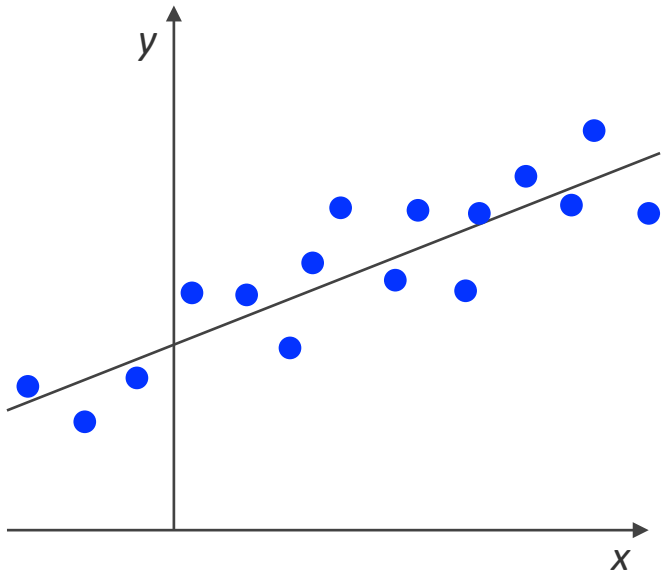
Maxeler Apps
Linear Regression

Dec 2014

Linear regression

- Statistical method that creates a linear model between a scalar response variable *y*, and one or more explanatory variables *x*.
- Goal: forecasting or regression for values of *y* for which no data is available (yet).
- Approach: given a dataset of *(x,y)* values, fit a linear function to the dataset.



- Regression model parameters can often be derived with standard estimation techniques such as *least squares*.
- Assumptions for using standard estimation models:
  - Underlying model is linear
  - Variables x are error-free
  - Constant variance of errors in response variables y
  - Errors in response variables y are uncorrelated

MAXELER Technologies

Simple linear regression

- Simple case: one explanatory variable *x*.
- Find best fit of a straight line with intercept *a* and slope *b* through data set: $y = a + b \cdot x$
- Least squares estimation of regression model, i.e. determine line such that sum of squared error terms is minimal:

$$\min_{a,b} \sum_{i=1}^{n} (y_i - a - b \cdot x_i)^2$$

- Solve to:

$$b = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n} \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} x_i \right)^2}$$

$$a = \bar{y} - b \cdot \bar{x} = \frac{1}{n} \left(\sum_{i=1}^{n} y_i - b \sum_{i=1}^{n} x_i \right)$$

MAXELER Technologies

A simple software implementation

```
void linearRegression(int dataPoints, float *x, float *y, float *a, float *b){
    int i;
    float sumX = 0;     // sum of x
    float sumX2 = 0;    // sum of x^2
    float sumXY = 0;    // sum of x*y
    float sumY = 0;     // sum of y

    for(i = 0; i < dataPoints; i++){
        sumX  +=  x[i];             //compute sums
        sumX2 +=  x[i]*x[i];
        sumXY +=  x[i]*y[i];
        sumY  +=  y[i];

        meanX = sumX / (i+1);   //compute mean values for x and y
        meanY = sumY / (i+1);

        b[i] = (sumXY - sumX * meanY) / (sumX2 - sumX * meanX); //compute b (slope)
        a[i] = meanY - b[i] * meanX;                            //compute a (intercept)
    }
}
```

Compute regression in a continuous fashion, i.e. compute a new *a*, *b* value pair for each added *x*, *y* data point.

Main computation in this loop, accelerate with DFE kernel.

MAXELER Technologies

Using the DFE kernel for regression

original C function:

```
void linearRegression(int dataPoints, float *x, float *y, float *a, float *b);
```

