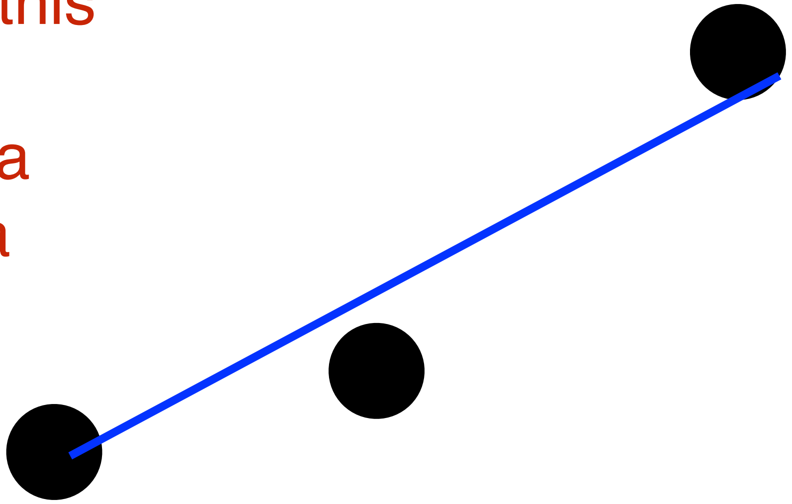
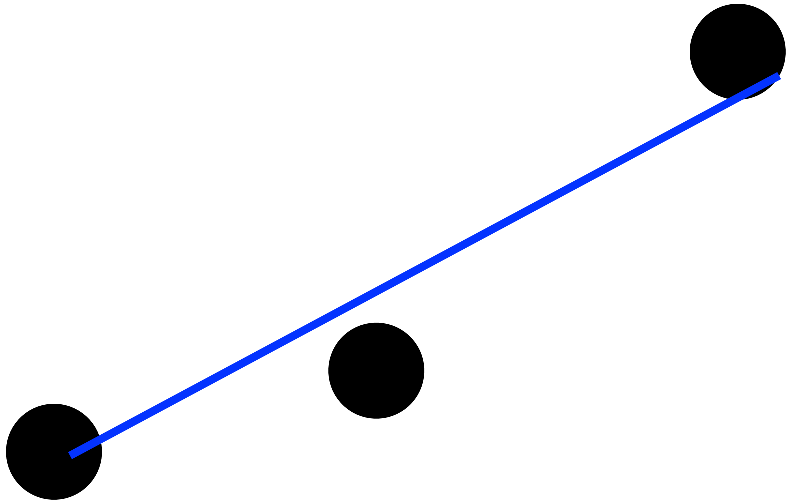
Some familiar slides

For those who took my computational physics class, this should look familiar. But… imagine we have some data points, and we want to fit a straight line to them

