Sparsely Wired Mortal LLM Inference

Mincheol Park, Sunwoo Lee, Sukjin Lee, Wooram Yang,
Yongmin Tai, Sang Joon Kim, Jaehoon Yu*
Samsung Advanced Institute of Technology
{danimc.park, sw.o.lee, sukjin57.lee, wooram.yang,
ym.tai, sangjoon0919.kim, jae-hoon.yu}@samsung.com

Abstract

This paper rethinks Large Language Model (LLM) inference from a *Mortal Computing* perspective, in which computation is embodied in hardware and functionality is inseparable from its physical substrate. Rather than treating inference optimization as merely a matter of arithmetic simplification or precision reduction, we explore hardwiring as a way to eliminate weight access at the root of data movement. We show that viable hardwiring begins not by retaining weights in their multiplicative form, but by factorizing them into combinations of shiftable bases and rewiring computation at the base level, where redundancy is exposed. This view, however, introduces a fundamental challenge in the dramatic growth of wires required to reconstruct weights. To address this, we propose *Moira*, a budgeted greedy search that formulates wire-count control as a Lagrangian relaxation problem and selects only the bases that satisfy the objective under a wiring budget. What sets Moira apart is its adaptive base selection, which gives it a dual characteristic of pruning and quantization while preserving model expressivity under sparse wiring without retraining. Extensive evaluations demonstrate that Moira achieves near-lossless performance with, on average, only about two bases per weight and outperforms existing compression methods. Moira opens a concrete path beyond the von Neumann paradigm, marking a first step toward *Digital Mortal Computing* for LLMs. The code is available at `https://github.com/SAITPublic/Moira`.

1 Introduction

Large Language Models (LLMs) have attracted significant attention as a key enabler of cutting-edge AI applications [1, 2] and natural language understanding [3, 4]. Their remarkable capabilities are largely driven by their enormous scale, often reaching tens of billions of parameters [5]. However, this scale comes at the cost of substantial inference and poses a major barrier to efficient deployment. As a workaround, model compression techniques, pruning [6] and quantization [7], have become central to faster inference. Yet reducing the arithmetic or bit width alone does not necessarily eliminate the dominant cost of data movement. Their need for continued retrievals of sparse and mixed-precision data remains a source of memory accesses, leaving the core bottlenecks unresolved.

One possible way to address the cost of data movement at its root is *Mortal Computing* [8], which treats software and its physical substrate as inseparable. It is *mortal* because computation is bound to a device's idiosyncratic analog properties and thus cannot be replicated across substrates. While highly energy efficient, this also makes computation inherently device-specific. We revisit this idea through a reproducible digital path, which we term *Digital Mortal Computing*. Although digitally replicable, it remains *mortal* in a broader sense because computation is still inseparable from the hardwired substrate. In digital mortal computing, fixed LLM-weight multiplications are embedded into the substrate [9], eliminating weight fetches from memory and producing outputs in place.

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

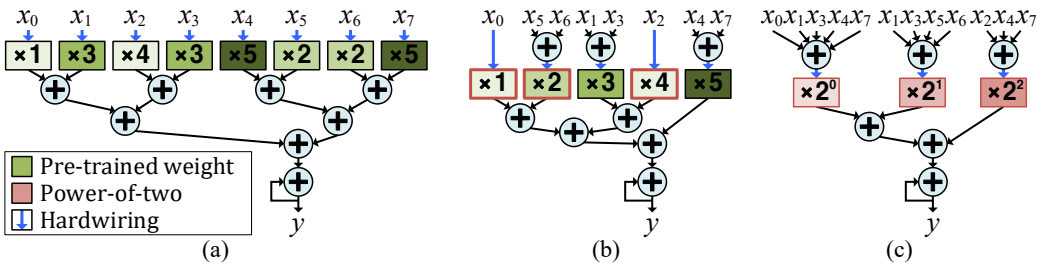


Figure 1: Example MAC circuits for digital mortal computing: **(a)** full hardwiring, preserving high throughput with large area from redundant multiplications; **(b)** redundancy-aware hardwiring via distributivity, merging repeated weights; **(c)** base-weight hardwiring with shared power-of-two bases, replacing multiplication with shifts and enabling optimized adders despite increased connectivity.

Despite its promise for rapid inference, this approach faces a major trade-off in constructing feed-forward circuits. As shown in Figure 1, implementing multiplication as a primitive operation in a multiply-accumulate (MAC) circuit leads to substantial area due to redundant, individual constant multipliers, whereas simplifying the multiplication logic tends to increase the number of wiring connections. Recent approaches focus on reducing area due to not feasibility but economic struggles [9]. Still, the proliferation of physical wires and their extensive connectivity poses real fabrication challenges, ranging from layout complexity to manufacturing mask generation.

In this paper, we propose Moira to accelerate LLMs in a new aspect of mortal computation that places a minimal set of constant multipliers and realizes computation only through essential wire assignment. As shown in Figure 1(c), our central idea is to restrict these multipliers not to arbitrary weight values, but to powers of two. This converts multiplication into faster *shift* [10] and limits weights represented by a given precision format (*e.g.*, FP16) to the most indivisible computing element. This construction, however, expands each original weight connection into a combinatorial set of candidate wires to the elements. We then propose a wire-assignment formula formulated as a Lagrangian relaxation problem to control wire counts and suppress wiring growth while preserving model expressivity.

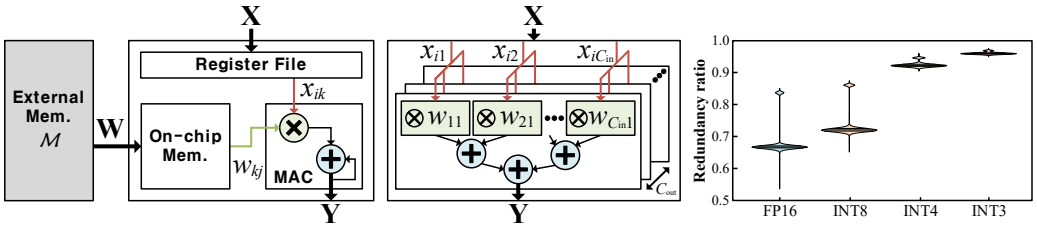
Merely by expanding over a set of power-of-two bases, Moira reveals an intriguing duality, in that removing all wires used for a weight is equivalent to pruning, whereas restricting each weight to only a few wires amounts to quantization. Yet unlike either, Moira performs adaptive base selection while obviating the need for retraining. Remarkably, our tests confirm that across deployed LLMs, about two bases per weight can suffice on average to preserve performance almost losslessly. This highly sparse wiring brings LLM inference closer to a key property of mortal computation, extreme energy efficiency by the elimination of weight access. To our knowledge, Moira is a first step toward efficient Digital Mortal Computing of LLMs, opening a concrete path beyond the von Neumann paradigm.

Our contributions can be summarized as threefold:

- We shed new light on LLM acceleration by shifting the focus from reducing weight footprint to eliminating weight access by hardwiring. Our formal analysis establishes that the key is not to hardwire multiplications as given, but to break them into shiftable values and rebuild computation using shift-accumulate units instead of MACs. This perspective opens a new path toward digital mortal computation of LLMs beyond the von Neumann paradigm.
- We present wire explosion as the key challenge of establishing hardwired shiftable circuits and formulate its control as a Lagrangian relaxation problem. The proposed nested greedy search remains simple and effectively matches the target budget once the iteration is fixed.
- We confirm that minimizing connections across indivisible compute units gives Moira a dual nature of pruning and quantization. Our tests show that its distinct, adaptive selection of critical bases yields more robust and efficient LLMs than existing compression methods.

2 Related Work

LLM pruning and quantization. A large body of work has aimed to accelerate LLM inference by compressing `gemv` and `gemm` operations. To reduce the need for costly retraining, prior methods incorporate activation-aware scoring alongside weight magnitudes, enabling more accurate pruning



(a) Von Neumann-style MAC (b) Hardwired constant MAC (c) On weights of LLaMA-2 7B

Figure 2: MAC design overview and breakdown of weight redundancy: **(a)** a von Neumann-style MAC requires two operands from register files or memory subsystem to compute an output; **(b)** a hardwired constant MAC produces the output immediately once the input operand is given; **(c)** a violin plot of per-channel weight redundancy across all linear layers, defined as $1-|\mathrm{unique}(\mathbf{W}_{:j})|/\mathrm{n}(\mathbf{W}_{:j})$, where $n(\cdot)$ is the number of entries, from FP16 to lower-precision formats obtained via GPTQ [7].

and quantization decisions. In pruning, such criteria [6, 11–13] can produce either irregularly sparse or semi-structured tensors [14], but acceleration depends on divergent memory accesses [15] or dedicated tensor cores [16, 17] to process the sparsity. Quantization faces a similar problem. As the model size grows, error-sensitive values motivate mixed-precision allocation [18, 19] and error-propagation techniques [7]. However, these approaches also require dedicated hardware kernels due to heterogeneous precision formats [20], lookup tables [21], and fine-tuning [22]. Unlike pruning or quantization, our method focuses not on direct reductions in computation or on rounding to small bit-widths, but on factorizing weights for efficient shift-based execution and on controlling wire growth for the actual realization of mortal-compute circuits.

