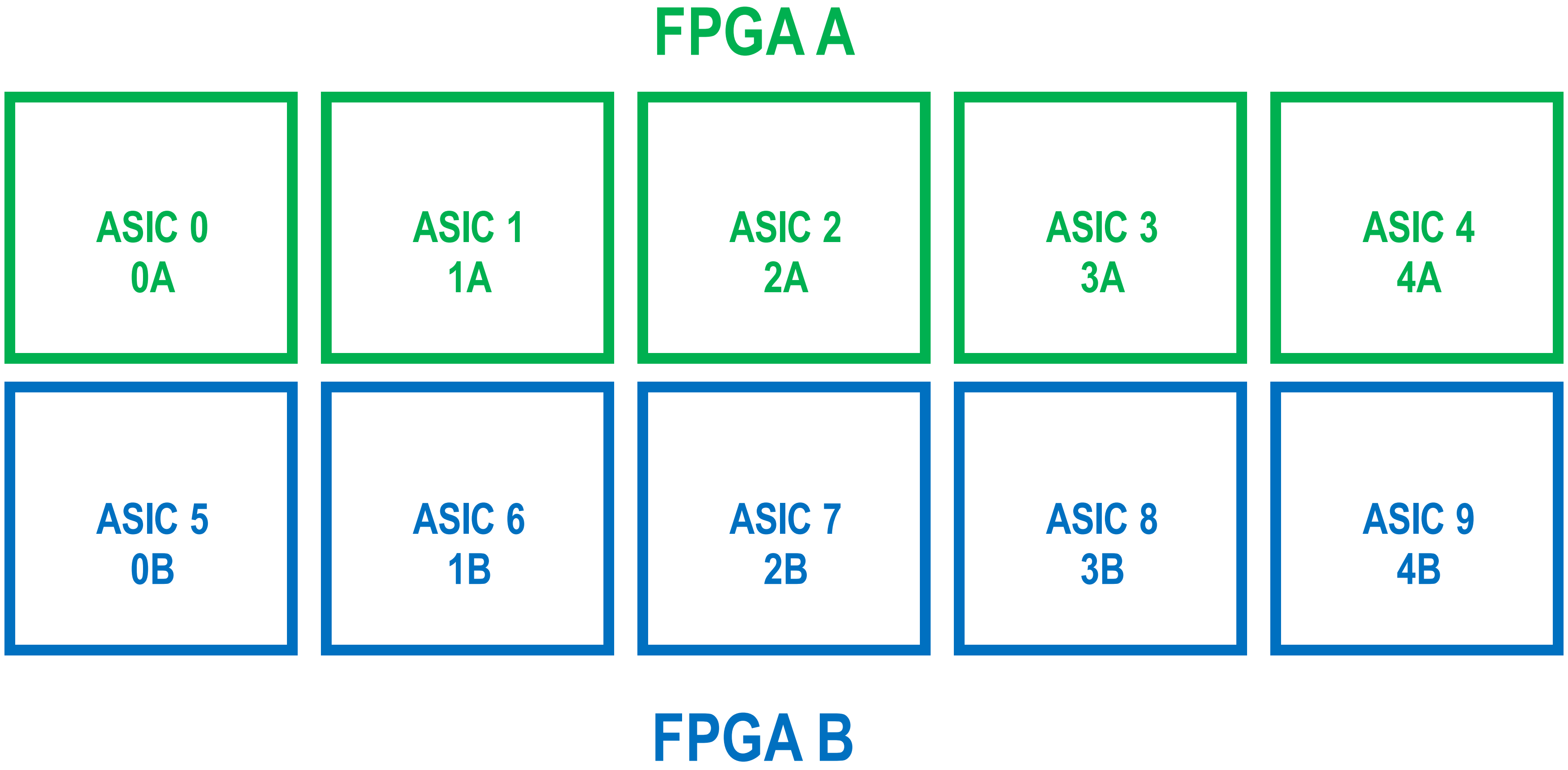
CASE 1:
LOW threshold verified on two or more pixels.
Hitmap generation.

CASE 3:
Light pollution.
Hitmap generation.
No data digitization.



