Webinar II

HPP-Heterogeneous Processing Platform
Programming Models, Performance and Power Profiling for the HPP

Yassine Hariri
Senior Engineer, Platform Design
Hariri@cmc.ca

Hugh W. Pollitt-Smith
Senior System Design Engineer
Pollitt-smith@cmc.ca
Agenda

• Overview
• emSYSCAN Development Systems Update
• HPP: Heterogeneous processing platform
• HPP GPU
  – GPU Programming
    • Libraries, OpenACC and Programming langages
  – CUDA Development Using NVIDIA Nsight, Eclipse Edition
    • Project Management, Edit, Build, Debug and Profile
  – Power profiling using nvidia-smi
  – Live demo
• HPP FPGA
  – OpenCL for FPGA
  – The AOCL FPGA Programming Flow
  – Power and performance profiling
  – Live demo
• HPP schedule
• Heterogeneous processing workshop
Microsystems Rapid-Prototyping, Characterization and Integration Labs

4 Universities:
- UBC
- U Manitoba
- Queen’s
- École Polytechnique

Multi-Technology Design Environment
- System architecture exploration
- Multi-technology simulation
- Design of custom devices for manufacturing

Development Systems
- System validation and proof-of-concept demonstration

License Management Appliance
Installed Design Environment
Development System Hardware

Real-Time Embedded Software Lab
University of Waterloo

- Design, analysis, debug of real-time software on next-generation processor systems

Common, Shared Platforms
Interconnected Community of Users
Knowledge Repository
Centralized Management & Operations

Installed Design Environment
License Management Appliance
Development System Hardware

Embedded Systems Canada (emSYSCAN)
$54M investment in Canada’s National Design Network
37+ universities, 250+ faculty, 5 years

© 2013, CMC Microsystems

12/16/2015
Development Systems for Proof of Concept

Images courtesy of National Instruments, Xilinx, BEECube, NVIDIA

12/16/2015
National Research Platform: Enriched Projects; Results Sooner

- Common set of programmable research platforms with proof of concept features
- Pooled equipment timeshared among users
- Sharing of knowledge on equipment usage
- Adaptive over time in terms of equipment quantities and equipment features
- Large community of users, institutions
- Leveraged industrial partners (e.g., STMicro.)

National project scope and sizeable outcomes enabled by centralized project implementation and management by CMC Microsystems
Installation and usage

• Shared access systems can be accessed at no charge but require Designer level subscription
  – Subscription provides access to support, tools, reference designs, forums, workshops, travel, select/swap, training, additional discounts
• Systems delivered on site, remote access
• Designated Development System coordinator(s) at each site
  – Communicate institutional needs for purchase specifications
  – Local advocate, information source
  – Encourage participation in National Project
emSYSCAN Development Systems delivered (Gen1)

- Embedded Systems Platform:
  - Xilinx ML605, Altera DE4-530
- Advanced Processing Platform
  - BEEcube BEE3, BEE4, miniBEE
- Software-Defined Radio Platform
  - BEEcube miniBEE, RF daughtercard
- Simulation Acceleration Platform
  - Nallatech P385-D5 (Altera Stratix V, OpenCL)
- Multiprocessor Array Platform
  - NVIDIA Tesla K20 GPU
  - Intel Xeon Phi
- Microsystems Integration Platform
  - National Instruments PXI-based, FPGA, MEMS, microfluidics, RF, photonics features
The following Development Systems have been delivered to the National Design Network (NDN) for shared access. The designated Coordinator/Contact can provide additional details on availability and how to access:

<table>
<thead>
<tr>
<th>System Product</th>
<th>Location/University</th>
<th>Quantity</th>
<th>Coordinator/Contact</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEEcube BEE3 (Advanced Processing Platform)</td>
<td>CMC Microsystems (online access)</td>
<td>2</td>
<td>Hugh Politt-Smith</td>
</tr>
<tr>
<td></td>
<td>McMaster University</td>
<td>1</td>
<td>Dr. Nicola Nicolici</td>
</tr>
<tr>
<td></td>
<td>University of Guelph</td>
<td>1</td>
<td>Dr. Stefano Gregori</td>
</tr>
<tr>
<td></td>
<td>McGill University</td>
<td>1</td>
<td>Dr. Zeljko Zilic</td>
</tr>
<tr>
<td></td>
<td>University of New Brunswick</td>
<td>1</td>
<td>Dr. Kenneth Kent</td>
</tr>
<tr>
<td></td>
<td>University of Saskatchewan</td>
<td>1</td>
<td>Dr. Seok-Bum Ko</td>
</tr>
<tr>
<td></td>
<td>Université du Québec à Chicoutimi</td>
<td>1</td>
<td>Dr. Hung-Tien Bui</td>
</tr>
<tr>
<td></td>
<td>Université du Québec à Trois-Rivières</td>
<td>1</td>
<td>Dr. Adel Omar Dahmane</td>
</tr>
<tr>
<td></td>
<td>Université du Québec à Outaouais</td>
<td>1</td>
<td>Dr. Ahmed Lakhissassi</td>
</tr>
<tr>
<td></td>
<td>University of Windsor</td>
<td>1</td>
<td>Dr. Rashid Rashidzadeh</td>
</tr>
</tbody>
</table>
Upcoming emSYSCAN Development Systems deployments

• Embedded Systems Platform
  – Xilinx Virtex-7, Ultrascale, Zynq options
  – Altera Arria 10 and Arria 10 SoC options
  – Shipping Q1 2016

• Advanced Processing Platform
  – RFP currently in evaluation
  – Shipping Q1/Q2 2016

• Software-defined Radio
  – BEEcube nano/megaBEE (2x2 up to 16x16 MIMO options)
  – Shipping Q1 2016
Canada’s National Design Network – ADEPT Management & Operations

32 Institutions
Design, compute, store on local resources; secure download
Secure, remote access to platforms, compute infrastructure (thin client)

Benefiting Canadians
- Information and Communications Technologies
- Healthcare
- Transportation
- Energy
- Manufacturing
- Security

ADEPT: Advanced Design Platform Technology
- Design Platforms
  - Multi-technology Interposer
  - Sensor/Actuator
  - Heterogeneous Embedded
  - Silicon Photonics
  - User Platform 1
- Intellectual Property Blocks & Physical Design Kits
- Advanced Design Methods
- Fabrication Process Repository
  - Equipment database
  - Recipes
  - TCAD
- Computer-Aided Design Tools
- License Management System
- Access Infrastructure
  - User accounts, storage
  - Accelerators
- Compute Servers (Compute Canada)
- Cybersecurity Testbed
  - Observation, analysis, delivery
- Fabrication Laboratories
  - Lab Capabilities
  - Recipe Development
  - Prototyping
- Vendors & Partners
  - CAD tools
  - Intellectual Property Blocks & Physical Design Kits
  - Design Methods
  - Multi-project wafer services & scale-up manufacturing

12/16/2015
HPP Distribution

• Based on Development Systems Coordinator consultations in April 2014:
    • USask, UQTR, Outaouais, McGill, York, Windsor, Waterloo, Western, Ottawa, Ryerson, RMC, Victoria
    • Memorial, Guelph, McMaster, Toronto, Polytechnique, UQTR, Outaouais
HPP main components

• The HPP workstation integrates the following main components:
  – Dual core Intel Xeon E5-2620 V3
  – NVidia GPU (Tesla K20)
  – FPGA board (Nallatech P385-A72).
  – Xeon Phi 7120A
• Key Platform Benefits
  – Customizability: Select the right mix of accelerators for your application
  – Greater flexibility for HW/SW exploration
  – Scalability: Create one node and scale up by adding more nodes
  – Fast automated setup and configuration
  – Faster path to commercialization
  – Technical support and training from CMC Microsystems
HPP fully installed system
HPP Pre-Installed Software Components

These software components required by the HPP are pre-installed on the workstation:

- RedHat Enterprise Linux 6.6 (Kernel version: 2.6.32-504.el6.x86_64)
- Java Runtime Environment (JRE)
- gcc compiler and toolchain

