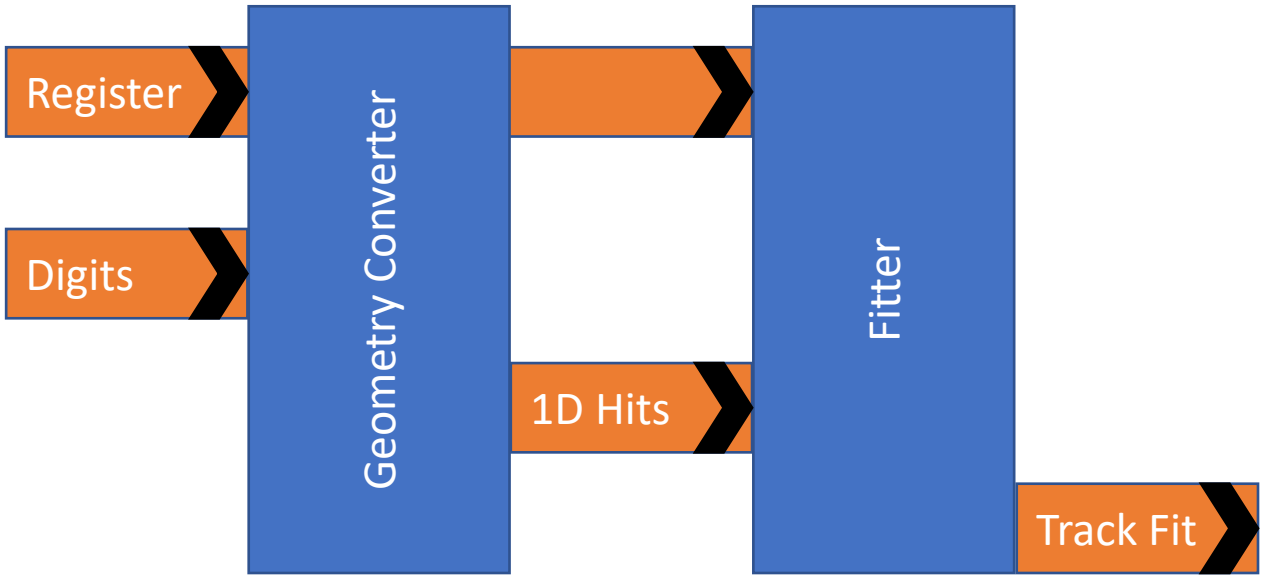
Updates on KLM TSIM Software Activities

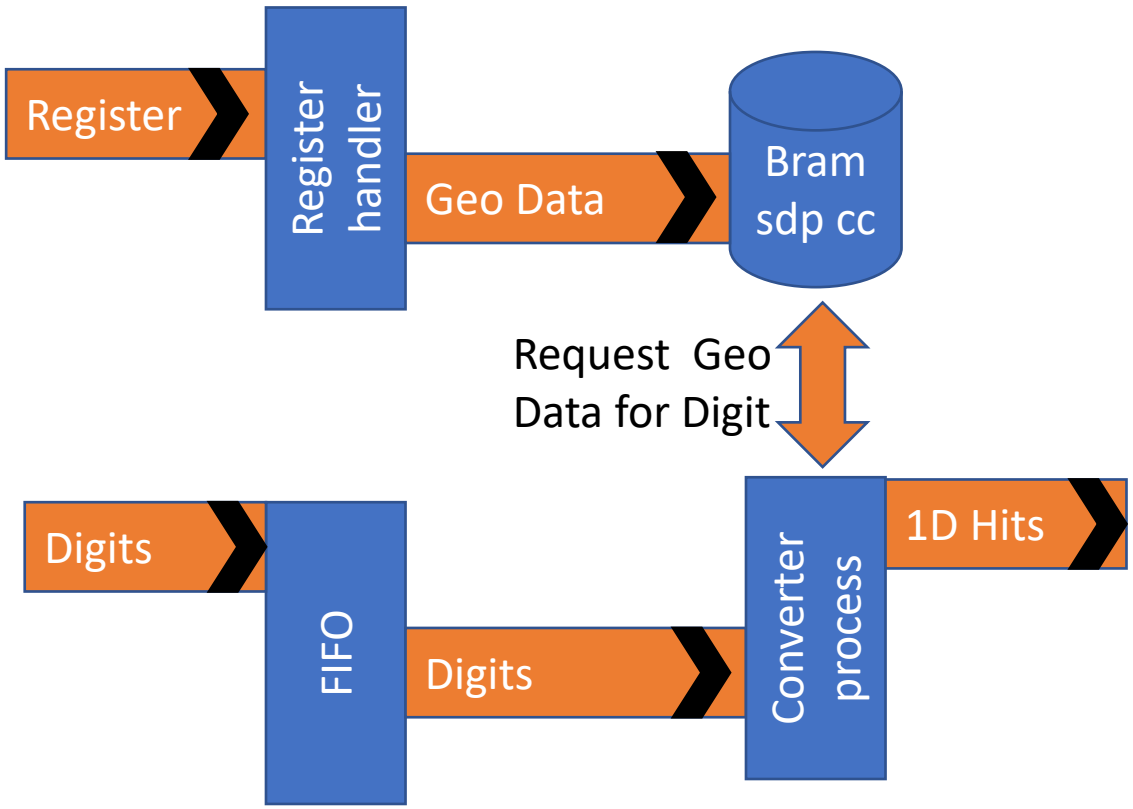
2023-06

Straight-Line Fitter



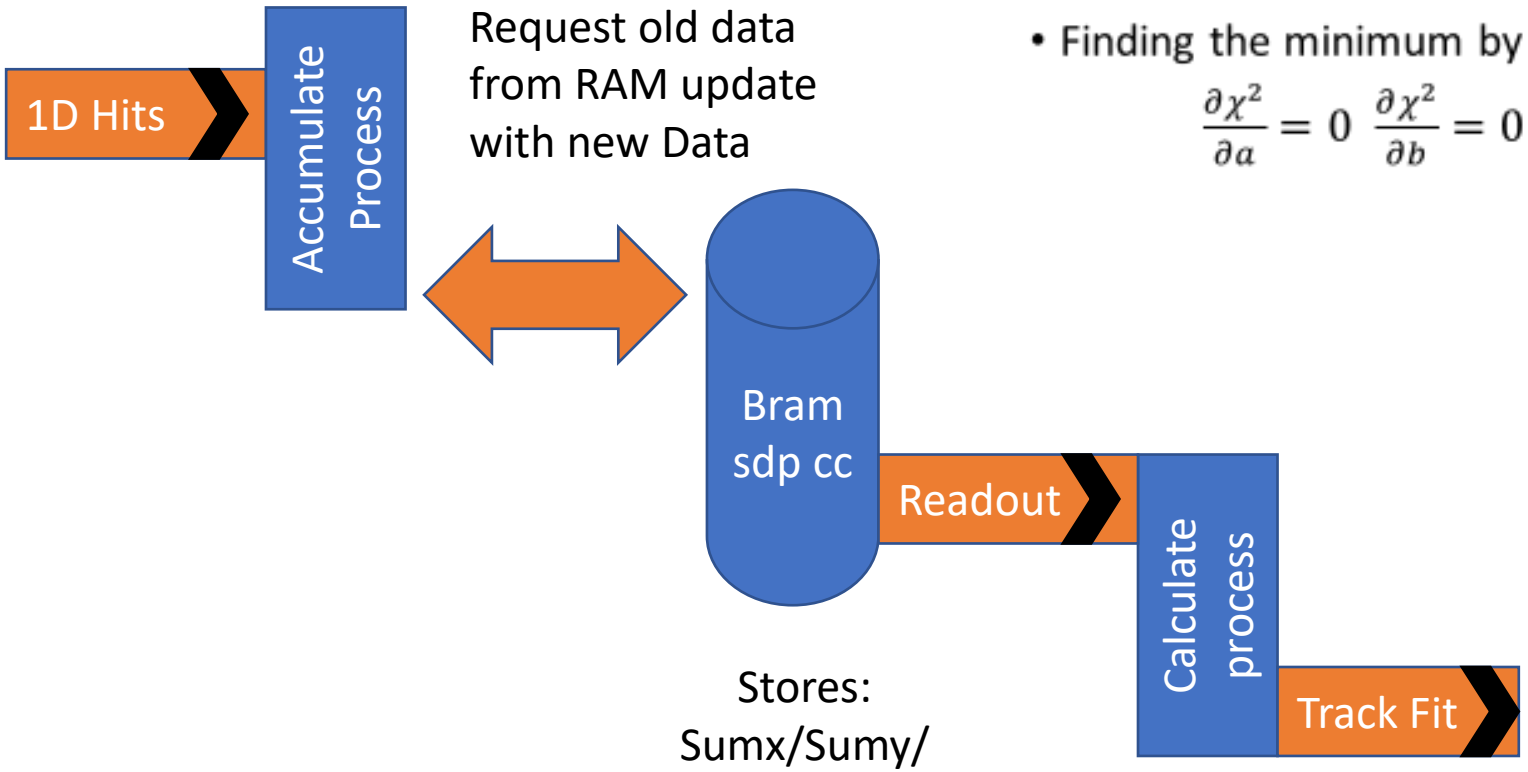
- The Straight-line fitter entity is split into two sub-entities.
- Geometry converter:
  Takes raw KLM Digits and converts them to 1D hits with X-Y Coordinates
- Fitter:
  The fitter takes the 1D hits and uses a least square fit to make the track fit

