

BASE: A Reusable Streaming Preprocessing Accelerator for Heterogeneous FPGA Systems

Rye Stahle-Smith, Carter Antley, Jason D. Bakos, and Rasha Karakchi

Dept. of Computer Science and Engineering, University of South Carolina, Columbia, SC, USA

{ryes, ctantley}@email.sc.edu, {jbakos, karakchi}@cec.sc.edu

Abstract—Modern heterogeneous high-performance computing (HPC) systems increasingly rely on FPGA-based accelerators to improve throughput and energy efficiency for data-intensive applications. While considerable effort has focused on accelerating computation, many streaming workloads remain constrained by byte-level preprocessing and data movement, which frequently dominate end-to-end execution time.

This paper presents BASE (*Byte-level Analytics and Statistical Engine*), a reusable streaming preprocessing accelerator that integrates statistical feature extraction directly into the FPGA communication path. BASE performs single-pass histogram generation, entropy estimation, variance computation, and byte-transition analysis while data is transferred through the AXI DMA interface, overlapping preprocessing with communication and reducing host-side computation.

We demonstrate the proposed architecture using deployment-time FPGA bitstream screening as a representative streaming workload. Experimental results on the ZCU104 show that BASE reduces preprocessing latency by $5.04\times$ and improves end-to-end execution time by $4.23\times$. Evaluation on the U55C further demonstrates platform-dependent performance, with BASE achieving a $1.32\times$ preprocessing speedup and a $1.78\times$ end-to-end speedup. These results demonstrate that integrating reusable streaming preprocessing primitives into the FPGA communication path substantially improves data movement efficiency and heterogeneous resource utilization across data-intensive applications. Beyond deployment-time FPGA screening, BASE is applicable to a broad class of streaming analytics workloads, including telemetry processing, packet inspection, binary-stream analysis, and AI data preprocessing, making it a practical building block for future reconfigurable HPC systems.

Index Terms—FPGA Acceleration, High-Performance Computing, Streaming Architectures, Data Movement, Statistical Preprocessing, AXI DMA, Reconfigurable Computing

I. INTRODUCTION

The increasing adoption of heterogeneous high-performance computing (HPC) systems has fundamentally changed the way data-intensive applications are executed [1]–[3]. Modern HPC platforms integrate CPUs with domain-specific accelerators, including field-programmable gate arrays (FPGAs), to improve throughput, reduce energy consumption, and efficiently process increasingly diverse workloads [4]–[10]. Beyond traditional scientific simulation, contemporary HPC environments support machine learning, streaming analytics, real-time telemetry, network processing, and dynamically deployed accelerator services, all of which generate large volumes of binary and streaming data. As these workloads continue to

grow in scale, overall system performance is increasingly constrained by data movement and preprocessing rather than by computational throughput alone [1], [11], [12].

Many streaming applications require byte-level statistical preprocessing before higher-level analysis or machine learning can be performed [13]–[15]. Operations such as histogram generation, entropy estimation, statistical moment computation, and transition analysis are common across numerous application domains, including network telemetry, binary-stream analytics, malware analysis, FPGA deployment screening and validation. These preprocessing stages typically require multiple passes over large datasets, resulting in significant memory traffic, increased CPU utilization, and limited overlap between communication and computation. Consequently, preprocessing frequently becomes the dominant contributor to end-to-end execution time in heterogeneous computing systems [16]–[18].

Reconfigurable architectures provide an attractive solution to this challenge by enabling customized streaming datapaths that process information as it moves through the system [19]. Rather than treating preprocessing as a separate software stage executed after data transfer, FPGAs can integrate statistical feature extraction directly into the communication pipeline. This approach reduces redundant memory accesses, overlaps preprocessing with data movement, and allows host processors to focus on application control and higher-level analysis.

This paper presents BASE (*Byte-level Analytics and Statistical Engine*), a reusable streaming preprocessing accelerator for heterogeneous FPGA systems. BASE integrates directly into the AXI DMA datapath and performs single-pass statistical feature extraction, including byte-frequency histogram generation, entropy estimation, variance computation, and byte-transition analysis, entirely in programmable logic. By transforming preprocessing from a CPU-bound software stage into an inline hardware service, BASE overlaps communication with computation and significantly reduces host-side preprocessing overhead while preserving compatibility with existing heterogeneous execution models.

To evaluate the proposed architecture, we use deployment-time FPGA bitstream screening as a representative data-intensive workload. Modern cloud and HPC infrastructures increasingly deploy FPGAs as shared accelerators whose configuration bitstreams are dynamically transferred and programmed at runtime. Existing deployment-time screening workflows rely heavily on CPU-based preprocessing, where statistical feature extraction dominates the overall execution

time [20]–[22]. This workload therefore provides an ideal case study for evaluating the effectiveness of inline streaming preprocessing. Although deployment-time FPGA screening motivates our evaluation, the proposed architecture is independent of the application domain and is equally applicable to other binary-stream analytics workloads that require statistical preprocessing.

We implement BASE on heterogeneous ARM+FPGA platforms using Vivado HLS and integrate the accelerator into the AXI DMA communication path. We demonstrate the proposed architecture using deployment-time FPGA bitstream screening as a representative streaming workload. On the ZCU104, BASE reduces statistical preprocessing latency by $5.04\times$ and improves end-to-end execution time by $4.23\times$ through inline preprocessing and communication overlap. Our evaluation of BASE on the U55C demonstrates that these relative gains are platform-dependent: on that system, BASE improves preprocessing latency by $1.32\times$ and end-to-end latency by $1.78\times$. These results demonstrate that integrating reusable streaming preprocessing directly into the FPGA communication path significantly improves heterogeneous system performance while maintaining modest hardware resource requirements. Although evaluated using deployment-time FPGA screening, BASE is designed as a reusable preprocessing primitive that can support a broad class of streaming analytics applications, including telemetry processing, packet inspection, binary-stream analysis, and AI preprocessing.