<table>
<thead>
<tr>
<th>Accelerator</th>
<th>Software and tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU Tesla K20</td>
<td>• NVIDIA CUDA 7 Toolkit</td>
</tr>
<tr>
<td>Nallatech FPGA P385</td>
<td>• Altera Quartus 15.0, Altera SDK for OpenCL 1.0</td>
</tr>
<tr>
<td></td>
<td>• Nallatech FPGA P385 Board Support Package</td>
</tr>
<tr>
<td>Xeon Phi 7120a</td>
<td>• Intel Manycore Platform Software Stack (MPSS) 3.5.1 for Linux</td>
</tr>
<tr>
<td></td>
<td>• Intel Parallel Studio XE 2015, Professional Studio for C++, Linux version</td>
</tr>
</tbody>
</table>
HPP: GPU
Applications Acceleration

Applications
- Libraries
- OpenACC Directives
- Programming Languages
Libraries

- **Ease of use:** Deep knowledge of GPU programming is not required
- **“Drop-in”:** Standard APIs, minimal code changes
- **Quality:** High-quality implementations
- **Performance:** NVIDIA libraries are highly optimized
CUDA Libraries Ecosystem

- CUDA Tools and Ecosystem described in detail on NVIDIA Developer Zone:
  
  developer.nvidia.com/cuda-tools-ecosystem
Applications Acceleration

Applications

Libraries

OpenACC Directives

Programming Languages
OpenACC Directives

Program myscience
... serial code ...
!$acc kernels
  do k = 1,n1
  do i = 1,n2
    ... parallel code ...
    enddo
  enddo
!$acc end kernels
... 
End Program myscience

12/16/2015
Applications Acceleration

Libraries

OpenACC Directives

Programming Languages
GPU Programming Languages

- Numerical analytics: MATLAB, Mathematica, LabVIEW
- Fortran: OpenACC, CUDA Fortran
- C: OpenACC, CUDA C
- C++: Thrust, CUDA C++
- Python: PyCUDA, Copperhead
HPP GPU:
CUDA DEVELOPMENT USING NVIDIA NSIGHT, ECLIPSE EDITION
NVIDIA® Nsight™ Eclipse Edition

- CUDA Integrated Development Environment
- Project Management
- Edit
- Build
- Debug
- Profile
Powered By Eclipse

- Extensible via robust selection of open-source and commercial plugins
- Revision control: CVS, SVN, Git, Perforce, …
- Issue tracking
- ...
- Strong cross-platform support
- Nsight Eclipse Edition available for Linux and Mac OS X
Included In CUDA Toolkit

```
[root@HPPPrototype ~]# export PATH=$PATH:/usr/local/cuda-7.0/bin
[root@HPPPrototype ~]# export LD_LIBRARY_PATH=/usr/local/cuda-7.0/lib64:$LD_LIBRARY_PATH
[root@HPPPrototype ~]# export CUDA_PATH=/usr/local/cuda-7.0
[root@HPPPrototype ~]# nsight&
```

12/16/2015
NVIDIA® Nsight™ Eclipse Edition

- CUDA Integrated Development Environment
  - Project Management
  - Edit
  - Build
  - Debug
  - Profile
Nsight Project Support

• CUDA C / C++
• Project Types
  – Executable
  – Shared Library
  – Static Library

• New vs. Existing
  – New project, build managed by Nsight
  – Existing project, Nsight can use your Makefile
Creating A New CUDA Project
Nsight main window after creating a new project
• CUDA Integrated Development Environment
  – Project Management
  – Edit
  – Build
  – Debug
  – Profile
CUDA Editor

- CUDA-aware syntax highlighting
- Host / Device code highlighting
- Smart code assist
- As-you-type error detection
- CUDA API documentation pop-ups
- Automatic code refactoring

```c
// Launch the Vector Add CUDA Kernel
int threadsPerBlock = 256;
int blocksPerGrid = (numElements + threadsPerBlock - 1) / threadsPerBlock;
printf("CUDA kernel launch with %d blocks of %d threads\n", blocksPerGrid, threadsPerBlock);

vectorAdd<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, numElements);
```

```c
/**
 * CUDA Kernel Device code
 * Computes the vector addition of A and B into C. The 3 vectors have the same
 * number of elements numElements.
 */

__global__ void
vectorAdd(const float *A, const float *B, float *C, int numElements)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if (i < numElements)
```
• CUDA Integrated Development Environment
  – Project Management
  – Edit
  – Build
  – Debug
  – Profile
CUDA Builder

