Design and Emulation Methodology for Atomic-Scale Systolic Arrays: An LLM Accelerator Case Study in Silicon DB Logic

Samuel S. H. Ng, Marcel Walter, Simon Hofmann, Jan Drewniok, Robert Wille, and Konrad Walus

***Abstract*— LLMs increasingly strain memory bandwidth and compute resources as CMOS scaling plateaus. Emerging technologies such as atomic-scale computing with silicon dangling bonds (DBs) promise ultra-dense, low-power logic, yet application-level validation still lacks an executable, clock-driven hardware emulation framework. To address this gap, this work introduces a cross-layer flow that compiles register-transfer level (RTL) Verilog to a clock-driven, Verilator-based emulator exposed to Python via a co-simulation hardware abstraction layer (HAL). DB-aware RTL rules formalized in this work ensure representative emulation across the full systolic array, while allowing the same RTL to drive logic synthesis through *fiction*, a technology-specific EDA toolkit, to yield dot-accurate DB layouts. As a representative use case, a ternary DB matrix multiply unit (MXU) is designed in Verilog to target BitNet b1.58 acceleration, achieving up to $34\times$ area reduction compared to prior DB MXUs and generating LLM tokens under cycle-accurate software emulation while matching GPU-baseline outputs. This bridges layout-centric studies and workload-driven evaluation, enabling reproducible, cross-layer accelerator design for this emerging technology.**

I. Introduction

At a time when LLMs push compute and energy demands to unprecedented levels, efforts to miniaturize CMOS are increasingly stifled by engineering and economic headwinds. This tension motivates the exploration of emerging technologies that offer alternative means to scale computational density while maintaining reasonable power envelopes. Atomic-scale computing, which realizes digital logic by positionally encoding bit information in quantum dots made of silicon dangling bonds (DBs) on hydrogen-terminated silicon [1], [2], offers a potential path to higher device densities under strict locality and clocking constraints. As a member of the broader field-coupled nanocomputing (FCN) family, atomic-scale computing encodes and manipulates information through local electrostatic interactions and propagates signals under the clocking effects of nearby electrodes [3], [4], resulting in strictly local switching activity. This shifts the dominant energy dissipation from CMOS's through-current conduction to the adiabatic clocking network, enabling low-power operation [1], [4].

The experimental demonstration of a DB OR gate measuring just $5 \times 6\,\mathrm{nm}^2$ [1] catalyzed the development of CAD tools such as *SiQAD* [3] and specialized EDA frameworks such as *fiction* [5]–[7], enabling DB logic research from the gate and circuit level [8] to the application-design level [9]–[11]. Notably, [11] implemented a quantized matrix multiply unit (MXU) in register-transfer level (RTL) Verilog and demonstrated a semi-automated pipeline capable of synthesizing it to a dot-accurate DB layout ready for fabrication, bringing the field closer to validating DB-based ML accelerators. However, open questions remain about the practicality and benefits of developing and integrating DB accelerators into existing CMOS fabrics, in part because practical hardware emulation is lacking, leaving crucial gaps in quality assurance, validation under real workloads, and application-level hardware co-design.

To this end, a workflow is proposed that compiles RTL to a clock-driven software emulator using Verilator [12] and integrates it with practical ML inference tasks. A set of DB-aware RTL design rules is formalized to ensure adherence to four-phase clocking constraints and nearest-neighbor pipeline connectivity, enabling the same RTL to serve emulation and synthesis. As a representative workload, BitNet b1.58 [13] is targeted, employing ternary weights $w \in \{-1, 0, 1\}$ and quantized integer activations to achieve significant reductions in hardware complexity [13]–[15]. Its arithmetic and data-flow patterns map naturally to the deeply pipelined operations of atomic-scale computing. Beyond its hardware fit, BitNet b1.58 offers competitive LLM accuracy at substantially lower compute and memory bandwidth than higher-precision baselines, with publicly available pre-trained models and reference code enabling reproducible, application-level evaluation. Coupled with an RTL-to-DB synthesis flow from [11] and optimizations to the placement-and-routing algorithm contributed by this work, the same Verilog is used to seed an end-to-end stack for the first time in atomic-scale computing: BitNet b1.58 inference is driven through a low-overhead co-simulation hardware abstraction layer (HAL) via Python bindings atop a Verilator-generated C++ model, and dot-accurate DB layouts are generated via *Yosys* [16], *ABC* [17], and the *fiction* framework [5].