The primary contributions of this work are as follows:

- We present **BASE**, a reusable FPGA-based streaming preprocessing primitive that computes multiple byte-level statistical features in a single pass directly within the AXI DMA communication pipeline.
- We propose a heterogeneous hardware/software architecture that overlaps communication with preprocessing, reducing memory traffic and host-side computation while improving throughput for streaming applications.
- We provide a comprehensive implementation and evaluation on heterogeneous ARM+FPGA platforms, demonstrating significant reductions in preprocessing latency and end-to-end execution time through inline hardware preprocessing.

II. RELATED WORK

The increasing adoption of heterogeneous FPGA platforms has motivated extensive research in accelerator architectures, streaming dataflow systems, FPGA security, and deployment-time analysis [1]–[3]. While prior work has demonstrated significant advances in computation acceleration and machine-learning-based FPGA security [20], [21], [23], [24], comparatively little attention has been given to preprocessing as a reusable hardware service. Our work bridges this gap by integrating statistical preprocessing directly into the FPGA communication pipeline, allowing preprocessing and data movement to execute concurrently.

A. FPGA Acceleration in Heterogeneous HPC Systems

Field-programmable gate arrays (FPGAs) have become an important component of heterogeneous HPC systems because they provide customizable datapaths capable of improving performance and energy efficiency for data-intensive applications [3], [4]. Cloud providers and HPC centers increasingly deploy FPGAs alongside CPUs and GPUs to accelerate machine learning, networking, database processing, scientific computing, and real-time analytics [25], [26]. Unlike fixed-function accelerators, reconfigurable hardware enables application-specific pipelines that exploit fine-grained parallelism while reducing unnecessary data movement [19]. Recent research has therefore focused on improving runtime programmability, communication efficiency, and resource utilization in heterogeneous FPGA systems [27].

Despite these advances, many FPGA accelerators continue to treat preprocessing as a host-side software operation [25], [26], [28]–[32]. Consequently, applications often require multiple transfers between memory and the accelerator before computation begins, increasing latency and limiting sustained throughput.

B. Streaming and Dataflow Architectures

While existing streaming accelerators focus primarily on application-specific kernels [33]–[35], relatively little work has explored reusable preprocessing primitives that can support multiple application domains. In contrast, BASE performs application-independent statistical preprocessing directly within the AXI DMA communication path, enabling the same hardware architecture to support diverse streaming workloads.

C. Machine Learning for FPGA Bitstream Analysis

Machine learning has recently emerged as an effective technique for detecting malicious FPGA configurations without requiring access to register-transfer level (RTL) descriptions or complete reverse engineering of configuration bitstreams. Prior work has investigated convolutional neural networks, Random Forests, support vector machines, and other learning algorithms for identifying malicious configurations and classifying hardware Trojans [20], [23], [24], [36].

These studies primarily emphasize classifier accuracy while assuming that feature extraction has already been completed. As a result, preprocessing remains a software bottleneck that dominates execution time in deployment-time workflows. Our work complements these approaches by accelerating the statistical preprocessing stage rather than modifying the inference algorithm itself.

D. Deployment-Time FPGA Screening and Validation

Commercial FPGA platforms employ cryptographic mechanisms, such as AES decryption and digital signature verification, to ensure the integrity and authenticity of configuration bitstreams before device programming. However, these mechanisms cannot detect malicious functionality embedded within authenticated bitstreams [27], [37]–[40]. Alternative

approaches based on reverse engineering, static analysis, and formal verification provide deeper inspection but incur substantial computational overhead, making them impractical for deployment-time execution in cloud and HPC environments [39], [40].

Recent machine-learning-based approaches address this challenge by classifying bitstreams using statistical features extracted from the raw binary stream [20]. However, prior work showed that software-based statistical preprocessing accounts for over 90% of the end-to-end execution time, creating a *preprocessing wall* that limits deployment throughput and underutilizes heterogeneous FPGA resources. These observations motivate efficient deployment-time architectures that perform statistical preprocessing directly within the heterogeneous execution pipeline rather than as a CPU-bound software stage.

As described above, existing work generally emphasizes one of three objectives: accelerating application-specific computation, improving machine-learning accuracy, or enhancing FPGA security. In contrast, BASE addresses a different systems problem by treating preprocessing as a reusable streaming service integrated directly into the FPGA communication path. Rather than introducing a new detection algorithm, the proposed architecture reduces preprocessing overhead, overlaps communication with computation, and provides a general-purpose statistical preprocessing primitive applicable to a broad range of streaming analytics workloads.

III. BASE: A REUSABLE STREAMING PREPROCESSING ARCHITECTURE

The central premise of this work is that preprocessing, rather than machine learning inference, is the dominant performance bottleneck for many streaming binary analytics applications. Profiling the software implementation of our deployment-time FPGA screening workflow revealed that statistical feature extraction accounts for more than 90% of the total execution time [20]–[22], while inference contributes less than 10%. These observations indicate that improving inference alone provides limited system-level benefit. Instead, reducing preprocessing overhead and minimizing data movement offer substantially greater opportunities for performance improvement.

To address this challenge, we propose **BASE (Byte-level Analytics and Statistical Engine)**, a reusable streaming preprocessing architecture implemented entirely in FPGA programmable logic. Rather than executing preprocessing as a separate software stage after data transfer, BASE performs statistical feature extraction directly within the AXI DMA communication path. Incoming data are analyzed as they stream through the accelerator, allowing preprocessing and communication to proceed concurrently. This transforms preprocessing from a CPU-bound software task into an inline hardware service that can be reused across multiple streaming applications.

A. System Overview

Figure 1 illustrates the proposed heterogeneous architecture. The system consists of two cooperating execution domains.

- **Processing System (PS):** responsible for application control, DMA management, runtime scheduling, machine learning inference, and deployment decisions.
- **Programmable Logic (PL):** responsible for streaming statistical preprocessing performed by BASE.

Binary streams are first staged in external memory before being transferred through the AXI DMA engine into programmable logic. Instead of forwarding the stream directly to software, the data first pass through BASE, where statistical descriptors are extracted in a single streaming pass. The resulting feature vector is written back to memory, after which the processing system performs lightweight application-specific analysis.