- Full CUDA toolchain support
  - All nvcc features
  - Debug, release, and custom build configurations
- Dependent project support
  - Static libraries
  - Shared libraries
  - Manages all build dependencies
- Source-correlated error reporting
float *d_A = NULL;
err = cudaMalloc((void**)&d_A, size, 3);

if (err)
{
    fprintf(stderr, "no instance of overloaded function "cudaMalloc" matches the argument list", cudaGetErrorString(err));
}

// Allocate the device input vector B
float *d_B = NULL;
err = cudaMalloc((void**) &d_B, size);
Run Application

12/16/2015

[Code snippet]

```c
int main(int argc, char** argv) {
    float *d_A = NULL;
    err = cudaMalloc((void**)&d_A, size);
    if (err != cudaSuccess)
        fprintf(stderr, "Failed to allocate device vector A (error code %s)\n", cudaGetErrorString(err));
    exit(EXIT_FAILURE);
}
```

[Report summary]

- Vector addition of 56000 elements
- Copy input data from the host memory to the CUDA device
- CUDA kernel launch with 196 blocks of 256 threads
- Copy output data from the CUDA device to the host memory
- Test PASSED
- Done
• CUDA Integrated Development Environment
  – Project Management
  – Edit
  – Build
  – Debug
  – Profile
CUDA Debugger

- Unified CPU / GPU Debugging
  - Simultaneous visibility into both CPU and GPU state
  - Multi-GPU support

- Full GPU debugging
  - Set kernel breakpoints
  - Single-step, run until, etc.
  - View variables, registers, and expression values across multiple GPU threads at the same time
  - Examine thread, warp, block state
  - Source and assembly level debugging
Add a break point in the code

```c
__global__ void
vectorAdd(const float *A, const float *B, float *C, int numElements)
{
    int i = blockDim.x * blockIdx.x + threadIdx.x;
    if (i < numElements)
    {
        C[i] = A[i] + B[i];
    }
    float);
GPU / CPU Threads, Call Stacks
Stepping
GPU / CPU Threads, Call Stacks

12/16/2015
Edit

- CUDA Integrated Development Environment
  - Project Management
  - Edit
  - Build
  - Debug
  - Profile
CUDA Profiler

- Unified GPU / CPU profiler
- Visualize GPU / CPU interactions Identify GPU utilization and efficiency bottlenecks
- View low-level counters and metrics
- Multi-GPU support
- Automated application analysis identifies optimization opportunities
- Online documentation gives direction on how to exploit opportunities to get performance improvement
Unified GPU / CPU Timeline
9.1. Data Transfer Between Host and Device

The peak theoretical bandwidth between the device memory and the GPU is much higher (177.6 GB/s on the NVIDIA Tesla M2090, for example) than the peak theoretical bandwidth between host memory and device memory (8 GB/s on the PCIe x16 Gen2). Hence, for best overall application performance, it is important to minimize data transfer between the host and the device, even if that means running kernels on the GPU that do not demonstrate any speedup compared with running them on the host CPU.

Note: High Priority: Minimize data transfer between the host and the device, even if it means running some kernels on the device that do not show performance gains when compared with running them on the host CPU.

Intermediate data structures should be created in device memory, operated on by the device, and destroyed without ever being mapped by the host or copied to host memory.

Also, because of the overhead associated with each transfer, batching many small transfers into one larger transfer performs significantly better than making each transfer separately, even if doing so requires packing non-contiguous regions of memory into a contiguous buffer and then unpacking after the transfer.

Finally, higher bandwidth between the host and the device is achieved when using page-locked (or pinned) memory, as discussed in the CUDA C Programming Guide and the Pinned Memory section of this document.

9.1.1. Pinned Memory

Page-locked or pinned memory transfers attain the highest bandwidth between the host and the device. On PCIe x16 Gen2 cards, for example, pinned memory can attain roughly 6GB/s transfer rates.

Pinned memory is allocated using the `cudaHostAlloc()` functions in the Runtime API. The `bandwidthTest` CUDA Sample shows how to use these functions as well as how to measure memory transfer performance.

For regions of system memory that have already been pre-allocated, `cudaHostRegister()` can be used to pin the memory on-the-fly without the need to allocate a separate buffer and copy the data into it.

Pinned memory should not be overused. Excessive use can reduce overall system performance because pinned memory is a scarce resource, but how much is too much is difficult to know in advance. Furthermore, the pinning of system memory is a heavyweight operation compared to most normal system memory allocations, so as with all optimizations, test the application and the systems it runs on for optimal performance parameters.
Power profiling using nvidia-smi

- `nvidia-smi -q -d POWER -i 0 -l 1 -f out.log`
  - `-q, --query` Display GPU or Unit info
  - `-d TYPE, --display=TYPE` Display only selected information: MEMORY, UTILIZATION, ECC, TEMPERATURE, POWER, CLOCK, COMPUTE...
  - `-i, --id=ID` Display data for a single specified GPU or Unit.
  - `-f FILE, --filename=FILE` Redirect query output to the specified file in place of the default stdout. The specified file will be overwritten.

Power Consumption GPU (Watts)
Live Demo
HPP: FPGA
FPGA pains

• Programmability is an issue
  – Hardware Description Languages (HDLs) are complexe
  – At Register transfer level (RTL) abstraction
  – Designer needs to:
    • Create Custom Memory Hierarchies
    • Manage Communication with PC
    • Create PC side SW that plays nice with all this
OpenCL for FPGA

- Unified programming model
  - More accessible to the software developers
  - Host CPU-FPGA communication
  - C based programming language
  - Memory Hierarchy auto generated
  - Potential to integrate existing VHDL/Verilog IP
OpenCL for FPGA

Kernel Program

