**SUPRA on oneAPI**

**Getting Started Guide**

***Sept 2020***

**Legal Notices and Disclaimers**

Intel technologies may require enabled hardware, software or service activation.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications.  Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

© Intel Corporation.  Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries.  Other names and brands may be claimed as the property of others.

***Contents***

[1.0 Introduction 6](#_Toc51678744)

[1.1 How to use this document 7](#_Toc51678745)

[1.2 Terminology 7](#_Toc51678746)

[2.0 Host System Setup 9](#_Toc51678747)

[2.1 Host Development System 9](#_Toc51678748)

[2.2 Install Basic Packages 9](#_Toc51678749)

[2.3 Install Intel oneAPI Toolkits 9](#_Toc51678750)

[3.0 SUPRA Setup 10](#_Toc51678751)

[3.1 Get SUPRA Source Code and Differentiate Patches of oneAPI 10](#_Toc51678752)

[3.2 Build under oneAPI env 11](#_Toc51678753)

[3.3 Run SUPRA Demo 14](#_Toc51678754)

[4.0 Migrate from CUDA to DPC++ 15](#_Toc51678755)

[4.1 Step1: Migrate with DPC++ Compatibility Tool 15](#_Toc51678756)

[4.2 Step2: Migrate Manually 16](#_Toc51678757)

Figures

[Figure 1. SUPRA Demo UI 15](#_Toc51678764)

**Tables**

***Revision History***

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision** | **Description** |
| May 2020 | 0.1 | First draft |
| Aug 2020 | 0.8 | Add DPC++ migration process. Update oneAPI from Beta06 to Beta07 |

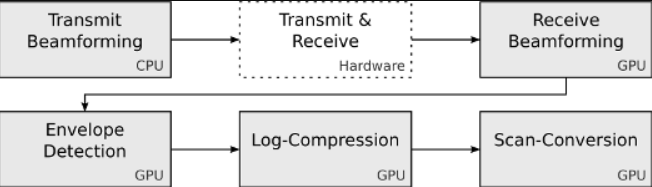
# Introduction

The purpose of this document is to guide users in migrating your CUDA\* applications into standards-based DPC++ (Data Parallel C++) code. This will allow you to manually tune and optimize your code in DPC++ and then compile your code to run on a more general platform, for example, an Intel NUC, with a general purpose GPU.

We would use oneAPI toolkit -- Intel® DPC++ Compatibility Tool to implement the migration from CUDA to standard DPC++. For more details, please refer to: <https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/dpc-compatibility-tool.html>

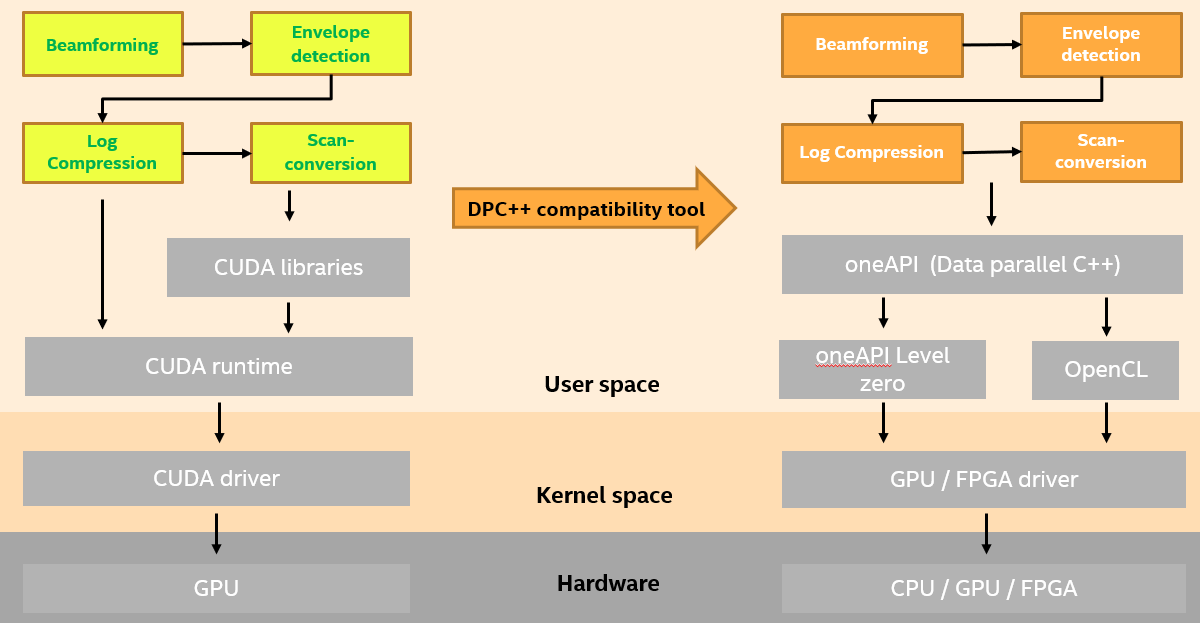
The CUDA-based sample application is SUPRA. It is an open source software defined ultrasound processing for real-time applications. Covering everything from beamforming to output of B-Mode images, SUPRA can help reproducibility of results and allows modifications to the image acquisition.

