Accelerating Weather Prediction using Near-Memory Reconfigurable Fabric

GAGANDEEP SINGH, ETH Zürich, Switzerland
DIONYSIOS DIAMANTOPOULOS, IBM Research Europe, Zürich Lab, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
CHRISTOPH HAGLEITNER, IBM Research Europe, Zürich Lab, Switzerland
SANDER STUIJK, Eindhoven University of Technology, The Netherlands
HENK CORPORAAAL, Eindhoven University of Technology, The Netherlands
ONUR MUTLU, ETH Zürich, Switzerland

Ongoing climate change calls for fast and accurate weather and climate modeling. However, when solving large-scale weather prediction simulations, state-of-the-art CPU and GPU implementations suffer from limited performance and high energy consumption. These implementations are dominated by complex irregular memory access patterns and low arithmetic intensity that pose fundamental challenges to acceleration. To overcome these challenges, we propose and evaluate the use of near-memory acceleration using a reconfigurable fabric with high-bandwidth memory (HBM). We focus on compound stencils that are fundamental kernels in weather prediction models. By using high-level synthesis techniques, we develop NERO, an FPGA+HBM-based accelerator connected through OCAPI (Open Coherent Accelerator Processor Interface) to an IBM POWER9 host system. Our experimental results show that NERO outperforms a 16-core POWER9 system by $5.3 \times$ and $12.7 \times$ when running two different compound stencil kernels. NERO reduces the energy consumption by $12 \times$ and $35 \times$ for the same two kernels over the POWER9 system with an energy efficiency of 1.61 GFLOPS/Watt and 21.01 GFLOPS/Watt. We conclude that employing near-memory acceleration solutions for weather prediction modeling is promising as a means to achieve both high performance and high energy efficiency.

CCS Concepts: • Hardware → Hardware-software codesign; • Computer systems organization → Reconfigurable computing, Heterogeneous (hybrid) systems.

Additional Key Words and Phrases: FPGA, Near-Memory Computing, Weather Modeling, High-Performance Computing, Processing in Memory

ACM Reference Format:
Gagandeep Singh, Dionysios Diamantopoulos, Juan Gómez-Luna, Christoph Hagleitner, Sander Stuijk, Henk Corporaal, and Onur Mutlu. 2021. Accelerating Weather Prediction using Near-Memory Reconfigurable Fabric. 1, 1 (December 2021), 27 pages.

Authors’ addresses: Gagandeep Singh, gagangagandeepsingh@safari.ethz.ch, ETH Zürich, Switzerland; Dionysios Diamantopoulos, did@zurich.ibm.com, IBM Research Europe, Zürich Lab, Switzerland; Juan Gómez-Luna, juan.gomez@safari.ethz.ch, ETH Zürich, Switzerland; Christoph Hagleitner, hle@zurich.ibm.com, IBM Research Europe, Zürich Lab, Switzerland; Sander Stuijk, s.stuijk@tue.nl, Eindhoven University of Technology, The Netherlands; Henk Corporaal, h.corporaal@tue.nl, Eindhoven University of Technology, The Netherlands; Onur Mutlu, omutlu@ethz.ch, ETH Zürich, Switzerland.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2021 Association for Computing Machinery.
Manuscript submitted to ACM
1 Introduction

Accurate weather prediction and climate modeling using detailed weather models is essential to make weather-dependent and climate-related decisions in a timely manner. These models are based on physical laws that describe various components of the atmosphere [137]. The Consortium for Small-Scale Modeling (COSMO) [59] built one such weather model to meet the high-resolution forecasting requirements of weather services. The COSMO model is a non-hydrostatic atmospheric prediction model currently being used by a dozen nations for meteorological purposes and research applications.

The central part of the COSMO model (called dynamical core or dycore) solves the Euler equations on a curvilinear grid and applies explicit discretization in the horizontal dimension (i.e., parameters are dependent on each other at the same time instance [42]) and explicit discretization in the horizontal dimension (i.e., a solution is dependent on the previous system state [42]). The use of different discretizations leads to three computational patterns [161]: 1) horizontal stencils, 2) tridiagonal solvers in the vertical dimension, and 3) point-wise computation. These computational kernels are compound stencil kernels that operate on a three-dimensional grid [79]. Vertical advection (vadvc) and horizontal diffusion (hdiff) are such compound kernels found in the dycore of the COSMO weather prediction model. These kernels are representative of the data access patterns and algorithmic complexity of the entire COSMO model. They are similar to the kernels used in other weather and climate models [97, 125, 177]. Their performance is dominated by memory-bound operations with unique irregular memory access patterns and low arithmetic intensity that often results in <10% sustained floating-point performance on current CPU-based systems [165].

Figure 1 shows the roofline plot [173] for an IBM 16-core POWER9 CPU (IC922).1 After optimizing the vadvc and hdiff kernels for the POWER architecture2 by following the approach in [175], they achieve 29.1 GFLOP/s and 58.5 GFLOP/s, respectively, for 64 threads. Our roofline analysis indicates that these kernels are constrained by the host DRAM bandwidth. Their low arithmetic intensity limits their performance, which is one order of magnitude smaller than the peak performance, and results in a fundamental memory bottleneck that standard CPU-based optimization techniques cannot overcome.

Heterogeneous computing has emerged as an answer to improve the system performance in an energy-efficient way. Heterogeneous computing entails complementing processing elements with different compute capabilities, each to perform the tasks to which it is best suited. In the HPC domain, coupling specialized compute units with general-purpose cores can meet the high-performance computing demands with the ability to realize exascale systems needed to process data-intensive workloads [123]. The graphics processing unit (GPU) is one of the most popular acceleration platforms. GPUs have been used to accelerate workloads like computer graphics and linear algebra [166] because of their many-core architecture. However, GPUs are power-hungry due to high transistor density and, depending on the power constraints, may not always be the ideal platform for implementation. Recently, the use of field-programmable gate array (FPGA) in accelerating machine learning workloads with high energy efficiency has inspired researchers to explore the use of FPGAs instead of GPUs for various high-performance computing applications [48, 61]. FPGAs provide a unique combination of flexibility and performance without the cost, complexity, and risk of developing custom application-specific integrated circuits (ASICs). Modern FPGAs show four key trends.

---

1IBM and POWER9 are registered trademarks or common law marks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies.

2We use single instruction, multiple data (SIMD) [38, 65], and simultaneous multithreading (SMT) [164] techniques to fill the hardware pipelines. We use the same tiling size for both CPU and FPGA-based designs. While compiling these kernels, we use the IBM XL C [9] 16 C/C++ compiler that is optimized for IBM POWER [134] machines with the following flags: qarch=pwr9, qtune=pwr9, O3, q64, qprefetch=aggressive, qsmp=omp, and qsimd=auto.
The advancements in the stacking technology with high-bandwidth memory (HBM) [7, 8, 10, 22, 104] blends DRAM on the same package as an FPGA. This integration allows us to implement our accelerator logic in close proximity to the memory with a lower latency and much higher memory bandwidth than the traditional DDR4-based FPGA boards. Memory-bound applications on FPGAs are limited by the relatively low DDR4 bandwidth (72 GB/s for four independent dual-rank DIMM interfaces [21]). HBM-based FPGAs can overcome this limitation with a peak bandwidth of 410 GB/s [95].

New cache-coherent interconnects, such as Open Coherent Accelerator Processor Interface (OCAPI) [156], Cache Coherent Interconnect for Accelerators (CCIX) [39], and Compute Express Link (CXL) [145], allow tight integration of FPGAs with CPUs at high bidirectional bandwidth (on the order of tens of GB/s). This integration reduces programming effort and enables us to coherently access the host system’s memory through a pointer rather than having multiple copies of the data.

The introduction of UltraRAM (URAM) [17] along with the BlockRAM (BRAM) that offers massive scratchpad-based on-chip memory next to the logic. URAM is more denser than BRAM, but is not as distributed in the FPGA layout as the BRAM.

FPGAs are being manufactured with an advanced technology node of 7-14nm FinFET technology [67] that offers higher performance.

These above trends suggest that modern FPGA architectures with near-memory compute capabilities can alleviate the memory bottleneck of real-world data-intensive applications [148]. However, a study of their advantages for real-world memory-bound applications is still missing. In this work, our goal is to overcome the memory bottleneck of weather

---

**Fig. 1.** Roofline [173] for POWER9 (1-socket) showing vertical advection (vadvc) and horizontal diffusion (hdiff) kernels for single-thread and 64-thread implementations. The plot shows also the rooflines of the FPGAs used in our work with peak DRAM and on-chip BRAM bandwidth.
prediction kernels by exploiting near-memory computation capability on FPGA accelerators with high-bandwidth memory (HBM) [7, 104, 105] that are attached to the host CPU. Figure 1 shows the roofline models of the two FPGA cards (AD9V3 [2] and AD9H7 [1]) used in this work. FPGAs can handle irregular memory access patterns efficiently and offer significantly higher memory bandwidth than the host CPU with their on-chip URAMs (UltraRAM), BRAMs (block RAM), and on-package HBM (high-bandwidth memory for the AD9H7 card). However, taking full advantage of FPGAs for accelerating a workload is not a trivial task. To compensate for the higher clock frequency of the baseline CPUs, our FPGAs must exploit at least one order of magnitude more parallelism in a target workload. This is challenging, as it requires sufficient FPGA programming skills to map the workload and optimize the design for the FPGA microarchitecture.

We aim to answer the following research question: **Can FPGA-based accelerators with HBM mitigate the performance bottleneck of memory-bound compound weather prediction kernels in an energy-efficient way?** As an answer to this question, we present NERO, a near-HBM accelerator for weather prediction. We design and implement NERO on an FPGA with HBM to optimize two kernels (vertical advection and horizontal diffusion), which notably represent the spectrum of computational diversity found in the COSMO weather prediction application. We co-design a hardware-software framework and provide an optimized API to interface efficiently with the rest of the COSMO model, which runs on the CPU. Our FPGA-based solution for **hdiff** and **vadvc** leads to performance improvements of $5.3 \times$ and $12.7 \times$ and energy reductions of $12 \times$ and $35 \times$, respectively, with respect to optimized CPU implementations [175].

The major contributions of NERO are as follows:

- We perform a detailed roofline analysis to show that representative weather prediction kernels are constrained by memory bandwidth on state-of-the-art CPU systems.
- We propose NERO, the first near-HBM FPGA-based accelerator for representative kernels from a real-world weather prediction application.
- We optimize NERO with a data-centric caching scheme with precision-optimized tiling for a heterogeneous memory hierarchy (consisting of URAM, BRAM, and HBM).
- We evaluate the performance and energy consumption of our accelerator and perform a scalability analysis. We show that an FPGA+HBM-based design outperforms a complete 16-core POWER9 system (running 64 threads) by $5.3 \times$ for the vertical advection (**vadvc**) and $12.7 \times$ for the horizontal diffusion (**hdiff**) kernels with energy reductions of $12 \times$ and $35 \times$, respectively. Our design provides an energy efficiency of 1.61 GLOPS/Watt and 21.01 GFLOPS/Watt for **vadvc** and **hdiff** kernels, respectively.

