Integrating FPGAs in a heterogeneous and portable parallel programming model

Gabriel Rodriguez-Canal, Yuri Torres and Arturo Gonzalez-Escribano
Universidad de Valladolid, Spain
grodriguez-canal@acm.org
{yuri.torres|arturo}@infor.uva.es

SuperComputing 2020, November 2020

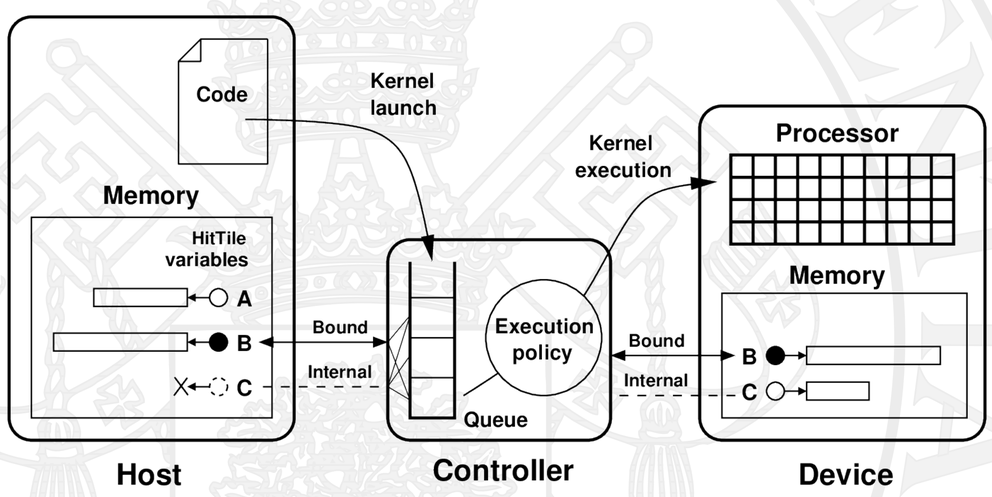
Universidad de Valladolid

Grupo Trasgo
Universidad de Valladolid

Introduction

- **FPGAs** have attracted a wider scope of HPC researchers thanks to HLS (High Level Synthesis) languages and tools, such as those for **OpenCL**.
  - √ Higher abstraction, reduced programming effort.
  - × Cumbersome details, like manual command-queue and data-transfer management and synchronization.
- **More abstract models:** SYCL/DPC++. Use modern C++ concepts and abstractions, advocate single source code for both host and device.
- **Different approach: Controller** [1], a parallel heterogeneous programming model with current support for CPUs, GPUs and Xeon Phi co-processors.
  - ⇒ Compiler agnostic: Implemented as a C99 functions library, interoperable with other programming models such as OpenMP, etc.
  - ⇒ Portable host code + library of optimized versions of kernels for device types or families (using different algorithms or lower-level languages if needed).
  - ⇒ Runtime: Selects the best suited version of a kernel for the chosen device.
- **Other common features:** Automatic and transparent management of data-transfers, overlapping with concurrent host and device computations.
- **Goal**: **Integrate Intel FPGAs in Controller**, offload workloads to FPGAs with no change to host codes.

Controller model



FPGA integration

- **Extension of the kernel library structure:** Support for FPGA OpenCL and AOC offline compiler.
- **Kernel conf. parameters**: Pipeline, SIMD, compute units.
- **Three execution models**: ordinary, emulation, profiling.
- **Threads vs. data structure sizes:** Transparent padding for extra threads in last groups.
- **Synchronicity:** Integration of Controller's synchronous and asynchronous execution policies. Events control, support devices with half-duplex or full-duplex transfers, overlapping, etc.

Controller programming example: (Code excerpts)

```
/* FILE: my_header.h */
#include "Ctrl.h"
Ctrl_NewType( float );
CTRL_KERNEL_PROTO( k_compute, 1, FPGA, 2, IN,HitTile_float,A, OUT,HitTile_float,B );

/* FILE: kernels.c */
#include "my_header.h"
CTRL_KERNEL( k_compute, FPGA, PIPELINE( NDRANGE, SIMD( 8, 64, 64, 1 ) ),
                  PARAM( IN, OUT ), KHitTile_float A, KHitTile_float B )
{
  // Kernel code written in OpenCL
}

/* FILE: main.c */
#include "my_header.h"
int main(int argc, char *argv[]) {
  ...
  Ctrl_ThreadInit(threads,rows,columns); Ctrl_ThreadInit(group,tileRows,tileColumns);
  __ctrl_block__(1)
  {
    PCtrl ctrl = Ctrl_Create( CTRL_TYPE_FPGA, CTRL_POLICY_ASYNC, device, platform );
    HitTile_float arr1 = Ctrl_Alloc( ctrl, float, hitNewShapeSize(rows, columns) );
    HitTile_float arr2 = Ctrl_Alloc( ctrl, float, hitNewShapeSize(rows, columns) );
    Ctrl_HostTask( ctrl, read_matrix, arr1 );

    for( i=0; i<num_iter; i++ ) {
      Ctrl_Launch( ctrl, k_compute, threads, group, arr1, arr2 );
      Ctrl_HostTask(ctrl, process_result, arr2);
      Ctrl_Launch( ctrl, k_prepare_next_iteration, threads, group, arr2 );
      swap( arr1, arr2 );
    }
    Ctrl_Free(ctrl, arr1, arr2);
    Ctrl_Destroy(ctrl);
  }
}
```

