OpenCL Extensions

Matt Sellitto
Dana Schaa
Northeastern University
NUCAR

Extensions

- Optional features supported by OpenCL
  - Not required for a conformant OpenCL implementation
- OpenCL C naming convention

  cl_<vendor_name>_<name>
- OpenCL C compiler directive

  #pragma OPENCL EXTENSION <extension> : enable
- Example (enable double precision):

  #pragma OPENCL EXTENSION cl_khr_fp64 : enable
  - khr = khronos

Extensions

- Compiler begins as if all extensions were disabled

```
#pragma OPENCL EXTENSION all : disable
```

- Extensions add a preprocessor #define
  - Can be used to optionally take advantage of features
  - Example:

```
#ifdef cl_khr_fp64
    // do something using the extension
#else
    // do something else or #error!
#endif
```

Double Precision

- OpenCL 1.1 added support for optional double precision

  ```
  #pragma OPENCL EXTENSION cl_khr_fp64 : enable
  ```

- The double data type must confirm to the IEEE-754 double precision storage format
- Vendor implementations that don't conform to IEEE-754 use vendor-specific extension

  ```
  #pragma OPENCL EXTENSION cl_amd_fp64 : enable
  ```

  - NVIDIA doesn't use an extension
    - Demotes doubles to single-precision when not supported (still 64-bits though)

Double Precision

- On AMD 5XXXX hardware
  - 5 VLIW units combine for 1 double precision
  - 2.7 TFLOPS → 544 GFLOPS
- On AMD 6XXX hardware
  - 4 VLIW units combine for 1 double precision
  - 2.7 TFLOPS → 675 GFLOPS
- NVIDIA Fermi hardware
  - 2 SP PEs combines for 1 double precision PE
  - 1.03 TFLOPS → 515 GFLOPS

Atomic Operations

- Optional support for 32-bit atomic signed, unsigned, and floating point operations
- Example (addition):

  int atomic_add (volatile __global int *p, int val)
- Data must live in either global or local memory
  - Each location is enabled separately (most devices support local, only more recent support global)

    cl_khr_global_int32_base_atomics

    cl_khr_local_int32_base_atomics

Byte-Addressable Stores

- Restriction that writes to sub-32-bit data types are not guaranteed to be supported
  - Medical data is sometimes unsigned chars or shorts

  cl_khr_byte_addressable_store
- Required for AMD compiler
- Not required for NVIDIA compiler

Printf

- Adds printf functionality to OpenCL kernels

  cl_amd_printf

- Both CPU and GPU targets
  - Printing is buffered and is displayed *after* kernel completes
    - Can’t print from a kernel that crashes

Printf

- Exercise:
  - Add printf code the VectorAddition program
  - Have the thread with ID 0 in each work group print a value
  - Have only thread 5 in workgroup 2 print a value