Updates on KLM TSIM Software Activities

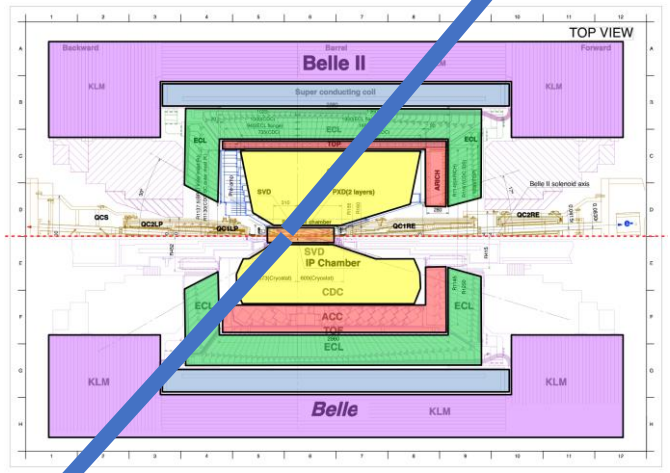
2022-08-22

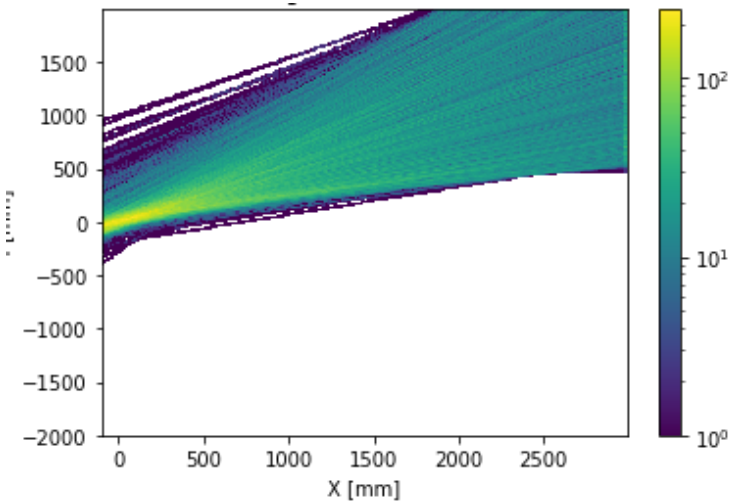
Work Packages

Work Packages	Topic	Status
WP1: Current Trigger Logic	Verifying status of current version	Done
	IO Model of Current Firmware	Done
WP2: Additional Parameters	Back-to-Back flag	Done
	Number of sectors parameter	Done
WP3: Perform Study	MuMu B2B event	In Progress
	Background studies	In Progress
WP4: Publishing results	Webspace for publishing results (stash)	Done
WP5: Upgrade	Simulation of straight line fit	In Progress
	Implementation of straight line fit	In Progress