Geometry Converter



- 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

Fitter



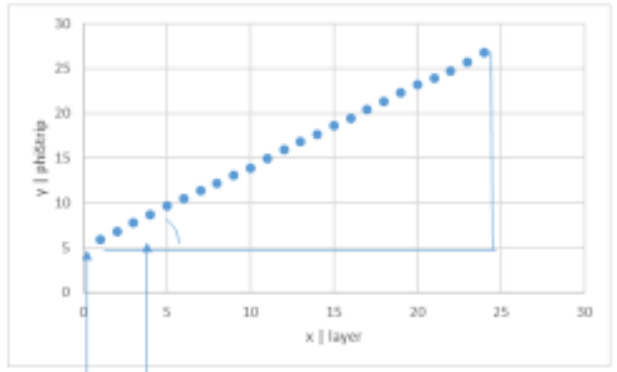
Fit Parameter I

- 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

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

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

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

$$sum_y = \sum_i^N x_i$$

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

Test Setup

KEK Operations

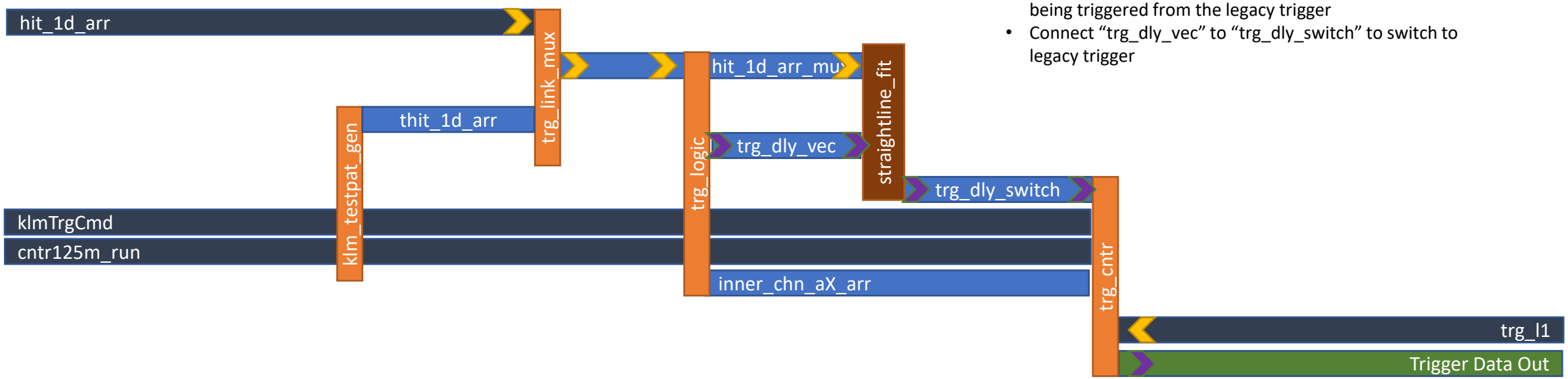
- First Firmware test were successful in so far as they showed that all debug signals were accessible, and the geometry conversion was working correctly
- However, there was a server update and so far, I have not been able to get full access to the UT3 again.