This work extends our previous work [152] as follows. First, we add new results using a state-of-the-art OpenCAPI (OCAPI) interface [156]. OCAPI provides two key opportunities compared to our previous CAPI2 implementation: (1) OCAPI has double the bitwidth (1024-bit) of our previously used CAPI2 interface, and (2) the memory coherency logic has been moved to the host CPU side, which provides more area and allows us to run our design at a higher frequency. Our implementation and evaluation with state-of-the-art OCAPI improves the performance for our two main workloads from weather modeling (**vadvc** and **hdiff**) by 37% and 44%, respectively, compared to a CAPI-based HBM design [152]. All the functions in our design now operate on a 1024-bit per clock rather than 512-bit per clock, utilizing the maximum processing throughput of OCAPI (POWER9 cache line is 1024-bit). Thus, we can build a dataflow accelerator with wide AXI streams of 1024-bit. Second, we develop **HBM_multi+OCAPI**-based versions of our workloads that make use of multiple channels per processing element (PE). This multi-channel implementation allows the PEs
Accelerating Weather Prediction using Near-Memory Reconfigurable Fabric

...to exploit significantly higher bandwidth. As a result, the workloads achieve an average speedup of 1.5x over the single-channel version for a single PE. Third, we provide a discussion section (Section 5) that provides various insights and takeaways while designing HBM-based FPGA accelerators, which we believe would be useful for future FPGA architects and programmers. Fourth, we implement and evaluate the copy stencil [161] (Figure 3), a stencil from the COSMO model to benchmark the peak performance on a platform. Fifth, we use the OC-Accel framework instead of the SNAP framework for OCAPI accelerator development and deployment. Sixth, we provide a comparison of state-of-the-art works in stencil acceleration (Table 3).

2 Background

In this section, we first provide an overview of the vadvc and hdiff compound stencils, which represent a large fraction of the overall computational load of the COSMO weather prediction model. Second, we introduce the OC-Accel (OpenCAPI Acceleration) framework that we use to connect our NERO accelerator to an IBM POWER9 system.

2.1 Representative COSMO Stencils

A stencil operation updates values in a structured multidimensional grid based on the values of a fixed local neighborhood of grid points. In weather and climate modeling, several stencil operations are compounded together that operate on multiple input elements to generate one or more output results. Vertical advection (vadvc) and horizontal diffusion (hdiff) from the COSMO model are two such compound stencil kernels, which represent the typical code patterns found in the dycore of COSMO. Algorithm 1 shows the pseudo-code for vadvc and hdiff kernels. The horizontal diffusion kernel iterates over a 3D grid, performing Laplacian and flux to calculate different grid points, as shown in Figure 2a. A single Laplacian stencil accesses the input grid at five memory offsets, the result of which is used to calculate the flux stencil. hdiff has purely horizontal access patterns and does not have dependencies in the vertical dimension. Thus, it can be fully parallelized in the vertical dimension. Figure 2b shows the memory layout for the horizontal diffusion kernel. We observe that the indirect memory accesses of the input grid domain can severely impact cache efficiency on our current CPU-based systems.

Vertical advection has a higher degree of complexity since it uses the Thomas algorithm [162] to solve a tridiagonal matrix of weather data (called fields, such as air pressure, wind velocity, and temperature) along the vertical axis. vadvc consists of a forward sweep that is followed by a backward sweep along the vertical dimension. vadvc requires access

3https://github.com/OpenCAPI/oc-accel
Singh, et al.

to the weather data, which are stored as array structures while performing forward and sweep computations. Unlike the conventional stencil kernels, vertical advection has dependencies in the vertical direction, which leads to limited available parallelism and irregular memory access patterns. For example, when the input grid is stored by row, accessing data elements in the depth dimension typically results in many cache misses [175].

Such compound kernels are dominated by memory-bound operations with complex memory access patterns and low arithmetic intensity. This poses a fundamental challenge to acceleration. CPU implementations of these kernels [175] suffer from limited data locality and inefficient memory usage, as our roofline analysis in Figure 1 exposes. In Figure 3 we implement a copy stencil from the COSMO weather model to evaluate the performance potential of our HBM-based FPGA platform for the weather prediction application. A copy stencil performs an element-wise copy operation over the complete input grid. It is the simplest stencil found in the COSMO model and, hence, serves as a benchmark to characterize the achievable peak performance on a platform for weather kernels. To implement a copy stencil, we divide the 3D grid data among the processing elements (PEs), where each PE performs an element-wise copy operation. We were able to enable only 24 HBM memory channels because adding more HBM channels leads to timing constraint violations. From the figure, we make two observations. First, as we increase the number of processing elements (PEs), we can exploit data-level parallelism because of dedicated HBM channels serving data to a PE. Second, the maximum achievable performance tends to saturate after 16 PEs. Since we implement our design in a dataflow pipeline manner, all the functions run in parallel, and the overall latency is equal to the maximum latency out of all the functions. After 16 PEs, for copy, we observe that most of the time is spent in the FPGA computation logic rather than the transfer of data from an HBM memory channel.

![Figure 3](image_url)

Fig. 3. Performance of copy stencil on our HBM-based FPGA platform.

### 2.2 OC-Accel Framework

The OpenPOWER Foundation Accelerator Workgroup [14] created the OC-Accel framework, an open-source environment for FPGA programming productivity. OC-Accel provides three key benefits [172]: (i) it enables an improved developer productivity for FPGA acceleration and eases the use of CAPI’s cache-coherence mechanism, (ii) it places FPGA-accelerated compute engines, also known as FPGA actions, closer to relevant data to achieve better performance, and (iii) access to FPGA memory via user-level DMA (Direct Memory Access) semantics. OC-Accel provides a simple API to invoke an accelerated action and provides programming methods to instantiate customized accelerated actions.
Algorithm 1: Pseudo-code for vertical advection and horizontal diffusion kernels used by the COSMO [59] weather prediction model.

1 Function verticalAdvection(float* ccol, float* dcol, float* wcon, float* ustage, float* upos, float* uten, float* utenstage)
2     for c ← 2 to column − 2 do
3         for r ← 2 to row-2 do
4             Function forwardSweep(float* ccol, float* dcol, float* wcon, float* ustage, float* upos, float* uten, float* utenstage)
5                 for d ← 1 to depth do
6                     /* forward sweep calculation */
7                 end
8             end
9         end
10     end
11 end
12
13 Function horizontalDiffusion(float* src, float* dst)
14     for d ← 1 to depth do
15         for c ← 2 to column − 2 do
16             for r ← 2 to row-2 do
17                 /* Laplacian calculation */
18                 lapCR = laplaceCalculate(c, r) /* row-laplacian */
19                 lapCRM = laplaceCalculate(c, r - 1)
20                 lapCP = laplaceCalculate(c, r + 1) /* column-laplacian */
21                 lapCMR = laplaceCalculate(c - 1, r)
22                 lapCRM = laplaceCalculate(c + 1, r) /* column-flux calculation */
23                 fluxC = lapCP - lapCR
24                 fluxCM = lapCRM - lapCR
25                 fluxCm = lapCP - lapCMR
26                 fluxR = lapCRM - lapCR
27                 fluxRm = lapCP - lapCMR
28                 /* output calculation */
29                 dest[d][c][r] = src[d][c][r] - c1 * (fluxCR - fluxCMR) + (fluxCR - fluxCRm)
30             end
31         end
32     end
33 end

on the FPGA side. These accelerated actions can be specified in C/C++ code that is then compiled to the FPGA target using the Xilinx Vivado High-Level Synthesis (HLS) tool [20].
The benefits of employing such cache-coherent interconnect links for attaching FPGAs to CPUs, as opposed to the traditional DMA-like communication protocols (e.g., PCIe), are not only the ultra lower-latency and the higher bandwidth of the communication, but most importantly, the ability of the accelerator to access the entire memory space of the CPU coherently, without consuming excessive CPU cycles. Traditionally, the host processor has a shared memory space across its cores with coherent caches. Attached devices such as FPGAs, GPUs, network, and storage controllers are memory-mapped and use a DMA to transfer data between local and system memory across an interconnect such as PCIe. The attached devices can not see the entire system memory but only a part of it. Communication between the host processor and attached devices requires an inefficient software stack, including user-space software, drivers, and kernel-space modules, in comparison to the communication scheme between CPU cores using shared memory. Especially when DRAM memory bandwidth becomes a constraint, requiring extra memory-to-memory copies to move data from one address space to another is cumbersome and low performance [62, 139]. This is the driving force of the industry to push for coherency and shared memory across CPU cores and attached devices, like FPGAs. This way, the accelerators act as peers to the processor cores. Note that CAPI2 is built on top of PCIe. However, CAPI2 provides the following two advantages. First, a CAPI-attached device, unlike a PCIe device, can perform Direct Memory Access (DMA) to application memory without calls to a device driver or underlying operating system kernel, resulting in a reduction in latency. Avoiding these unnecessary memory calls improves performance significantly compared to the traditional PCIe I/O model [157]. Second, CAPI2 provides cache-coherent access to the CPU memory, allowing the FPGA to directly access the host memory. Such direct cache-coherent access reduces the FPGA developer’s burden and debugging time by overcoming read after write (RAW) and write and read (WAR) dependencies, which is typically the FPGA developer’s responsibility. OCAPI is a new technology built from the ground up. It includes a faster PHY layer (BlueLink 25Gb/s x8 lanes [156]) than its CAPI predecessors, providing double the bitwidth between the host and an accelerator.

3 Design Methodology

3.1 NERO, A Near HBM Weather Prediction Accelerator

The low arithmetic intensity of real-world weather prediction kernels limits the attainable performance on current multi-core systems. This sub-optimal performance is due to the kernels’ complex memory access patterns and their inefficiency in exploiting a rigid cache hierarchy, as quantified in the roofline plot in Figure 1. These kernels cannot fully utilize the available memory bandwidth, which leads to high data movement overheads in terms of latency and energy consumption. We address these inefficiencies by developing an architecture that combines fewer off-chip data accesses with higher throughput for the loaded data. To this end, our accelerator design takes a data-centric approach [24, 25, 43, 44, 71, 85, 86, 100, 118, 120, 121, 147, 150] that exploits near high-bandwidth memory acceleration.

Figure 4a shows a high-level overview of our integrated system. An HBM-based FPGA is connected to a server system based on an IBM POWER9 processor using the Open Coherent Accelerator Processor Interface (OCAPI). The FPGA consists of two HBM stacks[4], each with 16 pseudo-memory channels [3]. A channel is exposed to the FPGA as a 256-bit wide port, and in total, the FPGA has 32 such ports. The HBM IP provides 8 memory controllers (per stack) to handle the data transfer to and from the HBM memory ports. Our design consists of an accelerator functional unit (AFU) that interacts with the host system through the TLx (Transaction Layer) and the DLx (Data Link Layer), which are the OCAPI endpoint on the FPGA. An AFU comprises of multiple processing elements (PEs) that perform compound stencil

---

[4] In this work, we enable only a single stack based on our resource and power consumption analysis for the vadvc kernel.
computation. Figure 5 shows the architecture overview of NERO. As vertical advection is the most complex kernel, we depict our architecture design flow for vertical advection. We use a similar design for the horizontal diffusion kernel.