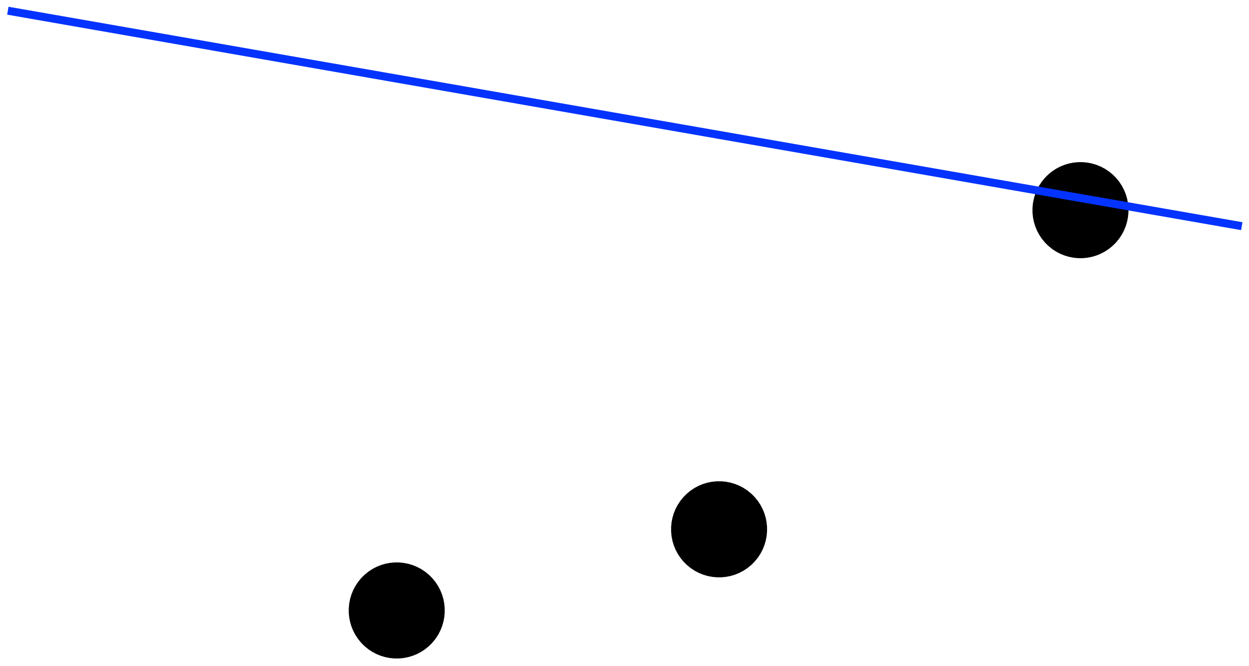


Linear fits



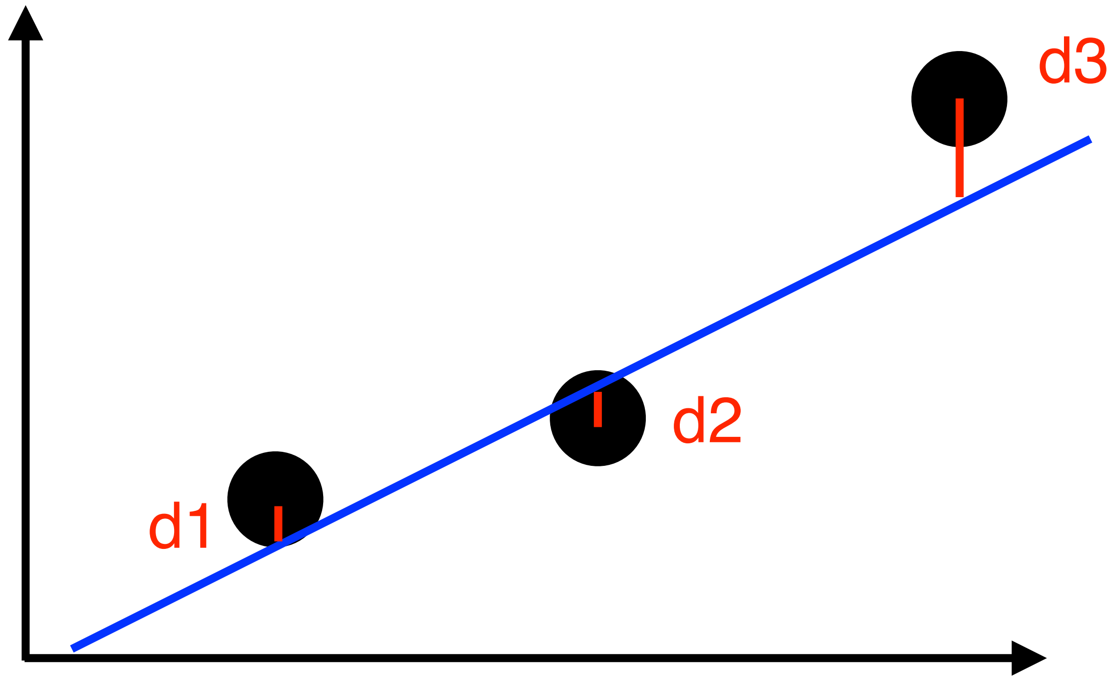
This looks like a reasonable guess and a good fit (of course, we will be more mathematically rigorous shortly)

How to do our linear fits?