This organization enables communication and computation to execute concurrently while significantly reducing CPU preprocessing overhead.

B. Streaming Computation Model

Consider an input byte stream

$$S = \{b_1, b_2, \dots, b_N\},$$

where each byte b_i is processed exactly once.

BASE computes a statistical feature vector

$$F(S) = \{H(S), E(S), \mu(S), \sigma^2(S), T(S)\},$$

where

- $H(S)$ is the byte-frequency histogram,
- $E(S)$ is the estimated Shannon entropy,
- $\mu(S)$ is the stream mean,
- $\sigma^2(S)$ is the stream variance,
- $T(S)$ is the byte-transition statistic.

Unlike our software baseline, which computes these quantities using multiple passes over memory via separate library calls, BASE updates all statistics simultaneously during a single traversal of the input stream.

C. Hardware Architecture

BASE is organized as a collection of independent streaming modules connected through a fully pipelined dataflow architecture.

Each incoming byte is broadcast to four processing engines that execute concurrently.

1) *Histogram Engine:* A BRAM-based histogram maintains 256 counters corresponding to every possible byte value. Each incoming byte increments its corresponding counter without interrupting the streaming pipeline.

2) *Streaming Statistics Engine:* Running accumulators implemented using FPGA DSP resources compute the first and second statistical moments required for estimating the stream mean and variance.

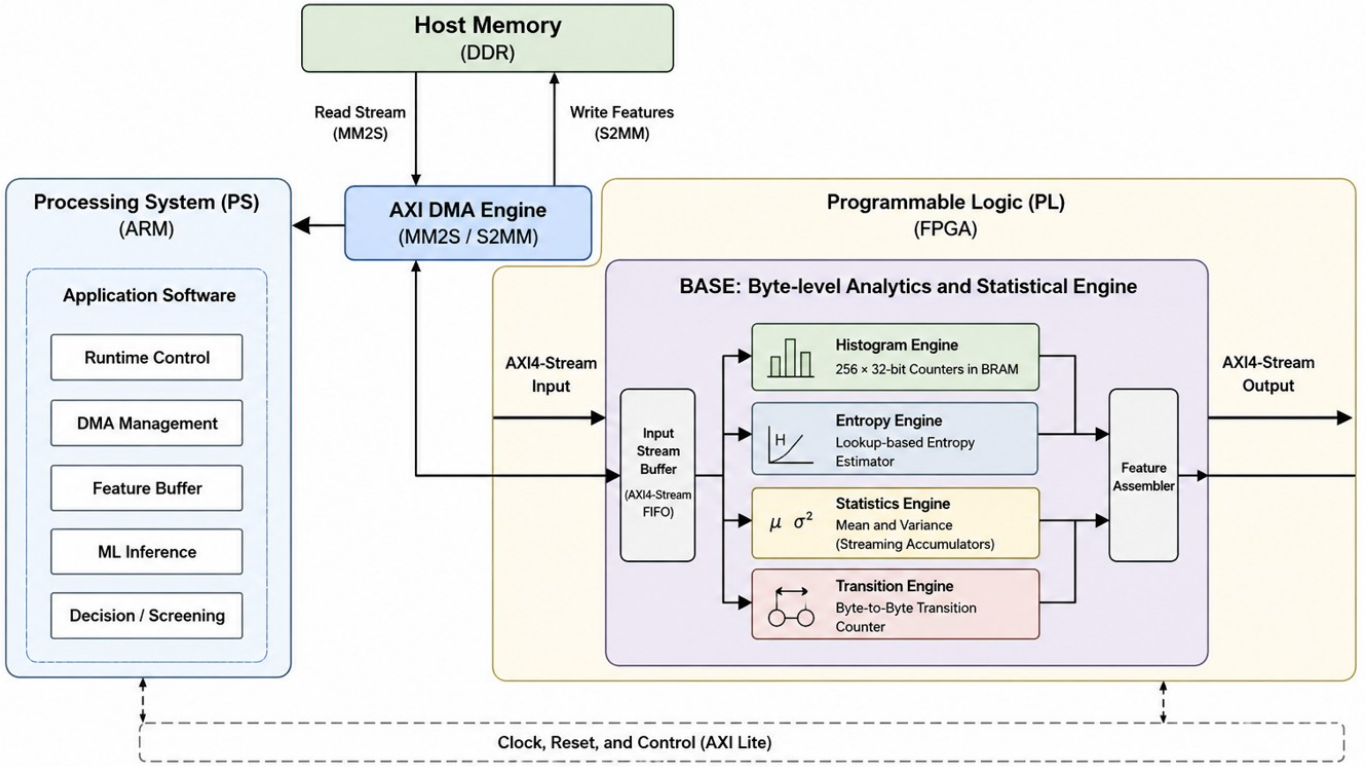


Fig. 1. Overview of the proposed BASE architecture for heterogeneous FPGA systems. Binary data streams are transferred from host memory to the FPGA through the AXI DMA engine, where the Byte-level Analytics and Statistical Engine (BASE) performs inline statistical preprocessing in a single streaming pass. The architecture concurrently computes byte-frequency histograms, entropy, statistical moments, and byte-transition features while data are in transit, overlapping communication with computation. The resulting feature vector is written back to host memory for application-specific analysis, demonstrating a reusable streaming preprocessing primitive for heterogeneous HPC and data-intensive workloads.

3) *Entropy Engine*: Entropy is estimated using a lookup-table approximation that eliminates expensive floating-point logarithmic operations while preserving high throughput.

4) *Transition Engine*: A lightweight comparator monitors the transitions between consecutive bytes, providing structural information that complements histogram-based statistics.

D. Pipeline Execution

Figure 2 compares the execution timeline of a conventional software implementation with the proposed BASE architecture. In the software pipeline, memory access, statistical preprocessing, and inference execute sequentially, causing preprocessing to dominate the end-to-end runtime. In contrast, BASE performs statistical preprocessing concurrently with AXI DMA transfers, allowing communication and computation to overlap and significantly reducing overall execution time.