KLM Trigger Upgrade: Straight Line fit

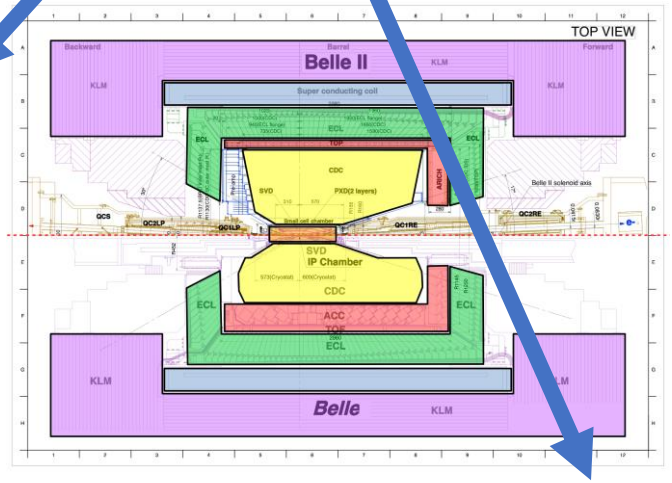
Signal Event



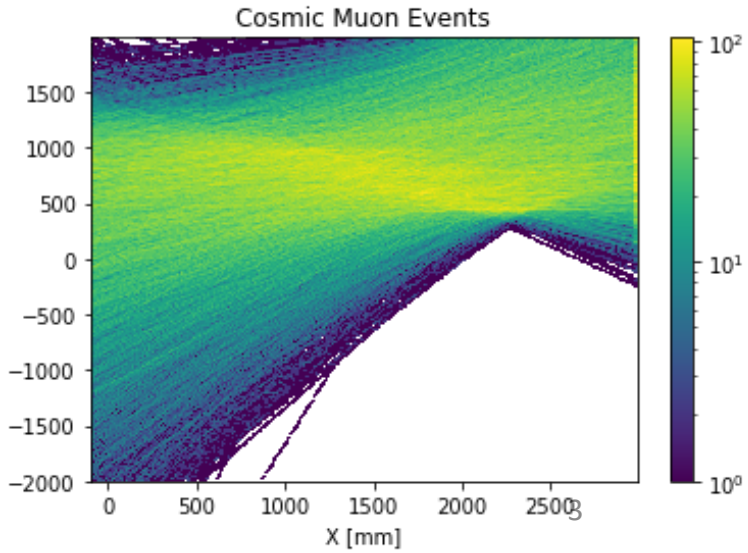


Trigger rate Goals:
Single Muon ≤0.5 kHz
Di Muon ≤0.25 kHz

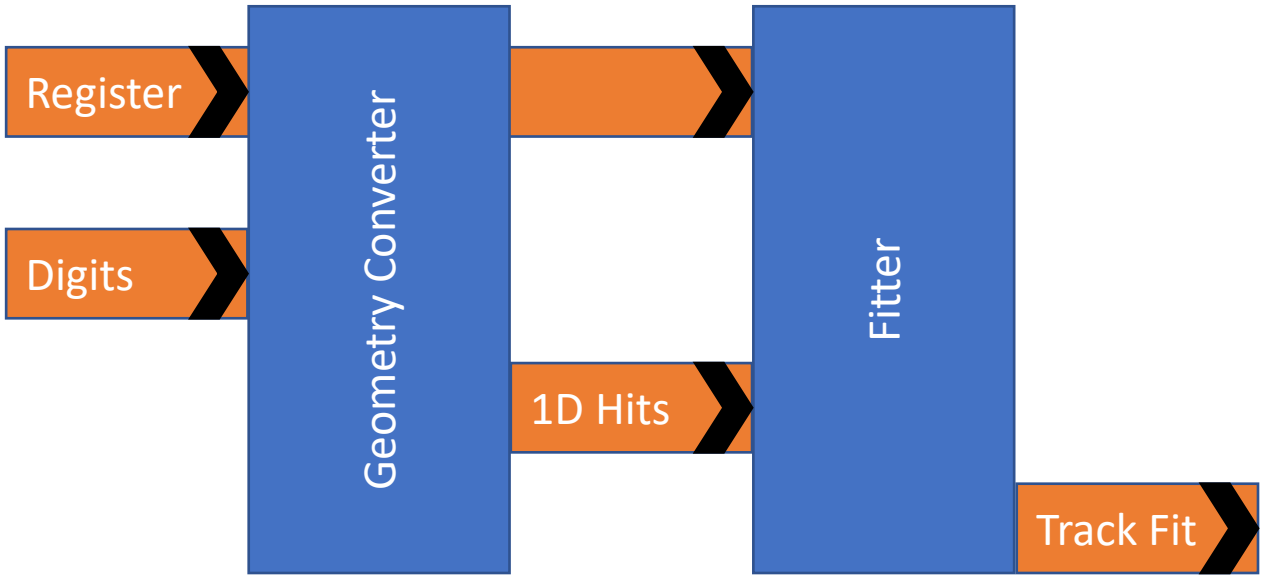
Cosmic Muons



Trigger rate:
Cosmic Muons ~ 5 kHz



Straight-Line Fitter



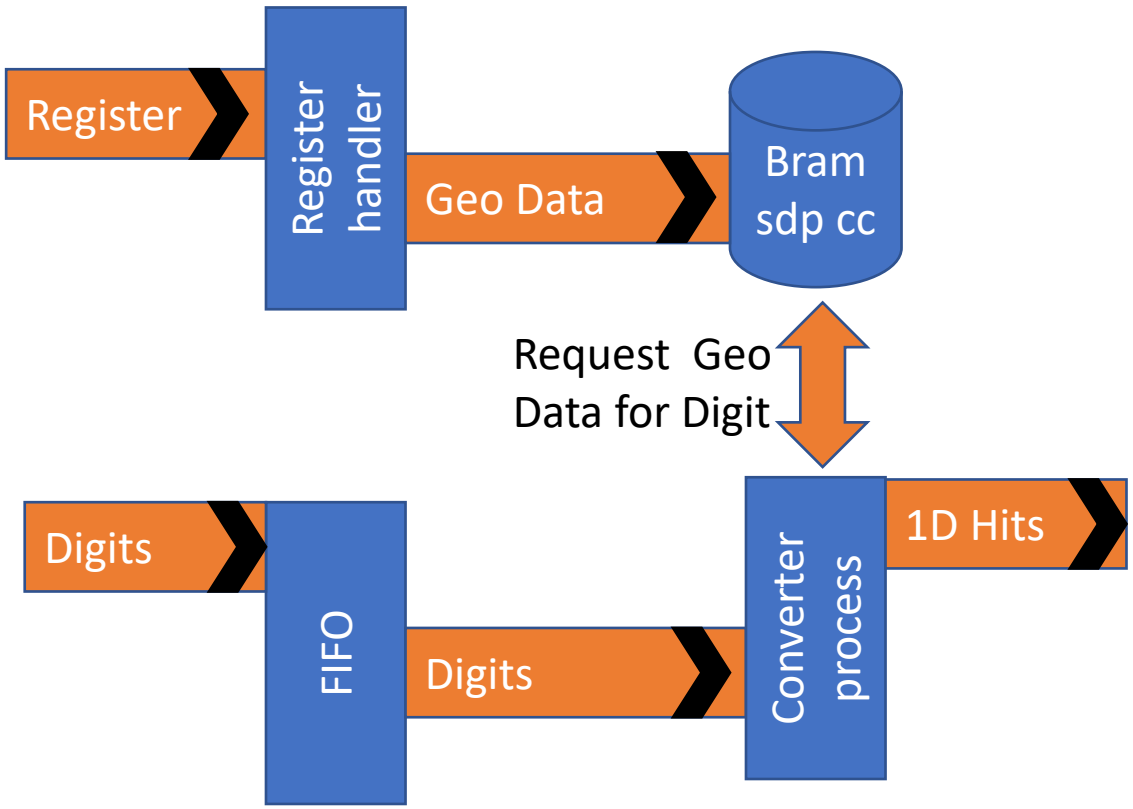
- The Straight-line fitter entity is split into two sub-entities.
- Geometry converter:
  Takes raw KLM Digits and converts them to 1D hits with X-Y Coordinates
- Fitter:
  The fitter takes the 1D hits and uses a least square fit to make the track fit