```c
__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];
}
```

Figure 4: White Paper: Implementing FPGA Design with the OpenCL Standard
The AOCL FPGA Programming Flow

1. Kernel Source Code #1 (.cl) → Altera Offline Compiler for OpenCL Kernels → Consolidated Kernel Binary A (.aocx)
2. Kernel Source Code #2 (.cl) → Altera Offline Compiler for OpenCL Kernels → Consolidated Kernel Binary A (.aocx)
3. Kernel Source Code #3 (.cl) → Altera Offline Compiler for OpenCL Kernels → Consolidated Kernel Binary B (.aocx)
4. Kernel Source Code #4 (.cl) → Altera Offline Compiler for OpenCL Kernels → Consolidated Kernel Binary B (.aocx)
5. Kernel Source Code #5 (.cl) → Altera Offline Compiler for OpenCL Kernels → Consolidated Kernel Binary B (.aocx)
7. Host Code → Standard C Compiler → Host Binary

Load .aocx into memory

Altera SDK for OpenCL Programming Guide
AOCL design flow steps

1. **Intermediate compilation (aoc -c [-g] <your_kernel_filename>.cl)**
   - Checks for syntatic errors
   - Generates a .aoco file without building the hardware configuration file
   - Generate estimated resource usage summary <your_kernel_filename>.log

2. **Emulation (aoc -g <your_kernel_filename>.cl)**
   - The AOCL Emulator generates a .aocx file that executes on x86-64 Windows or Linux host
   - Assess the functionality of your OpenCL kernel

3. **Profiling (aoc --profile <your_kernel_filename>.cl)**
   - aocl report <your_kernel_filename>.aocx profile.mon
   - Instruct the Altera Offline Compiler to instrument performance counters in the Verilog code in the .aocx file
   - During execution, the performance counters collect performance information which you can then review in the Profiler GUI.

4. **Full deployment**
   - Execute the .aocx file on the FPGA
OpenCL Overview

White Paper: Implementing FPGA Design with the OpenCL Standard (Page: 7)
The key features of the 385 include:
- PCI Express form factor
  - 8-lane PCI Express up to 3.0 host interface
- FPGA options
  - Altera Stratix-V 5SGXMA7H2F35C2N
  - Altera Stratix-V 5SGSMD5H2F35C2N
- Two 10G LAN/WAN/FC Ethernet channels accessed via two SFP+ ports
- Two banks of SDRAM memory, each bank with 4GByte, x72, DDR3 SDRAM running at 1600 MT/s
BSP and Altera SDK for OpenCL

- Four layers of the Altera Software Development Kit (SDK) for OpenCL:

  - The Nallatech OpenCL BSPs provide a memory-mapped device (MMD) layer necessary for communication with the accelerator board and the lower level kernel mode driver.
  - All other upper software layers are provided by the Altera SDK for OpenCL installation.
Linux Vector Addition Walkthrough

• Step 1 Compile
  – `aoc -v --board p395_hpc_ab vectorAdd.cl`
• Step 2 Build the Host code
  – `make`
• Step 3 Execute
  – `./vector_add`

• Generated files:
  – `vectorAdd.log` – kernel compilation with estimated resource usage and Qsys generation.
  – `quartus_sh_compile.log` – Quartus tools build log for generating the actual FPGA hardware `aocx` file.
  – `acl_quartus_report.txt` – Final results for the generated design including final resource usage figures.

12/16/2015
Power consumption reading

- Note that this requires that you have already opened the card using the normal opencl SDK routines (i.e. in init_opencl in this case).
- `aocl_mmd_card_info("acl0", AOCL_MMD_POWER, sizeof(float), (void*) &power, &returnedSize);
- `printf("before run Power = %f W\n", power);
- `aocl_mmd_card_info("acl0", AOCL_MMD_BOARD_UNIQUE_ID, sizeof(int), (void*) &uniqueId, &returnedSize);
- `printf("Board Unique ID = %d\n", uniqueId);