Fig. 4. (a) Heterogeneous platform with an IBM POWER9 system connected to an HBM-based FPGA board via OCAPI. We also show components of an FPGA: flip-flop (FF), lookup table (LUT), UltraRAM (URAM), and Block RAM (BRAM). (b) Execution timeline with data flow sequence from the host DRAM to the on-board FPGA memory.

Fig. 5. Architecture overview of NERO with data flow sequence from the host DRAM to the on-board FPGA memory via POWER9 cachelines. We depict a single processing element (PE) fetching data from a dedicated HBM port. The number of HBM ports scales linearly with the number of PEs. Heterogeneous partitioning of on-chip memory blocks reduces read and write latencies across the FPGA memory hierarchy.

The weather data, based on the atmospheric model resolution grid, is stored in the DRAM of the host system (1 in Figure 5). We employ the double buffering technique between the CPU and the FPGA to hide the PCIe (Peripheral Component Interconnect Express [114]) transfer latency. By configuring a buffer of 64 cache lines, between the AXI4 interface of OCAPI/TLx-DLx and the AFU, we can reach the theoretical peak bandwidth of OCAPI (i.e., 32 GB/s).
create a specialized memory hierarchy from the heterogeneous FPGA memories (i.e., URAM, BRAM, and HBM). By using a greedy algorithm, we determine the best-suited hierarchy for our kernel. The memory controller (shown in Figure 4a) handles the data placement to the appropriate memory type based on the programmer’s directives.

On the FPGA, following the initial buffering (2), the transferred grid data is mapped onto the HBM memory (3). As the FPGA has limited resources, we propose a 3D window-based grid transfer from the host DRAM to the FPGA, facilitating a smaller, less power-hungry deployment. The window size represents the portion of the grid a processing element (PE in Figure 4a) would process. Most FPGA developers manually optimize for the right window size. However, manual optimization is tedious because of the huge design space, and it requires expert guidance. Further, selecting an inappropriate window size leads to sub-optimal results. Our experiments (in Section 4.3) show that: (1) finding the best window size is critical in terms of the area vs. performance trade-off, and (2) the best window size depends on the datatype precision. Hence, instead of pruning the design space manually, we formulate the search for the best window size as a multi-objective auto-tuning problem taking into account the datatype precision. We make use of OpenTuner [34], which uses machine-learning techniques to guide the design-space exploration [151].

Our design consists of multiple PEs (shown in Figure 4a) that exploit data-level parallelism in COSMO weather prediction kernels. A dedicated HBM memory port is assigned to a specific PE; therefore, we enable as many HBM ports as the number of PEs. This allows us to use the high HBM bandwidth effectively because each PE fetches from an independent port. In our design, we use a switch, which provides the capability to bypass the HBM, when the grid size is small, and map the data directly onto the FPGA’s URAM and BRAM. The HBM port provides 256-bit data, which is a quarter of the size of the OCAPI bitwidth (1024-bit). Therefore, to match the OCAPI bandwidth, we introduce a stream converter logic (4) that converts a 256-bit HBM stream to a 1024-bit stream (OCAPI compatible) or vice versa. From HBM, a PE reads a single stream of data that consists of all the fields that are needed for a specific COSMO kernel computation. The PEs use a fields stream splitter logic (5) that splits a single HBM stream to multiple streams (1024-bit each), one for each field.

To optimize a PE, we apply various optimization strategies. First, we exploit the inherent parallelism in a given algorithm through hardware pipelining. Second, we partition on-chip memory to avoid the stalling of our pipelined design, since the on-chip BRAM/URAM has only two read/write ports. Third, all the tasks execute in a dataflow manner that enables task-level parallelism.avadc is more computationally complex than hdiff because it involves forward and backward sweeps with dependencies in the z-dimension. While hdiff performs only Laplacian and flux calculations with dependencies in the x- and y-dimensions. Therefore, we demonstrate our design flow by means of the vadvc kernel (Figure 5). Note that we show only a single port-based PE operation. However, for multiple PEs, we enable multiple HBM ports.

We make use of memory reshaping techniques to configure our memory space with multiple parallel BRAMs or URAMs [58]. We form an intermediate memory hierarchy by decomposing (or slicing) 3D window data into a 2D grid. This allows us to bridge the latency gap between the HBM memory and our accelerator. Moreover, it allows us to exploit the available FPGA resources efficiently. Unlike traditionally-fixed CPU memory hierarchies, which perform poorly with irregular access patterns and suffer from cache pollution effects and cache miss latency, application-specific memory hierarchies are shown to improve energy and latency by tailoring the cache levels and cache sizes to an application’s memory access patterns [163].

5Fields represent atmospheric components like wind, pressure, velocity, etc. that are required for weather calculation.
The main computation pipeline consists of a forward and a backward sweep logic. The forward sweep results are stored in an intermediate buffer to allow for backward sweep calculation. Upon completion of the backward sweep, results are placed in an output buffer that is followed by a degridding logic. As there is only a single output stream (both in vadvc and hdiff), we do not need extra logic to merge the streams. The 1024-bit wide stream goes through an HBM stream converter logic that converts the stream bitwidth to HBM port size (256-bit).

Figure 4b shows the execution timeline from our host system to the FPGA board for a single PE. The host offloads the processing to an FPGA and transfers the required data via DMA (direct memory access) over the OCAPI interface. The OC-Accel framework allows for parallel execution of the host and our FPGA PEs while exchanging control signals over the AXI lite interface [4]. On task completion, the AFU notifies the host system via the AXI lite interface and transfers back the results via DMA.

3.2 NERO Application Framework

Figure 6 shows the NERO application framework to support our architecture. Our previous work [152, 153] describes the corresponding application framework using SNAP-CAPI2. A software-defined COSMO API handles offloading jobs to NERO with an interrupt-based queuing mechanism. This allows for minimal CPU usage (and, hence, power usage) during FPGA operation. NERO employs an array of processing elements to compute COSMO kernels, such as vertical advection or horizontal diffusion. Additionally, we pipeline our PEs to exploit the available spatial parallelism. By accessing the host memory through the OCAPI cache-coherent link, NERO acts as a peer to the CPU. This is enabled through the TLx (Transaction Layer) and the DLx (Data Link Layer). OC-Accel allows for seamless integration of the COSMO API with our OCAPI-based accelerator. The job manager dispatches jobs to streams, which are managed in the stream scheduler. The execution of a job is done by streams that determine which data is to be read from the host memory and sent to the PE array through DMA transfers. The pool of heterogeneous on-chip memory is used to store the input data from the main memory and the intermediate data generated by each PE.

4 Results

4.1 Experimental Setup

We evaluate our accelerator designs for vadvc, and hdiff in terms of performance, energy consumption, and FPGA resource utilization on two different FPGAs, and two different external data communication interfaces between the CPU and the FPGA board. We implement our accelerator designs for vadvc, and hdiff on both 1) an Alpha-Data ADM-PCIE-9H7 card featuring the Xilinx Virtex Ultrascale+ XCVU37P-FSVH2892-2-e with 8GiB HBM2 and 2) an Alpha-Data ADM-PCIE-9V3 card featuring the Xilinx Virtex Ultrascale+ XCVU3P-FFVC1517-2-i with 8GiB DDR4, connected to an IBM POWER9 host system. For the external data communication interface, we use both CAPI2 and the state-of-the-art OCAPI (OpenCAPI) interface. We compare these implementations to execution on a POWER9 CPU with 16 cores (using all 64 hardware threads). Table 1 provides our system parameters. We co-design our hardware and software interface around the OC-Accel framework while using the HLS design flow. Our development machine is an Intel® Xeon® 7.9.2009 distribution with GNU Compiler Collection (GCC) version 4.8.5. We use Xilinx Vivado 2019.2 suite to develop our accelerator designs.
We run our experiments using a 256 x 256 x 64-point domain similar to the grid domain used by the COSMO weather prediction model. We employ an auto-tuning technique to determine a Pareto-optimal solution (in terms of performance and resource utilization) for our 3D window dimensions. The auto-tuning with OpenTuner exhaustively searches for every tile size in the x- and y-dimensions for vadvc. For hdi ff, we consider sizes in all three dimensions. We define our auto-tuning as a multi-objective optimization with the goal of maximizing performance with minimal resource utilization. Section 3 provides further details on our design. We evaluate frequency values between 50-400 MHz, with an increment of 50 MHz, utilizing the complete spectrum of compatible frequency configurations supported by the
OC-Accel framework [13]. Figure 7 shows hand-tuned and auto-tuned performance and FPGA resource utilization results for vadvc, as a function of the chosen tile size. From the figure, we draw two observations.

![Graph showing performance and resource utilization for vadvc](image)

Fig. 7. Performance and FPGA resource utilization of single vadvc PE, as a function of tile-size, using hand-tuning and auto-tuning for (a) single-precision (32-bit) and (b) half-precision (16-bit). We highlight the Pareto-optimal solution that we use for our vadvc accelerator (with a red circle). Note that the Pareto-optimal solution changes with precision.

First, by using the auto-tuning approach and our careful FPGA microarchitecture design, we can get Pareto-optimal results with a tile size of $64 \times 2 \times 64$ for single-precision vadvc, which gives us a peak performance of 8.49 GFLOP/s. For half-precision, we use a tile size of $32 \times 16 \times 64$ to achieve a peak performance of 16.5 GFLOP/s. We employ a similar strategy for hdiff to attain a single-precision performance of 30.3 GFLOP/s with a tile size of $16 \times 64 \times 8$ and a half-precision performance of 77.8 GFLOP/s with a tile size of $64 \times 8 \times 64$.

Second, in FPGA acceleration, designers usually rely on expert judgement to find the appropriate tile-size and often adapt the design to use homogeneous tile sizes. However, as shown in Figure 7, such hand-tuned implementations lead to sub-optimal results in terms of either resource utilization or performance.

We conclude that the Pareto-optimal tile size depends on the data precision used: a good tile-size for single-precision might lead to poor results when used with half-precision.

### 4.3 Performance Analysis

Figure 8 shows single-precision performance results for the (a) vertical advection (vadvc) and (b) horizontal diffusion kernels (hdiff). For both kernels, we implement our design on an HBM- and a DDR4-based FPGA board. For the DDR4-based design, we use CAPI2 (DDR4+CAPI2 in Figure 8. For the HBM-based design, we use CAPI2 (HBM+CAPI2) and OCAPI. We evaluate two versions of the HBM-based design with OCAPI: (1) one with a single channel per PE (HBM+OCAPI), and (2) one with multiple channels (i.e., 4 HBM pseudo channels) per PE (HBM_multi+OCAPI). To compare the performance of these four versions, we scale the number of PEs and analyze the change in execution time. We also tested different domain sizes, varying from $64 \times 64 \times 64$-point to $1024 \times 1024 \times 64$-point and observe that the runtime scales linearly and the overall performance (GLOP/s) remain constant. This shows the scalability of our accelerator design.

