National Science Foundation's Industry/University Cooperative Research (I/UCRC) Program

R&D Activities at SHREC@UF*
Using Intel FPGAs

Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE, AND RESILIENT COMPUTING (SHREC)

Herman Lam, *Site Director*

Greg Stitt

University of Pittsburgh · BYU Brigham Young University · Virginia Tech · UF University of Florida

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

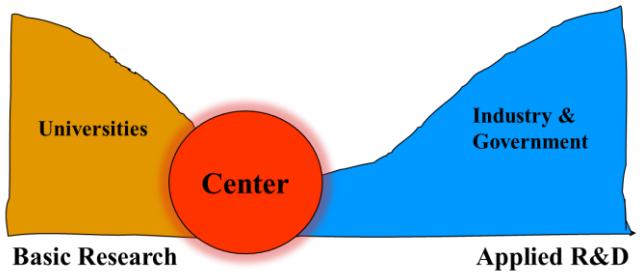
University of Florida

Outline

- Overview of NSF Center for Space, High-Performance, and Resilient Computing (SHREC)
- F1: Acceleration of Scientific Deep Learning Models using Intel FPGAs
- F1: Compute Cache for Compute-near-Memory & Compute-in-Memory Processing
- F2: FPGA Virtualization for High-Performance and Secure Application Design

What is SHREC?

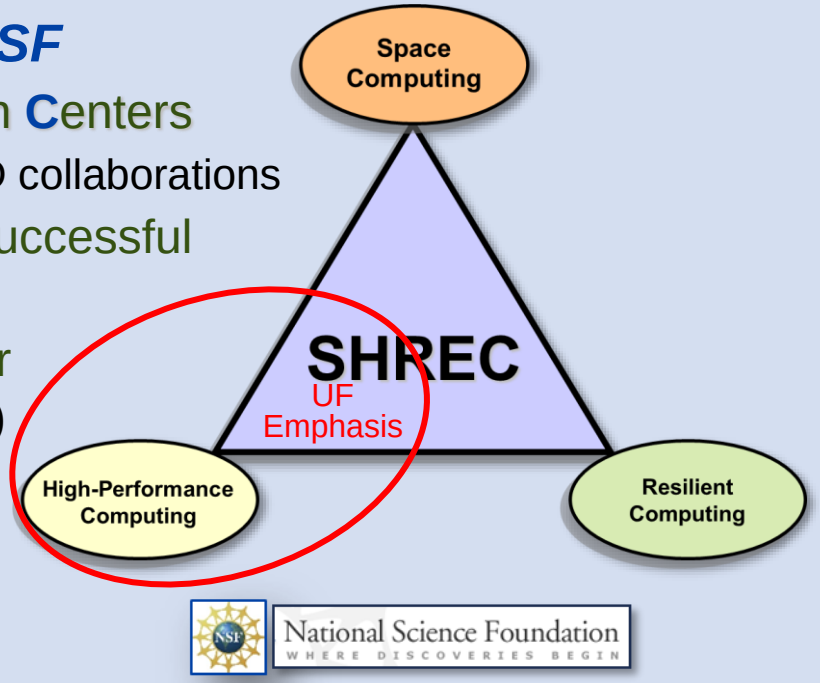
Mission-Critical Computing
NSF CENTER FOR SPACE, HIGH-PERFORMANCE, AND RESILIENT COMPUTING (SHREC)



***** NSF Center for Space, High-Performance, & Resilient Computing

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

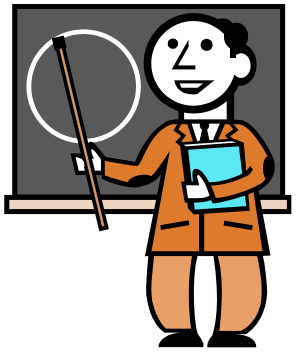
- **I**ndustry-**U**niversity **C**ooperative **R**esearch **C**enters
  - Fostering university, agency, & industry R&D collaborations
- Founded in Sep. 2017, replacing highly successful 10-year NSF CHREC Center
- SHREC is both National Research Center (universities) and Consortium (members)
  - University of Pittsburgh (lead site)
  - Brigham Young University
  - ***** University of Florida (UF)
  - Virginia Tech



