Parallel Computing with GPUs: Visual Studio Guide for CUDA

Dr Paul Richmond
http://paulrichmond.shef.ac.uk/teaching/COM4521/

The University Of Sheffield.

Compiling a CUDA program

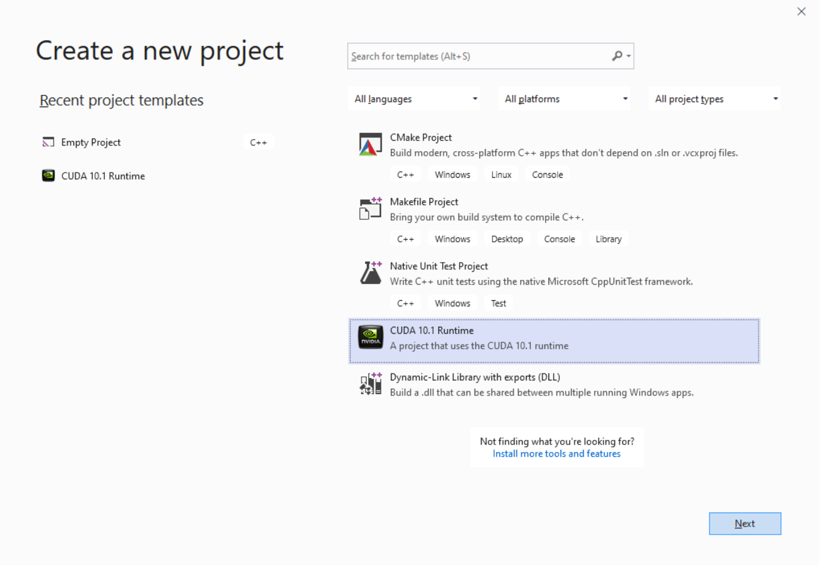
- CUDA C Code is compiled using **nvcc** e.g.
- Will compile host AND device code to produce an executable

```
nvcc –o example example.cu
```

- We will be using Visual Studio to build our CUDA code so we will not need to compile at the command line (unless you are running on ShARC)

Creating a CUDA Project

- Create New CUDA Project
  - Select *NVIDIA -> CUDA 11.1*
  - This will create a project with a default kernels.cu file containing a basic vector addition example



Create a new project

Search for templates (Alt+S)

Recent project templates

Empty Project C++

CUDA 10.1 Runtime

All languages | All platforms | All project types

CMake Project
Build modern, cross-platform C++ apps that don't depend on .sln or .vcxproj files.
C++ Windows Linux Console

Makefile Project
Bring your own build system to compile C++.
C++ Windows Desktop Console Library

Native Unit Test Project
Write C++ unit tests using the native Microsoft CppUnitTest framework.
C++ Windows Test

CUDA 10.1 Runtime
A project that uses the CUDA 10.1 runtime

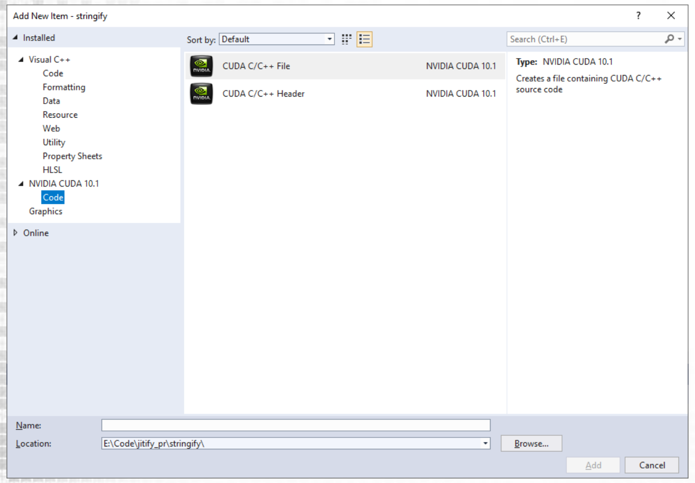
Dynamic-Link Library with exports (DLL)
Build a .dll that can be shared between multiple running Windows apps.

