Accelerator Extension for OpenCL

Authors: Nico Galoppo, Craig Hansen-Sturm
Reviewers: Ben Ashbaugh, David Blythe, Hong Jiang, Stephen Junkins, Raun Krisch, Krzysztof Laskowski, Matt McClellan, Teresa Morrison, Bartosz Sochacki

This document presents the accelerator extension for OpenCL, which provides an abstraction for domain-specific acceleration engines in the OpenCL runtime. Accelerator objects provide a black-box abstraction of software- and/or hardware-accelerated functionality that can be provided by OpenCL vendors.

1.1 Accelerator Objects

1.2 Bugzilla Bugs

TBD

1.3 OpenCL Spec

OpenCL 1.2

1.4 Motivation

Fixed function processors, such as the built-in OpenCL sampler, are logically characterized by collections of parameters. Depending on the underlying fixed function processor being represented, these parameter sets may be diverse; however, it is quite useful to manage fixed function processors in a generic and unified manner. Accelerators abstract the programmable state of an acceleration engine, which may be implemented by software, hardware fixed function engine(s), or both. This configurable state persists past the invocation of an accelerator, which can be beneficial when hardware state updates are costly. In summary, the accelerator extension provides *encapsulation, abstraction,* and *potential efficiency advantages* for domain-specific acceleration engines to be implemented and defined by OpenCL vendors.

1.5 Overview

The accelerator extension consists of a unified set of OpenCL runtime APIs to create, query, and manage the lifetime of objects which represent acceleration processors, engines, or algorithms. Accelerator object instances are referenced with the generic **cl_accelerator_intel** type by the runtime API, but they are always associated with a specific acceleration engine type, which is assigned by the application at accelerator object creation time. Descriptors are used to assign acceleration engine-specific properties to the accelerator objects. This mechanism of specialized creation with generic referencing is analogous to the way image objects are managed in the OpenCL 1.2 runtime API via generic **cl_mem** referencing but specialized descriptor-based creation with **clCreateImage**().

Each accelerator object is assigned a unique type id. Vendors are freely encouraged to define new accelerator types and ids. This base extension provides unified mechanism for the creation and lifetime management of new accelerator types, but the semantics and creation descriptors of these types are to be defined in acceleration engine-specific extensions.

Accelerator objects can be supplied to kernels as arguments. Unless an extension defines a new OpenCL C language type for the engine-specific accelerator (e.g. sampler_t), accelerator objects can only be used in conjunction with built-in kernels. If an extension does define such a new type, or if it provides a mechanism to access accelerator data within kernels, then accelerator objects can act as regular kernel arguments as well.

2 Proposal

2.1 New Extension

This proposal adds a new vendor extension, named **cl_intel_accelerator**, which adds runtime APIs for accelerator object creation, query, and lifetime management functions.

If this extension is supported by an implementation, the above string will be present in the CL_PLATFORM_EXTENSIONS or CL_DEVICE_EXTENSIONS string described in *Table 4.3*, OpenCL v1.2 spec.

2.2 Header File

The proposed interfaces for this extension will be provided in the **cl_ext.h** header.

2.3 New Host Functions

```
cl_accelerator_intel
clCreateAcceleratorINTEL
(
    cl_context                      context,
    cl_accelerator_type_intel       accelerator_type,
    size_t                          descriptor_size,
    const void*                     descriptor,
    cl_int*                         errcode_ret
);

cl_int
clRetainAcceleratorINTEL
(
    cl_accelerator_intel            accelerator
);

cl_int
clReleaseAcceleratorINTEL
(
    cl_accelerator_intel            accelerator
);

cl_int
clGetAcceleratorInfoINTEL
(
    cl_accelerator_intel            accelerator,
    cl_accelerator_info_intel       param_name,
    size_t                          param_value_size,
    void*                           param_value,
    size_t*                         param_value_size_ret
);
```

2.4 New Tokens

Returned by **clCreateAcceleratorINTEL** and **clGetAcceleratorInfoINTEL**

```
CL_INVALID_ACCELERATOR_INTEL
CL_INVALID_ACCELERATOR_TYPE_INTEL
```

```
CL_INVALID_ACCELERATOR_DESCRIPTOR_INTEL
CL_ACCELERATOR_TYPE_NOT_SUPPORTED_INTEL
```

