DOI: 10.35598/mcfpga.2026.014

Fpga Implementations Of A Divisioned Model Of Two-Ball Quorum-Sectional Processing Machine For High-Convantage Flows

Oleksandr Kupriianov
ORCID 0009-0003-8155-1321
Department of Computer Systems
National University "Odeska Polytechnic",
Odessa , Ukraine
kupriyanov.a062@gmail.com

Maksimenko Andriy
ORCID 0000-0003-3117-1657
Department of Computer Systems
National University "Odeska Polytechnic",
Odessa , Ukraine
maksymenko.a.o@op.edu.ua

***Abstract*—** This paper describes an FPGA prototype of a distributed two-layer quorum-sectioned state processing model for high-throughput streams of functionally dependent events. The architectural decisions regarding the organization of quorum sections, the distribution of arithmetic operations across independent pipelines, the quorum arbitration mechanism, and the state verification procedure are presented. The prototype hardware decomposition and its operating principles on the Xilinx UltraScale+ platform are discussed

Keywords— FPGA, prototype, quorum-sectioned processing, functionally dependent events, distributed state model, arithmetic pipeline, quorum arbiter, UltraScale+

I. Introduction

Processing streams of functionally dependent events is a fundamental task in real-time systems across industrial automation, financial monitoring, and telecommunications infrastructure. A functional dependency between two events e_i and e_j is defined as a property whereby computing the state of e_i is incorrect unless the state of e_j has been accounted for, either prior to or simultaneously with e_i[1]. Violating these dependencies leads to the accumulation of systematic errors in the aggregated system state, which is unacceptable in applications such as supervisory control or financial transaction processing [2].

Software implementations running on general-purpose processors cannot guarantee deterministic latency by design, owing to non-deterministic operating system scheduling, cache coherence traffic, and memory bus contention. FPGA-based hardware implementations are free of these constraints: the pipeline structure guarantees a fixed latency per clock cycle regardless of the current system load.

An additional challenge is the fault-tolerance requirement: in safety-critical applications the system must tolerate the loss of individual compute nodes without compromising result correctness. The classical Triple Modular Redundancy (TMR) scheme [3] addresses this by replicating computations identically across all sections; however, because all sections execute the same operations, systematically correlated hardware faults remain undetectable. The proposed approach eliminates this shortcoming by functionally distributing arithmetic operations across the quorum sections, so that each section computes the same mathematical result via a structurally distinct path.

The objective of this paper is to describe the architectural decisions embodied in the FPGA prototype of the two-layer quorum-sectioned state processing system and to provide the rationale for the key design choices.

II. Prototype Architecture

The prototype is implemented on the Xilinx UltraScale+ xcvu9p-flgb2104-2-i device using Vivado 2024.1. The selection of the UltraScale+ platform is based on its high-performance architectural features and optimized resource distribution [4]. The architecture comprises two processing layers—local (L1) and global (L2)—each sectioned according to a 2-of-3 majority quorum scheme. Fig. 1 shows the overall system structure.

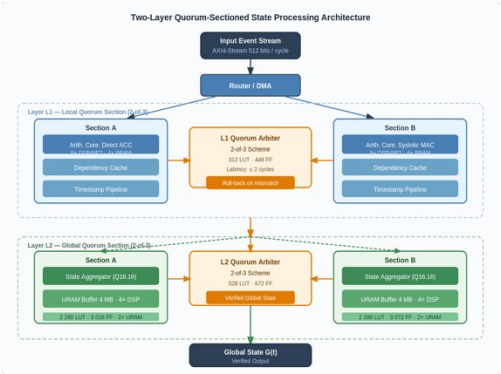

Fig. 1. Block diagram of the two-layer quorum-sectioned architecture.

A key design principle is the full hardware independence of all sections: each section has dedicated memory blocks, an independent arithmetic pipeline, and a separate clock domain driven by its own BUFG buffer. This prevents fault propagation through shared resources and is a necessary condition for quorum correctness.

The incoming event stream is received via an AXI4-Stream interface at 512 bits per clock cycle. The router distributes events across L1 sections and attaches timestamps to enforce causal ordering. The verified output

of Layer L2 is forwarded to consumers over the same interface.

