Architecting and Programming of Future Extremely Heterogeneous Systems

Jeffrey S. Vetter

With many contributions from FTG Group and Colleagues

MODSIM 2019
Seattle
14 Aug 2019

http://ft.ornl.gov vetter@computer.org
DARPA Domain-Specific System on a Chip (DSSoC) Program

DARPA ERI DSSoC Program: Dr. Tom Rondeau
The history of GNU Radio inspiring the key technical areas of DSSoC

**2001**  
Single core, single thread (Pentium 4)

**2004**  
Growing with technology (Pentium D)

**2008**  
Multi-threaded scheduling (Cell Broadband Engine)

**2008**  
Moving to multi-core processors (i7)

**2013**  
Embedded computing (Arm and Xilinx)  
Introduction of VOLK

**2014**  
Porting to other SoCs (TI Keystone II)

**2015**  
Porting to the smart phone (Android devices)

**Today**

**Processors**
- Intel
- Arm
- PowerPC

**Computer**
- Server farms
- Workstations
- Laptops
- Smart Phones
- Embedded Systems

**I/O**
- USB 2.0
- USB 3.0
- Thunderbolt 2
- 1 GigE
- 10 GigE
- PCIe x4

Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)  
Slide courtesy of Dr. Tom Rondeau, DARPA MTO
Getting the best out of specialization when we need programmability

**Three Optimization Areas**
1. Design time
2. Run time
3. Compile time

**Addressed via five program areas**
1. Intelligent scheduling
2. Domain representations
3. Software
4. Medium access control (MAC)
5. Hardware integration

---

Slide courtesy of Dr. Tom Rondeau, DARPA MTO
DSSoC performer domains and applications

CV+SDR
- Multi-domain application
- Multi-spectral processing
- Communications

Arizona State University
Daniel W. Bliss
Univ. of Michigan, Carnegie Mellon University, General Dynamic Mission Systems, Arm Ltd., EpiSys Science

SDR
- Unmanned aerial
- Small robotic & leave-behind
- Universal soldier systems
- Multifunction systems

Raytheon/Xilinx
Tom Kazior

• Xilinx ACAP
• Visual system integrator
• Improved reconfigurability of processing

Computer Vision
- Still image and video processing
- Autonomous navigation
- Continuous surveillance
- Augmented reality

Stanford University
Mark Horowitz
Clark Barrett, Kayvon Fatahalian, Pat Hanrahan, Priyanka Raina

Oak Ridge National Laboratory
Jeffrey Vetter

SDR
- Communications and signal processing focused
- Up-front processing / data cutdown
- Improving understanding of processing systems

Google/YouTube

IBM

PlastyForma

Raytheon

Stanford

Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)

Slide courtesy of Dr. Tom Rondeau, DARPA MTO
ORNL Cosmic Project

Jeffrey S. Vetter
Seyong Lee
Mehmet Belviranli
Roberto Gioiosa
Richard Glassbrook
Abdel-Kareem Moadi
Development Lifecycle

Applications
- Create scalable application Aspen models manually, with static or dynamic analysis, or using historical information.

Ontologies
- Ontologies based on Aspen models using statistical and machine learning techniques.

Programming systems
- Programming systems built to support ontologies.
- Query Aspen models and PFU for automatic code generation, optimization, etc.

Runtime and Scheduling
- Intelligent runtime scheduling uses models and PFU to inform dynamic decisions.
- Dynamic resource discovery and monitoring.

DSSoC Chip
- DSSoC design quantitatively derived from application Aspen models.
- Early design space exploration with Aspen.

Performance Functional API
- As feature of DSSoC, PFU API provides dynamic performance response of deployed DSSoC to intelligent runtime and programming system.

Precise configuration and benchmark data for static analysis, mapping, partitioning, code generation, etc.

Dynamic Performance Feedback including profiling and configuration response.

DSSoC Design
- DSSoC design quantitatively derived from application Aspen models.
- Early design space exploration with Aspen.
Cosmic Castle | Project Overview

**Applications**
- Streaming (e.g., SW Radio)
- Sensing (e.g., SAR, vision)
- Deep learning (e.g., CNN)
- Analytics (e.g., graphs)
- Robotics (e.g., sense and react)
- Science and Engineering (e.g., CFD)

Ontologies include Motifs, Parallel Execution Patterns, Locality, Affinity, Sync, etc.