This streaming execution model provides three important advantages:

- 1) Single-pass statistical preprocessing.
- 2) Reduced external memory traffic.
- 3) Overlap between communication and computation.

Consequently, preprocessing latency is largely hidden behind DMA transfer, improving sustained throughput for heterogeneous streaming applications.

E. Computational Complexity

We characterize complexity relative to our software baseline, a NumPy-based implementation that computes the histogram, entropy, statistical moments, and byte-transition features as separate library calls. Each call independently traverses the input stream, so the baseline's observed preprocessing cost is proportional to

$$O(kN),$$

where k denotes the number of independent preprocessing stages invoked. We note that this scaling is a property of our baseline's implementation strategy rather than a fundamental limitation of software: a custom, hand-fused software implementation could in principle compute these statistics in a single pass and achieve the same $O(N)$ asymptotic complexity as BASE. However, standard vectorized libraries such as NumPy do not expose a fused, single-pass API for this combination of statistics, and reimplementing such a routine in optimized native code was outside the scope of the baseline used for comparison in this work.

In contrast, BASE computes all statistical descriptors concurrently during a single streaming pass over the AXI DMA data path, achieving

$$O(N)$$

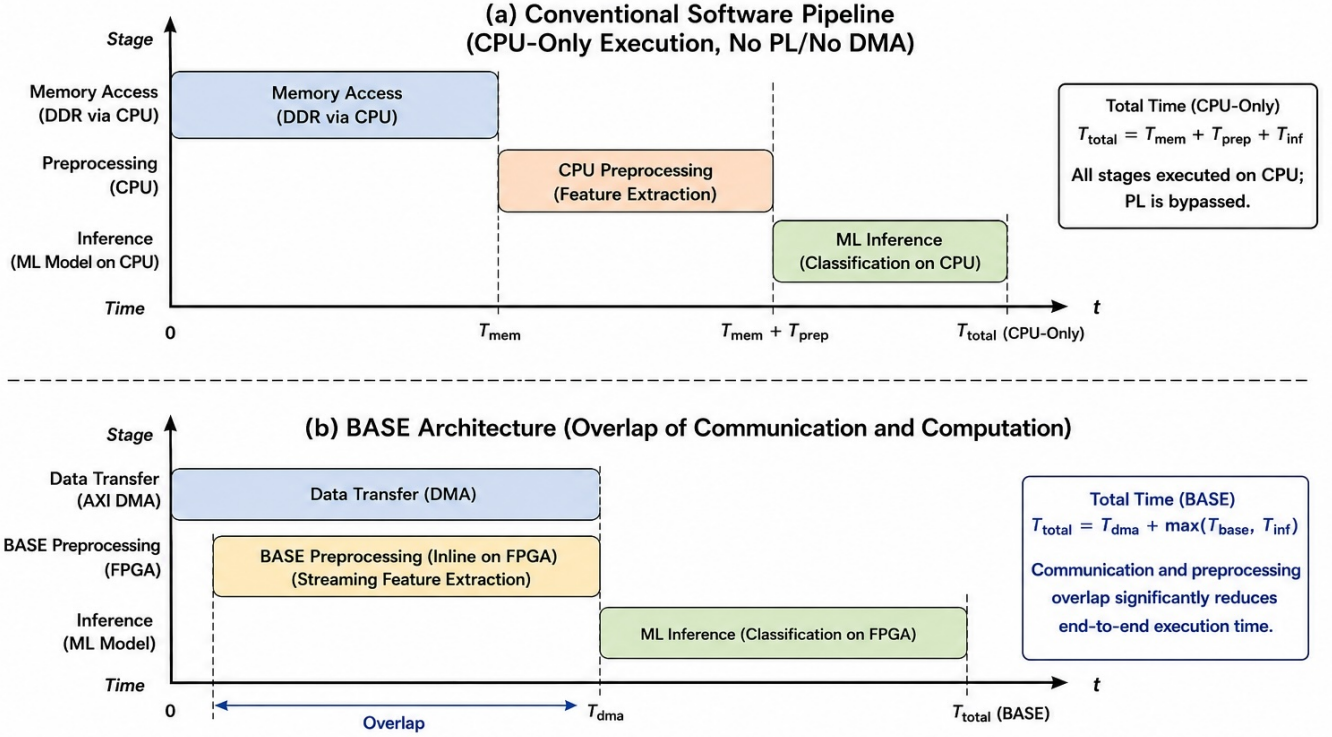


Fig. 2. Pipeline execution comparison between a conventional software implementation and the proposed BASE architecture. (a) In the software pipeline, DMA transfer, CPU-based statistical preprocessing, and machine-learning inference execute sequentially, resulting in preprocessing dominating the end-to-end runtime. (b) In the proposed BASE architecture, statistical preprocessing is integrated directly into the AXI DMA communication path, enabling communication and computation to overlap. This streaming execution model significantly reduces preprocessing overhead, minimizes memory traffic, and shortens the overall execution time.

complexity while eliminating redundant memory accesses.

Although an optimally fused single-pass software implementation has equivalent asymptotic complexity, BASE additionally exploits hardware parallelism, overlaps computation with DMA transfers, and reduces reliance on the CPU–memory hierarchy.

F. Latency and Throughput Scaling with Input Size

To empirically validate the $O(N)$ scaling behavior described in Section III-E and to characterize how the performance gap between the software baseline and BASE evolves with bitstream size, we measure feature-extraction latency and throughput across a range of input sizes. The bitstreams were binned by file size/ generated at fixed sizes based on the FPGA model (the current data set only targets PYNQ-Z1 and spans 6 MB per file). For this reason, we have constructed a controlled microbenchmark using byte streams of varying length ([64 KB to 8 MB], generated by truncating representative bitstream files from our dataset. The benchmark measures feature-extraction latency and throughput at each size. This benchmark is independent of the classifier-facing evaluation and is intended solely to characterize the computational scaling behavior of the two implementations.

