

Article

Hardware Accountability for Energy-Efficient Stream-Oriented Data-Plane Processing in 5G/6G Edge Telecommunication Nodes

Yurii Herman ^{1,*} , Oleh Krulikovskiy ^{1,2} , Dmytro Vovchuk ³  and Vjaceslavs Bobrovs ³ ¹ Department of Radioengineering and Information Security, Yuriy Fedkovych Chernivtsi National University, 58002 Chernivtsi, Ukraine; o.krulikovskiy@chnu.edu.ua² Department of Computers, Electronics and Automation, Stefan cel Mare University of Suceava, 720229 Suceava, Romania³ Institute of Photonics, Electronics and Telecommunications, Riga Technical University, LV-1048 Riga, Latvia; dmytro.vovchuk@rtu.lv (D.V.); vjaceslavs.bobrovs@rtu.lv (V.B.)

* Correspondence: herman.yurii@chnu.edu.ua

Abstract

Continuous stream-oriented data-plane processing in 5G/6G edge nodes increases the energy and latency cost of CPU-centered execution. This paper studies this boundary on an Intel Cyclone V SoC FPGA and proposes Hardware Accountability: a partitioning discipline in which Linux performs supervisory control while high-rate payload processing remains in programmable logic. The evaluation uses the Strumok stream cipher, adopted as the Ukrainian national standard DSTU 8845:2019, as a secure fronthaul/payload workload with XOR- and shift-dominated logic. On the evaluated USB 2.0/Cortex-A9/Linux path, the software-driven stream approaches saturation near 20 MSPS. In contrast, the RTL core reaches 9.6 Gbps at 150 MHz and occupies less than 6% of the available logic. Quartus Prime vectorless power analysis estimates 24.00 mW dynamic power for the RTL computational core, corresponding to approximately 2.5 pJ/bit. Control-plane measurements show P99 orchestration jitter below 1 ms under Spatial Isolation, conservative full context reloads near 1290 per second, and more than 7200 shadow-register context/state update operations per second. A design-space exploration then projects an 83.2 Gbps multi-core data path when external DDR traffic is avoided through internal stream aggregation and elastic buffering.

Keywords: 5G/6G edge systems; stream-oriented data-plane processing; secure fronthaul; SoC FPGA; hardware–software partitioning; control-plane/data-plane separation; MMIO orchestration; external memory wall; embedded telecommunications; green wireless infrastructure



Academic Editor: Christos J. Bouras

Received: 18 June 2026

Revised: 11 July 2026

Accepted: 21 July 2026

Published: 24 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](https://creativecommons.org/licenses/by/4.0/)[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The exponential growth of continuous data streams in 5G/6G edge access nodes introduces a significant challenge: ensuring energy-efficient data-plane processing while preserving the tight latency and energy constraints expected in next-generation edge telecommunications systems [1–3]. While extensive research focuses on active antenna arrays, radio frequency (RF) front-ends, and DSP-intensive physical-layer kernels, the energy dissipation associated with continuous fronthaul and payload processing remains a major system-level bottleneck at the edge node.

Traditional Software-Defined Radio (SDR) architectures typically rely on general-purpose processors (CPUs) for data stream processing [4,5]. Experimental stress testing of the evaluated ARM Cortex-A9-based edge node processing continuous quadrature (I/Q)

data reveals a clear architectural limitation in this approach. The software-driven execution model reaches a practical saturation threshold near 20 megasamples per second (MSPS). Figure 1 does not present a direct measurement trace; instead, it provides an analytical interpretation of the observed regime transition in terms of interrupt growth, scheduling overhead, and the finite processing budget of the CPU-centered path.

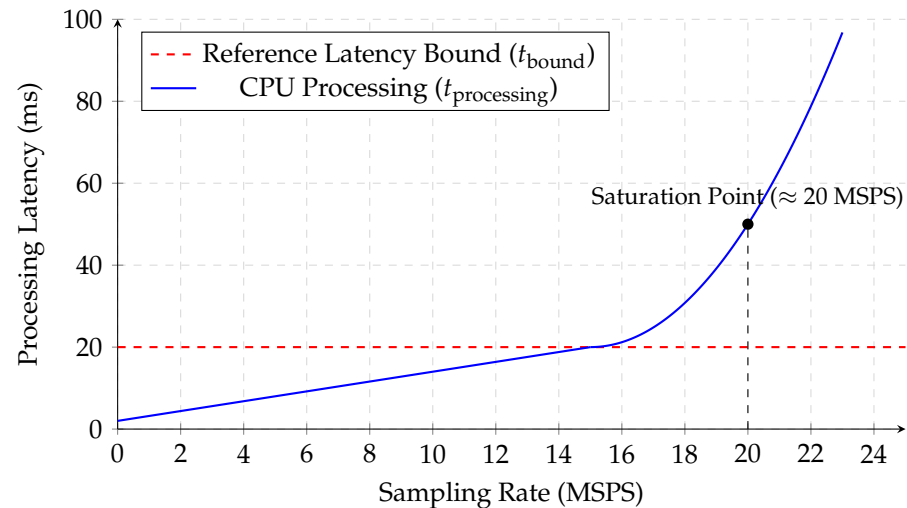


Figure 1. Analytical interpretation of the empirically observed software-path saturation regime. The schematic relates sampling-rate growth to interrupt and scheduling overhead on the evaluated USB 2.0/Cortex-A9 integration path and marks the practical saturation region near 20 MSPS.

These observations motivate the heterogeneous architectural transition studied here. To overcome this processing bottleneck and align with the principles of energy-proportional computing, this paper proposes a heterogeneous System-on-Chip (SoC) FPGA architecture grounded in a hardware–software partitioning strategy defined here as Hardware Accountability [6,7]. In this work, Hardware Accountability denotes a design discipline in which the programmable logic fabric carries the high-rate data path, while the processor and operating system remain responsible for supervisory control, configuration, context updates, and health monitoring. The term therefore does not claim that generic hardware/software partitioning is new; rather, it describes an explicitly measured boundary that assigns throughput, latency, and energy costs to the hardware or software resource that actually carries them.

We introduce an architectural dichotomy that functionally and spatially isolates the Control Plane from the Data Plane. The general-purpose operating system (Linux) is removed from the energy-intensive stream-processing path and confined to supervisory control, system orchestration, network integration, and dynamic context management [6,8]. Concurrently, the time-critical, high-throughput stream-oriented data-plane processing is offloaded to the Programmable Logic (PL).

To experimentally validate the proposed architecture, a throughput-intensive stream processing workload is selected, namely continuous payload encryption for secure fronthaul links. Specifically, the Strumok stream cipher, adopted as the Ukrainian national standard DSTU 8845:2019 [9], is employed as a representative stream-cipher workload. In the following, the term Strumok refers to the cipher specified in [9].

Accordingly, the study focuses on the stream-oriented data-plane layer of an SoC FPGA edge node. DSP-intensive physical-layer kernels such as FFT, FIR filtering, and beamforming are left for separate evaluation because they exercise different arithmetic resources and memory-access patterns. Strumok is used here as the evaluation workload because its continuous XOR/SHIFT-dominated stream structure makes the control-plane/data-plane

boundary observable without consuming DSP blocks that would otherwise be reserved for MAC-intensive physical-layer processing.

Unlike multiply-accumulate (MAC)-intensive RF baseband processing kernels, such as Fast Fourier Transforms (FFT) or beamforming, which rely heavily on dedicated DSP blocks, the considered cryptographic stream transformation is implemented entirely using combinational XOR- and shift-based logic [10,11]. The synthesized core therefore utilizes zero DSP blocks, explicitly reserving FPGA multiplier resources for physical-layer RF processing. This separation enables accurate isolation and quantification of data-plane energy efficiency, expressed in pJ/bit, for bit-level streaming operations.