We draw five observations from the figure.
First, the maximum number of PEs that we can fit on the FPGA boards varies for different versions of our design. For the DDR4-based design, we can accommodate only 4 PEs/8 PEs for vadvc/hdiff on the 9V3 board. For the HBM-based design, we can fit 14 PEs/16 PEs for vadvc/hdiff for both HBM+CAPI2 and HBM+OCAPI versions before exhausting the on-board resources. The HBM_multi+OCAPI version can only fit 3 PEs (i.e., 12 HBM channels) for both vadvc and hdiff because adding more HBM channels leads to timing constraint violations.

Second, the full-blown HBM+OCAPI versions (i.e., with the maximum number of PEs) of vadvc and hdiff outperform the 64-thread IBM POWER9 CPU version by $5.3 \times$ and $12.7 \times$, respectively. We achieve 37% and 44% higher performance for vadvc and hdiff, respectively, with HBM+OCAPI than HBM+CAPI2 due to the following two reasons: (1) OCAPI provides double the bitwidth (1024-bit) of the CAPI2 interface (512-bit), which provides a higher bandwidth to the host CPU, i.e., 22.1/22.0 GB/s R/W versus 13.9/14.0 GB/s; and (2) with OCAPI, memory coherency logic is moved onto the IBM POWER CPU, which provides more FPGA area and allows us to run our accelerator logic at a higher clock frequency (250MHz for OCAPI versus 200MHz for CAPI2). We observe that when implementing our accelerator designs with targets above those frequencies, the respective timing report results in the worst negative slack (WNS) higher than 200ps, which OC-Accel developers regard as dangerous for system stability. At lower frequencies, we achieved lower performance, regardless of the number of PEs. Our single-precision HBM+OCAPI-based FPGA implementations provide 157.1 GFLOP/s and 608.4 GFLOP/s for vadvc and hdiff, respectively. For half-precision, if we use the same amount of PEs as in single precision, our accelerator reaches a performance of 329.9 GFLOP/s for vadvc ($2.1 \times$ the single-precision performance) and 1.5 TFLOP/s for hdiff ($2.5 \times$ the single-precision performance).

Third, for a single PE, DDR4+CAPI2 is faster than HBM+CAPI2 for both vadvc and hdiff. This higher performance is because the HBM-based design uses one HBM channel per PE, and the bus width of the DDR4 channel (512 bits) is larger than that of an HBM channel (256 bits). Therefore, the HBM channel has a lower transfer rate of 0.8-2.1 GT/s (Gigatransfers per second) than a DDR4 channel (2.1-4.3 GT/s), resulting in a theoretical bandwidth of 12.8 GB/s and 25.6 GB/s per channel, respectively. One way to match the DDR4 bus width is to have a single PE fetch data from multiple HBM channels in parallel. In Figure 8, our multi-channel setting (HBM_multi+OCAPI) uses 4 HBM pseudo channels per PE to match the bitwidth of the OCAPI interface. We observe that by fetching more data from multiple
channels, compared to the single-channel-single PE design (HBM+OCAPI). HBM_multi+OCAPI achieves 1.2× and 1.8× performance improvement for vadvc and hdiff, respectively.

Fourth, as we increase the number of PEs, we divide the workload evenly across PEs. As a result, we observe linear scaling in the performance of HBM-based designs, where each PE reads and writes through a dedicated HBM channel. For multi-channel designs, we observe that the best-performing multi-channel-single PE design (i.e., using 3 PEs with 12 HBM channels for both workloads) has 4.7× and 3.1× lower performance than the best-performing single-channel-single PE design (i.e., 14 PEs for vadvc and 16 PEs for hdiff, respectively). This observation shows that there is a tradeoff between (1) enabling more HBM pseudo channels to provide each PE with more bandwidth, and (2) implementing more PEs in the available area. For both vadvc and hdiff, data transfer and computation take a comparable amount of time. Therefore, we are able to achieve a linear execution time reduction with the number of PEs.

Fifth, the performance of the DDR-based designs scales non-linearly for vadvc and hdiff with the number of PEs, as all PEs access memory through the same channel. Multiple PEs compete for a single memory channel, which causes frequent memory stalls due to contention in the memory channel.

4.4 Energy Efficiency Analysis

We compare the energy consumption of our accelerator to a 16-core POWER9 host system. We use the AMESTER tool to measure the active power consumption. We measure 99.2 Watts for vadvc and 97.9 Watts for hdiff by monitoring built power sensors in the POWER9 system. For vadvc and hdiff on the HBM- and DDR4-based designs, Figure 9 and Figure 10 shows the active power consumption and the energy efficiency (GFLOPS per Watt), respectively.

We make five observations from Figure 9 and Figure 10.

![Active Power Consumption](image)

**Fig. 9.** Active Power Consumption for (a) vadvc and (b) hdiff on HBM- and DDR4-based FPGA boards. For HBM-based design, we implement our accelerator with both the CAPI2 interface and the state-of-the-art OpenCAPI (OCAPI) interface (with both single channel and multiple channels per PE).

First, the full-blown HBM+OCAPI designs (i.e., 14 PEs for vadvc and 16 PEs for hdiff) achieve energy efficiency values of 1.61 GFLOPS/Watt and 21.01 GFLOPS/Watt for vadvc and hdiff, respectively. These represent improvements of 12× and 35× compared to the IBM POWER9 system for vadvc and hdiff, respectively.

7 https://github.com/open-power/amester
8 Active power denotes the difference between the total power of a complete socket (including CPU, memory, fans, I/O, etc.) when an application is running compared to when it is idle.
Second, the DDR4-CAPI2 designs for vadvc and hdiff are slightly more energy-efficient (1.1× to 1.5×) than the HBM-CAPI2 designs when the number of PEs is small. This observation is in line with our discussion about performance with small PE counts in Section 4.3. However, as we increase the number of PEs, the HBM-CAPI2 designs provide higher energy efficiency since they make use of multiple HBM channels.

Third, the energy efficiency of the HBM-based designs (HBM+CAPI2, HBM+OCAPI) for hdiff increases with the number of PEs until a saturation point (8 PEs). This trend is because every additional HBM channel increases power consumption by ~1 Watt (for the HBM AXI3 interface operating at 250MHz with a logic toggle rate of ~12.5%).

Fourth, HBM+OCAPI, HBM+CAPI2, and DDR4+CAPI2 versions of vadvc achieve their highest energy efficiency at a number of PEs that is smaller than the maximum possible. There is a large amount of control flow in vadvc, which leads to large resource utilization. As a result, as shown in Figure 9, increasing the PE count increases power consumption dramatically, causing lower energy efficiency.

Fifth, the multi-channel-single PE designs (HBM_multi+OCAPI) are more energy-efficient than the single-channel-single PE designs (HBM+OCAPI) for the same number of PEs. However, HBM+OCAPI designs achieve higher energy efficiency for higher numbers of PEs, which are not affordable for HBM_multi+OCAPI designs.

### 4.5 FPGA Resource Utilization

Table 2 shows the resource utilization of vadvc and hdiff on the AD9H7 board. We draw two observations. First, there is a high BRAM consumption compared to other FPGA resources. This is because we implement input, field, and output signals as hls::streams. In high-level synthesis, by default, streams are implemented as FIFOs that make use of BRAM. Second, vadvc has a much larger resource consumption than hdiff because vadvc has higher computational complexity and requires a larger number of fields to perform the compound stencil calculation. Note that for hdiff,
we can accommodate more PEs, but in this work, we make use of only a single HBM stack. Therefore, we use 16 PEs because a single HBM stack offers up to 16 memory channels.

Table 2. FPGA resource utilization in our highest-performing HBM-based designs for vadvc and hdiff.

| Algorithm | BRAM | DSP | FF | LUT | URAM |
|-----------|------|-----|----|-----|------|
| vadvc     | 94%  | 39% | 37%| 55% | 53%  |
| hdiff     | 96%  | 4%  | 10%| 15% | 8%   |

5 Discussion and Key Takeaways

A wide range of application domains have emerged with the ubiquity of computing platforms in every aspect of our daily lives. These modern workloads (e.g., machine learning, graph processing, and bioinformatics) demand high compute capabilities within strict power constraints [71]. However, today’s computing systems are getting constrained by current technological capabilities, making them incapable of delivering the required performance. This paper presents our recent efforts to leverage near-memory computing capable FPGA-based accelerators to accelerate two major kernels from the weather prediction application in an energy-efficient way. We summarize the most important insights and takeaways as follows.

First, our evaluation shows that High-Bandwidth Memory-based near-memory FPGA accelerator designs can improve performance by $5.3 \times - 12.7 \times$ and energy efficiency by $12 \times - 35 \times$ over a single-socket high-end 16-core IBM POWER9 CPU.

Second, our HBM-based FPGA accelerator designs employ a dedicated HBM channel per PE. Such a design avoids memory access congestion, which is typical in DDR4-based FPGA designs, and ensures memory bandwidth scaling with the number of PEs. As a result, in most of the data-parallel applications, performance scales linearly with the number of PEs. Therefore, HBM provides an attractive solution for scale-out computation.

Third, the data needs to be adequately mapped to each HBM channel’s address space. A data mapping scheme should map data in such a way that the data required by the processing unit is readily available in the vicinity (data and code co-location). An inefficient data mapping mechanism can severely hamper the benefits of processing close to memory.

Fourth, we make use of OCAPI in a coarse-grained way, since we offload the entire application to the FPGA. In this case, OCAPI ensures that the FPGA accelerators access the entire CPU memory with the minimum number of memory copies between the host and the FPGA, e.g., avoiding the intermediate buffer copies that a traditional PCIe-based DMA invokes [52]. However, depending on the application, the CAPI protocol can be employed in finer-grained algorithm-hardware co-design, like the ExtraV [106], where the authors aggressively utilize the fine-grained communication capability of OCAPI to boost graph analytics performance.

Fifth, the maximum performance of our HBM-based design is reached using the maximum PE count that we can fit in the reconfigurable fabric, with each PE having a dedicated HBM channel. However, adding more PEs could lead to timing constraint violations for HBM-based designs. As shown with our multi-channel setting (Section 4.3), where we can fit only 3 PEs for both vadvc and hdiff, enabling more HBM channels leads to timing constraint violations. HBM-based FPGAs consist of multiple super-logic regions (SLRs) [18], where an SLR represents a single FPGA die. All HBM channels are connected only to SLR0, while other SLRs have indirect connections to the HBM channels. Therefore, if a PE is implemented in a non-SLR0 region for a large design, it might make timing closure difficult. A possible way to alleviate timing issues is by running the AFU at a lower frequency, which eases the place and route.
Sixth, the energy efficiency of our HBM-based designs tends to saturate (or even reduces) as we increase the number of PEs beyond some point. The highest energy efficiency is achieved with a PE count that is smaller than the highest-performing PE count. The major reason for a decrease in the energy efficiency is the increase in power consumption with every additional HBM channel.

Seventh, the emerging cache-coherent interconnects standards like CXL [145], CCIX [39], and OCAPI [156] could be vital in improving the performance and energy efficiency of big data workloads running on FPGA-based devices because they avoid having multiple data copies. However, a very small number of works, such as [106], leverage the coherency aspect of these interconnects. More quantitative exploration is required to analyze the advantages and disadvantages of using these interconnects.

