LESSON 1

ERROR HANDLER

TOPICS

- How to detect error codes in CUDA programs

Key words: *error, cudaSuccess, cudaError, cudaGetLastError, cudaGetErrorString, cudaDeviceSynchronize*.

Detecting errors

- In the first topics, we saw in the syntax of the device functions that they returned a *cudaError_t* code.
- This value is used for CUDA programmers to detect errors in the programs. If the function was successfully executed, it returns *cudaSuccess*, in other case an error code will be returned.
- We can distinguish between errors from the CUDA API and errors launched by the kernel calls.
- While the functions in the CUDA API return a *cudaError*, the kernel subroutines do not return a value.

- To catch the error we can use *cudaGetLastError()*.

- To have a human-readable description of the error codes we can use the *cudaGetErrorString()*.

```
cudaError_t cudaGetLastError ( void )
```

```
const char* cudaGetErrorString (cudaError_t error)
error - code to convert to string
```

- The *cudaGetLastError()* just sends the last error, if there is one that precedes the last one, it is not reported.

- The kernels are asynchronous and it is necessary to block the execution until the device has finished its work. To do this we can use the *cudaDeviceSynchronize* () function.

- Common errors

- error: a host function call cannot be configured

- invalid configuration argument

Example

- 01simple_kernel2_err: This program makes use of the Error.h header file, in order to detect errors in the source code.

Practice

- 02cube: This program must get as a result an array with the cube of the original values of the array, using the Error.h header.