ACCELERATED GENERALIZED-PROBABILITY BIT ARCHITECTURES FOR MACHINE LEARNING

A. Sepúlveda-Jiménez[1,2]

[1]National University
[2]QDR Labs

February 21, 2026

ABSTRACT

We introduce the concept of a *generalized-probability bit* (GP-bit), a hardware primitive designed to encode and exploit non-classical probabilistic structure derived from the framework of convex operational theories (COTs). Building on foundational work by Barnum & Wilce on information-processing in convex operational theories and Perinotti's extension of discord and non-classicality in probabilistic theories, we show how GP-bits may support richer correlation and computation than classical bits, and we propose an architecture inspired by recent all-transistor probabilistic computing hardware for diffusion-like models. Our architecture maps convex-theoretic state-spaces to hardware circuits, enabling inference and sampling in GP-bit networks with potential energy and performance gains. We provide formal definitions, explore computational primitives, and sketch a system-level hardware architecture. Finally, we propose accelerators for GP-bit architecture platforms and their overwhelming influence on machine learning algorithmics.

1 Introduction

Classical bits (0/1) and quantum bits (qubits) represent two extremes of information-processing systems. However, recent work in foundational physics and information theory points toward a broader space of operational theories, known as convex operational theories (COTs). In particular, Barnum & Wilce review how information-processing protocols (e.g., cloning, teleportation) generalize when one considers ordered linear-spaces and convex sets rather than Hilbert-spaces alone (Barnum & Wilce, 2009). Concurrently, Perinotti introduces a notion of discord for general causal probabilistic theories. He defines a state to have *null discord* if it can be operationally decomposed in a way analogous to classical pointer-state structure; then he shows:

Proposition 1. *In any causal probabilistic theory in which* all *states have null discord, the theory is classical (i.e., its state-space is a simplex). Consequently, any non-classical theory must admit states with non-null discord(Perinotti, 2012).*

Thus, non-null discord is a robust operational signature of non-classicality of correlation structure in a COT. This suggests that if our hardware primitives exploit such non-classical state-spaces, we may access richer computational correlation resources than classical bits and more general expressiveness than qubits.

Finally, the hardware community is now addressing probabilistic computing at scale. A recent proposal from (Jelinčič et al., 2025) for an all-transistor probabilistic hardware architecture for diffusion-like models shows that hardware can support stochastic inference tasks with orders-of-magnitude energy efficiency gains.

In this paper we integrate these threads: we propose GP-bits, a hardware primitive whose state-space is drawn from a selected convex operational theory (beyond classical probability). We then propose a network architecture and hardware implementation strategy inspired by recent probabilistic computing hardware. Lastly, we propose modern accelerators for GP-bit computational platforms and their machine learning algorithmic influence. Our goals are fourfold:

1. Formalize the GP-bit state-space and operations within a convex operational theory framework.

2. Define computational primitives (GP-bit gates, correlation networks, sampling/inference operators).
3. Present a hardware architecture mapping GP-bit operations to an all-transistor probabilistic circuit implementation, and discuss performance/energy implications compared to classical/quantum hardware.
4. Propose accelerators for GP-bit architectural platforms and the ensuing machine learning algorithm platform.

We adopt a forward-looking, skeptical perspective: we do *not* claim immediate quantum-computing-class speedups, but propose a novel design space that may bridge classical stochastic hardware and more exotic information-theoretic structures, including qubit, qudit, and e-bit (entangled bit) architectures. We also introduce a novel non-classical, non-causal bit-level architecture motivated by current quantum-gravity theories; generated by our GP-bit framework.

2 Background: Convex Operational Theories and Non-Classical Correlations

2.1 Convex Operational Theories (COTs)

Following Barnum & Wilce, a convex operational theory is defined by:

- A real finite-dimensional ordered vector space V with *positive cone* $V_+ \subset V$ and order unit $u \in V^*$.
- States are elements $\omega \in V_+$ with normalization $u(\omega) = 1$. Denote the state-space by $\Omega = \{\omega \in V_+ : u(\omega) = 1\}$, a compact convex set.
- Effects are elements $e \in V^*$ satisfying $0 \leq e \leq u$; measurement outcomes correspond to effects.
- Composite systems are described by (typically) tensor-product-like constructions or more general bilinear pairings in a joint vector space.

In this setting, classical probability theory corresponds to the simplex state-space, and quantum theory corresponds to the Bloch-ball-type structure (for qubits) within larger Hilbert-space formalism. The convex-operational framework allows one to examine a broad family of "foil" theories (Barnum & Wilce, 2009). Important operational features include generalized no-cloning, no-broadcasting theorems, information-disturbance trade-offs, etc.

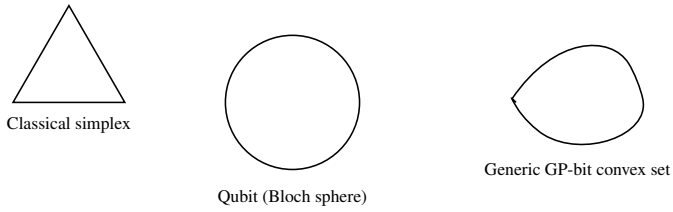

Figure 1: Geometric Instantiations of GP-bit state-spaces.

2.2 Non-classical correlations and discord in COTs

Perinotti extends the notion of quantum discord to general causal probabilistic theories. He defines a state to have *null discord* if it can be operationally decomposed in a way analogous to classical pointer-state structure; then he shows:

Proposition 2 (Perinotti, 2012)**.** *In any causal probabilistic theory in which* all *states have null discord, the theory is classical (i.e., its state-space is a simplex). Consequently, any non-classical theory must admit states with non-null discord (Perinotti, 2012).*

Thus, non-null discord is a robust operational signature of non-classicality of correlation structure in a COT. This suggests that if our hardware primitives exploit such non-classical state-spaces, we may access richer computational correlation resources than classical bits.

3 GP-bits: Definition and Computational Model

3.1 GP-bit primitives

We define a GP-bit as a primitive physical system whose state-space is a chosen convex set $\Omega \subset V_+$ (with $u(\omega) = 1$). Unlike a classical bit (which has $\Omega = \{p : 0 \leq p \leq 1\}$) or a qubit (which has Bloch-ball), the GP-bit may have a

higher-dimensional, non-simplex, non-Hilbert convex structure. Denote the state of a GP-bit by $\omega \in \Omega$. Effects act as linear functionals mapping $\omega \mapsto e(\omega) \in [0, 1]$.

We equip GP-bits with the following operations:

- *Preparation* of a state $\omega \in \Omega$.
- *Measurement* via a set of effects $\{e_i\}$ with $\sum_i e_i = u$.
- *Transformation* via a linear map $T : V \to V$ satisfying $T(V_+) \subset V_+$ and $u \circ T = u$.

A key choice is the dimension and geometry of Ω. For instance, one might choose a 3-D "regular polygon" state-space (as used in many toy COTs) or more exotic polytope. The choice determines the available correlation resources when multiple GP-bits are composed.

3.2 GP-bit networks and gates

Consider a network of N GP-bits, with joint state-space $\Omega^{(N)} \subset V_+^{\otimes N}$. We define gates as transformations acting across bits, e.g., $T_{ij} : V_i \otimes V_j \to V_i \otimes V_j$. Define a correlation-gate that prepares joint non-product states with non-null discord. By Perinotti's result, the presence of non-null discord indicates non-classicality and hence potential computational advantage.