II. Background

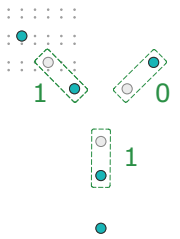
This section reviews foundational background topics, including the rapidly evolving CAD and EDA ecosystem for DB logic and an introduction to BitNet b1.58 LLMs.

A. Atomic-Scale Computing with Silicon Dangling Bonds

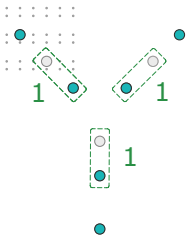
Silicon DBs are atomic-scale quantum dots fabricated with a scanning tunneling microscope on a hydrogen-passivated

S. Ng and K. Walus are with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada ({samueln, konradw}@ece.ubc.ca); M. Walter, S. Hofmann, J. Drewniok, and R. Wille are with the Chair for Design Automation, Technical University of Munich, BY, Germany ({marcel.walter, simon.t.hofmann, jan.drewniok, robert.wille}@tum.de). M. Walter, S. Hofmann and R. Wille are also with the Munich Quantum Software Company GmbH, Garching near Munich, BY, Germany. R. Wille is also with the Software Competence Center Hagenberg GmbH, Hagenberg, OÖ, Austria.

(a) A wire constructed from BDL pairs transmitting a logical `1`.



(b) BDL OR gate: `10` inputs.



(c) BDL OR gate: `11` inputs.

Fig. 1. Logical encoding and gate implementation in the DB platform, experimentally demonstrated in [1] and recreated using *SiQAD* [3].

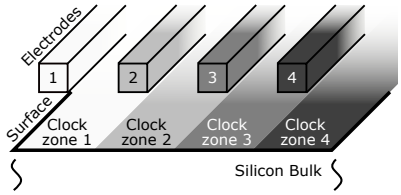


Fig. 2. A four-phase clocking scheme for DB circuits. A full cycle over four zones corresponds to one logical pipeline stage. Adapted from [6].

silicon surface [2]. Each DB can be negatively, neutrally, or positively charged, forming the basis for computation [1]. Logical bits are encoded in binary-dot logic (BDL) pairs, where the position of a shared negative charge between two proximal DBs represents `0` or `1` (Fig. 1a) [1]. Wires and logic gates are formed from arrangements of BDL pairs (Fig. 1), whose complex electrostatic interactions can be modeled and validated using CAD tools like *SiQAD* [3].

To enable sequential logic and directed data flow, DB circuits are partitioned into clock zones controlled by underlying electrodes [3], [4] (Fig. 2). A four-phase clocking scheme, driven by oscillating voltages, creates a propagating electrostatic field that ensures unidirectional signal flow. Due to electrostatic constraints, adjacent zones cannot simultaneously hold and latch distinct data, requiring intermediate buffer zones. This physical requirement imposes a key architectural rule: a full sequence of four clock zones is required to advance a signal, which directly translates to one logical pipeline stage at the RTL boundary. Interfacing with CMOS logic is also handled electrostatically: inputs are written by CMOS-controlled electrodes [4], and outputs are read by single-electron transistors that detect the output state [18].

B. Electronic Design Automation

A fundamental EDA tool for atomic-scale computing is the *fiction* [5] framework, which provides technology-independent algorithms for logic synthesis, placement, routing, clocking, and verification. Standard tile libraries such as the *Bestagon* [6] gate library enable technology-specific implementations for DBs, while notable physical design algorithms include the scalable heuristic *ortho* [19] for fast

```
module full_adder(input  x, y, cin, output s, cout);
assign {cout, s} = x + y + cin;
endmodule
```

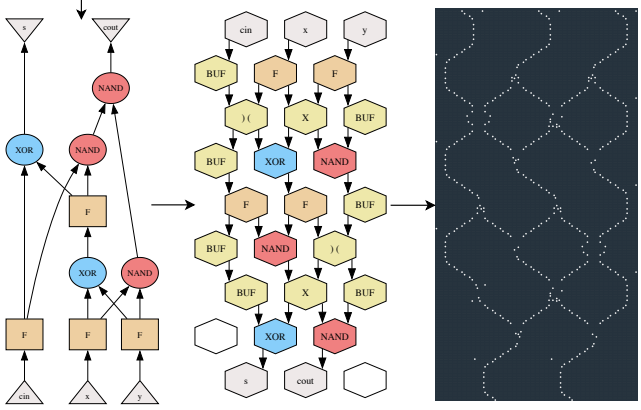


Fig. 3. Full adder design flow: a behavioral Verilog description is synthesized and technology-mapped to a logic network using only gates available in the *Bestagon* library [6], placed and routed as a gate-level layout, and realized by replacing each gate with its *Bestagon* implementation.