Center Members (2019)

1. AFRL Sensors Directorate
2. AFRL Space Vehicles Directorate
3. Army Research Laboratory
4. BAE Systems
5. Ball Aerospace
6. Boeing
7. Collins Aerospace
8. Dell
9. Draper Lab
10. Emergent Space Technologies
11. Fermilab
12. Harris
13. Honeywell
14. Intel
15. L3 Space and Sensors
16. Laboratory for Physical Sciences
17. Lockheed Martin
18. Los Alamos National Laboratory
19. MIT Lincoln Laboratory
20. NASA Ames Research Center
21. NASA Goddard Space Flight Center
22. NASA IV&V Facility
23. NASA Johnson Space Center
24. NASA Kennedy Space Center
25. NASA Langley Research Center
26. National Reconnaissance Office
27. National Security Agency
28. Naval Research Laboratory
29. Raytheon
30. Sandia National Laboratories
31. Satlantis
32. Space Micro
33. Walt Disney Animation Studios

Center Faculty



- **University of Pittsburgh (lead)**
  - **Dr. Alan George**, Mickle Chair Professor of ECE – *Founder & Director*
  - **Dr. Alex Jones**, Professor of ECE – Associate Director
  - **Dr. Ryad Benosman,** Professor of Ophthalmology and ECE
  - **Dr. Jingtong Hu**, Assistant Professor of ECE
  - **Dr. Brandon Grainger**, Assistant Professor of ECE
  - **Dr. Wei Gao**, Associate Professor of ECE
- **Brigham Young University**
  - **Dr. Michael Wirthlin**, Professor of ECE – *Co-Director*
  - **Dr. Brent Nelson**, Professor of ECE
  - **Dr. Brad Hutchings**, Professor of ECE
  - **Dr. Jeff Goeders**, Assistant Professor of ECE
- **University of Florida**
  - **Dr. Herman Lam**, Associate Professor of ECE – *Co-Director*
  - **Dr. Greg Stitt**, Associate Professor of ECE
  - **Dr. Ann Gordon-Ross**, Associate Professor of ECE
  - **Dr. Janise McNair**, Associate Professor of ECE
  - **Dr. David Ojika**, Research Associate
- **Virginia Tech**
  - **Dr. Wu Feng**, Professor of ECE and CS – *Co-Director*
  - **Dr. Chris North**, Professor of CS

Most importantly, SHREC features an exceptional team of students spanning our university sites

Outline

- Overview of NSF Center for Space, High-Performance, and Resilient Computing (SHREC)
- F1: Acceleration of Scientific Deep Learning Models using Intel FPGAs
- F1: Compute Cache for Compute-near-Memory & Compute-in-Memory Processing
- F2: FPGA Virtualization for High-Performance and Secure Application Design

Outline

- Overview of NSF Center for Space, High-Performance, and Resilient Computing (SHREC)
- F1: Acceleration of Scientific Deep Learning Models using Intel FPGAs
- F1: Compute Cache for Compute-near-Memory & Compute-in-Memory Processing
- F2: FPGA Virtualization for High-Performance and Secure Application Design

Heterogeneous Computing[1] for Deep Learning

Motivation

- Deep learning becoming *pervasive* for mission-critical computing
  - *Heterogeneous computing*[1] offers unique capabilities to accelerate DNNs[2]

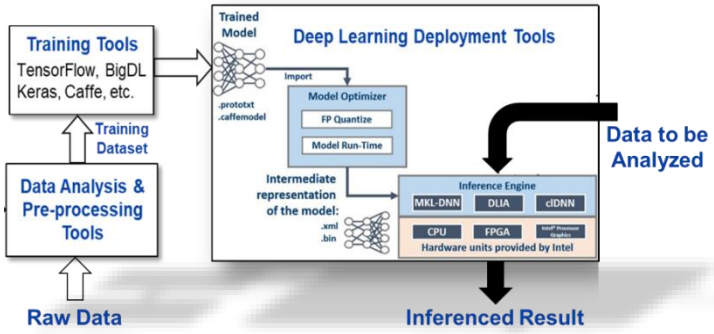
Goal

Perform design-space exploration:

- Of *emerging HGC*[1] *archs/tools* and *DNN*[2] *models*
- For *acceleration* of selected mission-critical apps

Approach

Focus on use of FPGAs to *accelerate inference stage* of the HGC workflow



Stages of HGC workflow

- Data analysis & pre-processing
- Model training
- DNN inference

DNN Models *from NERSC & CERN Openlab*

- HEP-CNN
- CosmoGAN
- 3D GAN

Experimental platforms & tools

- Dell EMC server: 2x Intel Xeon Gold 6130 CPU
- Intel PAC Arria 10 GX FPGA
- Intel OpenVino Toolkit & Deep Learning Accelerator (DLA) suite

[1] HGC: Heterogeneous Computing (CPU+GPU+FPGA)

[2] DNNs: Deep Neural Networks

Outline

- Overview of NSF Center for Space, High-Performance, and Resilient Computing (SHREC)
- F1: Acceleration of Scientific Deep Learning Models using Intel FPGAs
- F1: Compute Cache for Compute-near-Memory & Compute-in-Memory Processing
- F2: FPGA Virtualization for High-Performance and Secure Application Design

Compute Cache Arch. for Data-Analytics Apps

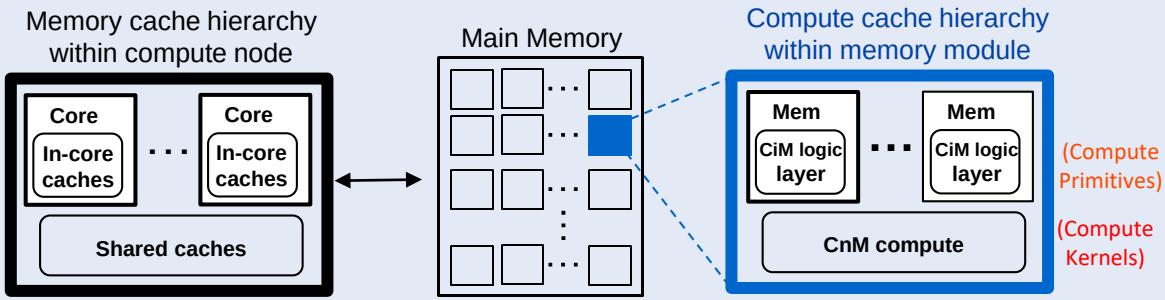
Motivation

- *Memory bottleneck -* critical for *memory-intensive* data-analytics apps
- Promise of CnM[1] & CiM[2] architectures and apps

Goal

Explore and evaluate emerging CnM & CiM *apps, archs,* and *tools* for next-gen mission-critical computing

Approach



Memory cache hierarchy

- Bring *memory close to compute*
- Near-core shared caches
- In-core caches

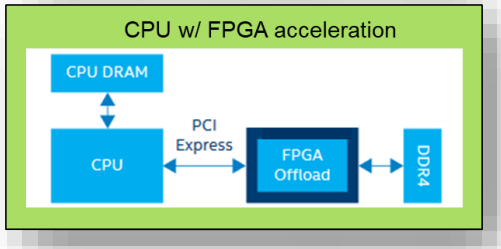
Complementary compute cache hierarchy

- Bring *compute close to memory*
- **CnM** *compute kernels* (e.g., FFT, Bloom filter)
- **CiM** *compute primitives* (e.g., add, data-ordering ops)

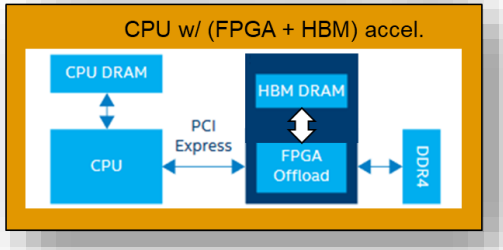
[1] CnM: Compute-near-Memory [2] CiM: Compute-in-Memory

CnM Processing for Kernel and App Acceleration

Standard *acceleration w/ FPGA*

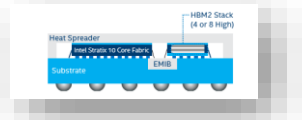


Acceleration w/ (*FPGA + HBM*)



