OpenCL - CM 3 (Exercise)

Jonàs Martínez

11/04/23

1 Reminders

- Do not forget to test for errors, for instance, :
  `err = clu_Queue->enqueueNDRangeKernel( ...  );`
  `cluCheckError(err,"kernel");`

2 Exercise

- Implement the CPU version of the bitonic sort.
- Modify the algorithm to output the ranks of the numbers being compared during execution and the 'stage' at which the comparisons occur (see slide where the sorting network is decomposed in stages). For instance, if there is an ascending (ASC) comparison $T[i]$ and $T[j]$ at column c of stage k, the program will write `stage` k`, column` c`, T[`i`] > T[`j`]` with k, c, i and j replaced by their actual values.
- Modify the code so that the C code writes the OpenCL kernel of each stage (one kernel per column stage) for network sorting arrays of size 64. The kernel will look like this:

```
void __kernel sortnet_stage_0_col_0(int *T)
{
  int gid = get_global_id(0);
  switch (gid)
  {
  case 0:
    compare_arrow_asc(T, I0, J0); // where I0 and J1 are values
    break;
  case 1:
    compare_arrow_desc(T ,I1 ,J1);
    break;
  ...
  }
}
```

- Implement a test code running all stages and sorting an array of 64 entries. Note that we are simply writing a specialized and verbose kernel for size 64, and we do not seek to fully implement the more general and complex Bitonic sort on OpenCL.