Eighth, we are witnessing an enormous amount of data being generated across multiple application domains [123, 151] like weather prediction modeling, radio astronomy, bioinformatics, material science, chemistry, health sciences, etc. The processing of the sheer amount of generated data is one of the biggest challenges to overcome. In this paper, we demonstrate the capabilities of near-memory reconfigurable accelerators in the domain of weather prediction, however, there are many other high-performance computing applications where such near-memory architectures can alleviate the data movement bottleneck.

6 Related Work

To our knowledge, this is the first work to evaluate the benefits of using FPGAs equipped with high-bandwidth memory (HBM) to accelerate real-world weather modeling stencils. We exploit the near-memory capabilities of such FPGAs to accelerate important weather prediction kernels. Exploiting the high-bandwidth memory in FPGAs, we answer the following questions with our work. First, do real-world weather prediction applications benefit from HBM-based FPGAs? Second, how can we scale the processing in terms of not only run-time but also energy efficiency? Third, what does the system look like regarding computation and data movement with an HBM-enabled FPGA when integrating accelerators for real-world weather prediction workloads?

Modern workloads exhibit limited locality and operate on large amounts of data, which causes frequent data movement between the memory subsystem and the processing units [43, 44, 71, 118–121]. This frequent data movement has a severe impact on overall system performance and energy efficiency. For example, in the domain of climate and weather modeling, there is a data avalanche due to large atmospheric simulations [137]. Major efforts are currently underway towards refining the resolution grid of climate models that would generate zettabytes of data [137]. These high-resolution simulations are useful to predict and address events like severe storms. However, the sheer amount of generated data is one of the biggest challenges to overcome. We find another relevant example in radio astronomy. The first phase of the Square Kilometre Array (SKA) aims to process over 100 terabytes of raw data samples per second, yielding of the order of 300 petabytes of SKA data produced annually [91, 149]. Recent biological disciplines such as genomics have also emerged as one of the most data-intensive workloads across all different sciences wherein just a single human genome sequence produces hundreds of gigabytes of raw data. With the rapid advancement in sequencing technology, the data volume in genomics is projected to surpass the data volume in all other application domains [124].

A way to alleviate this data movement bottleneck [43, 44, 71, 80, 118–121, 121, 144, 150] is near-memory computing (NMC), which consists of placing processing units closer to memory. NMC is enabled by new memory technologies, such as 3D-stacked memories [7, 43, 99, 102, 104, 105, 108, 118, 121, 130], and also by cache-coherent interconnects [39, 145, 156], which allow close integration of processing units and memory units. Depending on the applications and systems of interest (e.g., [24–26, 33, 36, 37, 40, 43, 44, 46, 47, 50, 63, 64, 68, 69, 73, 75, 76, 81, 82, 86, 93, 96, 98, 107, 109, 111],
FPGA accelerators are promising to enhance overall system performance with low power consumption. Past works [28–32, 49, 57, 74, 87, 90, 92, 94, 106] show that FPGAs can be employed effectively for a wide range of applications. FPGAs provide a unique combination of flexibility and performance without the cost, complexity, and risk of developing custom application-specific integrated circuits (ASICs). The researchers at CERN, for example, are using FPGAs to accelerate physics workload in CERN’s exploration of dark matter [61]. Microsoft’s Project Catapult [48] is another example of how FPGAs can be used in the data center infrastructure. Driven by Catapult’s promising research results, Microsoft further deployed the architecture on the Azure cloud marketplace [116]. Such integration for certain workloads can even offer more energy efficiency than CPU or GPU-based systems. The recent addition of HBM to FPGAs presents an opportunity to exploit high memory bandwidth with the low-power FPGA fabric. The potential of high-bandwidth memory [7, 104] has been explored in many-core processors [72, 131] and GPUs [72, 178]. Recent benchmarking works [95, 171] show the potential of HBM for FPGAs.

NERO is the first work to accelerate a real-world HPC weather prediction application using the FPGA+HBM fabric. Compared to a previous work [153] that optimizes only the horizontal diffusion kernel on an FPGA with DDR4 memory, our analysis reveals that the vertical advection kernel has a much lower compute intensity with little to no regularity. Therefore, our work accelerates both kernels that together represent the algorithmic diversity of the entire COSMO weather prediction model. Our current work differs from [152] in the following aspects. First, we design and evaluate both horizontal diffusion and vertical advection stencils. Vertical advection is the most complex stencil in the entire COSMO application. Second, we integrate and implement our accelerator design with an HBM-based FPGA. The bus width of the DDR4 channel (512 bits) is larger than that of an HBM channel (256 bits). Therefore, the HBM channel has a lower transfer rate of 0.8-2.1 GT/s (Gigatransfers per second) than a DDR4 channel (2.1-4.3 GT/s), resulting in a theoretical bandwidth of 12.8 GB/s and 25.6 GB/s per channel, respectively. However, HBM exposes 32 memory channels that provide 4x more bandwidth (410 GB/s for HBM [95]) compared to traditional DDR4 bandwidth (72 GB/s for four independent dual-rank DIMM interfaces [21]). Therefore, the use of HBM imposes an architectural shift. We evaluate and demonstrate the use of HBM for scaling an accelerator design with different channels provided by HBM. Third, we use an auto-tuning framework to find the right window size (Figure 6) that demonstrates the importance of finding the right window size. Fourth, we provide new results using a state-of-the-art OpenCAPI (OCAPI) interface with the OC-Accel framework. OCAPI provides two key opportunities compared to our previous CAPI2 implementation: (1) OCAPI has double the bitwidth of our previously used CAPI2 interface, (2) a major component of the memory coherency logic is moved to the host CPU side, which provides more FPGA area and enables designs with higher frequency. Due to the above optimizations, we improve the performance for horizontal diffusion by 1.2x on a DDR4-based board and 4.7x on an HBM-based board compared to our previous work NARMADA [153].

Enabling higher performance for stencil computations has been a subject of optimizations across the whole computing stack [35, 51, 53–55, 66, 77, 79, 83, 115, 135, 155, 160, 168, 169, 179]. Stencil computation is essential for numerical simulations of finite difference methods (FDM) [127] and is applied in iterative solvers of linear equation systems. We use stencil computation in a wide range of applications, including computational fluid dynamics [88], image processing [84], weather prediction modeling [59], etc.

Unlike stencils found in the literature [51, 55, 56, 135, 154, 168, 169], real-world compound stencils consist of a collection of stencils that perform a sequence of element-wise computations with complex interdependencies. Such
compound kernels have complex memory access patterns and low arithmetic intensity because they have limited operations per loaded value. Our work is the first work to accelerate both horizontal diffusion and vertical advection stencils, which are representative of data access patterns and the algorithmic complexity found in the entire COSMO weather model.

Table 3 lists recent works (including NERO) that use FPGA to accelerate stencil-based application. We also mention works that accelerate elementary stencils (7-point, 25-point Jacobi, Hotspot, and Diffusion). We make the following three observations. First, the elementary stencils can achieve much higher performance on comparable FPGA devices than complex weather stencils (such as hdiff) even without using HBM. This high performance is because elementary stencils have a higher arithmetic intensity than weather stencils. Due to their data-parallel nature, these elementary stencils can further take advantage of the increased bandwidth provided by HBM in an energy-efficient way. Second, weather stencils can reach only 2%-17% of the peak theoretical performance of an FPGA board. This low peak performance is because weather stencils have several elementary stencils cascaded together with data interdependencies that lead to complex memory access patterns. Third, compared to NARMADA [153], which uses a DDR4-based design, our HBM-based design achieves 4.7× performance improvement by exploiting the high bandwidth provided by the HBM.

Table 3. Overview of the state-of-art stencil implementations on FPGAs. For each work, we mention the technology node (Tech. node), DRAM memory technology (Mem. Tech.), theoretical peak floating-point performance (Peak Perf. (TFLOPS)), available peak memory bandwidth (Peak B/W (GB/s)), frequency of the accelerator logic (Freq. (MHz)), overall logic utilization (Logic Util.), overall memory utilization (Mem. Util.), achieved performance (Perf. (GOp/s)), and the percentage of achieved peak roofline performance (Ach. Roof.).

| Stencil | Work | Year | Device | Tech. node | Mem. Tech | Peak Perf. (TFLOPS) | Peak B/W (GB/s) | Freq. (MHz) | Logic Util. | Mem. Util. | Perf. (GOp/s) | Ach. Roof. |
|---------|------|------|--------|------------|-----------|---------------------|----------------|-------------|-------------|------------|--------------|------------|
| Diffusion 3D | [168] | 2019 | Arria 10 | TSMC 20nm | DDR3 | 1.4 | 34 | 276 | 52% | 87% | 626.0 | 64.5% |
| 25-point 3D | [165] | 2019 | Arria 10 | TSMC 20nm | DDR3 | 0.97 | 0.97 | 25.6 | 180 | 23.5% | 99% | 226.4 | 23.7% |
| Intel 14nm FinFet | [41] | 2019 | Stratix 10 | Intel 14nm FinFet | DDR4 | 9.2 | 7.8 | 190 | 160 | 49% | 99% | 327.7 | 51.4% |
| hdiff | [153] | 2019 | XC/VUP | TSMC 16FF | DDR4 | 0.97 | 0.97 | 25.6 | 190 | 49% | 99% | 568.2 | 6.2% |
| hdiff | [56] | 2021 | XC/VUP | TSMC 16FF | HBM | 9.2 | 7.8 | 252 | 125 | 50% | 99% | 488.4 | 6.9% |

† When simulated using an infinite memory bandwidth. ‡ Note that we use only a single HBM stack due to resource limitations.

Szustak et al. accelerate the MPDATA advection scheme on multi-core CPU [159] and computational fluid dynamics kernels on FPGA [133]. Singh et al. [154] explore the applicability of different number formats and exhaustively search for the appropriate bit-width for memory-bound stencil kernels to improve performance and energy efficiency with minimal loss in the accuracy. Bianco et al. [41] optimize the COSMO weather prediction model for GPUs. Thaler et al. [161], in a collaboration work with the Swiss National Supercomputing Centre (CSCS) and the Federal Institute of Meteorology and Climatology (MeteoSwiss), discuss the importance of horizontal diffusion and vertical advection kernels in the entire COSMO model. These kernels together represent the algorithmic diversity of the entire COSMO weather prediction model [41, 79, 161]. They port COSMO to a many-core system. Compared to their Intel KNL [11] (or NVIDIA P100 [12]) implementation, we observe that our FPGA-based vadv and hdiff design provides 1.5× (or 1.4×) and 3.2× (or 2.1×) performance improvements, respectively. Several works [56, 103, 110, 170] propose frameworks for generating optimized stencil code for FPGA-based platforms. Wahib et al. [167] develop an analytical performance model for choosing an optimal GPU-based execution strategy for various scientific applications, including COSMO. Gysi et al. [79] provide guidelines for optimizing stencil kernels for CPU–GPU systems.
7 Conclusion

We introduce NERO, the first design and implementation on a reconfigurable fabric with high-bandwidth memory (HBM) to accelerate representative weather prediction kernels, i.e., vertical advection (vadv) and horizontal diffusion (hdiff), from a real-world weather prediction application. These kernels are compound stencils that are found in various weather prediction applications, including the COSMO model. We show that compound kernels do not perform well on conventional architectures due to their complex data access patterns and low data reusability, which make them memory-bounded. Therefore, they greatly benefit from our near-memory computing solution that takes advantage of the high data transfer bandwidth of HBM. We use a heterogeneous system comprising of IBM POWER9 CPU with field-programmable gate array (FPGA) as our target platform. We create a heterogeneous domain-specific memory hierarchy using on-chip URAMs and BRAMs, and on-package HBM on an FPGA. Unlike conventional fixed CPU memory hierarchies, which perform poorly with irregular access patterns and suffer from cache pollution effects, application-specific memory hierarchies are shown to improve both energy and latency by tailoring the cache levels and cache sizes to an application’s memory access patterns.