- 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 + "near" HBM2*
  - FPGA-accelerated computation
  - High-bandwidth, lower-latency access to HBM2 cache

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



(Available 3rd quarter 2019)

Apps Under Study

In development & simulation in preparation for arrival of board

High-bandwidth cache

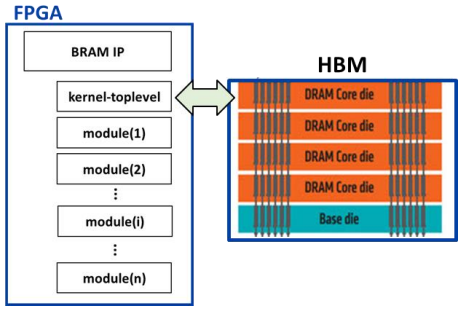
- Very Large (VL) matrix multiply: dense linear algebra
- VL FFT
- Deep neural nets

Random access

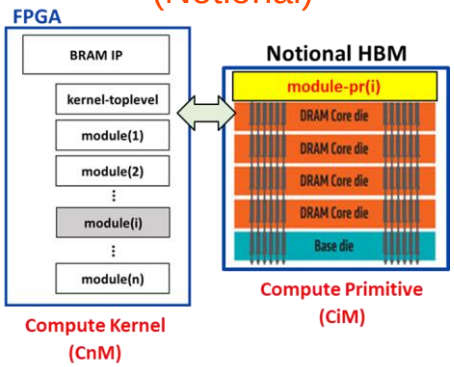
- Bitonic sort: graph traversal
- BFS: graph traversal search
- Bloom filter: large volume of small random accesses

Modeling & Simulation for Notional CiM Studies

Compute-near-Memory (only)
Ex. Stratix 10 MX board



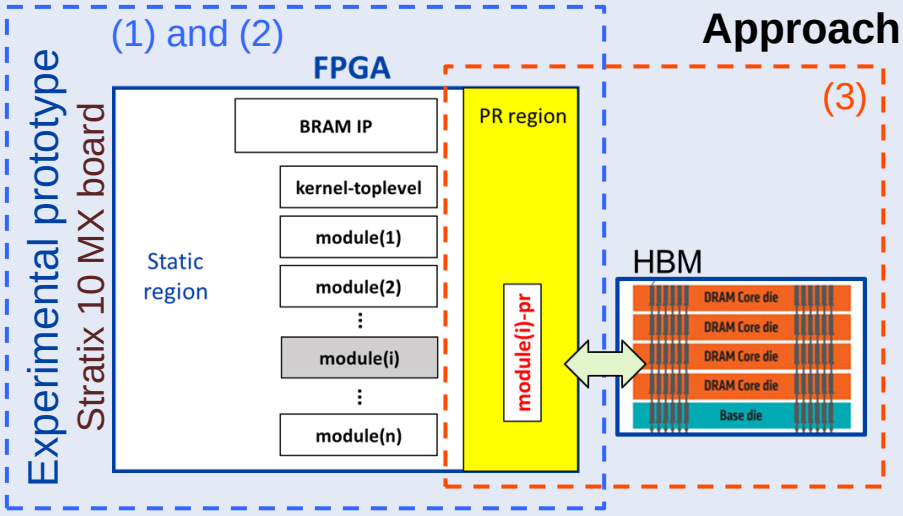
CnM + CiM
(Notional)



CiM considerations:

- Can algorithm be *re-factored* to take advantage of CnM+CiM arch?
- Compute primitive characteristics
  - light-weight ops; reduces data transfers; limited interaction with compute kernel, can be pipelined with compute kernel ops; … what else?…

Modeling & Simulation for Notional CiM Studies



Approach: (1) Experimental prototype
- Implement kernel entirely in FPGA, *using PR**
- Static region: compute kernel
- *PR region:* compute primitive(s)