Geometry Converter

```
entity geometryConverter_full is
  port (
      clk                       : in std_logic;
      rst                       : in std_logic;
      reg                       : in register_32T;


      hit_1d_i                  : in Trg1DHitFormat;
      hit_1d_i_missed_events    : out slv_16;

      hit1d_out                 : out hit1d;
      hit1d_out_ready           : in std_logic

  );
end entity;
```

- Common interface containing: clk, rst and register interface
- Input data-stream is KLM Digits. This record is already used by the current UT3 firmware
- KLM digits don’t allow for back pressure. Therefore, Digits are stored in a FiFo. If the FiFo Overflows, it will miss events. Missed events are counted in “hit_1d_i_missed_events”.
- “hit1d_out” is the 1D hit output from the converter.
- Together with hit1d_out_ready it uses an AXI4-Stream interface to send data to the next module. (It handles back pressure)

Geometry Converter



- The Register handler translates the register address in specific slope offsets for specific sectors axis
- The data is then stored in RAM.
- The Digits are first stored in the input FIFO and from there process by the converter process.
- The Converter process has read access to the RAM.
- For Ever Digits it requests the corresponding slope/offset information from RAM
- It then uses Slope/Offset to convert digits into x/y coordinates

Register Handler

```
read_data_s(reg, x_slope,  reg_x_slope);
read_data_s(reg, x_offset, reg_x_offset);
if is_slope_register (reg.address)  then
    write_value(ram_slope, convert_ram_addr(reg.address) , reg.value);
end if;

if is_offset_register(reg.address) then
    write_value(ram_offset, convert_ram_addr(reg.address) , reg.value);
end if;
```

Applying Geometry

```
if isReceivingData(in_fifo_TX_slave) and ready_to_send(data_out_tx_master) then
  observe_data(in_fifo_TX_slave, v_hit_1d );

  read_addra := convert_ram_addr(v_hit_1d);
  request_addr(ram_offset ,read_addra );
  request_addr(ram_slope , read_addra );

  if is_read_addr(ram_offset , read_addra) and is_read_addr(ram_slope, read_addra) then

    read_data(in_fifo_TX_slave, v_hit_1d );

    v_hit1d_out      := Make_1D_hit( layer         => v_hit_1d.Layer,
                                     Channel       => v_hit_1d.Channel,
                                     y_slope       => get_value(ram_slope ),
                                     y_offset      => get_value(ram_offset ) ,

                                    Sector        => v_hit_1d.Sector,
                                     Axis          => v_hit_1d.Axis,
                                     Subdetector   =>  v_hit_1d.subdetector
                        );

    send_data(data_out_tx_master, v_hit1d_out);
  end if;
end if;
```

Fitter

```
ENTITY linerFitter_full IS
  PORT (
    clk : IN STD_LOGIC;
    rst : IN STD_LOGIC;
    reg : IN register_32T;

    hit1d_in : IN hit1d;
    hit1d_in_ready : OUT STD_LOGIC;

    klm_trg_out : OUT klm_trg

  );
END ENTITY;
```