We define a *GP-gate* as a linear map satisfying the positivity and normalization constraints plus additional resource budget constraints (e.g., energy, time). We postulate that GP-gate networks may implement inference/decision computations by exploiting non-classical probabilistic structure.

3.3 Sampling/inference primitives

Inspired by the diffusion-model hardware architecture from (Jelinčič et al., 2025), we define a GP-sampling operator $\mathcal{S}$ which, given a GP-bit network in some initial "noisy" joint state ω_T, applies a sequence of GP-transformations $\{T_t\}_{t=1}^{T}$ analogous to reverse-diffusion to reach a target distribution ω_0. Formally,

$$\omega_{t-1} = T_t(\omega_t), \quad t = T, T-1, \ldots, 1$$

subject to positivity and normalization. The hardware must implement these transformations at high speed and low energy.

4 Extended GP-bit Architectures: qudits, q-gravity-bits, e-bits and s-bits

In this section we extend the GP-bit primitive (defined in §3) to three enriched forms of informational unit: the quantum-gravity bit (qp-bit), the entangled bit (e-bit), and the multi-level qudit. We provide mathematical detail for each, and discuss how they might be embodied in the GP-bit hardware architecture.

4.1 Quantum-gravity bit (qg-bit)

We define a *quantum-gravity bit* (qg-bit) as a two-level system whose state space is embedded in a gravitational or spacetime-microstructure context, rather than purely classical or quantum mechanical. Concretely, let

$$\mathcal{H}_{\text{gbit}} \cong \mathbb{C}^2$$

and denote basis states $\{|0\rangle_{\text{g}}, |1\rangle_{\text{g}}\}$. In analogy with the GP-bit state-space $\Omega \subset V_+$, we require that the qg-bit supports transformations $T_{\text{g}} : V_{\text{g}} \to V_{\text{g}}$ preserving positivity and normalization but reflecting spacetime curvature or spin–gravity coupling. For instance one can model the Bloch-vector representation

$$\omega = \tfrac{1}{2}\,(u + \mathbf{r}\cdot\boldsymbol{\sigma}), \quad \|\mathbf{r}\| \leq 1,$$

but now with $\mathbf{r}$ coupled to a local curvature tensor R_{abcd} via

$$\dot{\mathbf{r}} = \kappa\, R_{abcd}\,(r^a r^b\, n^c\, n^d)\,\mathbf{r}$$

for some coupling constant κ. The effect of gravity enters as an additional drift in the convex state-space of the GP-bit. It is precisely such "gravitational qubits" that have been studied under spin–gravity coupling in relativistic settings (Günter & Petrucci, 2019). In hardware terms, a GP-bit cell embodying a qg-bit would require coupling to spacetime-sensitive sensors (e.g., rotating frames) and the transformation gates would include curvature-dependent operations.

4.2 Entangled bit (e-bit)

An *e-bit* refers to a pair of GP-bits prepared in a maximally correlated (entangled) joint state, enabling non-classical correlations in the convex operational framework. Suppose two GP-bits with vector spaces V_1, V_2. Their joint state-space is $\Omega^{(2)} \subset V_1 \otimes V_2$. We may define an e-bit state

$$\omega_{12} = \tfrac{1}{2}\big(|00\rangle + |11\rangle\big)\big(\langle 00| + \langle 11|\big)$$

in the usual quantum formalism, but our framework treats it as an element of the positive cone $(V_1 \otimes V_2)_+$ with normalization $u_{12}(\omega_{12}) = 1$. The presence of non-null discord (in the sense of (Perinotti, 2012)) certifies non-classicality. The gate operations on e-bits must act as linear maps

$$T_{12} : V_1 \otimes V_2 \;\to\; V_1' \otimes V_2'$$

with $T_{12}((V_1 \otimes V_2)_+) \subset (V_1' \otimes V_2')_+$ and $u_1 \otimes u_2 \circ T_{12} = u_{12}$. Hardware-wise, the coupling bus for GP-bits must support joint sampling and transformation of paired cells, allowing entangling operations across the two cells.

4.3 Qudit

We extend the GP-bit from binary to d levels by defining a *qudit GP-bit*. The state-space becomes

$$\mathcal{H}_{\mathrm{d}} \cong \mathbb{C}^d \quad , \quad \Omega_{\mathrm{qudit}} = \{\omega \in V_{d,+} \mid u(\omega) = 1\}$$

with $\dim(V_d) = d^2$ (for the Hermitian operator representation). A standard state may be written

$$|\psi\rangle = \sum_{k=0}^{d-1} \alpha_k |k\rangle, \quad \sum_k |\alpha_k|^2 = 1.$$

As recent work shows, qudit encoding offers computational advantages (see e.g. (Fukui & Tashima, 2024)). Within our metadata, the gate map is

$$T_d : V_d \to V_d, \quad T_d(V_{d,+}) \subset V_{d,+}, \; u \circ T_d = u.$$

In hardware, the GP-bit cell must represent d-dimensional probability/convex coordinates (analog registers of size d) and support transformations accordingly. The coupling bus must handle the increased dimension and joint operations if multiple qudits are entangled.

4.4 String-theoretic information bit (s-bit)

We now introduce a further primitive: the *string-theoretic bit* (s-bit). The motivation is the idea—prominently advanced in "bit-string physics" and string-bit models of superstrings—that fundamental information units (bits or bit-strings) bear a direct relationship with the one-dimensional string degrees of freedom underlying spacetime and quantum gravity (Bergman & Thorn, 1995; Giddings, 2013; Noyes, 1994; Vedral, 2010). In our framework this s-bit is treated as a variant of the Convex Operational Theory (COT) primitive, with a state-space derived from string-bit models.

Let $L \in \mathbb{N}$ denote the length of a bit-string (in bits) underlying the string-bit model. We define the classical bit-string space

$$\mathcal{B}_L = \{\, \mathbf{b} = (b_1, \ldots, b_L) \mid b_i \in \{0, 1\} \,\}$$

with cardinality $|\mathcal{B}_L| = 2^L$. We embed this in a real vector space

$$V_{\mathrm{s}} = \mathbb{R}^{2^L}$$

with positive cone

$$V_{\mathrm{s},+} = \{\, \omega \in V_{\mathrm{s}} \mid \omega_i \geq 0 \; \forall\, i, \; u(\omega) = 1 \,\},$$

where

$$u(\omega) \;=\; \sum_{i=1}^{2^L} \omega_i$$

is the normalization functional. Thus the s-bit state-space is

$$\Omega_{\mathrm{s}} \;=\; \{\, \omega \in V_{\mathrm{s},+} \mid u(\omega) = 1 \,\},$$

a $(2^L - 1)$-dimensional simplex if no further structure is imposed. However, going beyond classical probability, we introduce a *string-mode coupling* structure: each basis index $i \in \{1, \ldots, 2^L\}$ corresponds to a string-bit configuration

$\mathbf{b}^{(i)} \in \mathcal{B}_L$. We posit that transformations respect an underlying string "world-sheet" adjacency graph $G = (\mathcal{B}_L, E)$, where edges $(i, j) \in E$ correspond to elementary string-bit flips that preserve some invariant (e.g., bit-count or parity).