By transforming its sequential arithmetic operations over Galois fields $GF(2^8)$ into spatially distributed, hardwired combinatorial logic (XOR/SHIFT networks), the architecture minimizes logical depth and eliminates the multi-cycle execution overhead inherent to CPUs. The synthesized core utilizes less than 6% of the FPGA logic capacity (2386 Adaptive Logic Modules, or ALMs) while delivering a deterministic throughput of 6.4 Gbps per core at a conservative 100 MHz system-integration clock; at the maximum characterized 150 MHz core frequency, the isolated RTL core reaches 9.6 Gbps. By bypassing external memory (DDR) bottlenecks, this cross-layer hardware solution provides an energy-aware and deterministic foundation for stream-oriented 5G/6G edge-node data paths.

2. Background and Related Work

The migration of baseband processing from centralized cloud infrastructures to distributed edge nodes is a defining characteristic of 5G and 6G network architectures, intensifying the dependence on local, on-premises computational resources [1,2].

Limitations of Traditional SDRs and General-Purpose CPUs: Traditional Software-Defined Radio (SDR) platforms rely predominantly on general-purpose processors (CPUs) or Graphics Processing Units (GPUs) for Digital Signal Processing (DSP) tasks. While software implementations offer maximum flexibility and rapid development cycles, they are subject to inherent sequential execution bottlenecks. As the spatial multiplexing order in Massive MIMO systems increases, the CPU instruction pipeline can become a limiting factor, contributing to latency violations under continuous stream-processing load. Furthermore, CPUs exhibit poor energy proportionality when handling continuous bit-level or byte-level data streams, expending the majority of their dynamic power on instruction fetching, cache coherence, and context switching rather than productive arithmetic computation [5,12].

The FPGA Paradigm and the Abstraction Gap: Field-Programmable Gate Arrays (FPGAs) have emerged as the primary alternative, offering massive spatial parallelism and deterministic latency. By mapping suitable algorithms directly into hardware logic, FPGAs can provide substantially better energy efficiency (Joules per bit) than CPU-centered execution for stream-oriented workloads. Despite these physical advantages, the classical FPGA design flow—relying on Hardware Description Languages (HDLs) such as VHDL or Verilog—imposes a severe abstraction penalty. Designing, verifying, and routing complex, multi-channel DSP pipelines at the Register Transfer Level (RTL) is highly time-consuming and rigid, making it difficult to adapt to rapidly evolving 5G/6G standards [10,13].

High-Level Synthesis (HLS) and System-on-Chip (SoC) Integration: To bridge this abstraction gap, High-Level Synthesis (HLS) tools have been introduced, allowing engineers to describe DSP algorithms in C/C++ and automatically generate RTL code. While HLS significantly accelerates the design of individual computational cores, it does not resolve the system-level integration challenges. Modern telecommunication nodes require more than isolated accelerators: they also need full network stacks, dynamic reconfiguration capabilities, and local data aggregation. This necessity has driven the

adoption of heterogeneous SoC FPGAs (e.g., Intel Cyclone V, Xilinx Zynq), which integrate a hard processor system (HPS) alongside programmable logic on a single die [13,14].

The open issue is therefore not whether an accelerator can be attached to an embedded processor. Established SoC-FPGA and PYNQ-style environments already show the value of combining Linux/Python software with programmable-logic accelerators [14,15]. The question addressed here is narrower: for a continuous stream, which traffic can safely stay in the OS/MMIO/control domain, which traffic must remain in the PL data path, and how this boundary changes the interpretation of throughput, latency, and energy.

In many conventional integrations, the FPGA is still reached through a DMA-mapped path, so payload movement continues to involve the CPU-side software stack or the external-memory hierarchy [14,16,17]. In this paper, Hardware Accountability names the boundary used to avoid that ambiguity: Linux is responsible for low-rate orchestration, while the programmable logic is responsible for high-rate payload movement and transformation.

Table 1 places this boundary next to two common alternatives: software SDR-style pipelines on general-purpose processors [4,5] and DMA-centric SoC-FPGA offloading, where the FPGA often behaves as an accelerator behind an external-memory path [14–17].

Table 1. Architectural comparison of edge-node stream-processing strategies. The comparison is supported by software-SDR and CPU-centered processing studies [4,5], SoC-FPGA and PYNQ-style accelerator integration work [14,15], and DMA/external-memory path analyses [16,17].

Architecture Type	Pure Software SDR (General-Purpose CPU)	Conventional SoC FPGA/PYNQ-Style Offload	Proposed Architecture (Hardware Accountability)
Data Path Routing	Via System RAM & CPU Cache	Via External SDRAM (DMA)	Direct Internal FPGA Buffering (Elastic FIFO)
Throughput Scalability	Low (Bounded by instruction pipeline)	Medium (Bounded by Memory Wall)	High (Bounded strictly by RTL logic limits)
Control Plane Determinism	Low (Vulnerable to OS scheduling & IRQ storms)	Medium (Subject to DMA interrupt overhead)	High ($P_{99} < 1$ ms via Spatial Isolation)
Energy Framing	System-level (Watts per channel)	High dynamic power (SDRAM pre-charging)	Estimated pJ/bit for computational logic

The prevailing methodologies force a strict compromise between software flexibility and hardware efficiency. Pure software SDRs suffer from non-deterministic latency and poor energy proportionality under continuous high-throughput loads. Conventional SoC FPGA implementations, while accelerating computation, typically retain a DMA-centric data path that leaves the system vulnerable to external memory bandwidth limits and interrupt handling overhead. This gap motivates a structural shift. To formally justify the proposed transition to the Hardware Accountability doctrine, the following section empirically quantifies the exact physical failure points of the software-driven pipeline before detailing our cross-layer architectural solution.

3. System-Level Bottlenecks and the Hardware Accountability Paradigm

Quantitative characterization of the physical and architectural limitations of software-driven baseband processing forms the necessary precondition for the proposed architectural transition. The fundamental bottlenecks manifest in two primary domains: the interrupt handling architecture of the processor and the physical throughput of external system memory.

3.1. Empirical Analysis of Software-Driven Processing Limits

To establish the performance ceiling of software-driven baseband processing on the target edge node, we characterized the saturation behavior under realistic streaming conditions. To accurately emulate the continuous data payload generated by 5G active antenna arrays, we constructed a hardware stress-test environment. The experimental setup utilized a dual-core ARM Cortex-A9 Hard Processor System (HPS) running a specialized embedded Linux distribution, interfaced with an external radio frequency (RF) front-end via a High-Speed USB 2.0 bus [6,18]. For the experiments reported in this study, this distribution was based on Linux kernel 6.18, and the orchestration prototype was implemented using Python 3.12.

To establish the baseline processing limits, the edge node was tasked with ingesting continuous 8-bit quadrature (I/Q) data streams. Under these stress conditions, the software-centered path remained workable up to approximately 15 MSPS and approached a practical saturation region near 20 MSPS, where sustained operation became unstable. In the evaluated setup, this threshold corresponds to a payload of roughly 320 Mbps and reflects the combined effect of USB protocol overhead, interrupt pressure, and software scheduling costs rather than arithmetic throughput alone. These findings provide empirical justification for bypassing the CPU-centered path in favor of direct internal FPGA routing [19,20].

If the received samples are expanded after ingress to a 16-bit I and 16-bit Q software representation, the corresponding memory-side traffic is doubled; however, the throughput and energy values in Table 2 are normalized to the external 8-bit I/Q payload stream.

Table 2. Architectural power and energy-efficiency contrast for the evaluated Cyclone V SoC. Note (*): Both power values are Quartus Prime Power Analyzer estimates, not physical board measurements. The HPS value reflects dual-core HPS dynamic power during software data routing at the 320 Mbps USB 2.0 HS saturation point, whereas the RTL value reflects vectorless dynamic power of the computational core only; these accounting boundaries are therefore different.