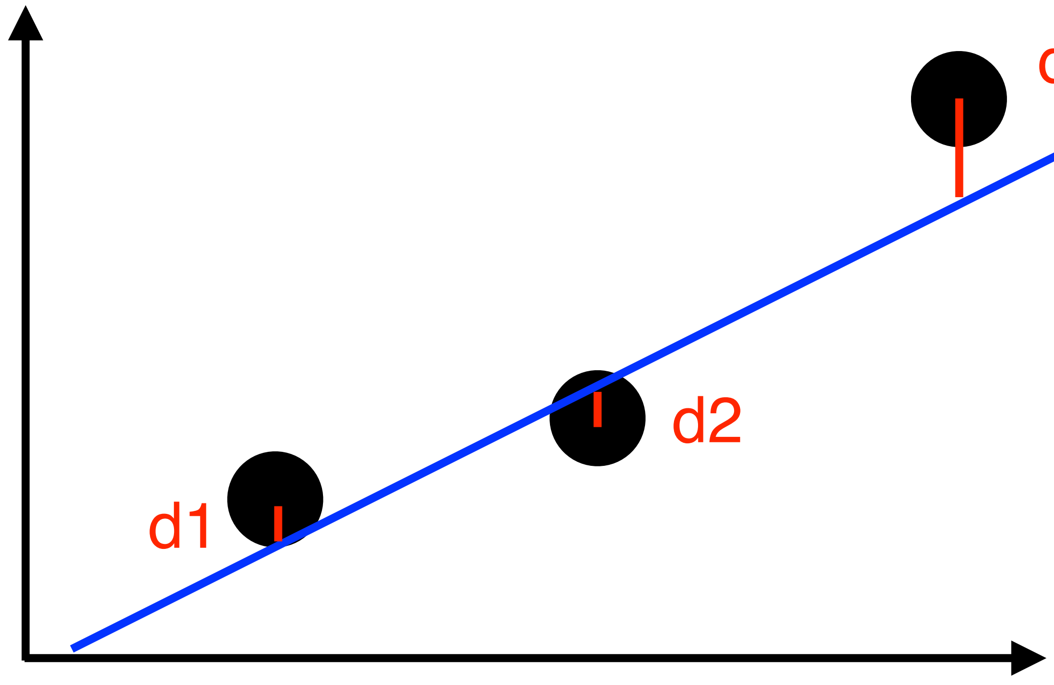
This is clearly not a good fit!

Why does the first line/fit look better?



Assume the errors are all the same (this assumption will be relaxed). Our straight line is given by $y(x) = a+bx$, so that our prediction for point x_i is $y_{pred,i} = a+bx_i$ and then the distance between this and the measured value y_i is given by $d_i = y_i - y_{pred,i} = y_i - a - bx_i$

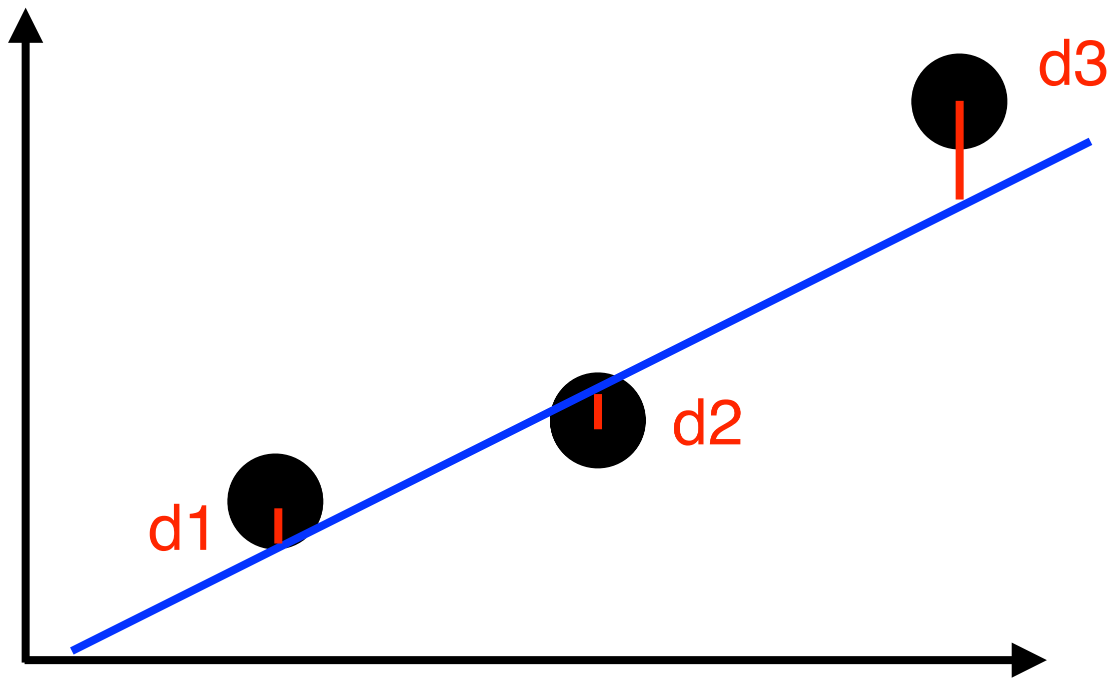
Why does the first line/fit look better?