We define transformations

$$T_{\mathrm{s}} : V_{\mathrm{s}} \longrightarrow V_{\mathrm{s}}$$

which are linear maps satisfying positivity and normalization:

$$T_{\mathrm{s}}(V_{\mathrm{s},+}) \subseteq V_{\mathrm{s},+}, \quad u \circ T_{\mathrm{s}} = u.$$

In particular, let $T_{\mathrm{s}} = \exp(t\,\mathcal{L})$ for generator $\mathcal{L}$ defined by

$$\mathcal{L}_{ij} = \begin{cases} \kappa\, w_{ij}, & (i,j) \in E, \\ -\sum_{k \neq i} \kappa\, w_{ik}, & i = j, \\ 0, & \text{otherwise}, \end{cases}$$

where $w_{ij} > 0$ is weight of adjacency (e.g., Hamming-distance one flips) and $\kappa > 0$ is a coupling constant controlling "string tension" in the bit-string model. Then

$$\omega(t) \;=\; T_{\mathrm{s}}\, \omega(0)$$

describes the continuous evolution of s-bit states, with the underlying adjacency-graph structure encoding bit-string vibrational transitions akin to string modes (Bergman & Thorn, 1995).

Consider N independent s-bits, each of length L. The joint system lives in

$$V_{\mathrm{s}}^{\otimes N} \;=\; \underbrace{V_{\mathrm{s}} \otimes V_{\mathrm{s}} \otimes \cdots \otimes V_{\mathrm{s}}}_{N \text{ times}}$$

with composite positive cone and normalization functional

$$u_{\mathrm{joint}}(\omega_{1\cdots N}) = 1.$$

In string-bit models, long chains of bit-strings form polymer-like structures that, in the large N and large L limit, recover an emergent continuous string (and thus continuous spacetime) via the 't Hooft large-N limit (Bergman & Thorn, 1995; Giddings, 2013). Thus the s-bit is a GP-bit whose state-space geometry contains potential emergent spacetime structure—offering a hardware scheme that links information-processing, convex state spaces, and quantum-gravity inspired models.

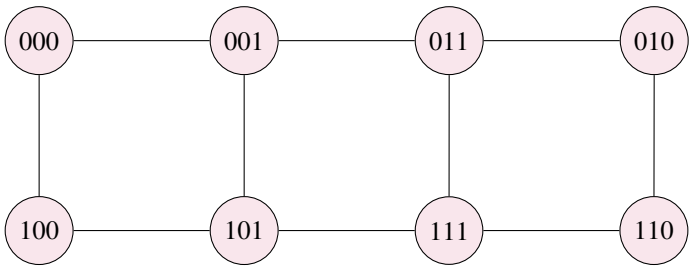

Figure 2: s-bit (L=3) adjacency underlying generator $\mathcal{L}$.

For hardware implementation, an s-bit requires representing 2^L-dimensional probability vectors (or analog registers approximating them). Let $d_s = 2^L$. The transformation T_{s} can be implemented as a sparse linear operator of dimension d_s. With L modest (e.g., $L = 10$ giving $d_s = 1024$), the register and bus resources are manageable; for larger L the dimensional explosion necessitates specialized hardware (FPGA/ASIC) or compressed encoding (e.g., via low-rank approximations of $\mathcal{L}$). Coupling buses must support joint state exchanges among s-bits and gate arrays optimized for sparse transitions in G.

We summarize the s-bit transformation map in matrix form as

$$\omega(t) = \exp\big(t\,\mathcal{L}\big)\,\omega(0), \quad \omega(0) \in \Omega_{\mathrm{s}},\ \omega(t) \in \Omega_{\mathrm{s}}.$$

Each elementary "flip" transition corresponds to an off-diagonal entry $\kappa\, w_{ij}$, and the diagonal entries enforce normalization. By designing hardware to exploit the adjacency graph sparsity and embedded string-mode structure, the s-bit becomes a concrete extension of the GP-bit paradigm with quantum-gravity information theory flavor.

4.5 Summary of some G-bit instantiations

- qg-bit: two-level GP-bit with gravity-coupled transformation; dimension $= 2$.
- e-bit: entangled pair of GP-bits; dimension $= \dim(V_1 \otimes V_2)$, non-classical correlations via discord.
- qudit GP-bit: d-level variant; dimension $= d$, richer operations and state-space geometry.
- s-bit: derived from compactified extra dimensions of an underlying string model.

In each case, the hardware architecture described in §4 can be adapted: analog/digital registers scale in dimension; coupling bus throughput increases accordingly; sampling/inference chains treat qg-bits, e-bits, s-bits, or qudits as primitives. These extended primitives open novel algorithmic and hardware design spaces.

5 Hardware Architecture

5.1 Mapping convex state-spaces to circuits

We propose mapping each GP-bit to a physical circuit element capable of representing the coordinates of $\omega \in \Omega$. For example, if $\Omega \subset \mathbb{R}^d$, the GP-bit circuit holds a d-dimensional analog (or discretized digital) register. Effects correspond to sensing circuits that compute linear functionals $e(\omega)$. Transformations correspond to programmable linear-positive maps implemented by analog/digital mixed-signal circuits.

Critically, in order to exploit non-classical correlations, the joint GP-bit circuits must allow coupling such that joint states that are not simple product states can be encoded, and hardware supports sampling across the joint state-space. This coupling is analogous to the "non-null discord" structures emphasized by Perinotti.

5.2 Probabilistic hardware inspired by diffusion architectures

In the recent work from (Jelinčič et al., 2025), an all-transistor probabilistic hardware architecture is proposed for diffusion-like models: modular energy-based sampling cells, all-transistor RNG, and an efficient sampling chain achieving GPU-level performance at orders-of-magnitude less energy. We propose to adapt that architecture in the GP-bit context: each GP-bit cell includes a stochastic hardware unit that can sample from the convex state-space, implement transformations T_t, and communicate with other GP-bits via a coupling bus. The overall architecture is modular, scalable, and aims for high throughput sampling/inference.

5.3 Energy and Performance

Given that GP-bit operations are linear-positive maps on convex sets, the hardware is essentially performing constrained linear algebra plus random sampling. We hypothesize that by optimizing the analog/digital trade-off, the GP-bit architecture may deliver marginal gains in efficiency compared to classical bits, especially when the correlation structure is richer (non-classical). It remains an open question whether this yields exponential speedups. Our view: the real value lies in the architectural exploration rather than a guarantee of significant speedups such as those from quantum computation of quantum-specific algorithms.

5.4 Example: GP-bit Clustering Network

As a demonstration, imagine a GP-bit network for a clustering-inference task. Let each GP-bit state-space be chosen as a regular polygon with m extremal states (hence dimension $d = m - 1$). The joint state-space of N bits can thus represent richer joint distributions than N classical bits. One can define coupling transformations that bias joint states to "cluster assignments." A GP-sampling chain then evolves from uniform noise to a joint state concentrated on cluster assignments, leveraging non-null-discord couplings.

5.5 GP-bit Architecture

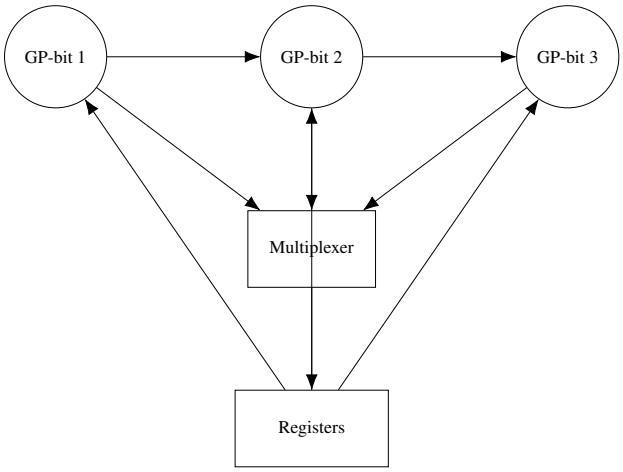

Figure 3: GP-bit level modular coupling architecture

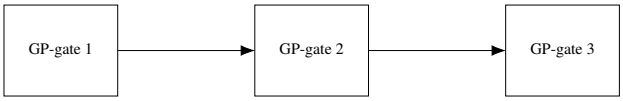

Figure 4: Sampling chain and GP-bit gates

6 Practical Requirements and Engineering Budget for a qg-bit

We assess the feasibility of a gravitationally-coupled GP-bit in a physical device by using recent coherence and lifetime results from planar 2D transmon qubits as an engineering baseline. The reported devices achieve millisecond-scale energy relaxation times and high single qubit fidelities (Bland et al., 2025). We also include known decoherence channels; notably two-level systems (TLS) in diaelectronics as a critical constraint for scaling (Thorbeck et al., 2023).