Evaluation Domain	Dynamic Power (mW)	Peak Throughput (Mbps)	Energy Efficiency (pJ/Bit)
Software Processing Subsystem (Dual-Core ARM Cortex-A9 Routing Path)	881.12	≈320 (Saturation)	~2754
RTL Computational Core (Data-Plane Core, 1 instance) (*)	24.00	9600	2.5

The analytical interpretation summarized in Figure 1 explains this transition.

Analytical modeling and empirical stress testing indicate an approximately proportional increase in CPU load below the observed saturation region. As the stream approaches the 15 to 20 MSPS transition range, the system enters a thrashing state, characterized by a non-linear degradation of scheduling latency and scheduler overhead. In the evaluated USB-based path, this saturation is strongly associated with the interrupt handling architecture of the peripheral buses.

Operating at 20 MSPS, the continuous data stream forces the I/O controller—which operates on 125 μ s microframes—to generate thousands of hardware interrupt requests (e.g., USB Request Blocks, or URBs) per second.

At this critical threshold, the frequency of hardware interrupts completely saturates the Linux kernel’s interrupt service routines (ISRs).

Based on the observed non-linear degradation of scheduling latency, we infer that the processor enters an “IRQ Storm” state. In this condition, the data indicate that CPU resources reach saturation due to excessive context switching, eliminating computational headroom and leading to deterministic latency violations. This behavior reflects a platform-

dependent scalability limit in the evaluated software-centric path: beyond the observed saturation region, increases in data rate amplify control overhead rather than useful computation. Ultimately, the observed limitation is dominated by interrupt handling overhead, with memory bandwidth pressure acting as a secondary amplification factor.

3.2. Architectural Dichotomy and Bypassing the Memory Wall

Beyond the computational limits of the processor, the continuous routing of wideband I/Q streams through external volatile memory introduces a severe energy penalty and integration overhead. In traditional Software-Defined Radio (SDR) architectures, the Data Plane intrinsically traverses the external system memory (typically DDR3 or DDR4) via Direct Memory Access (DMA).

A conventional Linux-managed SoC-FPGA DMA path also needs careful buffer ownership. Continuous DMA usually relies on the Contiguous Memory Allocator (CMA) to reserve physically contiguous memory, and the software must keep the ARM caches consistent with FPGA-side transactions. In streaming mode, this commonly means explicit cache invalidation or flushing around DMA buffers to avoid stale data [18,21,22]. The proposed high-rate path does not require cache coherence for payload processing; cache maintenance is discussed here as a cost of the external-memory route that the internally buffered PL path avoids.

Furthermore, as the spatial multiplexing order in 5G/6G systems increases, the aggregated data rate rapidly approaches the physical “Memory Wall”. For instance, in the evaluated Cyclone V SoC platform, the external DDR3 interface provides a theoretical maximum throughput of approximately 25.6 Gbps [18]. Attempting to scale baseband-adjacent stream processing beyond this physical memory barrier is not feasible within the evaluated system constraints and introduces additional memory-system energy and thermal overhead, because constant pre-charging and activating of SDRAM rows constitutes a major fraction of total dynamic power dissipation [17,23].

To mitigate interrupt saturation and memory bandwidth limitations, we propose an architectural transition based on the architectural doctrine of Hardware Accountability, enabling localized, energy-efficient data routing as depicted in Figure 2.

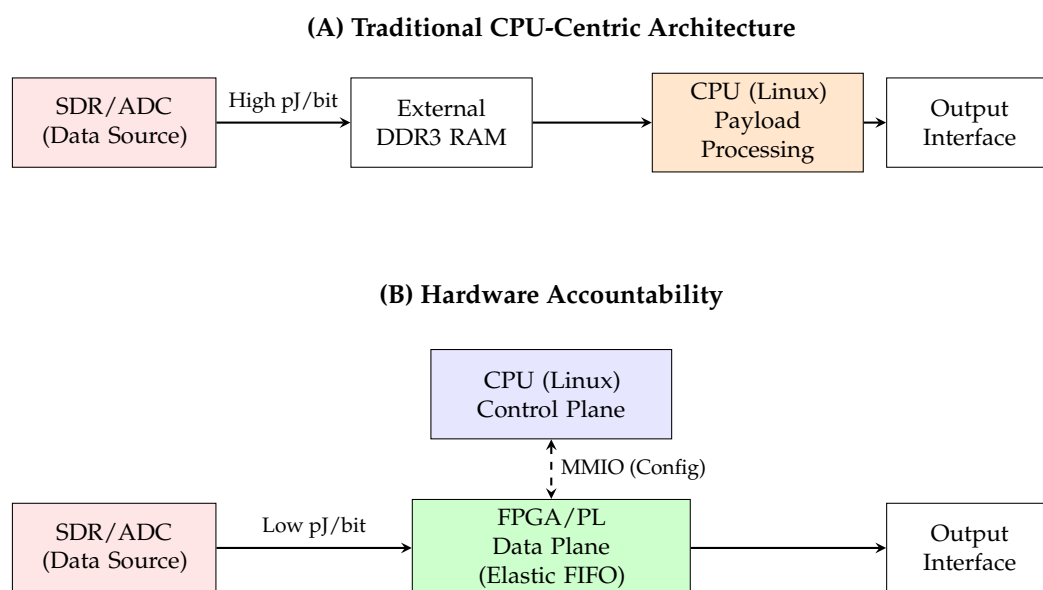


Figure 2. Architectural migration from a memory-bound, software-defined pipeline (A) to the proposed Hardware Accountability model (B), decoupling the high-throughput Data Plane from the energy-intensive OS context. The dashed MMIO link represents low-rate configuration and status traffic only; payload data are not carried through AXI4-Lite.

The core principle of this paradigm is the explicit spatial and functional decoupling of the Control Plane from the Data Plane. This transition removes high-rate data movement and real-time processing from the CPU domain, confining it to supervisory control and orchestration tasks. Consequently, this separation decouples timing-critical data processing from non-deterministic software execution, enabling predictable system behavior at high data rates [8]. Instead of relying on energy-intensive DMA transfers to external RAM, the wideband data streams are ingested, processed, and routed entirely within the Programmable Logic (PL) matrix.

The proposed architecture keeps this high-rate path inside the FPGA fabric and uses an on-chip Elastic FIFO for rate matching between the stream-processing cores and external RF or serial interfaces. Because the payload stream is not repeatedly written to and read from external SDRAM, the design avoids a major source of dynamic memory power. Linux still interacts with the hardware through a lightweight AXI4-Lite/MMIO interface, but that interface is reserved for context updates, status, and supervisory orchestration [14,22].

3.3. System Dataflow and Interface Roles

The system separates two paths. The payload path begins with a continuous radio-side I/Q or secure fronthaul stream arriving from the SDR/ADC-facing side of the node. In the scaled architecture, this stream enters the PL data path, is transformed by the RTL processing core or core cluster, packed by the Stream Aggregator, and rate-matched by the Elastic FIFO before being forwarded toward the high-speed RF, optical, or serial output interface. The Elastic FIFO provides local buffering for rate matching and clock-domain crossing; it is not intended as a software-visible DDR substitute.

The control path is far narrower. Linux writes key material, initialization vectors, START/status words, and context-management commands through the AXI4-Lite/MMIO register interface. These transactions configure the hardware state machines and shadow registers, but they do not carry high-throughput payload samples. Thus, AXI4-Lite is used only where it is appropriate: low-rate, memory-mapped supervisory control. The high-rate data path is represented by AXI4-Stream-style internal links in the scaled model, with the Stream Aggregator and Elastic FIFO absorbing rate and clock-domain mismatches.

The corresponding system-level dataflow and interface diagram is shown in Figure 3, where the payload stream, control path, local buffering, output interface, and bypassed DDR route are separated explicitly.

