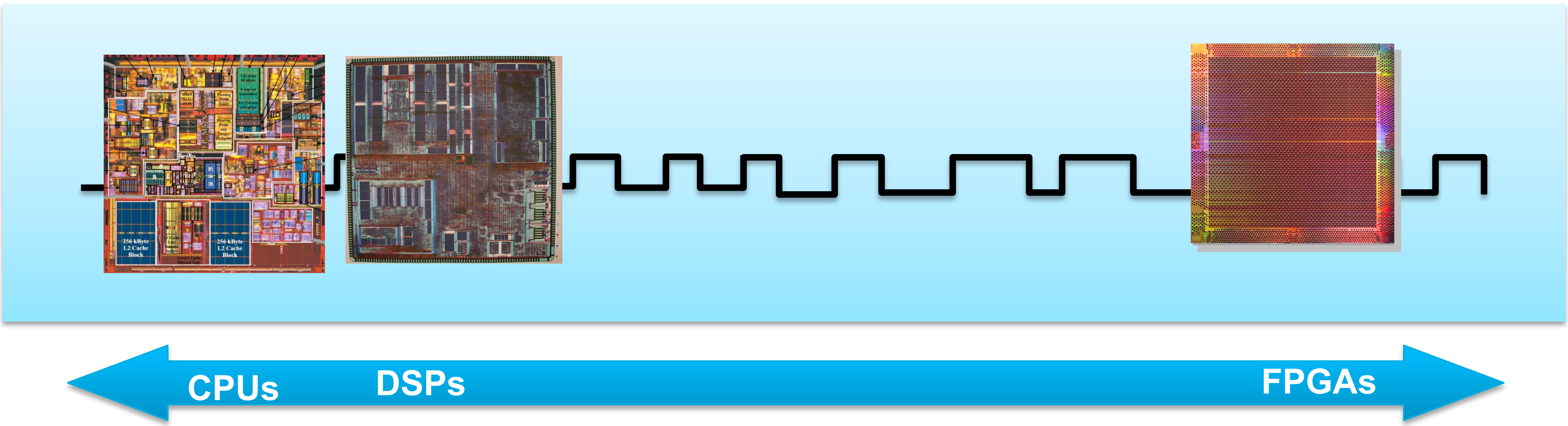
Higher Level Programming Abstractions for FPGAs using OpenCL

Desh Singh
Supervising Principal Engineer
Altera Corporation
Toronto Technology Center

