Simulation updates
What’s new

• Matrix dimensions update:
  – 200x256 pixels (50x256 sub-matrices)
  – 50 um pitch
What’s new

• Matrix dimensions update:
  – 200x256 pixels (50x256 sub-matrices)
  – 50 um pitch

• Code optimization
  – Substantial: more robust concentrators
  – Formal: code maintenance
What’s new

• Matrix dimensions update:
  – 200x256 pixels (50x256 sub-matrices)
  – 50 um pitch
• Code optimization
  – Substantial: more robust concentrators
  – Formal: code maintenance
• Monte Carlo generator
  – Rewritten from scratch
  – Now generation of clustered events
  – Knobs:
    • Cluster dispersion distribution in (z,phi)
    • Physical time resolution (test_clock), hit/miss thresholds =Global hit rate
What’s new

- Matrix dimensions update:
  - 200x256 pixels (50x256 sub-matrices)
  - 50 um pitch
- Code optimization
  - Substantial: more robust concentrators
  - Formal: code maintenance
- Monte Carlo generator
  - Rewritten from scratch
  - Now generation of clustered events
  - Knobs:
    - Cluster dispersion distribution in (z,phi)
    - Physical time resolution (test_clock), hit/miss thresholds =Global hit rate
  - Arbitrary pattern extraction by geometrical rules
What’s new

- Matrix dimensions update:
  - 200x256 pixels (50x256 sub-matrices)
  - 50 um pitch
- Code optimization
  - Substantial: more robust concentrators
  - Formal: code maintenance
- Monte Carlo generator
  - Rewritten from scratch
  - Now generation of clustered events
  - Knobs:
    - Cluster dispersion distribution in (z,phi)
    - Physical time resolution (test_clock), hit/miss thresholds =Global hit rate
  - Arbitrary pattern extraction by geometrical rules

Cluster spread extraction
Es. (5,5)
What’s new

• Matrix dimensions update:
  – 200x256 pixels (50x256 sub-matrices)
  – 50 µm pitch

• Code optimization
  – Substantial: more robust concentrators
  – Formal: code maintenance

• Monte Carlo generator
  – Rewritten from scratch
  – Now generation of clustered events
  – Knobs:
    • Cluster dispersion distribution in (z, phi)
    • Physical time resolution (test_clock), hit/miss thresholds = Global hit rate
  – Arbitrary pattern extraction by geometrical rules
What’s new

- Matrix dimensions update:
  - 200x256 pixels (50x256 sub-matrices)
  - 50 um pitch

- Code optimization
  - Substantial: more robust concentrators
  - Formal: code maintenance

- Monte Carlo generator
  - Rewritten from scratch
  - Now generation of clustered events
  - Knobs:
    - Cluster dispersion distribution in (z,phi)
    - Physical time resolution (test_clock), hit/miss thresholds = Global hit rate
  - Arbitrary pattern extraction by geometrical rules
  - Pattern check (grey pixels are forbidden)
What's new

• Matrix dimensions update:
  – 200x256 pixels (50x256 sub-matrices)
  – 50 um pitch

• Code optimization
  – Substantial: more robust concentrators
  – Formal: code maintenance

• Monte Carlo generator
  – Rewritten from scratch
  – Now generation of clustered events
  – Knobs:
    • Cluster dispersion distribution in (z,phi)
    • Physical time resolution (test_clock), hit/miss thresholds = Global hit rate
  – Arbitrary pattern extraction by geometrical rules
  – Pattern check (grey pixels are forbidden)
  – Pattern application to a random pixel of the matrix
Matrix interface

- Converging with Fabio and Giulia on a viable matrix interconnection solution
  - TSREQ bus x4 sub-matrices
  - MASK_WRITEb for general pixel reset
  - MaskColSel_b for TSREQ dependent pixel reset
Simulation Diagram