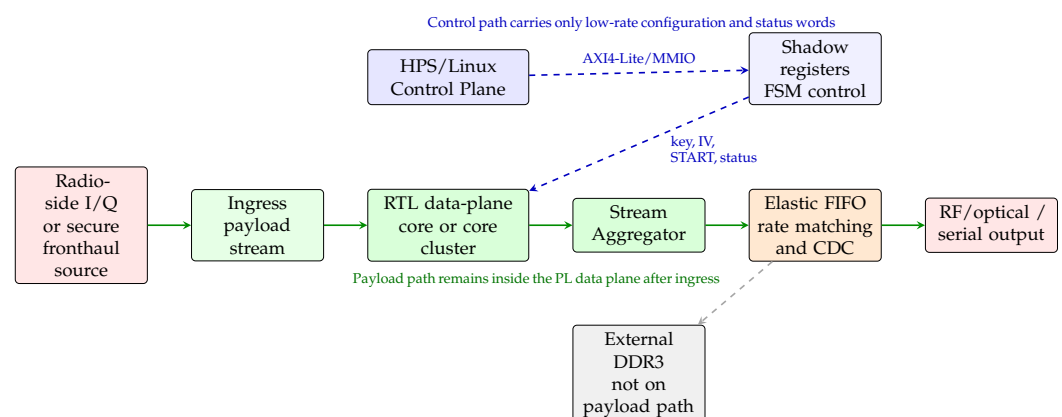


Figure 3. System-level dataflow and interface roles. High-rate payload data enter the PL data path, pass through the RTL core, Stream Aggregator, and Elastic FIFO, and leave through the output interface. AXI4-Lite/MMIO is used only for control and context words; external DDR3 is shown only as the memory route bypassed by the high-rate payload path.

4. Microarchitectural Optimization and Energy-Efficient Hardware Offloading

As established in the Introduction, the Strumok stream cipher, adopted as the Ukrainian national standard DSTU 8845:2019, serves as the reference data-path workload [9]. For international context, this is a Ukrainian national standard stream cipher structurally based on Linear Feedback Shift Registers (LFSR) and non-linear transformations over finite Galois fields ($GF(2^8)$ and $GF(2^{64})$).

We deliberately selected this algorithm because its structural topology—relying entirely on combinatorial XOR/SHIFT networks—is highly representative of stream-oriented payload encryption layers. Such bit-level operations are notoriously inefficient on general-purpose processors due to instruction fetch overhead and multi-cycle execution limits [11].

This stream cipher operates at the payload encryption layer, independently of and without contention for DSP resources that may be needed by MAC-intensive physical-layer operations such as FFT or beamforming; both operate on continuous multi-gigabit streams. By synthesizing this high-throughput cryptographic core with zero hardware DSP blocks and relying entirely on combinatorial XOR/SHIFT networks, the architecture leaves the FPGA’s dedicated DSP resources available for physical-layer RF mathematics. This constitutes a spatial division of logic, where the reported pJ/bit metric specifically quantifies the energy efficiency of non-linear payload transformations.

4.1. RTL Synthesis and Spatial Materialization of Arithmetic Operations

We propose a hybrid design methodology that physically separates the integration layer from the computational core. While High-Level Synthesis (HLS) and software-driven drivers are utilized for the control and orchestration layer (AXI-Lite interfaces, MMIO mapping), the critical data path is synthesized directly at the Register Transfer Level (RTL). This microarchitectural separation ensures that the control path logic does not introduce latency overhead into the continuous multi-gigabit data streams [10,13].

The primary mechanism for energy efficiency in the proposed RTL core is the spatial materialization of arithmetic operations. In a traditional software pipeline, operations such as multiplication by a primitive element α in $GF(2^8)$ require iterative shift and logical addition (XOR) instructions, consuming substantial dynamic power per byte. In contrast, our hardware offloading strategy transforms these operations into hardwired, spatially distributed combinatorial networks, as exemplified by the single-stage combinational logic path for multiplication by 0x02 shown in Figure 4 [9–11].

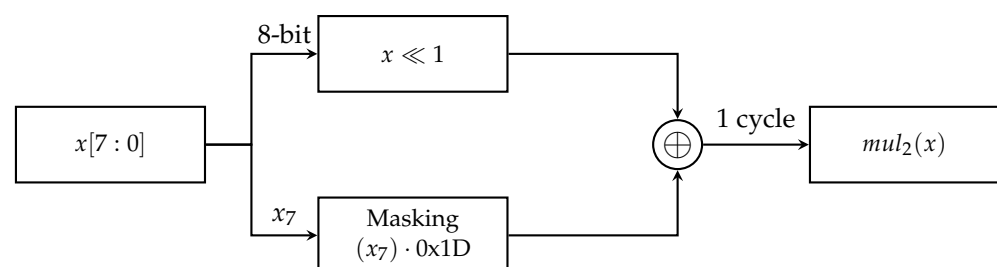


Figure 4. Combinational logic path for multiplication in $GF(2^8)$ demonstrating the spatial materialization of arithmetic operations and elimination of sequential instruction cycles.

Formally, multiplication by 0x02 in this network is defined by Equation (1):

$$mul_2(x) = (x \ll 1) \oplus ((x_7) \cdot 0x1D), \quad (1)$$

where x_7 is the most significant bit of the input byte x , and the polynomial addition of 0x1D is dynamically activated via hardware gating.

By embedding such operations exclusively within the combinational logic between data path registers, the architecture eliminates the multi-cycle instruction decoding overhead entirely. The execution of complex non-linear byte transformations occurs within a single clock cycle constraint ($t_{pd} < T_{clk}$). This eliminates the unpredictable latency variations caused by CPU cache misses or OS scheduler preemption, providing deterministic real-time execution under the defined system configuration.

To guarantee the temporal determinism of the data path and prevent race conditions during context switching, the execution phases are governed by a hardwired Finite State Machine (FSM) entirely independent of the CPU. The FSM enforces three sequential execution phases: First, the **INIT** phase atomically loads the 512-bit key and 256-bit initialization vector into the state registers. This is immediately followed by a deterministic **WARM-UP** phase, during which the non-linear logic loops internally without asserting the output ready flag, completing the specified initialization and mixing rounds before output is released. Only upon the completion of these cycles does the FSM transition to the **GEN** (generation) state, asserting the hardware ready strobe. This microarchitectural separation of the data path (combinatorial XOR networks) and the control path (FSM) guarantees that valid data-plane payload is generated with a predictable, cycle-accurate latency, immune to any software-induced execution stalls.

4.2. Resource Utilization and Power Profiling

The synthesized RTL core demonstrates high resource efficiency. Synthesized for the Intel Cyclone V SoC platform (5CSEBA6) [18], a single processing core occupies 2386 Adaptive Logic Modules (ALMs), accounting for less than 6% of the total logic capacity. Notably, the design utilizes zero Digital Signal Processing (DSP) blocks or Block RAMs (M10K), relying entirely on distributed XOR/SHIFT logic arrays. This low-footprint synthesis ensures minimal static power leakage.

Operating at its maximum characterized core clock frequency (F_{max}) of 150 MHz, a single isolated core achieves a deterministic throughput of 9.6 Gbps. Based on vectorless power estimation using the Intel Quartus Prime Power Analyzer v23.1 with default toggle rates, the estimated dynamic power dissipation of the RTL computational core at 150 MHz amounts to 24.00 mW. When evaluated against the effective throughput, this synthesis-derived estimate corresponds to an energy-efficiency metric of ≈ 2.5 picojoules per bit (pJ/bit), as detailed in Table 3.

Table 3. Provenance of headline results used in the architectural comparison.

