Benchmarking Floating Point Performance of Massively Parallel Dataflow Overlays on AMD Versal Compute Primitives

Mohamed Bouaziz and Suhaib A. Fahmy
King Abdullah University of Science and Technology (KAUST)
Thuwal, Saudi Arabia
mohamed.bouaziz@kaust.edu.sa

***Abstract*—Many massively parallel applications require floating-point (FP) precision, necessitating specialized hardware support. Novel Reconfigurable Dataflow Accelerators (RDAs) and Coarse Grained Reconfigurable Arrays (CGRAs) like the AMD Versal AI Engine Array can implement optimized datapaths. The AMD Versal FPGA family integrates AI Engine cores for vector FP operations as well as DSP58 primitives in the programmable logic for use in fine-grained architectures. Configurability at these levels involves factors, particularly data movement, that complicate measuring empirical compute performance limits. This paper presents an architectural model to isolate and measure these limits. Using this model, we compare resources, showing the superior operator density and performance of the DSP58 over the previous generation AMD UltraScale+ DSP48E2 for massively parallel dataflow overlays. We also show that the DSP58 can outperform programmable AI Engines in massively parallel feedforward applications.**

***Index Terms*—Dataflow, CGRA, massive parallelism.**

I. Introduction and Related Work

Floating Point (FP) representation is essential in many massively parallel applications across domains, such as fluid dynamics, machine learning, and others [1]. Many general-purpose architectures come with optimized parallel/vector FP units such as Advanced Vector Extensions (AVX) [2] in various x86 Intel and AMD CPUs and Scalable Vector Extension (SVE) [3] on ARM CPUs that support a high number of parallel SIMD FP operations. Similarly, the Ampere architecture [4] of Nvidia GPUs can perform FP tensor operations with different precisions. Those architectures can be leveraged for massively parallel applications. However, dataflow accelerators still find use with compute patterns that do not fit these general-purpose architectures well. Hence, we must consider these operations when designing efficient dataflow overlays on architectures like the AMD Versal [5].

The AMD Versal architecture combines fine-grained and coarse-grained modalities to implement dataflow architectures, as shown in Fig. 1. At the fine-grained level, the Programmable Logic (PL) contains optimized hardware resources that can be combined into arbitrary arrangements. In particular DSP blocks, are used to implement arithmetic in the PL. Specifically, the DSP58 [6] in the AMD Versal natively supports single-precision FP operations, unlike the older DSP48E2 [7] on AMD UltraScale+ [8] and previous generations that only supported fixed-point operations directly. FP operations can be implemented using older DSP blocks without native FP support, but this requires additional hardware to compose fixed-point/integer operations and results in extra timing overhead. Floating point operator synthesis can be done through vendor or open source IPs [9], [10] and High-Level Synthesis (HLS) tools also automate this for supported datatypes.

The AMD Versal has a new additional resource: an array of software-programmable cores, called AI Engines (AIEs), that offer SIMD FP and integer support and that can be used for massively parallel applications. These are connected via streaming interconnect and access data from off-chip and on-chip memory.

Although both the DSP58s and AIEs support native FP operations, data movement patterns play a crucial role in their overall throughput. Analyzing the practical limits of these primitives is complex as compute performance is not solely determined by raw arithmetic capability. DSP blocks tend to absorb most of the arithmetic workload in FPGA designs, but achieving optimal mapping, especially from HLS, remains challenging [11]. Consequently, evaluating DSP efficiency requires an application-level perspective, as computational capability does not solely determine overall effectiveness. Furthermore, comparing newer primitives such as the Versal DSP58 and AIEs against older ones like the UltraScale+ DSP48E2 is complicated by differences in architectural integration, such as the presence of High-Bandwidth Memory (HBM) in devices like the Alveo U280 [12], while not being available on any architecture with the AI Engines.

This work aims to propose an architectural benchmarking model for analyzing the implementation of feed-forward massively parallel dataflow overlays that leverage FP operations across AMD Versal primitives. The proposed model provides a way to separate the compute primitives' performance indicators from the data movement implications. It also enables fair comparison with the older DSP48E2.