Programmable Solutions: 1985-2002

- Technology scaling favors programmability

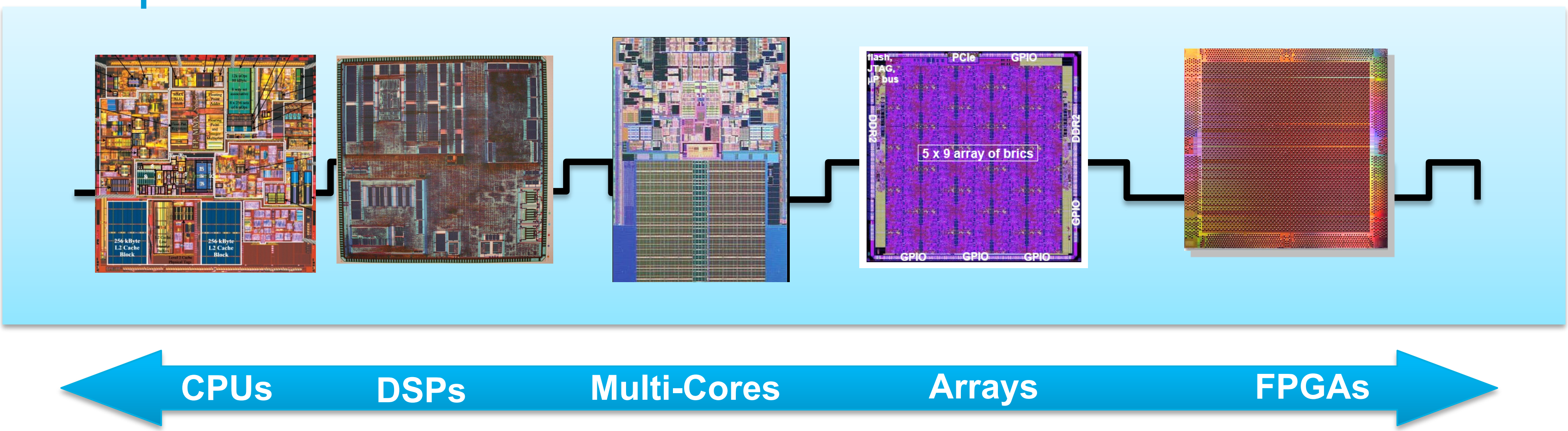


Single Cores

Fine-Grained Arrays

Programmable Solutions: 2002-20XX

- Technology scaling favors programmability and parallelism



Single Cores

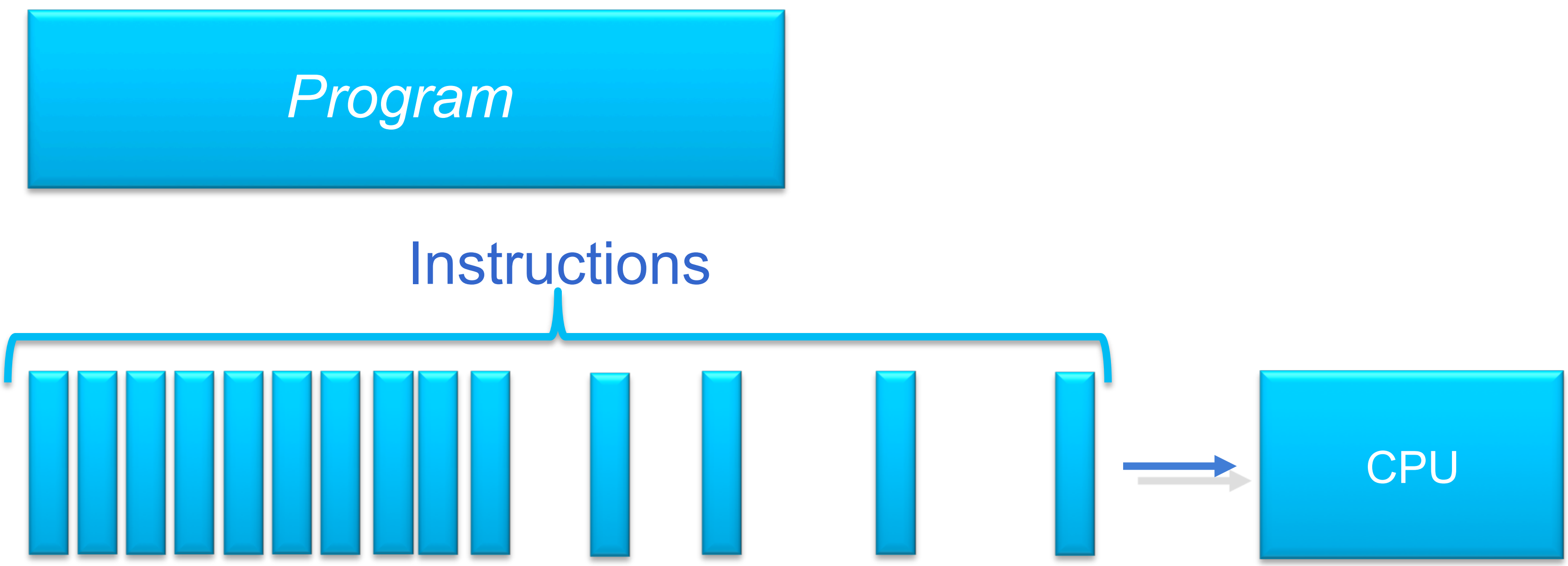
Multi-Cores
Coarse-Grained
CPUs and DSPs

