



# Compute Cache Architecture for the Acceleration of Mission-Critical Data Analytics



**Dr. Herman Lam**

UF Site Director

**Dr. David Ojika**

Post Doctoral Associate

**Sharath Bhat**

**Karthikeyan Rajasekaran**

**Vishnu Srinivasan**

Research Assistants



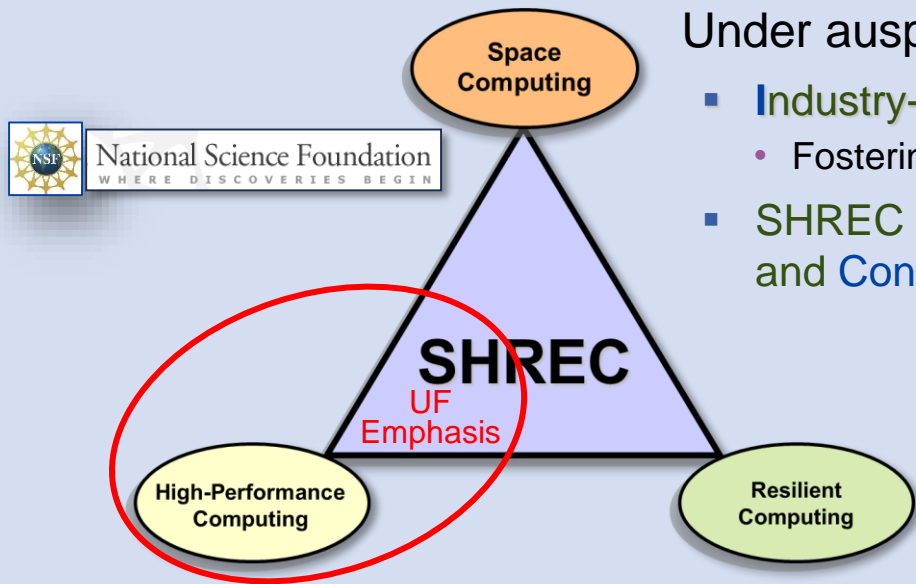
**NSF Center for Space,  
High-Performance, and Resilient  
Computing (SHREC)**

University of Florida

# Introduction to SHREC@UF\*



\* NSF Center for **S**pace, **H**igh-Performance, & **R**esilient **C**omputing



Under auspices of **IUCRC** Program at **NSF**

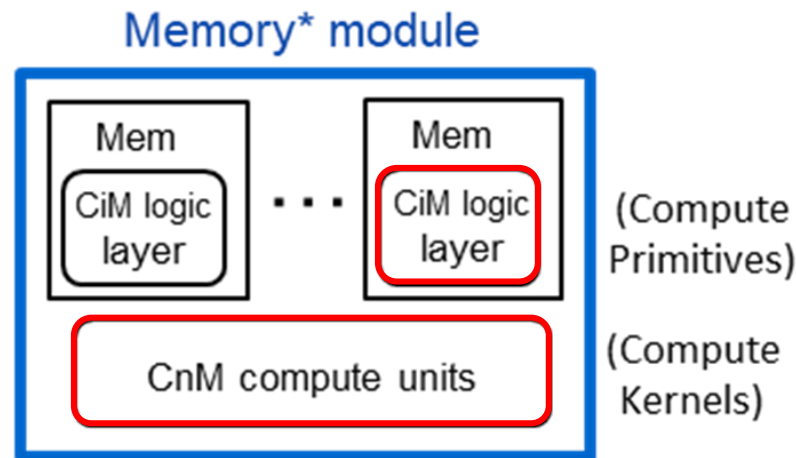
- **Industry-University Cooperative Research Centers**
  - Fostering university, agency, & industry R&D collaborations
- SHREC is both **National Research Center** (universities) and **Consortium** (member organizations)
  - University of Pittsburgh (lead site)
  - Brigham Young University
  - \* University of Florida (UF)
  - Virginia Tech

# Outline

- Overview: Compute caches
  - Compute near Memory (CnM), Compute in Memory (CiM):
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of **compute caches** in SLAMS\* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results



# Introduction: CnM and CiM



\* Ex: DDR/HBM memory; Storage-class memory

## ■ CnM: Compute-near-memory

- Ex. Intel Stratix-10 MX (HBM2 in same package)
- CnM compute units – *compute kernels* (e.g., FFT, Bloom filter) to accel. an app

## ■ CiM: Compute-in-Memory

- Ex. Emerging memory\* with built-in logic layer(s)
- CiM logic – *compute primitives* (e.g., add, mult, data-ordering ops)

**Approach:** Combination of *experimental* and *ModSim* studies

- **R&D Thrust 1:** Amplify the acceleration capabilities of FPGAs (CnM processing)
  - FPGA + (*Mem w/ low-latency, high-bandwidth*); e.g., larger MatMult, FFTs, DNN training
- **R&D Thrust 2:** System-level ModSim of heterogeneous HPC systems
  - Ex: Emerging HPC system: Some combination (CPU, accelerators, *compute cache\*\**)

# Outline

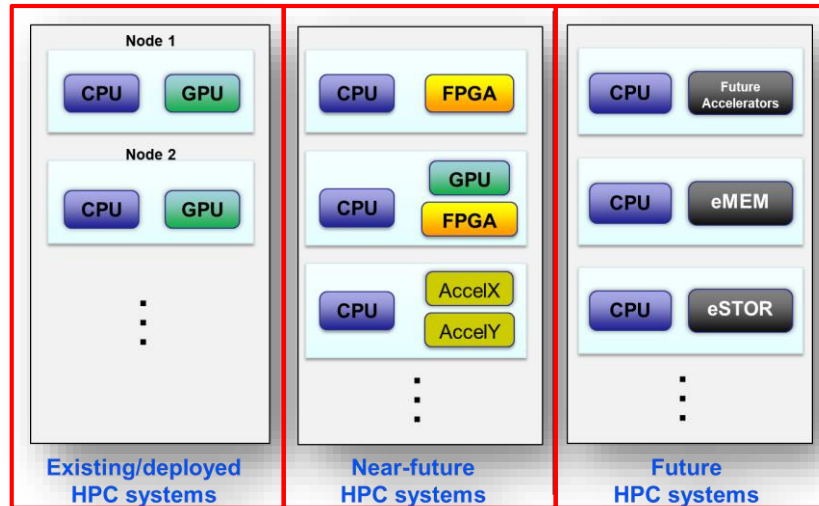
- Overview: Compute caches
  - Compute near Memory (CnM), Compute in Memory (CiM):
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of **compute caches** in SLAMS\* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results



