OpenCL on an FPGA

Elías Todorovich
etodorov@exa.unicen.edu.ar
July 2013

Challenges

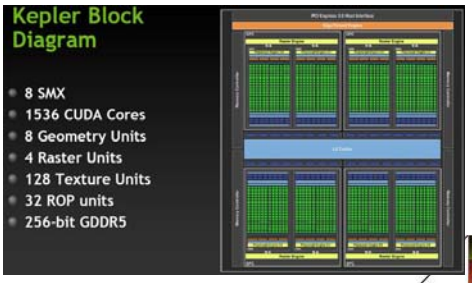
- Complexity
- Time-to-market
- Speed
- Productivity
- Vendor specific APIs
- CPU – GP/GPU - FPGA Programming gap

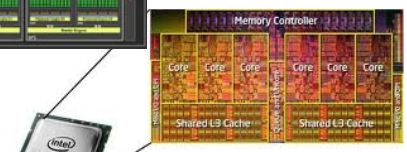
After the “power wall” performance comes from parallelism.

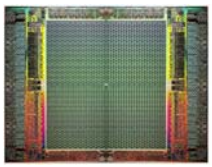
Different types of apps.

- Control
  - Searching
  - Parsing
- Data intensive
  - Image processing
  - Data mining
- Compute intensive
  - Iterative methods
  - Financial modeling

- Select most suitable HW







Types of parallelism

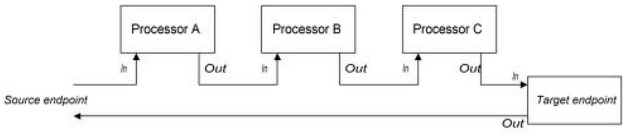
- Data parallelism (scatter-gather)

```
for (i=0; i<N; i++) c[i]=a[i]*b[i];
```

- Thread parallelism (divide and conquer)

```
task1; // CPU 1
task2; // CPU 2
```

- Pipeline parallelism (producer-consumer, streaming)



- Data sharing and synchronization must be considered!

C/C++ Synthesis Approach

- Several commercial offerings for C/C++/System C High-Level-Synthesis to RTL:
  - C-to-Silicon Compiler (Cadence)
  - Impulse C (Impulse Accelerated Technologies)
  - Catapult C (Mentor)
  - SynphonyC (Synopsys)
  - AutoESL (now Xilinx) -> Vivado 14.x
  - Cynthesizer (Forte),
  - NEC

Altera's approach

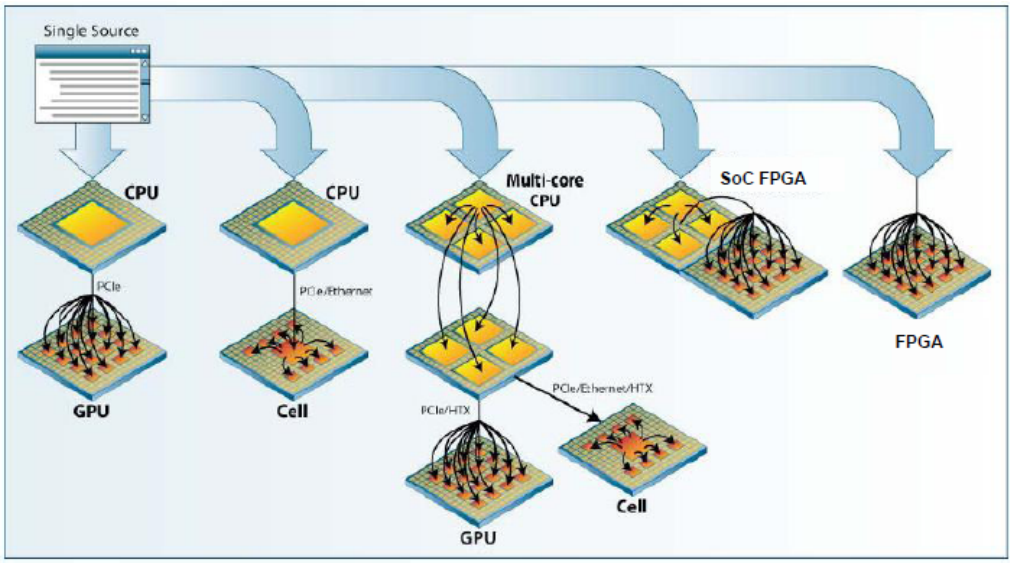
- Open Computing Language (OpenCL) is a standard cross-platform programing language based upon C for portable parallel applications.
  - Task parallel and data parallel applications
- Altera is using OpenCL on FPGA:

  + Combining OpenCL, a standard parallel programming language,

  + with the parallel performance capability of FPGAs

  = provides a powerful solution for **system acceleration**.

