Intel® FPGA host pipe extension for OpenCL™ applications

Michael Kinsner, Dirk Seynhaeve

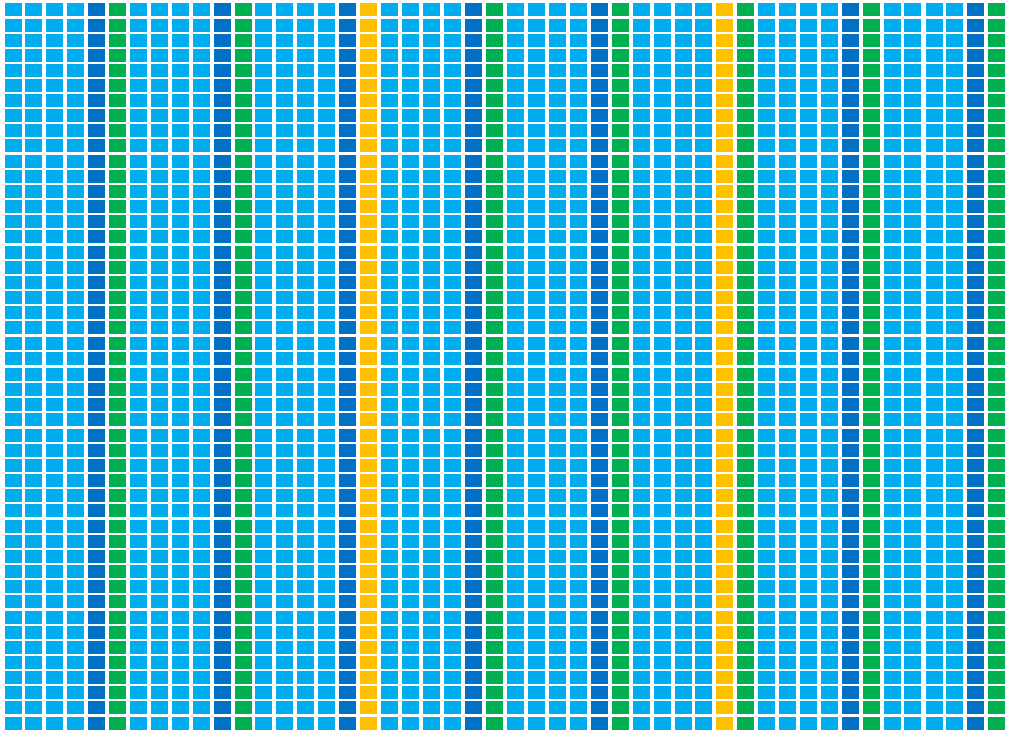
IWOCL 2018

Topics

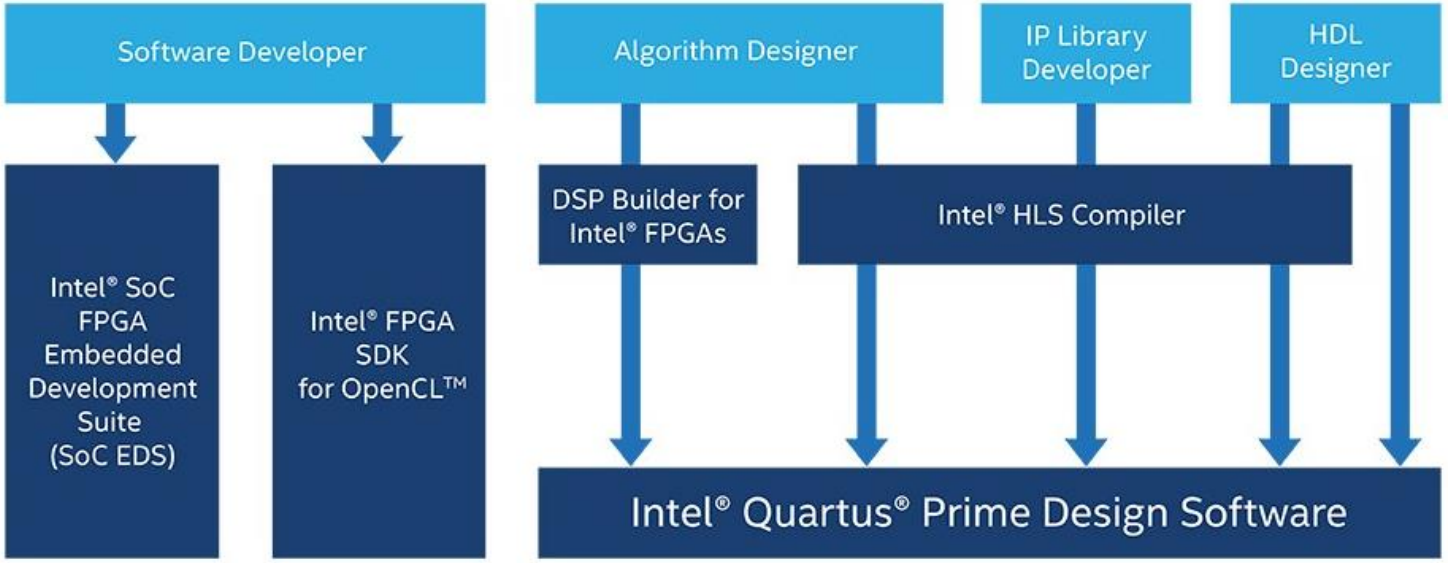
1. FPGA overview
2. Motivating application classes
3. Host pipes
4. Some data