Coarse-Grained
Massively
Parallel
Processor
Arrays

Fine-Grained
Massively
Parallel
Arrays

Programmable and *Sequential*

- Reaching the limit
  - After four decades of success…



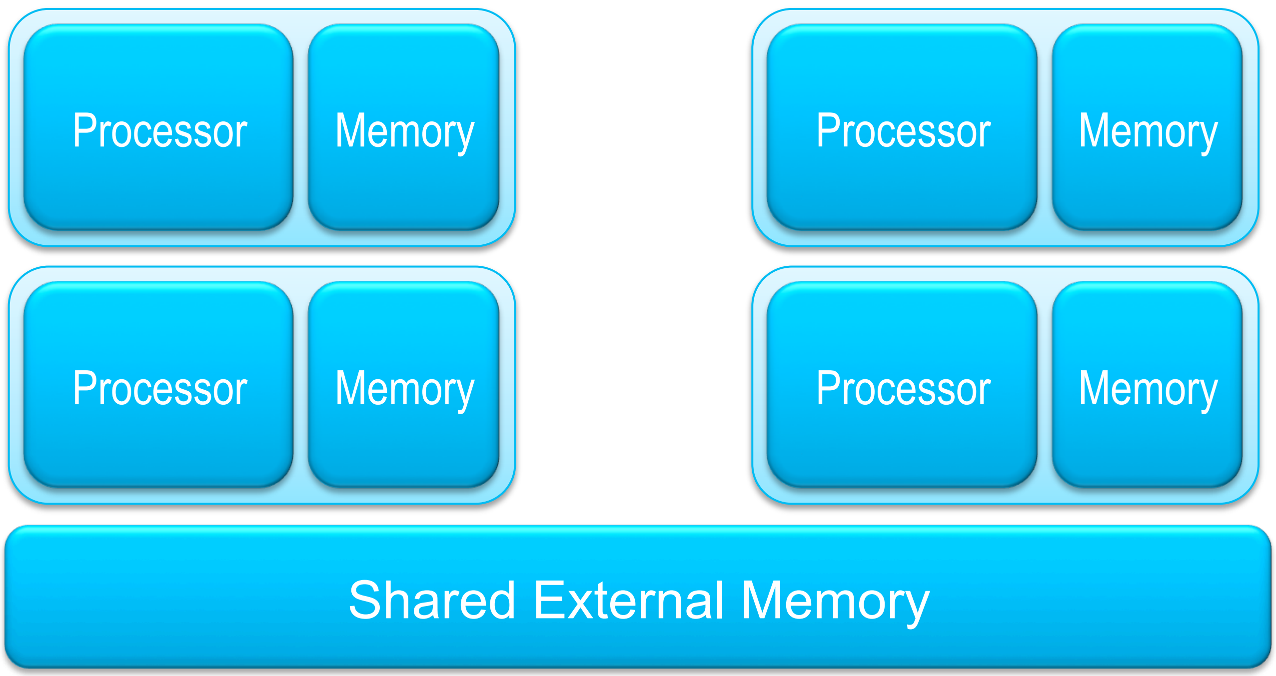
"The End of Denial Architecture" - William J. Dally [DAC'2009]

- Single thread processors are in denial about parallelism and locality
- They provide two illusions:

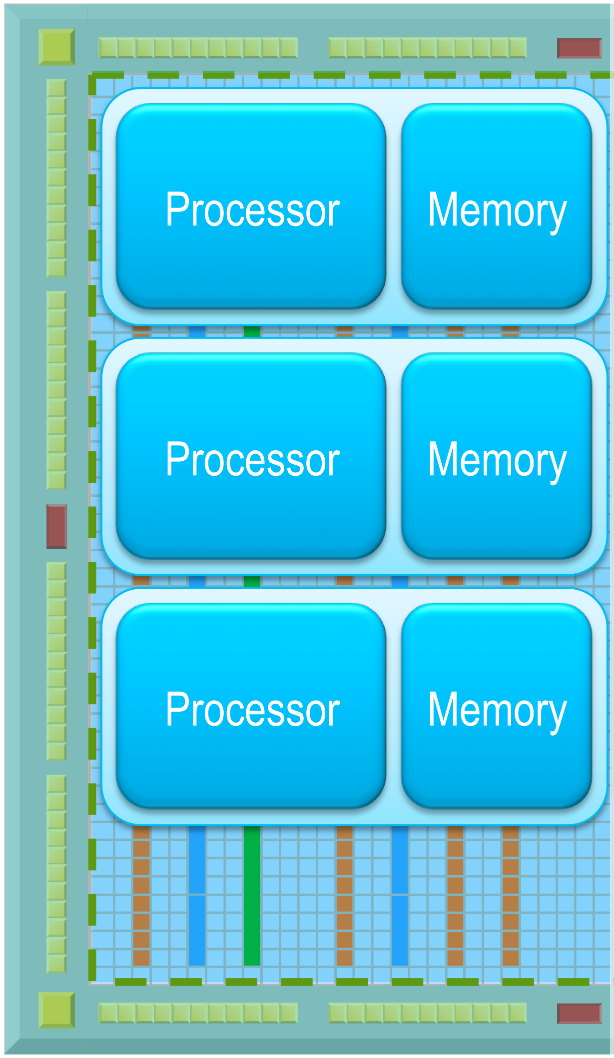
- Serial execution
  - Denies parallelism
  - Tries to exploit parallelism with ILP – limited scalability
- Flat memory
  - Denies locality
  - Tries to provide illusion with caches – very inefficient when working set doesn't fit in the cache

Programmable and *Parallel*

- Exploit parallelism on a chip
  - Take advantage of Moore's law
    - Processors not getting faster, just wider
  - Keep the power consumption down
- Use more transistors for information processing



FPGA : Ultimately Configurable Multicore



- Many coarse-grained processors
  - Different Implementation Options
    - Small soft scalar processor
    - or Larger vector processor
    - or Customized hardware pipeline
  - Each with local memory

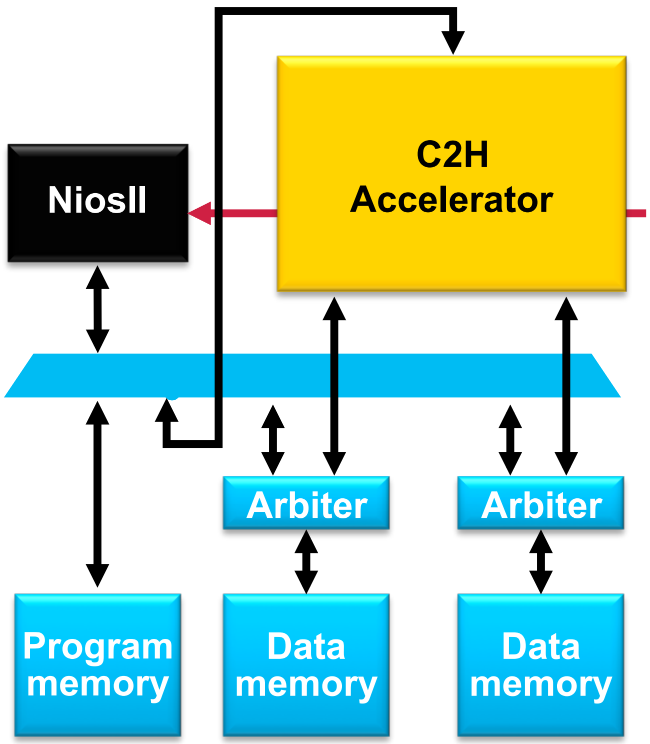
- Each processor can exploit the fine grained parallelism of the FPGA to more efficiently implement it's "program"