It includes all processing stages of a usual ultrasound pipeline; it can be executed in 2D and 3D on consumer GPUs in real- time. Pipeline shown as below:



More detail of SUPRA project, please refer to <https://github.com/IFL-CAMP/supra>.

SUPRA’s key algorithm is written in CUDA code. Now, we use DPC compatibility tool to migrate them to DPC++, shown as below:



## How to use this document

If you would like to setup SUPRA demo application to make it run on an Intel device, please follow **Chapter2. Host System Setup**, and **Chapter3. SUPRA Setup**.

Then, if you want to know more detail about how to migrate SUPRA CUDA-based code to SUPRA DPC++-based code, please learn more from **Chapter4. Migrate from CUDA to DPC++**. You could try to generate the patches by yourself. (The patches are mentioned in **Chapter3.1**, you could retrieve it from <https://github.com/intel/supra-on-oneapi>.)

## Terminology

**Table 1. Terminology**

|  |  |
| --- | --- |
| **Term** | **Description** |
| oneAPI | oneAPI is a cross-industry, open, standards-based unified programming model that delivers a common developer experience across accelerator architectures—for faster application performance, more productivity, and greater innovation. Please refer to <https://www.oneapi.com/>.  Intel® oneAPI products will deliver the tools needed to deploy applications and solutions across the architectures. Please refer to <https://software.intel.com/content/www/us/en/develop/tools/oneapi.html> |
| SUPRA | An open-source pipeline for fully software defined ultrasound processing for real-time applications. Covering everything from beamforming to output of B-Mode images, SUPRA can help reproducibility of results and allows modifications to the image acquisition. |
| DPC++ | At the core of the oneAPI specification is DPC++, an open, cross-architecture language built upon the ISO C++ and Khronos SYCL standards. |
| Intel® DPC++ Compatibility Tool | The Intel® DPC++ Compatibility Tool assists in migrating your existing CUDA code to Data Parallel C++ (DPC++) code. Refer to <https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/dpc-compatibility-tool.html> |

§

# Host System Setup

In this section, we’ll explain how to set up your development system and necessary software packages.

## Host Development System

The preferred (and tested) development host platform is PC with Ubuntu 18.04. The PC should have a graphics processor. The hardware requirement used for this demo: Intel CPU with Gen9 or later graphics. In this document, all steps were verified on Intel i7-8700K CPU with Intel(R) UHD Graphics 630, please refer to: <https://ark.intel.com/content/www/us/en/ark/products/126684/intel-core-i7-8700k-processor-12m-cache-up-to-4-70-ghz.html>

## Install Basic Packages

$ sudo apt-get install cmake cmake-gui qt5-default libtbb-dev libopenigtlink-dev git build-essential clang

## Install Intel oneAPI Toolkits

Please follow the steps in : <https://software.intel.com/content/www/us/en/develop/articles/installation-guide-for-intel-oneapi-toolkits.html>

In this document, we choose 2021.1-beta07 version of [Intel® oneAPI Base Toolkit](https://software.intel.com/content/www/us/en/develop/tools/oneapi/base-toolkit.html). [**Note**] Please use “**Local Installer**” method, and use below command to get Intel® oneAPI Base Toolkit Beta07 version (it is not the latest version of oneAPI Base Toolkit), instead of the default link of “**Download**”

wget http://registrationcenter-download.intel.com/akdlm/irc\_nas/16702/l\_BaseKit\_b\_2021.1.7.1506\_offline.tar.gzd