We employ the model to compare the performance of the Versal DSP58 block against the UltraScale+ DSP48E2 by measuring the maximum achievable frequency for various design points for PL-based designs. As the AIEs are situated within the AIE Array, which is distinct from the fine-grained PL, we compare performance against the PL-based designs

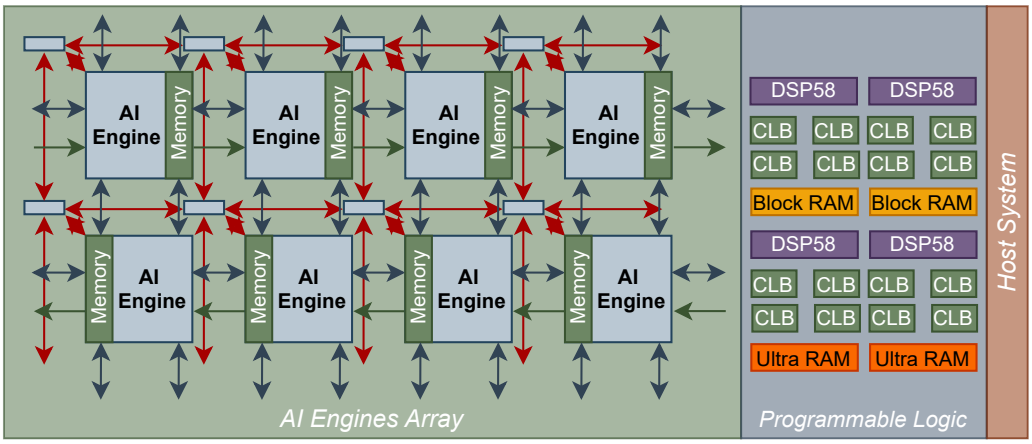

Fig. 1: AMD Versal SoC architecture.

```
void aie_vmul_stream(input_window<float> *in0,
                     input_window<float> *in1,
                     output_windw<float> *out) {
  aie::vector<float, 8> a = window_readincr_v<8>(in0);
  aie::vector<float, 8> b = window_readincr_v<8>(in1);
  aie::vector<float, 8> res = aie::mul(a, b);
  window_writeincr<8>(out, res);
}
```

Fig. 2: Element-wise vector multiplication on AIE.

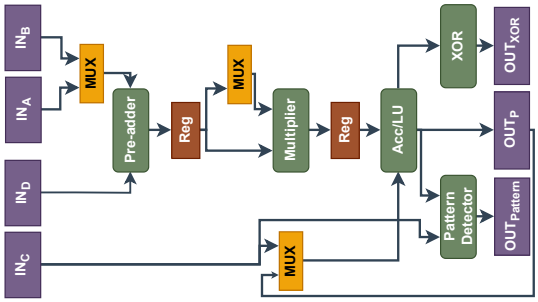

Fig. 3: DSP Block simplified block diagram.

leveraging the DSP blocks in terms of speedup and energy efficiency. We take massively parallel element-wise vector multiplication as an example in our model, as it represents the backbone of many important acceleration workloads, including matrix-matrix and matrix-vector multiplication. We show that Versal primitives (DSP58s and AIEs) offer higher performance for massively parallel overlays than UltraScale+ and enable scalable dataflow implementation. This is despite lacking HBM, present in UltraScale+, which enables high throughput data movement. We open-source the model in this repository [13].

II. Floating Point on DSP Blocks vs AI Engines

This section explores the differences in FP datapaths between AIEs and DSP58 primitives, detailing how each can be configured for SIMD FP operations. It also discusses optimization techniques, mainly through pipelining, and explains how these primitives can be structured to extract pipelining efficiency.

A. DSP vs AIEs Datapath Differences

As a RISC core optimized for highly vectorized workloads across various data types, the AI Engine natively supports FP operations. In particular, using the AIE-API [14], it can perform 8-lane SIMD FP multiplication per cycle using `aie::mul`, as shown in Fig. 2.

DSP blocks in AMD FPGAs (illustrated in Fig. 3) are highly flexible, with runtime programmable control inputs that can modify their datapath configuration to implement different graphs of add/multiply/accumulate and similar functions. This flexibility has been exploited at a low level to build programmable overlays [15], however, current synthesis tools will usually statically set these to implement fixed graphs of arithmetic functionality. Due to this flexibility, DSP blocks tend to implement most integer arithmetic operations during synthesis. The AMD LogiCore Floating-Point Operator IP [9] uses DSP blocks with additional logic to implement FP operations. The number of DSP blocks required, cascading scheme (if applicable), and configuration vary based on the DSP block version and the operation.

B. DSP58 vs DSP48E2 Functionality

We analyze two state-of-the-art AMD FPGA DSP blocks: the DSP48E2 [7], featured in the UltraScale+ architecture, and the DSP58 [6], included in the newer Versal architecture. Notably, the DSP58 is a functional superset of the older DSP48E2, offering backward compatibility.