Multiplication-less models. A separate line of work seeks to accelerate inference by replacing costly multiplications in `gemv` and `gemm` operations with simpler computational primitives, including binary [23] or ternary [24], and shift-and/or-add forms [25–28]. These ideas, first explored in CNNs [29, 30], have since been extended to ViTs [31] and LLMs [32], including attention-specific variants [33, 34]. Despite their diversity, most of these approaches obtain efficiency by rewriting the original model into a new arithmetic form, often through training from scratch [35], substantial fine-tuning [31], or explicit reparameterization [36] with lookup tables [10]. In contrast, our method starts with a deployed floating-point LLM and avoids applying a uniform shift term across weights. Instead, it adaptively selects only the minimal set of shifters for each weight, resulting in a dynamic decomposition rather than a uniformly reparameterized, multiplier-less model.

3 Prerequisite: Digital Mortal Computation on Constant MAC

Digital mortal computation seeks to materialize software dataflow directly in the underlying hardware. This contrasts with traditional von Neumann-style execution, where arithmetic operations require that operands be loaded first. This execution makes data movement a major source of latency and energy when input data and weights must be fetched through the memory hierarchy.

As shown in Figure 2, let $\mathbf{X} \in \mathbb{R}^{T\times C_{\text{in}}}$ be the input data, $\mathbf{W} \in \mathbb{R}^{C_{\text{in}}\times C_{\text{out}}}$ the weight matrix, and $\mathbf{Y} \in \mathbb{R}^{T\times C_{\text{out}}}$ the output matrix. Common linear layers in LLMs typically compute as follows:

$$\mathbf{Y} = \mathbf{X}\mathbf{W}; \quad y_{ij} = \sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} w_{kj}, \quad \text{s.t.} \quad w_{kj} \xleftarrow{\text{ld}} \mathcal{M}, \tag{1}$$

where T is a token length. This computation proceeds as a sequence of MAC operations. Here, we consider a dataflow in which input data $\mathbf{X}$ are consumed sequentially and already reside in the register file for subsequent execution. By contrast, weights $\mathbf{W}$ still require repeated loading from the memory subsystem $\mathcal{M}$, since their large footprint and limited on-chip memory capacity often prevent them from remaining on chip [19].

Mortal computing breaks this computation by etching primitive computation directly into the physical substrate. In other words, weight itself is absorbed into hardwired constant operators. To distinguish this from the original MAC, we introduce an absorbed operator $\otimes$, and write the new computation as:

$$\mathbf{Y} = \mathbf{X} \otimes \mathbf{W}; \quad y_{ij} = \sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \left[\otimes w_{kj}\right], \tag{2}$$

where $[\otimes w]$ represents a hardwired multiplication and connection applied to corresponding inputs.

Since weights are represented in a finite-precision format, the constants appearing in $\mathbf{W}$ are finite and may be redundant. Therefore, we can consider both sets $\mathcal{U}_j = \{u \in \mathbb{R} \mid \exists k,\ w_{kj} = u\}$ and $\mathcal{K}_j(u) = \{k \in \{1, ..., C_{\text{in}}\} \mid w_{kj} = u\}$ for an output channel j. We then obtain a distributive merging property for repeated constants, where inputs sharing the same constant are pre-summed before a single absorbed multiplication.

Definition 1 (Distributive property)**.** For every fixed constant $u \in \mathbb{R}$, the absorbed arithmetic operator $\circ u$ is linear in its left operand. Then, it satisfies the distributive property as follows:

$$x \circ u + x' \circ u = (x + x') \circ u. \tag{3}$$

Proposition 1. *Following the distributive property, where every fixed constant u and its multiplication $\otimes u$ is linear in its left operand, for every token, the hardwired constant MAC is formulated as:*

$$y_{ij} = \sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \left[\otimes w_{kj}\right] = \sum\nolimits_{u \in \mathcal{U}_j} \left(\sum\nolimits_{k' \in \mathcal{K}_j(u)} x_{ik'} \right) \left[\otimes u\right], \tag{4}$$

where $[\otimes u]$ denotes hardwired multiplication to summing inputs over $\mathcal{K}_j(u)$. Proposition 1 holds that repeated $\otimes u$ is simplified before execution by first aggregating inputs that share the same constant. However, as shown in Figure 1(b), this formulation still treats full constants as primitive coefficients and is too coarse to capture redundancy exposed under the deployed precision, as seen in Figure 2(c).

4 Methodology

This section revisits establishing hardwired constant MACs in Proposition 1, makes them over finer shiftable bases, and formulates an optimization that controls the resulting growth of candidate wires.

4.1 Rebuilding Weights to Shiftable Bases

A key property of the given precision format is that it can bound the deployed weights and their space. More specifically, each deployed weight is represented in a floating-point format. Typical precision formats represent each value through a sign, an exponent, and a mantissa (or significand). Without loss of generality, each weight is written as $w := \mathrm{sm}2^{\mathrm{e}}$. Here, we take e as the effective exponent after accounting for the format bias, and s as the sign. Then, m admits the binary expansion:

$$\mathrm{m} = 1 + \sum\nolimits_{b=1}^{B} \mathcal{B}(b) \cdot 2^{-b}, \quad \mathcal{B}(b) \in \{0, 1\}, \tag{5}$$

where $\mathcal{B}(b)$ indicates whether the b-th bit of the B-bit mantissa is active, taking the value 1 if the bit is set and 0 otherwise. Substituting Eq. (5) into the weight expression directly shows that each finite-precision weight is a finite signed sum of powers of two:

$$w = \mathrm{s} \left(2^{\mathrm{e}} + \sum\nolimits_{b=1}^{B} \mathcal{B}(b) \cdot 2^{\mathrm{e}-b} \right) \triangleq \sum\nolimits_{e \in \mathcal{E}_w} \tau_w(e) \cdot 2^e, \tag{6}$$

where the set $\mathcal{E}_w = \{\mathrm{e}\} \cup \{\mathrm{e} - b \mid \mathcal{B}(b) = 1,\ b = 1, \cdots, B\}$ collects all active exponents induced by the finite-precision representation of w, and $\tau_w(e) \in \{-1, 0, +1\}$ specifies the signed contribution of each active power-of-two term with respect to w. This form exposes two core advantages. First, the weight space is reduced to a finite numerical space: under a given floating-point format, each deployed weight activates only a finite set of exponents. Second, every model parameter is shiftable, since multiplication by 2^e is established purely as a shift. Thus, full constant MACs can be replaced with a direct route to more efficient, multiplication-free [10] hardwired shift-accumulate circuits.

Accordingly, our mortal computing circuit is defined by replacing each full weight coefficient w with its base-factorized form. In other words, the absorbed shift operations in linear layers hold:

$$\mathbf{Y} = \mathbf{X} \otimes \mathbf{W}; \quad y_{ij} = \sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \left[\otimes \sum\nolimits_{e \in \mathcal{E}_{w_{kj}}} \tau_{w_{kj}}(e) \cdot 2^e \right]. \tag{7}$$

This factorization allows the linear accumulation to be reorganized in a base-centric manner.

Proposition 2. *Digital mortal computation in Eq.* (7) *expands into exponent-wise terms by the distributive property. Since all resulting sums are finite, Eq.* (7) *can be rewritten with shared exponent bases. Suppose $E_j = \cup_{k=1}^{C_{\text{in}}} \mathcal{E}_{w_{kj}}$, and define $\tau_{w_{kj}}(e) = 0$ for $e \notin \mathcal{E}_{w_{kj}}$. Then, we have:*

$$y_{ij} = \sum\nolimits_{e \in E_j} \left(\sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \cdot \tau_{w_{kj}}(e) \right) \left[\otimes 2^e\right], \tag{8}$$

where $[\otimes 2^e]$ represents hardwired base connection applied to candidate inputs x_{ik} with $\tau_{w_{kj}}(e) \neq 0$.

Proposition 2 shows that the base set E_j is finite and fixed by the given precision format. The main challenge lies in representing a larger weight space through signed combinations of these bases. This gives rise to many possible exponent-support patterns and, in turn, a larger candidate wire space. Section 4.2, therefore, focuses on finding a compact wire set while preserving model expressivity.

4.2 Minimization of Candidate Wires

In this section, we aim to determine how many hardwired shiftable bases should be assigned to each weight, given a prescribed wire budget. Using more bases improves reconstruction accuracy, but also increases wiring cost. To balance this trade-off, we introduce a Lagrange multiplier associated with the target number of wires per weight.