FPGA: Fine-grained Massive Parallelism

Intel® Stratix® 10 FPGA:
Over 5 Million Basic Elements!



Many Design Entry Options



OpenCL defines the full system
(e.g. host, memory hierarchy, host ↔ device communication)

https://www.altera.com/products/design-software/overview.html

How Can You Compile Software to Hardware?

OpenCL™ Code

```
kernel void
add( global int* Mem ) {
   ...
   Mem[100] += 42*Mem[101];
}
```

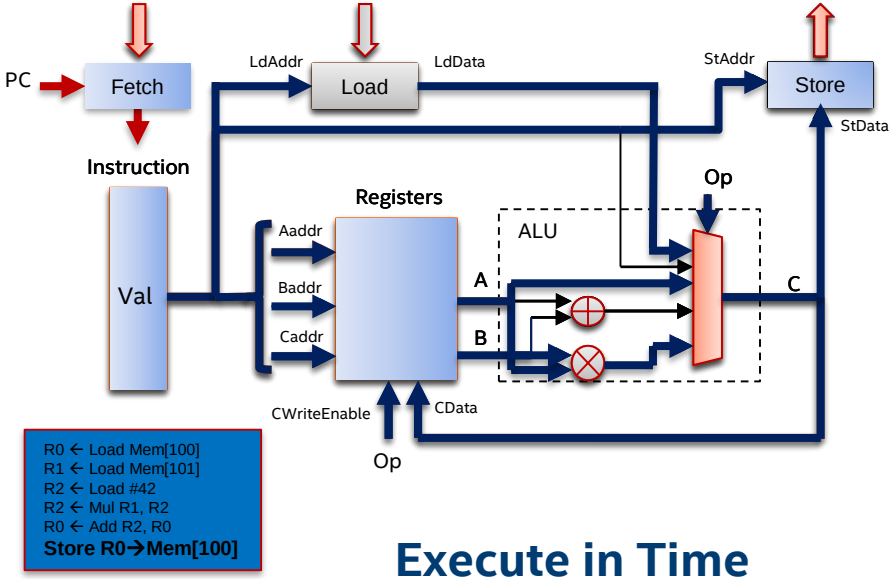
Instruction Level

```
add:
        R0 ← Load Mem[100]
        R1 ← Load Mem[101]
        R2 ← Load #42
        R2 ← Mul R1, R2
        R0 ← Add R2, R0
        Store R0 → Mem[100]
```

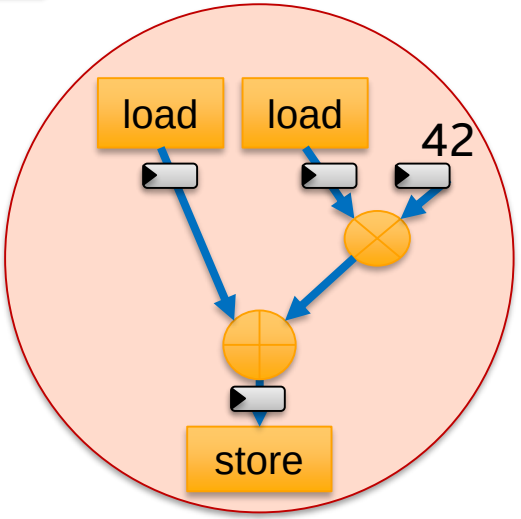
- Instruction sequence can either be executed sequentially in time (*temporal computing*), or executed in parallel in space (*spatial computing*)