The distance between the observed and measured value for point i is $y_i - a - bx_i$. Our total "distance" squared is then:

$$f(a,b) = \sum_i (y_i - a - bx_i)^2$$

How to get best-fit parameters



We want to **minimize** the following quantity:

$$f(a,b) = \sum_i (y_i - a - bx_i)^2$$

How to get best-fit parameters

We want to **minimize** the following quantity:

$$f(a,b) = \sum_i (y_i - a - bx_i)^2$$

$$\frac{\partial f}{\partial a} = -2\sum_i (y_i - a - bx_i)$$

$$\frac{\partial f}{\partial b} = -2\sum_i x_i(y_i - a - bx_i)$$

Two equations, two unknowns (a,b)

How to get best-fit parameters

$$\frac{\partial f}{\partial a} = -2\sum_i (y_i - a - bx_i) = 0$$

$$\frac{\partial f}{\partial b} = -2\sum_i x_i(y_i - a - bx_i) = 0$$

$$\sum_i (y_i - a - bx_i) = 0$$

Sum over i takes N possible values

$$\sum_i x_i(y_i - a - bx_i) = 0$$

$$\sum_i y_i - Na - b\sum x_i = 0$$

$$\sum_i x_i y_i - a\sum x_i - b\sum x_i^2 = 0$$

How to get best-fit parameters

$$\sum_i y_i - Na - b\sum x_i = 0$$

$$\sum_i x_i y_i - a\sum x_i - b\sum x_i^2 = 0$$

$$a = \frac{1}{N}\sum y_i - b\frac{1}{N}\sum x_i$$

$$a = \frac{1}{\sum x_i}\sum x_i y_i - b\frac{1}{\sum x_i}\sum x_i^2$$

How to get best-fit parameters

$$a = \frac{1}{N}\sum y_i - b\frac{1}{N}\sum x_i$$

$$a = \frac{1}{\sum x_i}\sum x_i y_i - b\frac{1}{\sum x_i}\sum x_i^2$$

Introduce notation to make this easier, set the 'a' values equal to each other

$$s_y/N - (b/N)s_x = (s_{xy}/s_x) - b(s_{xx}/s_x)$$

$$b(s_{xx}/s_x - s_x/N) = s_{xy}/s_x - s_y/N$$

$$b(Ns_{xx} - s_x^2) = Ns_{xy} - s_x s_y$$

$$b = (Ns_{xy} - s_x s_y)/(Ns_{xx} - s_x * s_x)$$

How to get best-fit parameters

$$a = \frac{1}{N}\sum y_i - b\frac{1}{N}\sum x_i$$

$$a = \frac{1}{\sum x_i}\sum x_i y_i - b\frac{1}{\sum x_i}\sum x_i^2$$

$$s_y/N - a = (b/N)s_x$$

$$s_{xy}/s_x - a = (b/s_x)s_{xx}$$

$$b = s_y/s_x - aN/s_x = s_{xy}/s_{xx} - as_x/s_{xx}$$

$$s_y/s_x - s_{xy}/s_{xx} = a(N/s_x - s_x/s_{xx})$$

$$a = (s_y/s_x - s_{xy}/s_{xx})/(N/s_x - s_x/s_{xx})$$

$$a = (s_y * s_{xx} - s_{xy} * s_x)/(N * s_{xx} - s_x^2)$$

How to get best-fit parameters

$$a = (s_y * s_{xx} - s_{xy} * s_x)/(N * s_{xx} - s_x^2)$$

$$b = (N * s_{xy} - s_x * s_y)/(N * s_{xx} - s_x^2)$$

Just a few simple sums to
calculate, pretty straightforward!

Is that the full picture?

We also want to know three additional, very important quantities:

1) The uncertainty on a
2) The uncertainty on b
3) The uncertainty per degree of freedom of the fit (aka the goodness of the fit). For our linear fit we have two constraints so the variance of the fit, or goodness of fit, is:

$$\sigma^2 = \frac{1}{n-2}\sum(y_i - y(x_i))^2$$