To this end, we exploit an activation-aware reconstruction score to well represent the next-layer input features [7, 13]. For a fixed weight w, let $\tilde{w}^N$ denote its approximation using N hardwired shiftable bases. We then consider the following minimization problem:

$$\min_{N_{kj}} \sum_{k=1,j=1}^{C_{\text{in}},C_{\text{out}}} |w_{kj} - \tilde{w}_{kj}^{N_{kj}}| \cdot \tilde{\mathbf{X}}_k, \quad \text{s.t.} \quad \frac{1}{C_{\text{in}}C_{\text{out}}} \sum_{k=1,j=1}^{C_{\text{in}},C_{\text{out}}} N_{kj} \leq N_{\text{target}}, \tag{9}$$

where $\tilde{X}_k$ is an expectation of norm of input activations [13]. Eq. (9) favors more accurate weight approximations under the observed activation scale, while the constraint limits the average number of instantiated wires. In other words, each weight is allowed to use more bases only when the resulting reduction in activation-weighted residual is worth the additional wiring cost.

This constrained problem in Eq. (9) is handled through a Lagrangian relaxation, yielding the following per-weight objective for selecting the factorization depth:

$$N_{kj}(\lambda) = \text{argmin}_N \, \mathcal{J}_{kj}(N;\lambda); \quad \mathcal{J}_{kj}(N;\lambda) = |w_{kj} - \tilde{w}_{kj}^N| \cdot \tilde{\mathbf{X}}_k + \lambda N. \tag{10}$$

Eq. (10) shows that the zero-wire case corresponds to pruning a weight, consistent with a method [13]. An additional wire is accepted only when the reduction in weighted residual exceeds the corresponding wire price. More generally, increasing the factorization depth from N to $N+1$ is beneficial only if:

$$\left(|w_{kj} - \tilde{w}_{kj}^N| - |w_{kj} - \tilde{w}_{kj}^{N+1}|\right) \cdot \tilde{\mathbf{X}}_k > \lambda. \tag{11}$$

Therefore, the factorization depth is not fixed a priori. Instead, each weight adaptively selects the smallest number of hardwired power-of-two bases that is worthwhile under the current wire price λ. Weights that are well approximated with only a few bases stop early, whereas weights whose residual continues to decrease substantially are assigned deeper factorization.

Budgeted greedy search. The remaining tasks are to construct the residual factorization for each weight and to fit λ so that the resulting average wire count matches the target budget. For each weight w_{kj}, we build its approximation progressively from the residual $w_{kj} - \tilde{w}_{kj}^N$ as follows:

$$\tilde{w}_{kj}^{N+1} = \tilde{w}_{kj}^N + b_{kj}^N; \quad b_{kj}^N = \text{sign}\left(w_{kj} - \tilde{w}_{kj}^N\right) 2^{e_{kj}^N}; \quad e_{kj}^N = \text{round}\left(\log_2 |w_{kj} - \tilde{w}_{kj}^N|\right), \tag{12}$$

where $\tilde{w}_{kj}^0 = 0$ and a initial residual is w_{kj}. This iteration continues only while the newly added bases remain worthwhile under Eq. (11). Specifically, the next base is accepted only if it reduces the activation-weighted residual by more than the corresponding wire price λ. Once this condition is no longer satisfied, the factorization stops, and the resulting depth is taken as $N_{kj}(\lambda)$. The remaining task is to choose λ. We perform this by binary search, using the average number of selected wires as the fitting result $N_{\text{target}} \simeq \bar{N}(\lambda) = \sum_{kj} N_{kj}(\lambda)/\text{n}(\mathbf{W})$. The overall budgeted greedy search, Moira, is summarized in Algorithms 1 and 2. In Section 5, we evaluate the performance using the reconstructed weights $\tilde{\mathbf{W}}$ produced by Moira, obtained by defactorizing the selected bases. Note that the actual circuit implementation is determined by the bases and their direct hardwiring.

5 Experiment

In this section, we aim to answer the following three main questions: (1) how effective Moira is, relative to pruning and quantization methods as practical alternatives, for reducing wires toward efficient mortal computing circuit design; (2) how perplexity varies with the averaged target wire count N_{target} across LLMs of different sizes; (3) whether Moira yields $\bar{N}$ close to N_{target} in practice.

Algorithm 1 Greedy Weight Factorization

Input: $w_{kj} \in \mathbf{W}, \tilde{\mathbf{X}}_k, \lambda$
Initialize: $n = 0, \tilde{w}_{kj}^0 = 0$
repeat ▷ $\forall w_{kj}$ **in parallel**
  Compute an active exponent e_{kj}^n and so a base b_{kj}^n
  Form current $\tilde{w}_{kj}^{n+1} \leftarrow \tilde{w}_{kj}^n + b_{kj}^n$
  Set both residuals $|w_{kj} - \tilde{w}_{kj}^n|$ and $|w_{kj} - \tilde{w}_{kj}^{n+1}|$
  Fix $N \leftarrow n$ and then $n \leftarrow n + 1$
until Eq.(11) is **no longer** satisfied under λ and $\tilde{\mathbf{X}}_k$
Output: $\forall N_{kj}, \tilde{\mathbf{W}} = \{\tilde{w}_{kj}^N \mid \forall k, j\}$

Algorithm 2 Moira via Wire-Budget Fitting

Input: $\mathbf{W}, \tilde{\mathbf{X}}_k, N_{\text{target}}, L$
Initialize: $\lambda_{\text{low}} = 0, \lambda_{\text{high}} = \max|w_{kj}| \cdot \tilde{\mathbf{X}}_k$
repeat
  Set $\lambda_{\text{mid}} \leftarrow (\lambda_{\text{low}} + \lambda_{\text{high}})/2$
  Get $\forall N_{kj}(\lambda_{\text{mid}})$, $\tilde{\mathbf{W}}$ for $\mathbf{W}$ and λ_{mid} ▷ Alg. 1
  if $\sum N_{kj}(\lambda_{\text{mid}})/\text{n}(\mathbf{W}) > N_{\text{target}}$ **then**
    Set $\lambda_{\text{low}} \leftarrow \lambda_{\text{mid}}$ **else** Set $\lambda_{\text{high}} \leftarrow \lambda_{\text{mid}}$
until $\bar{N}(\lambda_{\text{mid}}) \simeq N_{\text{target}}$ or L iters. are reached
Output: $\tilde{\mathbf{W}}$

Metric. To compare Moira with pruning and quantization under a common notion of wiring cost, we measure wiring cost by the number of signed power-of-two components required to represent each scalar value. For a scalar value z, let $\hat{z}^M$ denote the result of the lossless residual factorization after M steps. Starting from $\hat{z}^0 = 0$, we extract the nearest signed power-of-two term following Eq. (12):

$$\mathcal{N}(\mathbf{Z}) = \sum\nolimits_{i \in \text{dim}(\mathbf{Z})} \mathcal{N}(z_i); \quad \mathcal{N}(z_i) := \min\{M \in \mathbb{N} \mid \hat{z}_i^M = z_i\}. \tag{13}$$

Here, $\mathbf{Z}$ may denote sparsely pruned weights or quantized weight values after zero-point correction, and in the quantized case it also includes associated scales. Since each $z \in \mathbf{Z}$ is represented in a finite deployment precision format, this residual factorization terminates after a finite number of steps M.

We validate this proxy in Eq. (13) through actual hardware synthesis in Section 5.4 to examine whether it faithfully reflects the physical area cost across pruning and quantization methods.

Dataset and LLMs. We use WikiText-2 [37] for evaluation of large language models, using C4 [38] for calibration. We further report zero-shot results on BoolQ, RTE, HellaSwag (HellaS), WinoGrande (WinoG), ARC-Easy (ARC-E), ARC-Challenge (ARC-C), and OpenBookQA (OBQA) using `LM-Eval-Harness` [39], following the evaluation settings in [11]. We evaluate all methods on large language models of FP16 ranging from 7B to 70B parameters. Our experiments include public models such as LLaMA-2 7B, 13B, and 70B [40], as well as LLaMA 30B and 65B [41].

Baseline methods and experimental settings. For comparison with existing post-training efficient AI methods, we consider score-based pruning baselines, including Magnitude [42], SparseGPT [6], Wanda [13], and Pruner-Zero [11], as well as quantization baselines, including Round-to-Nearest (RTN), GPTQ [7], AWQ [19], and `e2m1` Float4 (F4) of QLoRA [22]. We apply Moira independently to every linear layer of Transformer blocks. We use $N_{\text{target}} = \{1.5, 1.1\}$ to compare with pruning and quantization, respectively, and set the full-step L to 30. The resulting allocations are then aggregated across layers and reported as our final wire count $\mathcal{N}$. For a fair comparison, we also apply Eq. (13) to non-zero pruned weights and quantized weights after computing zero-points, including their associated scales. Appendix C provides further implementation details.

5.1 Comparison with Pruning for Wire Reduction

Performance. Table 1 demonstrates that Moira achieves a better trade-off between wire reduction and model performance than existing pruning baselines. Compared with pruning methods that typically require about 2.0 wires per weight on average, Moira uses only about 1.5 wires per weight on average while incurring only negligible perplexity degradation. This result highlights the benefit of moving from weight-level pruning to a finer-grained optimization [16, 43] in the space that spans each weight, allowing the model to preserve only the necessary sub-components under a stricter wire-efficiency objective. Although some pruning methods remain competitive on certain zero-shot tasks, Moira attains the best average performance while using substantially fewer wires.