OpenCL Portability



Source: RapidMind

OpenCL Structure

- Natural separation between the code that runs on accelerators and the code that manages those accelerators.
  - The **host** code is pure software that can be executed on any sort of conventional microprocessor
    - Soft processor, Embedded hard processor, external x86 processor
  - The **kernel** code is C with a set of extensions (and restrictions) that allows for the specification of parallelism and memory hierarchy.

OpenCL by example

Consider a C program:

```
void inc (float* a, float
b, int N)
{for (int i=0, i<N; i++)
    a[i]+=b; }

void main()
{…
  increment (a,b,N);
…}
```

- In OpenCL we can parallelize the increment function.
- We do this by executing a “kernel”
  - A kernel is a function executed by many work-items.
  - Each work-item has an ID used to index into arrays, etc.

OpenCL by example

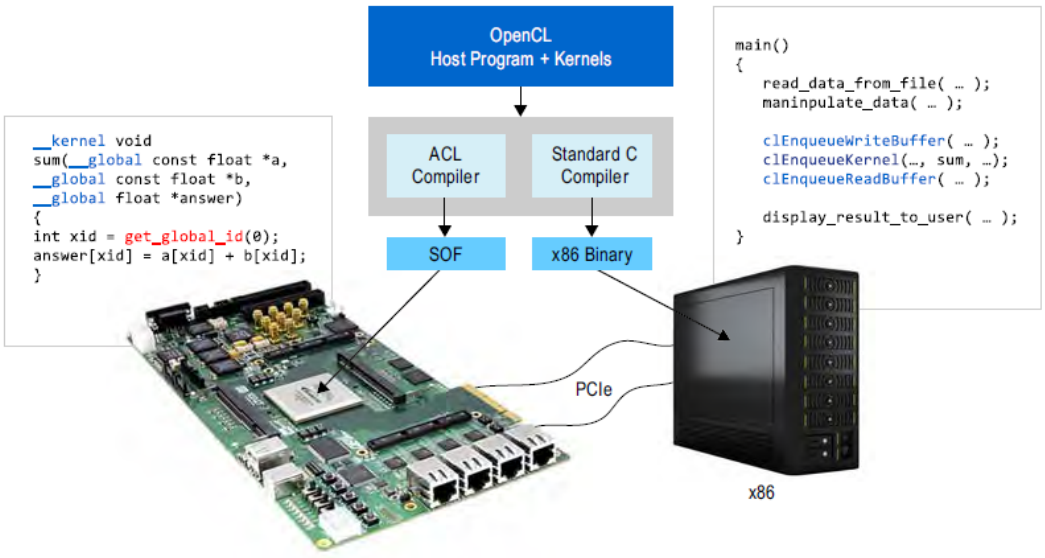
```
kernel
void inc (float* a, float
b, int N)
{int i=get_global_id(0);
 a[i]+=b; }

void main()
{…
  clEnqueueNDRangeKernel
(…,&N,…);
…}
```

- Kernel is executed by N work-items, each has an ID between 0 and N-1
- Parallelism is explicit: kernels are executed by many work-items
- Hierarchy of work-items: work-items are grouped into work-groups
- Explicit memory hierarchy
  - Global memory visible to all work-groups and work-items
  - Local memory visible work-items into a work-group
  - Private memory visible only to a single work-item

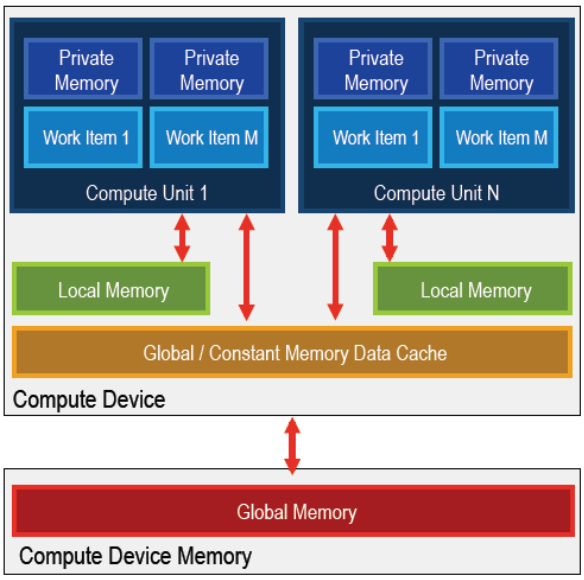
Architecture Model - Terminology

- Kernel – Smallest unit of execution, like a C function
- Host program – A collection of kernels
- Work item, an instance of kernel at run time
- Work group, a collection of work items

OpenCL for Altera devices



Architecture – Memory Model



OpenCL Software Stack

- **Host program**
  - Query compute devices
  - Create contexts

  → **Platform Layer**

  - Create memory objects associated to contexts
  - Compile and create kernel program objects
  - Issue commands to command-queue
  - Synchronization of commands
  - Clean up OpenCL resources