Organization of the VHDL test bench architecture

Simulation pads
Physical pads
Simulation signals
Physical signals

---Matrix Interface (*)
ColReadEna_b : std_logic_vector(MAT_X_SIZE-1 downto 0);
LatchEna_b : std_logic_vector(MAT_X_SIZE-1 downto 0);
Mask_Write_b : std_logic_vector(MAT_Y_SIZE-1 downto 0);
MaskColSel_b : std_logic_vector(MAT_X_SIZE-1 downto 0);
TSCNT : std_logic_vector(TS_WIDTH-1 downto 0);
TSREQ : std_logic_vector(N_XSUBMAT*TS_WIDTH-1 downto 0);
FastOr : std_logic_vector(MAT_X_SIZE-1 downto 0);
PIX_DATA_M0 : std_logic_vector(SUBMAT_Y_SIZE-1 downto 0);
PIX_DATA_M1 : std_logic_vector(SUBMAT_Y_SIZE-1 downto 0);
PIX_DATA_M2 : std_logic_vector(SUBMAT_Y_SIZE-1 downto 0);
PIX_DATA_M3 : std_logic_vector(SUBMAT_Y_SIZE-1 downto 0);

VHDL Test Bench Architecture

Equivalent Chip

Monte Carlo Generator

Files I/O

Files I/O

Simulation pads
Physical pads
Simulation signals
Physical signals

Digital Chip I/O
-RDclk, Fast_clk, BC_clk
-reset
-data_out(19:0) + data_valid
-MLena + Global_FastOR
-SDA, SCL, chip_addr(2:0)

Stimuli
-Monitors
-Checks
-Eff, Eval

Hit_counter, Already_hit_counter

reset, test_clock, seeds, gen_rate

 CMP(5199:0)

PXL_LATCH(5199:0)

Matrix Interface*
Now SQUARE architecture “ON AIR”:
Preliminary simulations with a matrix model made in Bo • for test bench structure trial. • ~100MHz/cm²

Try-outs (3000 us, 350k/400k hits):

Cluster spread in zeta/phi

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Cluster spread in zeta/phi

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50%</td>
<td>12.5%</td>
<td>4%</td>
<td>1%</td>
</tr>
<tr>
<td>2</td>
<td>12.5%</td>
<td>4%</td>
<td>1%</td>
<td>0%</td>
</tr>
<tr>
<td>3</td>
<td>4%</td>
<td>1%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>4</td>
<td>1%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

Generated hit distribution

About optimization: 350k hits $\rightarrow$ ~350k words

400k hits $\rightarrow$ ~300k words
Rule of thumb area estimations for the final front-end chip
Synthesized a SQUARE readout architecture fitted for a 200 x 256 matrix (50x256 sub-matrices)

Synthesis:
- **HCMOS9GP** library (ST 130 nm like APSEL4D, 4D_1, FE32x128)
- **240k** logic cells
- **3.5 mm²** only of std-cells area.
- **60 hours** of 100% processor time.

<table>
<thead>
<tr>
<th></th>
<th>Cells</th>
<th>Readout Area</th>
<th><strong>density</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>APSEL4D</td>
<td>100k</td>
<td>6 mm²</td>
<td>17k cell/mm²</td>
</tr>
<tr>
<td>FE4D32x128</td>
<td>60k</td>
<td>2.5 mm²</td>
<td>20k cell/mm²</td>
</tr>
<tr>
<td>APSEL3D_TC</td>
<td>12k</td>
<td>1.3 mm²</td>
<td>9.2k cell/mm²</td>
</tr>
<tr>
<td><strong>FINAL (ST130)</strong></td>
<td><strong>240k</strong></td>
<td><strong>(est) ~ 14 mm²</strong></td>
<td>(lowest ST) <strong>17k</strong> cell/mm²</td>
</tr>
<tr>
<td><strong>TC subm.(100x160)</strong></td>
<td>(est. 31%FINAL)~ 75k</td>
<td>(est) <strong>8.2 mm²</strong></td>
<td>(TC) <strong>9.2k</strong> cell/mm²</td>
</tr>
</tbody>
</table>

A layout without matrix interconnections geometry can be misleading.
The estimations above represent the most probable BUT NOT SURE values (especially for **TC process** that is also changing design kit).
Readout for ApselVI_1D

- MATRIX 100x160 (2 sub-m. 50 x160)
- Column divided in:
  - 5 sparsifiers
  - 32 rows for each sparsifier
  - 8 zones for each sparsifier ($W_{\text{zone}} = 4$ pixels)
Readout for Superpix1

- MATRIX 32x128 (2 sub-m. 16x128)
- Column divided in:
  - 4 sparsifiers
  - 32 rows for each sparsifier
  - 8 zones for each sparsifier ($W_{\text{zone}} = 4$ pixels)