Wire distribution. Figure 3 visualizes how the wires in Table 1 are distributed across power-of-two bases. While the pruning baselines exhibit broadly similar base-level wire distributions after factorization, Moira yields a much more concentrated allocation, using substantially fewer wires while focusing them on a narrow range of bases from 2^{-5} to 2^{-9}. Despite using only a small number of bases, this concentrated allocation maintains performance close to that of the vanilla. This suggests that, under a tight wire budget, performance is better preserved by concentrating representation capacity in the most expressive bases rather than spreading wires across the finer space.

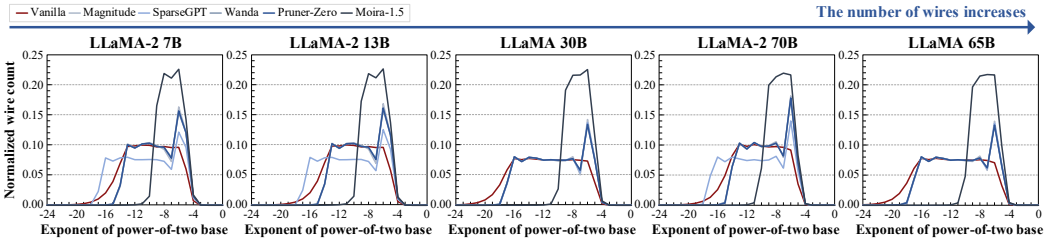


Figure 3: Comparison of wire densities between pruning baselines (50% unstructured sparsity) and Moira-1.5. The x-axis denotes available power-of-two bases, and the y-axis shows the normalized wire count associated with each base. Each count is obtained by dividing by the total wires $\mathcal{N}$.

Method	# Wires ($\mathcal{N}$)	Wikitext PPL	BoolQ.	RTE	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	AVG.
LLaMA-2 7B	22.20B	5.12	79.35	63.53	57.10	69.45	75.42	43.00	33.20	60.15
- Magnitude	11.45B	14.89	61.19	55.95	49.42	63.06	63.76	35.75	27.80	50.99
- SparseGPT	14.37B	6.51	75.81	54.87	52.78	69.06	70.91	38.22	28.60	55.75
- Wanda	11.50B	6.46	77.24	53.06	52.35	67.24	72.81	39.76	31.20	56.24
- Pruner-Zero	11.05B	6.26	70.94	53.06	51.10	67.24	72.30	37.79	28.00	54.35
- **Moira-1.5**	**9.70B**	**5.13**	79.54	60.64	57.09	69.37	76.17	43.25	33.60	**59.95**
LLaMA-2 13B	43.50B	4.57	82.14	67.14	60.18	72.53	78.95	47.35	34.60	63.27
- Magnitude	21.59B	6.37	57.70	57.40	53.79	61.32	68.39	38.90	24.20	51.67
- SparseGPT	27.93B	5.63	81.65	65.34	55.90	72.69	74.62	41.97	32.80	60.71
- Wanda	21.66B	5.58	81.77	58.48	57.07	70.79	76.38	43.85	33.20	60.22
- Pruner-Zero	21.66B	5.36	80.61	65.34	56.20	72.05	75.50	41.38	32.80	60.55
- **Moira-1.5**	**19.03B**	**4.59**	81.80	66.06	60.18	72.21	78.99	47.61	35.60	**63.21**
LLaMA 30B	142.09B	4.10	83.70	68.23	63.36	75.76	80.75	51.53	36.40	65.67
- Magnitude	71.06B	7.54	64.50	54.51	50.73	65.43	58.20	32.08	29.00	50.63
- SparseGPT	71.02B	5.33	83.80	57.03	59.23	75.05	79.50	49.23	34.80	62.66
- Wanda	71.00B	5.24	81.60	66.42	60.92	75.13	79.60	49.48	35.00	64.02
- Pruner-Zero	71.05B	5.01	81.35	68.95	59.99	73.95	78.70	48.72	33.60	63.61
- **Moira-1.5**	**48.14B**	**4.11**	83.66	67.50	63.25	76.63	81.18	51.62	37.20	**65.86**
LLaMA-2 70B	234.74B	3.12	85.40	69.31	65.31	80.42	82.85	53.83	36.20	67.62
- Magnitude	118.33B	4.98	71.65	58.84	61.56	75.61	77.25	48.37	33.20	60.92
- SparseGPT	152.43B	3.98	85.25	72.81	62.68	78.76	81.75	52.47	36.20	67.13
- Wanda	117.78B	3.97	83.55	72.56	62.67	79.40	81.40	52.73	38.00	67.18
- Pruner-Zero	118.24B	3.82	84.40	73.64	62.38	78.13	81.20	52.21	36.20	66.88
- **Moira-1.5**	**102.68B**	**3.13**	85.20	67.87	65.96	80.50	83.00	54.35	36.60	**67.64**
LLaMA 65B	286.63B	3.53	85.30	70.39	64.94	77.19	81.05	52.21	38.40	67.07
- Magnitude	143.76B	5.73	80.15	64.25	61.68	74.50	75.40	48.46	37.20	63.09
- SparseGPT	143.49B	4.59	84.10	70.75	62.17	77.26	79.75	49.48	38.80	66.04
- Wanda	143.48B	4.60	85.45	72.20	62.86	76.87	80.35	50.76	39.40	66.84
- Pruner-Zero	143.54B	4.35	85.65	71.48	61.97	76.63	79.75	49.31	36.60	65.91
- **Moira-1.5**	**97.09B**	**3.54**	85.00	72.39	64.46	77.11	81.00	52.38	38.60	**66.99**

Table 1: Comparison of sparsely pruned weight configurations for a hardwired constant-MAC design. Moira is compared against score-based pruning baselines at 50% unstructured sparsity. Models (FP16) are calibrated on C4 and evaluated on WikiText-2 and zero-shot tasks following Pruner-Zero [11]. The wire count $\mathcal{N}$ for the pruning methods is calculated based on the non-zero weights.

5.2 Comparison with Quantization for Wire Reduction

Performance. Representing weights with only a small number of power-of-two bases may appear similar to quantization. Table 2 compares Moira with prior quantization methods using, for each benchmark, the smallest N_{target} that reaches its best perplexity, together with the resulting wire counts and zero-shot performance. Existing quantization methods typically require about 1.1–1.2 wires per weight on average for INT4 and about 1.3 for FP4, whereas Moira uses fewer wires while achieving comparable or better perplexity. Zero-shot results vary across tasks, but Moira yields the best average performance. This indicates that, rather than determining the represented value solely through aggressive rounding, explicitly scoring and preserving the most important bases is more effective for highly efficient modeling, as further evidenced in Figure 6(b).

Wire distribution. Figure 4 shows how wires are distributed across power-of-two bases for quantization methods and Moira. The distribution in quantization largely depends on the representation format: FP4 requires additional bases to express subnormal values, leading to a broader, bimodal distribution and a larger total wire count. As discussed in Section 5.1, this distribution, which is not concentrated on the expressive bases, does not necessarily lead to better performance. By contrast,

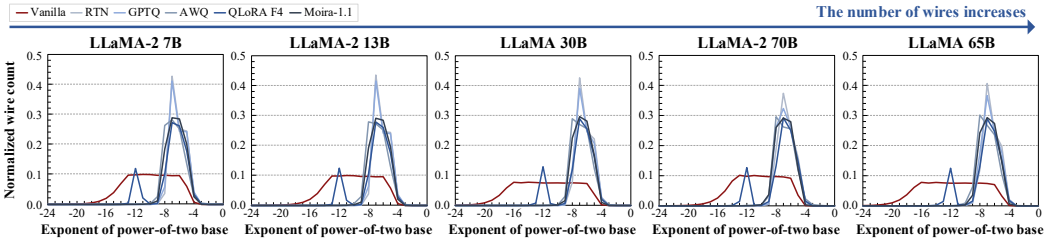


Figure 4: Comparison of wire densities between quantization baselines for INT4-g128 and FP4 and Moira-1.1. The x-axis denotes available power-of-two bases, and the y-axis shows the normalized wire count associated with each base. Each count is obtained by dividing by the total wires $\mathcal{N}$.