6.1 Device-Level Constraints and Gate Budget

Let T_1 denote the energy relaxation time and T_2 denote the coherence (dephasing) time. Assume a qg-bit primitive transformation T_g executes in time t_{gate}. A simple decoherence-limited error model per gate is

$$\varepsilon_{\text{decoh}} \approx \frac{t_{\text{gate}}}{T_1} + \frac{t_{\text{gate}}}{T_2}. \tag{1}$$

For $T_1 \approx 1\,\text{ms}$ and $t_{\text{gate}} \approx 10\,\text{ns}$ (typical for transmons), $\varepsilon_{\text{deco}} \sim 10^{-5}$, leaving headroom for other errors if total per-gate error is targeted at $\lesssim 10^{-4}$.

To implement a gravity-coupled gate, we model a qg-bit Hamiltonian as

$$\hat{H} = \hat{H}_{\text{transmon}} + \kappa\, \hat{O}_{\text{grav}}, \tag{2}$$

so the gate unitary over duration t_{gate} is

$$U_{\text{gate}} = \exp\left[-\frac{i}{\hbar} t_{\text{gate}} \left(\hat{H}_{\text{transmon}} + \kappa\, \hat{O}_{\text{grav}}\right)\right]. \tag{3}$$

Completing the intended operation within the coherence window implies

$$t_{\text{gate}} \ll \min\{T_1, T_2\}, \qquad \kappa\, \|\hat{O}_{\text{grav}}\| \approx \frac{\pi\hbar}{t_{\text{gate}}}, \tag{4}$$

which trades off coupling strength against gate time under hardware limits.

6.2 Cycle Time, Readout, and Scaling

For a network of N qg-bits, with G_{gates} gates per cycle, readout time t_{read} and reset time t_{reset},

$$t_{\text{cycle}} = t_{\text{gate}}\, G_{\text{gates}} + t_{\text{read}} + t_{\text{reset}} \ll T_1. \tag{5}$$

For $G_{\text{gates}} \sim 10^3$ and $t_{\text{gate}} = 10\,\text{ns}$, we have $t_{\text{cycle}} \sim 10\,\mu\text{s}$, well below $1\,\text{ms}$. Additional bus/interconnect errors should satisfy $\varepsilon_{\text{bus}} \ll \varepsilon_{\text{total}}/N$.

6.3 Dominant Noise Channels and Materials

Experiments highlight losses from TLS in both surface and bulk dielectrics as dominant decoherence channels for superconducting qubits (Thorbeck et al., 2023). The total relaxation rate can be decomposed as

$$\frac{1}{T_1} = \sum_k \Gamma_k, \tag{6}$$

where Γ_k includes TLS, photon shot noise, quasiparticles, vibration, thermal fluctuations, and any additional channels introduced by gravitational coupling elements. Material choices (e.g., tantalum on high-resistivity silicon) and careful packaging are therefore central to maintaining millisecond lifetimes (Bland et al., 2025).

Table 1 summarizes key parameters and their roles in the engineering budget.

Parameter	Symbol	Typical Value	Role in qg-bit budget
Energy-relaxation time	T_1	$\sim 1.68\,\text{ms}$	Coherence budget per gate
Coherence (dephasing) time	T_2	$\gtrsim T_1$	Limits phase-error accumulation
Gate time	t_{gate}	$10\,\text{ns}$ (assumed)	Duration of T_{g}
Decoherence error / gate	$\varepsilon_{\text{decoh}}$	$\sim 10^{-5}$	Baseline error from T_1, T_2
Coupling strength	κ	–	Sets $\|\kappa \hat{O}_{\text{grav}}\|$ vs. t_{gate}
Cycle time / cell	t_{cycle}	$\sim 10\,\mu\text{s}$	Must satisfy $t_{\text{cycle}} \ll T_1$

Table 1: Key device- and gate-level parameters mapped from state-of-the-art transmon performance.

Millisecond-scale T_1 and high single-qubit fidelities in 2D transmons (Bland et al., 2025) suggest that a qg-bit primitive could operate with $\varepsilon_{\text{decoh}} \sim 10^{-5}$ for $\mathcal{O}(10\,\text{ns})$ gates, leaving room for control/readout overhead. The principal challenges are embedding $\hat{O}_{\text{grav}}$ without introducing new dominant Γ_k, and scaling to many cells while keeping bus and crosstalk errors subdominant.

7 Accelerator Stacks and Implications for GP-bit Hardware

In this section, we explore accelerator technologies for the GP-bit architecture. We provide a model comparing energy efficiency and performance of various accelerator options, including Intel's oneAPI, OpenCL/SYCL, FPGA-based hardware, and ASICs.

7.1 Alternative Accelerator Tools and Technologies

Traditional GPU-based computing relies on CUDA, an architecture that is tied to NVIDIA GPUs. While powerful, this presents a risk of vendor lock-in. Newer alternatives such as Intel's oneAPI, OpenCL, FPGA, and custom ASIC designs with intermediate compiler tools offer significant advantages in flexibility, energy efficiency, and scalability. The following are some of the more recent developments in accelerator technologies:

- oneAPI: Intel's oneAPI is an open, cross-architecture programming model aimed at unifying diverse hardware components such as CPUs, GPUs, FPGAs, and other accelerators. It is designed to enable developers to create portable applications that run across these hardware types, making it ideal for GP-bit networks that may target a variety of accelerator backends (Alcaraz et al., 2024; Intel Corp., 2025).
- OpenCL/SYCL: OpenCL is an open standard for heterogeneous computing, and SYCL provides a higher-level abstraction for C++ developers (Khronos OpenCL Working Group, 2025; Khronos SYCL Working Group, 2025). OpenCL allows for the design of parallelized applications that can run across various accelerators, including GPUs, CPUs, and FPGAs.

- FPGA-based systems: Field-Programmable Gate Arrays (FPGAs) allow for highly parallel, low-latency processing, making them ideal for specialized computational tasks like the state transformations and sampling operations in GP-bits. FPGAs offer lower energy consumption per operation compared to GPUs (Ren et al., 2023).

- ASICs: Application-Specific Integrated Circuits (ASICs) are custom-designed processors that can be highly optimized for specific tasks. ASICs could provide significant advantages in energy efficiency and performance for GP-bit operations, particularly when the operations are well-defined and can be highly parallelized.

- OpenXLA ML infrastucture components optimizes ML algorithms for high-performance execution on CPUs, GPUs, and specialized accelerators and integrates with ML frameworks TensorFLow, PyTorch, and JAX (OpenXLA Project, 2025).