Quantity	Value	Basis and Accounting Boundary
Software-path saturation	≈ 20 MSPS/ ≈ 320 Mbps	Observed on the evaluated USB 2.0/Cortex-A9/Linux I/Q routing path; platform-specific payload-ingress limit.
HPS software-routing power	881.12 mW	Quartus Prime Power Analyzer estimate for dual-core HPS dynamic power during software data routing; not a board measurement.
RTL core throughput	9.6 Gbps at 150 MHz	Single synthesized computational core; isolated maximum characterized core clock.
RTL computational-core power	24.00 mW/ ≈ 2.5 pJ/bit	Quartus Prime Power Analyzer vectorless estimate for the RTL computational core only.
Control-plane latency	$P_{99} < 1$ ms	Measured orchestration jitter under Spatial Isolation.
Projected cluster throughput	83.2 Gbps	Design-space exploration projection using 13 cores at a conservative 100 MHz integration clock; not implemented board throughput.

On the dual-core Cortex-A9 HPS path, the evaluated 8-bit I/Q USB 2.0 stream saturates near 320 Mbps. Quartus Prime Power Analyzer reports 881.12 mW HPS dynamic power for the dual-core software-routing configuration, corresponding to roughly 2754 pJ/bit under that software-path accounting boundary. The RTL number is different in scope: it is an estimated 2.5 pJ/bit for the computational logic of the synthesized core. The comparison is useful for architectural accounting, but it should not be read as a component-by-component power comparison between the processor subsystem and the FPGA fabric.

Thus, this represents an architectural efficiency contrast demonstrating the system-level value of spatial offloading, rather than a direct like-for-like silicon-level measurement.

5. Dynamic Context Switching and Control Plane Stability

Although hardware offloading substantially improves Data Plane energy efficiency and throughput, 5G/6G edge nodes impose concurrent demands for dynamic operational reconfigurability. In practical deployments, operating modes and cryptographic contexts may need to be updated continuously during runtime. This introduces the challenge of Dynamic Context Switching, where the Control Plane must inject new parameters without stalling the continuous multi-gigabit data stream.

5.1. MMIO Adapter Architecture and Shadow Registers

To address this, our heterogeneous architecture implements an explicit microarchitectural separation using a Memory-Mapped I/O (MMIO) AXI4-Lite interface dedicated solely to the Control Plane [22]. To prevent race conditions, we designed a hardware Shadow Register File within the adapter layer. The Linux-based orchestration service asynchronously writes the new configuration (e.g., a 512-bit key context) into intermediate shadow registers. The transition to the active state is executed atomically within a single clock cycle upon receiving a hardware START strobe.

Experimental evaluation of this architecture demonstrates high control-plane maneuverability, but the operation being counted must be distinguished explicitly. A conservative full context reload through MMIO, including key and IV register writes, initialization, and synchronization, was measured at approximately 1290 reloads per second. The shadow-register architecture also supports faster context/state update operations above 7200 per second under Spatial Isolation: key and IV words remain resident in shadow registers, the IV need not be rewritten when it is unchanged, and the FSM advances the stream state internally. This distinction keeps the control-plane result tied to the actual measured operation while preserving the architectural point: MMIO remains suitable for low-rate orchestration, not high-rate payload transport [3].

5.2. Spatial Isolation and Jitter Mitigation

The deployment of a general-purpose OS such as Linux for hardware orchestration introduces phase jitter attributable to scheduler preemption and interrupt handling [8,19,20]. To guarantee the determinism of the Control Plane, we applied a Spatial Isolation methodology. By confining the OS background services and peripheral hardware interrupts (IRQ shielding) to a specific Cortex-A9 core (Core 0), while dedicating an isolated core (CPU1) exclusively to the orchestration pipeline, we reduced latency variability to bounded levels.

Spatial Isolation is achieved through native Linux kernel resource management mechanisms, avoiding the integration complexity and scheduling overhead of a hypervisor or RTOS [8,20]. Specifically, all background daemons and peripheral hardware interrupt requests (hard IRQs)—such as those from USB or network controllers that cause “IRQ storms”—were statically assigned to CPU0 using SMP affinity masks (`smp_affinity`). Concurrently, the orchestration application was bound exclusively to the isolated CPU1

using core affinity, and its execution policy was elevated to a real-time scheduling class (SCHED_FIFO).

Measured over a continuous stress test of $N \approx 8.4 \times 10^5$ context switch requests, the unmanaged Linux scheduler exhibits latency spikes reaching up to 24.40 ms, attributable to peripheral interrupts competing for CPU cycles. In contrast, the explicit routing of interrupts away from the orchestration core stabilizes the system, reducing the 99th percentile (P99) latency to just 0.98 ms (Figure 5).

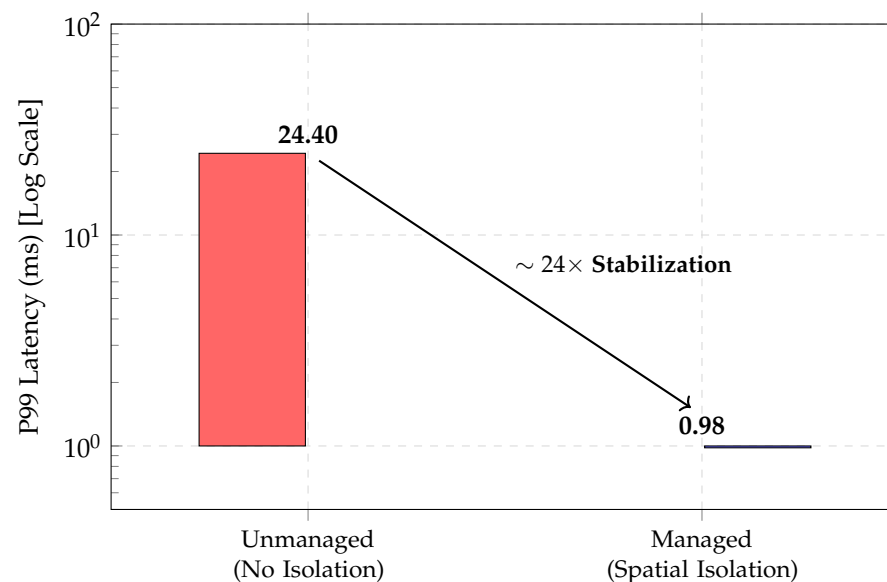


Figure 5. Impact of Spatial Isolation (IRQ shielding and CPU affinity) on Control Plane jitter. The managed architecture maintains sub-millisecond control-plane latency under stress.

To formally characterize the latency distribution of the MMIO interface and validate the architectural dichotomy, we conducted a time-domain decomposition of a single control transaction. As illustrated in Figure 6, the execution time is highly asymmetric. The hardware latency of the AXI4-Lite bridge (MMIO transaction) accounts for 60.0 μ s, while the Python-based OS stack adds 79.7 μ s. Consequently, the orchestration layers consume over 99% of the total transaction time (~ 140 μ s), whereas the actual RTL cryptographic transformation executes in 0.33 μ s. Migration of the orchestration service to a compiled language (e.g., C/C++) would reduce the software overhead, yet the characterized AXI MMIO latency of approximately 60 μ s per transaction remains the dominant hardware-side constraint within the evaluated MMIO path. Under the Shadow Register File access pattern, the measured NEXT-and-read throughput reaches approximately 462 kbps, consistent with a modeled cap of approximately 450 kbps for 64-bit-per-transaction MMIO transfers; under conservative READY-pollled operation, the measured throughput is 344–348 kbps. Nonetheless, remaining below a 1 ms control-plane latency target under this non-deterministic stack provides a practical upper bound on orchestration delay under Spatial Isolation. This supports the Hardware Accountability strategy: the processor and MMIO bus must remain confined to low-frequency orchestration, while the high-bandwidth payload is routed autonomously via internal FPGA buffering rather than through the CPU's sequential execution pipeline.