Method	# Wires ($\mathcal{N}$)	Wikitext PPL	BoolQ.	RTE	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	AVG.
LLaMA-2 7B	22.20B	5.12	79.35	63.53	57.10	69.45	75.42	43.00	33.20	60.15
- RTN	7.29B	5.39	77.55	64.62	56.54	69.29	75.08	42.83	32.00	59.70
- GPTQ	7.31B	5.29	76.94	62.45	55.87	68.50	74.49	42.83	32.60	59.10
- AWQ	7.64B	5.24	79.57	58.48	56.53	68.19	75.54	42.91	32.20	59.06
- QLoRA F4	8.57B	5.42	78.07	60.28	56.64	68.90	74.74	41.97	33.20	59.11
- **Moira-1.1**	**7.12B**	**5.24**	79.51	63.17	57.03	69.13	75.58	43.51	34.00	**60.28**
LLaMA-2 13B	43.50B	4.57	82.14	67.14	60.18	72.53	78.95	47.35	34.60	63.27
- RTN	14.31B	4.72	81.89	64.98	59.64	71.66	78.28	47.18	35.00	62.66
- GPTQ	14.33B	4.69	80.61	65.34	59.72	72.05	78.57	45.98	33.60	62.27
- AWQ	14.99B	4.65	81.95	65.70	60.01	73.79	78.57	46.84	34.20	63.01
- QLoRA F4	16.80B	4.75	80.36	68.59	59.51	71.58	78.11	45.81	34.40	62.62
- **Moira-1.1**	**13.96B**	**4.65**	81.92	66.81	59.96	72.92	78.57	47.09	35.40	**63.24**
LLaMA 30B	142.09B	4.10	83.70	68.23	63.36	75.76	80.75	51.53	36.40	65.67
- RTN	36.60B	4.30	82.29	66.78	62.84	76.00	80.59	50.00	35.60	64.87
- GPTQ	36.42B	4.24	83.54	68.95	62.67	76.87	80.05	50.68	36.00	65.54
- AWQ	38.11B	4.20	84.00	67.87	63.27	76.71	80.76	50.93	36.60	65.73
- QLoRA F4	42.84B	4.34	83.82	67.87	62.63	75.05	80.55	50.51	36.40	65.26
- **Moira-1.1**	**35.31B**	**4.19**	83.27	68.95	63.35	76.24	80.89	51.87	36.40	**65.85**
LLaMA-2 70B	234.74B	3.12	85.40	69.31	65.31	80.42	82.85	53.83	36.20	67.62
- RTN	77.73B	3.21	84.10	69.67	64.48	79.16	82.45	52.64	35.20	66.81
- GPTQ	77.69B	3.22	84.70	69.92	64.94	79.40	82.60	53.66	36.20	67.34
- AWQ	81.16B	3.20	85.19	70.39	65.01	79.47	82.40	54.52	36.40	67.63
- QLoRA F4	90.77B	3.31	85.20	67.87	64.46	79.08	82.80	54.43	35.80	67.09
- **Moira-1.1**	**75.21B**	**3.20**	85.25	68.23	66.36	79.95	83.15	54.26	36.20	**67.63**
LLaMA 65B	286.63B	3.53	85.30	70.39	64.94	77.19	81.05	52.21	38.40	67.07
- RTN	73.71B	3.67	84.60	68.95	64.67	77.26	81.40	51.27	40.40	66.94
- GPTQ	73.34B	3.65	84.80	70.39	64.30	77.19	81.35	53.32	39.40	67.25
- AWQ	76.57B	3.61	85.56	70.75	64.91	77.74	80.51	51.62	38.40	67.07
- QLoRA F4	86.37B	3.75	84.95	72.92	64.41	76.55	81.00	51.87	38.40	67.16
- **Moira-1.1**	**71.22B**	**3.61**	85.25	70.75	65.03	77.42	81.50	52.90	40.00	**67.55**

Table 2: Comparison of quantized weight configurations for hardwired constant MAC design. Moira is compared against RTN, GPTQ, and AWQ for INT4 with a 128-group size, and against F4 in QLoRA. Models are calibrated on C4 and tested on WikiText-2 and zero-shot tasks. The wire count $\mathcal{N}$ for the quantization methods is calculated from the scales and values unbiased by zero points.

INT4-based methods and Moira concentrate on highly expressive bases. The difference is that Moira uses fewer wires by allocating them more dispersed across the dominant bases. This indicates that adaptive base-level scoring is more effective than strict quantization rules for selecting useful bases.

5.3 Deep Analysis

Effects of varying target wire count. Figure 5 shows how, as N_{target} decreases, the total wire count relative to the factorized vanilla model decreases, how the remaining wires concentrate on fewer bases, and how model performance changes accordingly. We observe that Moira remains nearly lossless with negligible degradation as long as about two bases per weight are selected on average, mostly within the range from 2^{-11} to 2^{-5}. As N_{target} approaches 1, the selected wires concentrate around a few dominant bases near 2^{-8} and 2^{-5}, and performance degradation becomes visible. Even in this low-wire regime, the degradation decreases as model size increases, suggesting that larger models may admit simpler, hardwired circuit realizations with less performance loss.

Convergence of Moira. Figure 6(a) shows the convergence behavior of Moira. Because LLMs contain many linear layers (QKV-O and Gate, Up, and Down projection layers), we report block-wise

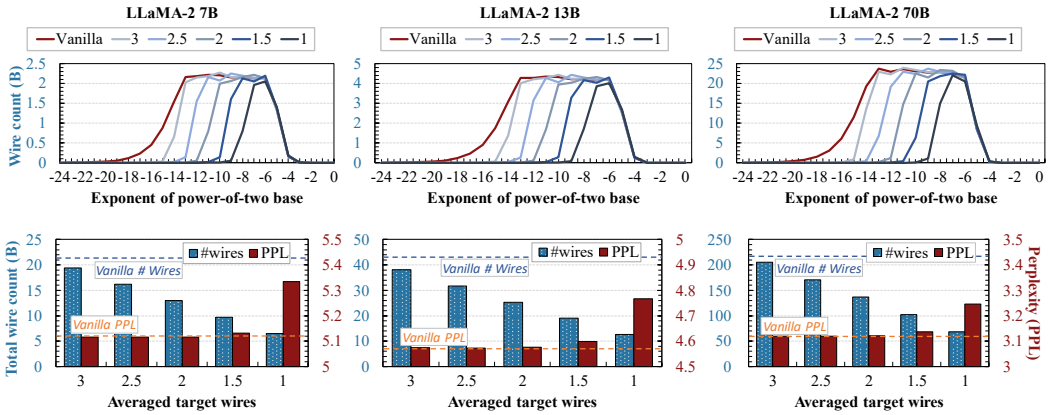
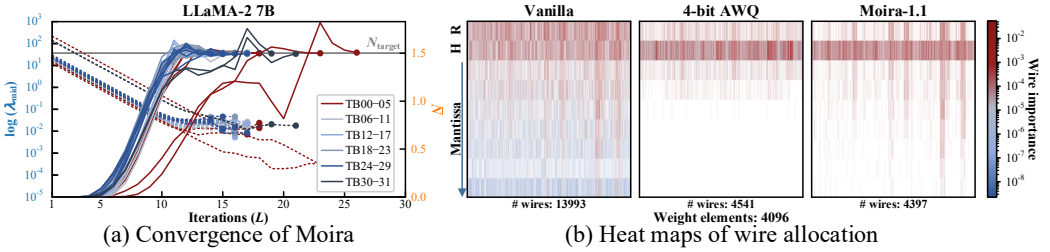


Figure 5: Effects of varying the averaged target wire count in Moira: **(top)** wire-frequency distributions over N_{target}, starting from the vanilla LLaMA-2 model (FP16); **(bottom)** total wire count and performance versus N_{target}. The dotted lines denote the wire count and PPL of the vanilla model.



(a) Convergence of Moira (b) Heat maps of wire allocation

Figure 6: Convergence and wire allocation analysis using LLaMA-2 7B: **(a)** λ_{mid} (left y-axis, log scale) and the induced average wire count $\bar{N}$ (right y-axis); **(b)** Results for the first output channel of the output projection layer of the first Transformer Block (TB). Each color denotes the magnitude of the wire's importance. The x-axis denotes the channel dimension, and one segment of the y-axis is determined by the mantissa bits with the hidden (H) and rounding (R) bits in a precision format.

averages of layer-level convergence statistics. Ignoring early convergence at individual layers, we compute the sample mean within each Transformer block. Except for the early blocks (00–05) and the final blocks (30–31), most blocks reach $\bar{N} \simeq N_{\text{target}} = 1.5$ within about 18 iterations. The slower convergence in the early and final blocks suggests outlier weights or activations that enlarge the initial value and search range of λ_{high}. Even so, $\bar{N}$ converges within the expected iteration budget $L = 30$. Appendix D provides additional plots supporting the practicality of the 30-step budget.

Visual comparison with quantization for hardwired realization. Figure 6(b) compares Moira with AWQ at the wiring level. Red and blue segments indicate wired bases, whereas white segments indicate unwired ones. Each color indicates the wire's importance, with redder segments indicating higher importance and bluer segments indicating lower importance. Note that the score of each wire is defined as its base magnitude multiplied by activation. Compared with vanilla, AWQ applies strict rounding, which can remove important wires from the representation. By contrast, we confirm that Moira scores and selects individual wires, allowing it to retain more important bases. Accordingly, Moira achieves similar or better performance with fewer wires than the state of the art.

5.4 Physical Validation of Wiring Cost

The wiring cost defined in Eq. (13) is intended as an analytical proxy for physical area. Yet, whether this metric $\mathcal{N}$ faithfully reflects the actual synthesized area remains to be validated. To ensure this connection, we realize Moira and all pruning and quantization baselines in RTL and evaluate their post-synthesis PPA (area, power, and latency) under the same operating conditions. Here, full-LLM RTL realization is impractical. Hence, we target and evaluate the same channel analyzed in Figure 6.