TABLE I
FEATURE EXTRACTION LATENCY AND THROUGHPUT VS. INPUT SIZE (ZCU104)

Input Size	CPU (ms)	BASE (ms)	Speedup	BASE (MB/s)
64KB	56.07	18.99	2.95x	3.29
256KB	204.64	49.89	4.10x	5.01
1.00MB	799.57	175.06	4.57x	5.71
2.00MB	1697.71	341.39	4.97x	5.86
4.00MB	3554.26	678.03	5.24x	5.90
6.00MB	5329.78	1011.62	5.27x	5.93
8.00MB	7167.72	1345.59	5.33x	5.95

G. Implementation

BASE was implemented using Vivado HLS and integrated into heterogeneous ARM+FPGA systems through AXI DMA and AXI4-Stream interfaces. The accelerator targets a 100MHz operating frequency and was synthesized, implemented, and validated using RTL co-simulation.

The modular architecture allows individual preprocessing units to be extended or replaced without modifying the surrounding communication infrastructure, enabling reuse across diverse streaming applications.

H. Experimental Setup

The experiments were conducted on heterogeneous ARM+FPGA platforms using Vivado 2018.3 (PYNQ-Z1 &

ZCU104) and Vitis 2025.1 (U55C). The processing system executes application control and machine-learning inference, while BASE performs statistical preprocessing within programmable logic. The extracted feature vector is consumed by a downstream hybrid CNN + Random Forest classifier (BLADEI [20]). The deployment-time FPGA bitstream screening is used as a representative streaming workload because it combines large binary streams with statistical preprocessing and lightweight inference.

I. Representative Workloads

The evaluation dataset is constructed from Trust-Hub FPGA Trojan benchmarks [41] and re-engineered for deployment on Xilinx FPGA platforms using Vivado v2023.2. The dataset spans multiple benchmark families including cryptographic accelerators, communication systems, and processors, as well as ISCAS and ITC benchmark circuits. Both benign and Trojan-inserted variants are included.

To improve structural diversity and avoid overfitting to implementation artifacts, we generate multiple place-and-route variants for each design using different synthesis directives and implementation seeds. This augmentation process produces structurally distinct bitstreams while preserving functional equivalence. Table II summarizes the dataset composition mentioned in Section III-F.

TABLE II
BENCHMARK COMPOSITION OF THE EVALUATION DATASET

Design Family	Benchmarks	Bitstreams
CRYPTO	29	290
COMMS	33	330
MCU / CPU	4	40
BUS / DISPLAY	4	40
ITC99	29	289
ISCAS89	10	100
ISCAS85	294	294
Total	403	1383

J. Runtime Profiling

To identify the dominant performance bottleneck, we first profile the complete software implementation before hardware acceleration. The software workflow is decomposed into three stages:

- 1) Bitstream loading
- 2) Statistical preprocessing
- 3) Machine-learning inference

Execution time for each pipeline stage was measured over three independent runs for each benchmark type in the BLADEI dataset suite. Stage latencies were obtained by recording the elapsed wall-clock time on the ARM processing system during execution. Table III reports the average execution time and standard deviation for the ZCU104.

As shown in Fig. 3, statistical preprocessing dominates the software execution pipeline, accounting for more than 90% of the measured runtime, whereas machine-learning inference contributes less than 10%. Although the individual statistical

TABLE III
RUNTIME PROFILING OF THE SOFTWARE BASELINE (ZCU104)

Stage	Latency (ms)	Std. Dev.	Contribution
Bitstream Loading	58.97	79.08	1.61%
Statistical Preprocessing	3322.01	26.89	90.68%
ML Inference	282.64	43.25	7.71%
Total	3663.62	149.22	100.00%

operations are computationally simple, feature extraction requires repeated traversal of the input bitstream to compute byte-frequency histograms, entropy, statistical moments, and transition statistics. These repeated memory accesses create a software preprocessing bottleneck that limits deployment throughput.

This profiling demonstrates that optimizing inference alone would have little impact on overall system performance. Instead, the largest opportunity for acceleration lies in moving statistical preprocessing into programmable logic and overlapping feature extraction with DMA communication. This observation directly motivates the proposed BASE architecture.

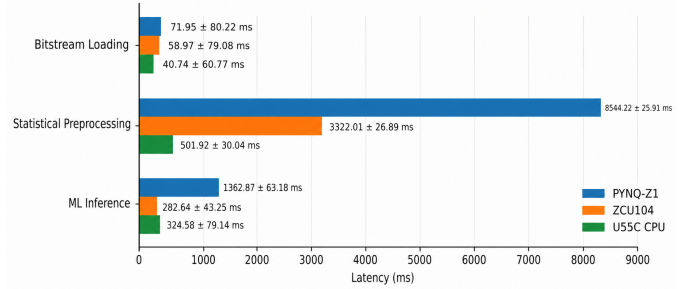


Fig. 3. Runtime breakdown of the software baseline. Statistical preprocessing dominates end-to-end latency, motivating its migration into programmable logic.

K. End-to-End Performance

We next compare the software implementation with the proposed BASE architecture. The software implementation executes preprocessing entirely on the ARM processor, whereas BASE performs statistical feature extraction inline during DMA transfers.

The deployed heterogeneous architecture partitions execution between:

- **Programmable Logic (PL):** streaming statistical preprocessing,
- **Processing System (PS):** loading and ML inference.

We evaluate the complete system using both software-only and hardware-accelerated configurations on the ZCU104. Table IV summarizes the results. The cross-platform evaluation in Section III-M is reported separately and is the only other quantitative platform comparison in the paper.

Beyond the raw latency and throughput gains, Table IV highlights two structural differences between the two implementations: BASE requires only a single pass over the input stream, and it overlaps statistical preprocessing with DMA

TABLE IV
END-TO-END SYSTEM PERFORMANCE (ZCU104)

Metric	CPU	BASE	Improvement
Feature Extraction (ms)	3371.94	668.66	5.04×
Total Runtime (s)	3.55	0.84	4.23×
End-to-end Throughput (MB/s)	1.09	4.58	4.20×
Memory Passes	Multiple	Single	–
Communication Overlap	No	Yes	–
CPU Time (core-s)	4.020	1.330	3.02×

communication rather than executing the two sequentially. These structural changes are the intended source of BASE’s preprocessing and data-movement benefit. The separate third-board experiment also shows that end-to-end performance can be influenced by platform-dependent differences in the host-side prediction stage, so the relative contribution of each pipeline stage should not be assumed to be invariant across platforms. The end-to-end throughput reported here includes the complete application pipeline and therefore differs from the preprocessing-only throughput reported in Table I.