Level 1 trigger -> FPGA

The hitmaps from 5 ASICs are collected by a redout FPGA which will identify the interesting patterns and will request the digitized data from the one or more ASICs (depending on the configuration).



FPGA A

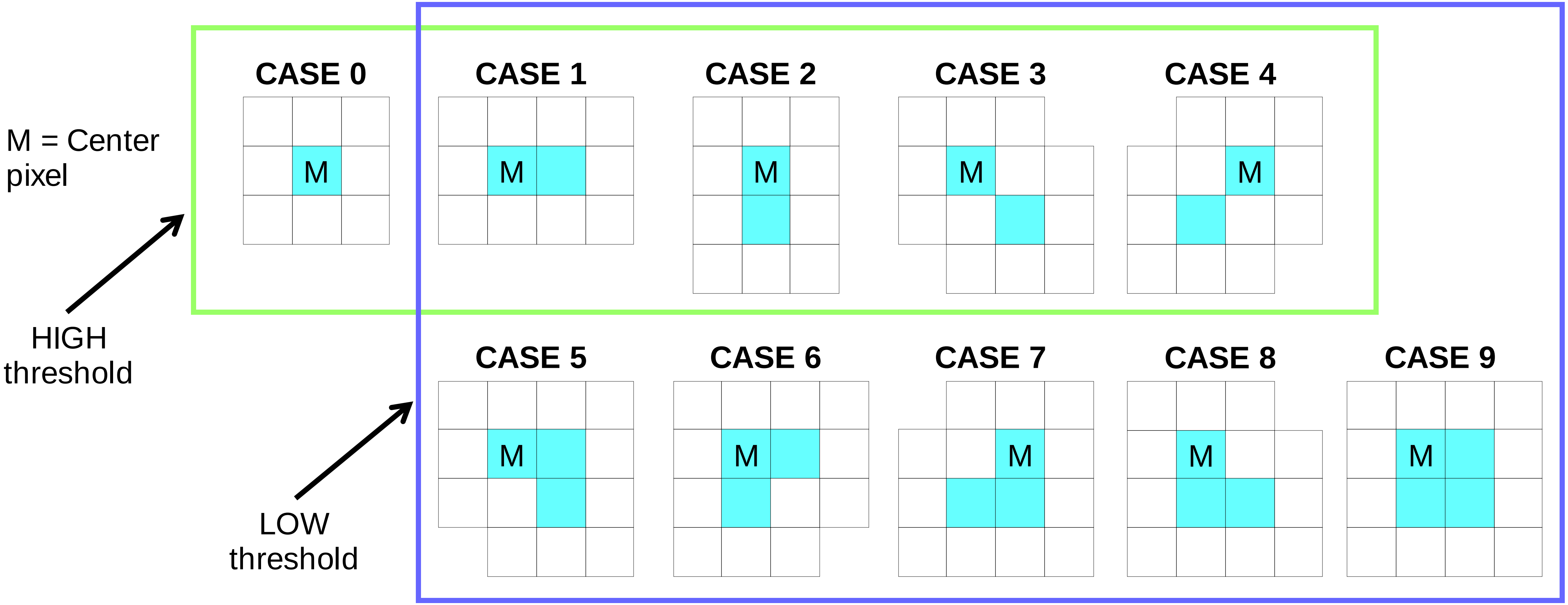
ASIC 0
0A | ASIC 1
1A | ASIC 2
2A | ASIC 3
3A | ASIC 4
4A

ASIC 0 0A	ASIC 1 1A	ASIC 2 2A	ASIC 3 3A	ASIC 4 4A
ASIC 5 0B	ASIC 6 1B	ASIC 7 2B	ASIC 8 3B	ASIC 9 4B

FPGA B

Hitmap pre-defined patterns

Each case can be enabled or disabled via a configuration parameter. More patterns could be added.