Hardware implementation. For a fair comparison, all methods use bit-serial input multiplication. Processing input bits in parallel could reduce latency at the cost of increased multiplier area. We

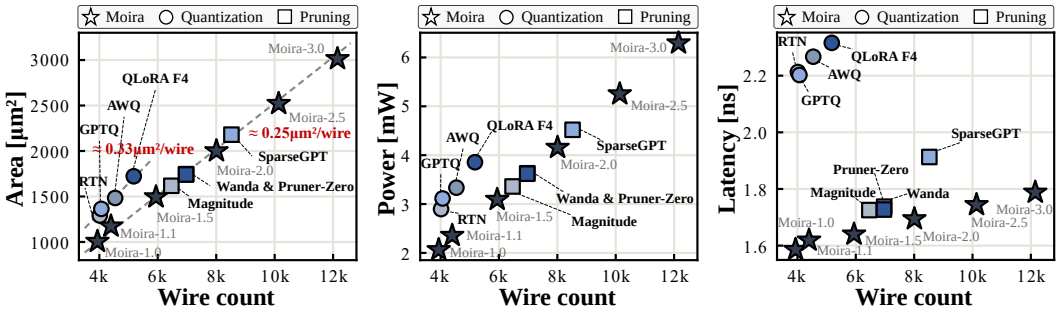


Figure 7: PPA versus wire count for the 1×4096 GEMV operation of the same output channel analyzed in Figure 6: **(left)** synthesized area, **(middle)** power consumption, and **(right)** latency.

instead prioritize minimal area, consistent with the area-minimizing goal of hardwiring. For the quantization baselines, all with a group size of 128, we adopt an area-efficient dataflow that first computes the dot product of 128 quantized weights and activations, then multiplies by a single scaling factor. Moira and pruning require neither scaling factors nor group-wise operations and thus follow the same hardware design methodology.

RTL Synthesis. We set a single target frequency for all designs so that PPA is compared under identical timing constraints. The operating frequency is chosen such that even the slowest variant meets timing with positive slack, without invoking area-inflating timing optimizations by ensuring the reported area reflects our method rather than synthesis timing effort. All designs are synthesized in `Samsung 4nm LPP` at `350MHz` under the worst-case corner (`SS, 0.675V, 125°C`).

Wiring cost as a physical proxy. Figure 7 (left) demonstrates that, across Moira (average wires 1.0–3.0) and all pruning methods, area scales linearly with wire count (slope ≈ 0.25 μm^2/wire). This strong correspondence shows that the wire count reflects the synthesized hardware area. It is worth noting that delay increases gently with wire count, as shown in Figure 7 (right), because the hardwired accumulation is organized as a parallel adder tree. Increasing the number of wires primarily widens the accumulation tree rather than introducing sequential accumulation stages along the critical path.

Wiring cost as a lower bound. As shown in Figure 7 (left), for the quantization baselines, area increases more rapidly with wire count (slope ≈ 0.33 μm^2/wire), placing them consistently above the Moira/pruning trend. In fact, their actual hardware cost exceeds what their wire count predicts. For the INT4 quantization baselines, scaling-factor multiplication adds an area cost not fully captured by the wire count. After summing 128 INT4 products through the adder tree, the result's bit width increases, and multiplying this widened value by the scaling factor requires a correspondingly large shifter/adder. This added stage also lengthens the critical path and increases delay beyond what the wire count alone suggests. For instance, GPTQ uses fewer wires than Moira-1.1 (4,068 vs. 4,397), yet exhibits larger area (1,363.49 μm^2 vs. 1,173.67 μm^2), higher power (3.1144 mW vs. 2.3551 mW), and longer delay (2.2024 ns vs. 1.6189 ns).

Generalization. These PPA results in Figure 7 demonstrate that the wire count defined by Eq. (13) is a faithful yet conservative proxy for physical area across methods. Since linear layers share the same computational structure despite differing dimensions, the observed relation between wire count and area can be projected across the linear layers of full LLMs.

6 Conclusion

This paper argues that the path to mortal computing for large language models lies not in treating weights as indivisible parameters, but in rebuilding them in the interconnect space that hardware actually executes. At that level, the goal is no longer merely to remove weights or reduce bit-widths, but to decide which shiftable bases should remain physically wired. This change in perspective makes explicit that the real challenge lies in controlling the rapid growth of hardwired connections. Our results show that this challenge is met by scoring wires directly. Moira remains simple, preserves model performance, and points to a new regime of LLM compression whose behavior reflects both pruning and quantization while belonging fully to neither. We believe this viewpoint about wire-level optimization may become a central principle for constructing future physical circuits for LLMs.

Acknowledgments and Disclosure of Funding

We thank the reviewers, Area Chairs, Senior Area Chairs, and Program Chairs for their time, and deeply appreciate their thoughtful insights that helped improve this work.

References

[1] Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nvidia nemotron 3: Efficient and open intelligence. *arXiv preprint arXiv:2512.20856*, 2025.

[2] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *AI open*, 2024.

[5] Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[6] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *Proceedings of International Conference on Machine Learning (ICML)*, 2023.

[7] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[8] Geoffrey Hinton. Two paths to intelligence. In *Public Lecture, University of Cambridge*, 2023.

[9] Yang Liu, Yi Chen, Yongwei Zhao, Yifan Hao, Zifu Zheng, Weihao Kong, Zhangmai Li, Dongchen Jiang, Ruiyang Xia, Zhihong Ma, et al. Hardwired-neuron language processing units as general-purpose cognitive substrates. In *Proceedings of the 31st ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2026.

[10] Haoran You, Yipin Guo, Yichao Fu, Wei Zhou, Huihong Shi, Xiaofan Zhang, Souvik Kundu, Amir Yazdanbakhsh, and Yingyan Celine Lin. Shiftaddllm: Accelerating pretrained llms via post-training multiplication-less reparameterization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[11] Peijie Dong, Lujun Li, Zhenheng Tang, Xiang Liu, Xinglin Pan, Qiang Wang, and Xiaowen Chu. Pruner-zero: Evolving symbolic pruning metric from scratch for large language models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2024.

[12] Lian Liu, Xiandong Zhao, Guanchen Li, Dong Li, Mengdi Wang, Yinhe Han, Xiaowei Li, and Ying Wang. Bawa: Automatic optimizing pruning metric for large language models with balanced weight and activation. In *Proceedings of International Conference on Machine Learning (ICML)*, 2025.

[13] Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[14] Gongfan Fang, Hongxu Yin, Saurav Muralidharan, Greg Heinrich, Jeff Pool, Jan Kautz, Pavlo Molchanov, and Xinchao Wang. Maskllm: Learnable semi-structured sparsity for large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[15] John Kloosterman, Jonathan Beaumont, Mick Wollman, Ankit Sethia, Ron Dreslinski, Trevor Mudge, and Scott Mahlke. Warppool: Sharing requests with inter-warp coalescing for throughput processors. In *Proceedings of the 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2015.

[16] Cheonjun Park, Mincheol Park, Hyun Jae Oh, Minkyu Kim, Myung Kuk Yoon, Suhyun Kim, and Won Woo Ro. Balanced column-wise block pruning for maximizing gpu parallelism. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[17] Maohua Zhu, Tao Zhang, Zhenyu Gu, and Yuan Xie. Sparse tensor core: Algorithm and hardware co-design for vector-wise sparse neural networks on modern gpus. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019.

[18] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[19] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

[20] Shaobo Ma, Chao Fang, Haikuo Shao, and Zhongfeng Wang. Apt-llm: Exploiting arbitrary-precision tensor core computing for llm acceleration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2025.

[21] Gunho Park, Baeseong park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. Lut-gemm: Quantized matrix multiplication based on LUTs for efficient inference in large-scale generative language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[22] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[23] Felix Juefei-Xu, Vishnu Naresh Boddeti, and Marios Savvides. Local binary convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[24] Peng Chen, Bohan Zhuang, and Chunhua Shen. Fatnn: Fast and accurate ternary neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[25] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. Deepshift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[27] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[28] Yixing Xu, Chang Xu, Xinghao Chen, Wei Zhang, Chunjing Xu, and Yunhe Wang. Kernel based progressive distillation for adder neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[29] Yihui He, Xianggen Liu, Huasong Zhong, and Yuchun Ma. Addressnet: Shift-based primitives for efficient convolutional neural networks. In *Proceedings of the 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[30] Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[31] Haoran You, Huihong Shi, Yipin Guo, and Yingyan Lin. Shiftaddvit: Mixture of multiplication primitives towards efficient vision transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[32] Hongyu Wang, Shuming Ma, Lingxiao Ma, Lei Wang, Wenhui Wang, Li Dong, Shaohan Huang, Huaijie Wang, Jilong Xue, Ruiping Wang, Yi Wu, and Furu Wei. Bitnet: 1-bit pre-training for large language models. *Journal of Machine Learning Research*, 2025.