layout generation, the multiobjective graph-oriented layout design (*gold*) [20] algorithm for A*-guided placement and routing, post-layout optimization (*PLO*) [21] techniques for layout refinement, and *hexagonalization* [7] for technology transformation between different FCN implementations. As an example, Fig. 3 illustrates the generation of atomic-scale layouts using *fiction*: a Verilog full adder description is synthesized to the gate level, a hexagonal layout is placed and routed, and the gates are replaced with their cell-level implementations from the *Bestagon* library. Importantly, not all stages are performed by *fiction*: *Yosys* [16] typically performs the initial high-level synthesis of behavioral Verilog, and subsequent logic optimizations and technology mapping are carried out with *mockturtle* and *ABC* [17], which *fiction* also offers as a callback; the mapped netlist is then consumed by *fiction* for the application steps of technology-specific physical design, verification, and cell-library application. One design constraint is that *fiction* supports only combinational layout synthesis [5].

C. BitNet b1.58 LLMs

Beyond the DB technology reviewed above, this paper employs a recently proposed LLM, BitNet b1.58. Higher-precision quantized accelerators (e.g., $8\,\mathrm{bit}$) are widely deployed in production [22], delivering latency and energy gains; by contrast, BitNet b1.58 constrains layer weights to the ternary set $\{-1, 0, +1\}$ with integer activations quantized to $8\,\mathrm{bit}$ [13], [14] or $4\,\mathrm{bit}$ [15]. Because each elementwise product $w \cdot a$ collapses to $\{-a, 0, +a\}$, dot-products require only additions/subtractions/skips with integer accumulation. Simplified arithmetic reduces both memory bandwidth and multiply-accumulate (MAC) energy consumption. The arithmetic and data flow patterns map naturally to deeply pipelined, clocked DB logic. Compared to a $16\,\mathrm{bit}$ floating point baseline at $700\,\mathrm{M}$ parameters, BitNet b1.58 uses $2.6\times$

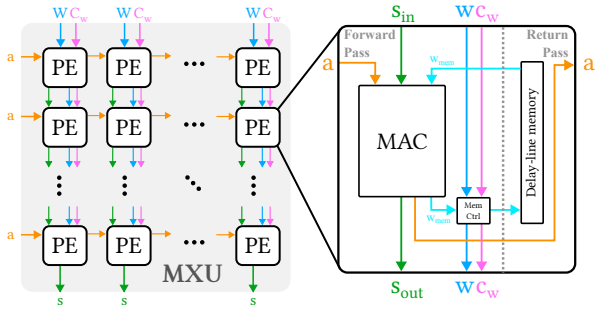


Fig. 4. A systolic array MXU with activations (a), weights (w), control signals (c_w), and partial sums (s). In a PE, there is a MAC unit and memory controller on the forward path, and a delay-line memory on the return path.

less memory and $\approx 71\times$ lower arithmetic energy on a $7\,\mathrm{nm}$ energy model with minimal accuracy loss [13]. In this work, BitNet b1.58 therefore serves as a representative, publicly available workload that both stresses the proposed MXU and enables end-to-end reproducibility.

III. Related Work

The advent of specialized EDA tools and frameworks has lowered the barrier to DB logic research. Within the DB literature, much of the prior work still emphasizes gate- and circuit-level designs [8], with comparatively fewer studies at the system level. At somewhat higher abstraction levels, related FCN workflows have been explored in nanomagnetic logic, where *ToPoliNano* maps VHDL descriptions to clocked layouts and *FUNCODE* derives HDL netlists from custom layouts for simulator-based analysis [23], [24], but since those methods target a different device technology altogether, they do not address DB-specific device and clocking constraints. Against this backdrop, early application-scale analyses of DB systems relied on hand-crafted DB building blocks to extrapolate area and latency [9], [10]. In particular, [10] presented an MXU for matrix-vector multiplication (MVM) using quantized $8\,\mathrm{bit}$ weights and activations (W8A8) implemented as a 2-D systolic array of identical PEs (Fig. 4). The array operated in two phases: a *load* phase that streamed weights downward for storage, and a *compute* phase that streamed activations left-to-right while accumulating partial sums column-wise. The PE micro-architecture organized signals into a forward and return path (Fig. 4). The forward path performed MAC, multiplying the weight and activation and accumulating the product with the incoming partial sum to yield an updated partial sum, and controlled a local weight delay-line; the return path time-aligned activations for the neighbor and closed the delay-line loop for weight storage. The design embraced deep pipelining to reflect DB clocking constraints and suggested that multiple payloads could be interleaved across pipeline stages for high utilization; however, comprehensive array-level validation remained an open question.