L. HLS Synthesis and Resource Utilization

BASE is implemented and integrated into the heterogeneous ARM+FPGA system through AXI DMA and AXI4-Stream interfaces. We report synthesis timing, resource utilization, and pipeline characteristics for the generated hardware. Table V summarizes the HLS synthesis results for the ZCU104 implementation.

TABLE V
BASE HLS SYNTHESIS RESULTS (ZCU104)

Metric	Value	Utilization
Target Frequency	100 MHz (constraint)	–
Estimated Clock Period	9.16 ns (HLS estimate)	–
Initiation Interval (II)	8 (<code>process_words</code> loop)	–
BRAM	16 (HLS estimate)	2%
DSP	18 (HLS estimate)	1%
LUTs	52,927 (HLS estimate)	22%
Flip-Flops	18,594 (HLS estimate)	4%
RTL Co-Simulation	Pass (Verilog)	–

The synthesized design meets timing at 100 MHz while requiring modest FPGA resources. Resource utilization indicates that the proposed architecture occupies only a fraction of the available programmable logic, leaving substantial capacity for application-specific accelerators.

M. Cross-Platform Evaluation

To evaluate scalability across heterogeneous FPGA systems, we deploy BASE across multiple FPGA platforms spanning edge, embedded, and datacenter-class devices.

- **PYNQ-Z1**: edge-class Zynq-7000 platform
- **ZCU104**: Zynq UltraScale+ embedded platform
- **U55C (Alveo)**: Datacenter-class HBM accelerator

The U55C evaluation uses its onboard High Bandwidth Memory (HBM) subsystem to examine BASE within an ultra-fast environment representative of datacenter-class FPGA deployments. This evaluation characterizes BASE within a

datacenter-class FPGA environment and examines how its performance changes relative to embedded platforms. Table VI summarizes the cross-platform evaluation of BASE.

TABLE VI
CROSS-PLATFORM EVALUATION

Metric	PYNQ-Z1	ZCU104	U55C
Clock Frequency	100 MHz	100 MHz	100 MHz
CPU Feature Extraction Latency	8869.52 ms	3371.94 ms	501.92 ms
BASE Feature Extraction Latency	1898.52 ms	668.66 ms	378.89 ms
BASE Preprocessing Throughput	2.03 MB/s	4.58 MB/s	10.18 MB/s
CPU→BASE Preprocessing Speedup	4.61×	5.04×	1.32×
CPU Total Latency	10.25 s	3.55 s	0.87 s
BASE Total Latency	3.73 s	0.84 s	0.49 s
BASE End-to-end Throughput	1.03 MB/s	4.60 MB/s	7.93 MB/s
CPU→BASE End-to-End Speedup	2.75×	4.23×	1.78×
Total On-Chip Power Consumption	1.757 W	3.568 W	17.561 W

IV. CONCLUSION

This work presented **BASE (Byte-level Analytics and Statistical Engine)**, a reusable streaming preprocessing accelerator that integrates byte-level statistical feature extraction directly into the FPGA communication path. Profiling a CPU-based deployment-time FPGA screening workflow revealed that statistical preprocessing, rather than machine-learning inference, dominates end-to-end runtime, accounting for over 90% of total execution time and creating a significant bottleneck for heterogeneous deployment pipelines.

To address this limitation, BASE moves histogram generation, entropy estimation, statistical moment computation, and byte-transition analysis into FPGA programmable logic and embeds this feature extraction directly within the AXI DMA data path. By overlapping preprocessing with data movement, BASE transforms deployment-time screening from a sequential, CPU-bound software stage into a single-pass streaming hardware operation embedded within the heterogeneous execution pipeline.

We implemented and evaluated BASE using the BLADEI screening pipeline. On the ZCU104, feature-extraction latency was reduced from 3371.94 ms to 668.66 ms and total end-to-end runtime was reduced from 3.55 s to 0.84 s, while throughput improved from 1.09 MB/s to 4.58 MB/s, corresponding to a 5.04× reduction in preprocessing latency and a 4.23× improvement in end-to-end execution time. On the U55C, using its HBM subsystem, feature-extraction latency was reduced from 501.92 ms on the CPU to 378.89 ms with BASE, while total end-to-end runtime was reduced from 0.87 s to 0.49 s, corresponding to a 1.32× reduction in preprocessing latency and a 1.78× improvement in end-to-end execution time. Together, these results demonstrate that BASE provides substantial performance benefits across heterogeneous FPGA platforms, while its relative advantage depends on the host processor, memory subsystem, and accelerator balance of the target platform.

Although this paper evaluates BASE in the context of deployment-time FPGA bitstream screening, the underlying architecture performs application-independent statistical preprocessing over raw byte streams and is not specific to bit-

stream analysis. Because BASE operates directly on the AXI4-Stream interface without assumptions about the semantics of the incoming data, the same reusable hardware primitive is directly applicable to a broader class of streaming analytics and data-centric HPC workloads, including network telemetry analysis, packet inspection, binary malware and firmware analysis, storage-level anomaly detection, and preprocessing for AI/ML training pipelines. In each of these domains, byte-level statistical descriptors such as entropy, byte-frequency distribution, and transition structure serve as common front-end features, and BASE's ability to compute them inline and in a single pass makes it a candidate building block for a wide range of reconfigurable dataflow systems, rather than a workload-specific accelerator.

V. FUTURE WORK

Future work will focus on extending the evaluation of BASE across a broader range of heterogeneous FPGA platforms and streaming workloads. Although the current evaluation includes both the ZCU104 and the U55C, further experiments are needed to investigate how platform characteristics, including memory bandwidth, DMA configuration, and accelerator resources, affect BASE's scalability in datacenter-class environments equipped with high-bandwidth memory (HBM). We also plan to evaluate BASE using large-scale streaming telemetry datasets, including CICIDS2017 and MAWI, to characterize the architecture as a general-purpose streaming preprocessing engine beyond FPGA bitstream screening. Finally, we will investigate additional lightweight statistical and structural features together with tighter hardware/software co-design to further establish BASE as a reusable building block for real-time analytics and security monitoring in heterogeneous cloud and HPC infrastructures.