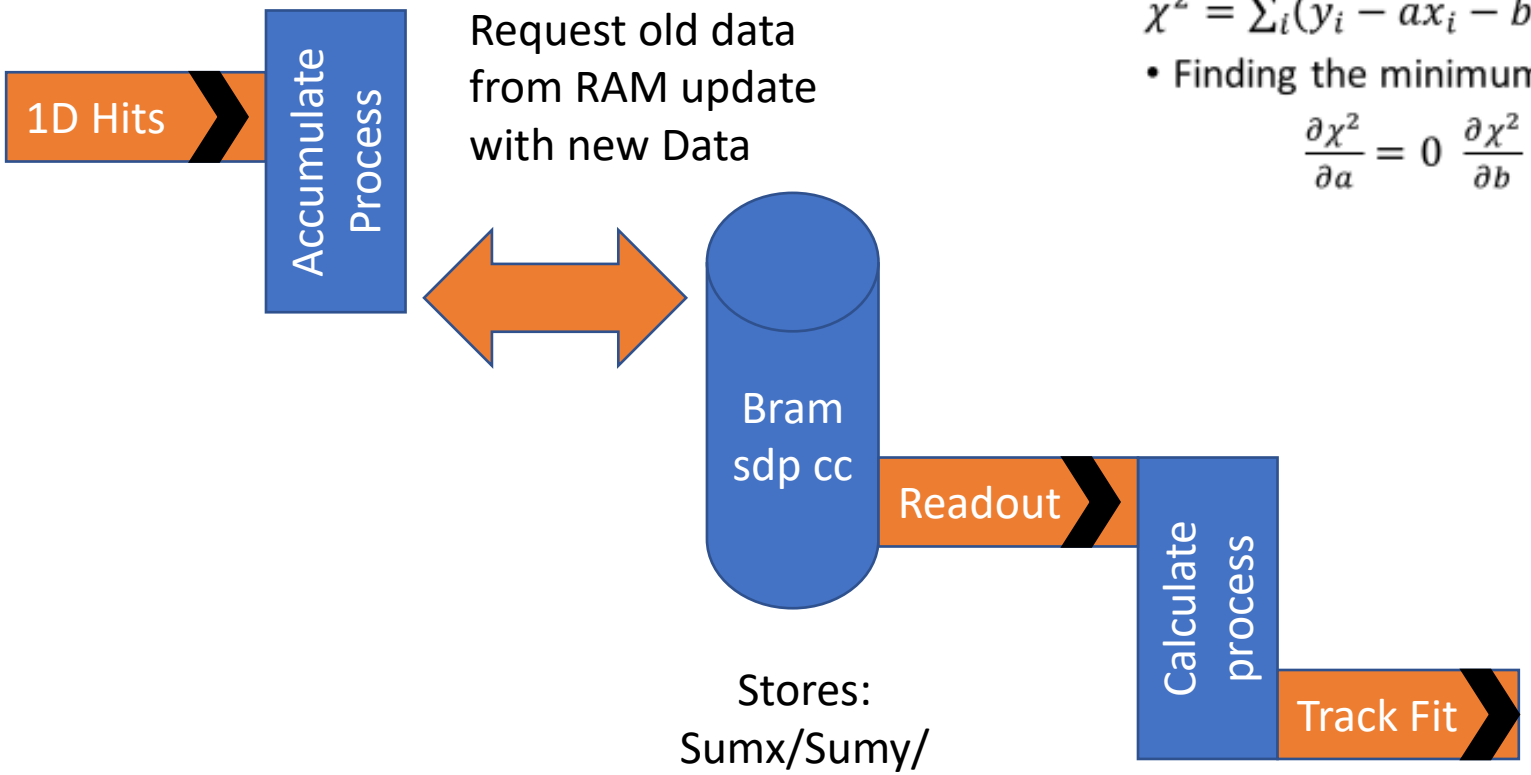
- Common interface containing: clk, rst and register interface
- Input Data-Stream is 1D hits.
- Data stream output is fitted track data (klm_trg).
- Klm_trg_out is streamed out after a certain amount of clock cycles without new input.

Output Record

```
type klm_trg is record
  event_nr    : slv_32;
  slopeXY     : slv_32;
  interceptXY : slv_32;
  ipXY        : slv_32;
  chisqXY     : slv_32;
  nHits       : slv_32;
  Axis        : slv_32;
  Sector      : slv_32;
  Subdetector : std_logic;
  valid       : std_logic;
end record;
```

- This record is the 1 to 1 equivalent of the TSIM output class
- It is not yet clear how the trigger group wants to use this information therefore the output types are all slv_32. Also, at the moment there is not back pressure implemented

Fitter



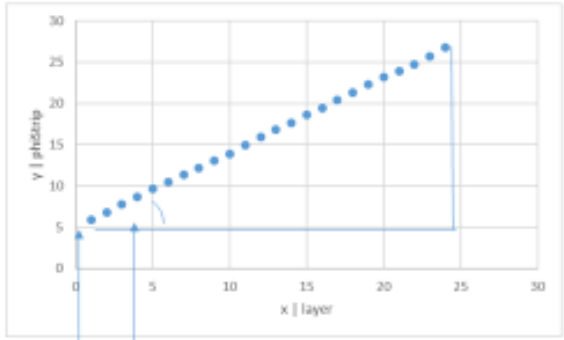
Fit Parameter I

- Linear Equation:
  $y=ax+b$
- Definition of χ^2

$$\chi^2 = \sum_i (y_i - ax_i - b)^2$$

- Finding the minimum by solving

$$\frac{\partial \chi^2}{\partial a} = 0 \quad \frac{\partial \chi^2}{\partial b} = 0$$



- Final equations for a and b:

$$a = \frac{(\sum_i x_i \sum_i y_i - N\sum_i x_i y_i)}{(\sum_i x_i)^2 - N\sum_i x_i^2}$$

$$b = \frac{(\sum_i x_i \sum_i x_i y_i - \sum_i y_i \sum_i x_i^2)}{(\sum_i x_i)^2 - N\sum_i x_i^2}$$