Working within *fiction*'s combinational synthesis capabilities, a subsequent study [11] partitioned the MXU design in [10] into hierarchical RTL Verilog definitions, including:

1) A *combinational* core in which the forward-path logic is a single block, including MAC and state/memory control, for Verilog-to-DB synthesis; and
2) A *clocked* shell that wraps the core, implements forward and return pipelines via registers, and provides a self-contained PE interface for systolic-array integration.

In that work, the authors validated the clocked shell using Verilog testbenches and synthesized the combinational core to a physical DB layout with the following flow:

1) The RTL was synthesized to an and-inverter graph (AIG) using *Yosys* [16] and optimized using *ABC* [17].
2) The optimized AIG was loaded into *fiction* [5].
3) Technology mapping was used to rewrite the network with Boolean gates available in the Bestagon library [6].
4) The network was placed and routed onto a Cartesian grid using the *ortho* algorithm [19].
5) *PLO* reduced the size of the layout [21].
6) The layout was remapped onto a hexagonal grid [7].
7) The Bestagon gate library was applied to the layout [6], producing a dot-accurate DB layout.

Collectively, the hierarchical Verilog provided a single source of truth, producing DB layouts for the physical design flow and enabling PE-level logical validation. However, the implementation was confined to the PE layer and did not include a full systolic array, leaving full MVM validation incomplete. In addition, the W8A8 netlist exceeded the scaling limits of *fiction*'s recent placement-and-routing algorithms, such as *gold* [20], precluding their use and the associated improvements in area and routability. Prior work likewise did not articulate DB-aware RTL design rules that reflect four-phase clocking at register boundaries or provide scheduling mechanisms at the systolic-array level for parallel MVM jobs. Finally, the absence of a cycle-accurate emulator hindered integration with real inference pipelines, preventing systematic end-to-end evaluation. This gap is particularly notable when compared to parallels in the mature CMOS domain, such as the *Gemmini* generator, where synthesizable RTL for systolic-array accelerators can be coupled with Verilator simulation [25]. However, the present work targets DB-specific hardware requirements and a *fiction*-focused RTL flow, limiting the overlap. These shortcomings are addressed in the following section.

IV. Proposed Methodology

This section presents the primary contribution: a clock-driven emulation and co-simulation framework for BitNet b1.58 on DB logic. It covers the hardware architecture and DB-aware RTL design rules, an adapted RTL-to-DB flow, and a co-simulation HAL atop a Verilator model [12] that schedules interleaved MVM jobs and enables end-to-end BitNet b1.58 LLM inference on the emulated hardware.

A. BitNet b1.58 Accelerator Design

Following the design methodology established in the literature [10] and reviewed in Section III, the accelerator for BitNet b1.58 MVM adopts a 2-D systolic array of homogeneous PEs organized as a hierarchical RTL structure [11].

Compared with W8A8, BitNet b1.58 requires PE changes for ternary weights. Signed $8\,\mathrm{bit}$ weights are replaced with a two-bit encoding, and the MAC no longer uses a multiplier: for each weight-activation pair, the weight either applies a sign to the activation ($w = \pm 1$) or suppresses the product ($w = 0$) before accumulation with the partial sum. Otherwise, the PE architecture remains consistent with the W8A8 counterpart [10]: the forward path implements arithmetic and control logic, and the return path aligns signals.

The pipelined nature of DB logic (Section II) means that the PE's forward and return paths must be split into balanced pipelines. The total pipeline depth $P = P_{\text{forward}} + P_{\text{return}}$ must be informed by the synthesized physical path length together with the four-phase rule (four clocking zones per logical stage per Section II-A). P_{forward} is therefore estimated to be

$$P_{\text{forward}} = \left\lceil \frac{H_{\text{f}} + H_{\text{r}}}{4\,p_{\text{e}}} \right\rceil, \tag{1}$$

with H_{f} the synthesized forward-path height, p_{e} the inter-electrode pitch, and H_{r} an allowance for additional routing at the PE level (pin escape to PE edges and spacing between forward/return paths). With balanced paths, $P = 2\,P_{\text{forward}}$.