- Possibly heterogeneous
  - Optimized for different tasks

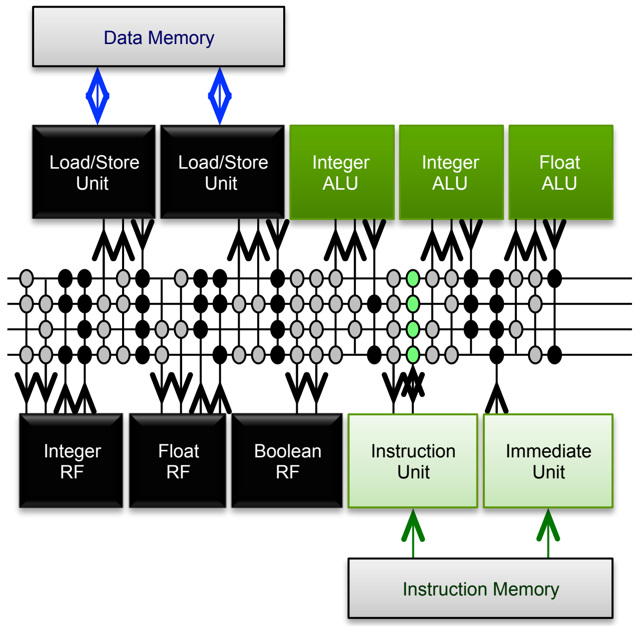
- Customizable to suit the needs of a particular application

Processor Possibilities

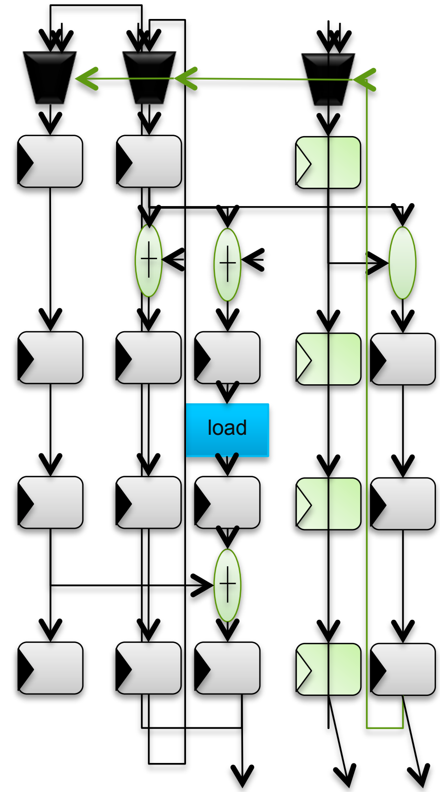
Scalar Soft Proc + Accelerator



VLIW / Vector / TTA Soft Proc



Custom Pipeline



Our challenges

- Generally, programmers have difficulty using FPGAs as massive multi-core devices to accelerate parallel applications

- Need a programming model that allows the designer to think about the FPGA as a configurable multi-core device

- Today, the FPGA's programming model revolves around RTL (VHDL / Verilog)
  - State machines, datapaths, arbitration, buffering, etc.

An ideal programming environment …

- Has the following characteristics:
  - Based on a standard multicore programming model rather than something which is FPGA-specific
  - Abstracts away the underlying details of the hardware
    - VHDL / Verilog are similar to “assembly language” programming
      - Useful in rare circumstances where the highest possible efficiency is needed
  - The price of abstraction is not too high
    - Still need to efficiently use the FPGA’s resources to achieve high throughput / low area
  - Allows for software-like compilation & debug cycles
    - Faster compile times
    - Profiling & user feedback

OPENCL :
BRIEF INTRODUCTION

What is OpenCL

- OpenCL is a programming model developed by the Khronos group to support silicon acceleration
- An industry consortium creating open API standards
- Enables software to leverage silicon acceleration
- Commitment to royalty-free standards
  - Making money from enabled products – not from the standards themselves

OpenCL

- OpenCL is a parallel language that provides us with two distinct advantages
  - Parallelism is declared by the programmer
    - Data parallelism is expressed through the notion of parallel threads which are instances of computational kernels
    - Task parallelism is accomplished with the use of queues and events that allow us to coordinate the coarse grained control flow
  - Data storage and movement is explicit
    - Hierarchical Memory model
      - Registers
      - Accelerator Local Memory
      - Global off-chip memory
    - It is up to the programmer to manage their memories and bandwidth efficiently

