HCFFT Documentation

Release

Aparna Suresh

May 30, 2017

Contents

1 1. Introduction **1**
1.1 1. Getting Started 1
1.2 2. hcFFT API Reference 6

CHAPTER 1

1. Introduction

The hcFFT library is an implementation of FFT (Fast Fourier Transform) targetting the AMD heterogenous hardware via HCC compiler runtime. The computational resources of underlying AMD heterogenous compute gets exposed and exploited through the HCC C++ frontend. Refer here for more details on HCC compiler.

To use the hcFFT API, the application must allocate the required input buffers in the GPU memory space, fill them with data, call the sequence of desired hcFFT functions, and then upload the results from the GPU memory space back to the host. The hcFFT API also provides helper functions for writing and retrieving data from the GPU.

The following list enumerates the current set of FFT sub-routines that are supported so far.

- R2C : Single Precision real to complex valued Fast Fourier Transform
- C2R : Single Precision complex to real valued Fast Fourier Transform
- C2C : Single Precision complex to complex valued Fast Fourier Transform
- D2Z : Double Precision real to complex valued Fast Fourier Transform
- Z2D : Double Precision complex to real valued Fast Fourier Transform
- Z2Z : Double Precision complex to complex valued Fast Fourier Transform

1. Getting Started

1.1. Introduction

The hcFFT library is an implementation of FFT (Fast Fourier Transform) targetting the AMD heterogenous hardware via HCC compiler runtime. The computational resources of underlying AMD heterogenous compute gets exposed and exploited through the HCC C++ frontend. Refer here for more details on HCC compiler.

To use the hcFFT API, the application must allocate the required input buffers in the GPU memory space, fill them with data, call the sequence of desired hcFFT functions, and then upload the results from the GPU memory space back to the host. The hcFFT API also provides helper functions for writing and retrieving data from the GPU.

The following list enumerates the current set of FFT sub-routines that are supported so far.

- R2C : Single Precision real to complex valued Fast Fourier Transform
- C2R : Single Precision complex to real valued Fast Fourier Transform
- C2C : Single Precision complex to complex valued Fast Fourier Transform
- D2Z : Double Precision real to complex valued Fast Fourier Transform
- Z2D : Double Precision complex to real valued Fast Fourier Transform
- Z2Z : Double Precision complex to complex valued Fast Fourier Transform

1.2. Prerequisites

This section lists the known set of hardware and software requirements to build this library

1.2.1. Hardware

- CPU: mainstream brand, Better if with >=4 Cores Intel Haswell based CPU
- System Memory >= 4GB (Better if >10GB for NN application over multiple GPUs)
- Hard Drive > 200GB (Better if SSD or NVMe driver for NN application over multiple GPUs)
- Minimum GPU Memory (Global) > 2GB

1.2.2. GPU cards supported

- dGPU: AMD R9 Fury X, R9 Fury, R9 Nano
- APU: AMD Kaveri or Carrizo

1.2.3 AMD Driver and Runtime

- Radeon Open Compute Kernel (ROCK) driver : https://github.com/RadeonOpenCompute/ROCK-Kernel-Driver
- HSA runtime API and runtime for Boltzmann: https://github.com/RadeonOpenCompute/ROCR-Runtime

1.2.4. System software

- Ubuntu 14.04 trusty
- GCC 4.6 and later
- CPP 4.6 and later (come with GCC package)
- python 2.7 and later
- HCC 0.9 from here

1.2.5. Tools and Misc

- git 1.9 and later
- cmake 2.6 and later (2.6 and 2.8 are tested)
- firewall off
- root privilege or user account in sudo group

1.2.6. Ubuntu Packages

- libc6-dev-i386
- liblapack-dev
- graphicsmagick

1.3. Tested Environments

This sections enumerates the list of tested combinations of Hardware and system softwares.