And what do we do if the uncertainty on each point is not equal? Minimize the chi2 instead!

$$\chi^2(a,b) = \sum\left(\frac{y_i - a - bx_i}{\sigma_i}\right)^2$$

Using point-by-point errors (if errors on each value are not all equal)

Uncertainties on parameters!

$$b = \frac{1}{S_{tt}} \sum_{i=0}^{n-1} \frac{t_i y_i}{\sigma_i}\,, \qquad a = \frac{S_y - S_x b}{S} \qquad \sigma_a^2 = \frac{1}{S}\left(1 + \frac{S_x^2}{S S_{tt}}\right)\,, \qquad \sigma_b^2 = \frac{1}{S_{tt}}$$

with

$$t_i = \frac{1}{\sigma_i}\left(x_i - \frac{S_x}{S}\right)\,, \qquad S_{tt} = \sum_{i=0}^{n-1} t_i^2\,,$$

$$S = \sum_{i=0}^{n-1} \frac{1}{\sigma_i^2}\,, \qquad S_x = \sum_{i=0}^{n-1} \frac{x_i}{\sigma_i^2}\,, \qquad S_y = \sum_{i=0}^{n-1} \frac{y_i}{\sigma_i^2}$$

And goodness of fit :

$$\chi^2(a,b)/ndof = \frac{1}{n-2} \sum \left(\frac{y_i - a - bx_i}{\sigma_i}\right)^2$$

Using point-by-point errors

And goodness of fit :

$$\chi^2(a,b)/ndof = \frac{1}{n-2}\sum\left(\frac{y_i - a - bx_i}{\sigma_i}\right)^2$$

If our data really follow a linear distribution AND we have estimated our errors correctly, we should get chi2/ndf values ~1. Either way, we can use the chi2 and n to calculate a p-value indicating our goodness of fit

HW assignment!

In the HLS folder of the git repo there is a HW folder with data.txt. Each line is a new set of data. The format is:

x,y,sigma_y,last

Where sigma_y is the estimated uncertainty on that y value and "last" tells you if this was the last input for this event (there should be 6 events in the file).

Write HLS code to read those events in to memory from the file. Note that you don't know in advance how big each event is, and you shouldn't hard-code that. Try and optimize the code as much as you can.

For each event, print out a,b, the uncertainty on those two parameters and the chi2/ndf. Note that some of the events should fit very well to the model. Others might not! **Due in 1 week!**

More about your assignment

You don’t have to do editing in the Vitis GUI. It’s honestly not bad for code editing, but the X session make things run slow. So you can use your favorite other text-based editor to edit the code you’re writing! Jahred is partial to emacs. If you want to use vim, go ahead, but you are dead to him. Or use something else and more modern IDE environments if you can.

One thing to be careful about is how to store values. You should use the HLS types for integers and floats, but you need to think about how many bits are required. If you don’t use the right number of bits, you can get funny behavior such as overflowing integers that suddenly become negative!