Common term: Denom:

$$denom = \left(\sum_i x_i\right)^2 - N\sum_i x_i^2$$

N... Number of points

$$sum_{yy} = \sum_i^N y_i^2 \; ;$$

$$sum_{xy} = \sum_i^N x_i y_i \; ;$$

$$sum_y = \sum_i^N y_i \; ;$$

$$sum_y = \sum_i^N x_i$$

$$sum_{xx} = \sum_i^N x_i^2$$

Fitter: Accumulate Process

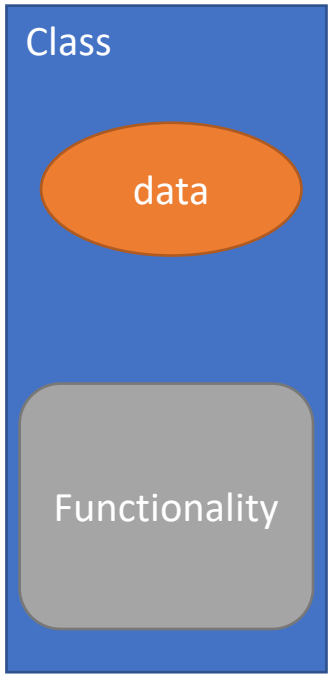
```
IF isReceivingData(in_fifo_TX_slave) THEN
  i_readout_counter<= (OTHERS => '0');
  observe_data(in_fifo_TX_slave, v_hit1d_in_1);
  nHits <= nHits + 1;
  request_addr(ram_data, geo_position_to_number(v_hit1d_in_1));
  IF is_read_addr(ram_data, geo_position_to_number(v_hit1d_in_1)) THEN
    read_data(in_fifo_TX_slave, v_hit1d_in_1);
    update_add(ram_data, X => v_hit1d_in_1.x, Y => v_hit1d_in_1.y );
  END IF;
END IF;
```

```
procedure update_add(signal self : inout linear_fit_ram_T ;    X : in  signed;  Y : in signed )  is
 begin
    self.s2m <= linear_fit_ram_T_s2m_Z;
    self.m2s.wea    <= '1';
    self.m2s.addra  <= self.addrb_current;
    self.m2s.sumX_r_dina  <= self.s2m.sumX_r_doutb + X;
    self.m2s.sumY_r_dina  <= self.s2m.sumY_r_doutb + y;
    self.m2s.sumXX_r_dina  <= self.s2m.sumXX_r_doutb +  x*x;
    self.m2s.sumYY_r_dina  <= self.s2m.sumYY_r_doutb +  y*y;
    self.m2s.sumXY_r_dina  <= self.s2m.sumXY_r_doutb +  x*y;
    self.m2s.nHits_r_dina  <= self.s2m.nHits_r_doutb +  1;
 end procedure;
```

Fitter: Calculate process

```
IF readout = '1’ THEN
  addr_counter <= STD_LOGIC_VECTOR(unsigned (addr_counter) + 1);
  request_addr(ram_data, addr_counter);

  hit_geo := number_to_geo_position(ram_data.addrb_current);

  IF unsigned(ram_data.s2m.nHits_r_doutb) > 0 THEN
    reset_addr(ram_data, ram_data.addrb_current);

    klm_trg_out.valid <= make_klm_tr(hit_geo);
    klm_trg_out.nHits <= ram_data.s2m.nHits_r_doutb;

    denom := calc_denom(ram_data);
    Slope_nom := calc_Slope_nom(ram_data);
    offset_nom := calc_offset_nom(ram_data);

    IF (denom /= 0) THEN

      fixed_point_shift(  denom , Slope_nom , offset_nom );
      klm_trg_out.slopeXY <= Slope_nom /denom;
      klm_trg_out.interceptXY <= offset_nom /denom;

    END IF;
  END IF;
END IF;
```

Designing Objects in VHDL



(Pseudo) Class Definition:

```
type axiStream_m is record
  last  : std_logic;
  valid : std_logic;
  data  : std_logic_vector(31 downto 0);
  ready : std_logic;
end record;
```

Connects to the in/out ports of the entity
Absorbs interface specific code that needs to run on every clock cycle

```
procedure pull       ( self : inout axiStream_m;  signal tx     :  in   axiStream_s2m);
procedure push       ( self : inout axiStream_m;  signal tx     :  out  axiStream_m2s);
procedure send_data  ( self : inout axiStream_m;         dataIn :  in   std_logic_vector);
function  ready_to_send ( self : axiStream_m) return boolean;
```

Procedures can modify inputs but do not return something
Functions cannot modify inputs but do return something

Call syntax

- **C++:**

  Obj.send_data(MyData) ;

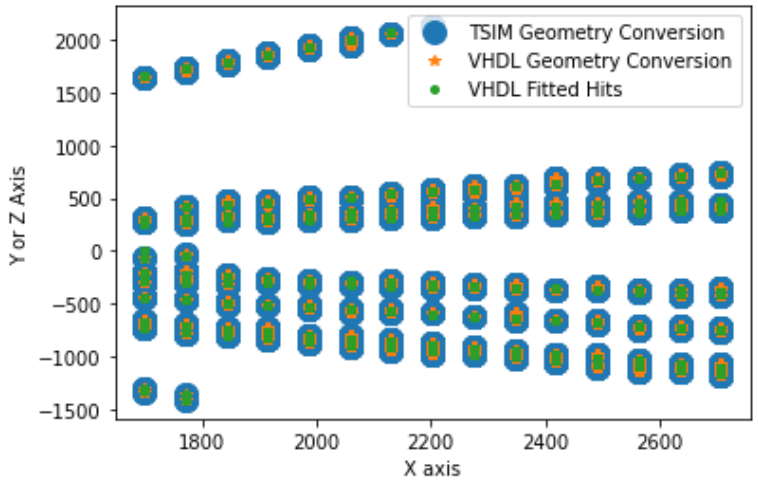
- **VHDL:**

  send_data(obj, MyData);