**Programming Systems**
- Compiler
- DSL
- JIT
- Libraries
- etc

Performance Models combine Ontologies and Machine Models to produce both static and dynamic cost estimates.

**Runtime and Operating Systems**
- Task scheduling
- Memory Mgt
- IO
- Synchronization

Machine Models include architecture configuration information, microbenchmarks, and API for PFU.

**DSSoC Hardware**
- CPU
- GPU
- FPGA
- Accelerator
- DSP
- Deep Memory
- Neuromorphic

Aspen
- Ontologies
  - Structured
  - Composable
  - Interoperable

Parameterizable Performance Models
- Concurrency
- Work
- Communication
- Capacities
- Synchronization

Machine Model
- Compute
- Memory
- Interconnects

PFU API
NVIDIA DGX Workstation

• 4X Tesla V100 GPUs
• TFLOPS (Mixed precision) 500
• GPU Memory 128 GB total system
• NVIDIA Tensor Cores 2,560
• NVIDIA CUDA® Cores 20,480
• CPU Intel Xeon E5-2698 v4 2.2 GHz (20-Core)
• System Memory 256 GB RDIMM DDR4
• Full NVIDIA stack
• Other compilers/tools installable on request
Intel Stratix 10 FPGA

- Intel Stratix 10 FPGA and four banks of DDR4 external memory
  - Board configuration: Nallatech 520 Network Acceleration Card
- Up to 10 TFLOPS of peak single precision performance
- 25 MBytes of L1 cache @ up to 94 TBytes/s peak bandwidth
- 2X Core performance gains over Arria® 10
- Quartus and OpenCL software (Intel SDK v18.1) for using FPGA
- Provide researcher access to advanced FPGA/SOC environment
NVIDIA Jetson AGX Xavier SoC

- NVIDIA Jetson AGX Xavier:
- High-performance system on a chip for autonomous machines
- Heterogeneous SoC contains:
  - Eight-core 64-bit ARMv8.2 CPU cluster (Carmel)
  - 1.4 CUDA TFLOPS (FP32) GPU with additional inference optimizations (Volta)
  - 11.4 DL TOPS (INT8) Deep learning accelerator (NVDLA)
  - 1.7 CV TOPS (INT8) 7-slot VLIW dual-processor Vision accelerator (PVA)
  - A set of multimedia accelerators (stereo, LDC, optical flow)
- Provides researchers access to advanced high-performance SOC environment
Qualcomm 855 SoC (SM8510P)

Adreno 640
Snapdragon X7R modem
Hexagon 690
Wi-Fi/BT/Location
Spectra 360
Secure

Kyro 485

Connectivity (5G)
- Snapdragon X24 LTE (855 built-in) modem LTE Category 20
- Snapdragon X50 5G (external) modem (for 5G devices)
- Qualcomm Wi-Fi 6-ready mobile platform: 802.11ax-ready, 802.11ac Wave 2, 802.11ay, 802.11ad
- Qualcomm 60 GHz Wi-Fi mobile platform: 802.11ay, 802.11ad
- Bluetooth Version: 5.0
- Bluetooth Speed: 2 Mbps
- High accuracy location with dual-frequency GNSS.

Spectra 360 ISP
- New dedicated Image Signal Processor (ISP)
- Dual 14-bit CV-ISPs; 48MP @ 30fps single camera
- Hardware CV for object detection, tracking, stereo depth process
- 6DoF XR Body tracking, H265, 4K60 HDR video capture, etc.

Kyro 485 (8-ARM Prime+BigLittle Cores)
- Quad thread Scalar Core
- DSP + 4 Hexagon Vector Xccelerators
- New Tensor Xccelerator for AI
- Apps: AI, Voice Assistance, AV codecs

Hexagon 690 (DSP + AI)
- Quad thread Scalar Core
- DSP + 4 Hexagon Vector Xccelerators
- New Tensor Xccelerator for AI
- Apps: AI, Voice Assistance, AV codecs

Vulkan, OpenCL, OpenGL ES 3.1
- Apps: HDR10+, HEVC, Dolby, etc
- Enables 8k-360° VR video playback
- 20% faster compared to Adreno 630