Can recommend you start by writing the code and then trying to optimize it after that.

You should send Jahred and Rick a link to your git repo with your code. And in the repo include screenshots of the code output (ie fit parameters and errors and chi2/ndf) and also the resource utilization. You may want to send that resource utilization before and after code optimization.

Some help with your HW assignment

We have not really discussed in much detail how one streams data into and out of an FPGA (we're short on lecture time). So for now we are going to write HLS code, synthesize it (ie turn it into commands for logic gates and flip flops) and simulate the chip, which means we can just write output using std::cout, which you of course can't do in a real working environment. And we can just send the data in as c arrays! (Again, not in a real chip)

How do we write code we will synthesize? Make a c source file with a function you want to synthesize. It can take as inputs arrays of all the values mentioned on the previous slide. And put the function inside of an extern command like:

```
extern "C" {
void myFunction(…)
}
```

Add a header that declares your function in the same way

```
extern "C" {
void myFunction(…);
}
```

Some help with your HW assignment

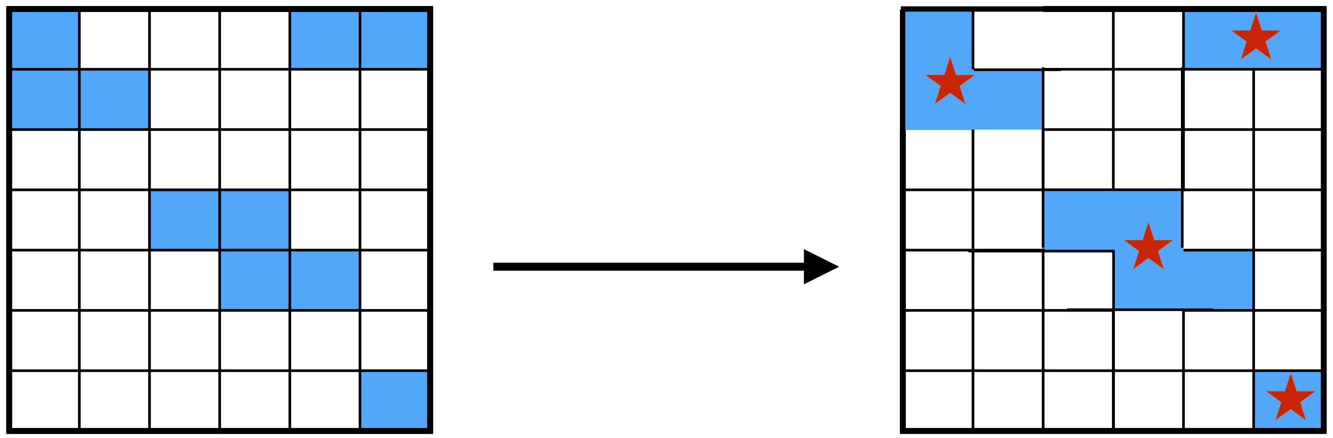
How do we simulate our function? Write a new c file with one function. You can call the function main(), but you don't need to call it that. Declare the function in a header you include in the source code. And in that header create the input arrays.

In the source code for the main function, call your synthesized code, passing the arrays as inputs! You can return the outputs (ie results of the fit) to the main function or just print them in the synthesized function itself, as you prefer

We have kindly created main.C and main.h for you to get you started. It's in the repository with these files