UH Setup

- Emulating the UT3 with a standard Pynq board
- Since vivado and ISE are not 100% Compatible the firmware cannot just be compiled with vivado
- A Simplified version is now used for the test

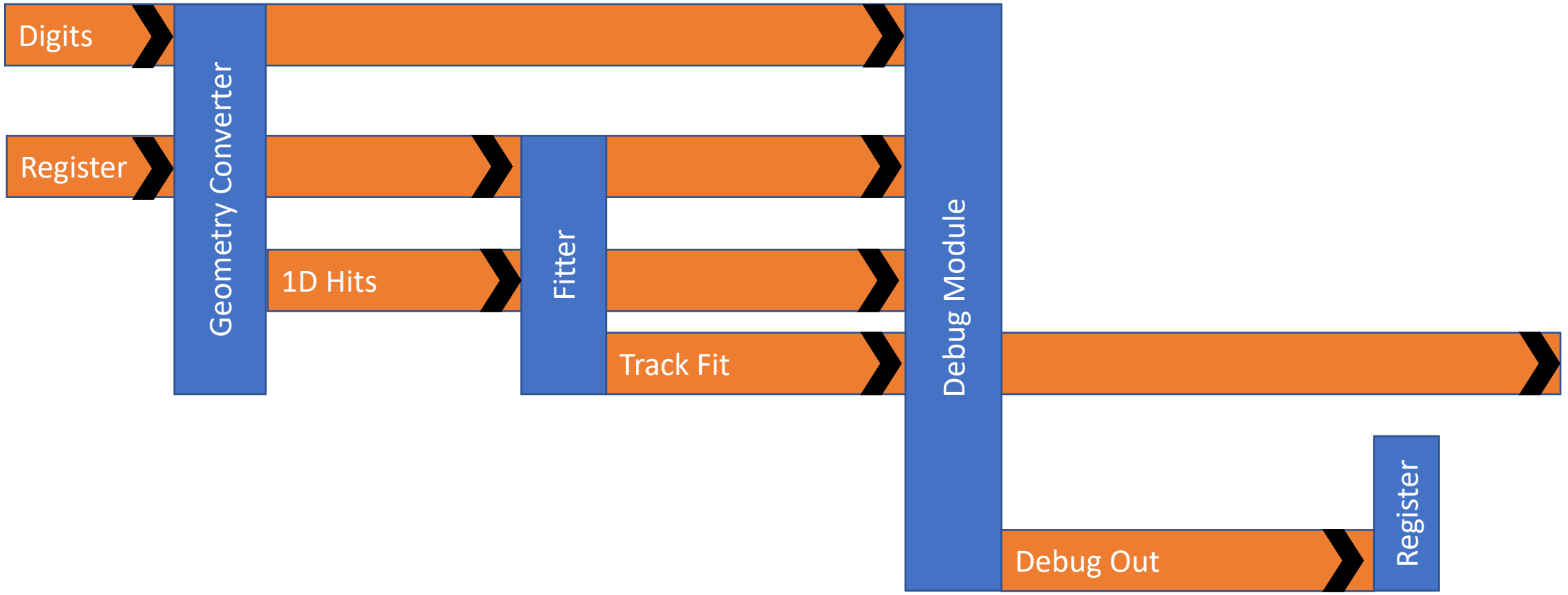
subdetector_top

- Straight line fit trigger runs parallel to current trigger
- Readout of the new trigger can be done by timeout or by being triggered from the legacy trigger
- Connect “trg_dly_vec” to “trg_dly_switch” to switch to legacy trigger



- SLF is instantiated inside subdetector_top
- This means it runs on subdetector level (EKLM BKLM)
- The module can easily be scaled down to run on sector level if more speed is required
- It can also be scaled up to handle the entire KLM (EKLM+BKLM) (its lower but needs less recourses)

Straight-Line Fitter Debug Module



Testing (pynq) test_pattern_storage

In [174]:
```
for i in range(10):
    klm.Channel = i+2
    klm.Layer = i
    klm.WEN = 1
    klm.Sector = 1
    klm.subdetector = 1
    print(klm.serialize(),  klm.Layer, klm.Channel )
    tpg.fill_test_pattern(klm.serialize())
    # slf.fill_klm_digits(klm.serialize())
```

```
4107 0 2
6283 1 3
8459 2 4
10635 3 5
12811 4 6
14987 5 7
17163 6 8
19339 7 9
21515 8 10
23691 9 11
```