NERO’s implementations of vadv and hdiff outperform the optimized software implementations on a 16-core POWER9 with 4-way multithreading by 5.3× and 12.7×, with 12× and 35× less energy consumption, respectively. We conclude that hardware acceleration on an FPGA+HBM fabric is a promising solution for compound stencils present in weather prediction applications. We hope that our reconfigurable near-memory accelerator inspires developers of different high-performance computing applications that suffer from the memory bottleneck.

Acknowledgments

This work was performed in the framework of the Horizon 2020 program for the project “Near-Memory Computing (NeMeCo)”. It is funded by the European Commission under Marie Sklodowska-Curie Innovative Training Networks European Industrial Doctorate (Project ID: 676240). Special thanks to Florian Auernhammer and Raphael Polig for providing support with the IBM systems. We appreciate valuable discussions with Kaan Kara and Ronald Luijten. We would also like to thank Bruno Mesnet and Alexandre Castellane from IBM France for help with the SNAP and OC-Accel framework. This work was partially supported by the H2020 research and innovation programme under grant agreement No 732631, project OPRECOMP. We also thank Google, Huawei, Intel, Microsoft, SRC, and VMware for their funding support to the SAFARI Research Group.

References

[1] ADM-PCIe-9H7-High-Speed Communications Hub, https://www.alpha-data.com/dcp/products.php?product=adm-pcie-9h7.
[2] ADM-PCIe-9V3-High-Performance Network Accelerator, https://www.alpha-data.com/dcp/products.php?product=adm-pcie-9v3.
[3] AXI High Bandwidth Memory Controller v1.0, https://www.xilinx.com/support/documentation/ip_documentation/hbm/v1_0/pg276-axi-hbm.pdf.
[4] AXI Reference Guide, https://www.xilinx.com/support/documentation/ip_documentation/ug761_axi_reference_guide.pdf.
[5] CentOS-7 (2009) Release Notes, https://wiki.centos.org/Manuals/ReleaseNotes/CentOS7.2009.
[6] GCC, the GNU Compiler Collection, https://gcc.gnu.org/.
[7] High Bandwidth Memory (HBM) DRAM (JESD235), https://www.jedec.org/document_search?search_api_views_fulltext=jesd235.
[8] IBM XL C/C++ for Linux, https://www.ibm.com/products/xl-cpp-linux-compiler-power.
[9] Intel Stratix 10 MX FPGAs, https://www.intel.com/content/www/us/en/products/programmable/sip/stratix-10-mx.html.
[10] Intel® Xeon Phi™ Processor 7230 (16GB, 1.30 GHz, 64 core), https://www.intel.com/content/www/us/en/products/processor/7230-16gb-1-30-ghz-64-core-specifications.html.
[11] NVIDIA® TESLA® P100 GPU ACCELERATOR, https://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-PCIe-datasheet.pdf.
[12] OC-Accel, https://openapi.github.io/oc-accel-doc/.
Accelerating Weather Prediction using Near-Memory Reconfigurable Fabric

A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis. In MICRO 2020.

A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou, and D. Burger. A Cloud-Scale Acceleration Architecture. In MICRO 2016.

Li-Wen Chang, Juan Gómez-Luna, Izrat El Hajj, Sitao Huang, Deming Chen, and Wen-mei Hwu. Collaborative Computing for Heterogeneous Integrated Systems. In ICFP 2017.

Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongquan Lu, Yu Wang, and Yuan Xie. PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory. In ISCA 2016.

Yure Chi, Jason Cong, Peng Wei, and Peipei Zhou. SODA: Stencil with Optimized Datallow Architecture. In ICCAD 2018.

Young-kyu Choi, Jason Cong, Zhenman Fang, Yuchen Hao, Glenn Reinman, and Peng Wei. A Quantitative Analysis on Microarchitectures of Modern CPU-FPGA Platforms. In DAC 2016.

Matthias Christen, Olaf Schenk, and Helmar Burkhart. PATUS: A Code Generation and Autotuning Framework for Parallel Iterative Stencil Computations on Modern Microarchitectures. In IPDPS 2011.

Kaushik Datta, Shoaib Kamil, Samuel Williams, Leonid Oliker, John Shalf, and Katherine Yelick. Optimization and Performance Modeling of Stencil Computations on Modern Microprocessors. In SIAM review 2009.

Johannes de Fine Licht, Michaela Blott, and Torsten Hoeftler. Designing scalable FPGA architectures using high-level synthesis. In PPoPP 2018.

Johannes de Fine Licht, Andreas Kuster, Tiziano De Matteis, Tal Ben-Nun, Dominic Hofer, and Torsten Hoeftler. StencilFlow: Mapping large stencil programs to distributed spatial computing systems. In FOG 2021.

Dionysios Diamantopoulos, Heiner Giefer, and Christoph Hagl點tner. ecTALK: Energy Efficient Coherent Transprecision Accelerators – The Bidirectional Long Short-Term Memory Neural Network Accelerator. In COOL CHIPS 2018.

Dionysios Diamantopoulos and Christoph Hagl點tner. A System-Level Transprecision FPGA Accelerator for BLSTM Using On-chip Memory Redshaping. In FPT 2018.

G Doms and U Sch♀tler. The Nonhydrostatic Limited-Area Model LM (Locak-model) of the DWD. Part I: Scientific Documentation. In DWD, GB Forschung und Entwicklung 1999.

Mario Drumond, Alexandros Daglis, Nooshin Mirzadeh, Dmitriu Ustiu, Javier Picorel, Babak Falsafi, Boris Grot, and Dionisios Pnevmatikatos. The Mondrian Data Engine. In ISCA 2017.

Javier Duarte, Song Han, Philip Harris, Sergo Jindariani, Edward Kreinar, Benjamin Kreis, Jennifer Ngadiuba, Maurizio Piri, Ryan Rivera, Nhan Tran, and Z Wu. Fast inference of deep neural networks in FPGAs for pinproceedings physics. In JINST 2018.

Jian Fang, Yvo T. B. Mulder, Jan Hidders, Jinho Lee, and H. Peter Hofstete. In-memory database acceleration on FPGAs: a survey. In VLDB 2020.

A. Farmahini-Farahan, J. H. Ahn, K. Morrow, and N. S. Kim. NDA: Near-DRAM Acceleration Architecture Leveraging Commodity DRAM Devices and Standard Memory Modules. In HPCA 2015.

Ivan Fernandez, Ricardo Quislant, Eladio Gutiérrez, Oscar Plata, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, and Onur Mutlu. NATS: A New-Data Processing Accelerator for Time Series Analysis. In ICCD 2020.

Michael J Flynn. Very High-Speed Computing Systems. In Proceedings of the IEEE 1966.

Haohuan Fu and Robert G Clapp. Eliminating the memory bottleneck: an FPGA-based solution for 3D reverse time migration. In FPGA 2011.

Brian Gaide, Dinesh Gaitonde, Chirag Ravishankar, and Trevor Bauer. Xilinx Adaptive Compute Acceleration Platform: Versal™ Architecture. In FPGA 2019.

Fei Gao, Georgios Triantou, and David Wentzlaff. ComputeDRA:M: In-Memory Compute Using Off-the-Shelf DRAMs. In MICRO 2019.

Mingyu Gao, Grant Ayers, and Christos Kozyrakis. Practical Near-Data Processing for In-Memory Analytics Frameworks. In PACT 2015.

M. Gao and C. Kozyrakis. HRL: Efficient and Flexible Reconfigurable Logic for Near-Data Processing. In HPCA 2016.

Saugata Ghose, Amirali Boroumand, Jeremy S Kim, Juan Gómez-Luna, and Onur Mutlu. Processing-in-memory: A workload-driven perspective. In IBM JRD 2019.

Saugata Ghose, Tianshi Li, Nastaran Hajzana, Da,mla Senol Cali, and Onur Mutlu. Demystifying Complex Workload-DRAM Interactions: An Experimental Study. In PPoPP 2019.

Christina Giannoula, Nandita Vijaykumar, Nikela Papadopoulou, Vasileios Karakostas, Ivan Fernandez, Juan Gómez-Luna, Lois Orosa, Nectarios Koziros, Georgios Goumas, and Onur Mutlu. SyncCron: Efficient Synchronization Support for Near-Data Processing Architectures. In HPCA 2021.

Heiner Giefer, Raphael Polig, and Christoph Hagl點tner. Accelerating arithmetic kernels with coherent attached FPGA coprocessors. In DATE 2015.

Juan Gómez-Luna, Izrat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu. Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture. In arXiv 2021.

Juan Gómez-Luna, Izrat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu. Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware. In CUT 2021.

José González and Antonio González. Speculative Execution via Address Prediction and Data Prefetching. In ICS 1997.

Boncheol Gu, Andre S. Yoon, Duck-Ho Ba, Insoon Jo, Jinyoung Lee, Jonghyun Yoon, Jeong-Uk Kang, Mooonsang Kwon, Chanho Yoon, Sangyeun Cho, Jaeheon Jeong, and Duckhyun Chang. Biscuit: A Framework for Near-data Processing of Big Data Workloads. In ISCA 2016.

Tobias Gyssi, Tobias Grosser, and Torsten Hoeftler. MODESTO: Data-centric Analytic Optimization of Complex Stencil Programs on Heterogeneous Architectures. In SC 2015.
[80] Nastaran Hajinaz, Geraldo F Oliveira, Sven Gregorio, Joao Ferreira, Nika Mansouri Ghasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gómez Luna, and Onur Mutlu. SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM. In ASPLOS 2021.

[81] Milad Hashemi, Eiman Ebrahimi, Onur Mutlu, and Yed N Patt. Accelerating Dependent Cache Misses with an Enhanced Memory Controller. In ISCA 2016.

[82] Milad Hashemi, Onur Mutlu, and Yale N Patt. Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads. In MICRO 2016.

[83] Tom Henretty, Kevin Stock, Louis-Noé Pouchet, Franz Franchetti, J Ramanujam, and P Sadayappan. Data Layout Transformation for Stencil Computations on Short-Vector SIMD Architectures. In CC 2011.

[84] Tszmin Hermesilla, E Bermejo, A Balaguer, and Luis A Ruiz. Non-linear fourth-order image interpolation for subpixel edge detection and localization. In IMAVIS 2008.

[85] Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladri Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W Keckler. Transparent Offloading and Mapping (TOM). Enabling Programmer-Transparent Near-Data Processing in GPU Systems. In ISCA 2016.