[33] Han Shu, Jiahao Wang, Hanting Chen, Lin Li, Yujiu Yang, and Yunhe Wang. Adder attention for vision transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[34] Guangting Wang, Yucheng Zhao, Chuanxin Tang, Chong Luo, and Wenjun Zeng. When shift operation meets vision transformer: An extremely simple alternative to attention mechanism. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[35] Albert Tseng, Tao Yu, and Youngsuk Park. Training llms with mxfp4. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025.

[36] Haoran You, Xiaohan Chen, Yongan Zhang, Chaojian Li, Sicheng Li, Zihao Liu, Zhangyang Wang, and Yingyan Lin. Shiftaddnet: A hardware-inspired deep network. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[37] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[38] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

[39] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, et al. A framework for few-shot language model evaluation. *Zenodo*, 2021.

[40] Hugo Touvron, Louis Martin, Kevin R Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Niko-lay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[41] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[42] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

[43] Mincheol Park, Dongjin Kim, Cheonjun Park, Yuna Park, Gyeong Eun Gong, Won Woo Ro, and Suhyun Kim. Reprune: channel pruning via kernel representative selection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

[44] Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[45] Qwen Team. Qwen3.5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

[46] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[47] Hiroyuki Ochi, Kosei Yamaguchi, Tetsuaki Fujimoto, Junshi Hotate, Takashi Kishimoto, Toshiki Higashi, Takashi Imagawa, Ryutaro Doi, Munehiro Tada, Tadahiko Sugibayashi, et al. Via-switch fpga: Highly dense mixed-grained reconfigurable architecture with overlay via-switch crossbars. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2018.

A Proofs

A.1 Proof of Proposition 1

Proof. If multiple input channels share the same weight value, the distributive property in Definition 1 lets us aggregate their contributions before applying the corresponding hardwired operation.

For a fixed output channel j, consider the linear contribution in a hardwired MAC operation:

$$y_{ij} = \sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \left[\otimes w_{kj}\right]. \tag{14}$$

Let $\mathcal{U}_j$ denote the set of unique weight values appearing in the j-th output channel, formally defined as $\mathcal{U}_j = \{u \in \mathbb{R} \mid \exists k,\ w_{kj} = u\}$. For each $u \in \mathcal{U}_j$, define the corresponding index set $\mathcal{K}_j(u) = \{k \in \{1, \ldots, C_{\text{in}}\} \mid w_{kj} = u\}$. Then Eq. (14) can be equivalently written as:

$$\begin{aligned} y_{ij} &= \sum\nolimits_{u \in \mathcal{U}_j} \left(\sum\nolimits_{k' \in \mathcal{K}_j(u)} x_{ik'} \left[\otimes w_{k'j}\right] \right) \\ &= \sum\nolimits_{u \in \mathcal{U}_j} \underbrace{\left(\sum\nolimits_{k' \in \mathcal{K}_j(u)} x_{ik'} \left[\otimes u\right] \right)}_{\text{For example, } = x_{ik_1}[\otimes u] + x_{ik_2}[\otimes u] + \cdots + x_{ik_n}[\otimes u]} \\ &= \sum\nolimits_{u \in \mathcal{U}_j} \left(\sum\nolimits_{k' \in \mathcal{K}_j(u)} x_{ik'} \right) \left[\otimes u\right], \end{aligned} \tag{15}$$

Thus, for each repeated weight value, the corresponding inputs can be pre-aggregated, and the shared hardwired operation associated with that value is applied once to the aggregated input.

This completes the proof of Proposition 1. □

A.2 Proof of Proposition 2

Proof. For a fixed output channel j, suppose that each weight w_{kj} has already been written as a finite signed combination of power-of-two bases following Eq. (6):

$$w_{kj} = \sum\nolimits_{e \in \mathcal{E}_{w_{kj}}} \tau_{w_{kj}}(e) \cdot 2^e, \quad \tau_{w_{kj}}(e) \in \{-1, 0, 1\}. \tag{16}$$

Substituting this decomposition into the hardwired MAC expression gives

$$\begin{aligned} y_{ij} &= \sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \left[\otimes w_{kj}\right] \\ &= \sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \left[\otimes \sum\nolimits_{e \in \mathcal{E}_{w_{kj}}} \tau_{w_{kj}}(e) \cdot 2^e \right] \\ &= \sum\nolimits_{k=1}^{C_{\text{in}}} \sum\nolimits_{e \in \mathcal{E}_{w_{kj}}} \tau_{w_{kj}}(e) x_{ik} \left[\otimes 2^e\right]. \end{aligned} \tag{17}$$

At this stage, the summation is still organized by the original weight index k. We now reorganize it by shared exponent bases. Define the union of all exponent sets appearing in the output channel j as $E_j = \bigcup_{k=1}^{C_{\text{in}}} \mathcal{E}_{w_{kj}}$. For notational convenience, extend $\tau_{w_{kj}}(e)$ by setting $\tau_{w_{kj}}(e) = 0$ for $e \notin \mathcal{E}_{w_{kj}}$. Then Eq. (17) becomes

$$\begin{aligned} y_{ij} &= \sum\nolimits_{k=1}^{C_{\text{in}}} \sum\nolimits_{e \in E_j} \tau_{w_{kj}}(e) x_{ik} \left[\otimes 2^e\right] \\ &= \sum\nolimits_{e \in E_j} \left(\sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \cdot \tau_{w_{kj}}(e) \left[\otimes 2^e\right] \right) \end{aligned} \tag{18}$$

Since the sums are finite, the order of summation can be exchanged:

$$\begin{aligned} y_{ij} &= \sum\nolimits_{e \in E_j} \underbrace{\left(\sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \cdot \tau_{w_{kj}}(e) \left[\otimes 2^e\right] \right)}_{\text{For example, } = x_{ik_1}[\otimes 2^e] - x_{ik_2}[\otimes 2^e] - x_{ik_5}[\otimes 2^e] + \cdots + x_{ik_n}[\otimes 2^e]} \\ &= \sum\nolimits_{e \in E_j} \left(\sum\nolimits_{k=1}^{C_{\text{in}}} x_{ik} \cdot \tau_{w_{kj}}(e) \right) \left[\otimes 2^e\right]. \end{aligned} \tag{19}$$

Thus, once weights are factorized into signed power-of-two bases, the computation can be reorganized base-wise: first aggregate all signed input contributions with respect to the same exponent base, then apply each shared base operation once per output channel.

This completes the proof of Proposition 2. □

B Analysis of the Lagrangian-Guided Search in Moira

We analyze why the search procedures in Algorithms 1 and 2 are aligned with Lagrangian relaxation. The goal is not to claim optimality for the original discrete constrained problem, but to clarify that the proposed nested procedure is not ad hoc. Specifically, we show that: (1) the inner update follows a threshold structure induced by the Lagrangian objective, (2) the outer loop performs binary search over a monotone budget map with geometric contraction in the dual variable, and (3) Moira can achieve ε-accurate budget matching whenever an attainable feasible plateau exists.

We begin with the following constrained optimization problem:

$$\min_{N_{kj}} \sum_{k=1,j=1}^{C_{\text{in}},C_{\text{out}}} |w_{kj} - \tilde{w}_{kj}^{N_{kj}}| \cdot \tilde{\mathbf{X}}_k, \quad \text{s.t.} \quad \frac{1}{C_{\text{in}}C_{\text{out}}} \sum_{k=1,j=1}^{C_{\text{in}},C_{\text{out}}} N_{kj} \le N_{\text{target}}, \tag{20}$$

where, for each weight w_{kj}, $\tilde{w}_{kj}^{N}$ denotes the reconstruction obtained after selecting N bases. Using a single Lagrange multiplier, we relax the average wire-count constraint as

$$\begin{aligned} \mathcal{J} &= \sum_{k=1,j=1}^{C_{\text{in}},C_{\text{out}}} |w_{kj} - \tilde{w}_{kj}^{N_{kj}}| \cdot \tilde{\mathbf{X}}_k + \tilde{\lambda} \left(\frac{1}{C_{\text{in}}C_{\text{out}}} \sum_{k=1,j=1}^{C_{\text{in}},C_{\text{out}}} N_{kj} - N_{\text{target}} \right) \\ &= \sum_{k=1,j=1}^{C_{\text{in}},C_{\text{out}}} \left(|w_{kj} - \tilde{w}_{kj}^{N_{kj}}| \cdot \tilde{\mathbf{X}}_k + \lambda N_{kj} \right) - \underbrace{\tilde{\lambda} N_{\text{target}}}_{\text{Constant}}, \end{aligned} \tag{21}$$

where $\lambda = \tilde{\lambda}/(C_{\text{in}}C_{\text{out}})$. Since the last term is constant with respect to N_{kj}, we can drop it when determining the minimizer. This yields the following per-weight Lagrangian objective:

$$\mathcal{J}_{kj}(N;\lambda) = |w_{kj} - \tilde{w}_{kj}^{N}| \cdot \tilde{\mathbf{X}}_k + \lambda N. \tag{22}$$

The first term in Eq. (22) favors accurate reconstruction, whereas the second term penalizes further wiring. Thus, for fixed λ, the selected factorization depth is determined by balancing residual reduction against wire cost.