  → **Runtime**

- **Kernels**
  - C code with some restrictions and extensions

  → **Language**

Structure of OpenCL host program

Get information about platform and devices available on system

Select devices to use

Create an OpenCL command queue

Create memory buffers on device

Transfer data from host to device memory buffers

Create kernel program object

Build (compile) kernel in-line (or load precompiled binary)

Create OpenCL kernel object

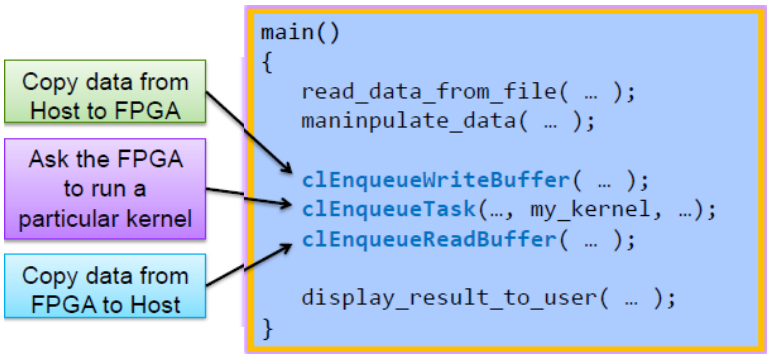
Set kernel arguments

Execute kernel

Read kernel memory and copy to host memory.

OpenCL Host Program

- Pure software written in standard C
- Communicates with the Accelerator Device via a set of library routines which abstract the communication between the host processor and the kernels



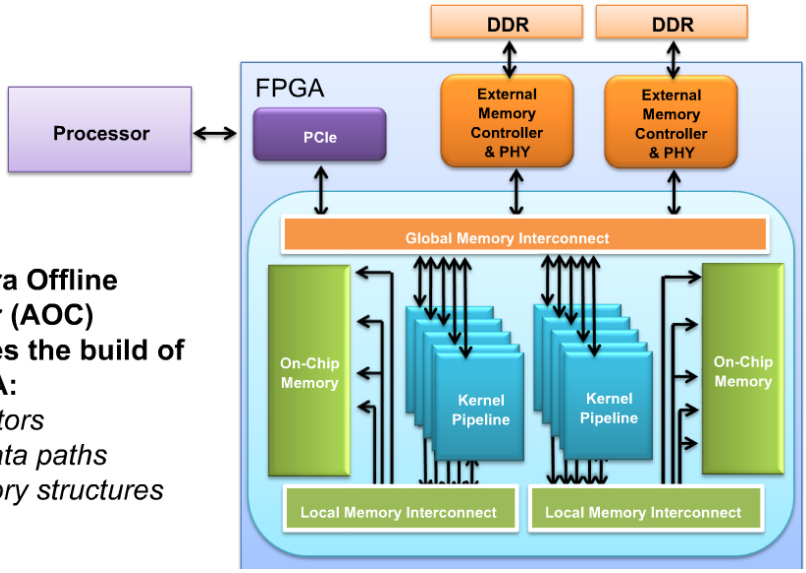
OpenCL

- Can make query on available devices and build a context of the available devices.
- That means the programmers would be able to program more freely for any kind of device.
- Also application would survive even if the hardware change in the future.

Kernels in FPGAs

- Kernels are written in standard C; however, they are annotated with constructs to specify
  - parallelism and
  - memory hierarchy.
- Unlike CPUs and GPUs, in FPGAs kernel functions can be transformed into dedicated deeply pipelined hardware circuits that are inherently multithreaded.
  - Each of these pipelines can be replicated many times to provide even more parallelism.
- The compiler can create the circuitry to manage the communication to the external system (e.g. DDR).
- Similarly, PCIe IP is automatically instantiated and connected to the kernel so that an x86 host can communicate with the FPGA accelerator via the OpenCL APIs.

OpenCL Compiler for FPGA



The Altera Offline Compiler (AOC) completes the build of the FPGA:
-accelerators
-all the data paths
-all memory structures

Conclusions

- HLS is an interesting solution for the *performance+productivity* problem nowadays.
- You might find OpenCL difficult. But when you master it, you will master parallel computing.

More info…

- David B. Kirk, Wen-mei W. Hwu, "Programming Massively Parallel Processors: A Hands-on Approach", Morgan Kaufmann; 2010, ISBN-10: 0123814723.
- Aaftab Munshi, Benedict Gaster, Timothy G. Mattson, James Fung, Dan Ginsburg, "OpenCL Programming Guide", Addison-Wesley, 2011. ISBN: 978-0-321-74964-2.
- Matthew Scarpino, "OpenCL in Action". Manning Publications, 2011. ISBN-10: 1617290173.
- Benedict Gaster, "Heterogeneous Computing with OpenCL", Morgan Kaufmann, 2012. ISBN-10: 0124058949.
- Altera's OpenCL Program: www.altera.com/OpenCL
- The Khronos Group—The OpenCL Standard: www.khronos.org/opencl

OpenCL on an FPGA

Elías Todorovich
etodorov@exa.unicen.edu.ar
July 2013