OpenCL Structure

- Natural separation between the code that runs on accelerators* and the code that manages those accelerators
  - The management or “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 minimal set of extensions that allows for the specification of parallelism and memory hierarchy
    - Likely only a small fraction of the total code in the application
    - Used only for the most computationally intensive portions

* Accelerator = Processor + Memory combo

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

Copy data from Host to FPGA

Ask the FPGA to run a particular kernel

Copy data from FPGA to Host

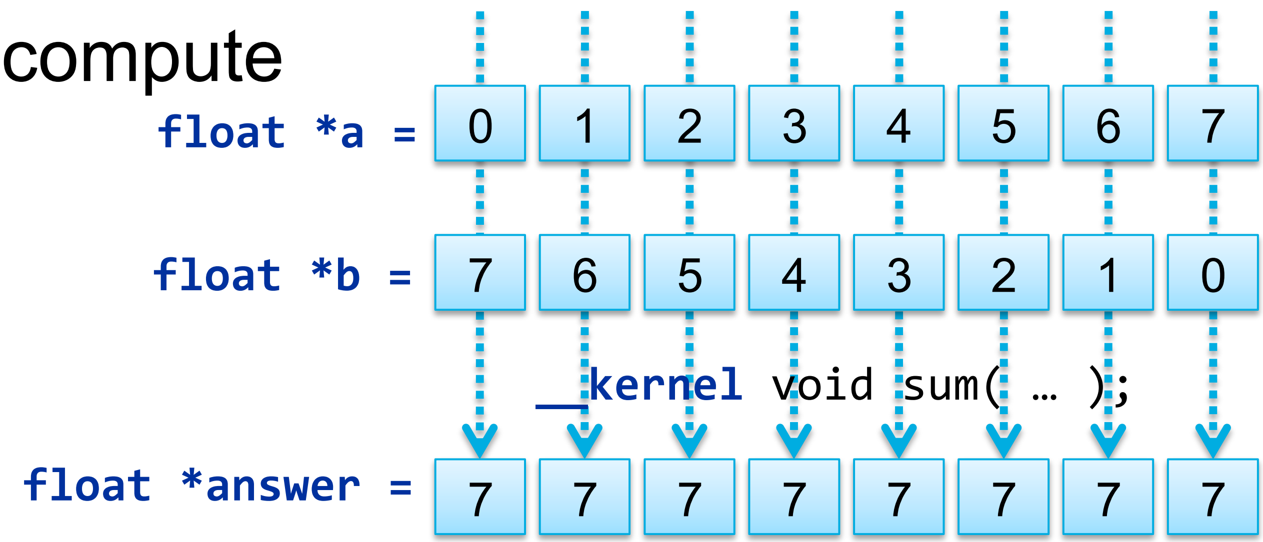
```
main()
{
   read_data_from_file( … );
   maninpulate_data( … );

   clEnqueueWriteBuffer( … );
   clEnqueueTask(…, my_kernel, …);
   clEnqueueReadBuffer( … );

   display_result_to_user( … );
}
```

OpenCL Kernels

- Data-parallel function
  - Defines many parallel threads of execution
  - Each thread has an identifier specified by “**get_global_id**”
  - Contains keyword extensions to specify parallelism and memory hierarchy