The AMD LogiCore Floating-Point Operator IP [9] enables the implementation of a single-precision FP multiplication operation. As shown in Table I, an 8-lane multiplier (including data movement circuitry) using DSP48E2 blocks requires 3 DSPs per lane. This employs the `MAXDSP` IP configuration, which cascades and operates the DSPs as an FP multiplier using extra surrounding logic. The same operation on the

TABLE I: Resource usage of 8-lane PL-based multipliers.

Board	DSP blocks	BRAMs	LUTs	FFs
UltraScale+ U280	24×DSP48E2	12	7287	10182
Versal VCK5000	8×DSP58	12	5841	9124
Usage comparison	3×	1×	1.25×	1.16×

DSP58 primitive requires only one DSP per lane due to native FP support and reduced surrounding logic. This is using the `PRIMITIVEDSP` IP configuration.

C. Pipelining DSP blocks

Pipelining complex datapaths into multiple stages enables higher operating frequencies on the PL. Depending on the IP implementation and the DSP primitive used, different numbers of configurable pipeline stages are supported: using the DSP58 up to 4 stages, while the DSP48E2 based implementation can use up to 7 stages for FP multiplication.

DSP blocks are optimized for integration into deep custom dataflow pipelines, leveraging a spatial computing paradigm to deliver high performance. On the other hand, the AIE is a processor based on the traditional von Neumann architecture, with an instruction memory that stores a program. Its performance stems from vector execution and the high 1 GHz operating frequency. DSP blocks can also be used to implement programmable processors when needed [16]. This paper focuses on a massively parallel feed-forward dataflow application.

III. Proposed Architectural Model

In this section, we propose an architectural model of a dataflow overlay that can be implemented on the FPGA PL using DSP blocks and on the AIE Array using AI Engines. This architectural model aims to provide high FP operation density and reduced logic and control complexity. This is to distil the effective compute performance of the available resources, without significant impact from data movement and control.

A. Architectural Model

Constructing a benchmarking model to evaluate the performance of FP primitives involves several factors. The PL is more fine-grained than the AIE Array, with data movement being completely custom, unlike the regular vertical and horizontal streams on the AIE Array. Besides, current AMD Versal boards with AIEs lack HBM, which limits highly parallel off-chip data movement compared to AMD UltraScale+ boards. Internal data communication optimizations, such as shared buffers, can also impact the measurement of compute performance. Therefore, we require the following to build an architectural model that isolates the performance of FP primitives for comparison:

- The dataflow should only include feed-forward streams of data.

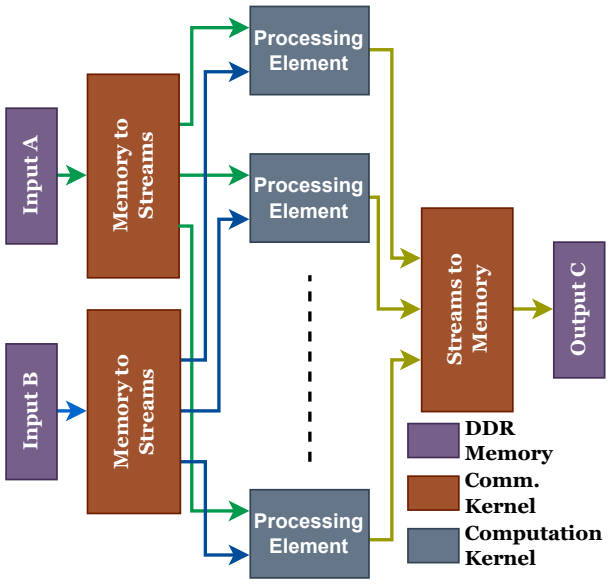

Fig. 4: Implemented architectural model.

- Off-chip data communication must be done with DRAM (no HBM).
- Computation per FP primitive does not share or synchronize intermediate results with other primitives.

Fig. 4 shows the architectural model used in this analysis. This model aims to ensure similar data movement and compute patterns across the DSP blocks and AIEs and Versal and UltraScale+ platforms.

For data movement, the Memory-to-Stream (for input) and Stream-to-Memory (for output) communication kernels, always mapped on the PL, are identical across all implementations. The channels are 32 bits wide. All channels handle vectors with sizes that are multiples of the total parallel inputs and outputs.