[**Note**] To align with oneAPI Base Toolkit Beta07 version, please follow the steps in <https://github.com/intel/compute-runtime/releases/tag/20.27.17231> to install the computer-runtime packages, which are low level user space driver for oneAPI.

|  |
| --- |
| $ mkdir neo  $ cd neo  $ wget https://github.com/intel/compute-runtime/releases/download/20.27.17231/intel-gmmlib\_20.2.2\_amd64.deb  $ wget https://github.com/intel/compute-runtime/releases/download/20.27.17231/intel-igc-core\_1.0.4241\_amd64.deb  $ wget https://github.com/intel/compute-runtime/releases/download/20.27.17231/intel-igc-opencl\_1.0.4241\_amd64.deb  $ wget https://github.com/intel/compute-runtime/releases/download/20.27.17231/intel-opencl\_20.27.17231\_amd64.deb  $ wget https://github.com/intel/compute-runtime/releases/download/20.27.17231/intel-ocloc\_20.27.17231\_amd64.deb  $ wget <https://github.com/intel/compute-runtime/releases/download/20.27.17231/intel-level-zero-gpu_0.8.17231_amd64.deb>  $ sudo dpkg -i \*.deb |

# SUPRA Setup

## Get SUPRA Source Code and Differentiate Patches of oneAPI

Download the source code from <https://github.com/IFL-CAMP/supra.git>

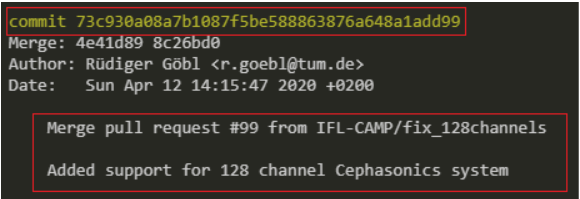
$ mkdir my\_supra

$ cd my\_supra

$ git clone <https://github.com/IFL-CAMP/supra.git>

$ git reset --hard 73c930a08a7b1087f5be588863876a648a1add99

Then ‘git log’, you would find your head commit is located at:



Then, please get the patches of supra-on-oneapi, and apply all the patches to SUPRA git stream: (How does the patches generate? In fact, it uses oneAPI DPCT tool to migrate CUDA code to Intel DPC++ code, please refer to **Chapter4, Migrate from CUDA to DPC++**)

$ git clone https://github.com/intel/supra-on-oneapi

$ git am ../supra\_for\_oneapi\_patch/\*.patch

## Build under oneAPI env

After source code download, and patches applied, let’s start compile it. Initialize one API environment:

$ source /opt/intel/inteloneapi/setvars.sh

:: initializing environment ...

advisor -- latest

ccl -- latest

compiler -- latest

Warning: Intel PAC A10 device is not found.

Please install the Intel PAC card to execute your program on an FPGA device.

Warning: Intel PAC S10 device is not found.

Please install the Intel PAC card to execute your program on an FPGA device.

daal -- latest

debugger -- latest

dev-utilities -- latest

dpcpp-ct -- latest

intelpython -- latest

ipp -- latest

mkl -- latest

mpi -- latest

oneDNN -- latest

tbb -- latest

vpl -- latest

vtune -- latest

:: oneAPI environment initialized ::

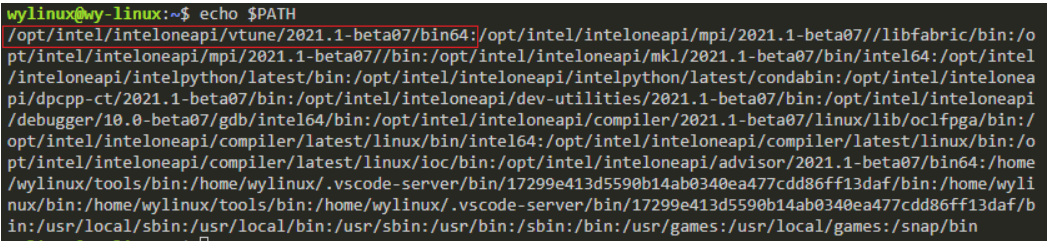
Generate makefiles:

$ cd ~/my\_supra/ultrasound-emu/port\_process/supra

$ mkdir build

$ cd build/

[**Note**] run ‘echo $PATH’, you could get env parameters as below,



Remove the value in red rectangle, copy the reset of parameters to ‘PATH’. For example:

$ **export PATH=**/opt/intel/inteloneapi/mpi/2021.1-beta07/libfabric/bin:/opt/intel/inteloneapi/mpi/2021.1-beta07/bin:/opt/intel/inteloneapi/mkl/2021.1-beta07/bin/intel64:/opt/intel/inteloneapi/intelpython/latest/bin:/opt/intel/inteloneapi/intelpython/latest/condabin:/opt/intel/inteloneapi/dpcpp-ct/2021.1-beta07/bin:/opt/intel/inteloneapi/dev-utilities/2021.1-beta07/bin:/opt/intel/inteloneapi/debugger/8.3-beta07/gdb/intel64/bin:/opt/intel/inteloneapi/compiler/2021.1-beta07/linux/lib/oclfpga/bin:/opt/intel/inteloneapi/compiler/latest/linux/bin/intel64:/opt/intel/inteloneapi/compiler/latest/linux/bin:/opt/intel/inteloneapi/compiler/latest/linux/ioc/bin:/opt/intel/inteloneapi/advisor/2021.1-beta07/bin64:/home/qingx/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin

Use opencl as low-level library:

$ export SYCL\_BE=PI\_OPENCL

Configure project:

$ CC=clang CXX=dpcpp cmake ..

-- Building SUPRA

-- The C compiler identification is Clang 11.0.0

-- The CXX compiler identification is Clang 11.0.0

-- Check for working C compiler: /opt/intel/inteloneapi/compiler/latest/linux/bin/clang

-- Check for working C compiler: /opt/intel/inteloneapi/compiler/latest/linux/bin/clang -- works

-- Detecting C compiler ABI info

-- Detecting C compiler ABI info - done

-- Detecting C compile features

-- Detecting C compile features - done

-- Check for working CXX compiler: /opt/intel/inteloneapi/compiler/latest/linux/bin/dpcpp

-- Check for working CXX compiler: /opt/intel/inteloneapi/compiler/latest/linux/bin/dpcpp -- works

-- Detecting CXX compiler ABI info

-- Detecting CXX compiler ABI info - done

-- Detecting CXX compile features

-- Detecting CXX compile features - done

-- CMAKE\_GENERATOR: Unix Makefiles

-- CMAKE\_CL\_64:

-- Building SUPRA\_Lib

-- Looking for TBB in ''

-- CUBLAS not found, NOT adding minimum variance beamformer

-- Building SUPRA Commandline Interface

-- Building SUPRA Executor

-- Building SUPRA Graphic Interface

-- Building SUPRA Wrapper

-- Downloading sample data. This may take a while.

-- [download 0% complete]

-- [download 1% complete]

-- [download 2% complete]

-- [download 3% complete]

......

-- [download 98% complete]

-- [download 99% complete]

-- [download 100% complete]

-- Extracting sample data

-- Configuring done

-- Generating done

-- Build files have been written to: /home/qingx/my\_supra /ultrasound-emu/port\_process/supra/build

Compile source code:

$ make -j5

Scanning dependencies of target NodeEditor

Scanning dependencies of target SUPRA\_Lib

[ 1%] Creating directories for 'NodeEditor'

[ 2%] Building CXX object src/SupraLib/CMakeFiles/SUPRA\_Lib.dir/ContainerFactory.cpp.dp.cpp.o

[ 4%] Building CXX object src/SupraLib/CMakeFiles/SUPRA\_Lib.dir/SyncRecordObject.cpp.o

[ 5%] Building CXX object src/SupraLib/CMakeFiles/SUPRA\_Lib.dir/SupraManager.cpp.o

[ 7%] Building CXX object src/SupraLib/CMakeFiles/SUPRA\_Lib.dir/RecordObject.cpp.o

[ 8%] Performing download step (git clone) for 'NodeEditor'

[ 9%] Building CXX object src/SupraLib/CMakeFiles/SUPRA\_Lib.dir/TrackerData.cpp.o