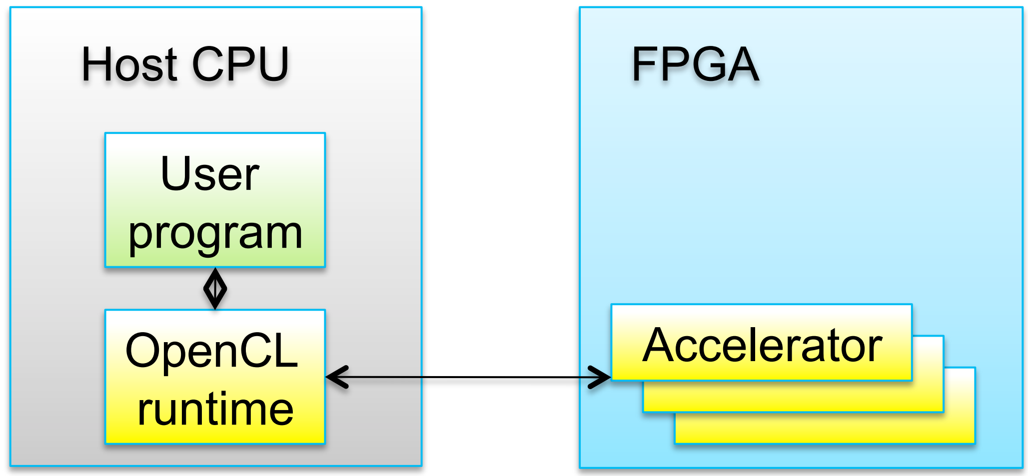
```
__kernel void
sum(__global const float *a,
__global const float *b,
__global float *answer)
{
int xid = get_global_id(0);
answer[xid] = a[xid] + b[xid];
}
```