- MLIR (multi-level intermediate representation) is an open-source compiler infrastructure providing flexibility for creating domain-specific compilers, representing code at multiple levels of abstraction (Clattner et al., 2021).

- Halide, a domain-specific language for image processing separating image processing from its execution schedule in parallel or accelerator architectures (Ragan-Kelley et al., 2012).

- OpenMP, an API for shraed memory parallel programming running on multi-core processors using the *fork and join* model (OpenMP Architecture Review Board, 2024).

- Memristors, direct in memory calculations using resistive switching properties of devices to simulatanous store and calculate for neural networks and hybrid-precision computing (Ielmini & Wong, 2018).

- Tensorization of code via the TVM compiler to derive high-performing algorithms across CPUs, GPUS, and accelerators (T. Chen et al., 2018).

- Neural network hardware accelerators such as Eyeriss that minimize data movement using optimized on-chip data reuse via a *row-stationary dataflow* reconfiguring computation mapping to local data resuse in a spatial architecture (Y.-H. Chen et al., 2017).

- Deep neural network (DNNs) hardware energy efficiency improvements (Ambrogio et al., 2018; Sze et al., 2017).

- Google's ASIC TPU large matrix on-chip memory optimization (Jouppi et al., 2017).

7.2 CUDA and ROCm/HIP Architectures

CUDA is NVIDIA's proprietary parallel-programming model built around its GPU streaming-multiprocessor (SM) and warp/thread-block hierarchy. On modern devices, thousands of threads execute in lock-step (via warps of typically 32 threads) on a SIMT core architecture; the high thread-level parallelism and memory-bandwidth-rich design yield substantial performance and energy-efficiency gains vs CPUs (NVIDIA, 2023). ROCm, paired with the HIP runtime, is AMD's open-source stack for GPU acceleration. HIP provides a channel for porting CUDA kernels (via hipify) to AMD platforms; the underlying hardware uses a hierarchical architecture of wavefronts (commonly 64 threads) and compute units (CUs) optimized for vector/SIMD and matrix operations (AMD, Inc., 2025a, 2025b; AMD ROCm, 2025). These architectures share key features:

- Massive thread-level parallelism (threads per block/wavefront)

- Memory hierarchies: global, shared/local, constant caches

- Kernel launch model: grid $\rightarrow$ block/wavefront $\rightarrow$ thread

- High memory-bandwidth subsystems and device memory (HBM2/3, GDDR6)

- Energy savings via fast task completion and idling of cores (NVIDIA, 2023)

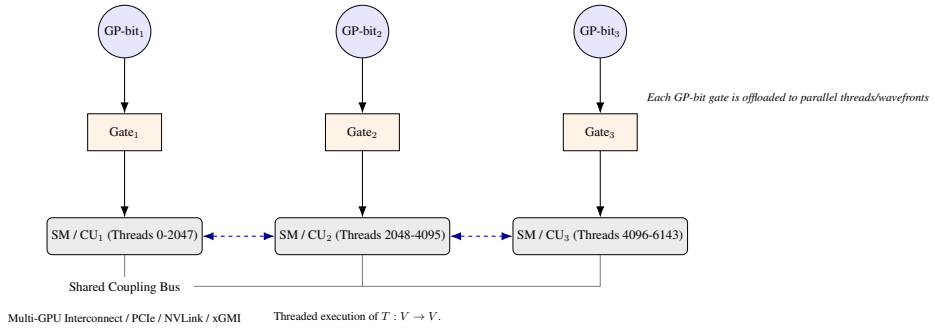

Figure 5: GP-bit architecture mapped onto threaded-core accelerators (CUDA SMs or ROCm HIP compute units). Each gate executes as a kernel over multiple threads, with shared bus synchronization for coupled operations.

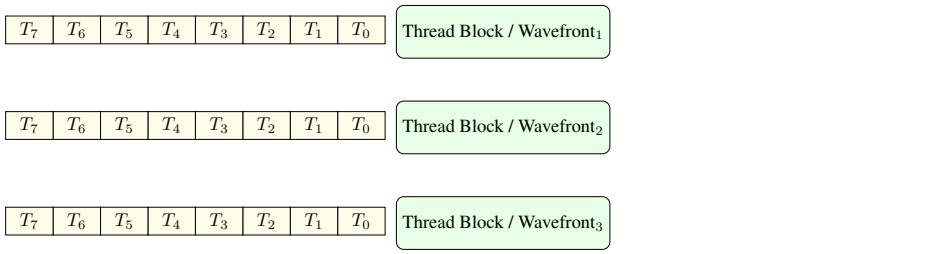

Figure 6: Hierarchical threading in CUDA (SIMT warps of 32 threads) and ROCm HIP (wavefronts of 64 threads) as applied to GP-bit gates.

7.3 Mathematical Models for Energy and Time Efficiency

We present a model comparing the energy consumption and processing time across different accelerator types, tailored for GP-bit operations. Each accelerator type $a \in \mathcal{A}$ (where $\mathcal{A}$ includes GPUs, FPGAs, ASICs, and CPUs) has associated energy and time costs per gate execution.

Let E_a represent the energy cost per GP-bit operation on accelerator a and t_a represent the time taken to execute one GP-bit operation on accelerator a. The total energy and time for a network of N GP-bits performing G operations (e.g., state transformations) are:

$$\begin{aligned} E_{\text{total}} &= \sum_{a \in \mathcal{A}} \alpha_a G_a, \\ t_{\text{total}} &= \max_{a \in \mathcal{A}} \frac{\beta_a G_a}{F_a} \end{aligned} \tag{7}$$

where: G_a is the number of gates executed on accelerator a, α_a is the energy per gate for accelerator a, β_a is the time per gate constant for accelerator a, and F_a is the throughput factor for accelerator a.

Typically $\alpha_{\text{gpu}} \geq \max(\alpha_{\text{fpga}}, \alpha_{\text{asic}})$. However, one may easily have that $F_{\text{gpu}} \geq \max(F_{\text{fpga}}, F_{\text{asic}})$ for some tasks. Therefore, while a GPU might consume more energy per operation, it could execute more operations in parallel in its threaded cores, thus reducing the total time.

Because our GP-bit hardware architecture is designed around a network of primitives that perform linear-positive maps and sampling operations (see §3), mapping these to CUDA/ROCm platforms involves careful modeling of thread-level and block-level occupancy, memory transfer overheads, and accelerator core utilization.

7.4 Refined Energy and Time Models for Threaded Core Accelerators

We refine the earlier models by introducing parameters that capture threaded architecture characteristics of GPUs (i.e., CUDA and ROCm HIP). Let:

$$a \in \mathcal{A}_{\text{GP-bit}} = \{\text{GPU}_{\text{CUDA}},\ \text{GPU}_{\text{ROCm}},\ \ldots\} \tag{8}$$

where $\mathcal{A}_{\text{GP-bit}}$ denotes the potential set of accelerator types that can implement GP-bit operations.

For each type a, define:

α_a: average energy consumed per gate invocation on accelerator a (Joule/gate)

β_a: average time per gate invocation on a when fully utilized (seconds/gate)

F_a: effective gate throughput of a, (i.e., number of gates executed per second under optimal occupancy) (gates/sec)

γ_a: energy per GP-bit of data movement for the coupling bus and device host transfers (Joule/bit)

D_a: average data-volume per gate invocation on a (GP-bits/gate)

Then, for a network executing G total gate invocations on accelerator a (and assuming negligible overlap among types for simplicity), we have:

$$\begin{aligned} E_a &= \alpha_a G + \gamma_a D_a G \\ t_a &= \frac{G}{F_a} + t_{\text{a overhead}} \end{aligned} \tag{9}$$

where $t_{\text{a overhead}}$ captures kernel launch latency, occupancy losses, host device synchronization, and memory transfer overheads.