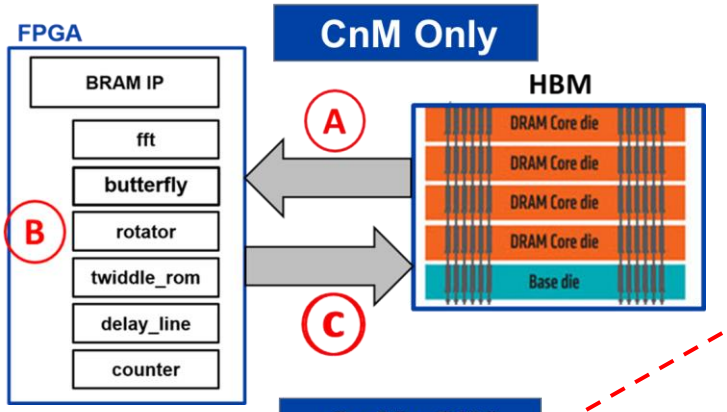
(2) Benchmarking & data collection
- Instrument code (both in static and PR regions) to collect data

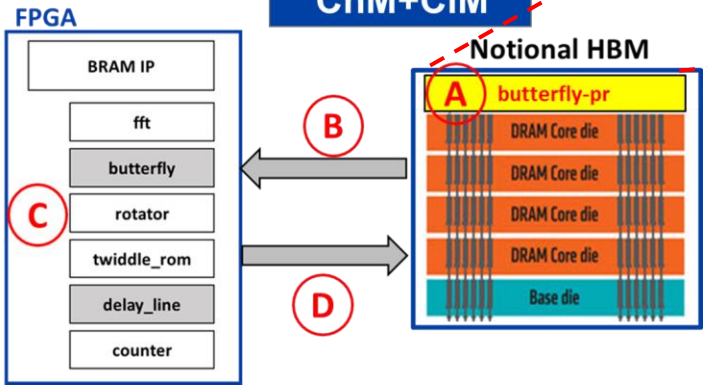
(3) Model & Simulation
- *Model* data benchmarked in the PR region (notional memory w/CiM capability)
- Design-space exploration using *simulation***

* PR: Partial Reconfiguration

** To be determined: memory simulators? SST (Structural Simulator Toolkit Sandia National Labs)?

Initial Case Study: FFT example

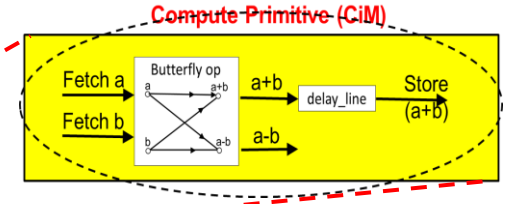




A
1. Read Real(a)
2. Read Im(a)
3. Read Real(b)
4. Read Im(b)

B
- Butterfly ops
- twiddle multiply by rotator

C
5. Write Real(a+b)
6. Write Im(a+b)
7. Write Real(W*(a-b))
8. Write Im(W*(a-b))

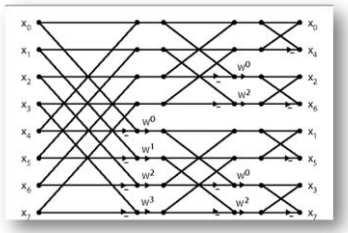


A
1. Fetch Real(a)
2. Fetch Im(a)
3. Fetch Real(b)
4. Fetch Im(b)
5. Butterfly ops
6. Store delayed Real(a+b)
7. Store delayed Im(a+b)

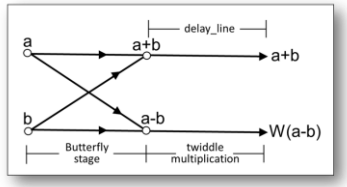
B
1. Read Real(a-b)
2. Read Im(a-b)

C
Twiddle multiply by rotator

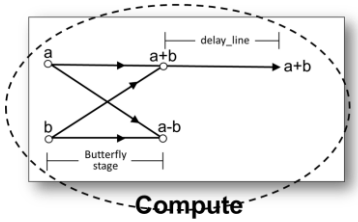
D
3. Write Real(W*(a-b))
4. Write Im(W*(a-b))