- Executed by compute object
  - CPU
  - GPU
  - Accelerator

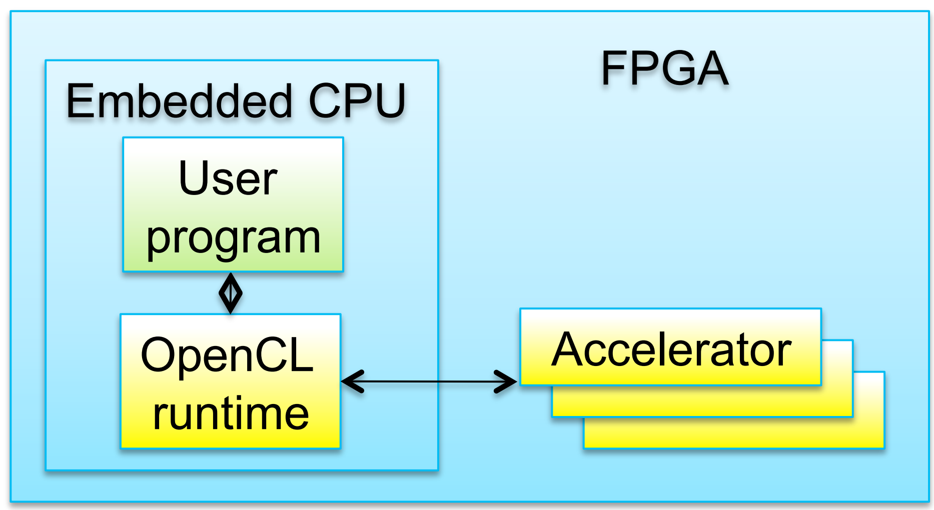


Altera supported system configurations

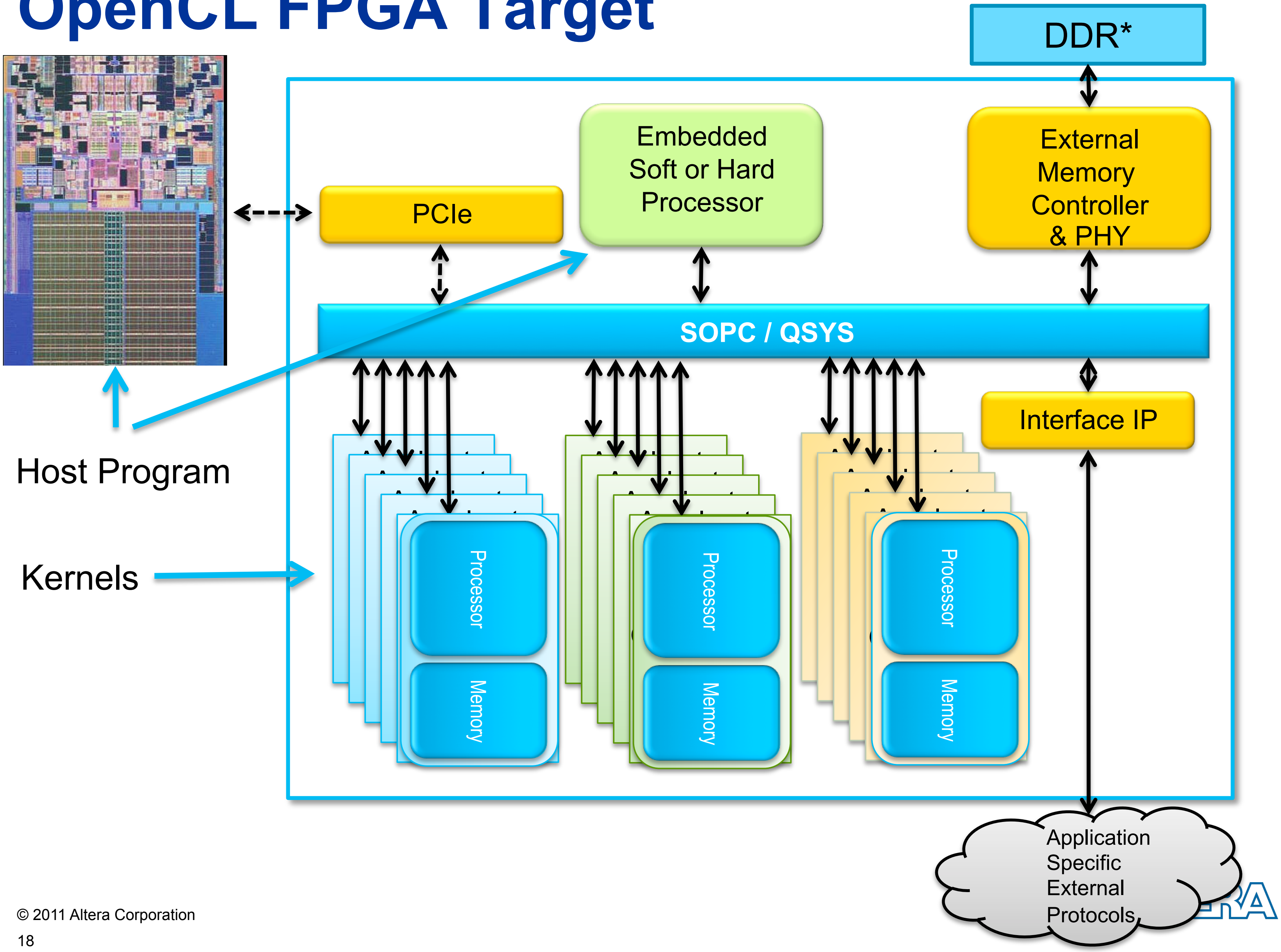
External host



Embedded



OpenCL FPGA Target



OpenCL : FPGA Programming Model

- OpenCL is a standard multi-core programming model that can be used to provide a higher-level layer of abstraction for FPGAs
- Research challenges abound
  - Need to collaborate with academics, third parties and members of the Khronos group
    - Require libraries, kernel compilers, debugging tools, pre-defined templates, etc.
  - We have to consider that our “competition” is no longer just other FPGA vendors
    - A broad spectrum of programmable multi-core devices targeting different market segments

Thank You

OpenCL Driving Forces

- Attempt at driving an industry standard parallel language across platforms
  - CPUs, GPUs, Cell Processors, DSPs, and even FPGAs

- So far driven by applications from:
  - The consumer space
    - Image Processing & Video Encoding
    - 1080p video processing on mobile devices
    - Augmented reality & Computational Photography
  - Game programming
    - More sophisticated rendering algorithms
  - Scientific / High Performance Computing
    - Financial, Molecular Dynamics, Bioinformatics, etc.

Challenges

- OpenCL's compute model targets an "abstract machine" that is not an FPGA
  - Hierarchical array of processing elements
  - Corresponding hierarchical memory structure

- It is more difficult to target an OpenCL program to an FPGA than targeting more natural hardware platforms such as CPUs and GPUs
  - These problems are research opportunities ☺