Executing in Time vs. Executing in Space

```
R0 ← Load Mem[100]
R1 ← Load Mem[101]
R2 ← Load #42
R2 ← Mul R1, R2
R0 ← Add R2, R0
Store R0 → Mem[100]
```



Execute in Time
CPU, GPU



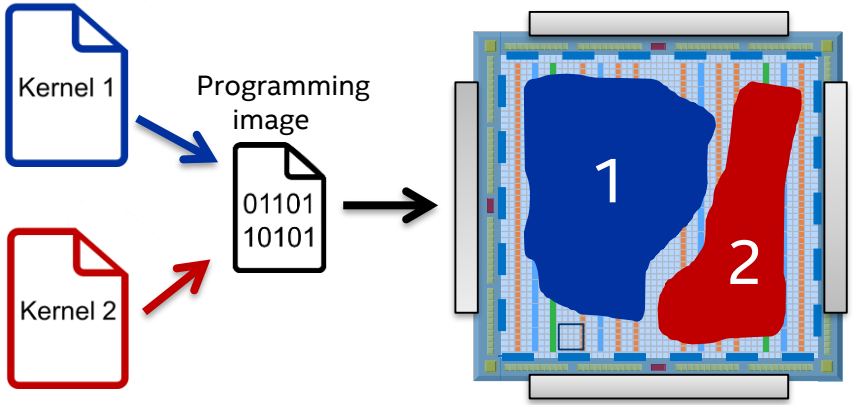
Fine grained pipeline parallelism

Execute in Space
FPGA

Kernels consume “space”

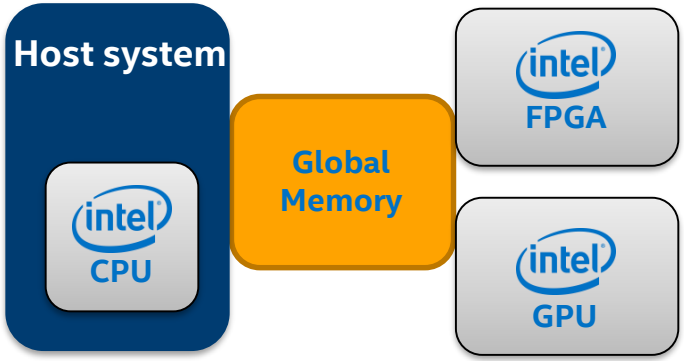
Conceptually map to regions of the FPGA in Intel’s OpenCL implementation

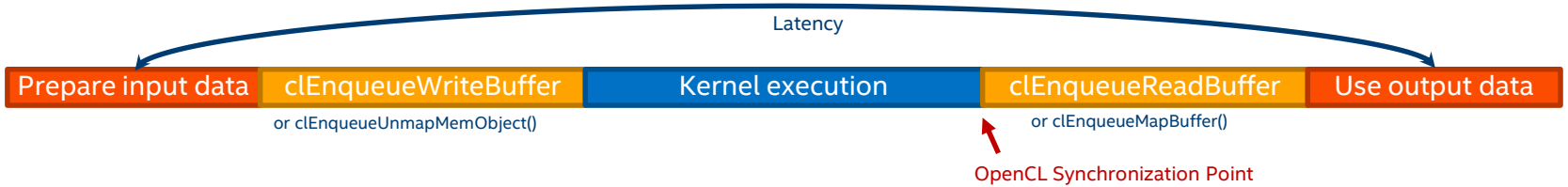
- Pipeline, data, and task parallelism

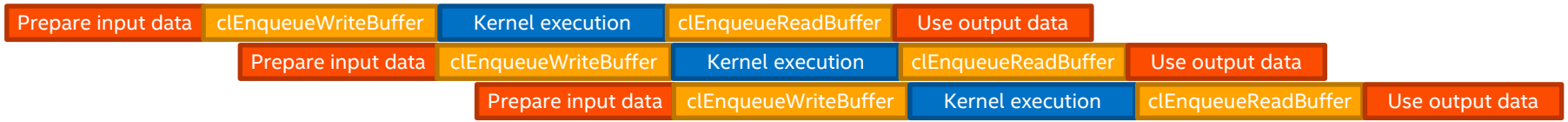


- Efficient use of FPGA architecture
- “Concurrent execution”
- Data flow processing
- Fine grained on-chip communication