M = Center pixel

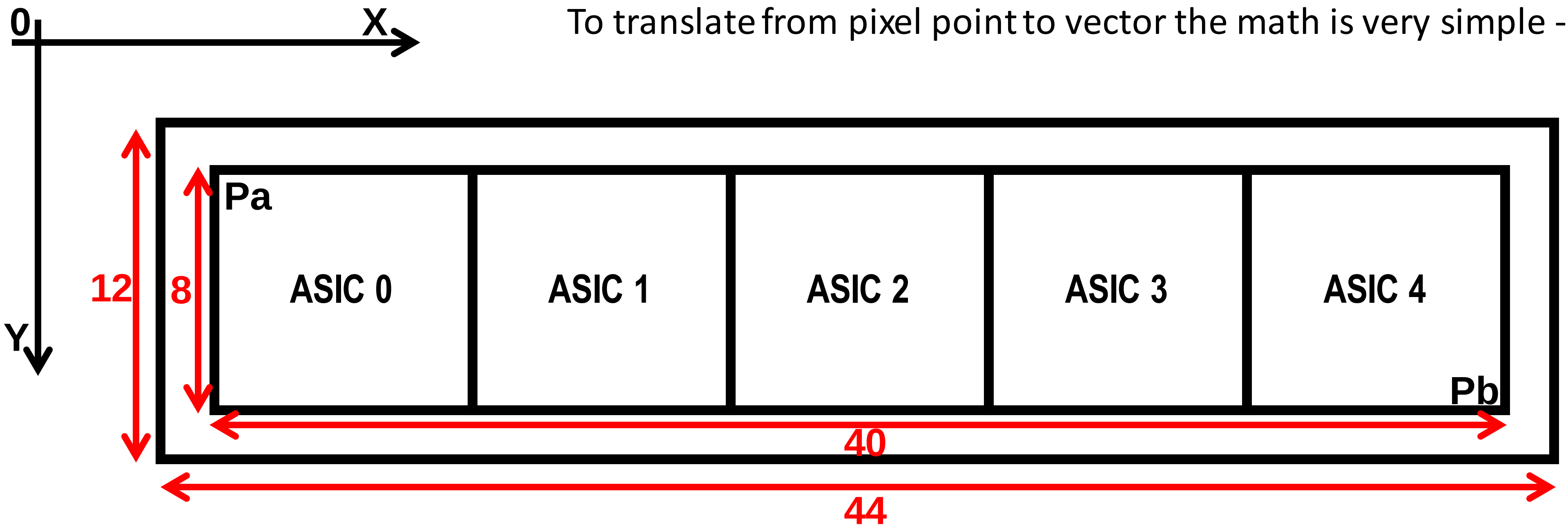
HIGH threshold

LOW threshold

Hitmap validator algorithm

After receiving the hitmap packet from each ASIC, the relevant 64 bit are extracted and combined into a single vector. We can visualize this as a map with the 5 chips placed as in the camera and with a 2-pixel wide padding around.

To translate from pixel point to vector the math is very simple -> P = X + 44Y



From Pa to Pb each pixel has an independent module that checks for pre-defined patterns.

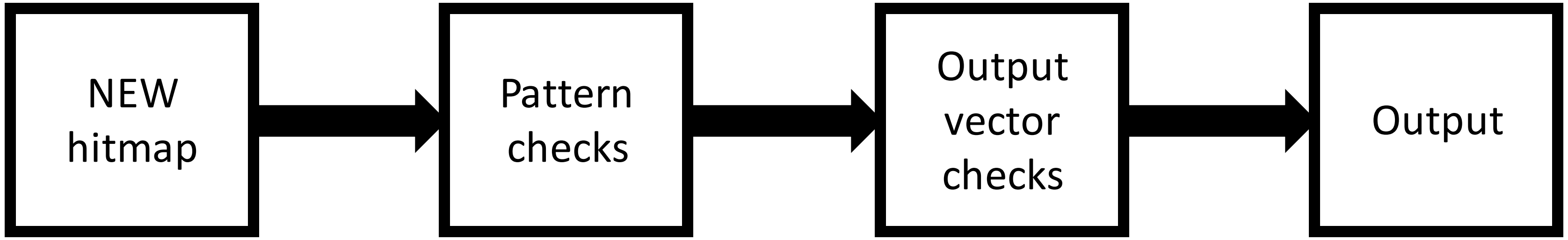
P = X + 44*Y
Pa = 2 + 44*2
Pb = 41 + 44*9