Snapdragon 855 Mobile Platform
- 7nm TSMC

Qualcomm Development Board connected to (mcumudo) HPZ820
- Connected Qualcomm board to HPZ820 through USB
- Development Environment: Android SDK/NDK
- Login to mcumudo machine
  $ ssh -Y mcumudo
- Setup Android platform tools and development environment
  $ source /home/nqx/setup_android.source
- Run Hello-world on ARM cores
  $ git clone https://code.ornl.gov/nqx/helloworld-android
  $ make compile push run
- Run OpenCL example on GPU
  $ git clone https://code.ornl.gov/nqx/opencl-img-processing
    • Run Sobel edge detection
      $ make compile push run fetch
- Login to Qualcomm development board shell
  $ adb shell
  $ cd /data/local/tmp

© Qualcomm Inc.
© Qualcomm Inc.
RISC-V

RISC-V Ecosystem

Software
- Open-source software:
  - Gcc, binutils, glibc, Linux, BSD, LLVM, QEMU, FreeRTOS, ZephyrOS, LiteOS, SylixOS, ...
- Commercial software:
  - Lauterbach, Segger, Micrium, ExpressLogic, ...

Hardware
- Open-source cores:
  - Rocket, BOOM, RISCY, Ariane, PicoRV32, Piccolo, SCR1, Hummingbird, ...
- Commercial core providers:
  - Andes, Bluespec, Cloudbear, Codasip, Cortus, C-Sky, Nuclei, SiFive, Syntacore, ...
- Inhouse cores:
  - Nvidia, +others
Applications

- Streaming (e.g., SW Radio)
- Sensing (e.g., SAR, vision)
- Deep learning (e.g., CNN)
- Analytics (e.g., graphs)
- Robotics (e.g., sense and react)
- Science and Engineering (e.g., CFD)

Ontologies include Motifs, Parallel Execution Patterns, Locality, Affinity, Sync, etc.

Programming Systems

- Compiler
- DSL
- JIT
- Libraries
- etc.

Performance Models combine Ontologies and Machine Models to produce both static and dynamic cost estimates

Runtime and Operating Systems

- Task scheduling
- Memory Mgt
- IO
- Synchronization

Machine Models include architecture configuration information, microbenchmarks, and API for PFU

DSSoC Hardware

- CPU
- GPU
- FPGA
- Accelerator
- DSP
- Deep Memory
- Neuromorphic
A radio communication system where components that have been traditionally implemented in hardware (e.g., mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a computer, phone, or embedded system (Wikipedia).

- Gnu Radio – open software for SDR
- Composable modules and workflows for signal processing
End-to-End System: Gnu Radio for Wifi on two NVIDIA Xavier SoCs

- Signal processing: An open-source implementation of IEEE-802.11 WIFI a/b/g with GNR OOT modules.
- Input / Output file support via Socket PDU (UDP server) blocks
- Image/Video transcoding with OpenCL/OpenCV
RF + SoC Hardware Testbed

- GNU Radio ➔ 2x Ettus B210: [RF-A: Loopback cable, RF-B: Antennas, Wifi Frequency: 5.89 GHz]
• Preliminary SDR Application Profiling:
  • Created fully automated GRC profiling toolkit
  • Ran each of the 89 flowgraph for 30 seconds
  • Profiled with performance counters
  • Major overheads:
    • Python glue code (libpython), O/S threading & profiling (kernel.kallsyms, libpthread), libc, ld, Qt
  • Runtime overhead:
    • Will require significant consideration when run on SoC
    • Cannot be executed in parallel
    • Hardware assisted scheduling is essential

<table>
<thead>
<tr>
<th>Library</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel.kallsyms</td>
<td>27.8547</td>
</tr>
<tr>
<td>libpython</td>
<td>18.6281</td>
</tr>
<tr>
<td>libgnuradio</td>
<td>11.7548</td>
</tr>
<tr>
<td>libc</td>
<td>7.7503</td>
</tr>
<tr>
<td>ld</td>
<td>3.8839</td>
</tr>
<tr>
<td>libvolk</td>
<td>3.7963</td>
</tr>
<tr>
<td>libperl</td>
<td>3.7837</td>
</tr>
<tr>
<td>[unknown]</td>
<td>3.6465</td>
</tr>
<tr>
<td>libQt5</td>
<td>2.9866</td>
</tr>
<tr>
<td>libpthread</td>
<td>2.1449</td>
</tr>
</tbody>
</table>
GRC statistics: Block Proximity Analysis

