Hello/Goodbye FPGA

3rd Patatrack Hackathon 25/05/18 Final Scram

Sioni Summers, Lukas Arnold, Shahzad Muzaffar

Aims & Overview

- Newest Intel devices have the best floating point performance of any FPGAs
- OpenCL promises fast kernel design, convenient packaging of compute code with host-FPGA interfaces
- Aimed to execute OpenCL code on a Intel FPGA, called from CMSSW
- → Towards porting real computation in a heterogeneous future
- Wanted to gain experience with these devices and this way of designing

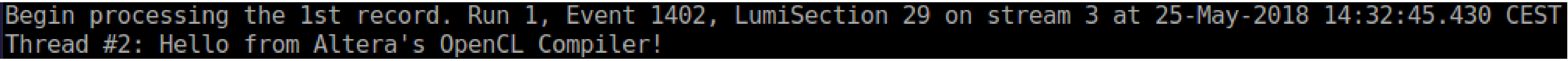
Challenges

- Sioni:
  - New to Intel FPGAs and OpenCL
- Lukas:
  - New to OpenCL and CMSSW
- Shahzad
  - Never worked with FPGAs
- Working with FPGAs is slow, what can we achieve in 4 days?
- We had never met before!

How we worked

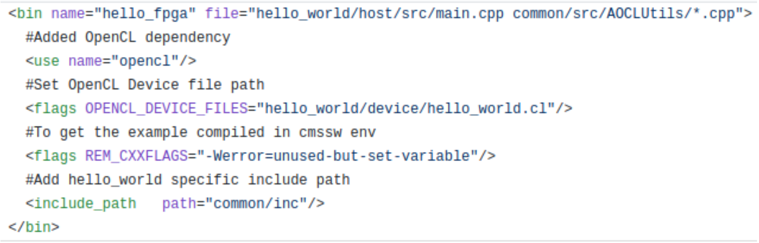
- Split work between host-side and device-side
  - Lukas on device-side
  - Sioni on host-side
- Shahzad on the compile flow
- All:
  - Worked through Intel's example codes

What we achieved



```
Begin processing the 1st record. Run 1, Event 1402, LumiSection 29 on stream 3 at 25-May-2018 14:32:45.430 CEST
Thread #2: Hello from Altera's OpenCL Compiler!
```

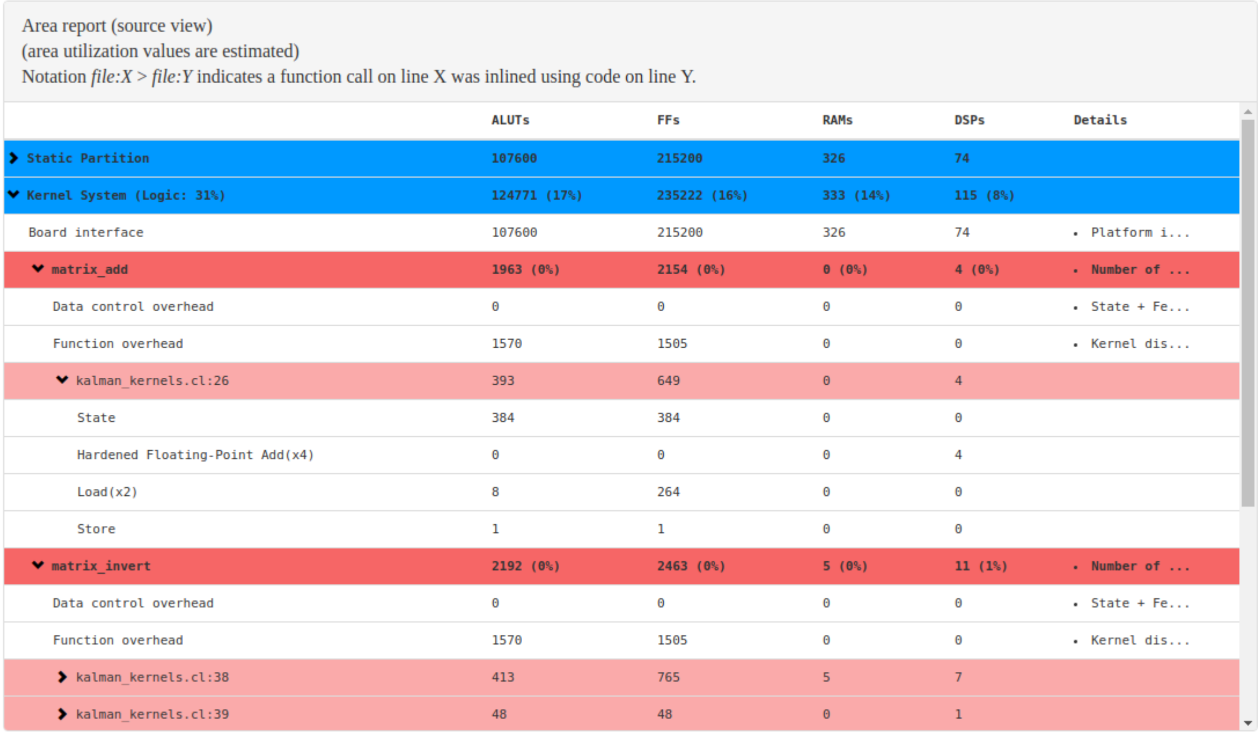
- 'Hello world' from CMSSW
- 'scram b' compiles OpenCL kernels for Intel FPGAs and C++ with OpenCL wrappers
- Designed kernels for Kalman Filter state update
  - Compiled for Hardware
  - Standalone testing



```
<bin name="hello_fpga" file="hello_world/host/src/main.cpp common/src/AOCLUtils/*.cpp">
  #Added OpenCL dependency
  <use name="opencl"/>
  #Set OpenCL Device file path
  <flags OPENCL_DEVICE_FILES="hello_world/device/hello_world.cl"/>
  #To get the example compiled in cmssw env
  <flags REM_CXXFLAGS="-Werror=unused-but-set-variable"/>
  #Add hello_world specific include path
  <include_path   path="common/inc"/>
</bin>
```