DIF-FFT



DFT Stage



Compute Primitive

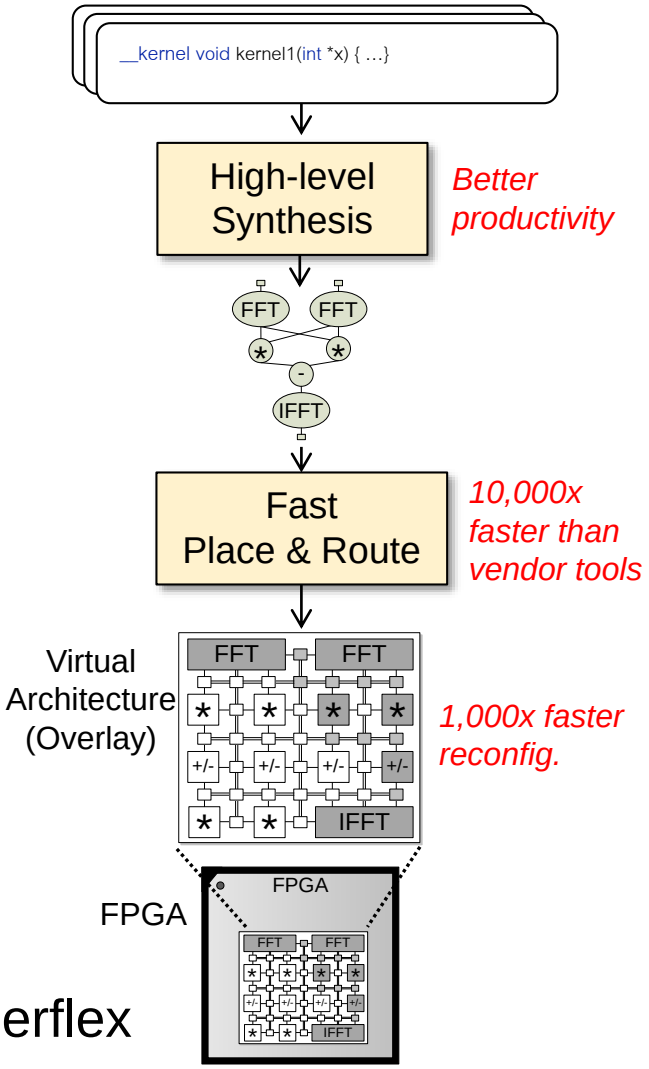
- In terms of memory accesses (not considering any overhead), *a reduction of 50% (4 vs. 8 memory accesses)*
- Ongoing detailed analysis: *prototype* (Stratix 10 MX board), *benchmark, model, simulation*.

Outline

- Overview of NSF Center for Space, High-Performance, and Resilient Computing (SHREC)
- F1: Acceleration of Scientific Deep Learning Models using Intel FPGAs
- F1: Compute Cache for Compute-near-Memory & Compute-in-Memory Processing
- F2: FPGA Virtualization for High-Performance and Secure Application Design

FPGA Virtualization: "Overlays"

- FPGAs suffer from **severe productivity bottlenecks**
- Virtualized architectures (overlays) abstract away fine-grained FPGA architecture
  - 10,000x faster compilation than vendor tools
- Can be paired with HLS for increased productivity
- Portability across different physical FPGAs
- 1000x faster reconfiguration than FPGAs