OpenCL 1.x – Host/Device Bulk Communication



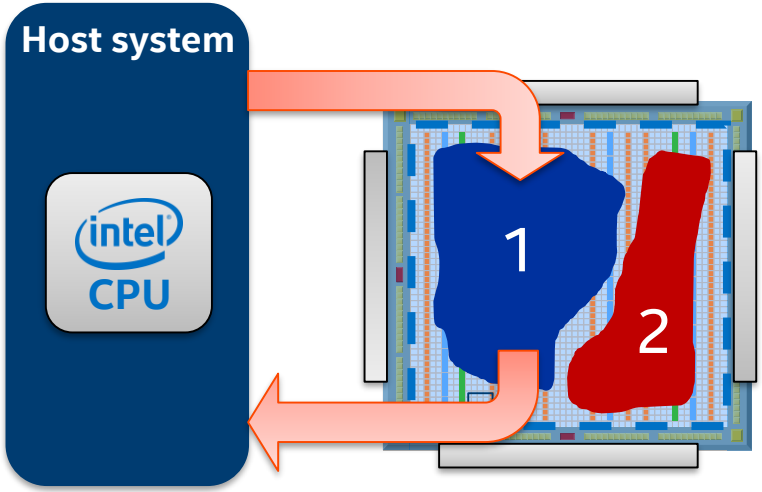




Motivating Classes of Applications

Long Running Kernel for Bursting or Massive Data

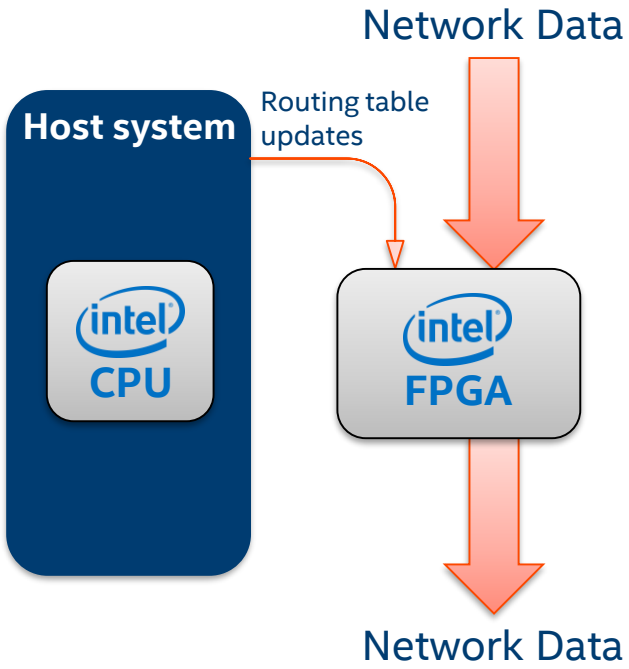
Long running / persistent kernel



- Processing more data than fits in global memory, using a single kernel instance
  - Or for lower latency processing of data arriving piecemeal on host
- Reduced latency processing of bursting data
  - Avoid launch overhead and state reconstruction

Network Routing / Processing

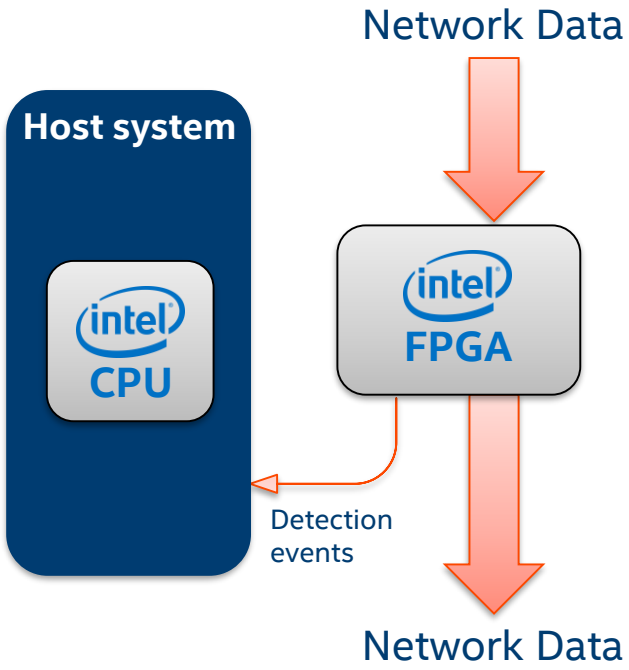
Routing table updates – low rate, non-periodic