[86] Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu. Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation. In ICCD 2016.

[87] Sitao Huang, Li-Wen Chang, Izzat El Hajj, Simon Garcia de Gonzalo, Juan Gómez-Luna, Sai Rahul Chalamalasetti, Mohamed El-Hadedy, Dejan Milojicic, Onur Mutlu, Deming Chen, and Wen-mei Hwu. Analysis and Modeling of Collaborative Execution Strategies for Heterogeneous CPU-FPGA Architectures. In ICPE 2019.

[88] HT Huynh, Zhi J Wang, and Peter E Vincent. High-order methods for computational fluid dynamics: A brief review of compact differential formulations on unstructured grids. In Computers & Fluids 2014.

[89] Zsolt István, David Sudler, and Gustavo Alonso. Carbou: Intelligent Distributed Storage. In VLDB 2017.

[90] Jantong Jang, Zheke Wang, Xue Liu, Juan Gómez-Luna, Nan Guan, Qingxu Deng, Wei Zhang, and Onur Mutlu. Boyi: A Systematic Framework for Automatically Deciding the Right Execution Model of OpenCL Applications on FPGAs. In FPGA 2020.

[91] R. Jongerius, S. Wijnholds, R. Nijboer, and H. Corporaal. An End-to-End Computing Model for the Square Kilometre Array. In Computer 2017.

[92] Sang-Woo Jun, Ming Liu, Sungjin Lee, Jamey Hicks, John Ankcorn, Myron King, Shuotao Xu, et al. BlueDBM: An Appliance for Big Data Analytics. In ISCA 2015.

[93] Yangwook Kang, Yang-suk Kee, Ethan L Miller, and Chanik Park. Enabling Cost-effective Data Processing with Smart SSD. In ASISt 2013.

[94] Kaan Kara, Christof Hagleitner, Dionysios Diamantopoulos, Dimitris Syrivelis, and Gustavo Alonso. High Bandwidth Memory on FPGAs: A Data Analytics Perspective. In FPL 2020.

[95] L. Ke, U. Gupta, B. Y. Cho, D. Brooks, V. Chandra, U. Diril, A. Firoozshahian, K. Hazelwood, B. Jia, H. S. Lee, M. Li, B. Maher, D. Mudigere, M. Naumov, M. Schatz, M. Smelyanskiy, X. Wang, B. Reagen, C. Wu, M. Hempstead, and X. Zhang. ReconMP: Accelerating personalized recommendation with near-memory processing. In ISCA 2020.

[96] Scott Kehler, John Hanesiak, Michelle Curry, David Sills, and Neil Taylor. High Resolution Deterministic Prediction System (HRDPS) Simulations of Manitoba Lake Breezes. In Atmosphere-Ocean 2016.

[97] Duckbwan Kim, Jaeha Kung, Sek Chai, Sudhakar Yalamanchili, and Saibal Mukhopadhyay. Neurcube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory. In ISCA 2016.

[98] J. Kim, C. S. Oh, H. Lee, D. L. Hwang, S. Hwang, B. Na, J. Moon, J. Kim, H. Park, J. Ryu, K. Park, S. K. Kang, S. Kim, H. Kim, J. Bang, H. Cho, M. Jung, C. Han, J. Lee, J. S. Choi, and Y. Jun. A 1.2 V 12.8 GB/s 2 Gb Mobile Wide-I/O DRAM With 4x128 I/Os Using TSV Based Stacking. In JSSC 2012.

[99] Jerome S Kim, Daamli Senol Cali, Hongxiu Xu, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hassan Hassan,欧阳Egin, Can Alkan, and Onur Mutlu. GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies. In BMC Genomics 2018.

[100] Gunjae Koo, Kiran Kumar Matam, Te I, H. V. Krishna Giri Narra, Jing Li, Hung-Wei Tseng, Steven Swanson, and Murali Annaram. Summarizer: Trading Communication with Computing Near Storage. In MICRO 2017.

[101] Young-Cheon Kwon, Suk Han Lee, Jaehoon Lee, Sang-Hyuk Kwon, Je Min Ryu, Jong-Pil Son, Seongd O, Hak-Soo Yu, Haesuk Lee, Soo Young Kim, Youngmin Cho, Jn Gu Kim, Jongyoon Choi, Hyun-Sung Shin, Jin Kim, Beng-Seng Phua, HyoungMin Kim, Myeong Jun Song, Ahn Choi, Daiho Kim, SooYoung Kim, Eun-Bong Kim, David Wang, Shinhaeng Kang, Yuhwan Ro, Seungwoo Seo, JoonHo Song, Jaeyoung Yoon, Kyomin Sohn, and Nam Sung Kim. A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications. In ISCC 2021.

[102] Yi-Hsiang Lai, Yuce Chi, Youwei Hu, Jie Wang, Cody Hao Yu, Yuan Zhou, Jason Cong, and Zhuru Zhang. HeteroCL: A Multi-Paradigm Programming Infrastructure for Software-Defined Reconfigurable Computing. In FPGA 2019.

[103] Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Samira Khan, and Onur Mutlu. Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost. In ACM TACO 2016.

[104] D. U Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, J. Y. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, J. H. Cho, K. H. Kwon, M. J. Kim, J. Lee, K. W. Park, B. Chung, and S. Hong. 25.2 A 1.2V 8-Channel 128GB/s High-Bandwidth Memory (HBM) Stacked DRAM with Effective Microbump I/O Test Methods Using 29nm Process and TSV. In JSSC 2014.
[106] Jinho Lee, Heesu Kim, Sungjoo Yoo, Kiyoung Choi, H. Peter Hofstee, Gi-Joon Nam, Mark R. Nutter, and Damir Jamsak. ExtraV: Boosting Graph Processing near Storage with a Coherent Accelerator. In VLDB 2017.

[107] Joo Hwan Lee, Jaewoong Sim, and Hyeseon Kim. BSSync: Processing Near Memory for Machine Learning Workloads with Bounded Staleness Consistency Models. In PACT 2015.

[108] Sukhan Lee, Shin-haeng Kang, Jaehoon Lee, Hyeonsu Kim, Eojin Lee, Seungwoo Seo, Hosang Yoon, Seungwon Lee, KYoungbwan Lim, Hyunsung Shin, Jinhyun Kim, Seongil G. Anand Iyer, David Wang, Kyomin Sohn, and Nam Sung Kim. Hardware Architecture and Software Stack for FIM Based on Commercial DRAM Technology. In ISCA 2021.

[109] Vincent T Lee, Amirita Marumdar, Carlo C. de Mundo, Armin Alaghi, Luis Ceze, and Mark Oskin. Application Codesign of Near-Data Processing for Similarity Search. In IPDPS 2018.

[110] Jiajie Li, Yuze Chi, and Jason Cong. HeteroHalide: From image processing DSL to efficient FPGA acceleration. In FPGA 2020.

[111] Jiawen Liu, Hengyu Zhao, Mathieu A Ogleari, Dong Li, and Jishen Zhao. Processing-in-Memory for Energy-efficient Neural Network Training: A Heterogeneous Approach. In MICRO 2018.

[112] Shuangchen Li, Cong Xu, Qiaoasha Zou, Jishen Zhao, Yu Lu, and Yuan Xie. Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories. In DAC 2016.

[113] Jiwen Liu, Hengyu Zhao, Mathieu A Ogleari, Dong Li, and Jishen Zhao. Processing-in-Memory for Energy-efficient Neural Network Training: A Heterogeneous Approach. In MICRO 2018.

[114] Zhiyu Liu, Irina Calciu, Maurice Herlihy, and Onur Mutlu. Concurrent Data Structures for Near-Memory Computing. In SPAA 2017.

[115] Jiayuan Meng and Kevin Skadron. A Performance Study for Iterative Stencil Loops on GPUs with Ghost Zone Optimizations. In JPPE 2011.

[116] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. A Modern Primer on Processing in Memory. In DATE 2019.

[117] Onur Mutlu, Saugata Ghose, Jian Gómez-Luna, and Rachata Ausavarungnirun. Enabling Practical Processing in and near Memory for Data-Intensive Computing. In DAC 2019.

[118] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. Processing Data Where It Makes Sense: Enabling In-Memory Computation. In Micro 2019.

[119] Onur Mutlu, Saugata Ghose, Jian Gómez-Luna, and Rachata Ausavarungnirun. A Modern Primer on Processing in Memory. In Emerging Computing: From Devices to Systems-Looking Beyond Moore and Von Neumann. Springer 2021.

[120] L. Nai, R. Hadidi, J. Sim, H. Kim, P. Kumar, and H. Kim. GraphPIM: Enabling Instruction-Level PIM Offloading in Graph Computing Frameworks. In HPCA 2017.

[121] R. Nair, S. F. Antao, C. Bertoldi, P. Bose, J. R. Brunheroto, T. Chen, C. . Cher, C. H. A. Costa, J. Doi, C. Evangelinos, B. M. Fleischer, T. W. Fox, D. S. Gallo, L. Grinberg, J. A. Gunnels, A. C. Jacob, P. Jacob, H. M. Jacobson, T. Karkhanis, C. Kim, J. H. Moreno, J. K. O’Brien, M. Ohmacht, Y. Park, D. A. Prener, B. S. Rosenberg, K. D. Ryu, O. Sellenave, M. J. Serrano, P. D. M. Siegl, K. Sugavanam, and Z. Sura. Active Memory Cube: A Processing-in-Memory Architecture for Exascale Systems. In IBM JRD 2015.

[122] Dário CP Navarro, Husein Mohsen, Chengfei Yan, Shantao Li, Mengting Gu, William Meyerson, and Mark Gerstein. Genomics and data science: an application within an umbrella. In BMC 2019.

[123] Richard B Neale, Chih-Chieh Chen, Andrew Gettelman, Peter H Lauritzen, Sungsu Park, David L Williamson, Andrew J Conley, Rolando Garcia, R. Nair, S. F. Antao, C. Bertolli, P. Bose, J. R. Brunheroto, T. Chen, C. . Cher, C. H. A. Costa, J. Doi, C. Evangelinos, B. M. Fleischer, T. W. Fox, D. S. Gallo, L. Grinberg, J. A. Gunnels, A. C. Jacob, P. Jacob, H. M. Jacobson, T. Karkhanis, C. Kim, J. H. Moreno, J. K. O’Brien, M. Ohmacht, Y. Park, D. A. Prener, B. S. Rosenberg, K. D. Ryu, O. Sellenave, M. J. Serrano, P. D. M. Siegl, K. Sugavanam, and Z. Sura. Active Memory Cube: A Processing-in-Memory Architecture for Exascale Systems. In IBM JRD 2015.

[124] Onur Mutlu, Saugata Ghose, Jian Gómez-Luna, and Rachata Ausavarungnirun. Scalable Tensor Reduction in Memory. In MICRO 2021.

[125] Shuangchen Li, Cong Xu, Qiaoasha Zou, Jishen Zhao, Yu Lu, and Yuan Xie. Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories. In DAC 2016.

[126] Jiwen Liu, Hengyu Zhao, Mathieu A Ogleari, Dong Li, and Jishen Zhao. Processing-in-Memory for Energy-efficient Neural Network Training: A Heterogeneous Approach. In MICRO 2018.