…

…

[100%] Linking CXX executable SUPRA\_GUI

[100%] Built target SUPRA\_GUI

## Run SUPRA Demo

Run SUPRA demo application

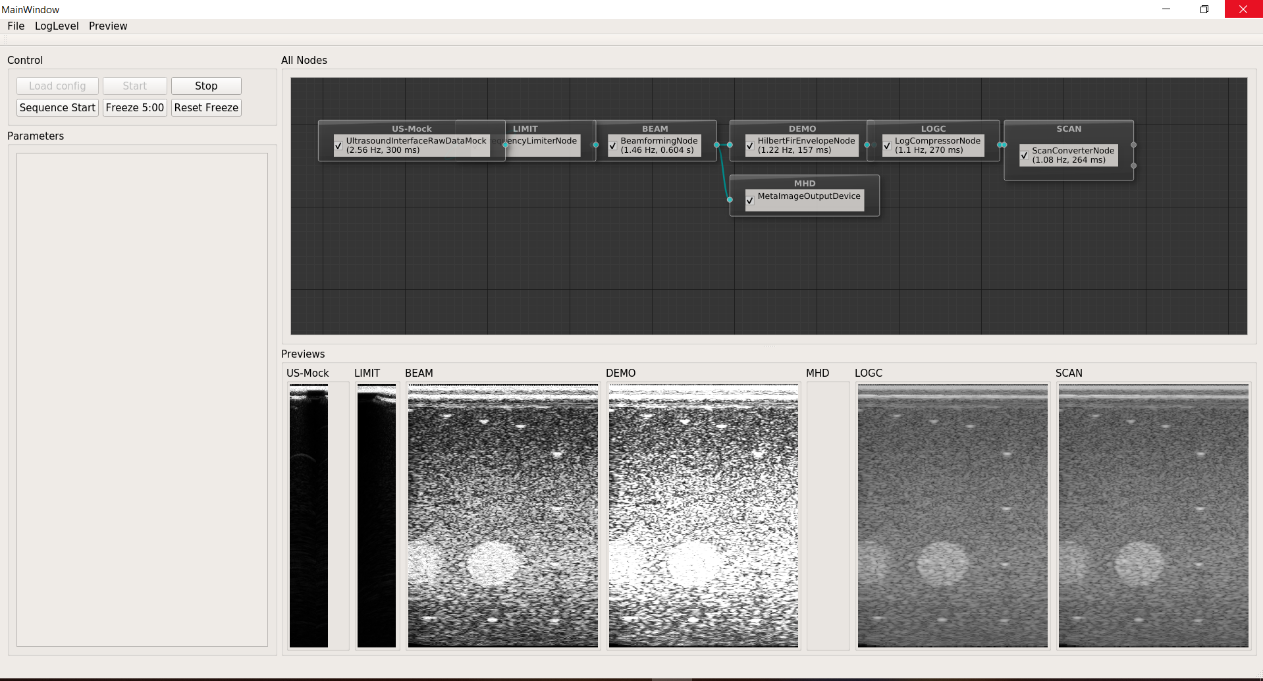
$ export SYCL\_BE=PI\_OPENCL

$ cd ~/my-supra/ultrasound-emu/port\_process/supra/build

$ src/GraphicInterface/SUPRA\_GUI -c data/configDemo.xml -a

Then, you could see UI as below. Press ‘Sequence Start’ button on upper left side, you could see the whole pipeline start to run, and its fps in the upper diagram.

Figure . SUPRA Demo UI



Check the performance, open supra.log in the build directory, it will show every node performance and the average performance in Millisecond.

$ cat supra.log

# Migrate from CUDA to DPC++

As mentioned before about SUPRA, it is written in CUDA code. By using the [Intel® DPC++ Compatibility Tool](https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/dpc-compatibility-tool.html), CUDA code could be migrated to DPC++ code.

Please note that automation tool could migrate 70~80% of CUDA code to DPC++ code. So, the reset of modification would be implemented manually.

## Step1: Migrate with DPC++ Compatibility Tool

Before you migrate, remove the space in “<< <…>> >” in RxBeamformerCuda.cu and ScanConverter.cu . In CUDA code, using <<<…>>>execution configuration syntax to specify the number of CUDA threads that execute that kernel for a given kernel call.

Run ‘dpct’ command to migrate from CUDA to DPC++:

$ dpct --in-root=./ --out-root=./migrated --extra-arg=-Isrc/SupraLib --extra-arg=-Isrc/SupraLib/Beamformer --extra-arg=-Isrc/SupraLib/utilities --extra-arg=-std=c++11 --extra-arg=-Wno-c++11-narrowing --extra-arg=-DHAVE\_CUDA ./src/SupraLib/Beamformer/ScanConverter.cu ./src/SupraLib/Beamformer/HilbertFirEnvelope.cu ./src/SupraLib/Beamformer/LogCompressor.cu ./src/SupraLib/Beamformer/RxBeamformerCuda.cu ./src/SupraLib/ContainerFactory.cpp ./src/SupraLib/utilities/DataType.cpp

After running the command, you could get migrated files which are located in folder “migrated”. Please also refer to patch “0001-remove-spaces-in-.cu-files-generate-new-files-in-one.patch”.