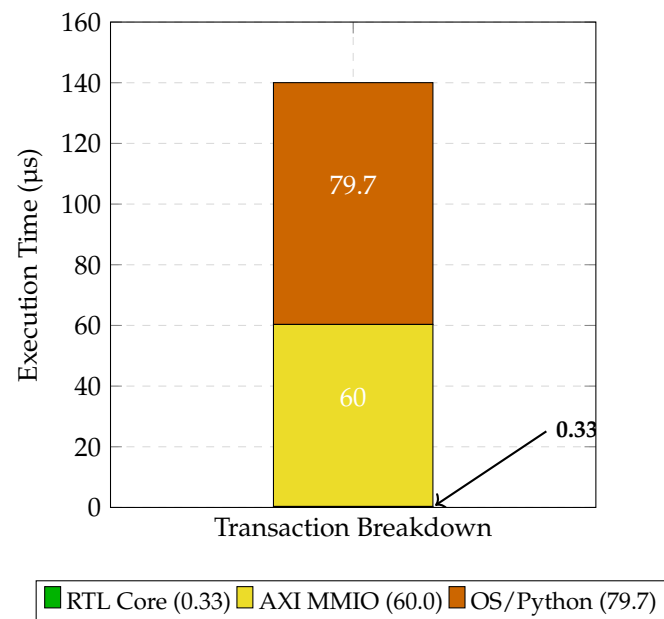


Figure 6. Time-domain decomposition of a single control transaction. The actual RTL computation takes less than 1% of the total time, showing that the CPU and MMIO bridge dominate this control path during data streaming.

The Python prototype contributes 79.7 μs of software-stack overhead, or slightly more than half of the measured transaction time. A compiled orchestration service, for example in C/C++, would reduce this part of the delay. However, the evaluated MMIO path still includes 60.0 μs of AXI MMIO hardware latency, so software optimization would not make AXI4-Lite suitable for payload transfer. The result is consistent with the intended split: MMIO is appropriate for low-rate orchestration, while multi-gigabit payload routing belongs in the internally buffered FPGA data path.

6. Design Space Exploration and System-Level Architecture

The following DSE is a scaling calculation based on the single-core synthesis result and the measured Control Plane behavior. The full multi-core cluster has not been implemented on the board. Among the DSE assumptions, timing closure and sustainable stream aggregation are the most critical practical risks because multi-core replication increases routing congestion, fan-out, buffering requirements, and pressure on shared interconnect resources. The thermal budget is an additional major uncertainty, while clock-domain crossing and RF/optical interface integration remain important system-level implementation risks. These limitations are summarized in the Discussion section.

The DSE starts from two measured or estimated inputs: $P_{99} < 1 \text{ ms}$ Control Plane jitter and the synthesis-derived RTL computational-core energy estimate at 150 MHz. The latency decomposition also shows that the MMIO adapter is a control interface, not a multi-gigabit payload interface. The scaling model therefore examines how far the PL data path can be extended before the external-memory hierarchy becomes the limiting factor [16,17,22].

Based on the logic synthesis results, a single RTL processing core occupies 2386 ALMs (<6% of the available logic capacity on the Cyclone V 5CSEBA6). Allocating a conservative 20% resource margin for system-level AXI4-Stream interconnects, Stream Aggregators, and orchestration wrappers, the programmable logic matrix can theoretically accommodate a parallel cluster of $N = 13$ independent processing cores. Extrapolating the core throughput based on a conservative, stable system integration clock of 100 MHz (yielding 6.4 Gbps per core, deliberately scaled down from the 150 MHz/9.6 Gbps isolated theoretical maximum

to reserve timing and thermal margin across a multi-core array), such a cluster provides a projected upper-bound data-plane capacity of 83.2 Gbps rather than an implemented board-level throughput result.

Translating this theoretical capacity into a physical implementation reveals a fundamental physical bottleneck: the “Memory Wall.” In traditional software-defined or standard DMA-centric architectures, the payload must traverse the external volatile memory. For the evaluated Cyclone V SoC platform, the FPGA-to-SDRAM bridge hierarchy is expected to become the dominant throughput limiter before the nominal DDR3 peak is reached [17,18,23]. For this reason, routing a projected 83.2 Gbps stream through external SDRAM is not a realistic target for the evaluated platform, as illustrated in Figure 7.

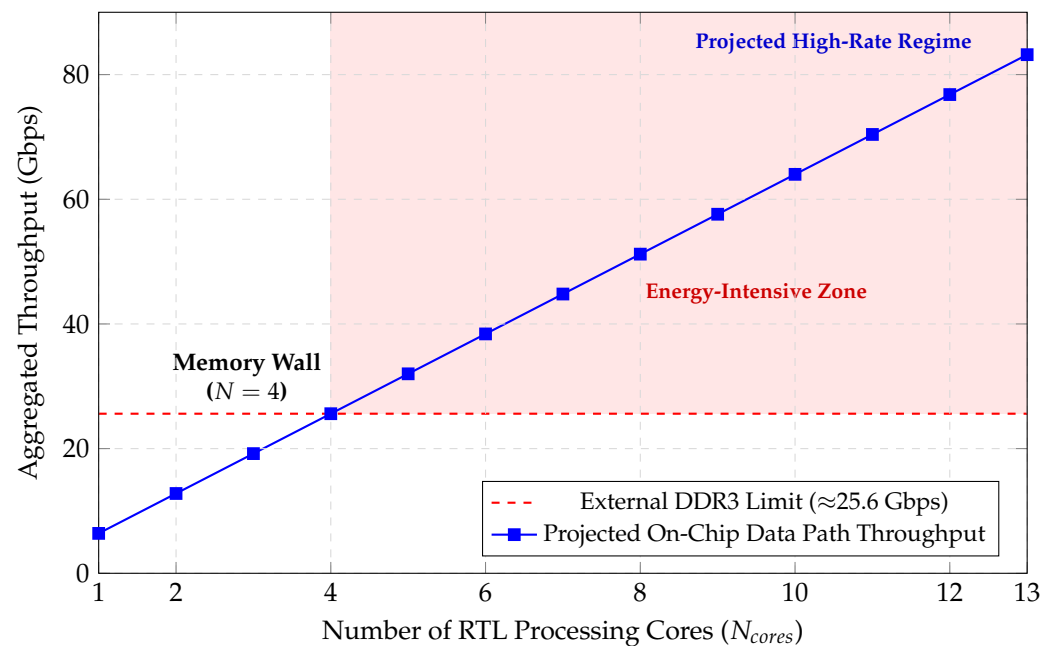


Figure 7. Design Space Exploration (DSE) of the stream-oriented data-plane processing cluster. In the evaluated platform model, scaling beyond 4 cores requires bypassing the external DDR3 path.

The scaled model therefore aggregates the stream-oriented core outputs inside the PL instead of staging them in external DDR3. The 13 RTL cores feed a Stream Aggregator, which packs the outputs into an 832-bit internal bus. The Elastic FIFO then provides rate matching and the Clock Domain Crossing (CDC) boundary toward the high-speed TX clock domain, as shown in Figure 8.

This pJ/bit metric quantifies the estimated dynamic power dissipation of the computational logic only; in a full-scale physical deployment, driving a projected 83.2 Gbps stream through high-speed transceivers and optical interfaces would add a separate power cost on the order of Watts. Keeping the stream in the internal Elastic FIFO avoids the additional memory-fetch overhead that would appear if the same traffic were routed through external DDR.

The empirical data from Control Plane validation support the architectural assumptions underlying this scaled model, provided that the operation type is kept explicit. At the measured rate of more than 7200 shadow-register context/state update operations per second, issuing one update operation to each of the 13 projected cores would take on the order of 1.8 ms. If a full key-and-IV reload with synchronization is required, the conservative benchmark of approximately 1290 reloads per second gives a worst-case bound on the order of 10 ms for the same 13-core cluster. Both bounds remain within the considered IMT-2020-oriented control interval [3], so the general-purpose CPU can

orchestrate dynamic parameter rotations without disrupting the continuous Data Plane stream. This supports the architectural viability of the proposed approach as a means of mitigating both the software processing bottleneck and the external memory wall.

