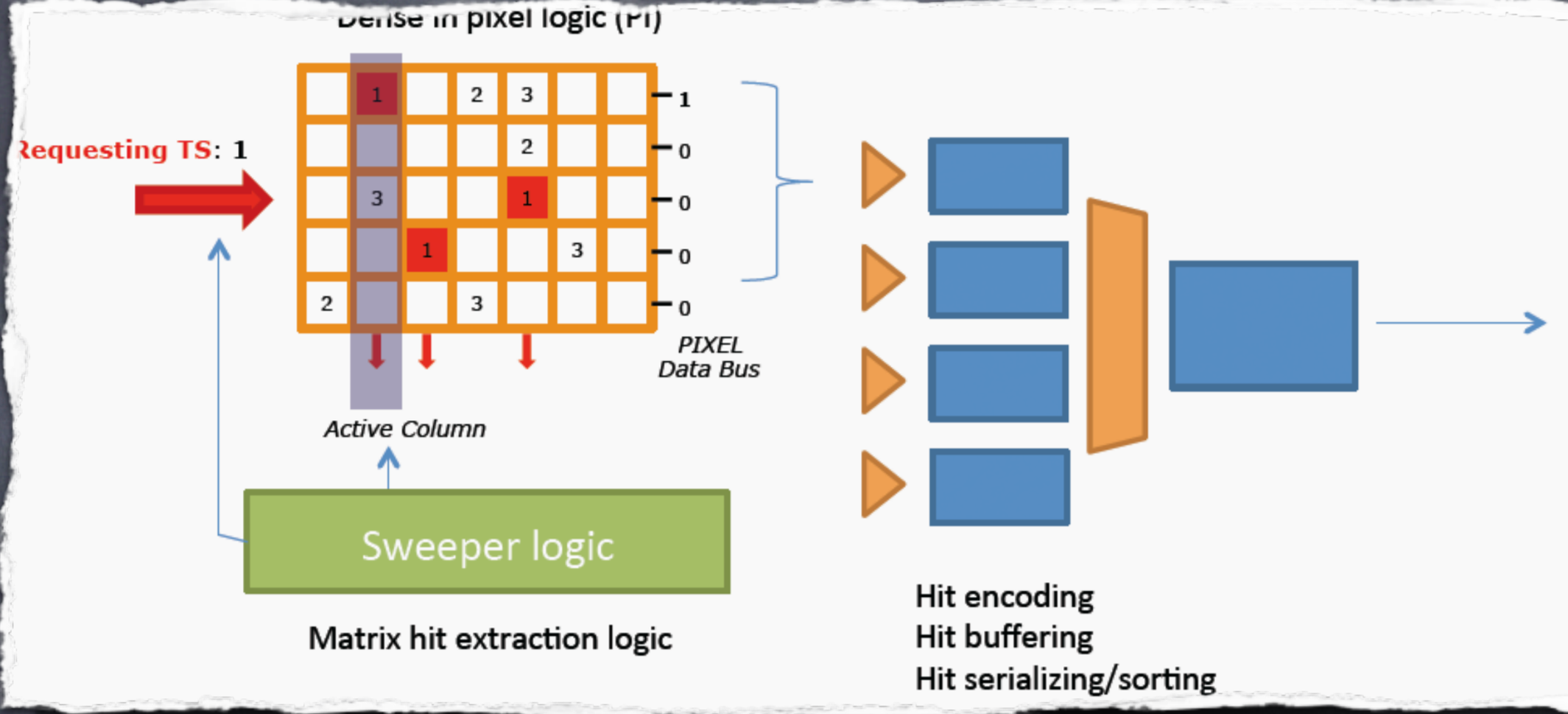
Update on in-strip logic design

Description of actual code

Roberto Beccherle, INFN - Pisa, Email: Roberto.Beccherle@pi.infn.it

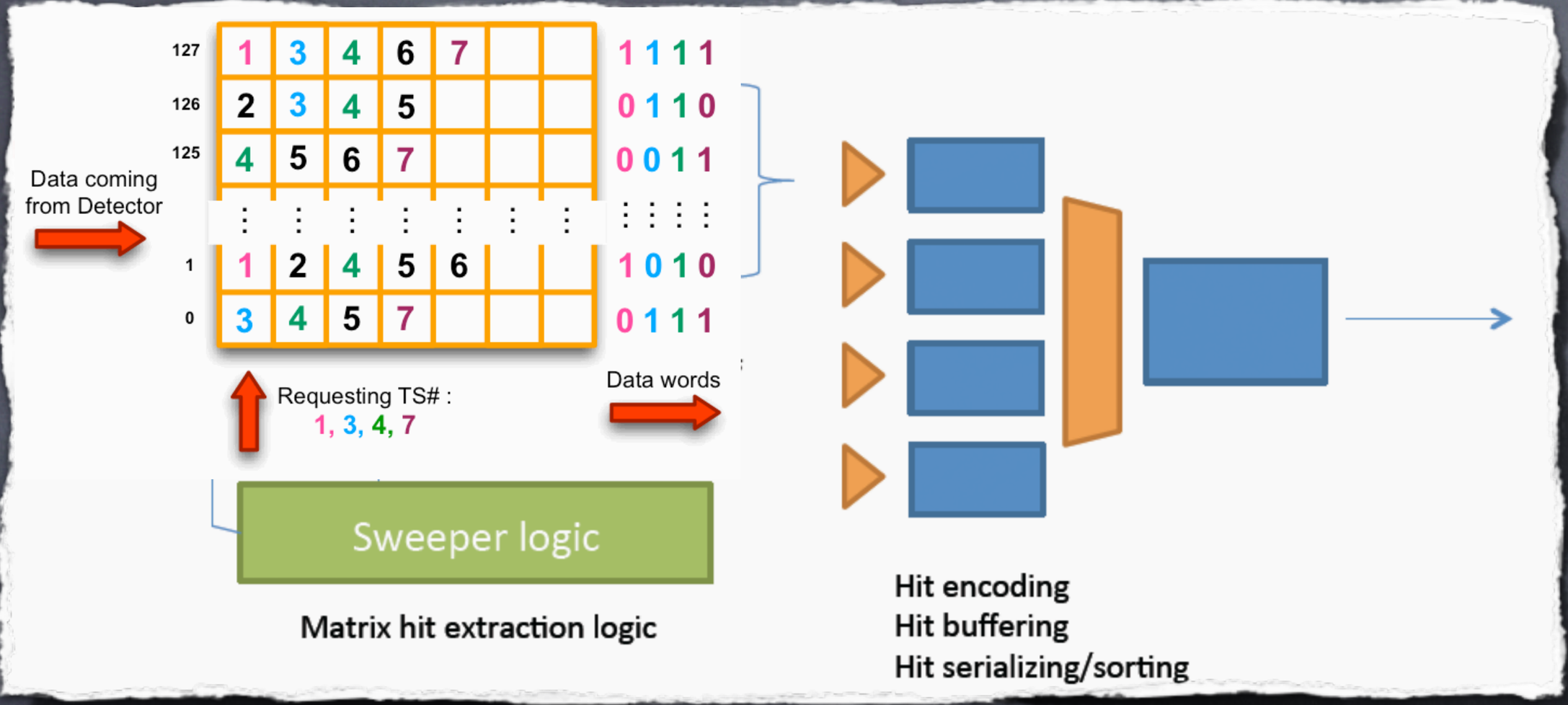
Present pixel readout



Design is strongly based on presently available blocks.

Architecture suited for reading out a Pixel matrix.

Proposed strip readout



Design is strongly based on presently available blocks.

Architecture suited for reading out a Pixel matrix.

The idea is to replace the "Pixel matrix" with a Strip matrix.

the size is 128 * MEM_SIZE buffers, subdivided in four submatrices, as before.

The sweeper logic requests a particular TimeStamp (TS)

The Matrix presents all Hits with that specific TS and clears unread data.