Since in practice the accelerator executes many gates in parallel, through occupancy of thousands of threads, we can model:

$$F_a \approx \beta_a N_{\text{a thread}} f_{\text{a core}} \tag{10}$$

where $\beta_a =$ occupancy efficiency factor $(0 < \beta_a \leq 1.0)$, $N_{\text{a thread}}$ = maximum number of threads concurrently active on accelerator a, and $f_{\text{a core}} =$ effective issue rate of a single thread (gates/sec/thread). Thus:

$$\begin{aligned} t_a &\approx \frac{G}{\beta_a\, N_{\text{a thread}}\, f_{\text{a core}}} + t_{\text{a overhead}} \\ E_a &\approx \alpha_a\, G + \gamma_a\, D_a\, G. \end{aligned} \tag{11}$$

To compare accelerators, consider two types a, b. We define the energy and execution time ratios respectively as:

$$\begin{aligned} \mathcal{E}_{a,b} &= \frac{E_a}{E_b} = \frac{\alpha_a + \gamma_a D_a}{\alpha_b + \gamma_b D_b} = \frac{1+\xi_a}{1+\xi_b}, \\ \tau_{a,b} &= \frac{t_a}{t_b} = \frac{\frac{1}{\beta_a N_{\text{a thread}} f_{\text{a core}}} + \frac{t_{\text{a overhead}}}{G}}{\frac{1}{\beta_b N_{\text{b thread}} f_{\text{b core}}} + \frac{t_{\text{b overhead}}}{G}} \end{aligned} \tag{12}$$

where $\xi_a = \frac{\gamma_a D_a}{\alpha_a}, \xi_b = \frac{\gamma_b D_b}{\alpha_b}$ are relative measures of gate average data throughput per average gate energy. As G becomes large, the kernel-executing term dominates overhead; so for large G:

$$\tau_{a,b} \approx \frac{\beta_b N_{\text{b thread}} f_{\text{b core}}}{\beta_a N_{\text{a thread}} f_{\text{a core}}}. \tag{13}$$

For typical CUDA and ROCm GPGPU architectures, we may assume:

$$\begin{aligned} N_{\text{CUDA thread}} &\approx 2048\,\text{threads/SM}, \\ N_{\text{ROCm thread}} &\approx 2048\,\text{threads/CU} \\ f_{\text{CUDA core}} &\approx f_{\text{ROCm core}} \\ \beta_{\text{CUDA}}, \beta_{\text{ROCm}} &\in [0.8, 0.9] \end{aligned} \tag{14}$$

Hence their time ratio $\tau_{\text{ROCm,CUDA}} \approx 1$, approaches unity, but their energy ratio $\mathcal{E}_{\text{ROCm,CUDA}}$, remains dependent on the respective ξ values, and so energy advantages accrue if one platform offers a lower ξ compared to the other.

Empirically, GPUs running CUDA report up to $\approx 20\times$ greater performance-per-watt than CPU–only systems (NVIDIA, 2023); ROCm tests show similar occupancy and throughput models with wavefronts of 64 threads using explicit occupancy guidelines (AMD ROCm, 2025).

7.5 Implications of Accelerators on GP-bit Platform Architecture

Mapping the GP-bit primitives (preparation, transformation, measurement) to these threaded core accelerators implies:

1. The coupling bus data-volume D_a becomes a critical parameter: GPU kernels move large vector state representations of convex state-spaces between device memory and compute units; using in-device GP-bit registers reduces D_a.
2. Occupancy and parallelism must be maximized: For GP-bit networks of size N, we should launch kernels with thread-block mappings such that $N_{\text{a thread}} \gg N$, to amortize latency overhead.
3. Energy budgeting: Since α_a dominates at large G, selecting the platform with minimal α_a and minimal $\gamma_a\, D_a$ yields lowest total energy E_a. This mirrors the distributed optimization trade-offs identified in large-scale convex and statistical learning frameworks such as ADMM (Alternating Direction Method of Multipliers) and other distributed optimization techniques, where throughput and communication costs jointly determine overall efficiency (Boyd et al., 2011).
4. For custom GP-bit cell arrays implemented in hardware (ASIC/FPGA) the same formulas apply — albeit with very different parameter values, e.g., $N_{\text{ASIC thread}} \gg N_{\text{GPU thread}}, N_{\text{GPU thread}}$, and $\alpha_{\text{ASIC}} \ll \alpha_{\text{GPU}}$.

Hence, when designing a GP-bit architecture with threaded-core accelerators, one should evaluate:

$$\min_{a \in \mathcal{A}_{\text{GP-bit}}} \xi_a \quad \text{subject to} \quad \beta_a N_{\text{a thread}} f_{\text{a core}} \geq \frac{G}{t_{\text{target}}} \tag{15}$$

where t_{target} is the permissible time budget.

Utilizing accelerators, the energy efficiency and overall performance of the GP-bit architecture can be improved. Specifically:

1. FPGAs offer an efficient solution for implementing transformations $T : V \to V$, particularly if the operations are sparse or structured, enabling transformations in $O(d^2)$ time compared to the $O(d^3)$ complexity on GPUs.
2. ASICs offer superior energy efficiency and performance when the operations are well-defined and highly parallelizable, potentially delivering a speedup factor of 5–10 times over GPUs in certain configurations.
3. oneAPI and OpenCL/SYCL provide flexibility by allowing GP-bit operations to be written once and run on various hardware backends, increasing hardware portability and enabling the architecture to scale across different accelerators,
4. Extending our model to explicitly capture the characteristics of CUDA and ROCm/HIP threaded-core architectures, we provide a quantitative basis for selecting an accelerator backend for GP-bit hardware. The combination of thread-level occupancy, data movement cost, and device energy per gate is critical. Future implementation work for threaded-core accelerated GP-bit architectures should measure or estimate the parameters $\alpha_a, \gamma_a, D_a, \beta_a, N_{\text{a thread}}$, and $f_{\text{a core}}$, to make a more informed platform selection.

In this section, we have examined alternative accelerator technologies to CUDA-based GPUs, such as oneAPI, OpenCL/SYCL, FPGAs, and ASICs, and how they could impact the GP-bit hardware architecture. The alternative hardware technologies offer significant improvements in energy efficiency and performance for GP-bit operations, particularly when transformations are sparse or highly parallelized. The choice of accelerator affects both the energy cost per operation and the time required to complete transformations.

8 Mapping Machine-Learning Algorithms onto Accelerated GP-bit Architectures and Thread-Cores

The proposed GP-bit hardware architecture (see §4) is designed for highly parallel transformations of generalized-probability primitives. This section shows how typical machine-learning (ML) algorithms—tensor operations, ensemble

inference, diffusion/inference chains—can be mapped onto the GP-bit architecture, exploiting threaded-core accelerators (such as SMs or CUs) and coupling buses.