1.3.1. Driver versions

- Boltzmann Early Release Driver + dGPU
  - Radeon Open Compute Kernel (ROCK) driver : https://github.com/RadeonOpenCompute/ROCK-Kernel-Driver
  - HSA runtime API and runtime for Boltzmann: https://github.com/RadeonOpenCompute/ROCR-Runtime
- Traditional HSA driver + APU (Kaveri)

1.3.2. GPU Cards

- Radeon R9 Nano
- Radeon R9 FuryX
- Radeon R9 Fury
- Kaveri and Carizo APU

1.3.3. Desktop System ^^^^^666^^^^^^^^^^^^^

- Supermicro SYS-7048GR-TR Tower 4 W9100 GPU
- ASUS X99-E WS motherboard with 4 AMD FirePro W9100
- Gigabyte GA-X79S 2 AMD FirePro W9100 GPU's

1.3.4. Server System

- Supermicro SYS 2028GR-THT 6 R9 NANO
- Supermicro SYS-1028GQ-TRT 4 R9 NANO
- Supermicro SYS-7048GR-TR Tower 4 R9 NANO

1.4. Installation steps

The following are the steps to use the library

- ROCM 1.0 Kernel, Driver and Compiler Installation (if not done until now)
- Library installation.

1.4.1. ROCM 1.0 Installation

To Know more about ROCM refer https://github.com/RadeonOpenCompute/ROCm/blob/master/README.md

a. Installing Debian ROCM repositories

Before proceeding, make sure to completely uninstall any pre-release ROCm packages.

Refer https://github.com/RadeonOpenCompute/ROCm#removing-pre-release-packages for instructions to remove pre-release ROCM packages.

Steps to install rocm package are,

```
wget -qO - http://packages.amd.com/rocm/apt/debian/rocm.gpg.key |
sudo apt-key add -
```

```
sudo sh -c 'echo deb [arch=amd64] http://packages.amd.com/rocm/apt/
debian/ trusty main > /etc/apt/sources.list.d/rocm.list'
```

```
sudo apt-get update
```

```
sudo apt-get install rocm
```

and Reboot the system

b. Verifying the Installation

Once Reboot, to verify that the ROCm stack completed successfully you can execute HSA vector_copy sample application:

- cd /opt/rocm/hsa/sample
- make
- ./vector_copy

1.4.2. Library Installation

1. Install using Prebuilt debian

```
wget https://bitbucket.org/multicoreware/hcfft/downloads/
hcfft-master-9607fd5-Linux.deb
```

```
sudo dpkg -i hcfft-master-9607fd5-Linux.deb
```

2. Build debian from source

```
git clone https://bitbucket.org/multicoreware/hcfft.git && cd
hcfft
```

```
chmod +x build.sh && ./build.sh
```

To use LC Compiler,

```
export HCC_HOME=1
```

build.sh execution builds the library and generates a debian under build directory.

1.4.3. Library UnInstallation

1. To uninstall the library, do

```
chmod +x clean.sh && ./clean.sh
```

1.5. Unit testing

1.5.1. Testing hcFFT against clFFT:

- Please refer to https://github.com/clMathLibraries/clFFT for clFFT Prerequisites/Installation/Pre-built binaries

1. Install clFFT library:

```
git clone https://github.com/clMathLibraries/clFFT.git && cd clFFT
```

```
mkdir build && cd build
```

```
cmake ../src
```

```
make && sudo make install
```

```
export AMDAPPSDKROOT=/opt/AMDAPPSDK-x.y.z/
```

2. Set Variables:

```
export CLFFT_LIBRARY_PATH=/path/to/clFFT/build/library/
```

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/hcFFT/
library[libhcfft.so]
```

3. Automated testing:

```
cd ~/hcfft/
```

```
./build.sh --test=on
```

4. Manual testing:

```
cd ~/hcfft/build/test/src/bin/
```

choose the appropriate named binary

Here are some notes for performing manual testing:

N1 and N2 - Input sizes of 2D FFT

- FFT

```
./fft N1 N2
```

1.6. Benchmarking

1.6.1. Benchmarking hcFFT:

1. Set Variables:

```
export CLFFT_LIBRARY_PATH=/path/to/clFFT/build/library/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/hcFFT/
library[libhcfft.so]
```

2. Automated Benchmarking:

```
cd ~/hcfft/
./build.sh --test=on --bench=on
```

3. Manual Benchmarking:

```
cd ~/hcfft/test/FFT_benchmark_Convolution_Networks
./runme_chronotimer.sh
```

It profiles hcFFT for input sizes in Input.txt and stores the output in Benchmark_fft.csv

1.6.2. Benchmarking hcFFT against clFFT:

1. Set Variables:

```
export CLFFT_LIBRARY_PATH=/path/to/clFFT/build/library/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/hcFFT/
library[libhcfft.so]
```

2. Manual Benchmarking:

```
cd ~/hcfft/test/FFT_benchmark_Convolution_Networks/
Comparison_tests
./run_clfftVShcfft.sh
```

It compares clFFT with hcFFT for input sizes in Input.txt and stores the output in Benchmark_clfftvshcfft.csv

2. hcFFT API Reference

This section provides a brief description of APIs and helper routines hosted by the library

HCFFT TYPES

Enumerations

enum hcfftResult_t {
HCFFT_SUCCESS = 0,
HCFFT_INVALID_PLAN = 1,
HCFFT_ALLOC_FAILED = 2,

HCFFT_INVALID_TYPE = 3,
HCFFT_INVALID_VALUE = 4,
HCFFT_INTERNAL_ERROR = 5,
HCFFT_EXEC_FAILED = 6,
HCFFT_SETUP_FAILED = 7,
HCFFT_INVALID_SIZE = 8,
HCFFT_UNALIGNED_DATA = 9,
HCFFT_INCOMPLETE_PARAMETER_LIST = 10,
HCFFT_INVALID_DEVICE = 11,
HCFFT_PARSE_ERROR = 12,
HCFFT_NO_WORKSPACE = 13
}
enum hcfftType_t {
HCFFT_R2C = 0x2a,
HCFFT_C2R = 0x2c,
HCFFT_C2C = 0x29,
HCFFT_D2Z = 0x6a,
HCFFT_Z2D = 0x6c,
HCFFT_Z2Z = 0x69
}

typedef float hcfftReal;
typedef float_2 hcfftComplex;
typedef double hcfftDoubleReal;
typedef double_2 hcfftDoubleComplex;

Detailed Description

HCFFT RESULT (hcfftResult_t)

This enumeration is the set of HCFFT error codes.

Enumerator	
HCFFT_SUCCESS	The hcFFT operation was successful.
HCFFT_INVALID_PLAN	hcFFT was passed an invalid plan handle.
HCFFT_ALLOC_FAILED	hcFFT failed to allocate GPU or CPU memory.
HCFFT_INVALID_TYPE	unsupported numerical value was passed to function.
HCFFT_INVALID_VALUE	User specified an invalid pointer or parameter.
HCFFT_INTERNAL_ERROR	Driver or internal hcFFT library error.
HCFFT_EXEC_FAILED	Failed to execute an FFT on the GPU.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.
HCFFT_INVALID_SIZE	User specified an invalid transform size.
HCFFT_UNALIGNED_DATA	No longer used.
HCFFT_INCOMPLETE_PARAMETER_LIST	Missing parameters in call.
HCFFT_INVALID_DEVICE	Execution of a plan was on different GPU than plan creation.
HCFFT_PARSE_ERROR	Internal plan database error.
HCFFT_NO_WORKSPACE	No workspace has been provided prior to plan execution.

HCFFT TYPE (hcfftType_t)

types of transform data supported by hcFFT

Enumerator	
HCFFT_R2C	Real to complex (interleaved).
HCFFT_C2R	Complex (interleaved) to real.
HCFFT_C2C	Complex to complex (interleaved).
HCFFT_D2Z	Double to double-complex (interleaved).
HCFFT_Z2D	Double-complex (interleaved) to double.
HCFFT_Z2Z	Double-complex to double-complex (interleaved).

HCFFT Helper functions

1. hcfftCreate()

*hcfftResult hcfftCreate(hcfftHandle *&plan)*

This function Creates only an opaque handle, and allocates small data structures on the host.

2. hcfftDestory()

hcfftResult **hcfftDestroy** (hcfftHandle plan)

This function frees all GPU resources associated with a hcFFT plan and destroys the internal plan data structure.