# SLAMS<sup>1</sup> of Heterogeneous<sup>2</sup> HPC Systems

**Goal:** Explore methods, tools, and research directions:

- To meet the challenges of *System-Level* Application Modeling and Simulation of *emerging* and *future heterogeneous* HPC systems



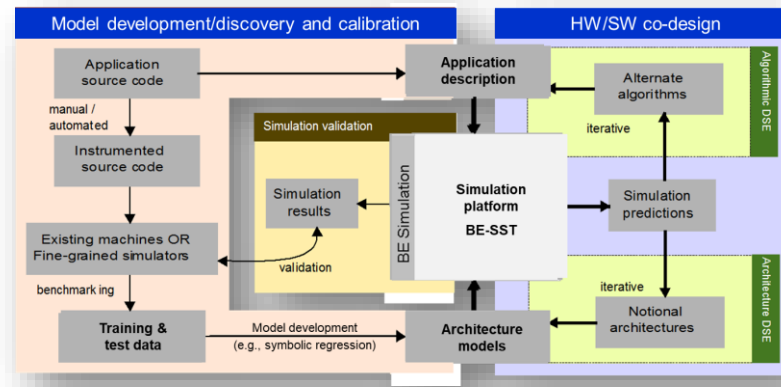
<sup>1</sup> SLAM: System-Level Application Modeling, IARPA-RFI-19-08, <https://www.iarpa.gov/index.php/working-with-iarpa/requests-for-information/slam>.

<sup>2</sup> 2018 Report for DOE ASCR Basic Research Needs Workshop on Extreme Heterogeneity <https://orau.gov/exheterogeneity2018/2018-Extreme-Heterogeneity-BRN-report-final.pdf>

- **Existing:** deployed complete HPC systems; available for benchmarking
- **Near-future:** no such deployed HPC systems exist yet
  - Containing desired accelerators (e.g., FPGAs)
  - Available nodes (e.g., servers with FPGAs) for benchmarking (ex. Stratix 10 MX)
- **Future:** no such deployed HPC systems exist yet
  - Device/node not available for benchmarking (eMEM\*, eSTOR\*, future accelerators)
  - Simulators available to collect data for modeling

\* eMem, eSTOR: emerging memory and storage (with CiM capabilities)

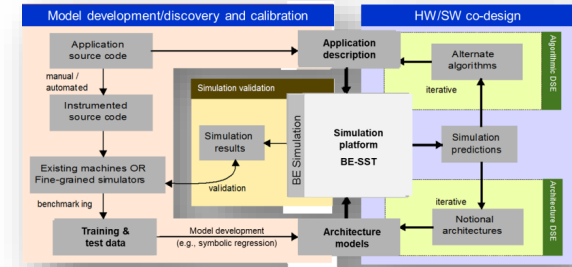
# Foundation: SLAMS of Hetero. HPC Systems



Based on current work at [Center of Compressible Multiphase Turbulence \(CCMT\)](#) at the University of Florida (for *homogeneous (CPU-only)* HPC systems)

- **Behavioral Emulation (BE):** a coarse-grained ModSim methodology for design-space exploration of applications on existing (e.g., [Vulcan](#), [Titan](#), [Quartz](#)) and notional HPC systems
- **BE-SST:** a distributed [parallel](#) simulation library for Behavioral Emulation, integrated into the [Structural Simulation Toolkit \(SST\)](#) from Sandia National Labs

# BE-SST Design Goals & Features



1. **Balance** of modeling *fidelity* and simulation *speed* (timeliness).

**Solution:** *coarse-grained* ModSim approach

2. **System-level** modeling and simulation

- Need to capture difficult-to-model app and machine-specific behaviors, without requiring deep knowledge of app and machine

**Solution:** *symbolic regression* modeling methods and tools

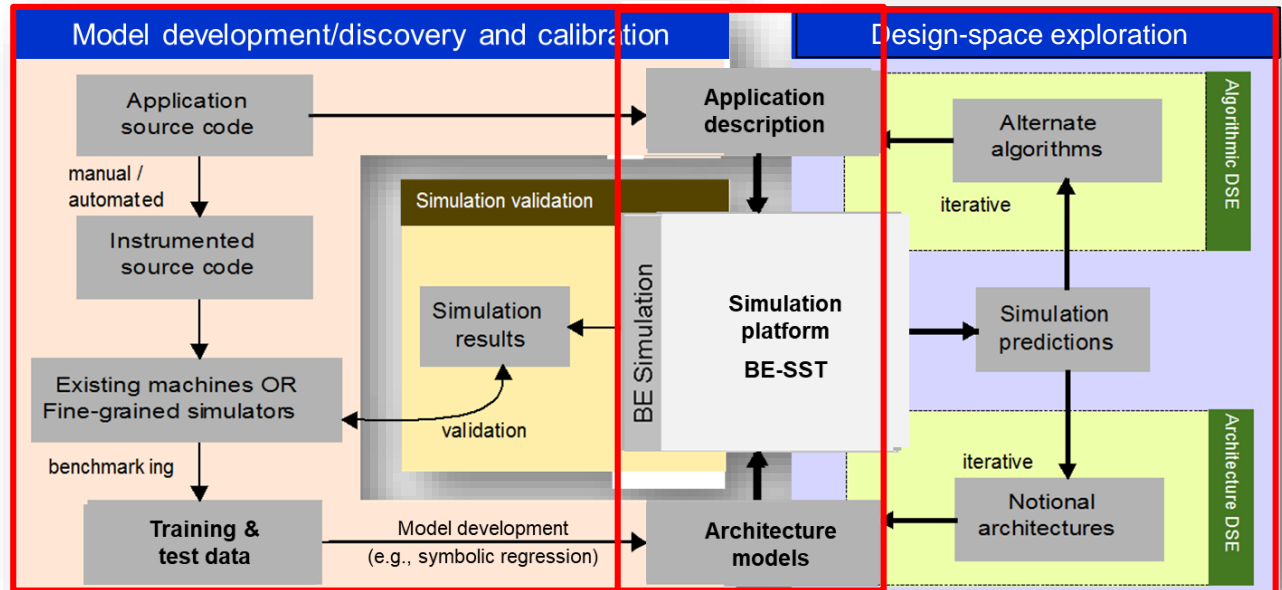
3. **ModSim** tools and system which are *scalable* and *useable* for other interested users.