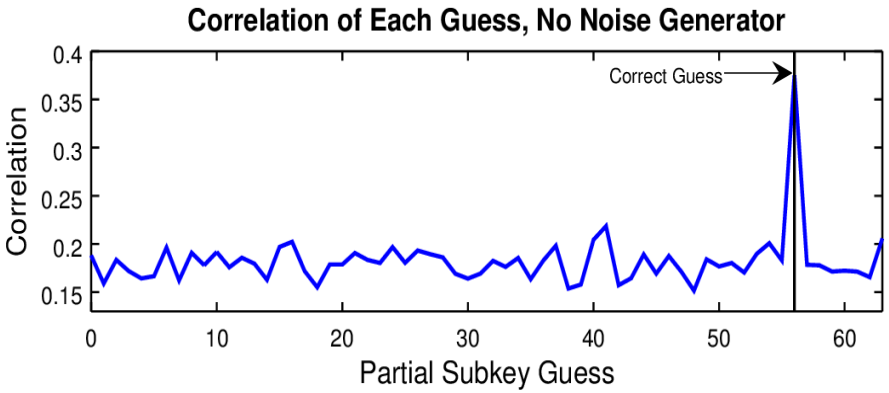


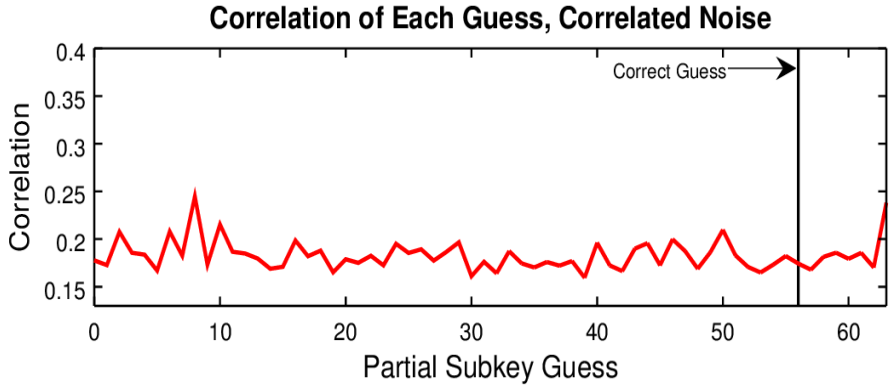
Some emerging benefits

- Overlay for hardware security
- Improve clock frequency: High-frequency absorption-FIFO pipelining for Stratix 10 Hyperflex

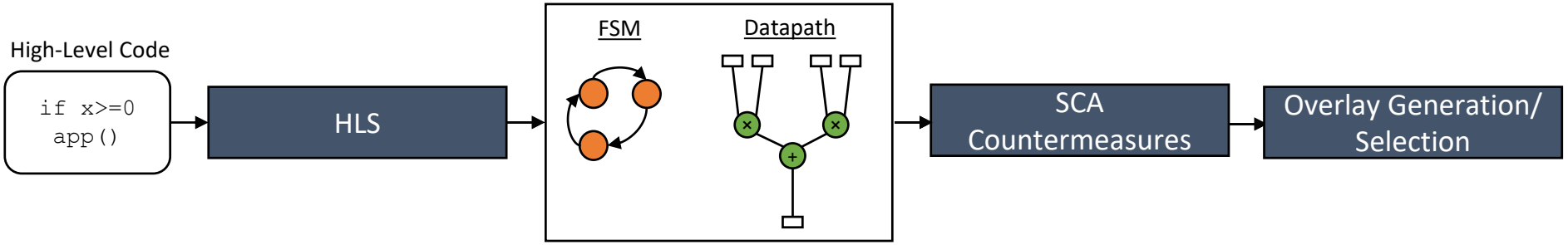
Overlays for Hardware Security

- FPGAs susceptible to side-channel analysis (SCA) attacks
  - Virtual architectures can protect application IP even on vulnerable FPGAs
  - **Reverse-engineered FPGA bitfile only reveals overlay, not application**
  - Proof of concept: noise generator DPA* countermeasure in overlay [MWSCAS '17]





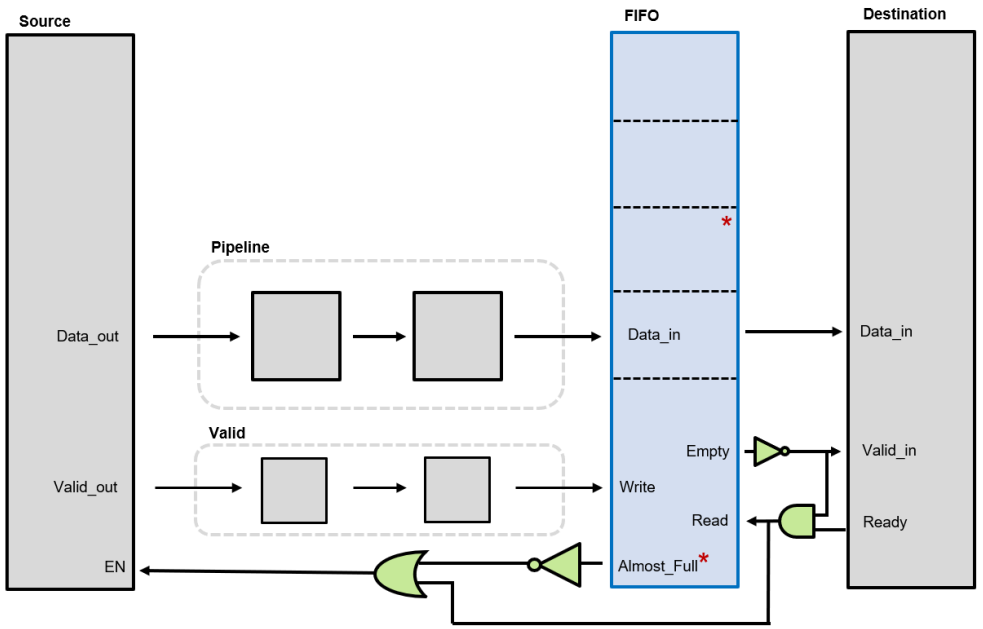
- Overall goal: integrate generic countermeasures during high-level synthesis (HLS)
  - Allows non-experts to automatically implement secure hardware designs