Block proximity analysis

- Creates a graph:
  - **Nodes**: Unique block types
  - **Edges**: Blocks used in the same GRC file.
  - Every co-occurrence increases edge weight by 1.
- This example was run
  - With `--mode proximityGraph`
  - On randomly selected sub-set of GRC files
Integrating Modeling Across the Stack with Aspen

Aspen
- Ontologies
  - Structured
  - Composable
  - Interoperable
- Parameterizable Performance Models
  - Concurrency
  - Work
  - Communication
  - Capacities
  - Synchronization
- Machine Model
  - Compute
  - Memory
  - Interconnects
- PFU API

Applications
- Streaming (e.g., SW and HW)
- Sensing (e.g., SAR, vision)
- Deep learning (e.g., CNN)
- Analytics (e.g., graphs)
- Robotics (e.g., sense and react)
- Science and Engineering (e.g., CFD)

Ontologies include Motifs, Parallel Execution Patterns, Locality, Affinity, Sync, etc.

Programming Systems
- Compiler
- DSL
- JIT
- Libraries
- etc

Performance Models combine Ontologies and Machine Models to produce both static and dynamic cost estimates

Runtime and Operating Systems
- Task scheduling
- Memory Mgt
- IO
- Synchronization

Machine Models include architecture configuration information, microbenchmarks, and API for PFU

DSSoC Hardware
- Performance Parameters
  - CPU
  - GPU
  - FPGA
  - Accelerator
  - DSP
  - Deep Memory
  - Neuromorphic
Aspen: Abstract Scalable Performance Engineering Notation

Model Creation
- Static analysis via compiler, tools
- Empirical, Historical
- Manual (for future applications)

Representation in Aspen
- Modular
- Sharable
- Composable
- Reflects prog structure

E.g., MD, UHPC CP 1, Lulesh, 3D FFT, CoMD, VPFFT, ...

Model Uses
- Interactive tools for graphs, queries
- Design space exploration
- Workload Generation
- Feedback to Runtime Systems

Source code

Aspen code

GNURadio Flowgraph to Aspen Application Model Conversion

model demod-uhd-sync{
    import uhd from "gr-uhd.aspen"
    import blocks from "gr-blocks.aspen"
    import digital from "gr-digital.aspen"

    kernel main {
        call uhd.usrp_source()
        call digital.fill_band_edge_cc()
        call digital.correlate_and_sync_cc()
    }

    parallel {
        sequence{
            parallel {
                call blocks.complex_to_mag()
                call blocks.complex_to_float()
            }
            call blocks_file_sink()
        }
        sequence{
            call digital.pfb_clock_sync()
            call digital.costas_loop_cc()
        }
    }
}

model blocks {
    kernel complex_to_mag {
        param aspen_param_default = 1
        param aspen_param_sizeof_float = 4
        param noutput_items = 8192
        param aspen_param_sizeof_FComplex = 8
        execute [noutput_items] "block_clComplexToMag_kernel14"
        flops [1] as integer
        execute "block_clComplexToMag_kernel14__intracommIN"
        intracomm [(aspen_param_sizeof_FComplex*noutput_items)] as copyin
        map [noutput_items] "mapblock_clComplexToMag_kernel14"
        execute "block_clComplexToMag_kernel17" {
            loads [(1*aspen_param_sizeof_FComplex)] as stride(1)
            loads [(1*aspen_param_sizeof_FComplex)] as stride(1)
            stores [(1*aspen_param_sizeof_float)] as stride(1)
            flops [4] as dp, simd
        }
        execute "block_clComplexToMag_kernel14__intracommOUT"
        intracomm [(aspen_param_sizeof_float*noutput_items)] as copyout
        kernel complex_to_float (...)
```cpp
class Zynq::Board-ZCU102 : Aspen::CompoundNode {
    // Processing units
    Zynq::APU cpu;
    ARM::Mali400MP2 gpu;
    ARM::CortexR5 rpu;
    Xilinx::UltraScale+<nFPUs=400M> fpga;
    // Memory
    Aspen::DDR3<$freq=2000MHZ, CL=16> systemMemory;
    // Memory controllers, switches, mmus
    Zynq::SMMU smmu;
    Aspen::Switch<$bw=100GBs, latency= 25ns> lpSwitch;
    Aspen::Switch<$bw=1TBs, latency= 35ns> centralSwitch;
    Aspen::PCIController<$ver=3, totalLanes=24> pciController;
    // Define interconnects (edges)
    Aspen::Bus<$bw=400GBps> cci_fp;
    Aspen::Bus<$bw=100GBps> cci_lp;
    Aspen::PCIe<$version=3,lane=16> pcieBus;

    @add
    cpu --cci_fp-- smmu;
    gpu --cci_fp-- smmu;
    fpga --cci_fp-- smmul
    systemMemory --cci_fp[2]-- smmu; // Multiple links
    smmu --cci_fp-- centralSwitch;
    smmu --cci_fp-- pciController;
    fpga --cci_fp[2]-- centralSwitch
    lpSwitch --cci_fp-- smmu; // Unidirectional link
    lpSwitch <<=cci_fp-- centralSwitch
    rpu --cci_l[2]-- lpSwitch;
    pciController --pcieBus --> Aspen::OUTPUT
    pciController <<=--pcieBus --> Aspen::INPUT
```