(527 downto 00) → 528 bit | 320 bit hitmap | 208 bit padding

(437 downto 90) → 348 bit Main

Hitmap validator algorithm output

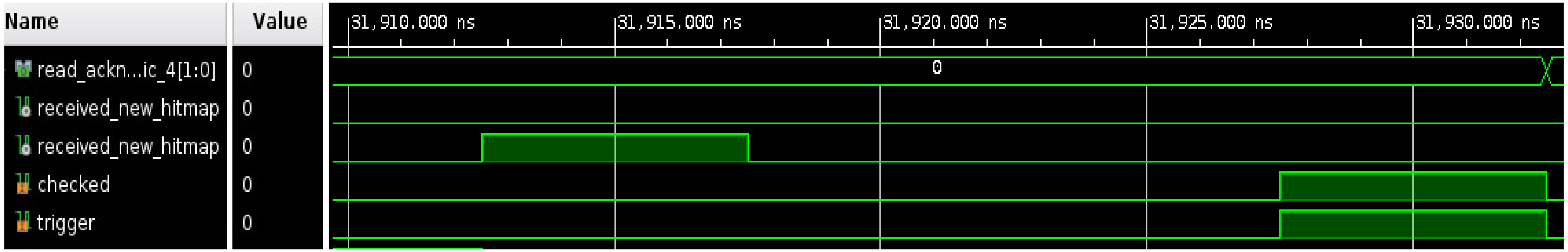
The output from each pixel pattern module gets inserted into an output vector. If the output vector has a bit not zero then the trigger is fired.


NEW hitmap

Pattern checks

Output vector checks

Output



Name	Value
read_ackn...ic_4[1:0]	0
received_new_hitmap	0
received_new_hitmap	0
checked	0
trigger	0

Low dead time, but relatively high FF utilization

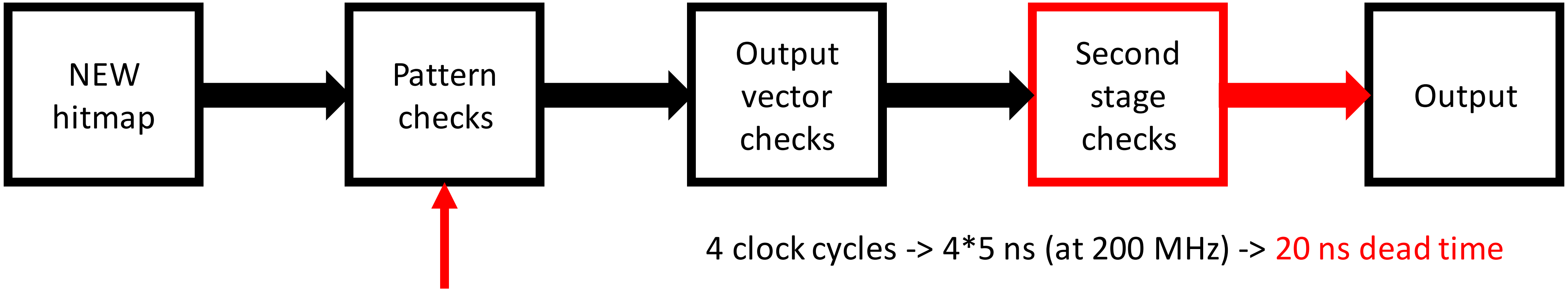
3 clock cycles -> 3*5 ns (at 200 MHz) -> 15 ns dead time In the FPGA, much less than the transmission time of a hitmap packet

Level 3 trigger -> DPCU

The hitmaps and the data packets will be al sent to the DPCU where a software algorithm will re-check all the events before sending them to earth.

Solutions for SPB3

- The same trigger algorithm with the addition of a second patter seeking stage could be used for a bi-focal system.
- The second stage would be looking for a “pattern of patterns”.