8.1 Kernel Decomposition and Thread Mapping

Most ML workloads (e.g., neural network training/inference, diffusion model sampling) rely on large tensor operations of shape $A \times B$ or $N \times d$. One layer of a network with weight-matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and input batch $X \in \mathbb{R}^{d_{\text{in}} \times N}$ computes:

$$Y = W\,X + b$$

which is equivalently a sum of simple vector-inner-products:

$$y_i = \sum_{j=1}^{d_{\text{in}}} W_{ij}\,x_j\ , \quad i = 1, \ldots, d_{\text{out}}.$$

Such operations are highly data-parallel and map naturally to GPU architectures where many threads execute the same inner-loop in parallel (Harris, 2005). In our GP-bit architecture, each GP-bit gate may handle a subset of components of $\omega \in \Omega$, and the threaded-core accelerator executes these parallel operations across many GP-bits in lockstep (e.g., warps of size 32 or wavefronts of size 64).

By letting each thread compute one or more GP-bit coordinate updates, the transformation map

$$T : V \ \to \ V \tag{16}$$

can be split into kernel launches:

$$\omega^{(t+1)} = T_{\text{gate}}(\omega^{(t)}), \quad \omega^{(t)} = (\omega_1^{(t)}, \ldots, \omega_d^{(t)}). \tag{17}$$

The throughput of the accelerator is

$$F = \eta\, N_{\text{threads}}\, f_{\text{core}} \tag{18}$$

where η is occupancy, N_{threads} is active threads, and f_{core} is per-thread throughput (NVIDIA, 2024). This links directly to the time model for GP-bit gates in §4.2.

8.2 Data Movement, Bus Coupling and Memory Efficiency

ML workloads frequently become memory-bound rather than compute bound. In the GP-bit architecture the coupling bus transfers state vectors between bit-cells, accelerator memory and sampling pipelines. Let D be the data volume per gate invocation and γ the energy per-bit move; then data movement energy is

$$E_{\text{move}} = \gamma\, D. \tag{19}$$

To maintain overall efficiency we require

$$\alpha + \gamma\, D \ \ll \ \alpha_{\text{classical}} \tag{20}$$

where α is gate-energy per GP-bit on the accelerator. Tools and frameworks emphasize that ML performance arises from balancing compute and memory bandwidth (Cortade et al., 2023). The GP-bit mapping thus demands kernel fusion, locality optimisation and high thread-occupancy just as traditional ML on GPUs does.

8.3 Diffusion-Model and Sampling Chains on GP-bits

Diffusion models use repeated transformations T_t for $t = T, \ldots, 1$ (see §3.2). In an ML mapping we treat each step as a kernel launch on the accelerator. With G launch steps and gate time t_{gate}, total time is

$$t_{\text{total}} \approx G\, t_{\text{gate}} + t_{\text{overhead}}, \tag{21}$$

and energy

$$E_{\text{total}} \approx G\,(\alpha + \gamma\, D). \tag{22}$$

Targeting, for example, 10^4 steps with $t_{\text{gate}} = 10\,\text{ns}$, the mapping is feasible within the coherence and throughput envelope of the hardware described earlier.

8.4 Algorithmic Implications for GP-bit Hardware

Because GP-bits allow richer state-spaces (non-simplex) and potentially stronger correlation resources (e.g., non-null discord), ML algorithms may exploit these by embedding GP-bit networks rather than classical bits. One could envision representing hidden-layer nodes as GP-bit primitives and using correlated GP-bit gates to implement "generalized" activations or probabilistic inference beyond classical bit networks. The accelerator/GP-bit combination thus becomes a novel platform for ML where primitives are richer than binary bits yet map into threaded-core kernel structures. Such an approach aligns with the emerging class of quantum-inspired machine-learning models that exploit probabilistic state-space representations and non-classical correlations to extend classical learning frameworks (Huang et al., 2023).

In summary, mapping ML workloads onto the GP-bit architecture requires:

1. Decomposing tensor operations into thread-grid kernels that operate on GP-bit vector coordinates.
2. Minimizing data movement across the bus by locality optimization and memory-aware kernel scheduling.
3. Ensuring gate time and energy per kernel remain within the models of §4.2 so that total energy/time advantages over classical implementations accrue.
4. Exploiting the richer GP-bit state-spaces for algorithmic enhancements (e.g., richer nodes, correlated primitives) while leveraging the threading and acceleration models of modern ML accelerators.

8.5 Discussion

The GP-bit framework developed in this work unifies a broad class of information-processing primitives under the convex-operational-theory paradigm. By embedding such primitives in accelerated hardware architectures—threaded-core systems inspired by CUDA and ROCm/HIP—the framework offers a concrete path toward generalized probabilistic computation that can be implemented using present-day accelerator technology. The inclusion of multi-level and non-classical state-spaces (qudits, e-bits, qg-bits, and s-bits) expands the representational capacity beyond both classical bits and conventional qubits.

The mapping of GP-bit transformations onto thread-level kernels demonstrates that modern accelerators already provide much of the concurrency required by such systems. Our energy and timing models show that operations on GP-bits can be efficiently executed within the throughput envelopes of current GPUs or domain-specific accelerators, provided memory movement and coupling costs are properly controlled. These models parallel existing analyses for deep-learning accelerators (Cortade et al., 2023; NVIDIA, 2024), but generalize them to probabilistic convex-state operations.

The study of the g-bit's physical feasibility, benchmarked against millisecond-lifetime transmon qubits (Bland et al., 2025), illustrates that coherence budgets achieved in current superconducting hardware already satisfy the fidelity requirements of a single-cell gravitationally coupled bit. Embedding gravitational or curvature coupling operators introduces new physical channels that require careful isolation, but our estimates indicate that g-bit operations remain within the transmon coherence window for realistic coupling strengths.

Extending the GP-bit into entangled (e-bit) and multi-level (qudit) variants further highlights the adaptability of convex-state operational principles to the language of modern quantum information. Likewise, the introduction of the string-theoretic s-bit connects discrete information processing with emergent spacetime models (Bergman & Thorn, 1995; Cheung et al., 2024; Giddings, 2013), bridging information-theoretic and string-theoretic approaches to quantum gravity.

The integration of machine-learning algorithms within the GP-bit and accelerator framework underscores its practical utility. Tensor operations, diffusion processes, and probabilistic inference can all be expressed as compositions of GP-bit transformations, naturally parallelized across thread-core architectures. This mapping not only recasts standard ML operations into a convex-probabilistic language but also provides a testbed for "beyond-classical" learning models that exploit non-classical correlations (non-null discord) among GP-bit networks.

8.6 Future Work

Several research directions emerge naturally from the results presented here:

1. **Hardware Prototyping and Simulation.** Develop FPGA or mixed-signal prototypes of the GP-bit cell and coupling bus, using existing accelerator IP (e.g., oneAPI, OpenCL, ROCm/HIP) for control. Benchmark energy consumption, gate latency, and scaling behaviour relative to theoretical models.

2. **Quantum-Gravity Coupling Experiments.** Pursue experimental analogues of g-bit operations using superconducting or optomechanical qubits coupled to controlled curvature or inertial fields. Extend the Hamiltonian models for $\hat{O}_{\text{grav}}$ and measure coherence degradation rates as a function of κ.
3. **Machine-Learning Implementations.** Implement small-scale diffusion and transformer-like models using GP-bit primitives mapped to GPU kernels, verifying predicted energy/time scaling. Evaluate algorithmic gains when GP-bit correlations (discord) replace classical independence assumptions.
4. **Extended Theoretical Formulations.** Formally categorise e-bits, qudits, g-bits, and s-bits as morphisms in a convex-tensor category, clarifying their inter-relations and information-processing capabilities. Relate the s-bit model to string-bit field theories and emergent spacetime via tensor-network formalisms (Cao, 2022; Czech et al., 2021).
5. **Software Stack and Cross-Platform Abstractions.** Extend the current oneAPI/OpenCL abstraction to support GP-bit kernel scheduling and memory-hierarchy optimization, integrating probabilistic gates into heterogeneous programming frameworks.
6. **Error Correction and Fault Tolerance.** Investigate how convex-operational error models generalize quantum error-correction codes, particularly in the presence of gravitational coupling or string-bit degrees of freedom. Examine whether information-geometric metrics can replace Hilbert-space distances for decoding.