For computation in the PL-based designs, the HLS design of the Processing Elements (PEs) is mapped to DSP blocks and LUTs/FFs. In the AIE-based designs, PEs are implemented as computation kernels using the AIE-API [14]. In this case, communication kernels connect to the AIE array through PL Interface tiles, which allow connections to multiple 32-bit channels.

In both designs, the PEs execute the single-precision FP multiplication operation in an 8-lane SIMD fashion, processing data as soon as it becomes available. Thus, both implementations follow the same data movement and execution pattern, allowing a fair comparison.

B. Experimental Setup

We use the AMD VCK5000 board with the Versal XCVC1902 device and the AMD Alveo U280 with the UltraScale+ XCU280 device, whose resource availabilities are shown in Table II.

The UltraScale+ Alveo U280 has 4.58× as many DSP blocks as the Versal VCK5000. However, FP designs require

TABLE II: Resource availability on target platforms.

Board	DSP	BRAM	LUT	FF
UltraScale+ U280	9024	2016	1304K	2607K
Versal VCK5000	1968	1934	899K	1799K
Resource comparison	×4.58	×1.04	×1.45	×1.45

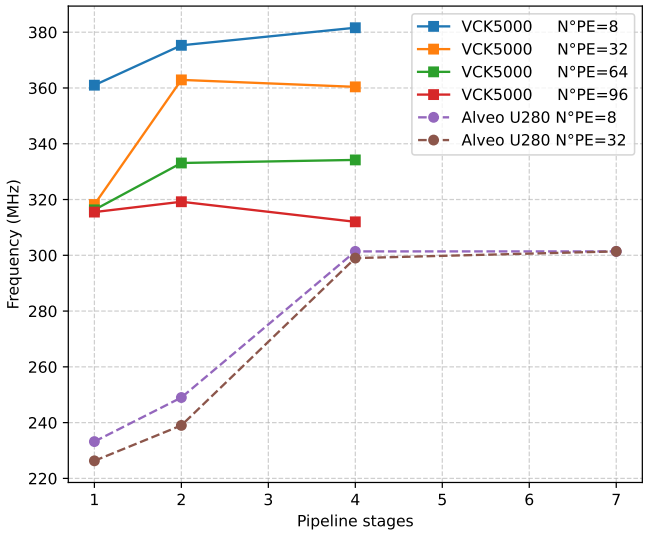

Fig. 5: Maximum achievable frequency for different numbers of 8-lane SIMD PEs.

one third of the DSP blocks per FP functional unit, as discussed in Section II-B, resulting in a normalized DSP usage for FP implementations of 1.53×. Moreover, the Alveo U280 features HBM and DRAM memories, while the VCK5000 only features DRAM. Hence, we only use the DRAM memory to satisfy the model constraints.

For the AIE-based designs, we use the communication kernels in the PL as the AIEs have higher bandwidth to the PL than to DRAM [17], and to ensure consistency with the PL-based designs.

C. Fine-Grained Primitives: DSP Blocks Comparison

The DSP primitives used in this analysis implement FP operations differently and support varying numbers of pipeline stages. Since pipelining impacts frequency, we compare the maximum achievable frequency for the PE designs based on both the DSP48E2 and DSP58.

Fig.5 illustrates how maximum frequency scales with the number of pipeline stages enabled in the FP multipliers for different numbers of PEs (in multiples of 8 SIMD lanes) on both the VCK5000 and the Alveo U280. As noted in Sec.II-C, the DSP58 FP multipliers can be pipelined to a maximum depth of 4 cycles.

As the number of PEs increases, the frequency drops due to higher DSP block utilization and more additional resources needed for pipelining causing some congestion, as discussed

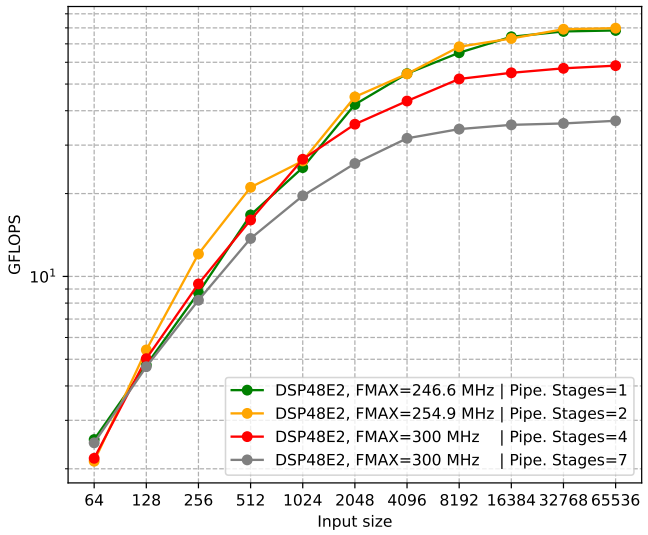

Fig. 6: Effect of pipelining on the execution time of one 8-lane multiplier Processing Element.