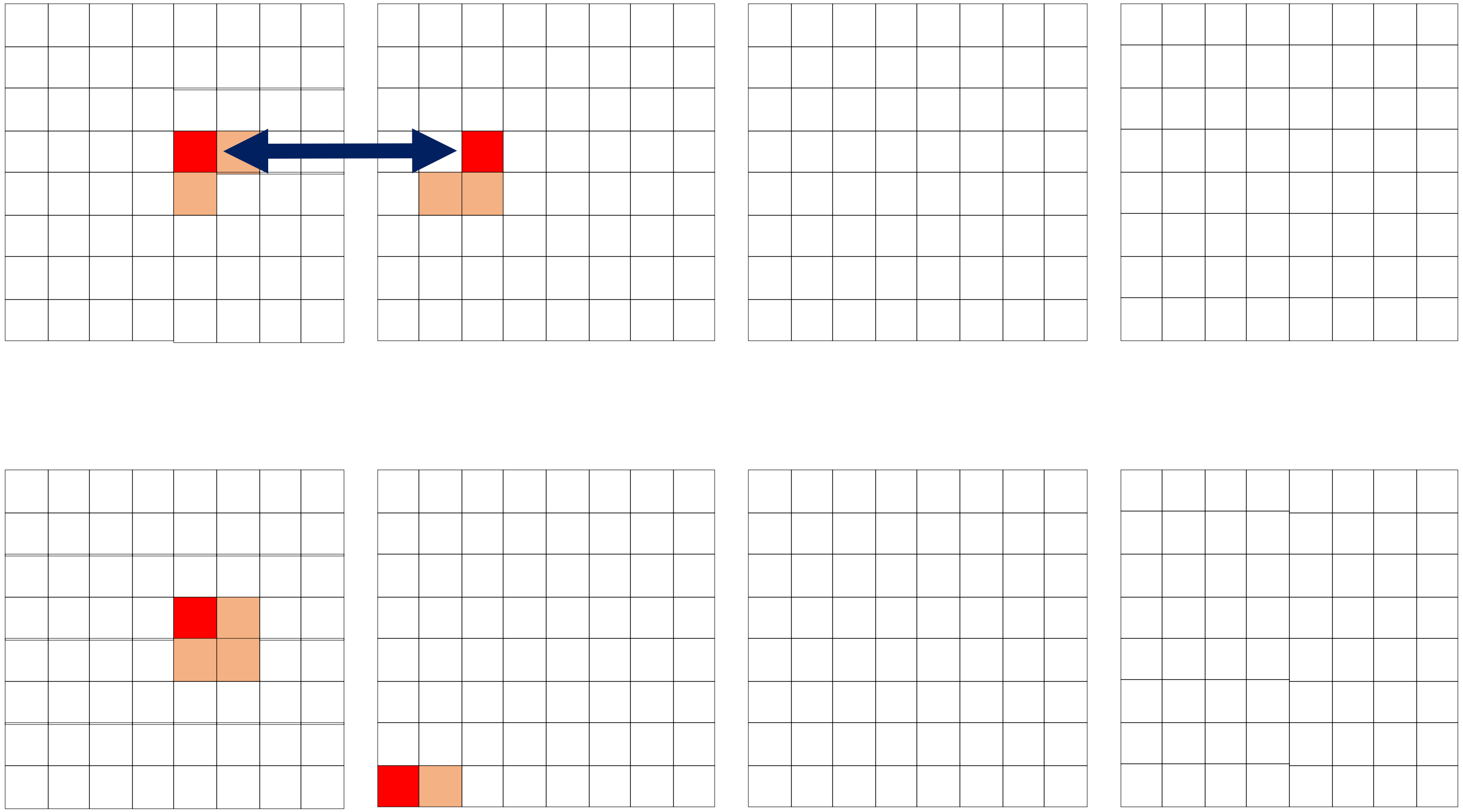


4 clock cycles -> 4*5 ns (at 200 MHz) -> 20 ns dead time

Modification required to account for the additional ASICs -> higher FPGA utilization

A two level hitmap

The output vector from the first stage could be considered as another hitmap where each bit at 1 corresponds to an interesting pattern.



YES

NO