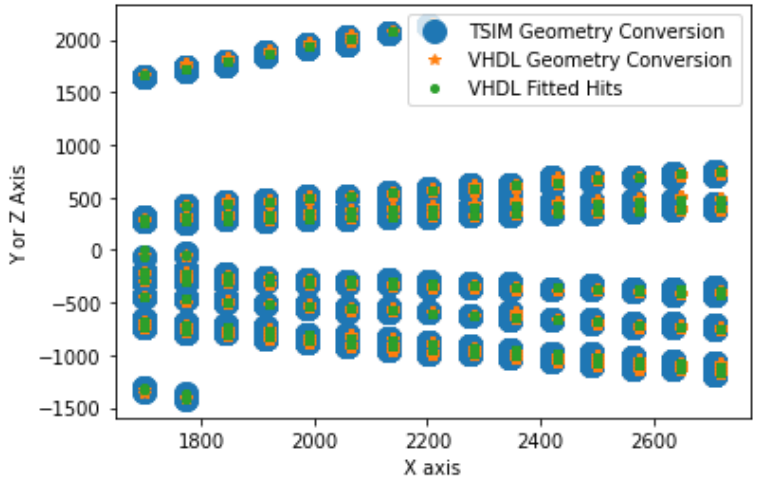
Overload resolution will pick the right function