in Section II-B. The best maximum achievable frequency on the Alveo U280 does not exceed the worst on the VCK5000, even with more pipeline stages. In addition, frequency plateaus at 4 pipeline stages. For the U280, designs with more than 32 PEs were unrouteable due to high congestion.

To isolate the effect of pipelining from the congestion issue on the Alveo U280, we pipelined a single PE with the same number of pipeline stages used previously. For a given operand vector size, the operation timing—how data flows in and out of the PE and how computation is performed—remains the same, allowing us to directly assess the impact of pipelining on performance. Fig. 6 shows that pipelining the DSP48E2 does not improve throughput as the problem size increases. The frequency plateaus at 4 pipeline stages and deeper pipelines introduce additional overhead for data movement.

This suggests that the FP multiplier design using three DSP48E2 blocks does not scale well, and congestion is a key constraint when implementing a high density of PEs, compared to the DSP58 primitive.

D. Coarse-Grained Primitives: AIE vs DSP

DSP blocks are situated within the PL, while the AIE Array is distinct from the PL. This means where data sources are within the PL, they can be ingested into massively parallel DSP block-based datapaths more easily than is possible for AIEs. So, while the raw computational capability and clock rate of the AIEs are high, data movement can dramatically impact overall application performance. To evaluate this, we analyze the speedup and energy efficiency of massively parallel dataflow using DSP blocks and AIEs. Out of the DSP blocks, we only consider the DSP58 primitive in this experiment as it allows scaling to a high number of PEs, as explained in Section III-C, enabling massive parallelism.

TABLE III: Design resource and power consumption.

	AIE Design	DSP58 Design
Processing Elements	384	184 (1472 DSP blocks)
Compute Resource Usage (%)	96% AIEs	74% DSP58
Total Design Power (W)	109.6	45.8

Speedup and energy efficiency are analyzed based on large input vectors to fully exploit the high number of PEs. Input sizes are chosen to occupy as many PEs as possible and are chunked in multiples of the total SIMD lanes of both the AIE and PL-based designs. The number of PEs used is given in Table III and is chosen empirically to maximize resource usage while not encountering congestion issues when routing.

As shown in Fig. 7, the FP vector element-wise multiplication operation is 1.74× times faster using the PL-based design than using the AIEs. Recall that our architectural model ensures similar communication and computation patterns for both architectures. Hence, this suggests that coupling communication through the PL with stream processing on the distinct AIE Array presents an potential practical performance bottleneck for the AIEs.

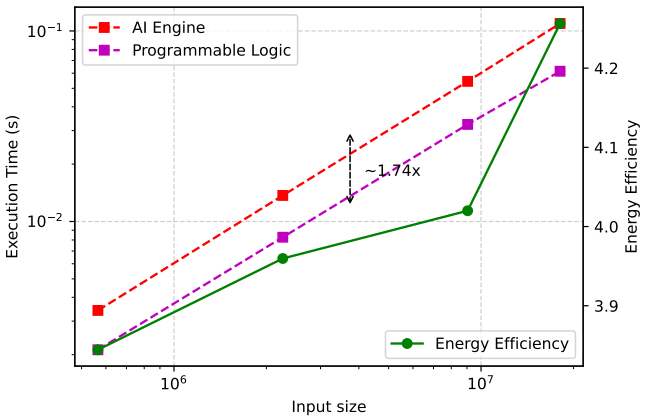

Fig. 7: Execution time and energy efficiency.

We extract the total chip power from the post-routing report. Table III shows that the PL-based design consumes less power than the array of the AIEs. We derive energy consumption from the measured runtime and power report. In Fig. 7, we also show the overall energy efficiency, which is defined as the ratio of the energy consumption of PL-based designs to the AIE design. It is demonstrated that PL-based designs are 3.8–4.3× as energy efficient as the AIE design for this large parallel FP multiplication application. In [18], it was shown that when there is data reuse, the AIEs can be more energy efficient as they retain data within their local memories for reuse. This confirms that while the AIE has a faster clock rate and an optimized datapath, its compute performance can only be fully realized with well-designed data movement patterns.