Above the clocked PE shell, a clocked systolic array module is defined to set the array dimensions and nearest-neighbor inter-PE links. Inputs and outputs are time-aligned to enable interleaving of multiple payloads across pipeline stages; each interleaving position is termed a job slot. With a PE pipeline depth of P, the array exposes P concurrent job slots for MVM operations. This systolic array module completes the BitNet b1.58 MXU hardware with configurable rows, columns, and P, validated by unit- and system-level testbenches. The array-level RTL is then compiled with Verilator [12] to a C++ model for clock-driven emulation.

This hierarchy supports design rules that carry through to both flows: pipeline registers partition long paths to mirror the clock-driven behavior of DB logic; communication is restricted to fixed row-wise and column-wise nearest neighbors; and pipeline stages are carefully aligned to enable highly parallel, interleaved operation. Under these constraints, the same PE RTL compiles to dot-accurate DB layouts and yields a faithful, clock-driven model. The approach extends to broader FCN applications, establishing an end-to-end, reusable, and scalable design framework.

B. RTL-to-DB Synthesis

Adopting a synthesis workflow similar to [11] and reviewed in Section III, the simpler PE for the BitNet b1.58 MXU enables optimizations not applicable in that setting. In the prior MXU, the W8A8 multiplication requires full-precision multipliers and more complex weight-memory control, whereas the ternary operator eliminates explicit multipliers and simplifies control. This sufficiently shrinks the netlist generated by *Yosys* [16] and *ABC* [17] to fall within the practical problem size of fiction's *gold* [20] algorithm for placement and routing, which previously could not scale to the W8A8 design. Consequently, this work explored the use of *gold* in place of *ortho* [19] for A*-guided placement and congestion-aware routing with the goal of achieving improved routability and reduced area.

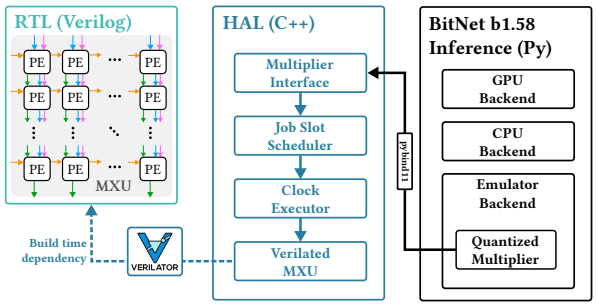


Fig. 5. Overview of the RTL-to-emulation workflow for BitNet b1.58. Arrows indicate the call direction at runtime.

Despite the smaller netlist from the BitNet MXU, initial attempts to place and route using *gold* resulted in a high failure rate, with no valid solutions achieved within the preset time limit. This work extends the algorithm with a configurable primary-input (PI) spacing option to spread inputs and reduce local wire congestion, improving routability. The inter-PI spacing can be randomized, allowing more diverse placement strategies to be explored and potentially yielding higher-quality layouts. The improvements increase the probability of finding routable layouts at the expense of a temporarily larger area footprint, which can be subsequently compensated by *PLO*. These combined adjustments were employed in placement-and-routing experiments in Section V-A.

C. Co-simulation of BitNet b1.58 with LLMs

In Section IV-A, a clock-driven C++ model of the systolic array was established by translating the RTL via Verilator [12]. However, using the raw model directly requires applications to orchestrate clocked pushes and reads. Therefore, a low-overhead co-simulation HAL is introduced that abstracts the clocking protocol and exposes a concise application-programming interface for MVM, enabling practical co-simulation with existing LLM inference pipelines. An overview of the workflow is provided in Fig. 5. Simulator lifetime and the array's clocking lifecycle are managed by the HAL, which issues pre- and post-clock actions for clocked data ingress and egress, and advances the clock once required actions in every pipeline state of all PEs are accounted for. To maximize utilization, a queue of MVM jobs is maintained, and job slots are filled as they become available. Access to job slots is time-aligned based on a monotonically increasing global clock. This alignment-driven scheme keeps slots continuously occupied under realistic loads without exposing low-level scheduling mechanics to the caller.

Beyond scheduling, the HAL provides integration conveniences for application codebases. Because real workloads often present matrix-vector shapes that differ from the compiled array dimensions, problems of arbitrary size are automatically decomposed into jobs that fit within the configured systolic array, and their output partial sums are aggregated into the final outputs. A *pybind11* wrapper is also created to expose the HAL to Python callers, enabling

TABLE I
EVALUATION OF SYNTHESIS APPROACHES.