III. Layer L1: Local Quorum Section

Layer L1 computes the aggregatedstate of each incoming event with respect to all functional dependencies. The state $S(e_i)$ is given by:

$$S(e_i) = \alpha \cdot V(e_i) \oplus \beta \cdot \Delta t(e_i) + \gamma \cdot \sum w(e_j, e_i), \quad (1)$$

where $V(e_i)$ is the 128-bit feature vector, $\Delta t(e_i)$ is the time offset relative to the section reference timestamp, $Dep(e_i)$ is the set of functionally dependent events, $w(e_j, e_i)$ is the dependency weight, and α, β, γ are Q8.8 fixed-point model parameters. The $\oplus$ operator is realised as bitwise XOR with masking to maintain independent feature spaces across sections.

Layer L1 contains two sections with structurally distinct arithmetic pipelines. The event processing pipeline is illustrated in Fig. 2.

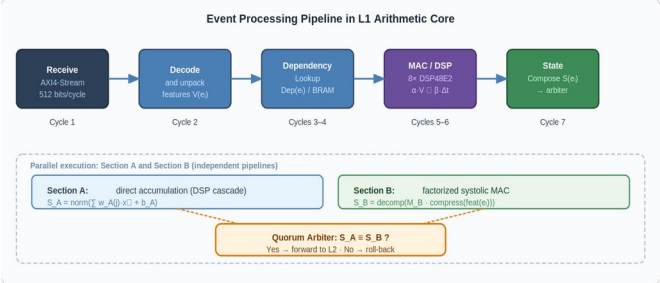

Fig. 2. Event processing pipeline in the L1 arithmetic core.

A. Section A — Direct Accumulation

Section A implements normalised weighted accumulation through a cascade of DSP48E2 blocks:

$$S\_A(e_i) = norm(\sum_j w\_A(j) \cdot x_j + b\_A), \quad (2)$$

where $feat(e_i)$ is the feature vector, w_A(j) are Section A weights, b_A is the bias, and norm(·) denotes Q16.16 fixed-point normalisation. Cascading DSP48E2 blocks eliminates intermediate registers between multiply and accumulate operations, reducing pipeline stage count [5].

B. Section B — Factorised Systolic MAC

Section B implements factorised matrix multiplication on a systolic array. The computation proceeds in two stages: projection of the feature vector onto a lower-dimensional subspace, followed by state reconstruction:

$$S\_B(e_i) = decomp(M\_B \cdot compress(feat(e_i)) + c\_B), \quad (3)$$

where M_B is a 32×128 projection matrix realised as a DSP48E2-based systolic array, c_B is the bias vector, and decomp(·) is the inverse mapping. Both (2) and (3) are mathematically equivalent for correct inputs but exhibit different numerical rounding profiles, enabling the quorum arbiter to detect silent data corruption faults undetectable in identical pipelines.

IV. Quorum Arbiter Design

A. L1 Arbiter

The L1 arbiter receives S_A and S_B upon pipeline completion and performs a 2-of-3 comparison. When results agree the state $S(e_i)$ is declared verified and forwarded to L2. On mismatch, the roll-back mechanism is activated: both sections are restored to the last verified checkpoint, an interrupt is issued to the dependency controller carrying the mismatch code, and the event is reprocessed in diagnostic mode.

The arbiter is implemented as a purely combinational circuit. Its decision latency does not exceed two clock cycles —a structural constraint imposed to maintain deterministic end-to-end pipeline latency. A mismatch accumulation register is exposed via AXI4-Lite for external monitoring.

B. L2 Arbiter

The L2 arbiter is structurally analogous to its L1 counterpart but compares aggregated global states G_A(t) and G_B(t) over a sliding time window. The update cycle corresponds to the window duration T rather than to an individual event, allowing minor differences in event arrival order between sections to be absorbed through the final equivalence of aggregated values.

V. Layer L2: Global State Aggregation

Layer L2 receives verified event states from one or more L1 instances and computes the global system state G(t) over a sliding window of width T:

$$G(t) = \sum_i \varphi_i(t) \cdot S(e_i), \quad \sum_i \varphi_i(t) = 1, \quad (4)$$

where $\varphi_i(t)$ is the time-dependent significance coefficient determined by the event position within the window and its priority in the dependency graph. All computations use Q16.16 fixed-point arithmetic, eliminating floating-point hardware and simplifying formal verification.

Each L2 section stores its aggregated state vector in URAM-288K blocks. URAM was chosen over BRAM for its higher storage density and support for pipelined reads without additional latency cycles during sequential access. The verified global state G(t) is published over the output AXI4-Stream interface with causal ordering preserved. URAM blocks were selected to facilitate high-density memory integration, ensuring low-latency data access required for state aggregation tasks [6].