IV. Conclusion

This paper proposed an architectural model that distils the empirical limits of FP performance on the AMD Versal FPGAs for massively parallel dataflow overlays. At the primitive level, we have shown that the DSP58 significantly outperforms the older DSP48E2 of the AMD UltraScale+ architecture when pipelined, as well as supporting higher-density parallel datapaths. This allows the AMD Versal to implement massively parallel applications. We have also shown that at the coarse-grained level, AIEs have higher energy consumption than DSP blocks with feed-forward stream processing pipelines. We aim to extend this analysis to more complex dataflow overlays to better understand the trade-offs in implementing FP datapaths on modern FPGAs and RDAs/CGRAs.

References

[1] A. Abdelfattah, H. Anzt, E. G. Boman, E. Carson, T. Cojean, J. Dongarra, A. Fox, M. Gates, N. J. Higham, X. S. Li, J. Loe, P. Luszczek, S. Pranesh, S. Rajamanickam, T. Ribizel, B. F. Smith, K. Swirydowicz, S. Thomas, S. Tomov, Y. M. Tsai, and U. M. Yang, "A survey of numerical linear algebra methods utilizing mixed-precision arithmetic," *The International Journal of High Performance Computing Applications*.

[2] "Intel AVX-512 instructions," https://www.intel.com/content/www/us/en/developer/articles/technical/intel-avx-512-instructions.html, 2014.

[3] "Introduction to SVE," https://developer.arm.com/documentation/102476/0100, 2022.

[4] "NVIDIA A100 tensor core GPU architecture," https://resources.nvidia.com/en-us-genomics-ep/ampere-architecture-white-paper.

[5] AMD, "Versal acap: A multi-tier heterogeneous compute platform for adaptive acceleration," White Paper WP505, 2021. [Online]. Available: https://docs.amd.com/v/u/en-US/wp505-versal-acap

[6] "Versal ACAP DSP Engine architecture manual (am004)," https://docs.amd.com/r/en-US/am004-versal-dsp-engine, 2022.

[7] "Ultrascale architecture DSP slice user guide (ug579)," https://docs.amd.com/v/u/en-US/ug579-ultrascale-dsp, 2021.

[8] AMD, "Ultrascale architecture: Enabling smarter systems," White Paper WP434, 2014. [Online]. Available: https://docs.amd.com/v/u/en-US/wp434-ultrascale-smarter-systems

[9] "Floating-point operator v7.1 LogiCORE IP product guide (pg060)," https://docs.amd.com/v/u/en-US/pg060-floating-point, 2020.

[10] F. De Dinechin and B. Pasca, "Designing custom arithmetic data paths with FloPoCo," *IEEE Design and Test of Computers*, vol. 28, no. 4, pp. 18–27, 2011.

[11] B. Ronak and S. A. Fahmy, "Mapping for maximum performance on FPGA DSP blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 573–585, 2016.

[12] AMD, "Alveo u280 data center accelerator card data sheet," Data Sheet DS963, 2020. [Online]. Available: https://docs.amd.com/r/en-US/ds963-u280

[13] GitHub Repoistory, 2025. [Online]. Available: https://github.com/accl-kaust/fp-versal-bench

[14] "AI engine kernel and graph programming guide (ug1079)," https://docs.amd.com/r/en-US/ug1079-ai-engine-kernel-coding, 2022.

[15] A. K. Jain, D. L. Maskell, and S. A. Fahmy, "Coarse grained FPGA overlay for rapid just-in-time accelerator compilation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1478–1490, 2021.

[16] H. Y. Cheah, F. Brosser, S. A. Fahmy, and D. L. Maskell, "The iDEA DSP block-based soft processor for FPGAs," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 7, no. 3, pp. 19:1–19:23, 2014.

[17] "Versal Adaptive SoC AI Engine architecture manual (am009)," https://docs.amd.com/r/en-US/am009-versal-ai-engine, 2023.

[18] J. Zhuang, J. Lau, H. Ye, Z. Yang, Y. Du, J. Lo, K. Denolf, S. Neuendorffer, A. Jones, J. Hu, D. Chen, J. Cong, and P. Zhou, "CHARM: Composing heterogeneous accelerators for matrix multiply on Versal ACAP architecture," in *International Symposium on Field Programmable Gate Arrays*, 2023.