Digital Signal Processing using CUDA

1.0

Generated by Doxygen 1.8.6

Mon Jan 27 2014 11:45:43

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 DataReader Class Reference

Public Member Functions

- **DataReader** (const std::string &filename, InputBuffer *buffer)
- int **_checkFileHeader** ()
- void **readToBufferAsync** ()
- int **isReading** ()
- void **stopReading** ()
- int **get_nSamp** ()
- int **get_nSeg** ()
- int **get_nWf** ()

The documentation for this class was generated from the following file:

- /home/fabian/DSP/src/DataReader.h

3.2 fitData Struct Reference

Public Attributes

- float **param** [COUNTPARAM]
- float **startValue**
- float **endValue**
- float **extremumPos**
- float **extremumValue**
- int **status**

The documentation for this struct was generated from the following file:

- /home/fabian/DSP/src/Types.h

3.3 Node Class Reference

```
#include <Node.h>
```

Public Member Functions

- Node (int deviceIdentifier, InputBuffer *input, OutputBuffer *output)
  Basic constructor.

3.3.1 Detailed Description

Each installed device should be handled by its own thread. This class provides all functions to create a thread, copy data to and from the device and start the kernel on the device.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Node::Node (int *deviceIdentifier,* InputBuffer * *input,* OutputBuffer * *output*)

Basic constructor.

Stats a new Thread. The new Thread reads data from the input buffer, copies them to the gpu and copy the result back to the output buffer.

Parameters

deviceIdentifier	Number of the Device
input	Buffer which provides the raw input data.
output	Buffer which will be filled with the result data.

The documentation for this class was generated from the following file:

- /home/fabian/DSP/src/Node.h

3.4 OutputStream Class Reference

`#include <OutputStream.h>`

Public Member Functions

- OutputStream (const std::string &file, int producer)
  Basic constructor.
- Ringbuffer< Output > * getBuffer ()
  Returns a reference of the buffer.
- void join ()
  Waits until the writing thread to stops.

3.4.1 Detailed Description

Class that provides all functions to write the results of the computation into a file.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 OutputStream::OutputStream (const std::string & *file,* int *producer*)

Basic constructor.

Constructor opens a filestream, initialise the output buffer and start the thread, which takes elements from the buffers and writes them into the file.

Parameters

file	Filename of the output file.

The documentation for this class was generated from the following file:

- /home/fabian/DSP/src/OutputStream.h

3.5 Ringbuffer< Type > Class Template Reference

Public Member Functions

- **Ringbuffer** (unsigned int bSize, int producer)
- int **writeFromHost** (Type *inputOnHost)
- int **copyToHost** (Type *outputOnHost)
- Type * **reserveHead** ()
- int **freeHead** ()
- Type * **reserveTail** ()
- int **freeTail** ()
- int **getSize** ()
- bool **isEmpty** ()
- bool **isFinished** ()
- void **producerQuit** ()

The documentation for this class was generated from the following file:

- /home/fabian/DSP/src/Ringbuffer.h

Chapter 4

File Documentation

4.1 /home/fabian/DSP/src/Constants.h File Reference

This File holds all configurations and constants.

```
#include <string>
```

Variables

- const unsigned int SAMPLE_COUNT = 1000

  Number of samples per event.
- const unsigned int CHUNK_COUNT = 100

  Number of events copied to the GPU in one step.
- const unsigned int CHUNK_BUFFER_COUNT = 2048

  Number of chunks in the input buffer.
- const cudaTextureFilterMode FILTER_MODE = cudaFilterModeLinear

  Interpolation mode.
- const std::string **OUTPUT_FILENAME** = "results.txt"
- const std::string **FILENAME_TESTFILE** = "../data/Al_25keV-259.cdb"
- const unsigned int **SAMPLE_COUNT_TESTFILE** = 1000
- const unsigned int **SEGMENT_COUNT_TESTFILE** = 1
- const unsigned int **WAVEFORM_COUNT_TESTFILE** = 100000
- const unsigned int MAXCOUNTDATA = 1000

  max. number of samples per event for compute capability 2.0 or higher - currently ca. 2450 is max. because (COUNTPARAM + 2) ∗ MAXCOUNTDATA ∗ sizeof(float) = 48 kB (= max. shared memory); for compute capability 1.x - currently ca. 800 is max. because (COUNTPARAM + 2) ∗ MAXCOUNTDATA ∗ sizeof(float) = 16 kB (= max. shared memory)
- const unsigned int MAXCALL = 100

  max. calls for Levenberg Marquardt until stops
- const float FITVALUETHRESHOLD = 0.0

  threshold between min (0.0) and max (1.0) value to define the data using interval to calculate the fit function
- const float STARTENDPROPORTION = 0.01

  proportion of countData for calculating the average of start/end value (e. g. 0.1 means average of the first 10% of data for start value and the last 10% for end value)
- const unsigned int COUNTPARAM = 3

  number of parameters for the fit function

4.1.1 Detailed Description

This File holds all configurations and constants.