- FPGAs have rich I/Os
- Often want long-running kernels
- Polling for memory-based host updates expensive
  - Plus memory consistency challenges
- FIFO semantics ideal

Streaming Content Analysis

Low rate, non-periodic detection events signaled to host

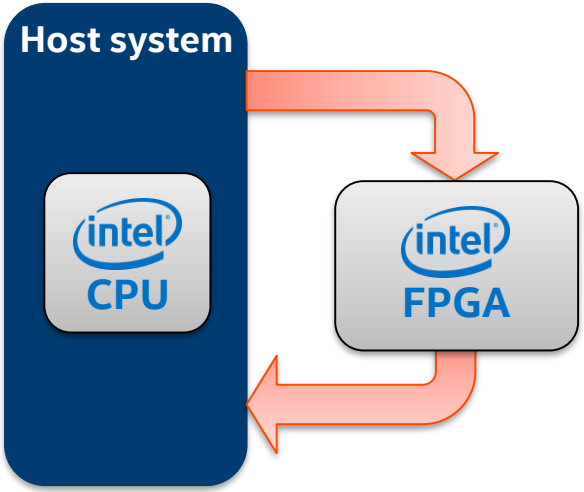


- Rich I/Os
- Long running kernels

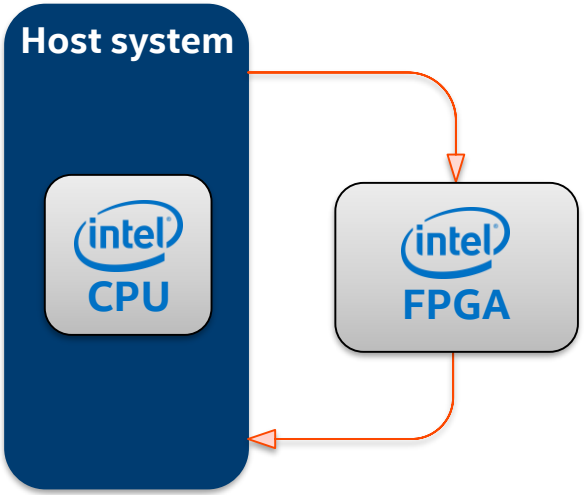
- Data consistency challenges
- FIFO semantics ideal