.max file defines engine interface

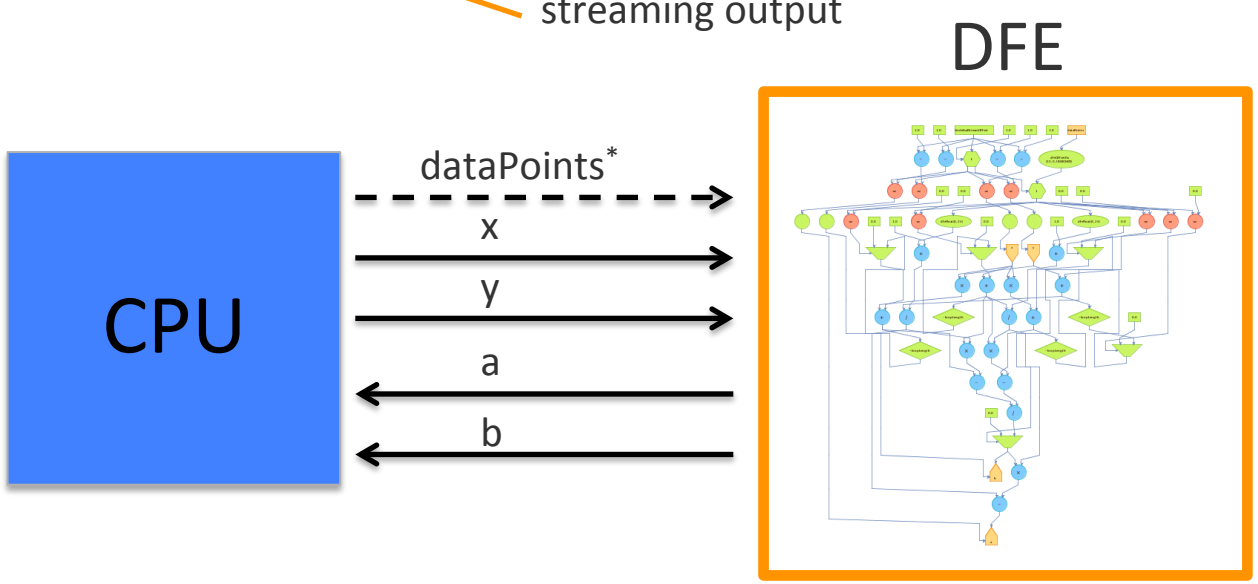
DFE engine interface:

scalar input*

streaming input

```
void linearRegression( int64_t dataPoints, const float *instream_x, const float *instream_y,
                       float *outstream_a, float *outstream_b );
```

streaming output



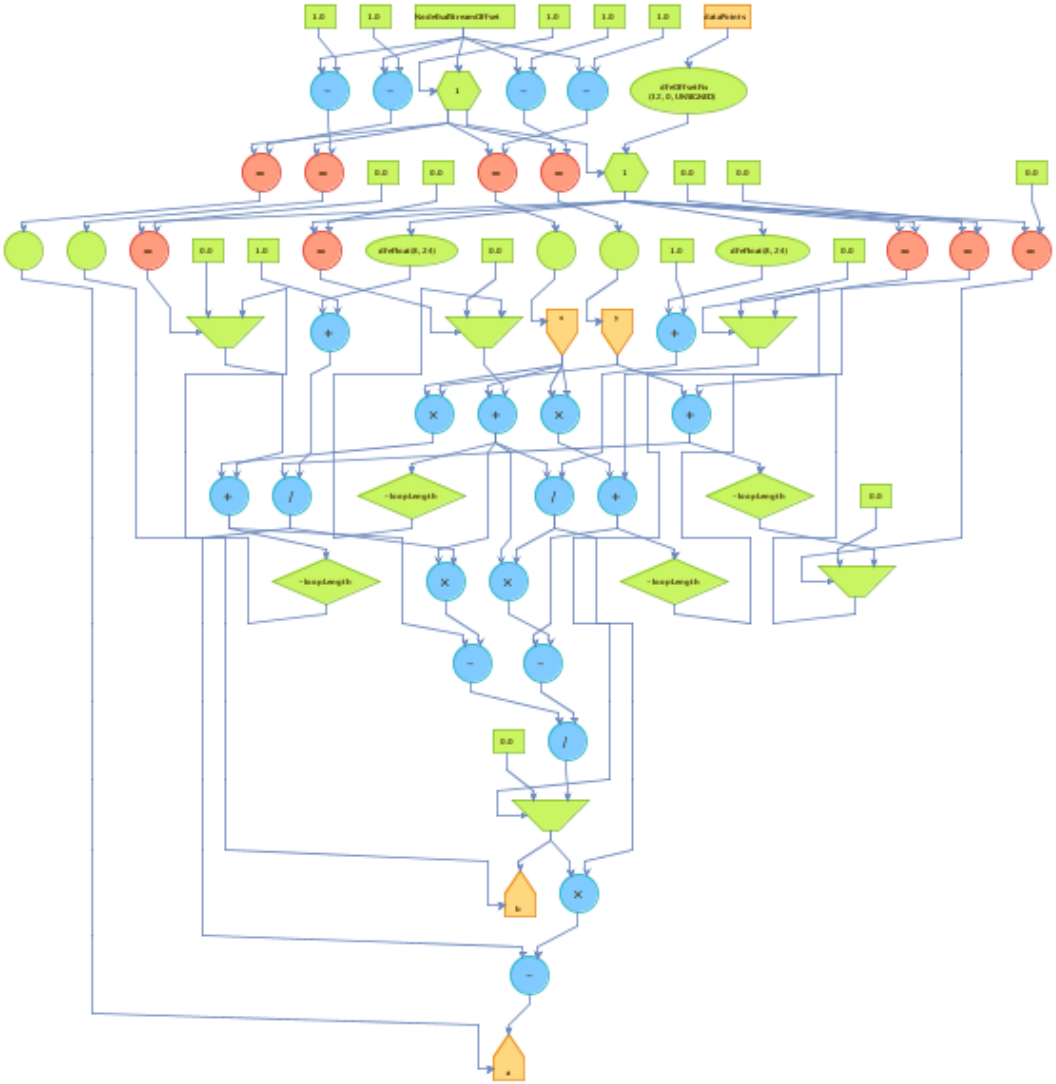
* scalar input is sent only once before stream processing starts