Belviranli, M E, et al, "FLAME: Graph-based Hardware Representations for Rapid and Precise Performance Modeling", DATE’19
Programming Systems

Applications
- Streaming (e.g., ISW Radio)
- Sensing (e.g., SAR, vision)
- Deep learning (e.g., CNN)
- Analytics (e.g., graphs)
- Robotics (e.g., sense and react)
- Science and Engineering (e.g., CFD)

Ontologies include Motifs, Parallel Execution Patterns, Locality, Affinity, Sync, etc.

Programming Systems
- Compiler
- DSL
- JIT
- Libraries
- etc

Performance Models combine Ontologies and Machine Models to produce both static and dynamic cost estimates

Runtime and Operating Systems
- Task scheduling
- Memory Mgt
- IO
- Synchronization

Machine Models include architecture configuration information, microbenchmarks, and API for PFU

DSSoC Hardware
- Performance Parameters
  - CPU
  - GPU
  - FPGA
  - Accelerator
  - DSP
  - Deep Memory
  - Neuromorphic
Heterogeneous Architecture Support by OpenARC

OpenACC/OpenMP Program

CUDA → Nvidia GPUs
OpenCL → AMD GPUs
OpenCL → Xeon Phis
Intel OpenCL → Intel FPGAs
MCL → DSSoCs
OpenMP → CPUs/Xeon Phis

OpenACC/OpenMP P Interoperability

Front End Compiler
Back End Compiler
OpenARC: Open Accelerator Research Compiler*

- OpenARC is the first open-sourced, OpenACC/OpenMP compiler supporting Altera FPGAs, in addition to NVIDIA/AMD GPUs and Intel Xeon Phis.

- OpenARC is a high-level intermediate representation based, extensible compiler framework, where various performance optimizations, traceability mechanisms, fault tolerance techniques, etc., can be built for the complex heterogeneous computing.

* OpenARC, Lee, HPDC ‘14.
Example Translation of Gnu Radio Module: OpenACC to MCL (Targeting General Heterogeneous Devices)

Input OpenACC code

Output MCL host code

Output MCL kernel code
• OpenARC automatically generates a structured Aspen performance model from the ported OpenACC code of the GNU Radio blocks.

• Aspen performance prediction tools digest the generated Aspen models and derive performance predictions for the target application.

COMPASS: A Framework for Automated Performance Modeling and Prediction
Example Translation of Gnu Radio Module: OpenACC to Aspen

Input OpenACC code

Output Aspen Application Model
Programming Systems

Applications
- Streaming (e.g., ISW Radio)
- Sensing (e.g., SAR, vision)
- Deep learning (e.g., CNN)
- Analytics (e.g., graphs)
- Robotics (e.g., sense and react)
- Science and Engineering (e.g., CFD)

Ontologies include Motifs, Parallel Execution Patterns, Locality, Affinity, Sync, etc.