EXPERIMENT	ALG.	$w \times h$	=	A	DB COUNT
W8A8 [11]	*ortho + PLO*	515×1043	=	537 145	1 645 777
BitNet b1.58	*ortho*	154×372	=	57 288	263 431
(This work)	*ortho + PLO*	98×233	=	22 834	178 778
	gold	87×190	=	16 530	133 882
	gold + PLO	86×184	=	15 824	132 041

w = width, h = height, A = resulting area (in tiles) of the layout.

inference workloads to redirect MVM calls to the wrapper for direct emulation. To maintain a single source of truth across layers, an automated build process propagates configuration parameters—such as array dimensions and pipeline depth—from Verilog through the generated C++ model and the HAL to the Python bindings, ensuring consistent binaries.

An open-weight BitNet b1.58 repository providing pretrained checkpoints and reference scripts by 1bitLLM [26] serves as the starting point for this work's implementation. When the emulator backend is selected, the repository's MVM and quantization primitives are replaced by the Python wrapper over the co-simulation HAL; CPU or GPU execution continues to use the original *PyTorch*-based inference path. Whereas the reference simulates quantization in float and then applies a floating-point MVM followed by bias addition, the co-simulation keeps arithmetic in the integer domain on the emulator via the HAL. In this arrangement, $\mathbf{y}_{\mathrm{int}}$ is computed within the HAL and passed to Python, which then applies dequantization and bias to obtain $\mathbf{y}_{\mathrm{fp}}$. Concretely, with $8\,\mathrm{bit}$ activations $\mathbf{x}_q$ and ternary weights $\mathbf{W}_q$, together with an activation scale s_x and a weight scale s_w:

$$\mathbf{y}_{\mathrm{int}} = \mathbf{W}_q\,\mathbf{x}_q \in \mathbb{Z}^m, \qquad \mathbf{y}_{\mathrm{fp}} = \frac{1}{s_x\,s_w}\,\mathbf{y}_{\mathrm{int}} + \mathbf{b}. \tag{2}$$

This is algebraically equivalent to the reference while keeping the emulator strictly integer.

Together, the RTL design rules, improved RTL-to-DB synthesis tooling, and a clock-driven, Verilator-based emulator with a co-simulation HAL are key contributions that enable, for the first time, a unified path from RTL to dot-accurate layouts and clock-driven, end-to-end BitNet b1.58 LLM inference. The contribution is therefore methodological rather than predictive: this work provides the infrastructure into which future, better-calibrated device- and system-level energy models and interfaces can be integrated to enable quantitatively grounded, production-oriented evaluation.

V. EXPERIMENTS AND DISCUSSION

This section evaluates the proposed methodology, covering synthesis of the PE core into DB layouts, the BitNet b1.58 emulation configuration, and co-simulation runtime.

A. RTL-to-DB Synthesis Results

Using the optimized synthesis workflow proposed in Section IV-B, multiple DB layouts were synthesized using various combinations of placement-and-routing algorithms. The synthesis results of the BitNet b1.58 MXU are presented in Table I, which details key metrics for each algorithm used. For consistency with the present methodology, the W8A8 MXU metrics from [11] were recomputed and included.

Across equivalent settings, the BitNet b1.58 accelerator synthesizes to substantially smaller DB layouts than the prior W8A8 MXU with a $23.4\times$ area reduction when placed-and-routed with *ortho + PLO*. Importantly, while the W8A8 MXU exceeded *gold*'s tractable problem size, BitNet b1.58's lean arithmetic requirements (Section IV-A) produce a sufficiently small netlist for *gold*'s scaling regime. Together with *PLO*, the area reduction increases to $34\times$, yielding a substantially smaller clocked layout footprint.

B. BitNet b1.58 Emulator Setup

With the established physical DB layout, the experimental configuration and evaluation protocol for clock-driven co-simulation of LLM inference are described next. Per Section IV-C, the key configurable parameters for the BitNet b1.58 MXU are: 1) systolic array dimensions; and 2) PE pipeline depth. Accordingly, arrays from 16×16 to 128×128 are evaluated, covering reported configurations [22], [25]. Using Eq. (1), by setting $H_{\mathrm{f}} = 2407\,\mathrm{nm}$ (minimum synthesized height achieved), $p_{\mathrm{e}} = 53.76\,\mathrm{nm}$ in alignment with [10] and consistent with $14\,\mathrm{nm}$ fabrication rules, and $H_r = 0$ due to the current lack of a procedure for routing the PE's pins to its outer boundaries, the resulting pipeline depth for a full PE is $P = 24$. Clock-driven emulation at this depth was found to be computationally intensive; therefore, experiments use $P = 8$ for tractable validation. This choice maintains deeply pipelined operation and exercises the interleaved job-slot scheduler, which suffices to validate correctness and parallelism while keeping runtimes tractable.