MAXELER Technologies

The simple regression DFE kernel

- Pure dataflow
- Concurrent execution of all operations
- Automatically pipelined

- Added over simple data flow diagram:
  - Scheduling of operations
  - Initialisation
  - Delay for feedback elemens (sums)

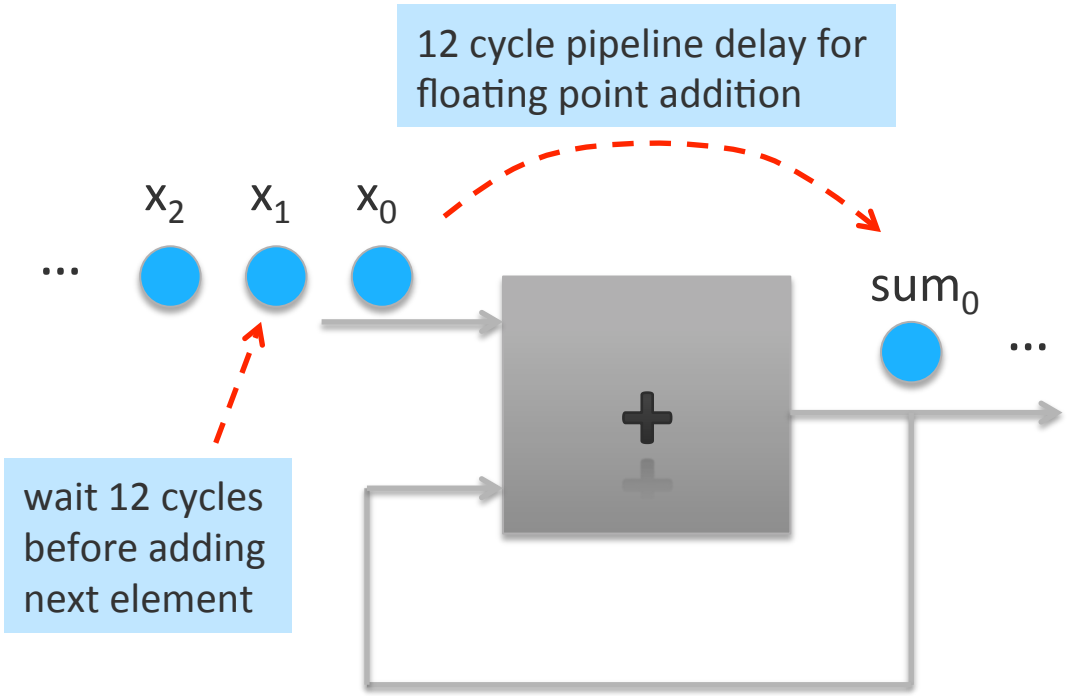


MAXELER Technologies

Feedback loops

- Performance critical aspect: feedback loops for carried sums

```
sumX += x[i];
```



- Solution: interleave multiple data streams to utilize idle cycles
  - Current DFE version does not interleave data
  - More information in MaxCompiler tutorial on Loops and Pipelines, section 10

MAXELER Technologies

Summary

- Create linear model for a set of data points
- Iterative processing accelerated by DFE
- Simple DFE kernel for demo purposes
  - Very small, single dataset to demonstrate principle

- Possible improvements:
  - Larger dataset (longer sequence of variables *x,y*)
  - Interleave multiple data streams (parallel processing of multiple regressions): utilize idle cycles of accumulators
  - Parallel processing of even more data streams: replicate pipeline and pack more operators into DFE

MAXELER Technologies