**Solution:** integration with *SST (BE-SST)*



# BE Workflow

## HW/SW co-design Algorithmic & arch. design-space exploration (DSE)



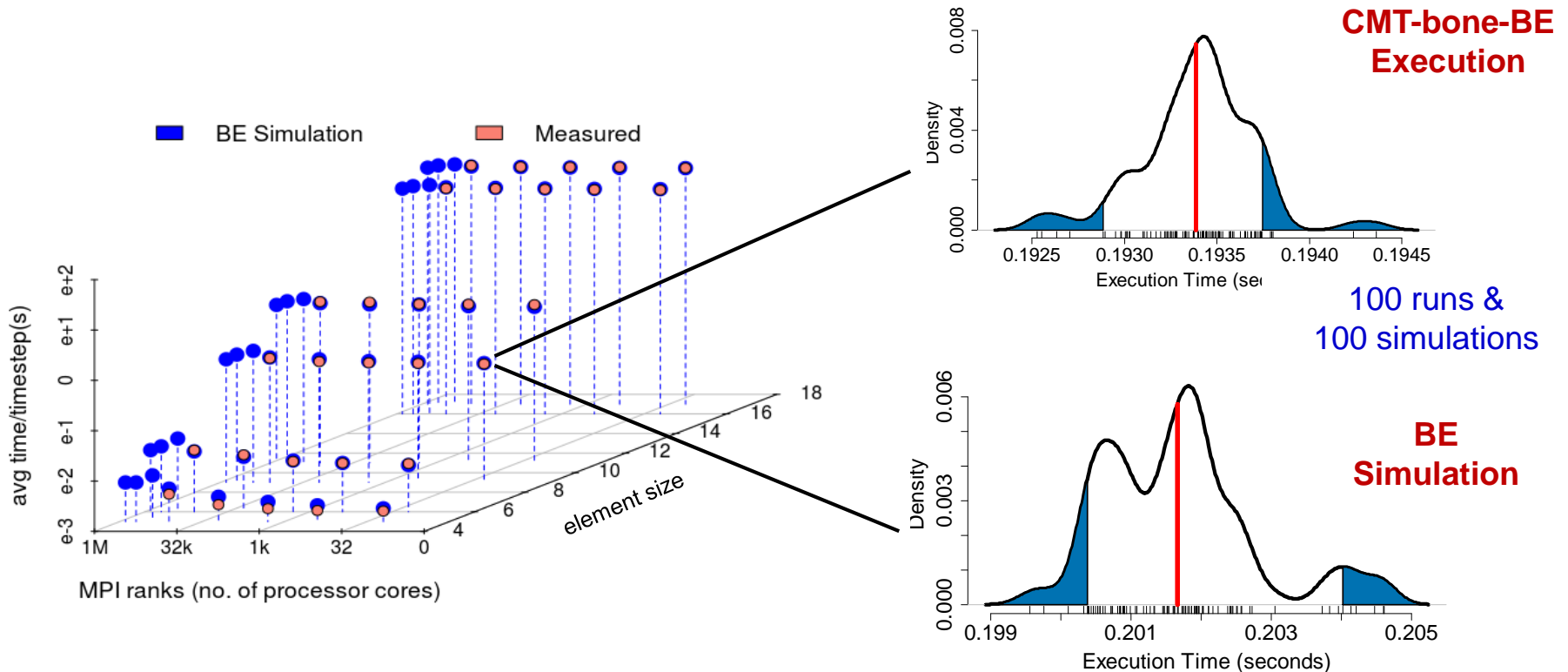
## BE-

- **Profile** code to determine key computational and communication kernels
- **Instrument & benchmark** code on HPC systems to collect training & test data (currently only homogeneous (CPU-only) HPC systems)
- **Develop/discover models** using training data to (e.g., *symbolic regression*)
- **Calibrate models** if necessary: compare simulation results vs. test data

## Design-space exploration phase:

- **Algorithmic & architectural** design-space exploration (DSE)

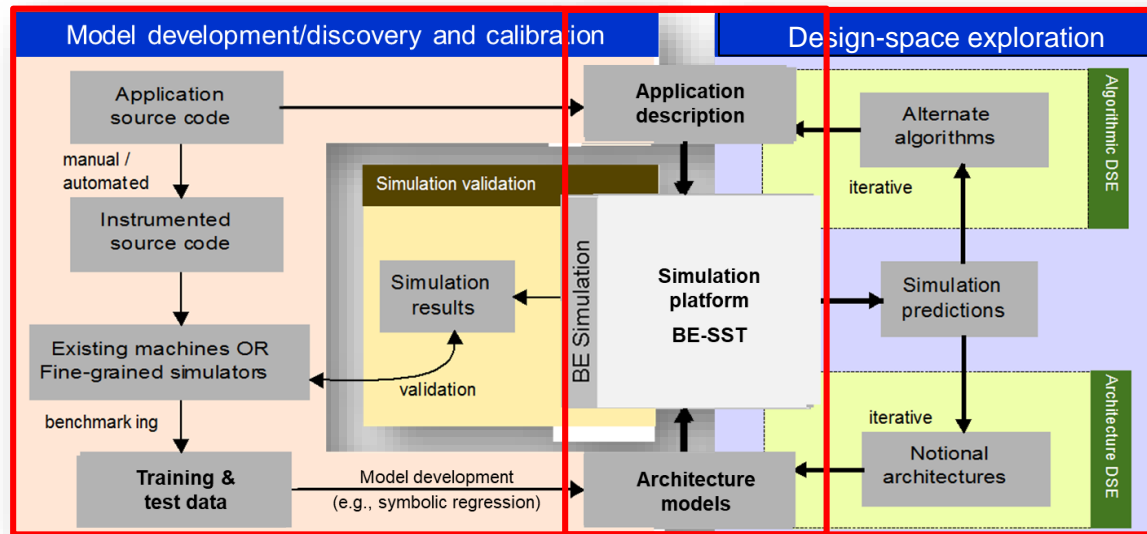
# BE Simulations of Titan+ (sample output)