2.5 New Types

Parameter type of **clRetainAcceleratorINTEL**, **clReleaseAcceleratorINTEL**, **clGetAcceleratorInfoINTEL** and a return type of **clCreateAcceleratorINTEL**:

```
typedef     _cl_accelerator_intel*        cl_accelerator_intel;
```

Parameter type of *accelerator_type* for **clCreateAcceleratorINTEL**:

```
typedef     cl_uint                       cl_accelerator_type_intel;
```

Parameter type of *param_name* for **clCreateAcceleratorINTEL**:

```
typedef     cl_uint                       cl_accelerator_info_intel;
```

2.6 Additions and Changes to the OpenCL Platform Layer

None

2.7 Additions and Changes to the OpenCL Runtime Layer

2.7.1. Modify section 5.7.2 "Setting Kernel Arguments"

As an additional description for argument types to **clSetKernelArg**, add the following:

- If the argument is of type *cl_accelerator_intel*, the *arg_value* entry must be a pointer to the accelerator object.

As an additional description for argument values for **clSetKernelArg**, add the following:

- If the argument is of type *cl_accelerator_intel*, the *arg_size* value must be equal to sizeof(*cl_accelerator_intel*).

As a non-success return value for **clSetKernelArg**, add the following:

- CL_INVALID_ACCELERATOR_INTEL for an argument declared to be of type *cl_accelerator_intel* when the specified *arg_value* is not a valid accelerator object, or, when *arg_value* is not equal to sizeof(*cl_accelerator_intel*).
- CL_INVALID_ACCELERATOR_TYPE_INTEL for an argument declared to be of type *cl_accelerator_intel* when the accelerator type of the specified *arg_value* does not match the specific accelerator type of that kernel argument.

2.7.2. Add a new section 5.14 "Accelerators"

Accelerator objects represent the programmable state of an acceleration processor, engine, or algorithm. Accelerator objects can be supplied to kernels as arguments. Unless an extension defines a new OpenCL C language type for the engine-specific accelerator (e.g. sampler_t), accelerator objects can

only be used in conjunction with built-in kernels. In this section, we discuss how accelerator objects are created and managed using OpenCL runtime API functions.

2.7.3. Add a new section 5.14.1 "Creating Accelerators"

The function

```
cl_accelerator_intel
clCreateAcceleratorINTEL
(
    cl_context                  context,
    cl_accelerator_type_intel   accelerator_type,
    size_t                      descriptor_size,
    const void*                 descriptor,
    cl_int*                     errcode_ret
);
```

creates an accelerator object. The accelerator object will be created with a reference count of one. Accelerator objects created with this function have semantics defined by the parameter *accelerator_type*, which are defined and described by extensions external to this document.

context must be a valid OpenCL context.

accelerator_type specifies the type of accelerator object created. The type constants are defined by acceleration engine-specific extensions. It is encouraged that extensions follow the naming scheme of *CL_ACCELERATOR_TYPE_{name}_INTEL* where {name} is a descriptive acceleration engine string.

descriptor_size is a value which specifies of the size, in bytes, of the immediately following descriptor structure.

descriptor is a pointer to a structure that defines the parameter set of the accelerator object. This parameter set describes the configurable state of the underlying object. The actual structure supplied must be consistent with *accelerator_type*. The descriptor structures are defined by acceleration engine-specific extensions. It is encouraged that extensions follow the naming scheme of *cl_{name}_desc_INTEL* where {name} is a descriptive acceleration engine string.

errcode_ret will return an appropriate error code. If *errcode_ret* is NULL, then no error value will be returned.

Errors
clCreateAcceleratorINTEL returns a valid non-zero accelerator object, and *errcode_ret* is set to CL_SUCCESS if the accelerator object is created successfully. Otherwise, it returns a NULL value with one of the following error values returned in *errcode_ret*:

- CL_INVALID_CONTEXT if *context* is not a valid context.
- CL_INVALID_ACCELERATOR_TYPE_INTEL if the supplied accelerator type is not valid.
- CL_INVALID_ACCELERATOR_DESCRIPTOR_INTEL if values specified in *accelerator_desc* are not valid (or a combination of values is not valid) or if *accelerator_desc* is NULL.
- CL_ACCELERATOR_TYPE_NOT_SUPPORTED_INTEL if the supplied accelerator type is not supported by the context.

- CL_INVALID_OPERATION if none of the devices in the context support accelerator objects.
- CL_OUT_OF_RESOURCES if there is a failure to allocate resources required by the OpenCL implementation on the device.
- CL_OUT_OF_HOST_MEMORY if there is a failure to allocate resources required by the OpenCL implementation on the host.