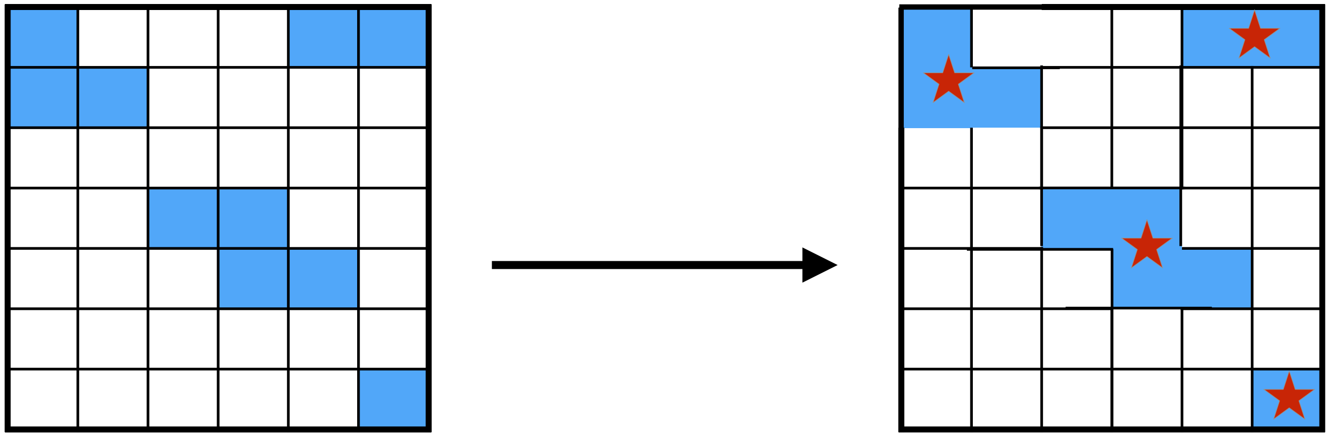
In vitis, you want to create a new project, selecting different source code for what you are trying to synthesize and what you are trying to run as simulation. When you have a working version,

Potential final project



Hayden worked you through his 1D silicon clustering algorithm. Now imagine we have data coming in from a 2D pixel silicon sensor. We want to cluster this data - hits that are adjacent to other hits should be combined into one cluster, and we want to calculate the centroid (★) of each cluster

Potential final project



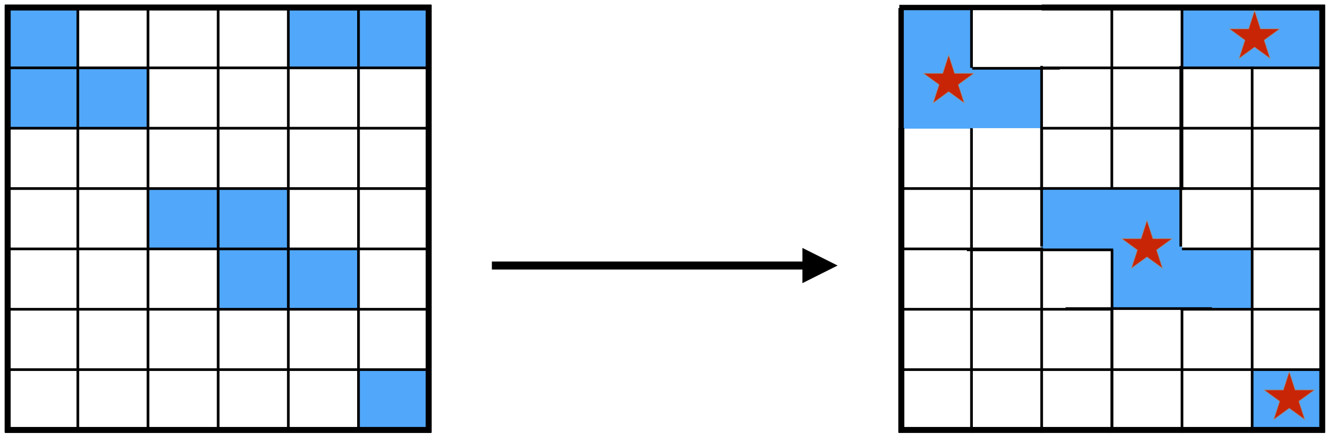
Note that the above algorithm is not trivial to implement. You should make up some interesting test data and visualize the input and output on several events. Make the grid big enough to big interesting, (ex: 100x100).

To make things more tractable, perhaps assume the hits arrive in some order (ie by column and then by row, or just column ordered).

Don't worry about data formats, to start you can just input hits by integer (row, column) and isLast.