VI. Dependency Controller And Timestamp Synchronisation

The dependency controller maintains the functional dependency graph in a consistent state and distributes dependency records to L1 and L2 sections. The graph is stored in sparse adjacency matrix format on BRAM blocks with atomic update support. On each incoming event the controller assembles the dependency vector $Dep(e_i)$ via a parallel CAM query, providing a fixed response time independent of graph size.

Timestamp synchronisation across sections is critical for correct $\Delta t(e_i)$ computation. The prototype implements a hardware reference counter at one-cycle resolution with symmetric broadcast to all sections. The timestamp skew between any two sections does not exceed one clock cycle, enforced by symmetric clock tree routing verified through the Vivado CDC analysis flow.

VII. Hardware Decomposition

The prototype is structured as a hierarchy of standalone IP blocks with standardised AXI interfaces, enabling independent verification and scaling. The primary subsystems are listed below, followed by the resource utilisation summary in Table I.

- event_rx — AXI4-Stream receive, header parsing, routing to sections;
- arith_core_a / arith_core_b — L1 arithmetic cores with independent pipelines and private BRAM caches;
- quorum_arb_l1 — L1 quorum arbiter, combinational with diagnostics register;
- dep_ctrl — dependency controller with CAM lookup and dynamic graph update;
- state_buf_a / state_buf_b — L2 state buffers on URAM with coherence logic;
- quorum_arb_l2 — L2 quorum arbiter with windowed state comparison;
- ts_sync — inter-section timestamp synchronisation block;
- event_tx — output AXI4-Stream block carrying verified global state.

TABLE I. Resource Utilisation Summary (Xilinx UltraScale+ xcvu9p)

Subsystem	LUT	FF	BRAM	DSP48
Arith. core L1 (Sec. A)	1 842	2 104	4	8
Arith. core L1 (Sec. B)	1 876	2 138	4	8
Quorum arbiter L1	312	448	0	0
State buffer L2 (Sec. A)	2 240	3 016	8	4
State buffer L2 (Sec. B)	2 288	3 072	8	4
Quorum arbiter L2	528	672	0	0
Dependency controller	1 104	1 440	4	2
Total	**12 986**	**16 570**	**32**	**26**

cycles, preserving deterministic pipeline latency. The dependency controller delivers a fixed dependency-vector assembly time independent of graph size.

The hierarchical IP decomposition and standard AXI interfaces enable independent subsystem replacement and scaling from a 2-of-3 to a 3-of-5 quorum configuration without redesigning the overall architecture.

Subsystem placement uses Pblock constraints: Sections A and B are placed in separate device columns to minimise cross-section routing and physically isolate critical paths. Arbiters are placed between their respective section pairs.

VIII. Functional Verification

Verification is structured across three levels. At the RTL level, each IP block is exercised in Vivado Simulator with scenarios covering: nominal processing, maximum dependency queue saturation, deliberate S_A/S_B mismatch injection to exercise roll-back, and post-roll-back recovery.

At the system level, a synthetic event generator reproduces industrial workload statistics: Poisson inter-event intervals and a power-law dependency fanout distribution [7]. At the in-system level, Xilinx ILA probes on key AXI buses capture traces analysed by automated Python scripts for protocol and timing violations.

IX. Conclusions

This paper has presented an FPGA prototype of a two-layer quorum-sectioned state processing system implemented on the Xilinx UltraScale+ xcvu9p. The main contributions are as follows.

Functional arithmetic distribution across quorum sections has been proposed and implemented. Unlike classical TMR, Sections A and B execute mathematically equivalent but structurally distinct computations, enhancing detection of systematic hardware faults.

A two-level state verification hierarchy has been designed: the L1 arbiter verifies individual events while the L2 arbiter verifies aggregated window state, together providing a two-stage correctness guarantee.

The roll-back mechanism is implemented entirely in hardware with a reaction latency of at most two clock

References

[1] L. Lamport, "Time, clocks, and the ordering of events in a distributed system," Commun. ACM, vol. 21, no. 7, pp. 558–565, July 1978.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Trans. Dependable Secure Comput., vol. 1, no. 1, pp. 11–33, 2004.

[3] B. W. Johnson, Design and Analysis of Fault Tolerant Digital Systems. Reading, MA: Addison-Wesley, 1989.

[4] Xilinx Inc., UltraScale Architecture and Product Data Sheet: Overview, DS890, 2023.

[5] Xilinx Inc., DSP48E2 Slice User Guide, UG579, 2023.

[6] Xilinx Inc., UltraRAM: Breakthrough Embedded Memory Integration on UltraScale+ FPGAs, WP477, 2022.

[7] M. Mitzenmacher and E. Upfal, Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis, 2nd ed. Cambridge: Cambridge University Press, 20