Two Use Models

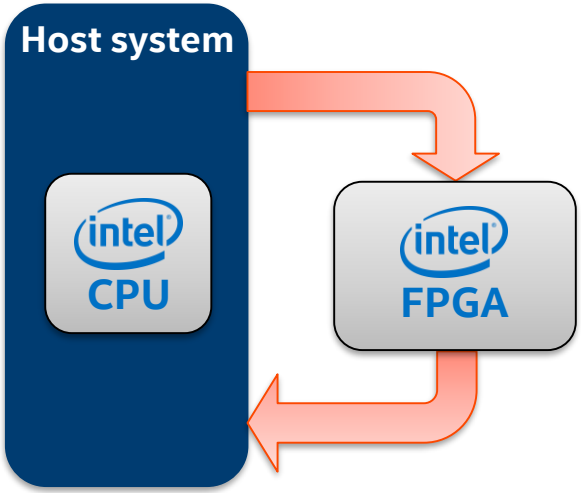
High throughput streaming



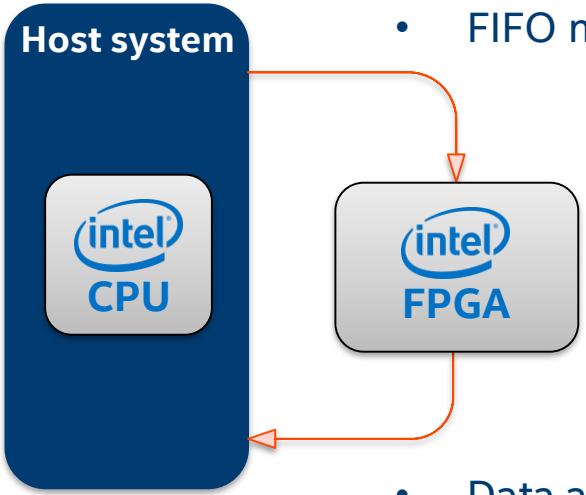
Asynchronous signaling/control



Two Use Models – OpenCL 1.x Challenges



- Data availability
- Cost of memory polling



- Data availability
- Cost of memory polling
- FIFO model

- Data availability
- FIFO model

OpenCL 2.0 Pipes – A Reminder

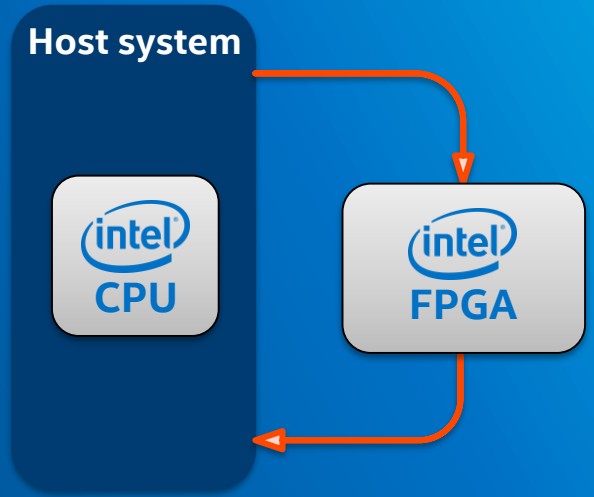
Kernels

```
kernel void producer (write_only pipe uint c0) {
      for (uint i=0;i<10;i++) {
            write_pipe( c0, &i );
      }
}

kernel void consumer (read_only pipe uint c0, global uint * restrict dst) {
      for (int i=0;i<5;i++) {
            read_pipe( c0, &dst[i] );
      }
}
```

Host

```
...
cl_mem pipe = clCreatePipe(context, 0, sizeof(cl_int), SIZE, NULL, &status);
status = clSetKernelArg(producer, 0, sizeof(cl_mem), &pipe);
status = clSetKernelArg(consumer, 0, sizeof(cl_mem), &out_buffer);
status = clSetKernelArg(consumer, 1, sizeof(cl_mem), &pipe);
...
```

OpenCL 2.0 pipes: Communication is **between kernels**



Intel® Host Pipes

Allow pipes to be read/written from the **host program** as well as in kernels

Host Pipe Extension

Small extension to OpenCL 2.x pipe API: cl_intel_fpga_host_pipe

- New flags legal in **clCreatePipe()**:

CL_MEM_HOST_READ_ONLY	Set host visibility
CL_MEM_HOST_WRITE_ONLY	
CL_MEM_READ_ONLY	Optional – From device perspective
CL_MEM_WRITE_ONLY	

Host program:

```
cl_mem read_pipe = clCreatePipe(
    context,
    CL_MEM_HOST_READ_ONLY,
    sizeof( cl_int ),
    128, // Number of packets that can be buffered
    NULL,
    &error
);
```

- New query / status enums:

API Enum	Parent Function
CL_KERNEL_ARG_HOST_ACCESSIBLE_PIPE_INTEL	clGetKernelArgInfo()
CL_DEVICE_MAX_HOST_READ_PIPES_INTEL	clGetDeviceInfo()
CL_DEVICE_MAX_HOST_WRITE_PIPES_INTEL	clGetDeviceInfo()
CL_PIPE_FULL	clWritePipeIntelFPGA()
CL_PIPE_EMPTY	clReadPipeIntelFPGA()

Kernel Interface

Used like normal OpenCL 2.x pipes

- Additional kernel argument attribute
- No reservation functionality (the OpenCL 2.x feature)

C kernel language:

```
kernel void foo (__attribute__((intel_host_accessible)) write_only pipe int p ) { .... }
read_pipe(P, &val)
write_pipe(P, &val)
```

C++ kernel language:

```
kernel void foo( [[cl::intel_host_accessible]] cl::pipe<int, cl::pipe_access::write> p ) { .... }
p.write( val )
p.read( &val )
```