Programming Systems
- Compiler
- DSL
- JIT
- Libraries
- etc

Runtime and Operating Systems
- Task scheduling
- Memory Mgt
- IO
- Synchronization

DSSoC Hardware
- Performance Parameters
  - CPU
  - GPU
  - FPGA
  - Accelerator
  - DSP
  - Deep Memory
  - Neuromorphic

Aspen
- Ontologies
  - Structured
  - Composable
  - Interoperable
- Parameterizable Performance Models
  - Concurrency
  - Work
  - Communication
  - Capacities
  - Synchronization
- Machine Model
  - Compute
  - Memory
  - Interconnects
- PFU API
• **Framework for programming extremely heterogeneous systems**
  • Programming model and programming model runtime
  • Maximize resource utilizations
  • Abstract low-level architecture details from programmers
  • Dynamically schedule work to available resources

• **Key programming features:**
  • Scheduler dispatches application tasks to available computing resources
  • **Asynchronous** execution of runnable tasks
  • Devices are managed by the scheduler and presented as “Processing Elements” to users
  • **Independent applications** submit tasks without having to synchronize with each other
  • Simplified APIs and programming model (e.g., compared to OpenCL)

• **Flexibility:**
  • Provides a scheduling framework in which new scheduling algorithm can be plugged in
  • **Multiple scheduling algorithms** co-exist
  • Users don’t need to port code when running on different systems
  • Executing tasks on different PEs doesn’t require user intervention or code modification
  • Resources allocated at the last moment
• Multiple applications submit tasks
• The IS algorithm dispatches each task to a PE
  – Ontology-based performance expectations
  – Contingent performance and power considerations
  – PFU thread results
• IS collects runtime introspective information
  – System status
  – Task execution characteristics (performance, power, occupancy, etc.)
  – Estimates completion time

---

**IS Architecture 1/2**

- Submitted Tasks
- IS Interface
- IS Algorithm
- PEs
- PFU Thread
## MCL Platforms

<table>
<thead>
<tr>
<th>System</th>
<th>CPU</th>
<th>GPU</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVIDIA Jetson TX1</td>
<td>4-core ARM A57</td>
<td>NVIDIA Maxwell</td>
<td></td>
</tr>
<tr>
<td>NVIDIA Xavier</td>
<td>8-core ARM A57</td>
<td>NVIDIA Volta</td>
<td>2xDLMA</td>
</tr>
<tr>
<td>Xilinx ZCU 102</td>
<td>4-core ARM A53</td>
<td>ARM Mali-400 MP2</td>
<td>2-core ARM R5</td>
</tr>
<tr>
<td>Apple iMac Pro</td>
<td>16-core Intel Xeon</td>
<td>ATI Radeon Vega</td>
<td></td>
</tr>
<tr>
<td>GPU compute node</td>
<td>2x 20-core Intel Xeon</td>
<td>NVIDIA Pascal</td>
<td></td>
</tr>
<tr>
<td>NVIDIA DGX1 P100</td>
<td>2x 20-core Intel Xeon</td>
<td>8x NVIDIA Pascal</td>
<td></td>
</tr>
<tr>
<td>NVIDIA DGX1 V100</td>
<td>2x 20-core Intel Xeon</td>
<td>4x NVIDIA Volta</td>
<td>Tensor cores</td>
</tr>
<tr>
<td>NVIDIA DGX1 V100</td>
<td>2x 20-core Intel Xeon</td>
<td>8x NVIDIA Volta</td>
<td>Tensor cores</td>
</tr>
<tr>
<td>IBM Witherspoon</td>
<td>2x 22-core IBM POWER9</td>
<td>6x NVIDIA Volta</td>
<td>Tensor cores</td>
</tr>
</tbody>
</table>

- Currently MCL runs on a variety of architectures and systems
- Time to port to a new system is $\leq 1$ hour (mostly due to installing OpenCL libraries)
- Same code runs out-of-the-box on new platforms
Conclusions

- **Domain Specific Systems on Chip (DSSoC)**
  - Developing software and architectures for specific domains
    - SDR
    - Heterogeneity by design

- **ORNL project**
  - SDR applications
  - Targeting diverse array of heterogeneous hardware including upcoming chips
  - Intelligent programming and runtime system
    - OpenARC
    - MCL
  - Aspen models provides predictions for design, programming, and runtime decisions

- **Modeling and simulation challenges**
  - Clean slate chip design is too open
    - Need hierarchy of models that allow early exploration through accurate design validation
  - Performance prediction will be necessary to make compilation and scheduling intelligent