* DPA: Differential Power Analysis

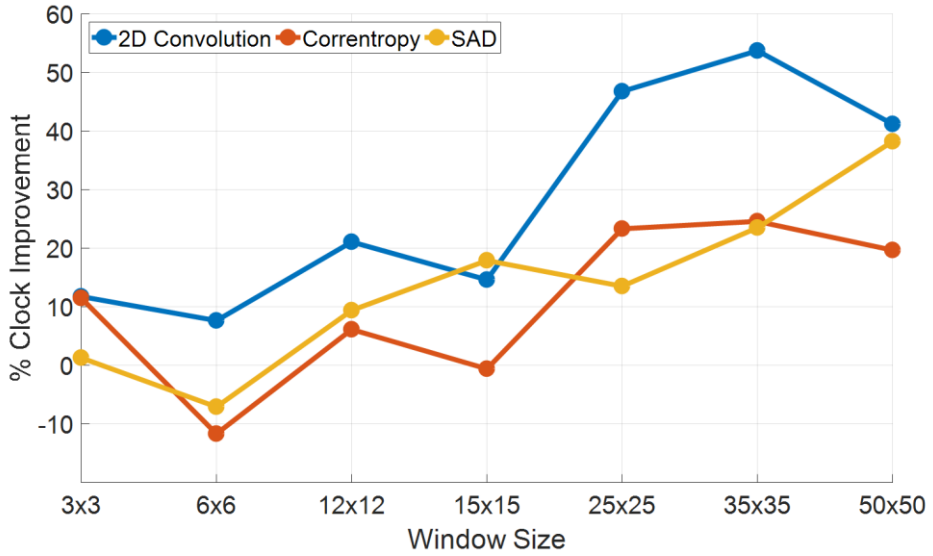
Improve Clock Frequency: High-Frequency Absorption-FIFO Pipelining for Stratix 10 Hyperflex

- FPGAs incur propagation delay from the reconfigurable interconnect
- Stratix 10 Hyperflex Architecture adds Hyper-Registers to interconnect to achieve higher clock frequencies
- **Problem: Hyper-Registers do not support backpressure (i.e. stalls)**
- Solution: Pipelining with Absorption FIFOs
  - Supports back-pressure with identical stall penalties as traditional pipelines
  - Enables clock improvements in devices with and without Hyperflex



Results : Stratix 10

- Achieved an increase in clock frequency with low-area overhead
  - Required RAM words $= 2^{\text{ceil}(\log_2(\text{pipeline depth}+1))}$
- Maximum clock frequency increase of 196MHz, 54% improvement
- Average improvement of 87MHz, 17% improvement



Goal: build clock optimizations like absorption FIFOs into overlays to provide performance improvements with no designer effort

Outline

- Overview of NSF Center for Space, High-Performance, and Resilient Computing (SHREC)
- F1: Acceleration of Scientific Deep Learning Models using Intel FPGAs
- F1: Compute Cache for Compute-near-Memory & Compute-in-Memory Processing
- F2: FPGA Virtualization for High-Performance and Secure Application Design

QUESTIONS