Results

TSIM Int

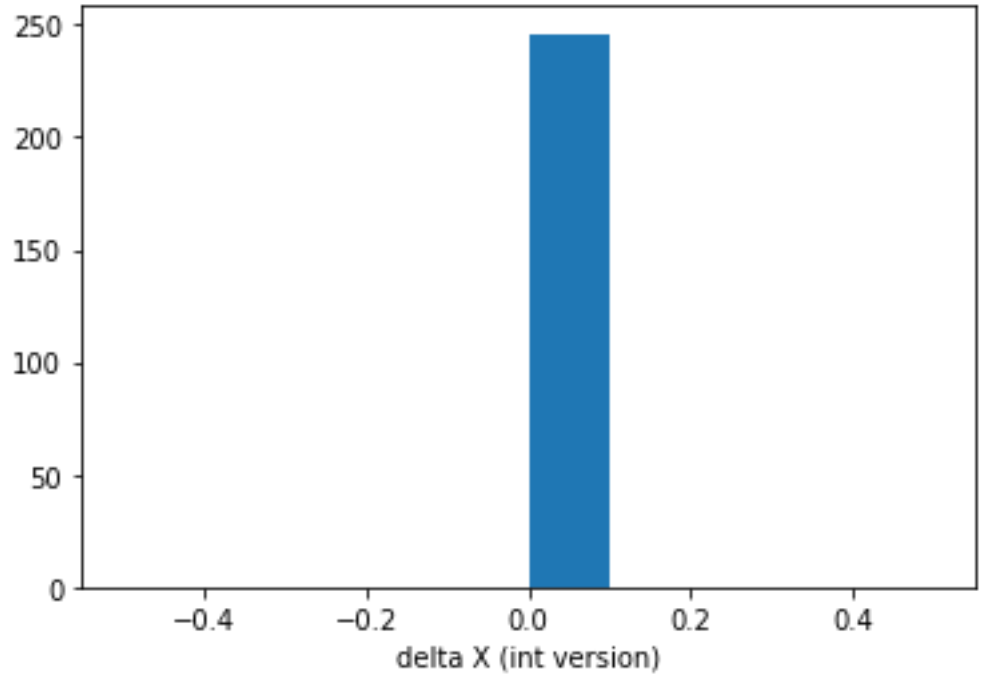


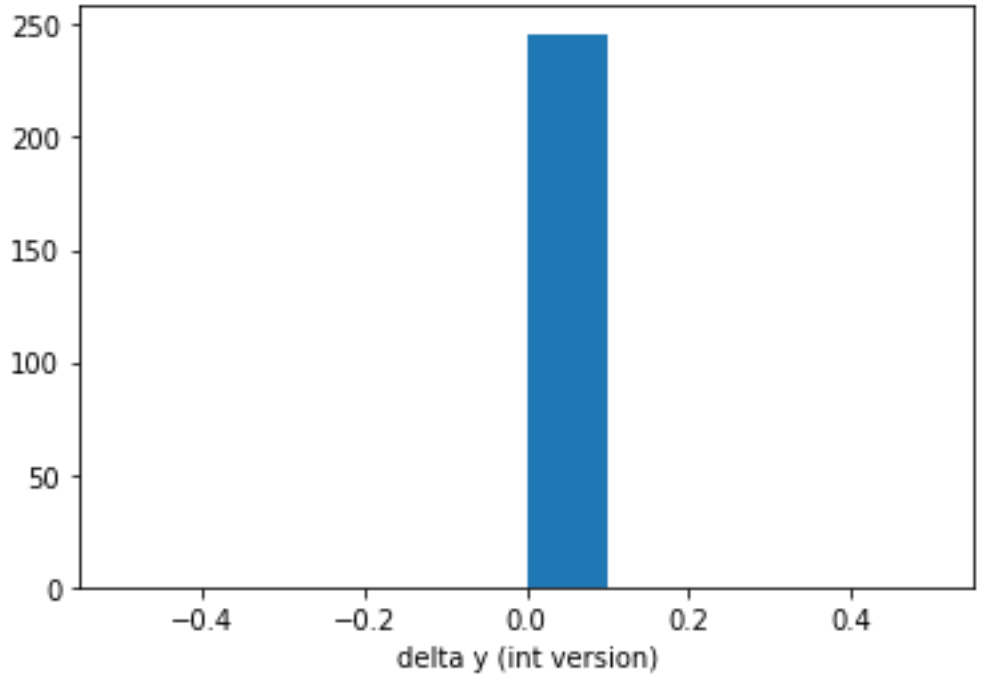
TSIM Float



- The VHDL implementation gives the same values as the C++ version
  - The top version using int values to represent 1D Hits in both C++ and VHDL
  - The bottom version uses floats for the C++ implementation

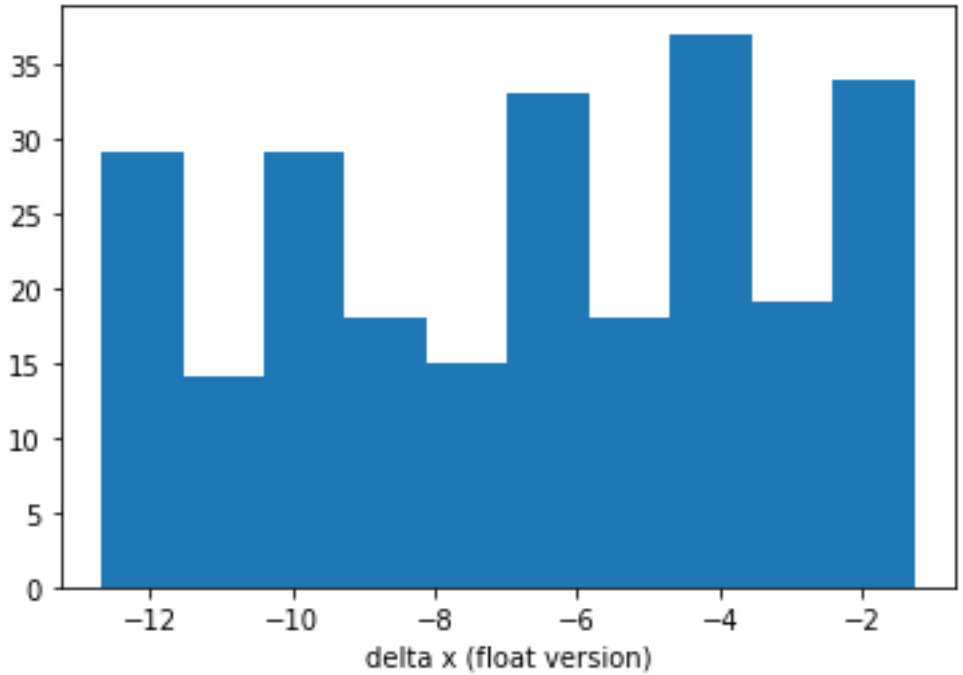
Residuals (int version)