A natural next step for this work is to further characterize BASE on datacenter-class accelerator platforms. The initial U55C evaluation demonstrates the feasibility of deploying BASE on an HBM-equipped accelerator card, but additional experiments are needed to determine how effectively the architecture can utilize HBM bandwidth at larger streaming data rates. In particular, future work will investigate the relationship between HBM memory bandwidth, DMA throughput, and BASE's internal processing pipeline, including whether partitioning internal state such as histogram storage and accumulator banks across HBM pseudo-channels can improve sustained throughput at datacenter scale. These experiments would further clarify the extent to which the benefits of inline preprocessing observed on the ZCU104 and U55C scale with available memory bandwidth and system resources.

Beyond its role as a general preprocessing primitive, we view the U55C-based deployment of BASE as a step toward a practical, line-rate cybersecurity tool for HPC datacenters. Because BASE computes statistical descriptors that are already used as front-end features for anomaly and intrusion detection, malware classification, and hardware Trojan screening, an HBM-backed, datacenter-class implementation provides a path toward continuously profiling high-volume data flows, such

as inter-node traffic, storage I/O, or dynamically deployed accelerator bitstreams. This positions BASE not only as a data-movement optimization but as infrastructure for real-time, in-line security monitoring in shared, multi-tenant FPGA and HPC environments.

Future work will therefore proceed along two directions: (1) evaluating the resulting system on larger and more diverse streaming workloads, including network telemetry traces such as CICIDS2017 and MAWI, to characterize BASE as a general-purpose streaming preprocessing engine beyond FPGA bitstream screening, and (2) extending BASE with additional lightweight in-line statistical or structural features that would further strengthen its utility as a real-time security-monitoring primitive for heterogeneous HPC datacenters.

VI. ACKNOWLEDGMENT

This work was supported by the Office of Undergraduate Research and McNair Junior Fellowship at the University of South Carolina.

OpenAI's ChatGPT and Anthropic's Claude were used during the preparation of this work to improve the readability of the text and to assist in the formatting of the figures. After using the tool, the authors reviewed and edited the content to ensure accuracy and maintain full responsibility for the final version.

REFERENCES

- [1] S. Gathu *et al.*, "High-performance computing and big data: Emerging trends in advanced computing systems for data-intensive applications," *Journal of Advanced Computing Systems*, vol. 4, no. 8, pp. 22–35, 2024.
- [2] P. O. A. Navaux, A. F. Lorenzon, and M. d. S. Serpa, "Challenges in high-performance computing," *Journal of the Brazilian Computer Society*, vol. 29, no. 1, pp. 51–62, 2023.
- [3] S. Usman, R. Mehmood, I. Katib, and A. Albeshri, "Data locality in high performance computing, big data, and converged systems: An analysis of the cutting edge and a future system architecture," *Electronics*, vol. 12, no. 1, p. 53, 2022. [Online]. Available: <https://www.mdpi.com/2079-9292/12/1/53>
- [4] C. Silvano, D. Ielmini, F. Ferrandi, L. Fiorin, S. Curzel, L. Benini, F. Conti, A. Garofalo, C. Zambelli, E. Calore *et al.*, "A survey on deep learning hardware accelerators for heterogeneous hpc platforms," *ACM Computing Surveys*, vol. 57, no. 11, pp. 1–39, 2025.
- [5] G. Singh, "Designing, modeling, and optimizing data-intensive computing systems," *arXiv preprint arXiv:2208.08886*, 2022.
- [6] H. Su *et al.*, "Explainable hardware trojan localization at lut level," *IEEE TCAD*, 2025.
- [7] P. Gonzalez-Guerrero, A. Butko, G. Micheliogianniakis, and J. Shalf, "Ai-enabled analysis and control for enhancing data transition and movement," in *Position Papers for the ASCR Workshop on Reimagining Codesign*, 2021, pp. 1–7.
- [8] R. Abu Bakar, "Data processing units for next-generation networks: Evaluation across user plane function, high-performance computing, and cybersecurity workload," 2026.
- [9] L. Pereira, C. Bentes, M. C. S. de Castro, and E. Garcia, "A case study of performance optimization in a heterogeneous environment," in *2017 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*. IEEE, 2017, pp. 13–18.
- [10] C. Blair, N. M. Robertson, and D. Hume, "Characterizing a heterogeneous system for person detection in video using histograms of oriented gradients: Power versus speed versus accuracy," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 2, pp. 236–247, 2013.