This function should be called once a plan is no longer needed, to avoid wasting GPU memory.

Return Values,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully destroyed the FFT plan.
HCFFT_INVALID_PLAN	The plan parameter is not a valid handle.

3. hcfftPlan1d()

hcfftResult **hcfftPlan1d** (hcfftHandle *&plan, int nx, hcfftType type)

This function Creates a 1D FFT plan configuration for a specified signal size and data type.
The batch input parameter tells hcFFT how many 1D transforms to configure.

Return Values,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully created the FFT plan.
HCFFT_ALLOC_FAILED	The allocation of GPU resources for the plan failed.
HCFFT_INVALID_VALUE	One or more invalid parameters were passed to the API.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.
HCFFT_INVALID_SIZE	Either or both of the nx or ny parameters is not a supported sizek.

4. hcfftPlan2d()

hcfftResult **hcfftPlan2d** (hcfftHandle *&plan, int nx, int ny, hcfftType type)

This function Creates a 2D FFT plan configuration according to specified signal sizes and data type.

Return Values,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully created the FFT plan.
HCFFT_ALLOC_FAILED	The allocation of GPU resources for the plan failed.
HCFFT_INVALID_VALUE	One or more invalid parameters were passed to the API.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.
HCFFT_INVALID_SIZE	Either or both of the nx or ny parameters is not a supported sizek.

5. hcfftPlan3d()

hcfftResult **hcfftPlan3d** (hcfftHandle *&plan, int nx, int ny, int nz, hcfftType type)

This function Creates a 3D FFT plan configuration according to specified signal sizes and data type.
This function is the same as hcfftPlan2d() except that it takes a third size parameter nz.

Return Values,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully created the FFT plan.
HCFFT_ALLOC_FAILED	The allocation of GPU resources for the plan failed.
HCFFT_INVALID_VALUE	One or more invalid parameters were passed to the API.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.
HCFFT_INVALID_SIZE	Either or both of the nx or ny parameters is not a supported sizek.

6. hcfftXtSetGPUs()

hcfftResult **hcfftXtSetGPUs** (accelerator &acc)

This function returns GPUs to be used with the plan

Return Values,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully created the FFT plan.
HCFFT_ALLOC_FAILED	The allocation of GPU resources for the plan failed.
HCFFT_INVALID_VALUE	One or more invalid parameters were passed to the API.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.
HCFFT_INVALID_SIZE	Either or both of the nx or ny parameters is not a supported sizek.

Modules

R2C

Single precision real to complex valued transform.

hcfftExecR2C() executes a single-precision real-to-complex, implicitly forward, hcFFT transform plan.
hcFFT uses as input data the GPU memory pointed to by the idata parameter. This function stores
the nonredundant Fourier coefficients in the odata array. Pointers to idata and odata are both required to be aligned
to hcfftComplex data type in single-precision transforms and hcfftDoubleComplex data type in double-precision transforms.
It does an out-of-place transform.

Functions

Function Prototype:

Note: Inputs and Outputs are HCC device pointers.

hcfftResult **hcfftExecR2C** (hcfftHandle plan, hcfftReal *idata, hcfftComplex *odata)

Detailed Description

Function Documentation