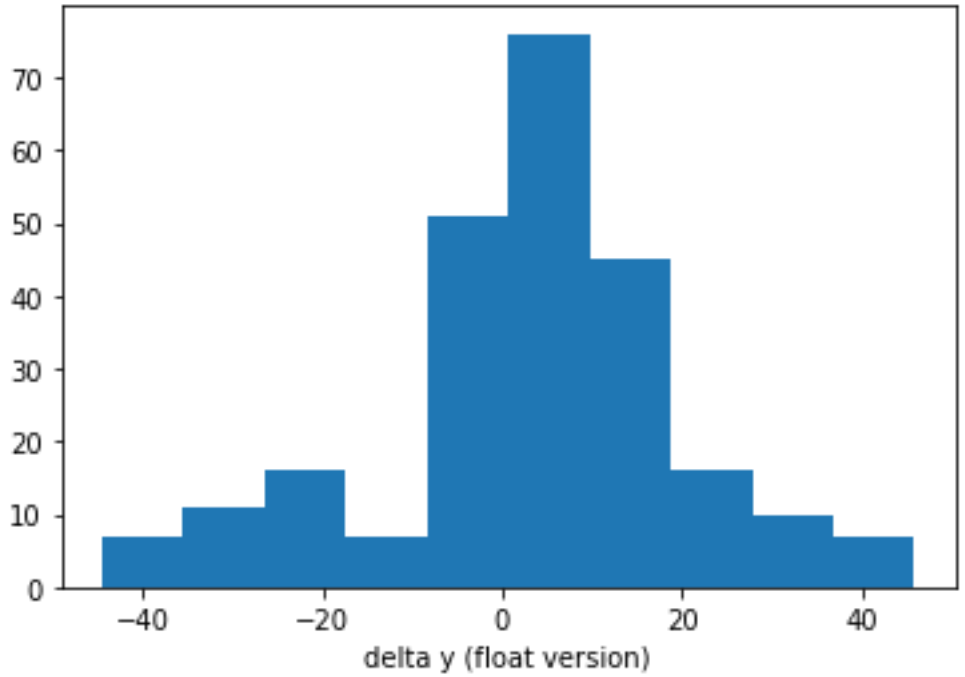




Delta x/y is the difference between the TSIM version and the VHDL version

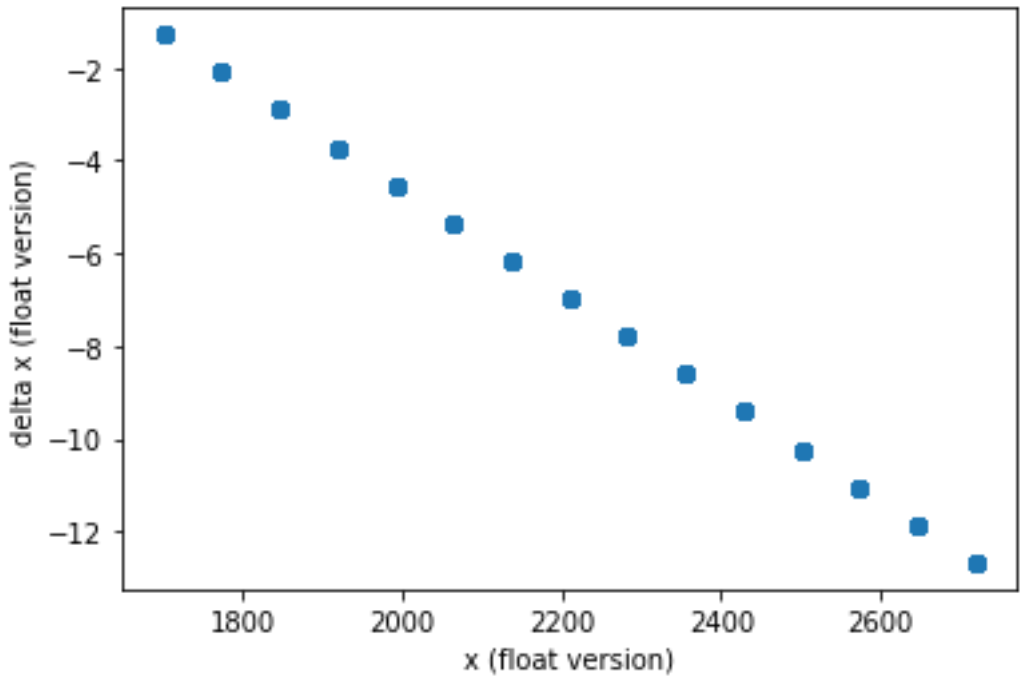
Residuals (float version)

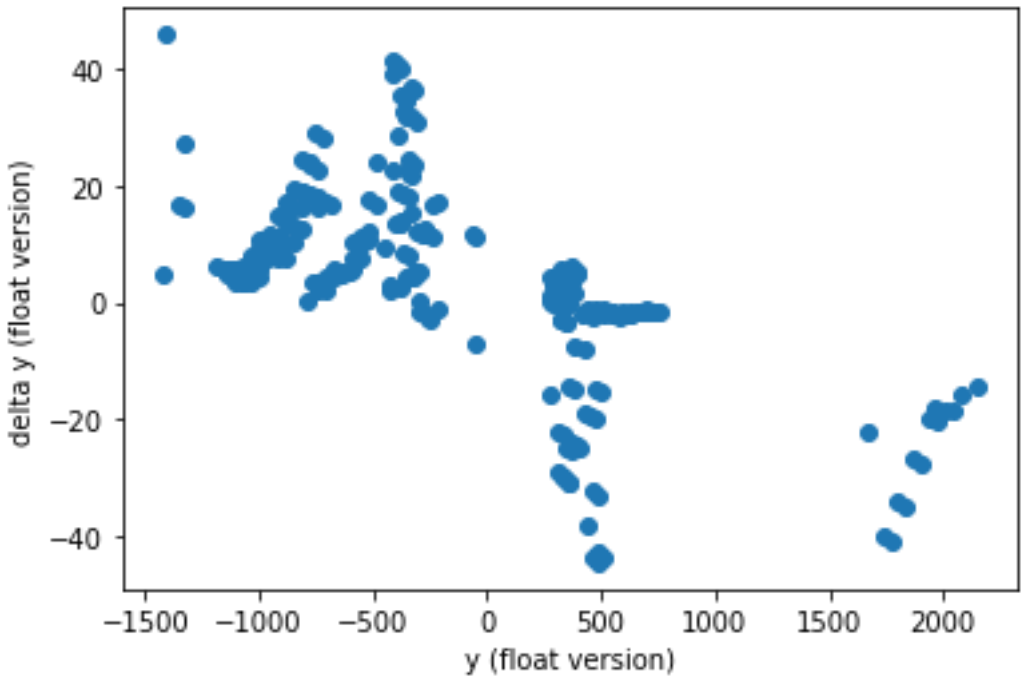




Delta x/y is the difference between the TSIM version and the VHDL version

Residuals (float version) as function of position





Delta x/y is the difference between the TSIM version and the VHDL version

Summary

- VHDL Version of the straight-line fit has been implemented
- In Simulation it shows perfect agreement with the C++ version
- Implementation has been synthesized on spartan 6 (SCROD board)

Input Needed

- Merging into the main project
- Input and output matching
- What information does the trigger group want to have from this fitter