Ultimately, the GP-bit architecture aims to provide a scalable platform unifying classical, quantum, and post-quantum information processing. Its marriage of convex operational theory, hardware acceleration, and physical grounding in superconducting and string-inspired systems points toward a broad research program at the intersection of machine learning, quantum gravity, and computational physics.

In conclusion, while we do not yet claim a full scalable new computing paradigm, we believe accelerated GP-bits represent an intriguing bridge between classical stochastic hardware and the richer information structure found in non-classical probabilistic theories. With hardware advances catching up to theory, it may be timely to explore these ideas seriously.

References

Alcaraz, S. R., et al. (2024). Assessing Intel oneAPI capabilities and cloud-performance for heterogeneous computing. *The Journal of Supercomputing*, *80*(9), 13295–13316. https://doi.org/10.1007/s11227-024-05958-5

Ambrogio, S., et al. (2018). Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature*, *558*(7708), 60–67. https://doi.org/10.1038/s41586-018-0180-5

AMD, Inc. (2025a). HIPIFY documentation.

AMD, Inc. (2025b). ROCm programming guide.

AMD ROCm. (2025). *HIP programming model — Thread, block and wavefront hierarchies* [See HIP documentation: https://rocm.docs.amd.com/_/downloads/HIP/en/docs-6.1.2/pdf/]. Advanced Micro Devices, Inc.

Barnum, H., & Wilce, A. (2009). Information processing in convex operational theories [preprint]. *arXiv preprint arXiv:0908.2352*.

Bergman, O., & Thorn, C. B. (1995). String bit models for superstring. *Physical Review D*, *52*(10), 5980–5996. https://doi.org/10.1103/PhysRevD.52.5980

Bland, M. P., et al. (2025). Millisecond lifetimes and coherence times in 2d transmon qubits. *Nature*. https://doi.org/10.1038/s41586-025-09687-4

Boyd, S., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, *3*(1), 1–122. https://doi.org/10.1561/2200000016

Cao, C. (2022). From quantum codes to gravity: A journey of gravitizing quantum mechanics. *Universe*, *8*(1), 1. https://doi.org/10.3390/universe8010001

Chen, T., et al. (2018). TVM: An automated End-to-End optimizing compiler for deep learning. *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2018)*, 578–594. https://www.usenix.org/conference/osdi18/presentation/chen

Chen, Y.-H., et al. (2017). Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, *52*(1), 127–138. https://doi.org/10.1109/JSSC.2016.2616357

Cheung, C., Hillman, A., & Remmen, G. N. (2024). Bootstrap principle for the spectrum and scattering of strings. *Physical Review Letters*, *133*(25), 251601. https://doi.org/10.1103/PhysRevLett.133.251601

Clattner, C., et al. (2021). MLIR: Scaling Compiler Infrastructure for Domain Specific Computation. *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, 2–14. https://doi.org/10.1109/CGO51591.2021.9370308

Cortade, G., et al. (2023). Efficient ML Computing: A Survey of Hardware Acceleration for Deep Learning. *Journal of Machine Learning Systems*, *2*(1), 55–78.

Czech, B., et al. (2021). Building tensor networks for holographic states. *Journal of High Energy Physics*, *2021*(5), 9. https://doi.org/10.1007/JHEP05(2021)009

Fukui, K., & Tashima, T. (2024). Efficient realization of quantum algorithms with qudits. *EPJ Quantum Technology*, *11*(1), 43. https://doi.org/10.1140/epjqt/s40507-024-00250-0

Giddings, S. B. (2013). Is string theory a theory of quantum gravity? *Foundations of Physics*, *43*(1), 115–139. https://doi.org/10.1007/s10701-012-9708-2

Günter, H., & Petrucci, F. (2019). Gravitational qubits: Spin–gravity coupling and two-level systems in curved spacetime. *Universe*, *5*(5), 123. https://doi.org/10.3390/universe5050123

Harris, M. (2005). Chapter 31. mapping computational concepts to GPUs. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation* (pp. 493–512). Addison-Wesley.

Huang, H.-J., Zhang, T., Zhao, Y., & Wang, X. (2023). Quantum-inspired machine learning: A review and roadmap. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2023.3257187

Ielmini, D., & Wong, H.-S. P. (2018). In-memory computing with resistive switching devices. *Nature Electronics*, *1*, 333–343. https://doi.org/10.1038/s41928-018-0092-2

Intel Corp. (2025). Intel® oneAPI programming guide [v2025.x].

Jelinčič, A., et al. (2025). An efficient probabilistic hardware architecture for diffusion-like models [preprint]. *arXiv preprint arXiv:2510.23972*.

Jouppi, N. P., et al. (2017). In-datacenter performance analysis of a tensor processing unit. *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. https://doi.org/10.1145/3079856.3080246

Khronos OpenCL Working Group. (2025). OpenCL 3.0 unified specifications (v3.0.19) [The Khronos Group].

Khronos SYCL Working Group. (2025). SYCL 2020 specification (revision 10) [The Khronos Group].

Noyes, H. P. Bit-string physics: A novel theory of everything [SLAC-PUB-6509; CONF-9411171-1; OSTI ID 28404]. In: In *Proceedings, international symposium on fundamental physics (1994)*. SLAC-PUB-6509; CONF-9411171-1; OSTI ID 28404. 1994.

NVIDIA. (2023). CUDA Accelerated Computing – Energy Efficiency Gains of GPU Acceleration [Accessed 2025-11-07].

NVIDIA. (2024). *GPU Performance Background User's Guide* (tech. rep.). NVIDIA. https://docs.nvidia.com/deeplearning/performance/dl-performance-gpu-background/index.html

OpenMP Architecture Review Board. (2024). OpenMP Application Programming Interface Version 6.0.

OpenXLA Project. (2025). OpenXLA: Portable, scalable ML compiler stack.

Perinotti, P. (2012). Discord and nonclassicality in probabilistic theories. *Physical Review Letters*, *108*(12), 120502. https://doi.org/10.1103/PhysRevLett.108.120502

Ragan-Kelley, J., et al. (2012). Decoupling algorithms from schedules for easy optimization of image processing pipelines. *ACM Transactions on Graphics (SIGGRAPH 2012)*, *31*(4). https://doi.org/10.1145/2185520.2185528

Ren, F., Marković, D., & Dorrance, R. (2023). A scalable sparse matrix-vector multiplication kernel for energy-efficient sparse-blas on fpgas [Preprint available via Arizona State University; see https://asu.elsevierpure.com/en/publications/a-scalable-sparse-matrix-vector-multiplication-kernel-for-energy-efficient-sparse-blas-on-fpgas]. *Proceedings of [Conference on Reconfigurable Computing / Sparse BLAS Acceleration] (exact venue TBD)*.

Sze, V., et al. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, *105*(12), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

Thorbeck, T., et al. (2023). Two-level-system dynamics in a superconducting qubit due to background ionizing radiation. *PRX Quantum*, *4*(020356). https://doi.org/10.1103/PRXQuantum.4.020356

Vedral, V. (2010). *Decoding reality: The universe as quantum information*. Oxford University Press.