

COVER SHEET: PROVISIONAL APPLICATION FOR PATENT

**This is a provisional application for patent under 35 U.S.C. § 111(b)
and 37 CFR § 1.53(c).**

Field	Detail
Title of Invention	Systems and Methods for Zero-Deserialization Direct-Memory Generative Interface Projection and Spatial Lattice Synchronization
Inventor	Brennan William DeCrow, Northridge, California, United States
Correspondence Address	18617 Chase Street, Northridge, CA 91324, United States
Attorney Docket Number	MM-GENUI-PROV-004
Related Disclosures	U.S. Provisional Application No. 64/159,586 (filed September 22, 2026), U.S. Provisional Application No. 64/161,921 (filed September 24, 2026), and MM-SLASH-PROV-003, by the same inventor.
Entity Status	Micro-Entity (Certification Form PTO/SB/15A)
Government Interest	None.

This document serves as the provisional application cover sheet under 37 CFR § 1.51(c)(1). Brennan William DeCrow is the sole inventor identified for this application. No attorney or agent is appointed. No U.S. Government agency has a property interest in the invention. The applicant requests treatment as a provisional application under 35 U.S.C. § 111(b) and 37 CFR § 1.53(c). If the filing fee is not paid on submission, the applicant will timely submit the required filing fee and any applicable late surcharge in response to the Office's notice.

Signature: _____ Date: September 25, 2026

Brennan William DeCrow

Sole Inventor & Pro Se Applicant

SPECIFICATION: SYSTEMS AND METHODS FOR ZERO-DESERIALIZATION DIRECT-MEMORY GENERATIVE INTERFACE PROJECTION AND SPATIAL LATTICE SYNCHRONIZATION

1. Technical Field

[0001] This disclosure relates generally to graphical user interfaces and computer systems architecture, and more particularly to systems, methods, and non-transitory computer-readable media for zero-deserialization, direct-memory generative interface projection, zero-allocation DOM state binding, and deterministic spatial lattice synchronization across heterogeneous browser and native execution contexts.

2. Background of the Invention

[0002] Modern generative user interfaces (GenUI) and web-based application frontends (such as those built with React, Vue, Angular, or Electron) suffer from crippling architectural bottlenecks when attempting to synchronize high-frequency state updates, machine-planner mutations, or simulation telemetry with visual browser surfaces.

[0003] In conventional web architectures, any state mutation produced by a backend process, AI agent, or physics simulation must undergo a multi-stage translation pipeline: 1. The backend serializes binary state into an intermediate text representation (typically JSON or XML); 2. The serialized text is transferred across an IPC, WebSocket, or HTTP boundary; 3. The browser engine parses the text, allocating thousands of transient JavaScript heap objects; 4. A virtual DOM tree is constructed and traversed to compute differences against a prior virtual DOM tree; 5. Mutated nodes are scheduled for browser layout and repaint.

[0004] This traditional pipeline introduces severe latency (often exceeding 50–100 milliseconds), catastrophic garbage collection (GC) pauses, and frame drops. When an AI planner or high-frequency telemetry stream generates hundreds of mutation events per second, existing frontend architectures freeze or fail to maintain interactive frame rates (e.g., 60 FPS or 120 FPS).

[0005] Furthermore, existing multi-window or cross-iframe web applications rely on fragmented storage mechanisms (such as `localStorage` string serialization) that lack atomic synchronization, zero-copy memory mapping, and cryptographic provenance guarantees.

[0006] Accordingly, there is an urgent technical need for a zero-deserialization, direct-memory generative interface projection architecture that binds visual DOM elements directly to unified binary memory slabs without intermediate object allocation, serialization overhead, or virtual DOM reconciliation.

3. Summary of the Invention

[0007] To overcome these deficiencies, the present disclosure provides systems, architectures, and methods for:

[0008] (1) Direct-Memory Unified Slab Projection:

A unified memory slab (comprising a preallocated `SharedArrayBuffer` or memory-mapped file) is established across execution contexts. Mutation payloads are written directly into pre-aligned binary memory slots at raw byte offsets. A lightweight registry indexes payload blocks by stable identifiers. Front-end visual elements query raw memory pointers (`getBinaryPointer(id)`) and read typed numeric fields directly in place, completely eliminating JSON serialization, string parsing, and intermediate object allocation.

[0009] (2) Zero-Allocation DOM Attribute and Shader Binding:

Visual web components bind directly to binary memory offsets. Updates to scalar telemetry values (such as phase angles, progress counters, latency metrics, and color channels) write directly to underlying CSS custom properties or WebGPU/WebGL uniform buffers, bypassing DOM tree re-construction and virtual DOM reconciliation.

[0010] (3) Deterministic Living Daylight Lattice:

Visual surfaces are structured upon an invariant daylight lattice that maps UI geometry to perceptual color spaces (such as OKLCH) and hardware display coherence units. The spatial lattice synchronizes visual HUD cards, daylight travel markers, and recessed wells directly to cryptographically stamped spine bus events.