[127] M Necati Özişik, Helcio RB Orlande, Marcelo J Colaço, and Renato M Cotta. 2017. Finite difference methods in heat transfer

[128] Geraldo Francisco Oliveira, Juan Gómez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan Fernandez, Mohammad Sadrosadati, and Onur Mutlu. DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks. In IEEE Access 2021.

[129] Jaehyun Park, Byeongho Kim, Sungmin Yun, Eojin Lee, Minsoo Rhu, and Jung Ho Ahn. TRiM: Enhancing Processor-Memory Interfaces with Tensor Reduction in Memory. In MICRO 2021.

[130] Satish Kumar Sadasivam, Brian W Thompto, Ron Kalla, and William J Starke. IBM POWER9 Processor Architecture. In IEEE Micro 2017.

[131] Constantin Pohl, Kai-Uwe Satller, and Goetz Graefe. Joins on high-bandwidth memory: a new level in the memory hierarchy. In VLDB 2019.

[132] Seth H Pugsley, Jeffrey Jestes, Huihui Zhang, Rajeev Balasubramonian, Vijayalakshmi Srinivasan, Al Davis, and Feifei Li. Heterogeneous Approach. In IPDPS 2018.

[133] Seth H Pugsley, Jeffrey Jestes, Huihui Zhang, Rajeev Balasubramonian, Vijayalakshmi Srinivasan, Al Davis, and Feifei Li. Heterogeneous Approach. In IPDPS 2018.

[134] Krzysztof Rojek et al. CFD Acceleration with FPGA. In PACT 2016.

[135] Seth H Pugsley, Jeffrey Jestes, Huihui Zhang, Rajeev Balasubramonian, Vijayalakshmi Srinivasan, Al Davis, and Feifei Li. Heterogeneous Approach. In IPDPS 2018.

[136] Seth H Pugsley, Jeffrey Jestes, Huihui Zhang, Rajeev Balasubramonian, Vijayalakshmi Srinivasan, Al Davis, and Feifei Li. Heterogeneous Approach. In IPDPS 2018.
[117] Christoph Schär, Oliver Fuhrer, Andrea Arteaga, Nikola Ban, Christophe Charpillot, Salvatore Di Girolamo, Laureline Hentgen, Torsten Hoefler, Xavier Laplantine, David Leutwyler, Katherine Osterried, Davide Panosetti, Stefan Rudolph, Linda Schlemmer, Thomas C. Schulthess, Michael Sprenger, Stefano Ubbiali, and Heini Wernli. Kilometer-scale Climate Models: Prospects and Challenges. In RAMS 2020.

[138] Vivek Seshadri, Kevin Hishek, Amirali Boroumand, Donghyuk Lee, Michael A Kozuch, Onur Mutlu, Phillip B Gibbons, and Todd C Mowry. Fast Bulk Bitwise AND and OR in DRAM. In CIL 2015.

[139] Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Phillip B Gibbons, Michael A Kozuch, et al. RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization. In MICRO 2013.

[140] Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A Kozuch, Onur Mutlu, Phillip B Gibbons, and Todd C Mowry. Buddy-RAM: Improving the Performance and Efficiency of Bulk Bitwise Operations Using DRAM. In arXiv 2016.

[141] Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A Kozuch, Onur Mutlu, Phillip B Gibbons, and Todd C Mowry. Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology. In MICRO 2017.

[142] Vivek Seshadri, Thomas Mullins, Amirali Boroumand, Onur Mutlu, Phillip B Gibbons, Michael A Kozuch, and Todd C Mowry. Gather-Scatter DRAM: In-DRAM Translation to Improve the Spatial Locality of Non-unit Strided Accesses. In MICRO 2015.

[143] Vivek Seshadri and Onur Mutlu. 2017. Simple operations in memory to reduce data movement. In Advances in Computers.

[144] Vivek Seshadri and Onur Mutlu. In-DRAM bulk bitwise execution engine. In arXiv 2019.

[145] DD Sharma. Compute Express Link. In CXL Consortium White Paper 2019.

[146] William Andrew Simon, Yasir Mahmood Qureshi, Marco Rios, Alexandre Levisse, and Michael Sprenger. BLADE: An in-Cache Computing Architecture for Edge Devices. In TC 2020.

[147] Gagandeep Singh et al. NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning. In DAC 2019.

[148] Gagandeep Singh, Mohammed Ather, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu. FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications. In IEEE Micro 2021.

[149] Gagandeep Singh, Lorenzo Chelini, Stefano Corda, Ahsan Javed Awan, Sander Stuijk, Roel Jordans, Henk Corporaal, and Albert-Jan Boonstra. A Review of Near-Memory Computing Architectures: Opportunities and Challenges. In LSD 2018.

[150] Gagandeep Singh, Lorenzo Chelini, Stefano Corda, Ahsan Javed Awan, Sander Stuijk, Roel Jordans, Henk Corporaal, and Albert-Jan Boonstra. Near-Memory Computing: Past, Present, and Future. In Micro 2019.

[151] Gagandeep Singh, Dionysios Diamantopoulos, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal. Modeling FPGA-Based Systems via Few-Shot Learning. In FPGA 2021.

[152] Gagandeep Singh, Dionysios Diamantopoulos, Christoph Haglettein, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal. NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Mapping. In JPL 2020.

[153] Gagandeep Singh, Dionysios Diamantopoulos, Christoph Haglettein, Sander Stuijk, and Henk Corporaal. NARMADA: Near-memory horizontal diffusion accelerator for scalable stencil computations. In JPL 2019.

[154] Gagandeep Singh, Dionysios Diamantopoulos, Sander Stuijk, Christoph Hagletterin, and Henk Corporaal. Low Precision Processing for Higher Order Stencil Computations. In Springer LNCS 2019.

[155] Robert Strzodka, Mohammed Shaheen, Dawid Pajak, and Hans-Peter Seidel. Cache Oblivious Parallelism in Iterative Stencil Computations. In ICS 2010.

[156] Jeffrey Stuechel et al. IBM POWER9 opens up a new era of acceleration enablement: OpenCAPI. In IBM JRD 2018.

[157] Jeffrey Stuechel, Bart Blaner, CR Johnson, and MS Siegel. CAPI: A Coherent Accelerator Processor Interface. In IBM JRD 2015.

[158] B. Sukhwani, T. Roewer, C. L. Haymes, K. Kim, A. J. McPadden, D. M. Dreps, D. Sanner, J. V. Lunteren, and S. Asaad. ConTutto – A Novel IBM JRD Technology. In MICRO 2019.

[159] Robert Strzodka, Mohammed Shaheen, Dawid Pajak, and Hans-Peter Seidel. Cache Oblivious Parallelism in Iterative Stencil Computations. In ICS 2010.

[160] Yuan Tang, Rezaul Alam Chowdhury, Bradley C Kuszmaul, Chi-Keung Luk, and Charles E Leiserson. The Pochoir Stencil Compiler. In SPAA 2011.

[161] Felix Thaler, Stefano Moorsbrugger, Carlos Osuna, Mauro Bianco, Hannes Vogt, Anton Afanasiev, Lukas Mosimann, Oliver Fuhrer, Thomas C Schulthess, and Torsten Hoefler. Porting the COSMO Weather Model to Manycore CPUs. In PASC 2019.

[162] Jeevdlyn Thomas. Elliptic Problems in Linear Differential Equations over a Network. In Watson Sci. Comput. Lab. Rept., Columbia University Watson Sci. Comput. Lab. Rept., Columbia University 1949.

[163] Po-An Tsai et al. Jenga: Software-Defined Cache Hierarchies. In ISCA 2017.

[164] Dean M Tullsen, Susan J Eggers, and Henry M Levy. Simultaneous Multithreading: Maximizing On-Chip Parallelism. In ISCA 1995.

[165] Jan van Lunteren, Ronald Luijten, Dionysios Diamantopoulos, Florian Auerhammer, Christoph Haglettein, Lorenzo Chelini, Stefano Corda, and Gagandeep Singh. Coherently Attached Programmable Near-Memory Acceleration Platform and its application to Stencil Processing. In DATE 2019.

[166] Vasile Volkov and James W Demmel. Benchmarking GPUs to tune dense linear algebra. In SC 2008.

[167] Mohamed Wahib and Naoya Maruyama. Scalable Kernel Fusion for Memory-Bound GPU Applications. In SC 2014.

[168] Hasitha Mathumala Waidyassooriya and Masanori Hariyama. Multi-FPGA accelerator architecture for stencil computation exploiting spacial and temporal scalability. In IEEE Access 2019.

[169] H. M. Waidyassooriya, Y. Takei, S. Tatsumi, and M. Hariyama. OpenCL-Based FPGA-Platform for Stencil Computation and Its Optimization Methodology. In TPDS 2017.
Accelerating Weather Prediction using Near-Memory Reconfigurable Fabric

[170] Shuo Wang and Yun Liang. A comprehensive framework for synthesizing stencil algorithms on FPGAs using OpenCL model. In DAC 2017.
[171] Zeke Wang, Hongjing Huang, Jie Zhang, and Gustavo Alonso. Shuhai: Benchmarking High Bandwidth Memory on FPGAs. In FCCM 2020.
[172] Lukas Wenzel, Robert Schmid, Balthasar Martin, Max Plauth, Felix Eberhardt, and Andreas Polze. Getting Started with CAPI SNAP: Hardware Development for Software Engineers. In Euro-Pар 2018.
[173] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An Insightful Visual Performance Model for Multicore architectures. In CACM 2009.
[174] Lingxi Wu, Rasool Sharifi, Marzieh Lenjani, Kevin Skadron, and Ashish Venkat. Sieve: Scalable In-situ DRAM-based Accelerator Designs for Massively Parallel k-mer Matching. In ISCA 2021.
[175] Jingheng Xu, Haoxuan Fu, Wen Shi, Lin Gan, Yuxuan Li, Wayne Luk, and Guangwen Yang. Performance Tuning and Analysis for Stencil-Based Applications on POWER8 Processor. In ACM TACO 2018.
[176] Dongping Zhang, Nuwan Jayasena, Alexander Lyashevsky, Joseph L Greathouse, Lifan Xu, and Michael Ignatowski. TOP-PIM: Throughput-Oriented Programmable Processing in Memory. In HPDC 2014.
[177] Jun A. Zhang, Frank D. Marks, Jason A. Sippel, Robert F. Rogers, Xuejin Zhang, Sundararaman G. Gopalakrishnan, Zhan Zhang, and Vijay Tallapragada. Evaluating the Impact of Improvement in the Horizontal Diffusion Parameterization on Hurricane Prediction in the Operational Hurricane Weather Research and Forecast (HWRF) Model. In Weather and Forecasting 2018.
[178] Maohua Zhu, Youwei Zhuo, Chao Wang, Wenguang Chen, and Yuan Xie. Performance Evaluation and Optimization of HBM-Enabled GPU for Data-intensive Applications. In VLSI 2018.
[179] Hamid Reza Zohouri, Artur Podobas, and Satoshi Matsuoka. Combined spatial and temporal blocking for high-performance stencil computation on FPGAs using OpenCL. In FPGA 2018.