```
hcfftResult hcfftExecR2C(hcfftHandle plan, hcfftReal *idata, hcfftComplex *odata)
```

In/out	Parameters	Description
[in]	plan	hcfftHandle returned by hcfftCreate.
[in]	idata	Pointer to the single-precision real input data (in GPU memory) to transform.
[out]	odata	Pointer to the single-precision complex output data (in GPU memory).

Returns,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully executed the FFT plan.
HCFFT_INVALID_PLAN	The plan parameter is not a valid handle.
HCFFT_INVALID_VALUE	At least one of the parameters idata and odata is not valid.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_EXEC_FAILED	hcFFT failed to execute the transform on the GPU.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.

C2R

Single precision complex to real valued transform.

hcfftExecC2R() executes a single-precision complex-to-real, implicitly inverse, hcFFT transform plan. hcFFT uses as input data the GPU memory pointed to by the idata parameter. The input array holds only the nonredundant complex Fourier coefficients. This function stores the real output values in the odata array. and pointers are both required to be aligned to hcfftComplex data type in single-precision transforms and hcfftDoubleComplex type in double-precision transforms. It does an out-of-place data transform.

Functions

Function Prototype:

Note: Inputs and Outputs are HCC device pointers.

hcfftResult **hcfftExecC2R** (hcfftHandle plan, hcfftComplex *idata, hcfftReal *odata)

Detailed Description

Function Documentation

```
hcfftResult hcfftExecC2R(hcfftHandle plan, hcfftComplex *idata, hcfftReal *odata)
```

In/out	Parameters	Description
[in]	plan	hcfftHandle returned by hcfftCreate.
[in]	idata	Pointer to the single-precision complex input data (in GPU memory) to transform
[out]	odata	Pointer to the single-precision real output data (in GPU memory).

Returns,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully executed the FFT plan.
HCFFT_INVALID_PLAN	The plan parameter is not a valid handle.
HCFFT_INVALID_VALUE	At least one of the parameters idata and odata is not valid.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_EXEC_FAILED	hcFFT failed to execute the transform on the GPU.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.

C2C

Single precision complex to complex valued transform.

hcfftExecC2C() executes a single-precision complex-to-complex transform plan in the transform direction as specified by direction parameter. hcFFT uses the GPU memory pointed to by the idata parameter as input data. This function stores the Fourier coefficients in the odata array.
It does an out-of-place data transform in the forward or backward direction.

Functions

Function Prototype:

> **Note: Inputs and Outputs are HCC device pointers.**

hcfftResult **hcfftExecC2C** (hcfftHandle plan, hcfftComplex *idata, hcfftComplex *odata, int direction)

Detailed Description

Function Documentation

```
hcfftResult hcfftExecC2C(hcfftHandle plan, hcfftComplex *idata, hcfftComplex *odata,␣
↪int direction)
```

In/out	Parameters	Description
[in]	plan	hcfftHandle returned by hcfftCreate.
[in]	idata	Pointer to the single-precision complex input data (in GPU memory) to transform.
[out]	odata	Pointer to the single-precision complex output data (in GPU memory).
[in]	direction	The transform direction: HCFFT_FORWARD or HCFFT_BACKWARD.

Returns,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully executed the FFT plan.
HCFFT_INVALID_PLAN	The plan parameter is not a valid handle.
HCFFT_INVALID_VALUE	At least one of the parameters idata and odata is not valid.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_EXEC_FAILED	hcFFT failed to execute the transform on the GPU.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.

D2Z

Double precision real to complex valued transform.

hcfftExecD2Z() executes a double-precision real-to-complex, implicitly forward, hcFFT transform plan. hcFFT uses as input data the GPU memory pointed to by the idata parameter. This function stores the nonredundant Fourier coefficients in the odata array. Pointers to idata and odata are both required to be aligned to hcfftComplex data type in single-precision transforms and hcfftDoubleComplex data type in double-precision transforms.
It does an out-of-place transform.

Functions

Function Prototype:

Note: Inputs and Outputs are HCC device pointers.

hcfftResult **hcfftExecD2Z** (hcfftHandle plan, hcfftDoubleReal *idata, hcfftDoubleComplex *odata)

Detailed Description

Function Documentation