- Simulating bigger system than Titan (approaching 1 Million cores)
- Average % error between CMT-bone-BE simulation and execution time is 4%
- Maximum error is 17%

$$\% \text{ error} = \frac{\frac{\sum_{i=1}^N t_{\text{measured } i}}{N} - \frac{\sum_{i=1}^N t_{\text{simulated } i}}{N}}{\frac{\sum_{i=1}^N t_{\text{measured } i}}{N}} \times 100$$

# What BE/BE-SST are NOT



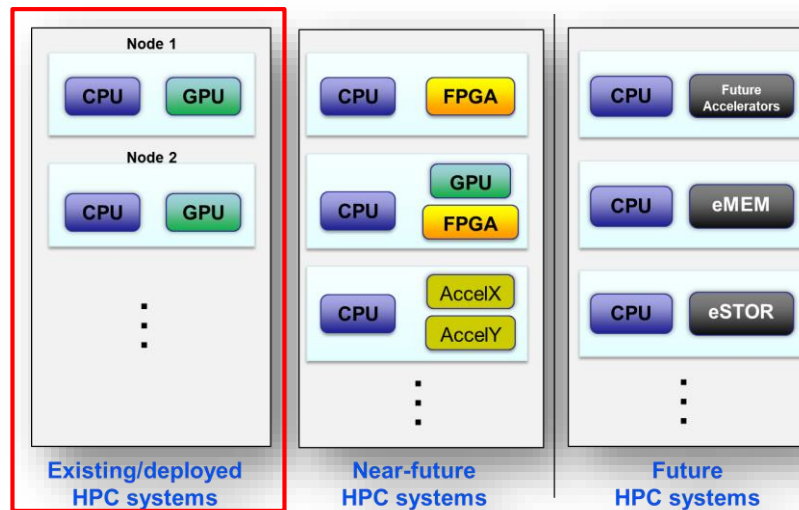
1. **Not** to “**componentize**” and model **every detail** of a complex system.
  - **Unless required**, difficult to model machine-specific behaviors **abstracted and discovered** through use of **training data & symbolic regression**
  - **Components under study** are modeled in detail
2. **Not** claim to solve the **general** co-design DSE problems
  - **Focus: application DSE** on given architecture and its **notional variations**

# Outline

- Overview
  - Compute near Memory (CnM); Compute in Memory (CiM)
  - Research directions D1, D2
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of **compute cache** in SLAMS\* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results



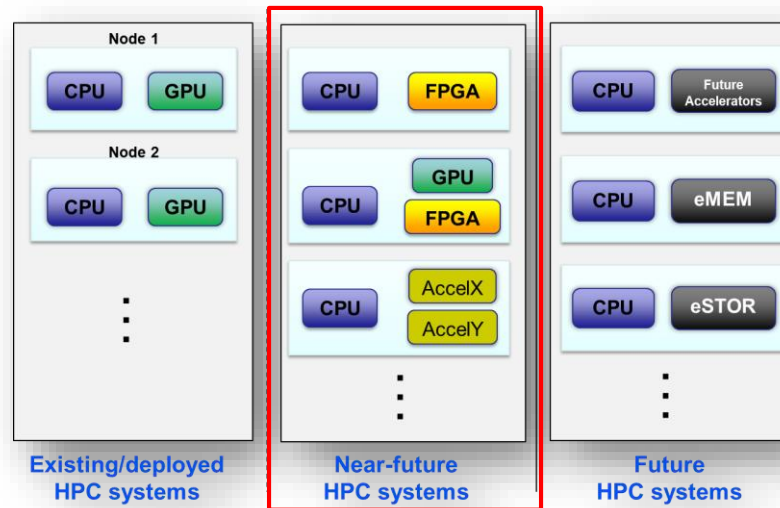
# SLAMS Approach: Existing HPC Systems



## Deployed *heterogeneous* HPC system

- Existing CPU+GPU HPC systems (e.g., Summit, Titan).
- Available for benchmarking
- We can use existing BE method for system-level ModSim of such systems
  - Instrument & benchmark code on nodes of HPC systems to collect training/test data
  - Develop/discover models (e.g., symbolic regression)
  - Perform design-space exploration

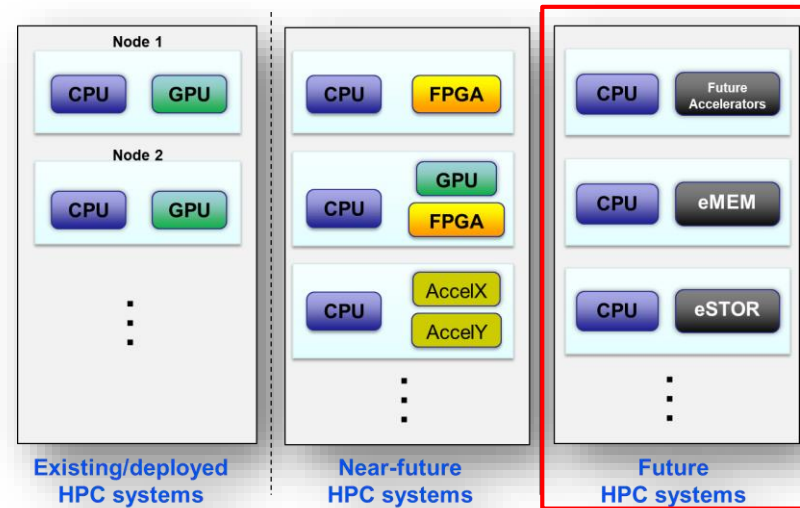
# SLAMS Approach: Near-Future HPC Systems