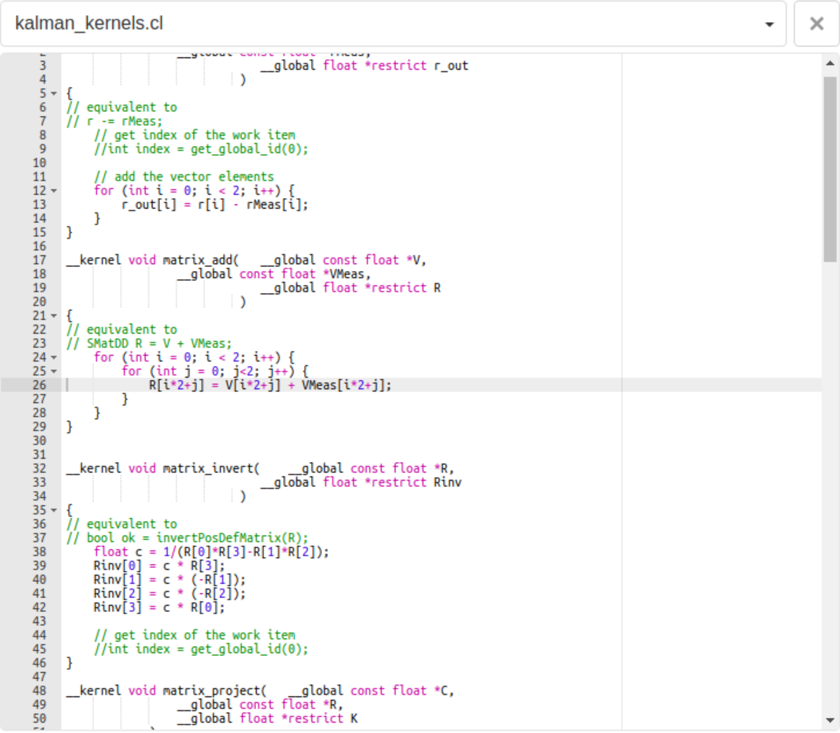
What we achieved

- Started evaluating compiled kernels
  - Thinking about resource usage, data flow

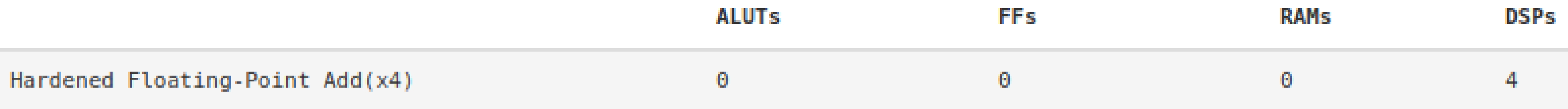


Area report (source view)
(area utilization values are estimated)
Notation *file:X* > *file:Y* indicates a function call on line X was inlined using code on line Y.

	ALUTs	FFs	RAMs	DSPs	Details
› Static Partition	107600	215200	326	74	
⌄ Kernel System (Logic: 31%)	124771 (17%)	235222 (16%)	333 (14%)	115 (8%)	
Board interface	107600	215200	326	74	• Platform i...
⌄ matrix_add	1963 (0%)	2154 (0%)	0 (0%)	4 (0%)	• Number of ...
Data control overhead	0	0	0	0	• State + Fe...
Function overhead	1570	1505	0	0	• Kernel dis...
⌄ kalman_kernels.cl:26	393	649	0	4	
State	384	384	0	0	
Hardened Floating-Point Add(x4)	0	0	0	4	
Load(x2)	8	264	0	0	
Store	1	1	0	0	
⌄ matrix_invert	2192 (0%)	2463 (0%)	5 (0%)	11 (1%)	• Number of ...
Data control overhead	0	0	0	0	• State + Fe...
Function overhead	1570	1505	0	0	• Kernel dis...
› kalman_kernels.cl:38	413	765	5	7	
› kalman_kernels.cl:39	48	48	0	1	