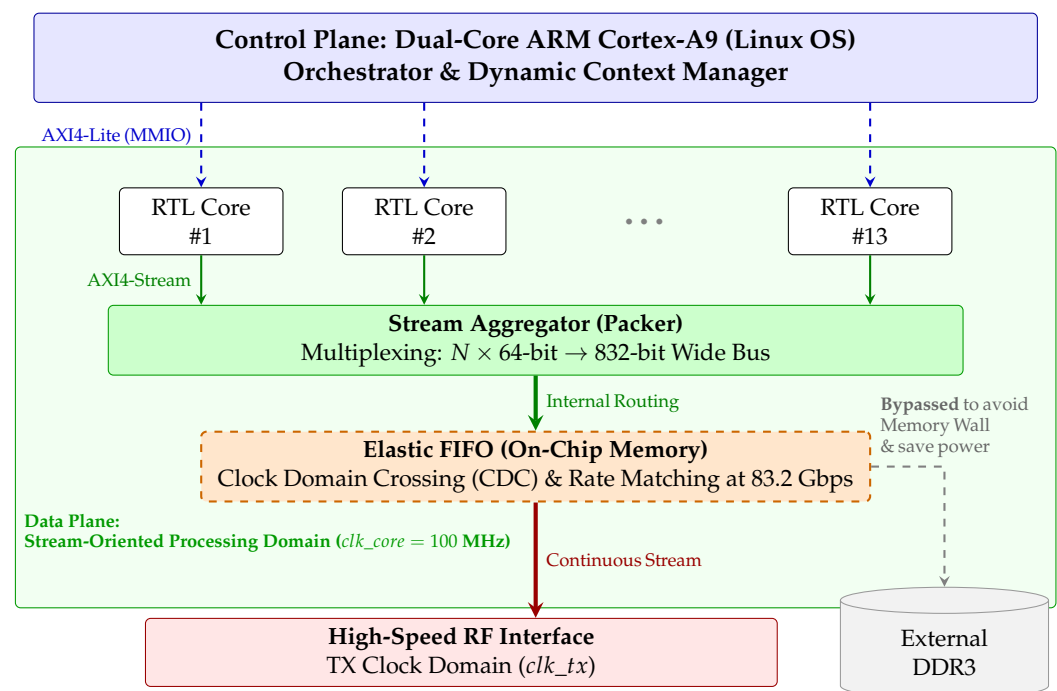


Figure 8. Projected multi-core data-plane architecture for the 83.2 Gbps DSE case. The schematic defines the Clock Domain Crossing (CDC) boundary for the scaled cluster. The Control Plane (HPS) orchestrates the RTL cluster via a 32-bit AXI4-Lite bus. The Data Plane uses 64-bit AXI4-Stream interconnects, which are packed into a wide internal bus. The Elastic FIFO bridges the asynchronous boundary between the stream-processing clock (clk_{core}) and the high-speed transceiver clock (clk_{tx}), while the high-rate payload path bypasses external DDR3 memory.

7. Discussion

The synthesized heterogeneous architecture reduces Data Plane routing overhead and estimated dynamic energy consumption for the evaluated workload. These findings suggest that, on such stream-oriented edge-node paths, energy optimization cannot rely solely on algorithmic improvements or RF front-end enhancements; it also depends on cross-layer architectural organization.

The Cost of General-Purpose Processing: The software-routed Cortex-A9 path illustrates the energy cost of moving continuous payload streams through a general-purpose processor, operating system, and USB/I/O stack. At the evaluated 8-bit I/Q USB 2.0 saturation point, the Quartus-estimated HPS dynamic power corresponds to approximately 2754 pJ/bit under the stated subsystem accounting boundary. As spatial multiplexing scales the data rates to tens of gigabits per second, routing this payload through a CPU-centered path introduces substantial thermal and scheduling overhead.

Energy-Proportional Architecture via Hardware Segregation: In the RTL core, the same byte-level transformations are mapped to combinational XOR networks. Quartus Power Analyzer estimates 24.00 mW dynamic power at 150 MHz for this computational core, or approximately 2.5 pJ/bit. Within the stated accounting boundary, this supports the use of the CPU for low-rate orchestration while keeping the high-rate payload path in the FPGA Data Plane.

Overcoming the Memory Wall: The DSE also shows why the external-memory path is a poor fit for the projected cluster [17,18,23]. Under the stated assumptions, a 13-core RTL

cluster projects to 83.2 Gbps, while the evaluated DDR3 path is limited to approximately 25.6 Gbps before protocol and arbitration losses. The projected architecture therefore keeps the high-rate Data Plane inside the FPGA routing fabric and sends it toward high-speed interfaces without staging the payload in external DDR.

Limitations and Future Work: The stream-oriented Data Plane architecture is optimized for sequential, throughput-bound workloads and is not directly applicable to algorithms requiring stochastic random-access memory patterns, such as large-scale matrix factorization. The present revision also does not include board-level power measurement of the RTL data path under realistic switching activity, nor does it include a same-workload implemented DMA-centric baseline. Therefore, the HPS and RTL power values should be interpreted as Quartus Power Analyzer estimates under different accounting boundaries, not as board-level measurements.

The observed 20 MSPS software-path saturation is also platform-specific to the evaluated USB 2.0/Cortex-A9/Linux integration path and should not be generalized as a universal processor limit. The 83.2 Gbps figure carries the same projection caveat: physical realization would require timing closure of the multi-core interconnect, thermal-budget validation, high-width stream aggregation, robust clock-domain crossing, and integration with actual RF/optical interfaces. Among these assumptions, timing closure and sustainable stream aggregation are the most critical practical risks, since they determine whether replicated cores can be routed, buffered, and connected without turning the internal stream fabric into the new bottleneck. Physical implementation and board-level validation of the full cluster therefore remain future work. Separately, DSP-heavy PHY kernels such as FFT, FIR filtering, and beamforming require their own resource and energy evaluation; the present measurements and estimates are limited to stream-oriented payload/data-plane processing.

Future research will focus on implementing the DMA-centric baseline, collecting board-level power measurements under realistic switching, and extending the evaluation to MAC-intensive PHY kernels. Heterogeneous memory architectures, such as High Bandwidth Memory (HBM), and tightly coupled CPU-FPGA cache-coherence protocols may improve flexibility for workloads that cannot remain fully inside the PL stream path, but they are not part of the proposed high-rate payload route evaluated in this paper.

8. Conclusions

The continued increase in continuous data-plane rates in 5G/6G edge communication nodes motivates a more explicit separation between low-rate orchestration and high-rate stream processing at the edge. This paper demonstrates through experimental evaluation that the tested USB 2.0/Cortex-A9/Linux software path encounters a platform-specific scalability ceiling, reaching a saturation regime at approximately 20 MSPS consistent with IRQ-driven overload.

To address this, we implemented and validated a heterogeneous SoC FPGA prototype based on the Hardware Accountability architectural doctrine. By decoupling the Control Plane (managed by a spatially isolated Linux OS) from the Data Plane (offloaded to the Programmable Logic), the architecture bypasses the sequential execution bottlenecks inherent to general-purpose CPUs for the evaluated stream-oriented workload. The microarchitectural synthesis of the RTL computational core yields a vectorless Power Analyzer-based energy estimate of approximately 2.5 pJ/bit at a maximum characterized core frequency of 150 MHz.

Experimental results of the integrated system confirm that the heterogeneous node achieves high control agility, with conservative full context reloads near 1290 per second, more than 7200 shadow-register context/state update operations per second, and stable

orchestration jitter of $P99 < 1$ ms. These results make the architecture suitable for latency-constrained control-plane operation in energy-efficient edge telecommunication systems. Furthermore, based on these empirical results and the identified limitations of MMIO data transfers, the paper proposes a DSE model projecting the architecture to an 83.2 Gbps processing cluster. Together, the measured control-plane results and the design-space projection provide a concrete foundation for energy-efficient stream-oriented data-plane processing at the wireless edge.