## Step2: Migrate Manually

1. my\_supra/ultrasound-emu/port\_process/supra/ CMakeLists.txt

Original:

|  |
| --- |
| FIND\_PACKAGE( OpenMP REQUIRED)  if(OPENMP\_FOUND)  set(CMAKE\_C\_FLAGS "${CMAKE\_C\_FLAGS} ${OpenMP\_C\_FLAGS}")  set(CMAKE\_CXX\_FLAGS "${CMAKE\_CXX\_FLAGS} ${OpenMP\_CXX\_FLAGS}")  set(CMAKE\_EXE\_LINKER\_FLAGS "${CMAKE\_EXE\_LINKER\_FLAGS} ${OpenMP\_EXE\_LINKER\_FLAGS}")  endif() |

**Modified**:

|  |
| --- |
| Comment out |

1. my\_supra/ultrasound-emu/port\_process/supra/src/ SupraLib/ CMakeLists.txt

|  |
| --- |
| 1. Comment out lin62-lin86, lin91-lin93, lin95, lin101-lin102, lin105-lin108, lin198-lin213, lin217-lin232, lin321-lin327, lin337-lin349, lin357-lin358, lin361-lin370, lin443-lin452, lin455-lin460, lin467-lin468, lin504-lin510 2. Change the files to migrated files.   For example:  lin 280:  original: Beamformer/RxBeamformerCuda.cu  modified: Beamformer/RxBeamformerCuda.dp.cpp |

1. my\_supra/ultrasound-emu/port\_process /supra/src/SupraLib/utilities/logging.h:

line 240:

|  |
| --- |
| **Migrated:** log(Severity::log, o) log(Severity::log, o, o, o);  **Modified:** Base::log(Severity::log, o...) |

line 273:

Original:

|  |
| --- |
| **Migrated:** log(Severity::error, red, o, reset);  **Modified:** Base::log(Severity::error, red, o..., reset); |

1. cudaStremDefault/cudaStreamPerThread

|  |
| --- |
| **Migrated:** cudaStreamDefault or cudaStreamPerThread  **Modified:** &(dpct::get\_defalut\_queue()); |

Files:

src/SupraLib/utilities/FirFilterFactory.h --lin92

src/SupraLib/Beamformer/WindowFunction.cpp --lin46

src/SupraLib/Beamformer/ScanConverter.dp.cpp

--line662-line666,lin672,lin785,lin816

src/SupraLib/Beamformer/RxBeamformerCuda.dp.cpp

--lin37,lin41,lin44,lin46

1. using ::max

error:  no member named 'max' in the global namespace.

|  |
| --- |
| **Migrated:** using ::max;  using ::min;  using ::round;  using ::floor;  using ::ceil;  **Modified:** using sycl::max;  using sycl::min;  using sycl::round;  using sycl::floor;  using sycl::ceil; |

Files:

src/SupraLib/utilities/cudaUtility.h --lin32-lin36

src/SupraLib/Beamformer/WindowFunction.h --lin32-lin33

1. error: call to ‘max’ is ambiguous

|  |
| --- |
| **Migrated:** maxTransitTime=max(…);  **Modified:** maxTransitTime=sycl::max(…); |

Files: src/SupraLib/Beamformer/Beamformer.cpp.dp.cpp --lin985

src/SupraLib/Beamformer/LogCompressor.dp.cpp

–- lin34,lin39 (log10(…)🡪sycl::log10(…));

1. thrust algorithms
2. thrust::transform

|  |
| --- |
| **Migrated:** Std::transform(dpstd::execution::make\_sycl\_policy<class Policy\_7c84d8>(Dpct::get\_default\_queue()),  Thrust::cuda::par.on(inImageData->getStream()),  inImageData->get(), inImageData->get()+(width\*height\*depth),pComprGpu->get(),c);  **Modified:**  auto inImageData\_t=inImageData->get();  auto pComprGpu\_t=pComprGpu->get();  inImageData->getStream()->submit([&](sycl::handler &h){  h.parallel\_for<>(sycl::range<1>( width\*height\*depth),  [=](sycl::id<1> idx){  pComprGpu\_t=c(inImageData\_t[idx]);  });  }); |