Not finding what you're looking for?
Install more tools and features

Next

Preferred Method!

Adding a CUDA source file



Add New Item - stringify

Installed
- Visual C++
  - Code
  - Formatting
  - Data
  - Resource
  - Web
  - Utility
  - Property Sheets
  - HLSL
- NVIDIA CUDA 10.1
  - Code
  - Graphics
- Online

Sort by: Default

Search (Ctrl+E)

CUDA C/C++ File	NVIDIA CUDA 10.1
CUDA C/C++ Header	NVIDIA CUDA 10.1

Type: NVIDIA CUDA 10.1
Creates a file containing CUDA C/C++ source code

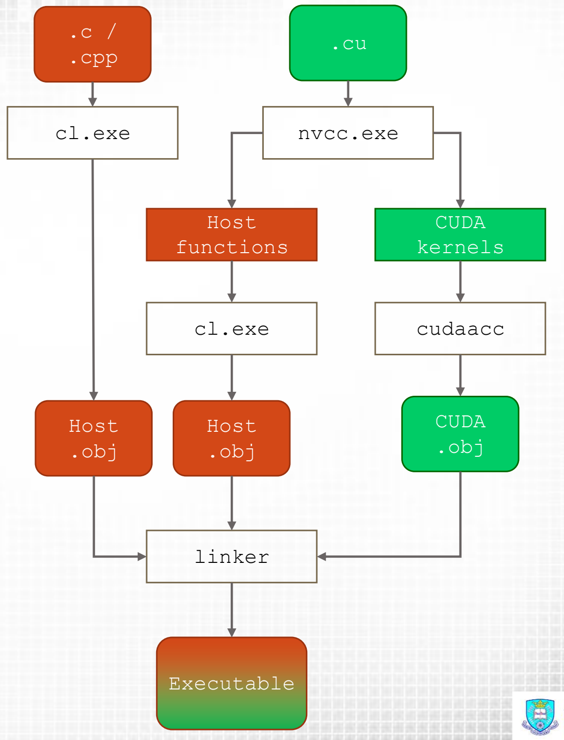
Name:
Location: E:\Code\jitify_pr\stringify\

Browse... Add Cancel

- Alternatively add a CUDA source file to an existing application
- If you do this you must modify the project properties to include CUDA build customisations
  - http://developer.download.nvidia.com/compute/cuda/6_5/rel/docs/CUDA_Getting_Started_Windows.pdf (section 3.4)

Compilation

- CUDA source file (`*.cu`) are compiled by `nvcc`
- An existing cuda.rules file creates property page for CUDA source files
  - Configures `nvcc` in the same way as configuring the C compiler
  - Options such as optimisation and include directories can be inherited from project defaults
- C and C++ files are compiled with `cl` (MSVCC compiler)