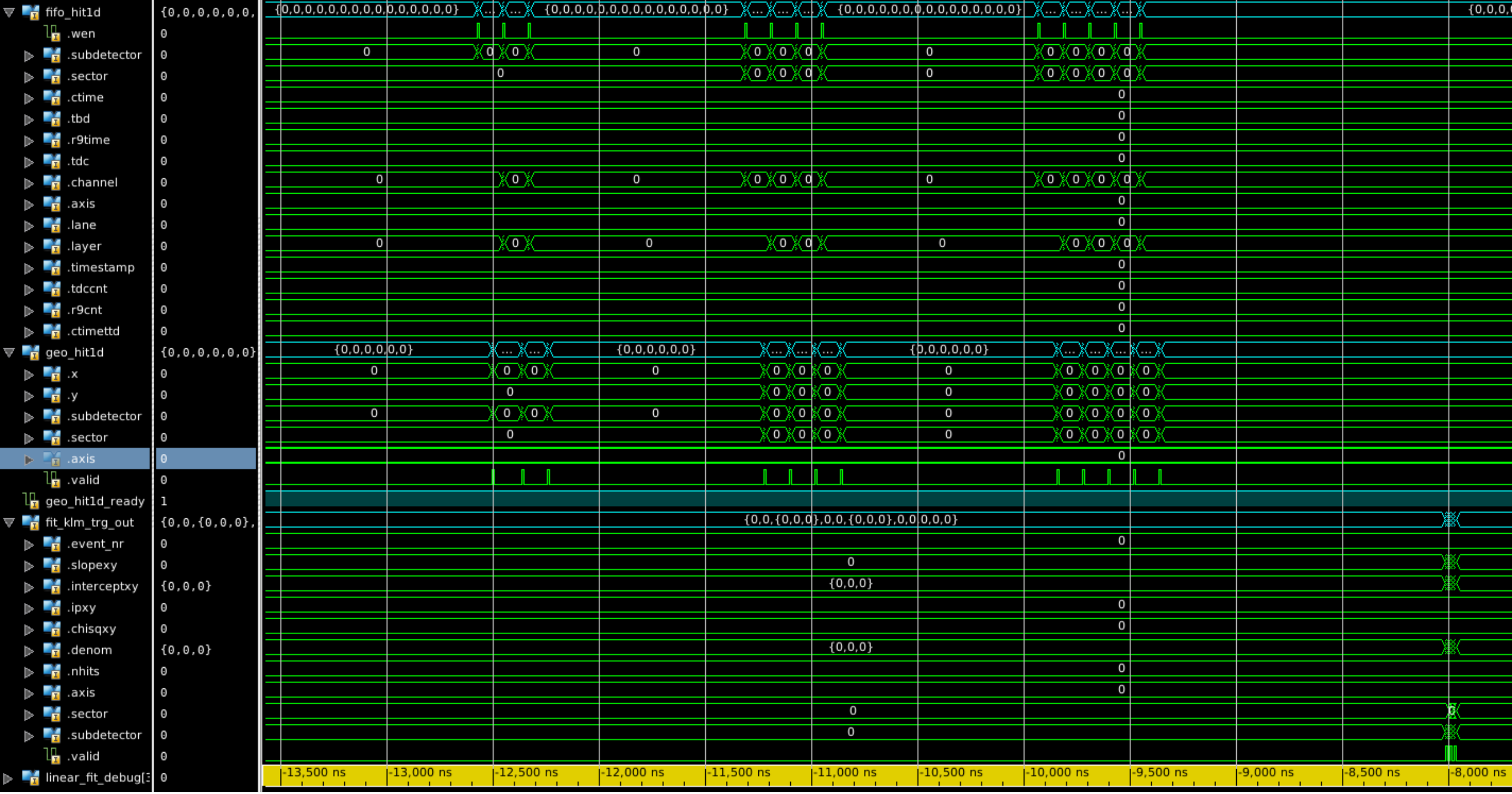
In [173]:
```
def to_digit(num):
    klm.des(num)
    return [num, klm.Layer, klm.Channel]

pd.DataFrame( [to_digit(x[3])
               for x in  slf.read_klm_digits_full()],
              columns = ["num", "layer" , "Channel"])
```

Out[173]:

	num	layer	Channel
0	4107	0	2
1	6283	1	3
2	8459	2	4
3	10635	3	5
4	12811	4	6
5	14987	5	7
6	17163	6	8
7	19339	7	9
8	21515	8	10
9	23691	9	11