## Near-future *heterogeneous* HPC system

- No such deployed heterogeneous HPC systems yet (e.g., CPU+FPGA, CPU+GPU+FPGA)
- But independent nodes for such systems do exist (e.g., servers with FPGA boards)
- Approach: BittWare 520N-MX: w/ Stratix 10 MX for compute-near-memory (CnM) research
  - Instrument & benchmark code on available nodes to collect training/test data
  - Develop/discover models (e.g., symbolic regression)
  - Perform design-space exploration

# SLAMS Approach: Future HPC Systems



## Future HPC systems (augmented w/eMEM\*, eSTOR\*, future accelerators)

- Devices and nodes for future systems **do not exist**
- Cannot collect benchmarking data at the node level of such systems
- Approach
  - Depend on simulators to generate training/test data
  - Develop/discover models (e.g., symbolic regression)
  - Perform design-space exploration

Compute-in-memory (CiM) research

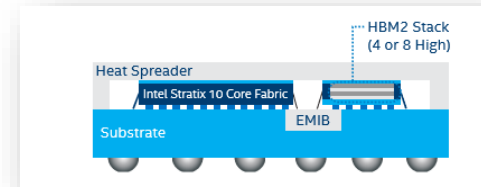
# Outline

- Overview
  - Compute near Memory (CnM); Compute in Memory (CiM)
  - Research directions D1, D2
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of compute cache in SLAMS\* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results



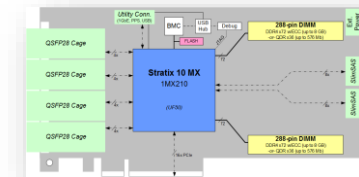


# Stratix 10 MX Board



## Bittware 520N-MX

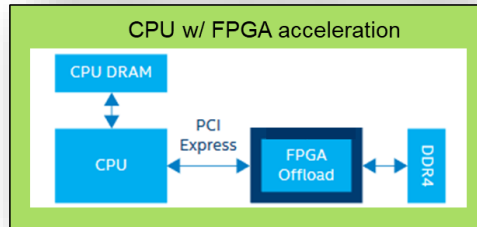
- Intel Stratix 10 MX 1MX210 FPGA
  - Up to 2 million logic elements
  - **16 GBytes HBM2 in same package**
- High Memory Bandwidth **512 Gbps**
- 2 PCIe hard IP blocks
- PCIe Interface - x16 PCIe Gen1, Gen2, Gen3
- Built-in Intel USB-Blaster
- UltraPort SlimSAS™ for serial expansion
- OpenCL support



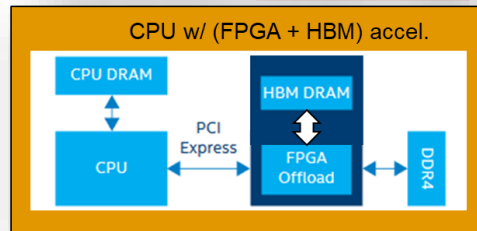
“Available 2019  
June/July August  
Late Sept/early October”

# CnM Processing for Kernel and App Acceleration

## Standard *acceleration w/ FPGA*

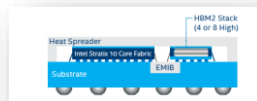


## Acceleration w/ *(FPGA + HBM)*



## Stratix 10 MX board

Stratix 10 + 16GB HBM2  
(in same package)



(Available 4<sup>th</sup> quarter 2019)

- FPGAs *effective as accelerators* for many data-analytics apps & kernels
  - As long as problem size can fit into the FPGA
- **Compute-*near-Memory* (CnN): Amplify** acceleration capabilities of FPGAs by exploiting *FPGA + nMEM\**
  - FPGA-accelerated computation
  - High-bandwidth, lower-latency access to HBM2 cache

## Apps Under Study

### High-bandwidth cache

- Very Large (VL) matrix multiply: dense linear algebra
- VL FFT
- Deep neural nets

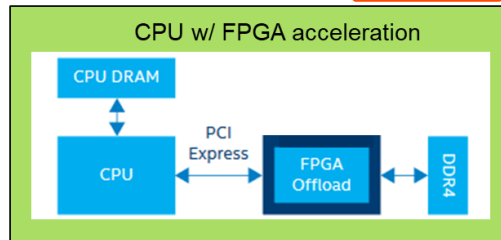
### Random access

- Bitonic sort: graph traversal
- BFS: graph traversal search
- Bloom filter: large volume of small random accesses

\* nMEM: near-memory e.g., Stratix 10 MX with in-package 16 MB HBM2

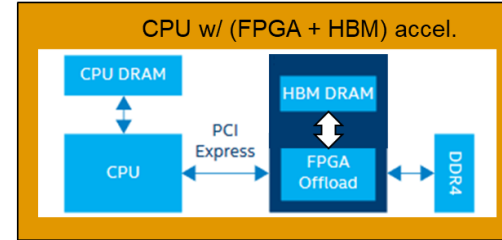
# Example Case Study CnM: FFT on 520N-MX

Standard *acceleration w/ FPGA*



vs.