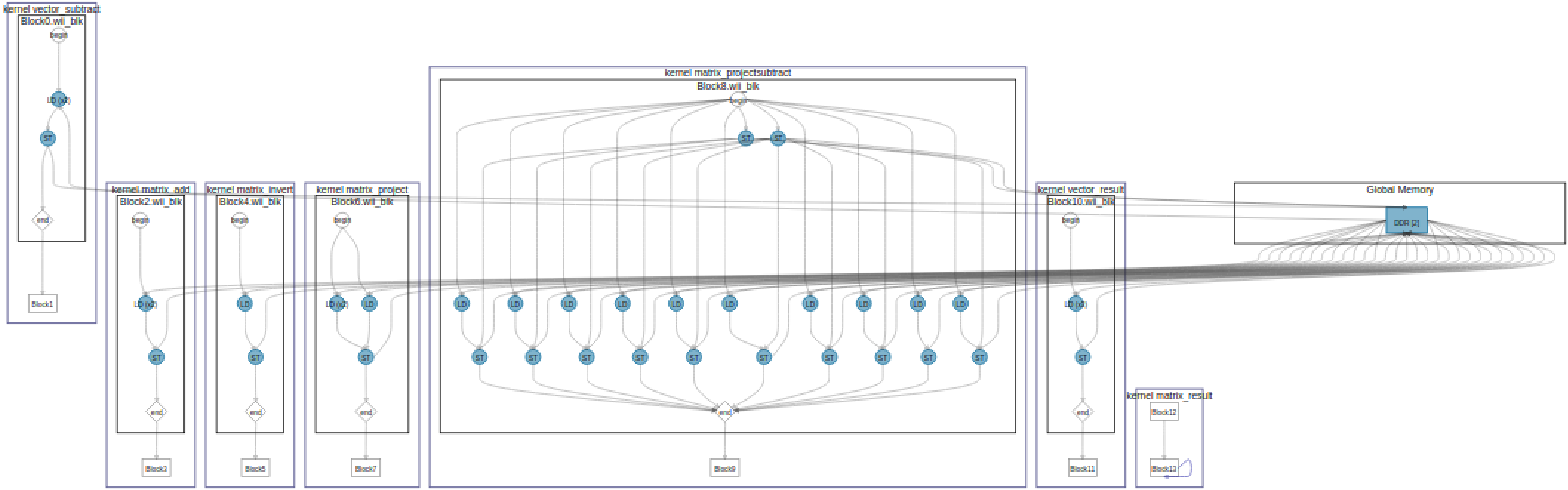
```
                                        __global float *restrict r_out
                                    )
{
// equivalent to
// r -= rMeas;
    // get index of the work item
    //int index = get_global_id(0);

    // add the vector elements
    for (int i = 0; i < 2; i++) {
        r_out[i] = r[i] - rMeas[i];
    }
}

__kernel void matrix_add(   __global const float *V,
                    __global const float *VMeas,
                                __global float *restrict R
                                    )
{
// equivalent to
// SMatDD R = V + VMeas;
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j<2; j++) {
            R[i*2+j] = V[i*2+j] + VMeas[i*2+j];
        }
    }
}


__kernel void matrix_invert(    __global const float *R,
                                __global float *restrict Rinv
                                    )
{
// equivalent to
// bool ok = invertPosDefMatrix(R);
    float c = 1/(R[0]*R[3]-R[1]*R[2]);
    Rinv[0] = c * R[3];
    Rinv[1] = c * (-R[1]);
    Rinv[2] = c * (-R[2]);
    Rinv[3] = c * R[0];

    // get index of the work item
    //int index = get_global_id(0);
}

__kernel void matrix_project(   __global const float *C,
                    __global const float *R,
                    __global float *restrict K
```



	ALUTs	FFs	RAMs	DSPs
Hardened Floating-Point Add(x4)	0	0	0	4

What we achieved

- Started evaluating compiled kernels
  - Thinking about resource usage, data flow



What we learned

- Lukas
  - Some OpenCL (by designing some kernels), CMSSW, tracking
- Sioni
  - Some OpenCL and the C++ Wrapper (that it's a bit ugly)
- Shahzad
  - About FPGAs, Intel's FPGA tools, and running them!
- All
  - Intel OpenCL FPGA compiler is pretty slow... (~3 hrs for a design that adds 2 numbers)

The future...

- We want to continue working together with these devices
- Towards a real algorithm (Kalman Filter)
- Tighten the integration with CMSSW

- Thanks to all the organisers!