All experiments used the BitNet b1.58 700 M checkpoints from 1bitLLM [26] (Section IV-C). End-to-end validation of the emulator used a curated set of short text-completion prompts consisting of single-clause stems of roughly 3–6 tokens targeting simple facts and arithmetic (e.g., "The capital of France is"). Each benchmark decodes 4 new tokens per prompt for runtime considerations, with greedy decoding and deterministic seeding applied to ensure consistency. Three backends were evaluated: a GPU reference, a CPU reference, and the Verilator-generated model driven through the co-simulation HAL described in Section IV-C.

C. BitNet b1.58 Co-simulation Evaluation

Table II summarizes the collected metrics for the tested array sizes averaged across all benchmark prompts. Within the emulation workflow, a text-match accuracy of $100\,\%$ relative to the GPU baseline was achieved across all prompts, validating functional equivalence for the tested cases. Performance-wise, f_{emul} decreases with array size, as expected from the larger simulated state. An inverse relationship between jobs per token and array size is also observed, as smaller arrays require finer segmentation of the workload. Thus, t_{token} lies in the $\approx$1–4 hour range depending on array size, achieving functional validation and hardware–software co-design within tractable runtimes.

TABLE II

HOST-SIDE CO-SIMULATION METRICS FOR BITNET B1.58.

CASE	f_{emul}[kHz]	JOBS/TOKEN	t_{token}[s]	t_{wall}[s]
GPU	–	–	0.09	0.35
CPU	–	–	7.51	30.04
Emulator Array				
16×16	20.89	2 654 208	5416.71	21 666.82
32×32	19.15	663 552	3435.02	13 740.07
64×64	4.90	165 888	6384.47	25 537.89
128×128	0.97	41 472	14 217.98	56 871.93

Metrics are averaged across prompts; emulator rows report Verilator host runtimes; f_{emul} = effective emulated clock rate; t_{token} = per-token decoding latency, t_{wall} = end-to-end wall time; an AMD Ryzen 9 9950X3D CPU and an Nvidia GeForce RTX 5090 GPU were used.

Overall, the synthesized MXU layouts and end-to-end LLM inference on the HAL-driven emulator confirm application-level validation and compact, dot-accurate layouts. The evaluation is methodological, not a full-system power, fabrication/yield, or larger-model study. Combined with the abstraction of the clock-driven protocol and practical guidance on pipeline depth and job interleaving, this capability is expected to move DB logic research beyond isolated case studies toward reproducible, workload-driven system design. The source code for the implementation is available in the public repository [27].

VI. CONCLUSION

Application-level validation for atomic-scale computing has been limited by the lack of executable, clock-driven models for real workloads. This work establishes a reproducible cross-layer path from synthesizable RTL to a Verilator-based, HAL-driven emulator and dot-accurate DB layouts, while preserving a single source of truth through DB-aware RTL design rules. Text completions on BitNet b1.58 matched the GPU baseline for the tested cases, validating functional behavior for hardware–software co-design, and improvements to *gold* produced layouts up to $34\times$ smaller than prior W8A8 MXU designs. Future work on feedback-path synthesis and calibrated energy models can extend this framework and help move DB accelerator research from isolated layouts toward reproducible, workload-driven system design.

ACKNOWLEDGMENT

This work used LLMs from Anthropic and OpenAI for assistance on writing and coding. All outputs were verified by the authors, who assume full responsibility for the content.

REFERENCES

[1] T. Huff et al., "Binary atomic silicon logic," *Nature Electronics*, vol. 1, no. 12, pp. 636–643, 2018.

[2] J. Pitters et al., "Atomically Precise Manufacturing of Silicon Electronics," *ACS Nano*, acsnano.3c10412, 2024.

[3] S. S. H. Ng et al., "SiQAD: A design and simulation tool for atomic silicon quantum dot circuits," *IEEE Transactions on Nanotechnology*, vol. 19, pp. 137–146, 2020.

[4] H. N. Chiu et al., "PoisSolver: A tool for modelling silicon dangling bond clocking networks," in *2020 IEEE 20th International Conference on Nanotechnology (IEEE-NANO)*, Montreal, QC, Canada: IEEE, 2020, pp. 134–139.