Files:

src/SupraLib/Beamformer/LogCompressor.dp.cpp

-- line72-line76

1. thrust::pair

|  |
| --- |
| **Migrated:** thrust::pair  **Modified:** std::pair |

Files:

src/SupraLib/Beamformer/ScanConverter.dp.cpp --

line93,line188,line218,line276

1. thrust::placeholders

|  |
| --- |
| **Migrated:** thrust::placeholders  **Modified:** remove . (oneAPI don’t support this functional, and only the namespace was used in the cuda codes, also don't have any placeholders, so just remove it ) |

Files：

src/SupraLib/Beamformer/HilbertFirEnvelope.dp.cpp –-lin25

1. thrust::unary\_function **[fixed on beta08]**

|  |
| --- |
| **Migrated:** thrust::unary\_function<In, Out>  **Modified:** std::unary\_function<In, Out> |

Files:

src/SupraLib/Beamformer/LogCompressor.dp.cpp –-lin26

1. dim3 Data Type **[fixed on beta08]**

|  |
| --- |
| **Migrated:** blockSize.x;  blockSize.y;  blockSize.z;  **Modified:** blockSize[0];  blockSize[1];  blockSize[2]; |

Files:

src/SupraLib/Beamformer/HilbertFirEnvelope.dp.cpp

--line93

src/SupraLib/Beamformer/ScanConverter.dp.cpp

-- line459,line481,line779

src/SupraLib/Beamformer/RxBeamformerCuda.dp.cpp

-- lin314, lin411

1. error: kernel parameter has non-trivial constructible class/struct

|  |
| --- |
| **Migrated:** WindowFunctionGpu(const WindowFunctionGpu& a):  m\_numEntriesPerFunction(a.m\_numEntriesPerFunction),  m\_data(a.m\_data),  m\_scale(a.m\_scale){};  **Modified:** remove. |

Files: src/SupraLib/Beamformer/WindowFunction.h –-lin53

1. error: kernel parameter has non-trivial constructible class/struct

**[fixed on beta08]**

For example: src/SupraLib/Beamformer/HilbertFirEnvelope.dp.cpp

–-lin104

|  |
| --- |
| **Migrated:** kernelFilterDemodulation(inImageData->get(),  m\_hilbertFilter->get(),pEnv->get(), numSamples,  numScanlines, m\_filterLength\_ct5, item\_ct1);  **Modified:** auto m\_inImageData\_t=inImageData->get();  auto m\_hilbertFilter\_t=m\_hilbertFilter->get();  auto m\_pEnv\_t=pEnv->get();  kernelFilterDemodulation(m\_inImageData\_t,  m\_hilbertFilter\_t, m\_pEnv\_t , numSamples,  numScanlines, m\_filterLength\_ct5, item\_ct1); |

Other files:

src/Beamformer/ScanConverter.dp.cpp -- line469,lin791

1. src/SupraLib/Beamformer/ScanConverter.dp.cpp --672

|  |
| --- |
| **Migrated:** cudaSafeCall((((cudaStream\_t)0x2)->memset(  m\_mask- >get(), 0, m\_mask->size()  \* sizeof(uint8\_t),cudaStreamPerThread), 0));  **Modified:** m\_mask->getStream()->submit([&](sycl::handler &h){  h.memset(m\_mask->get(),0,  m\_mask->size()\*sizeof(uint8\_t));}); |