[0011] (4) Cross-Room Entanglement Spine:

A cross-context spine bus enables multiple isolated browser windows, iframes, and worker threads to share the unified memory slab. When a binary mutation

occurs in one surface, a lightweight hardware or browser event fires with the raw memory pointer, allowing peer surfaces to observe the update synchronously with zero network transit latency.

4. Detailed Description of Preferred Embodiments

4.1 Direct Memory Layout and Pointer Registry

[0012] Figure 1 depicts the direct memory architecture of the Maglev Spine Bus. A unified memory slab $\mathcal{M}$ of size S_{slab} (e.g., 10 megabytes) is preallocated at initialization. A `DataView` provides structured typed access across byte boundaries.

[0013] Ingestion of a binary state payload B of length L_B executes as:

$$\text{offset}_{\text{start}} = \text{head}; \quad \mathcal{M}[\text{offset}_{\text{start}} \dots \text{offset}_{\text{start}} + L_B - 1] \leftarrow B; \quad \text{head} \leftarrow \text{head} + L_B$$

The registry maps stable identifier id to tuple $(\text{offset}_{\text{start}}, L_B)$.

[0014] When a visual element requests state for id , the spine returns the raw pointer tuple without constructing any intermediate representation. A WebGL/WebGPU shader or WebComponent consumes the raw bytes directly via DMA streaming into GPU VRAM.

4.2 WebComponent Direct State Binding

[0015] Custom DOM elements (e.g., `ManySpineFeed`) register listeners for the zero-copy ingest event (`maglev:spine:ingest`). Upon receipt of the event notification, the element reads the binary payload in place, updates target text content or layout transforms, and completes rendering within a single display refresh cycle (less than 8.33 milliseconds at 120 Hz).

CLAIMS: SYSTEMS AND METHODS FOR ZERO- DESERIALIZATION DIRECT-MEMORY GENERATIVE INTERFACE PROJECTION AND SPATIAL LATTICE SYNCHRONIZATION

- 1.** A computer-implemented interface projection system comprising: a preallocated unified memory slab established in a shared memory region accessible across a plurality of execution contexts; a pointer registry indexing binary state records in the preallocated unified memory slab by stable identifiers; and processing circuitry configured to: receive a binary state payload comprising mutation data produced by an execution thread; write the binary state payload directly into the preallocated unified memory slab at an offset determined by a write pointer, without text serialization or object instantiation; register the offset and a byte length of the binary state payload in the pointer registry; dispatch a zero-copy ingestion event identifying the stable identifier to a graphical user interface renderer; and in response to the zero-copy ingestion event, cause the graphical user interface renderer to read the binary state payload directly from the offset in the preallocated unified memory slab and project visual interface elements onto a display surface without performing string parsing or virtual DOM tree reconciliation.
- 2.** The system of claim 1, wherein the preallocated unified memory slab comprises a `SharedArrayBuffer` shared between a main browser thread and at least one WebWorker thread.
- 3.** The system of claim 1, wherein reading the binary state payload directly from the offset comprises streaming bytes of the binary state payload directly into a GPU uniform buffer via direct memory access (DMA).
- 4.** The system of claim 1, wherein projecting visual interface elements comprises updating CSS custom properties associated with a DOM element directly from typed numerical fields read from the binary state payload.
- 5.** The system of claim 1, wherein the visual interface elements are positioned upon an invariant daylight lattice, wherein colors of the visual interface elements are defined in an OKLCH perceptually uniform color space.
- 6.** A method for projecting generative interface state directly from memory, the method comprising: allocating a unified binary memory buffer in processing circuitry; receiving a binary mutation block from a computational process; storing the binary mutation block at a sequential address within the unified

binary memory buffer; updating an in-memory index mapping an entity identifier to the sequential address; broadcasting a notification carrying the entity identifier across an inter-context event bus; and retrieving, by a web component in response to the notification, a direct pointer to the sequential address from the in-memory index, and rendering graphical changes directly from memory at the sequential address without constructing an intermediate JSON representation.

7. The method of claim 6, wherein the computational process comprises an AI planner or a physical simulation loop.

8. The method of claim 6, wherein broadcasting the notification crosses an iframe boundary or a multi-window browser boundary.

9. The method of claim 6, wherein rendering graphical changes executes at a sustained frame rate of at least 60 frames per second with zero garbage collection allocations on the render loop.

10. A non-transitory computer-readable medium storing instructions that, when executed by processing circuitry, cause the processing circuitry to perform operations comprising: maintaining a shared memory buffer containing binary telemetry state; ingesting binary state updates into the shared memory buffer at contiguous offsets; emitting a zero-copy event referencing an update identifier; and updating visual DOM nodes and GPU shader uniforms directly from the shared memory buffer at the contiguous offsets without intermediate object deserialization.