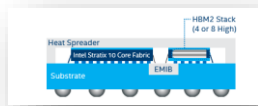
Acceleration w/ *(FPGA + HBM)*



## Compute near Memory: FPGA vs. FPGA + nHBM:

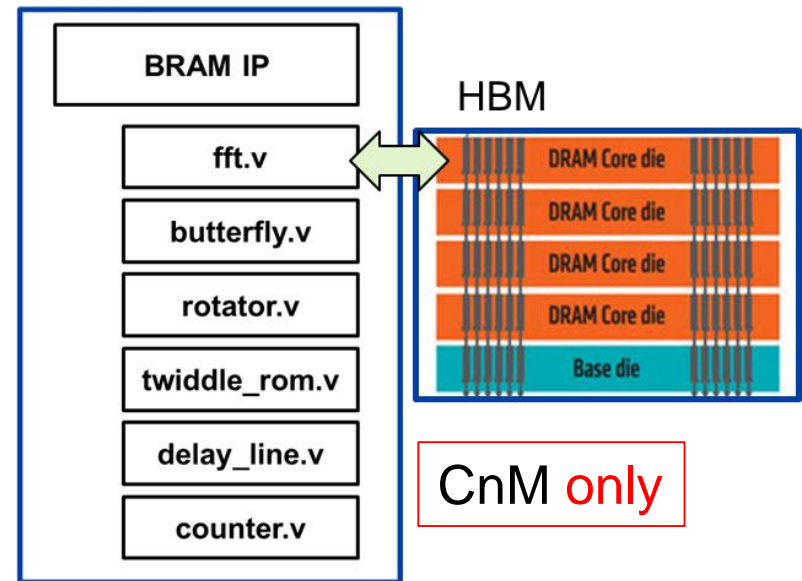
- Improve performance
- and/or FFT size

**Stratix 10 MX board**  
Stratix 10 + 16GB HBM2  
(in same package)

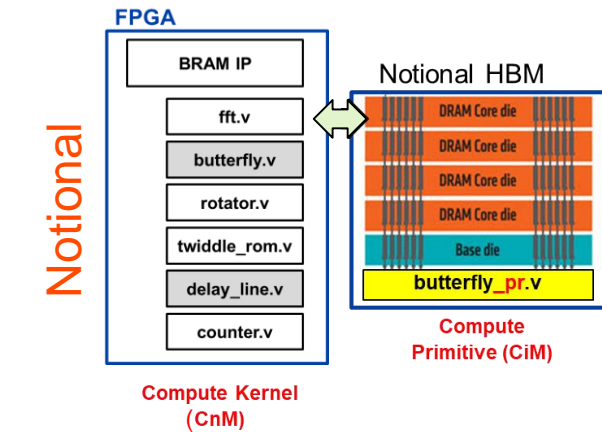


(Available 4<sup>th</sup> quarter 2019)

### FPGA



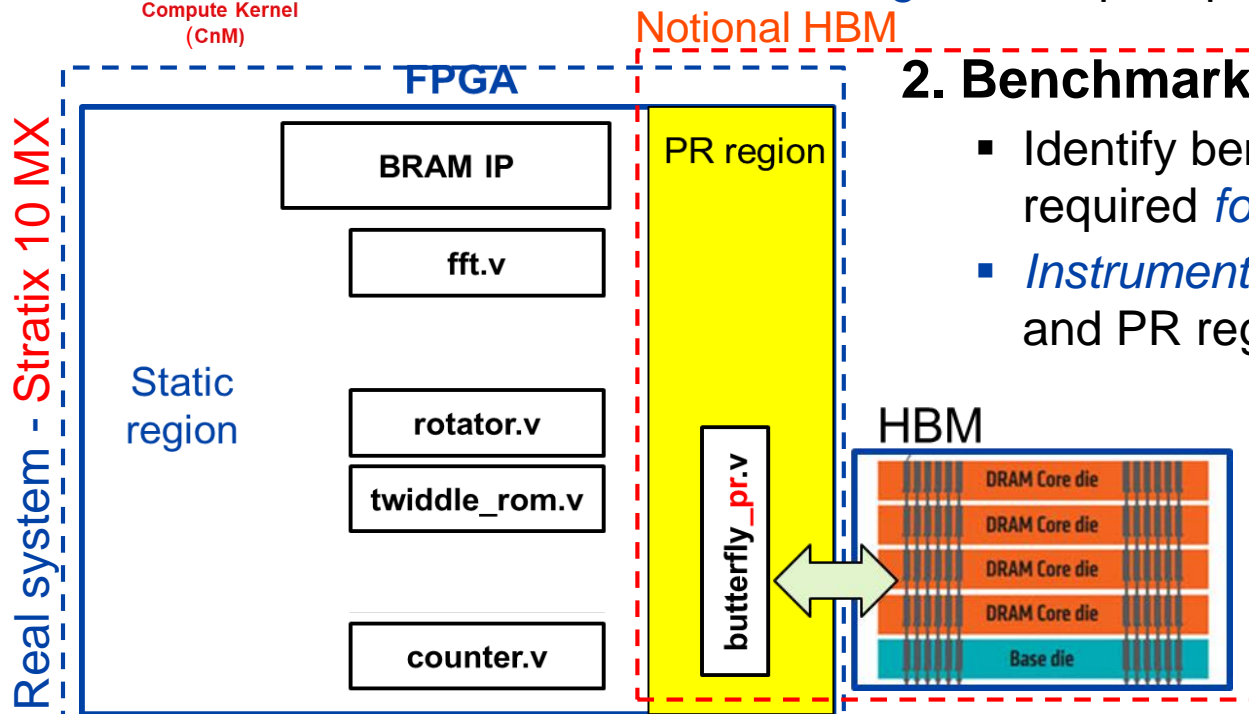
# CnM+CiM: 520N-MX Prototype & Benchmarking