1. src/SupraLib/Beamformer/ScanConverter.dp.cpp --lin791

|  |
| --- |
| **Migrated:**  computeParameterBB3D<Tf, Ti>( …,  static\_cast<vec3T<Tf>>((\*scanlines)[ scanlineIdxX ]  [scanlineIdxY ].position),  static\_cast<vec3T<Tf>>((\*scanlines)[ scanlineIdxX ]  [scanlineIdxY ].direction),  static\_cast<vec3T<Tf>>((\*scanlines)[ scanlineIdxX + 1 ]  [ scanlineIdxY ].position),  static\_cast<vec3T<Tf>>((\*scanlines)[ scanlineIdxX + 1 ]  [ scanlineIdxY ].direction),  static\_cast<vec3T<Tf>>((\*scanlines)[ scanlineIdxX ]  [ scanlineIdxY + 1 ].position),  static\_cast<vec3T<Tf>>((\*scanlines)[ scanlineIdxX ]  [ scanlineIdxY + 1 ].direction),  static\_cast<vec3T<Tf>>((\*scanlines)[ scanlineIdxX + 1 ]  [ scanlineIdxY + 1 ].position),  static\_cast<vec3T<Tf>>((\*scanlines)[ scanlineIdxX + 1 ]  [ scanlineIdxY + 1 ].direction),  …,  );  **Modified:**  auto scanlines\_buffer\_t =  (\*scanlines)[scanlineIdxX][scanlineIdxY];  computeParameterBB3D<Tf, Ti>( …,  static\_cast<vec3T<Tf>>( scanlines\_buffer\_t.position),  static\_cast<vec3T<Tf>>( scanlines\_buffer\_t.direction), static\_cast<vec3T<Tf>>(scanlines\_buffer\_t.position), static\_cast<vec3T<Tf>>(scanlines\_buffer\_t.direction), static\_cast<vec3T<Tf>>(scanlines\_buffer\_t.position), static\_cast<vec3T<Tf>>(scanlines\_buffer\_t.direction), static\_cast<vec3T<Tf>>(scanlines\_buffer\_t.position), static\_cast<vec3T<Tf>>(scanlines\_buffer\_t.direction),  …,); |

1. src/SupraLib/Beamformer/RxBeamformerCuda.dp.cpp

–-lin331, lin350, lin372, lin391 **[fixed on beta08]**

|  |
| --- |
| **Migrated:** rxBeamformingDTSPACE3DKernel(…,  x\_elemsDTsh\_acc\_ct1.get\_pointer(),  z\_elemsDTsh\_acc\_ct1.get\_pointer(),  functionShared\_acc\_ct1.get\_pointer(),…)  **Modified:** rxBeamformingDTSPACE3DKernel(…,  (LocationType\*)x\_elemsDTsh\_acc\_ct1.get\_pointer(),  (LocationType\*)z\_elemsDTsh\_acc\_ct1.get\_pointer(),  (supra::WindowFunction::ElementType  \*)functionShared\_acc\_ct1.get\_pointer(),…) |

1. src/SupraLib/SuperManerge.cpp --lin506, lin538, lin507, lin539

|  |
| --- |
| **Migrated:** tbb::flow::sender;  tbb::flow::receiver;  **Modified:** tbb::flow::interface11::sender;  tbb::flow::interface11::receiver; |

1. src/SupraLibContainerFactory.cp.dp.cpp --lin68

|  |
| --- |
| **Migrated:** cudaSafaCall(cudaMemGetInfo(...));  **Modified:** remove |

1. src/SupraLib/Container.h --lin335, lin337

|  |
| --- |
| **Migrated:** std::function<void(sycl::queue, \*int)>  **Modified:** std::function<void(sycl::queue\*, int)> |

1. src/SupraLib/Container.h

--line50, lin236,lin241,lin246,lin268,lin295,lin330,lin331

|  |
| --- |
| **Migrated:** m\_creationEvent  **Modified:** removed |

1. src/SupraLib/Container.h

--lin131, lin140, lin149, lin158, lin223

|  |
| --- |
| **Migrated:** cudaSafeCall((source.getStream()->memcpy(  this->get(),source.get(),source.size()\*sizeof(T))),0);  **Modified:** source.getStream()->submit([&](sycl::handler& h){  h.memcpy(this->get(),  source.get(),source.size()\*sizeof(T));}); |

1. src/SupraLib/Container.h --lin885

|  |
| --- |
| **Migrated:** CUDART\_CB  **Modified:** remove |

1. src/SupraLib/Container.h --lin875

|  |
| --- |
| **Migrated:** std::async([&]() {  m\_associatedStream->wait();  &(Container<T>::cudaDeleteCallback)  (m\_associatedStream,0, funcPointer);}  **Modified:** Remove “&” |

1. my\_supra/ultrasound-emu/port\_process/supra/src/SupraLib/Interfactory.cpp

|  |
| --- |
| Comment out those nodes which are not migrated in lin207-lin243 |

After above modification by manual, you could get the patches, which would be similar to the patch “0003-Modification-.dp.cpp-and-h-files-to-run.patch”.