```
hcfftResult hcfftExecD2Z (hcfftHandle plan, hcfftDoubleReal *idata,␣
↪hcfftDoubleComplex *odata)
```

In/out	Parameters	Description
[in]	plan	hcfftHandle returned by hcfftCreate.
[in]	idata	Pointer to the double-precision real input data (in GPU memory) to transform.
[out]	odata	Pointer to the double-precision complex output data (in GPU memory).

Returns,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully executed the FFT plan.
HCFFT_INVALID_PLAN	The plan parameter is not a valid handle.
HCFFT_INVALID_VALUE	At least one of the parameters idata and odata is not valid.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_EXEC_FAILED	hcFFT failed to execute the transform on the GPU.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.

Z2D

Double precision complex to real valued transform.

hcfftExecZ2D() executes a double-precision complex-to-real, implicitly inverse, hcFFT transform plan. hcFFT uses as input data the GPU memory pointed to by the idata parameter. The input array holds only the nonredundant complex Fourier coefficients. This function stores the real output values in the odata array. and pointers are both required to be aligned to hcfftComplex data type in single-precision transforms and hcfftDoubleComplex type in double-precision transforms. It does an out-of-place data transform.

Functions

Function Prototype:

Note: Inputs and Outputs are HCC device pointers.

hcfftResult **hcfftExecZ2D** (hcfftHandle plan, hcfftDoubleComplex *idata, hcfftDoubleReal *odata)

Detailed Description

Function Documentation

```
hcfftResult hcfftExecZ2D(hcfftHandle plan, hcfftDoubleComplex *idata, hcfftDoubleReal␣
↪*odata)
```

In/out	Parameters	Description
[in]	plan	hcfftHandle returned by hcfftCreate.
[in]	idata	Pointer to the double-precision complex input data (in GPU memory) to transform
[out]	odata	Pointer to the double-precision real output data (in GPU memory).

Returns,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully executed the FFT plan.
HCFFT_INVALID_PLAN	The plan parameter is not a valid handle.
HCFFT_INVALID_VALUE	At least one of the parameters idata and odata is not valid.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_EXEC_FAILED	hcFFT failed to execute the transform on the GPU.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.

Z2Z

Double precision complex to complex valued transform.

hcfftExecC2C() executes a double-precision complex-to-complex transform plan in the transform direction as specified by direction parameter. hcFFT uses the GPU memory pointed to by the idata parameter as input data. This function stores the Fourier coefficients in the odata array.
It does an out-of-place data transform in the forward or backward direction.

Functions

Function Prototype:

> **Note: Inputs and Outputs are HCC device pointers.**

hcfftResult **hcfftExecZ2Z** (hcfftHandle plan, hcfftDoubleComplex *idata, hcfftDoubleComplex *odata, int direction)

Detailed Description

Function Documentation

```
hcfftResult hcfftExecZ2Z(hcfftHandle plan, hcfftDoubleComplex *idata,␣
↪hcfftDoubleComplex *odata, int direction)
```

In/out	Parameters	Description
[in]	plan	hcfftHandle returned by hcfftCreate.
[in]	idata	Pointer to the double-precision complex input data (in GPU memory) to transform.
[out]	odata	Pointer to the double-precision complex output data (in GPU memory).
[in]	direction	The transform direction: HCFFT_FORWARD or HCFFT_BACKWARD.

Returns,

STATUS	DESCRIPTION
HCFFT_SUCCESS	hcFFT successfully executed the FFT plan.
HCFFT_INVALID_PLAN	The plan parameter is not a valid handle.
HCFFT_INVALID_VALUE	At least one of the parameters idata and odata is not valid.
HCFFT_INTERNAL_ERROR	An internal driver error was detected.
HCFFT_EXEC_FAILED	hcFFT failed to execute the transform on the GPU.
HCFFT_SETUP_FAILED	The hcFFT library failed to initialize.

Index

C

C2C, 16
C2R, 16

D

D2Z, 16

H

HCFFT_TYPES, 16

R

R2C, 16

Z

Z2D, 16
Z2Z, 16