Device Versions

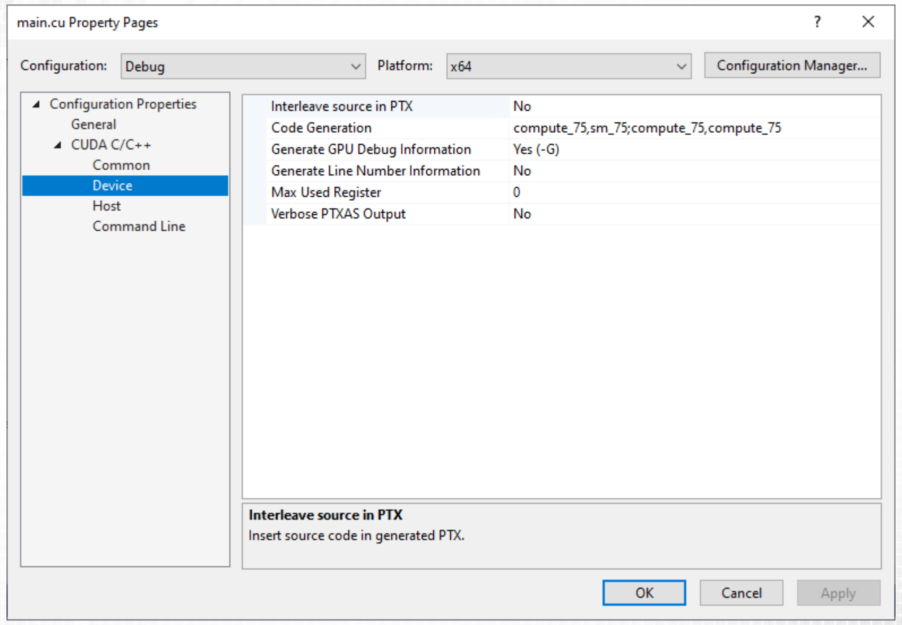
- Different generations of NVIDIA hardware have different compatibility
  - In the last lecture we saw product families and chip variants
  - These are classified by **CUDA compute versions**
- Compilation normally builds for CUDA compute version 35
  - See Project Properties, *CUDA C/C++Device->Code Generation*
  - Default value is “`compute_35,sm_35`”
  - Any hardware with greater than the compiled compute version can execute the code (backwards compatibility)
- You can build for multiple versions using separator
  - E.g. “`compute_35,sm_35;compute_60,sm_60`”
  - This will increase build time and execution file size
  - Runtime will select the best version for your hardware

https://en.wikipedia.org/wiki/CUDA#Supported_GPUs

Device Versions of Available GPUs

- Diamond Machines
  - Pascal Architecture
    - `compute_60,sm_60;`

CUDA Properties


main.cu Property Pages

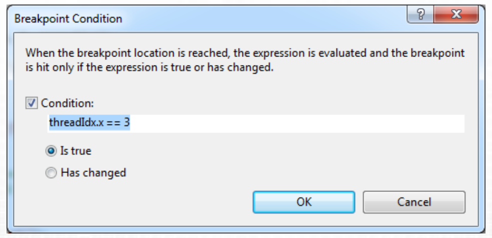
Configuration: Debug   Platform: x64

Interleave source in PTX	No
Code Generation	compute_75,sm_75;compute_75,compute_75
Generate GPU Debug Information	Yes (-G)
Generate Line Number Information	No
Max Used Register	0
Verbose PTXAS Output	No

Interleave source in PTX
Insert source code in generated PTX.

Debugging

- NSIGHT is a GPU debugger for debugging GPU kernel code
  - It does not debug breakpoints in host code
- To launch select insert a breakpoint and select NSIGHT-> Start CUDA Debugging
  - You must be in the debug build configuration.
  - When stepping all warps except the debugger focus will be paused
- Use conditional breakpoints to focus on specific threads
  - Right click on break point and select Condition



Breakpoint Condition

When the breakpoint location is reached, the expression is evaluated and the breakpoint is hit only if the expression is true or has changed.

☑ Condition:

threadIdx.x == 3

● Is true

○ Has changed

Error Checking

- `cudaError_t`: enumerator for runtime errors
  - Can be converted to an error string (`const char *`) using `cudaGetErrorString(cudaError_t)`
- Many host functions (e.g. `cudaMalloc`, `cudaMemcpy`) return a `cudaError_t` which can be used to handle errors gracefully

```
cudaError_t cudaStatus;

cudaStatus = cudaMemcpy(dev_a, a, size * sizeof(int), cudaMemcpyHostToDevice);
if (cudaStatus != cudaSuccess) {
    //handle error
}
```

- Kernels do not return an error but if one is raised it can be queried using the `cudaGetLastError()` function

```
addKernel<<<1, size>>>(dev_c, dev_a, dev_b);
cudaStatus = cudaGetLastError();
```