4.2 /home/fabian/DSP/src/LevMarq.h File Reference

```
#include <stdlib.h>
#include <math.h>
#include <float.h>
#include <stdio.h>
#include "Types.h"
#include "LevMarq.h"
```

Macros

- #define **PARAMSTARTVALUE** { 1, 1, 1 }
- #define **CUDA**
- #define **GETSAMPLE**(I, INDEXDATASET) tex2D(dataTexture, (I) + 0.5, (INDEXDATASET) + 0.5)
- #define **GLOBAL** __global__
- #define **DEVICE** __device__
- #define **SHARED** __shared__
- #define **LM_MACHEP** FLT_EPSILON
- #define **LM_DWARF** FLT_MIN
- #define **LM_SQRT_DWARF** sqrt(FLT_MIN)
- #define **LM_SQRT_GIANT** sqrt(FLT_MAX)
- #define **LM_USERTOL** 30∗LM_MACHEP
- #define **MIN**(A, B) (((A) <= (B)) ? (A) : (B))
- #define **MAX**(A, B) (((A) >= (B)) ? (A) : (B))
- #define **SQR**(X) ((X) ∗ (X))

Functions

- DEVICE void fitFunction (float x, float ∗param, float ∗y)

  fitFunction returns the y of a given x
- DEVICE void fitFunctionExtremum (float ∗param, float ∗x)

  fitFunctionExtremum returns the x of the min. or max. y value
- DEVICE void **evaluate** (float ∗param, int countData, float ∗fvec, int indexDataset, int xOffset)
- DEVICE void **qrSolve** (int n, float ∗r, int ldr, int ∗ipvt, float ∗diag, float ∗qtb, float ∗x, float ∗sdiag, float ∗wa)
- DEVICE void euclidNorm (int n, float ∗x, float ∗result)
- DEVICE void **lmpar** (int n, float ∗r, int ldr, int ∗ipvt, float ∗diag, float ∗qtb, float delta, float ∗par, float ∗x, float ∗sdiag, float ∗wa1, float ∗wa2)
- DEVICE void **qrFactorization** (int m, int n, float ∗a, int pivot, int ∗ipvt, float ∗rdiag, float ∗acnorm, float ∗wa)
- DEVICE void **lmdif** (int m, int n, float ∗x, float ∗fvec, float ftol, float xtol, float gtol, int maxfev, float epsfcn, float ∗diag, int mode, float factor, int ∗info, int ∗nfev, float ∗fjac, int ∗ipvt, float ∗qtf, float ∗wa1, float ∗wa2, float ∗wa3, float ∗wa4, int indexDataset, int xOffset)
- DEVICE void maxValue (int countData, int indexDataset, int ∗x, DATATYPE ∗y)

  maxValue returns the x and y where y has the greatest value
- DEVICE void averageValue (int start, int count, int indexDataset, float ∗y)

  averageValue returns the average of all y values in a given range
- DEVICE void xOfValue (int countData, int indexDataset, char fromDirection, DATATYPE minValue, int ∗x)

  xOfValue returns the first x of a value y that is greater or equal of a given min. value

- GLOBAL void kernel (int countData, struct fitData *result)

  kernel is the start method for calculation (you have to set the dataTexture (GPU mode) or data variable (CPU mode) before calling this method)

Variables

- texture< Precision,
  2, cudaReadModeElementType > **dataTexture**
- const char * **statusMessage** []

4.2.1 Function Documentation

4.2.1.1 DEVICE void averageValue (int *start*, int *count*, int *indexDataset*, float * *y*)

averageValue returns the average of all y values in a given range

Parameters

start	first x for average calculation
count	number of values for average calculation
indexDataset	index of the current dataset (GPU mode) or not used (CPU mode)
y	the returned average

4.2.1.2 DEVICE void euclidNorm (int *n*, float * *x*, float * *result*)

calculation of norm

4.2.1.3 DEVICE void fitFunction (float *x*, float * *param*, float * *y*) [inline]

fitFunction returns the y of a given x

Parameters

x	given x value to calculate y
param	parameters to define the concrete current fit-function
y	the returned y value

4.2.1.4 DEVICE void fitFunctionExtremum (float * *param*, float * *x*) [inline]

fitFunctionExtremum returns the x of the min. or max. y value

Parameters

param	parameters to define the concrete current fit-function
x	the returned x value

4.2.1.5 GLOBAL void kernel (int *countData*, struct fitData * *result*)

kernel is the start method for calculation (you have to set the dataTexture (GPU mode) or data variable (CPU mode) before calling this method)

Parameters

countData	number of samples
result	fit-function and other parameters, defined in fitData struct

4.2.1.6 DEVICE void maxValue (int *countData*, int *indexDataset*, int * *x*, DATATYPE * *y*)

maxValue returns the x and y where y has the greatest value

Parameters

countData	number of samples
indexDataset	index of the current dataset (GPU mode) or not used (CPU mode)
x	the returned x value
y	the returned y value

4.2.1.7 DEVICE void xOfValue (int *countData*, int *indexDataset*, char *fromDirection*, DATATYPE *minValue*, int * *x*)

xOfValue returns the first x of a value y that is greater or equal of a given min. value

Parameters

countData	number of samples
indexDataset	index of the current dataset (GPU mode) or not used (CPU mode)
fromDirection	
minValue	min. y value
x	the returned x value, -1 if there is no x with a y greater or equal minValue

4.2.2 Variable Documentation

4.2.2.1 const char* statusMessage[ ]

Initial value:

```
= {
    "fatal coding error (improper input parameters)",
    "success (the relative error in the sum of squares is at most tol)",
    "success (the relative error between x and the solution is at most tol)",
    "success (the relative errors in the sum of squares and between x and the solution are at most tol)",
    "trapped by degeneracy (fvec is orthogonal to the columns of the jacobian)",
    "timeout (number of calls to fcn has reached maxcall*(n+1))",
    "failure (ftol<tol: cannot reduce sum of squares any further)",
    "failure (xtol<tol: cannot improve approximate solution any further)",
    "failure (gtol<tol: cannot improve approximate solution any further)"
}
```

Index