As a bonus, you could calculate the centroids assuming hits can have some weighting factor based on something like time over threshold. Note that this is very related to (but not the same!) as calorimeter clustering

Potential final project



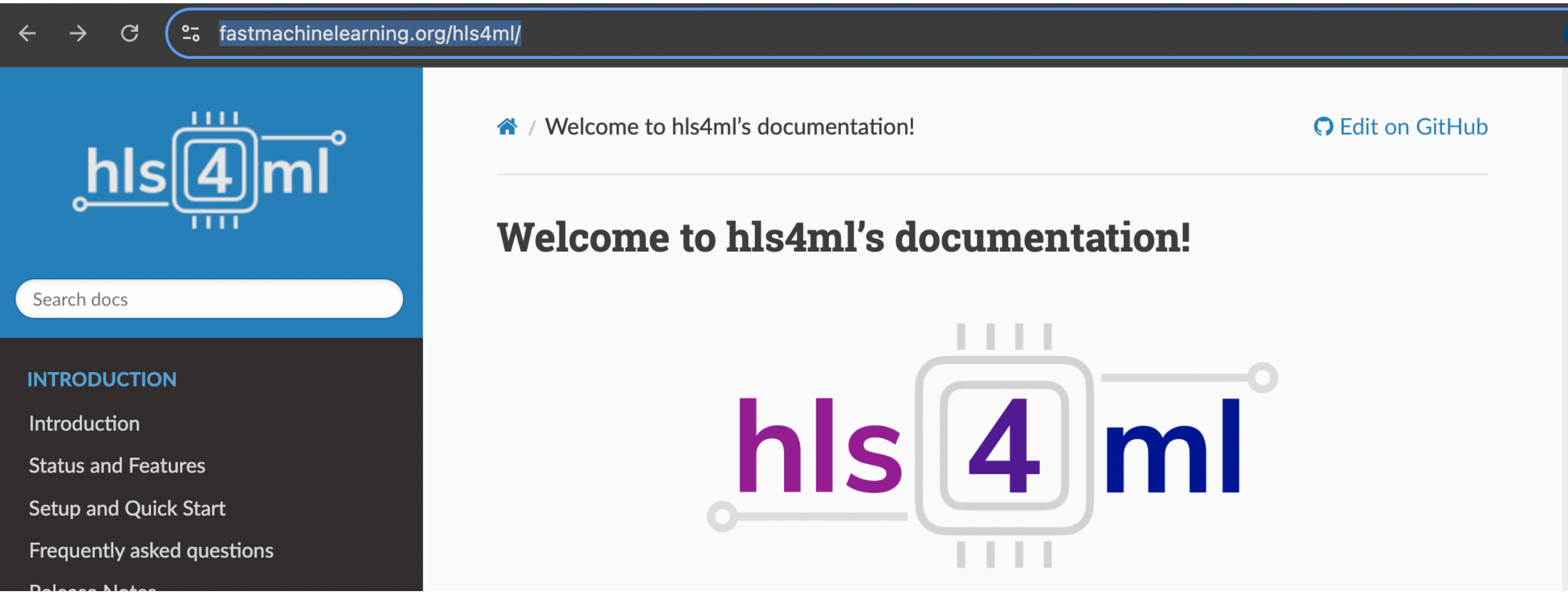
You might want to be careful about edge cases (literally the edges of the sensor/rows/columns) but also what happens if your clusters get too big! Do we want to truncate their size?

Note: Limiting the maximum size of the cluster will allow you to optimize the algorithm by clustering in semi-local regions, which might make the problem more tractable.

Perhaps one group member can investigate how to stream data into and out of the synthesized code/FPGA instead of passing arrays. Just an option, not required!

Think about how to optimize all of the above, we want to make this as efficient as possible.

Another potential final project (open-ended)


fastmachinelearning.org/hls4ml/

/ Welcome to hls4ml's documentation!

Welcome to hls4ml's documentation!

You will learn about how to access OpenData from LHC experiments in the future weeks. Can you skim that data to provide inputs to an FPGA ML algorithm that you design?

Use hls4ml, a package that Jovan talked about that connects ML tools to HLS code for FPGAs! You will have to dig a bit to figure out how to use this but it's part of the final project!

https://fastmachinelearning.org/hls4ml/