- Test Pattern Storage and sending works


fifo_hit1d
{0,0,0,0,0,0,0,
.wen
.subdetector
.sector
.ctime
.tbd
.r9time
.tdc
.channel
.axis
.lane
.layer
.timestamp
.tdccnt
.r9cnt
.ctimettd
geo_hit1d
{0,0,0,0,0,0}
.x
.y
.subdetector
.sector
.axis
.valid
geo_hit1d_ready
fit_klm_trg_out
{0,0,{0,0,0},
.event_nr
.slopexy
.interceptxy
{0,0,0}
.ipxy
.chisqxy
.denom
{0,0,0}
.nhits
.axis
.sector
.subdetector
.valid
linear_fit_debug[
{0,0,{0,0,0},0,0,{0,0,0},0,0,0,0}
-13,500 ns
-13,000 ns
-12,500 ns
-12,000 ns
-11,500 ns
-11,000 ns
-10,500 ns
-10,000 ns
-9,500 ns
-9,000 ns
-8,500 ns
-8,000 ns

Testing (pynq) geometryConverter_full

In [176]:
```
def to_x_y(num):
    geo.des(num)
    return [num, geo.x, geo.y]

pd.DataFrame( [ to_x_y(x[2])
                for x in slf.read_hit1d_full() ],
              columns = ["num", "x" , "y"])
```

Out[176]:

	num	x	y
0	1676995	1627	3
1	2205891	1699	4
2	2734787	1771	5
3	3263683	1843	6
4	3792579	1915	7
5	4321475	1987	8
6	4850371	2059	9
7	5379267	2131	10
8	5908163	2203	11
9	6437059	2275	12

- Geometry Conversion is Working as well
- This Example uses a simplified geometry

$$x = layer * M_x + B_x$$
$$y = Channel * M_y + B_y$$

- Mx = 72 | Bx = 1627
- My = 1 | By = 1

Testing (pynq) linerFitter_full

$$b = \frac{(\Sigma x\ \Sigma xy\ - \Sigma y\ \Sigma x^2)}{(\Sigma x)^2\ - N\ \Sigma x^2} = \frac{nom}{denom}$$

- All Parts are getting calculated correctly
- Timing issues with the division

Question:

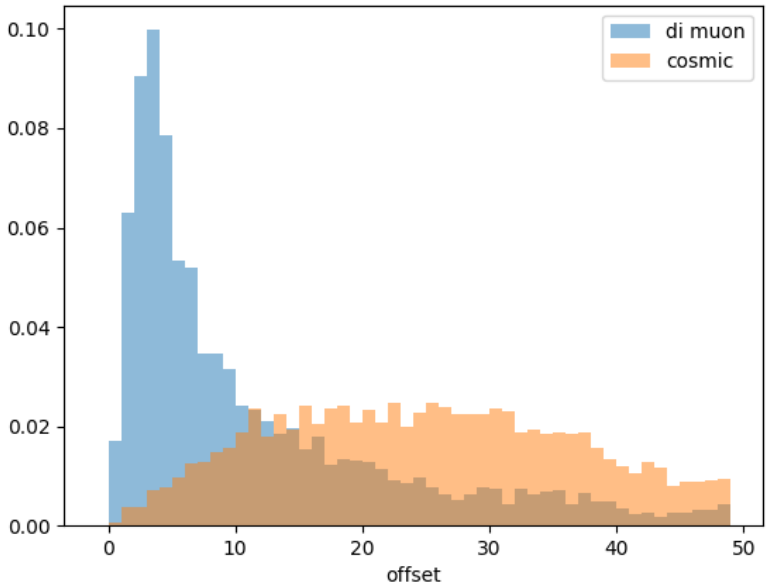
- Do we need the intercept, slope, chi_square?
- If not, we can live without divisions
- Should we move to UT4?

Timing Violations

$$b = \frac{(\Sigma x\ \Sigma xy\ - \Sigma y\ \Sigma x^2)}{(\Sigma x)^2\ - N\ \Sigma x^2} = \frac{nom}{denom}$$

$$abs\left(\frac{nom}{denom}\right) < cut\ Off$$

$$abs(nom) < cutoff\ \cdot abs(denom)$$



- I have some issues meeting timing
- One way of solve a timing violation was to remove divisions and replace them by multiplications
- Still some timing violations in the project
- This doesn’t affect testing since I do the testing currently on the pynq board which doesn’t have the timing violations

Summary & Outlook

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 UT3
- Simplified Version Implementation has been synthesized on Pynq Board

Outlook

- Support for multiple packages
- Testing with larger package sizes
- Integrating into the UT3 module