The function

```
cl_int
clRetainAcceleratorINTEL
(
    cl_accelerator_intel accelerator
);
```

increments the *accelerator* reference count. **clCreateAcceleratorINTEL** does an implicit retain.

accelerator is a valid accelerator object.

Errors
clRetainAcceleratorINTEL returns CL_SUCCESS if the function is executed successfully. Otherwise, it returns one of the following errors:

- CL_INVALID_ACCELERATOR_INTEL if *accelerator* is a not a valid accelerator object.
- CL_OUT_OF_RESOURCES if there is a failure to allocate resources required by the OpenCL implementation on the device.
- CL_OUT_OF_HOST_MEMORY if there is a failure to allocate resources required by the OpenCL implementation on the host.

The function

```
cl_int
clReleaseAcceleratorINTEL
(
    cl_accelerator_intel accelerator
);
```

decrements the *accelerator* reference count. The accelerator object is deleted after the reference count becomes zero and commands queued for execution on a command-queue(s) that reference *accelerator* have finished.

accelerator is a valid accelerator object.

Errors
clReleaseAcceleratorINTEL returns CL_SUCCESS if the function is executed successfully. Otherwise, it returns one of the following errors:

- CL_INVALID_ACCELERATOR_INTEL if *accelerator* is a not a valid accelerator object.

- CL_OUT_OF_RESOURCES if there is a failure to allocate resources required by the OpenCL implementation on the device.
- CL_OUT_OF_HOST_MEMORY if there is a failure to allocate resources required by the OpenCL implementation on the host.

2.7.4. Add a new section 5.14.2 "Accelerator Queries"

The function

```
cl_int
clGetAcceleratorInfoINTEL
(
    cl_accelerator_intel          accelerator,
    cl_accelerator_info_intel     param_name,
    size_t                        param_value_size,
    void*                         param_value,
    size_t*                       param_value_size_ret
);
```

returns information about the accelerator object.

accelerator specifies the accelerator object being queried.

param_name specifies the information to query. The list of supported param_name types and the information returned in param_value by clGetAcceleratorInfo is described in the table below.

param_value is a pointer to memory where the appropriate result being queried is returned. If param_value is NULL, it is ignored.

param_value_size is used to specify the size in bytes of the memory pointed to by *param_value*. This size is dependent on the *param_name*, and must be ≥ size of return type as described in the table below.

cl_accelerator_info_intel	Return type	Description
CL_ACCELERATOR_DESCRIPTOR_INTEL		Return the complete descriptor structure supplied when the accelerator was created. Refer to the external extension document where the descriptor is defined.
CL_ACCELERATOR_REFERENCE_COUNT_INTEL	cl_uint	Return the reference count associated with the accelerator
CL_ACCELERATOR_CONTEXT_INTEL	cl_context	Return the context specified when the accelerator was created.
CL_ACCELERATOR_TYPE_INTEL	cl_uint	Return the accelerator type specified when the accelerator was created. Refer to the external extension document where the descriptor is defined.

param_value_size_ret returns the actual size in bytes of data queried by *param_value*. If *param_value_size_ret* is NULL, it is ignored.

Errors

clGetAcceleratorInfoINTEL returns CL_SUCCESS if the function is executed successfully. Otherwise, it returns one of the following errors:

- CL_INVALID_VALUE if *param_name* is not valid or if size in bytes specified by *param_value_size* is < size of return type as described in the table above and *param_value* is not NULL.
- CL_INVALID_ACCELERATOR_INTEL if *accelerator* is a not a valid accelerator object.
- CL_OUT_OF_RESOURCES if there is a failure to allocate resources required by the OpenCL implementation on the device.
- CL_OUT_OF_HOST_MEMORY if there is a failure to allocate resources required by the OpenCL implementation on the host.

2.7.5. Add a new section 5.14.3 "Accelerator Descriptors"

Descriptors are defined by acceleration engine-specific extensions.

2.7.6. Add a new section 5.14.3 "Using Accelerators"

Accelerator semantics and usage are defined by acceleration engine-specific extensions.

2.8 Additions and Changes to the OpenCL Language

None

3 Interactions with Other Extensions

All follow-on/external accelerator extension documents, extend this base accelerator document, and must use the host-side functions defined within this document.

4 Examples

See the cl_motion_estimation_INTEL extension spec for an example usage of accelerators with built-in kernels.