Host Interface – Low Rate Signaling

Simple interface

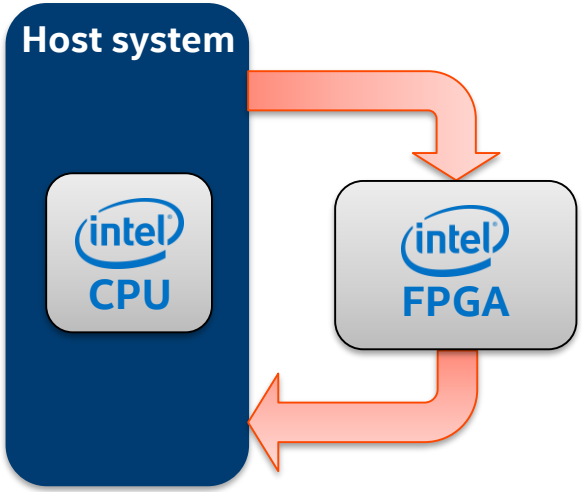
- Single word read/write
- Data transferred “as soon as possible”

```
cl_int clReadPipeIntelFPGA( cl_mem pipe, void *ptr );
cl_int clWritePipeIntelFPGA( cl_mem pipe, const void *ptr );
```

```
// Create pipes, kernels, other startup code
....

// Bind pipes to kernels
clSetKernelArg(read_kern, 0, sizeof(cl_mem), (void *)&write_pipe);
clSetKernelArg(write_kern, 0, sizeof(cl_mem), (void *)&read_pipe);

// Enqueue kernels
....

int float2;
if ( !clReadPipeIntelFPGA( read_pipe, &val )) {
    int result = clWritePipeIntelFPGA( write_pipe, (int)(val.x + val.y));
    // Check write success/failure and handle
    ....
}
```

Host Interface – High Throughput

```
void * clMapHostPipeIntelFPGA(
  cl_mem pipe,
  cl_map_flags map_flags,
  size_t requested_size,        <-- What you want
  size_t * mapped_size,         <-- What you got
  cl_int * errcode_ret );

cl_int clUnmapHostPipeIntelFPGA(
  cl_mem pipe,
  void * mapped_ptr,
  size_t requested_size,        <-- What you want
  size_t * unmapped_size );     <-- What you got
```



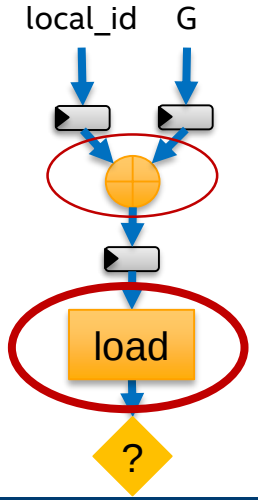

```
…
cl_int *buffer;

buffer = (cl_int*) clMapHostPipeIntelFPGA( pipe, 0, ask_size, &got_size, &status );

// Write data to buffer
…

clUnmapHostPipeIntelFPGA( pipe, buffer, buffer_size, NULL );
```

FIFO Access Within Kernels

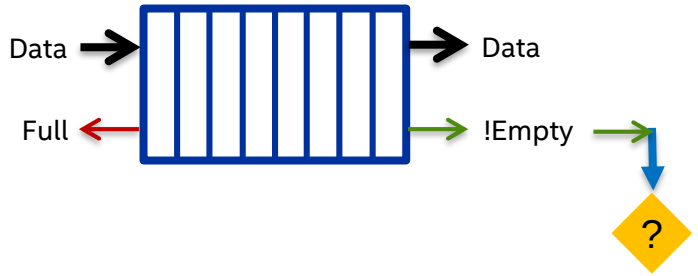
Checking FIFO for data availability is cheap

- Implicit control signals (ready/full), and low latency

```
kernel void
foo ( global int *G, … ) {
      if (G[ get_local_id(0) ]) { … }
}
```




```
kernel void
foo (read_only pipe int4 P … ) {
      int4 val;
      if (0 == read_pipe(P, &val)) { … }
}
```



Visibility and Latency

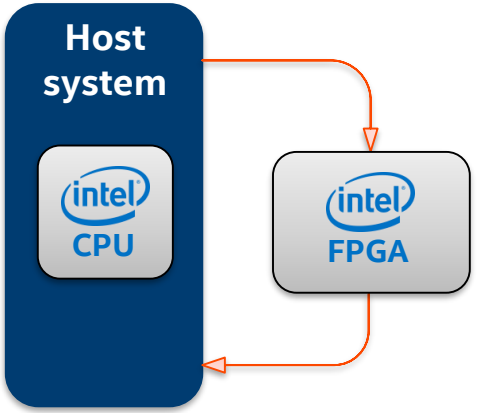
Additional memory model guarantee

- Data written to a pipe will eventually be visible on the read endpoint, without an OpenCL synchronization point. It is understood that an OpenCL implementation will make the data visible to the read endpoint “as soon as possible”
- No synchronization side effects with other pipes or memory