```
[root@HPPPPrototype bin]# ./matrix_mult
Matrix sizes:
  A: 2048 x 1024
  B: 1024 x 1024
  C: 2048 x 1024

MMD ERROR: MMD must be opened before calling aocl_mmd_card_info
Initializing OpenCL Platform: Altera SDK for OpenCL
Using 1 device(s)
  p385_hpc_a7 : PCIe385
Using AOCX: matrixMult.aocx
Reprogramming device with handle 1
Generating input matrices
before run Power = 17.894531 W
Board Unique ID = 7802811
About to call run
Launching for device 0 (global size: 1024, 2048)
During run Power = 18.947571 W

Time: 38.713 ms
Kernel time (device 0): 38.594 ms

Throughput: 110.94 GFLOPS
```

Computing reference output
Verifying
Verification: PASS
after run Power = 18.063171 W
Live Demo
Project status

**Status** (12 universities selected 18 systems G1, 7 universities selected 8 G2)
- Assembled, cloned, tested and shipped 18/18 units
- (1) Quick start guide: Heterogeneous Parallel Platform (HPP)
- Introduction to the HPP-Heterogeneous Parallel Platform: A combination of Multicores, GPUs, FPGAs and Many-cores accelerators (**August 26**th)

**In progress**
- (2) User Guide: Performance and Power profiling for the HPP
- Programming models, performance and power profiling for the HPP-Heterogeneous Parallel Platform (**December 2**nd)

**Next** (Webinars series for the HPP)
- Computer vision using OpenCV/OpenCL targeting the HPP- Heterogeneous Processing Platform (**January 13**th)
Workshop on Heterogeneous Computing Platforms (Scope)

Location: Toronto
Date: TBD (February 15 - February 19, 2016)

Objective
• Bring researchers from academia and expert from industry to discuss about heterogeneous computing and explore collaboration opportunities

Technical session: presentations
• Applications and Algorithms
• Software stack and tools
• Heterogeneous systems Architecture

Breakout session: Discuss and report
- What are the next generation platforms specifications?
  • What is the ranked list of platform features that would enable this research?
- What are the key research problems that needs acceleration?
- What are the top three barriers preventing access and effective use of these platforms?
- What are the challenges and Opportunities
- What barriers could CMC help lower?
- What would enable and encourage a vibrant community of researchers sharing platform configuration files and other knowledge?
- Any other feedback and guidance
Q&A

Yassine Hariri
Hariri@cmc.ca