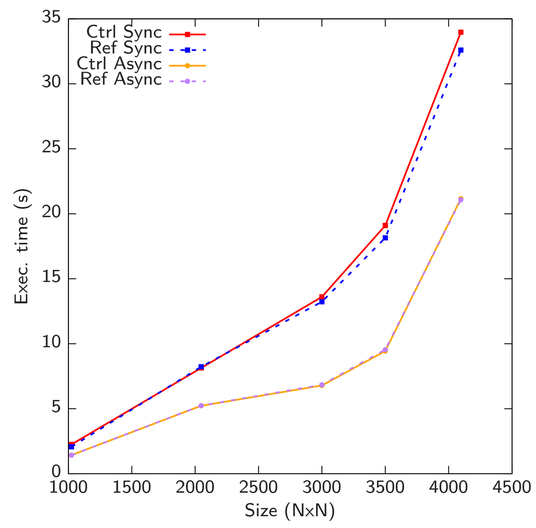
Experimental results

Comparison: Controller vs. OpenCL FPGA programs

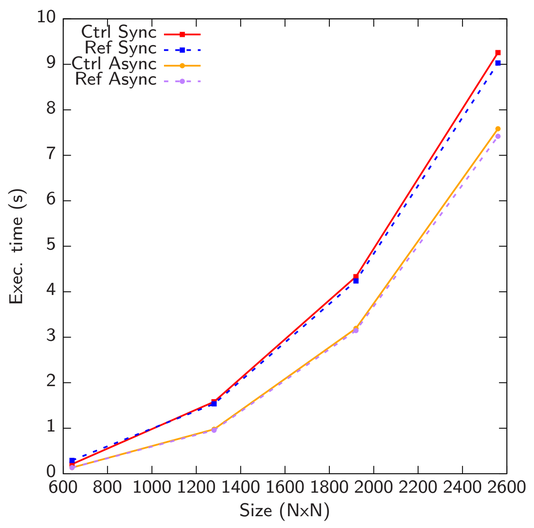
- Ref Sync: OpenCL synchronous.
- Ref Async: OpenCL asynchronous, using events.
- Ctrl Sync: Controller code, activating synchronous policy.
- Ctrl Async: Controller code, activating asynchronous policy.

Case study	Version	LOC	TOK	CCN	Halstead
Hotspot	Ctrl	230	1772	40	919321
	Ref Sync	339	2771	57	1770315
	Ref Async	401	3273	53	2332285
Matrix Pow	Ctrl	148	1509	21	525721
	Ref Sync	211	1922	30	1243644
	Ref Async	271	2348	29	1646456
Sobel filter	Ctrl	137	1231	22	907566
	Ref Sync	202	1944	28	1207349
	Ref Async	290	2561	38	1689124

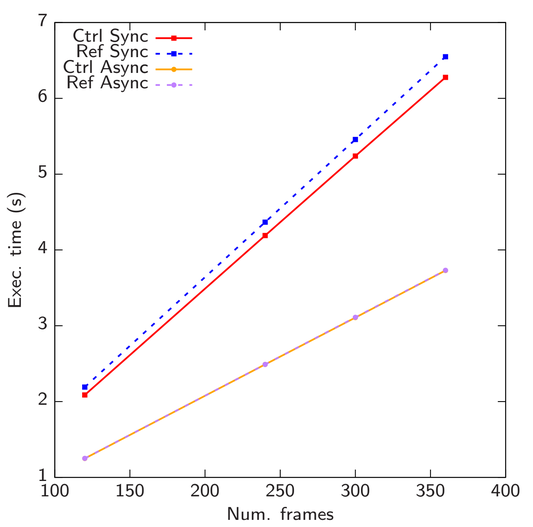
Development effort metrics: Lines of code (LOC), number of tokens (TOK), McCabe's cyclomatic complexity (CCN) and Halstead development effort.



Hotspot (400 iterations)



Matrix Pow (30 iterations)



Sobel Filter on YUV video

Conclusions and future work

- We have extended the Controller model with a **new backend with FPGA support**.
- Fully **portable host codes**, with generic or **device specialized kernels**.
- Different binaries for different devices, features, or execution modes automatically managed.
- Transparent synchronous and asynchronous execution policies.
- Performance **overheads of less than 1%** with 95% of confidence.
  - ⇒ Similar to other high-level heterogeneous programming models (SYCL [2]).
- Programming with Controller requires **much less development effort** than using OpenCL.
- **Future work:** Direct comparison with SYCL/DPC++; integrating HDL kernels for further tuning.

References

[1] Moreton-Fernandez, Ana and Ortega-Arranz, Hector and Gonzalez-Escribano, Arturo Controllers: An abstraction to ease the use of hardware accelerators. May, 2017 DOI: 10.1177/1094342017702962

[2] Deakin, Tom and McIntosh-Smith, Simon Evaluating the Performance of HPC-Style SYCL Applications. ACM. Proceedings of the International Workshop on OpenCL. April, 2020 DOI: 10.1145/3388333.3388643.

http://trasgo.infor.uva.es/controller/

Acknowledgements

This research has been partially supported by MICINN (Spain), the ERDF program of the European Union and Junta de Castilla y León: PCAS project (TIN2017-88614-R), Scottish Power Masters Scholarship, MECD (Beca de Colaboración, Spain), Salvador de Madariaga/Fulbright Scholar Grant (PRX17/00674) and Universidad de Zaragoza.