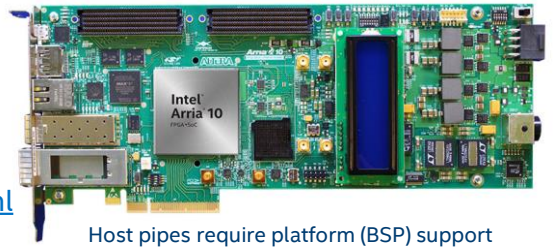
The host pipe API supports low latency communication

- An OpenCL extension is not enough to guarantee latency
  - Board support package
  - Drivers/OS
  - System load

- The host pipe API was designed to enable latency-sensitive applications
  - Talk to board and system provider if guarantees are required

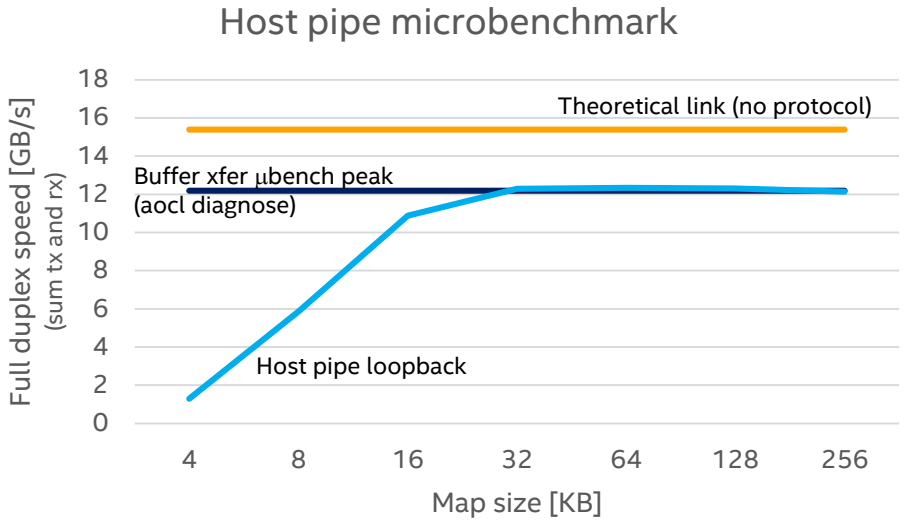


Host Pipe Microbenchmark



Host pipes require platform (BSP) support

- Results from an Intel® Arria® 10 GX FPGA Development Kit
  - https://www.altera.com/products/boards_and_kits/dev-kits/altera/kit-a10-gx-fpga.html
- Intel(R) Xeon(R) CPU E5-1650 v3 @ 3.50GHz
- Two CPU threads, each managing one host pipe direction. Loopback kernel
- 'aocl diagnose': Buffer transfer speed μbenchmark, that ships with the Intel® FPGA SDK for OpenCL™

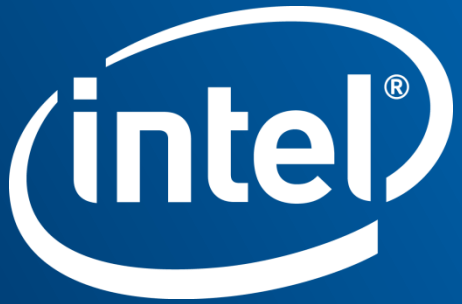


Now Available!

https://www.altera.com/products/design-software/embedded-software-developers/opencl/overview.html

Host pipes are shipping in the Intel® FPGA SDK for OpenCL™ 18.0

- Reference platform has some minor restrictions that will be relaxed in the future
  - # host pipes, width of host pipes
  - Some queries
- https://www.altera.com/products/design-software/embedded-software-developers/opencl/overview.html


intel®

Legal Notice and Disclaimers

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit **http://www.intel.com/benchmarks**.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

** Other names and brands may be claimed as the property of others.*

Legal Disclaimer and Optimization Notice

INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804