- [11] P. Naayini and S. Kamatala, "High-performance data computing: Parallel frameworks, execution strategies, and real-world deployments," *International Journal of Scientific Advances (IJSCIA)*, vol. 4, no. 6, pp. 1056–1064, 2023.
- [12] A. Chinnaraju, "Real time adaptive ai pipelines for edge cloud systems: Dynamic optimization based on infrastructure feedback," *World Journal of Advanced Engineering Technology and Sciences*, vol. 13, no. 2, pp. 887–908, 2024.
- [13] G. K. Kulatilleke, S. Layeghy, M. Baktashmotlagh, and M. Portmann, "Mambanetburst: Direct byte-level network traffic classification without tokenization or pretraining," *arXiv preprint arXiv:2605.11034*, 2026.
- [14] A. Orojo, E. El-Mahmoud, E. Leal, and P. Rivas, "Byteflow: A byte-level llm for deep packet inspection and network intelligence," in *2025 Annual Computer Security Applications Conference Workshops (ACSAC Workshops)*. IEEE, 2025, pp. 357–365.
- [15] X. Li, "Fpga security techniques with applications to cloud and multi-tenant use cases," 2023.
- [16] A. De Toro-Castro, M. Lupión, V. González-Ruiz, J. F. Sanjuan, and P. M. Ortigosa, "A parallel framework for data input pipelines and online data augmentation in deep learning," *The Journal of Supercomputing*, vol. 82, no. 7, p. 389, 2026. [Online]. Available: <https://link.springer.com/article/10.1007/s11227-026-08543-0>
- [17] N. Sharma, R. Sharma, and N. Jindal, "Machine learning and deep learning applications-a vision," *Global Transitions Proceedings*, vol. 2, no. 1, pp. 24–28, 2021. [Online]. Available: <https://doi.org/10.1016/j.gltp.2021.01.004>
- [18] D. Nichols, S. Singh, S.-H. Lin, and A. Bhatele, "A survey and empirical evaluation of parallel deep learning frameworks," *arXiv preprint arXiv:2111.04949*, 2021. [Online]. Available: <https://arxiv.org>
- [19] D. C. Cronquist, C. Fisher, M. Figueroa, P. Franklin, and C. Ebeling, "Architecture design of reconfigurable pipelined datapaths," in *Proceedings 20th Anniversary Conference on Advanced Research in VLSI*. IEEE, 1999, pp. 23–40.
- [20] R. Stahle-Smith and R. Karakchi, "Real-Time ML-Based Defense Against Malicious Payload in Reconfigurable Embedded Systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC25)*, 2025.
- [21] R. Stahle-Smith *et al.*, "Supervised machine learning for real-time Hardware Trojan detection in FPGA bitstreams via byte frequency analysis," *Journal of Cryptographic Engineering / MDPI Research*, 2024.
- [22] R. Stahle-Smith, C. Antley, J. D. Bakos, and R. Karakchi, "Hardware-accelerated line-rate bitstream screening for secure fpga reconfiguration," 2026. [Online]. Available: <https://arxiv.org/abs/2605.08984>
- [23] M. Oya, T. Yanajisawa, and N. Masuda, "Detection and classification of malicious bitstreams for FPGAs using CNN-based static feature mining," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2023, pp. 112–118.
- [24] D. D'Agostino, P. Destro, and A. Muccio, "Is your FPGA bitstream Hardware Trojan-free? Machine learning-based multi-parametric approaches," *Journal of Systems Architecture*, vol. 131, p. 102692, 2022.
- [25] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, M. Papamichael, M. Weil, G. Weisz, L. Woods *et al.*, "A cloud-scale acceleration architecture," in *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–13.
- [26] L. Roostmeh, J. Lavian, and S. A. Mahlke, "Accelerating database systems using FPGAs: A survey," *IEEE Micro*, vol. 38, no. 4, pp. 32–43, 2018.
- [27] D. Sidler, G. Alonso, and E. Z. Operating Systems Group, "Coyote: An operating system for multi-tenant FPGA cloud deployments," in *Proceedings of the 2021 USENIX Annual Technical Conference (ATC)*, 2021, pp. 325–340.
- [28] J. A. de Abreu-Faria, M. A. Figueroa-Martínez, and L. G. de la Fraga, "A survey on FPGA-based heterogeneous clusters architectures," *IEEE Access*, vol. 11, pp. 62 145–62 168, 2023.
- [29] J. V. Henderson, "Overcoming data movement bottlenecks in heterogeneous computing systems," Ph.D. dissertation, University of California, 2024.
- [30] A. S. Srivastava and R. K. Mitra, "High-performance statistical computing (HPSC) on heterogeneous pipelines," *arXiv preprint arXiv:2508.04013*, 2025.
- [31] Z. Yang, Z. István, M. Canini, and S. A. Fahmy, "Application-level function pipelines on in-network fpga accelerators," in *2026 IFIP Networking Conference (IFIP Networking)*. IEEE, 2026, pp. 1–10.
- [32] O. Jaiyeoba, A. T. Mughrabi, M. Baradaran, B. Gul, and K. Skadron, "Swift: A multi-fpga framework for scaling up accelerated graph analytics," in *2024 International Conference on Field Programmable Technology (ICFPT)*. IEEE, 2024, pp. 1–10.
- [33] T. Zhang, S. Wang, J. Liu, X. Chen, and Y. Zhou, "Accelerating recommender model ETL with a streaming FPGA-GPU dataflow," *arXiv preprint arXiv:2501.12032*, 2025.
- [34] N. Kaufmann, T. Schneider, and O. Bringmann, "Reconfigurable Smart-NICs: A comprehensive review of FPGA shells and heterogeneous offloading architectures," *MDPI Applied Sciences*, vol. 16, no. 3, p. 1476, 2026.
- [35] E. Melber, "Fluid: Raising the level of abstraction for FPGA accelerator design via service-oriented infrastructure," Ph.D. dissertation, Carnegie Mellon University, 2021, introduces hardware service abstractions at the module boundary to separate data movement/preprocessing from application-specific kernels.
- [36] A. Karakaya *et al.*, "Case of Hardware Trojan detection in FPGAs: RNN and LSTM architectures for bitstream verification," *MDPI Technologies / Cryptography*, vol. 8, no. 3, p. 36, 2024.
- [37] S. Skorobogatov, "Hardware cryptography in modern FPGAs: A double-edged sword for trust and security," in *Proceedings of the International Conference on Field-Programmable Technology (FPT)*. IEEE, 2019, pp. 45–52.
- [38] P. Swierczynski, M. Fyrbiak, P. Koppe, and C. Paar, "Security analysis of FPGA bitstream encryption and authentication mechanisms," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 11, no. 2, pp. 1–24, 2018.
- [39] B. Benchoff and I. G. Harris, "Evaluating the computational overheads of bitstream reverse engineering for malware detection," in *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2021, pp. 89–94.
- [40] A. Fedorov and D. Mukhopadhyay, "Formal verification vs. dynamic scanning: The scaling wall in deployment-time FPGA security," *Journal of Cryptographic Engineering*, vol. 10, no. 3, pp. 215–229, 2020.
- [41] C. Krieg, C. Wolf, and A. Jantsch, "Malicious lut: Stealthy fpga trojan," in *ICCAD*, 2016.