## Approach:

### 1. Experimental prototype (Statix10 MX)

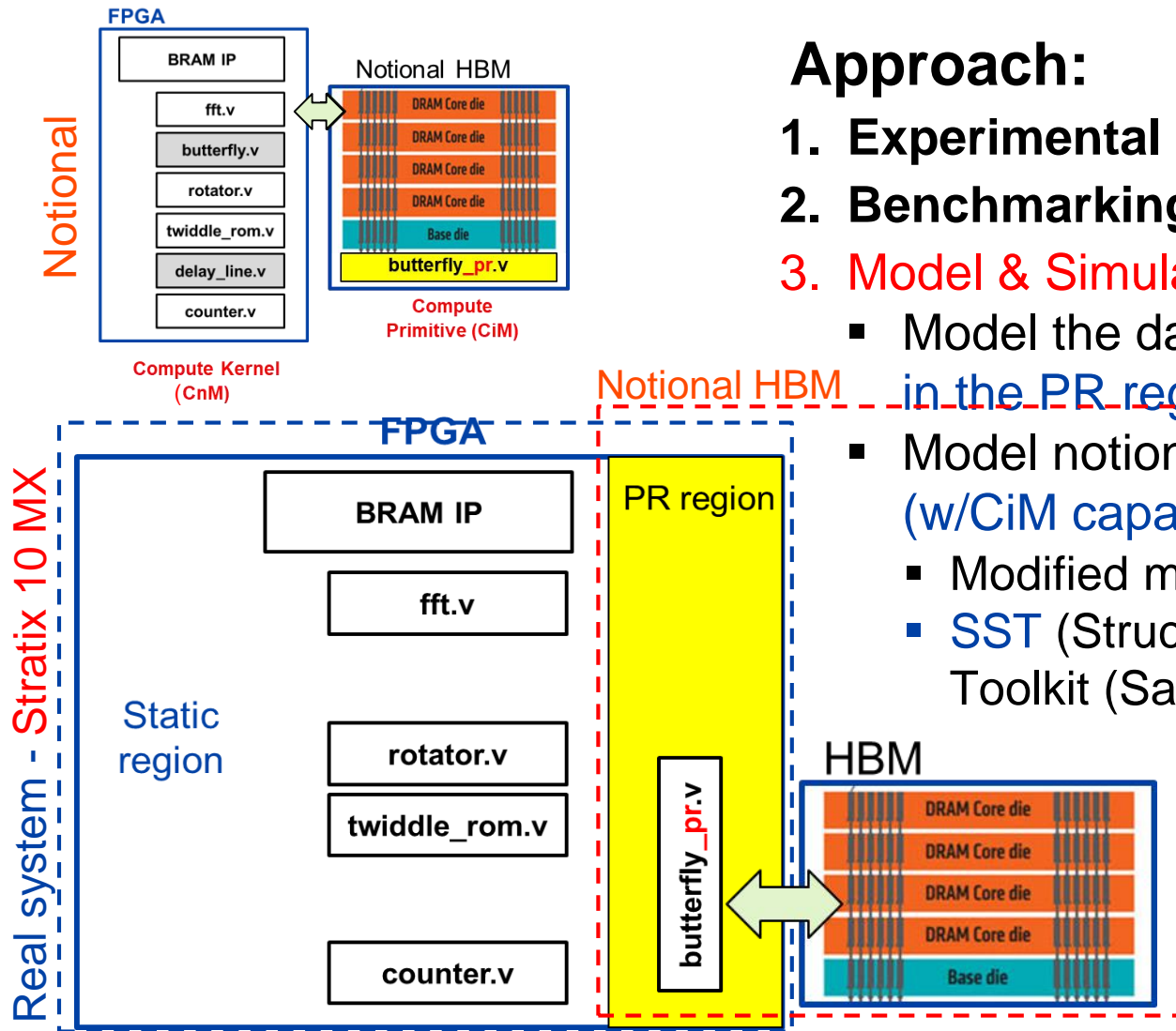
- Implement totally in FPGA, *using PR*
- **Static region:** compute kernel
- **PR region:** compute primitive(s)



### 2. Benchmarking/data collection

- Identify benchmarking data required *for model/simulation*
- *Instrument code* (both in static and PR regions) to collect data

# CnM+CiM: Model & Simulation



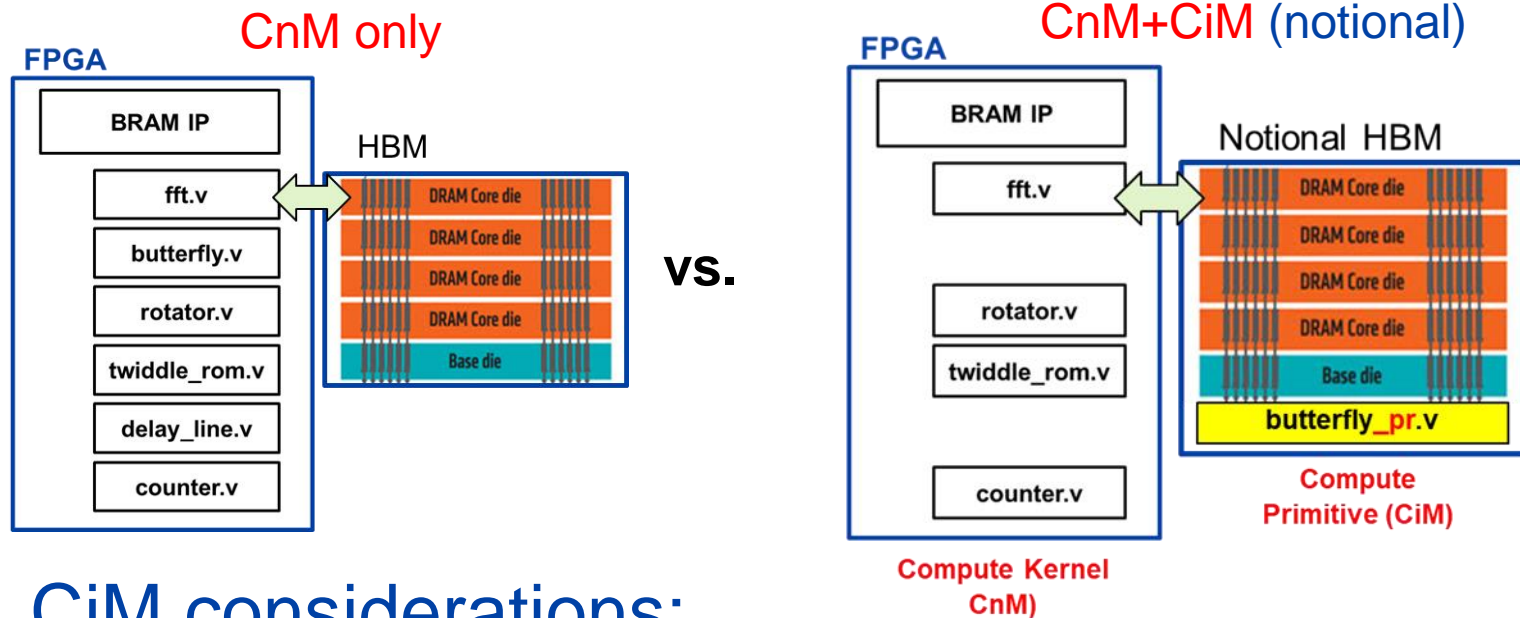
## Approach:

1. Experimental prototype
2. Benchmarking/data collection
3. Model & Simulation

- Model the data benchmarked in the PR region
- Model notional memory (w/CiM capability)
  - Modified memory simulator?
  - SST (Structured Simulation Toolkit (Sandia)?)