Author Contributions: Conceptualization, Y.H. and O.K.; methodology, Y.H. and O.K.; software, Y.H. and O.K.; validation, Y.H. and O.K.; formal analysis, Y.H. and O.K.; investigation, Y.H. and O.K.; resources, Y.H., O.K., D.V. and V.B.; data curation, Y.H. and O.K.; writing—original draft preparation, Y.H.; writing—review and editing, Y.H., O.K., D.V. and V.B.; visualization, Y.H.; supervision, O.K., D.V. and V.B.; project administration, O.K., D.V. and V.B.; funding acquisition, O.K., D.V. and V.B. All authors have read and agreed to the published version of the manuscript.

Funding: The work of Riga Technical University was supported by the Latvian Council of Science, Project LIGHTNESS, under Grant lzp-2024/1-0253.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: Y.H. and O.K. thank Altera for its support within the Altera University Program.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ALM	Adaptive Logic Module
AXI	Advanced eXtensible Interface
CMA	Contiguous Memory Allocator
CPRI	Common Public Radio Interface
CPU	Central Processing Unit
DDR	Double Data Rate
DMA	Direct Memory Access
DSE	Design Space Exploration
DSP	Digital Signal Processing
DTB	Device Tree Blob
FFT	Fast Fourier Transform
FHSS	Frequency Hopping Spread Spectrum
FIFO	First-In, First-Out
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
GF	Galois Field
HLS	High-Level Synthesis
HPS	Hard Processor System
I/Q	In-phase/Quadrature
IRQ	Interrupt Request
ISR	Interrupt Service Routine
LUT	Look-Up Table
MIMO	Multiple-Input Multiple-Output
MMIO	Memory-Mapped Input/Output
MSPS	Megasamples Per Second

NR	New Radio
OPS	Operations Per Second
OS	Operating System
PL	Programmable Logic
RF	Radio Frequency
RTL	Register Transfer Level
SDR	Software-Defined Radio
SMP	Symmetric Multiprocessing
SoC	System-on-Chip
URB	USB Request Block
URLLC	Ultra-Reliable Low-Latency Communication

References

- Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
- Gill, S.S.; Golec, M.; Hu, J.; Xu, M.; Du, J.; Wu, H.; Walia, G.K.; Murugesan, S.S.; Ali, B.; Kumar, M.; et al. Edge AI: A Taxonomy, Systematic Review and Future Directions. *Clust. Comput.* **2025**, *28*, 18. [CrossRef]
- ITU-R. *Minimum Requirements Related to Technical Performance for IMT-2020 Radio Interface(s)*; Report M.2410-0; International Telecommunication Union: Geneva, Switzerland, 2017.
- Oliva, M.J.; García, P.A.; Spinelli, E.M.; Veiga, A.L. Open-source SoC-FPGA Platform for Signal Processing. In Proceedings of the XI Southern Programmable Logic Conference (SPL 2023), San Luis, Argentina, 27–31 March 2023. Available online: <http://sedici.unlp.edu.ar/handle/10915/160854> (accessed on 20 July 2026).
- Owaida, M.; Alonso, G.; Fogliarini, L.; Hock-Koon, A.; Melet, P.E. Lowering the latency of data processing pipelines through FPGA based hardware acceleration. *Proc. VLDB Endow.* **2019**, *13*, 71–85. [CrossRef]
- Herman, Y.; Krulikovskiy, O.; Haliuk, S.; Subbotin, S. Development of an embedded operating system based on the Linux kernel for SoC FPGA. In Proceedings of the Seventh International Workshop on Computer Modeling and Intelligent Systems (CMIS-2024), Zaporizhzhia, Ukraine, 3 May 2024; Volume 3702, pp. 376–388. . [CrossRef]
- Krulikovskiy, O.; Herman, Y.; Verenko, O. Hardware/software communication architecture for a SoC-FPGA-based time-to-digital converter. In *Proceedings of the Seventeenth International Conference on Correlation Optics, Chernivtsi, Ukraine, 8–12 September 2025*; SPIE: Bellingham, WA, USA, 2025; Volume 13813, pp. 214–217. [CrossRef]
- Pagani, S.; Biondi, A.; Marinoni, M.; Buttazzo, G. FRED-Linux: A Linux-Based Support for Real-Time Applications on Heterogeneous Platforms. *Future Gener. Comput. Syst.* **2022**, *129*, 244–260. [CrossRef]
- State Enterprise “Ukrainian Research and Training Center of Standardization, Certification and Quality”. *Information Technology. Cryptographic Protection of Information. Algorithm of Symmetric Stream Transformation*; Technical Report DSTU 8845:2019; Ukrainian National Standard: Kyiv, Ukraine, 2019.
- Woods, R.; McAllister, J.; Lightbody, G.; Yi, Y. *FPGA-Based Implementation of Signal Processing Systems*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2017.
- Kheddar, H.; Azzouzi, O.; Anane, M.; Ghanem, M.C.; Himeur, Y. Efficient fine-grained LUT-based optimization of AES MixColumns and InvMixColumns for FPGA implementation. *Electronics* **2025**, *14*, 4912. [CrossRef]
- Dean, J.; Barroso, L.A. The tail at scale. *Commun. ACM* **2013**, *56*, 74–80. [CrossRef]
- Del Sozzo, E.; Conficconi, D.; Zeni, A.; Salaris, M.; Sciuto, D.; Santambrogio, M.D. Pushing the Level of Abstraction of Digital System Design: A Survey on How to Program FPGAs. *ACM Comput. Surv.* **2023**, *55*, 106:1–106:48. [CrossRef]
- Novickis, R.; Greitans, M. FPGA Master Based On-Chip Communications Architecture for Cyclone V SoC Running Linux. In *Proceedings of the 2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*; IEEE: Piscataway, NJ, USA, 2018. [CrossRef]
- Schmidt, A.G.; Weisz, G.; French, M. Evaluating Rapid Application Development with Python for Heterogeneous Processor-Based FPGAs. In *Proceedings of the 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*; IEEE: Piscataway, NJ, USA, 2017, pp. 121–124. [CrossRef]
- Sullivan, W.; Steinkamp, D. *Characterization of Throughput on the AXI DMA Bus for Burst Data Transfer over Ethernet*; Technical Report; Office of Scientific and Technical Information (OSTI), U.S. Department of Energy: Oak Ridge, TN, USA, 2024. [CrossRef]
- Vieira, J. Assessing On-Chip High-Performance Interfaces on Xilinx and Intel FPGA-SoC Devices. 2023. Available online: https://v2.joaomiguelvieira.com/~joaovieira/public/docs/reports/fpga_dma.pdf (accessed on 20 July 2026).
- Intel Corporation. *Cyclone V Hard Processor System Technical Reference Manual*; Intel Corporation: Santa Clara, CA, USA, 2022.
- de Oliveira, D.B.; Casini, D.; Cucinotta, T. Operating system noise in the Linux kernel. *IEEE Trans. Comput.* **2022**, *72*, 196–207. [CrossRef]

20. Reghenzani, F.; Massari, G.; Fornaciari, W.; Santambrogio, M.D. The Real-Time Linux Kernel: A Survey on PREEMPT_RT. *ACM Comput. Surv.* **2019**, *52*, 18:1–18:36. [[CrossRef](#)]
21. Koul, V. DMA Engine API Guide; The Linux Kernel Project, 2024. Available online: <https://docs.kernel.org/driver-api/dmaengine/client.html> (accessed on 20 July 2026).
22. Arm Limited. *AMBA AXI and ACE Protocol Specification*; Arm Limited: Cambridge, UK, 2020.
23. Ferry, E.; Yuki, T.; Derrien, S.; Rajopadhye, S. Increasing FPGA accelerators memory bandwidth with a burst-friendly memory layout. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *42*, 1546–1559. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.