Strips module interface

There is only one module that interfaces with:

Readout Block on one side and with the Detector on the other side

```
module Strips(
// Outputs
output reg [(TOT_SIZE*N_STRIP)-1:0]  StripData,// Data to be read (1024 bits)
output reg [N_SUBMAT-1:0]            FastOr,   // Active if one submatrix has data
// Inputs
input wire [N_STRIP-1:0]     HIT,              // Inputs coming from the Strip Detector
                             GlobalReset_b,    // Allows to reset single strip circuitry
                             StripEnable,      // If true corresponding strip is enabled
input wire [TS_SIZE-1:0]     TSCnt,            // Time Stamp Counter
input wire [(N_SUBMAT*TS_SIZE)-1:0] TSReq,     // Time Stamp Request.
                                               // There is one for each submatrix
input wire [N_SUBMAT-1:0]    ColClearData_b,   // Reads data if TS matches
                             ColReadData_b,    // Clears data if TS matches
input wire                   RDclk,            // System clock
                             BCclk);           // BC clock
```

HIT signal is connected to the MC generator that provides the hits

FastOr initiates the readout process and after a TSReq, StripData is set.

Two clock signals (RDclk and BCclk) are used within this module.

Strips module architecture

The main module is divided in two blocks: StripAnalog and StripDigital

```
StripAnalog Analog( // Inputs
                    .StrIn(HIT),                // N_STRIP wide signal
                    .BCK(BCK),.CK(CK),          // Clocks used by the design
                    // Outputs
                    .StripData(IntStripData)    // N_STRIP wide signal
                    );
StripDigital  Digital( // Inputs
                    .HitData(IntStripData),                 // N_STRIP    wide signal
                    .ColClearData_b(ColClearData_b),    // N_SUBMAT   wide signal
                    .ColReadData_b(ColReadData_b),      // N_SUBMAT   wide signal
                    .TSCnt(TSCnt),                      // TS_SIZE    wide signal
                    .TSReq(TSReq),                      // (N_SUBMAT*TS_SIZE) wide signal
                    //
                    .CK(CK),.BCK(BCK),                  // Clocks used by the design
                    .GlobalReset_b(GlobalReset_b),      // N_STRIP    wide signal
                    // Outputs
                    .StripData(StripData),      // DATA_SIZE wide signal
                    .FastOr(FastOr)             // N_SUBMAT  wide signal
                    );
```

StripAnalog presently just samples data coming from the MC generator and provides a "fake" ToT. It will be replaced with the real Analog model (FrontEnd + ToT logic) as soon as it will be available.

StripDigital

StripDigital contains 128 copies of SingleStrip, which stores Data in buffers.

```
module SingleStrip(// Inputs
        input wire [TS_SIZE-1:0]     TSCnt,     // TimeStamp Counter
        input wire [TS_SIZE-1:0]     TSReq,     // Requested Time Stamp
        input wire [TOT_SIZE-1:0]    ToT,       // ToT coming from the analog part
        input wire                   Read,      // If 1 we read out the strip memory data
                                     Clear,     // If 1 we clear the strip memory data
                                     Write,     // If true we have to write data
                                     CK,        // System clock
                                     BCK,       // BC clock
                                     Reset,     // Active high Reset
        // Outputs
        output reg [DATA_SIZE-1:0]   StripData, // Data to be read (DATA_SIZE Data words)
        output reg                   HasData    // True if the strip has data
        );
DataWord [MEM_SIZE-1:0]              Data;      // Data to store Hits
WritePointer                         WrPtr;     // Write pointer
ReadPointer                          RdPtr;     // Read pointer
...
```

DataWord is the buffer and is addressed by a RdPtr and a WrPtr.

MEM_SIZE is the depth of the buffer (16, 32, ..., bit deep)

DATA_SIZE is the width of each output word. (10 TS bit + 4 ToT bit wide)

Conclusions

- We presently have a working framework for simulating the entire chip:
  -> Digital In-strip logic has been implemented.
  -> Interfaces correctly with existing Digital output logic.

- Ongoing simulations will provide the depth of buffers with 100% efficiency.

Still to be done

- Implement "correct" Analog section, with chosen ToT generation.
- Investigate some crucial design parameters:

  -> Area of the Buffers in function of their depth.
  -> Possible power consumption.

Need to perform some preliminary synthesis with chosen technology.