- Design-space exploration: CiM algorithms & architectures

# App/Kernel of Interest: CiM-amenable?



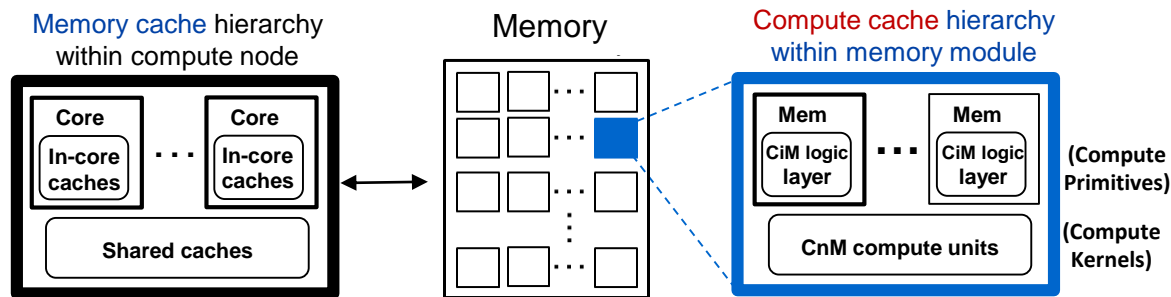
## CiM considerations:

- Can algorithm be *re-factored* to take advantage of CnM+CiM arch?
- Compute primitive characteristics
  - light-weight ops; reduces data transfer; not frequent interacting with compute kernel, can be pipelined with compute kernel ops, ... what else?...

# Summary: Compute Cache

*Compute cache* architecture for *heterogeneous* HPC system

- **Memory cache** hierarchy - bring *memory close to compute* devices
  - In-core caches & near-core shared caches



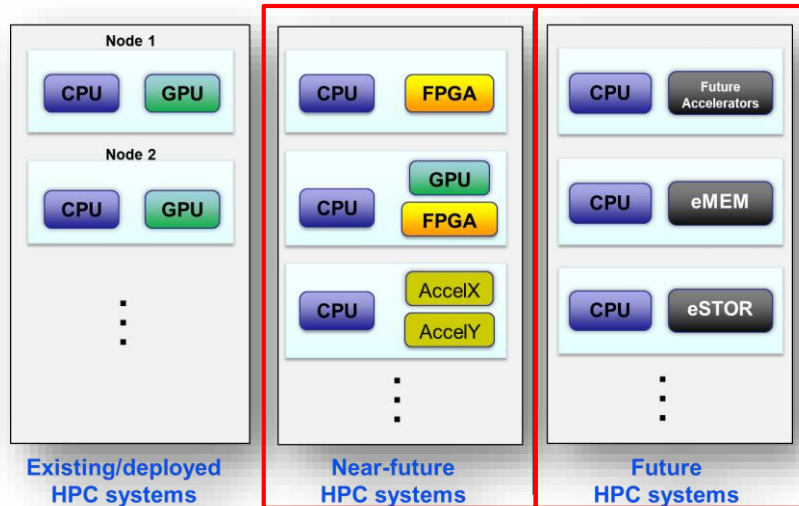
- Complementary **compute cache** hierarchy – bring *compute close to memory*
  - CiM logic – *compute primitives* (e.g., add, mult, data-ordering ops)
  - CnM compute – *compute kernels* (e.g., FFT, Bloom filter)

- **Goal:** System-level ModSim of emerging heterogeneous HPC systems
  - Emerging HPC system: Some combination (CPUs, accelerators, **compute caches**)

# SLAMS<sup>1</sup> of Heterogeneous<sup>2</sup> HPC Systems

**Going forward:** Explore methods, tools, and research directions:

- To meet the challenges of *System-Level* Application Modeling and Simulation of *emerging* and *future heterogeneous* HPC systems



<sup>1</sup> SLAMS: System-Level Application Modeling and Simulation, IARPA-RFI-19-08, <https://www.iarpa.gov/index.php/working-with-iarpa/requests-for-information/slam>.

<sup>2</sup> 2018 Report for DOE ASCR Basic Research Needs Workshop on Extreme Heterogeneity <https://orau.gov/exheterogeneity2018/2018-Extreme-Heterogeneity-BRN-report-final.pdf>

- Existing: deployed complete HPC systems; available for benchmarking
- **Near-future:** deployed HPC systems
  - Without desired accelerators (e.g., FPGAs)
  - Available nodes (e.g., servers with FPGAs) for benchmarking (ex. Stratix 10 MX)
- **Future:** deployed HPC systems
  - Device/node not available for benchmarking (eMEM, eSTOR, future accelerators)
  - Simulators available to collect data for modeling



# Outline

- Overview
  - Compute near Memory (CnM); Compute in Memory (CiM)
  - Research directions D1, D2
- *System-level* ModSim of *heterogeneous* HPC systems
  - Foundation of our approach
  - Approach: deployed, near-future, future heterogeneous HPC systems
  - Role of **compute cache** in SLAMS\* of such systems
- Current progress
  - Experimental platform:
    - Bittware 520N-MX, Intel Stratix 10 MX FPGA
  - Case study
    - In preparation for Bittware board
    - Preliminary results



# DISCUSSION