[5] M. Walter et al., "fiction: An open source framework for the design of field-coupled nanocomputing circuits," 2019. arXiv: 1905.02477 [cs.ET].

[6] M. Walter et al., "Hexagons are the Bestagons: Design automation for silicon dangling bond logic," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22, San Francisco, CA: Association for Computing Machinery, 2022, p. 6.

[7] S. Hofmann, M. Walter, and R. Wille, "Scalable Physical Design for Silicon Dangling Bond Logic: How a 45° Turn Prevents the Reinvention of the Wheel," in *2023 IEEE 23rd International Conference on Nanotechnology (NANO)*, Jeju City, Korea, Republic of: IEEE, 2023, pp. 872–877.

[8] M. D. Vieira et al., "Three-Input NPN Class Gate Library for Atomic Silicon Quantum Dots," *IEEE Design & Test*, pp. 1–1, 2022.

[9] H. N. Chiu, "Simulation and analysis of clocking and control for field-coupled quantum-dot nanostructures," M.S. thesis, University of British Columbia, 2020.

[10] S. S. H. Ng et al., "A Blueprint for Machine Learning Accelerators Using Silicon Dangling Bonds," in *2023 IEEE 23rd International Conference on Nanotechnology (NANO)*, Jeju City, Korea, Republic of: IEEE, 2023, pp. 1–6.

[11] S. S. H. Ng et al., "Building a Machine Learning Accelerator with Silicon Dangling Bonds: From Verilog to Quantum Dot Layout," in *2025 IEEE 25th International Conference on Nanotechnology (NANO)*, 2025, pp. 483–488.

[12] W. Snyder, "Verilator and SystemPerl," in *North American SystemC Users' Group, Design Automation Conference*, vol. 79, 2004, pp. 122–148.

[13] S. Ma et al., "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits," 2024. arXiv: 2402.17764 [cs].

[14] S. Ma et al., "BitNet b1.58 2B4T Technical Report," 2025. arXiv: 2504.12285 [cs.CL].

[15] H. Wang, S. Ma, and F. Wei, "BitNet a4.8: 4-bit Activations for 1-bit LLMs," 2024. arXiv: 2411.04965 .

[16] C. Wolf and J. Glaser, "Yosys-a free verilog synthesis suite," 2013.

[17] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proceedings of the 22nd International Conference on Computer Aided Verification*, ser. CAV'10, Edinburgh, UK: Springer-Verlag, 2010, 24–40.

[18] M. Kepenekian et al., "Electron transport through dangling-bond silicon wires on H-passivated Si(100)," *Journal of Physics: Condensed Matter*, vol. 25, no. 2, p. 025 503, 2013.

[19] M. Walter et al., "Scalable design for field-coupled nanocomputing circuits," in *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, ser. ASPDAC '19, New York, NY, USA: Association for Computing Machinery, 2019, pp. 197–202.

[20] S. Hofmann, M. Walter, and R. Wille, "Graph-Oriented Layout Design for Field-Coupled Nanocomputing via Parallel Multi-Objective Search Space Exploration," *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14, 2025.

[21] S. Hofmann, M. Walter, and R. Wille, "Efficient and Scalable Post-Layout Optimization for Field-Coupled Nanotechnologies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 10, pp. 3790–3803, 2025.

[22] N. P. Jouppi et al., "TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings," 2023. arXiv: 2304.01433 [cs.AR].

[23] F. Riente et al., "ToPoliNano: A CAD Tool for Nano Magnetic Logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 7, pp. 1061–1074, 2017.

[24] U. Garlando, F. Riente, and M. Graziano, "FUNCODE: Effective Device-to-System Analysis of Field-Coupled Nanocomputing Circuit Designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 3, pp. 467–478, 2021.

[25] H. Genc et al., "Gemmini: Enabling Systematic Deep-Learning Architecture Evaluation via Full-Stack Integration," in *Proceedings of the 58th Annual ACM/IEEE Design Automation Conference*, IEEE Press, 2022, pp. 769–774.

[26] 1bitLLM. "BitNet b1.58-large model repository on Hugging Face. " [Online]. Available: `https://huggingface.co/1bitLLM/bitnet_b1_58-large`

[27] S. S. H. Ng et al., "SiDB BitNet b1.58 RTL source code," `https://github.com/samuelshng/sidb-bitnet-verilog`, 2026.