We now define the activation-weighted residual and the corresponding marginal residual reduction as

$$\mathcal{R}_{kj}(N) := |w_{kj} - \tilde{w}_{kj}^{N}| \cdot \tilde{\mathbf{X}}_k, \quad \Delta_{kj}(N) := \mathcal{R}_{kj}(N) - \mathcal{R}_{kj}(N+1). \tag{23}$$

Since the reconstruction at depth $N+1$ can include the depth-N solution as a special case, adding one more base cannot worsen the best reconstruction at the next depth. Therefore, $\Delta_{kj}(N) \ge 0$. Comparing two consecutive objective values gives

$$\begin{aligned} \mathcal{J}_{kj}(N+1;\lambda) - \mathcal{J}_{kj}(N;\lambda) &= (\mathcal{R}_{kj}(N+1) - \mathcal{R}_{kj}(N)) + \lambda \\ &= -\Delta_{kj}(N) + \lambda. \end{aligned} \tag{24}$$

Therefore,

$$\mathcal{J}_{kj}(N+1;\lambda) \le \mathcal{J}_{kj}(N;\lambda) \quad \Longleftrightarrow \quad \Delta_{kj}(N) \ge \lambda. \tag{25}$$

Eq. (25) is the key structural property of the inner optimization. It shows that one more base is beneficial exactly when its marginal residual reduction is at least as large as the current wire price. In this sense, Moira's inner loop can be interpreted as a *thresholded* Lagrangian update.

To characterize the selected depth more explicitly, we use the nested ordering induced by Moira's reconstruction procedure. Moira increases the factorization depth by adding one additional base to the reconstruction. Under this nested ordering, we assume that the marginal residual reductions are non-increasing with respect to depth:

$$\Delta_{kj}(0) \ge \Delta_{kj}(1) \ge \Delta_{kj}(2) \ge \cdots. \tag{26}$$

This diminishing-return assumption is consistent with the greedy ordering used by Moira, where bases are added according to their contribution to reducing the activation-weighted reconstruction error.

Under this nested and diminishing-return structure, the selected factorization depth for fixed λ is characterized by the first index at which the marginal reduction falls below the wire price:

$$N_{kj}(\lambda) = \min\{N \mid \Delta_{kj}(N) < \lambda\}. \tag{27}$$

Equivalently, it is the largest depth such that all preceding marginal reductions still pass the threshold:

$$N_{kj}(\lambda) = \max\{N \mid \Delta_{kj}(0) \ge \lambda, \ldots, \Delta_{kj}(N-1) \ge \lambda\}. \tag{28}$$

Hence, λ acts as a global cutoff shared across all weights, while each weight responds to it through its own sequence of marginal reductions.

The next property is *monotonicity*. Let $\lambda_1 < \lambda_2$. If a certain depth increment satisfies $\Delta_{kj}(N) \ge \lambda_2$, then it necessarily also satisfies $\Delta_{kj}(N) \ge \lambda_1$. Thus, the set of admissible increments under λ_2 is contained in that under λ_1, which implies

$$N_{kj}(\lambda_1) \ge N_{kj}(\lambda_2). \tag{29}$$

Therefore, the selected depth is non-increasing in the wire price. Averaging over all weights yields

$$\bar{N}(\lambda) = \frac{1}{\mathcal{C}} \sum\nolimits_{k=1,j=1}^{C_{\text{in}}C_{\text{out}}} N_{kj}(\lambda), \quad \mathcal{C} = C_{\text{in}}C_{\text{out}}. \tag{30}$$

From the monotonicity of each $N_{kj}(\lambda)$, it follows that

$$\lambda_1 < \lambda_2 \quad \Longrightarrow \quad \bar{N}(\lambda_1) \geq \bar{N}(\lambda_2). \tag{31}$$

In other words, the induced average wire count is a monotone non-increasing function of λ.

This monotonicity makes the outer loop of Moira theoretically well-posed. The outer loop does not search over an arbitrary heuristic quantity. Instead, it binary searches over the monotone map $\lambda \mapsto \bar{N}(\lambda)$, aiming to match the prescribed target budget N_{target}.

Let $I^l = [\lambda^l_{\text{low}}, \lambda^l_{\text{high}}]$ denote the maintained search interval at iteration l, and define the midpoint as

$$\lambda^l_{\text{mid}} = \frac{\lambda^l_{\text{low}} + \lambda^l_{\text{high}}}{2}. \tag{32}$$

Moira's update rule replaces one endpoint with the midpoint, depending on whether $\bar{N}(\lambda^l_{\text{mid}})$ is above or below the target. Since only one endpoint is replaced by the midpoint at each iteration, the length of the search interval contracts by a factor of two:

$$\left|\lambda^{l+1}_{\text{high}} - \lambda^{l+1}_{\text{low}}\right| = \frac{1}{2}\left|\lambda^l_{\text{high}} - \lambda^l_{\text{low}}\right|, \quad \text{and hence} \quad \left|\lambda^l_{\text{high}} - \lambda^l_{\text{low}}\right| = 2^{-l}\left|\lambda^0_{\text{high}} - \lambda^0_{\text{low}}\right|. \tag{33}$$

This is the standard geometric *contraction* of binary search. In particular, the search over λ tightens exponentially with the number of iterations.

A subtle but important point is that this contraction occurs in the dual variable λ, not directly in the induced average wire count $\bar{N}(\lambda)$. Eq. (33) guarantees that the uncertainty in λ-space contracts geometrically. However, since each $N_{kj}(\lambda)$ is integer-valued, the induced budget map $\bar{N}(\lambda)$ is generally a monotone step function. Therefore, contraction in λ-space does not necessarily imply exact equality $\bar{N}(\lambda) = N_{\text{target}}$. Even when the interval in λ becomes very small, the corresponding average wire count may remain slightly above or below the target due to the discrete nature of the selected depths.

For this reason, Moira uses ε-accurate budget matching rather than requiring exact equality. Let $\varepsilon > 0$, and suppose there exists a nonzero-width interval

$$I_\varepsilon = [a, b] \subseteq I^0, \quad a < b, \quad \text{s.t.} \quad \lambda \in I_\varepsilon \Longrightarrow |\bar{N}(\lambda) - N_{\text{target}}| \leq \varepsilon. \tag{34}$$

We call such an interval an ε-feasible plateau. This means that any λ in this interval induces an average wire count within ε of the target. Since the binary-search interval contracts by a factor of two at every iteration, the current search interval becomes narrow enough to search at the scale of this feasible plateau after a sufficient number of iterations. In particular, if

$$l \geq \left\lceil \log_2 \frac{\lambda^0_{\text{high}} - \lambda^0_{\text{low}}}{b - a} \right\rceil, \tag{35}$$

then the current search interval satisfies

$$\lambda^l_{\text{high}} - \lambda^l_{\text{low}} \leq b - a. \tag{36}$$

Therefore, as long as the feasible plateau has not been excluded from the current search interval, the search has sufficient resolution to identify a λ satisfying

$$|\bar{N}(\lambda) - N_{\text{target}}| \leq \varepsilon. \tag{37}$$

The existence of such an ε-feasible plateau is plausible in linear layers of LLMs in practice. Although each $N_{kj}(\lambda)$ is integer-valued, the reported budget is averaged over $\mathcal{C} = C_{\text{in}}C_{\text{out}}$ weights. Hence, a single depth change contributes only $1/\mathcal{C}$ to $\bar{N}(\lambda)$. For modern LLM layers, $\mathcal{C}$ is extremely large, making the effective granularity of the average budget very fine. This does not guarantee exact target equality in general, but it makes small-tolerance ε-accurate matching practically attainable. This motivates the experimental stopping criterion $\varepsilon = 0.01$ used in Appendix C, which provides a tight practical tolerance for average wire-count matching while avoiding unnecessary iterations caused by the stepwise nature of $\bar{N}(\lambda)$.

Overall, Algorithms 1 and 2 in Moira can be understood as follows. The inner loop is a thresholded Lagrangian update in which each additional base is accepted only when its marginal residual reduction is at least as large as the shared wire price. Under the nested ordering and non-increasing marginal reduction assumption, this update characterizes the selected factorization depth for a fixed λ. This induces a monotone non-increasing budget map $\bar{N}(\lambda)$, and the outer loop exploits this monotonicity through geometric binary-search contraction in the dual variable. Exact target equality is not always guaranteed because the induced budget map is generally stepwise. However, whenever an attainable ε-feasible plateau exists, the procedure can achieve ε-accurate budget matching after enough iterations. In this sense, Moira's outer binary search is not a heuristic search, but a Lagrangian-induced threshold search over a monotone budget map.

Method	# Wires ($\mathcal{N}$)	Wikitext PPL	BoolQ.	RTE	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	AVG.
Mistral 7B	23.92B	4.95	83.66	68.23	61.15	74.03	79.96	50.76	35.20	64.71
- Magnitude	11.94B	7.28	69.84	63.53	57.28	67.48	73.44	42.83	32.00	58.06
- SparseGPT	15.49B	6.01	82.38	59.20	56.13	73.40	76.43	44.28	32.20	60.57
- Wanda	12.00B	6.07	82.78	58.12	55.24	72.21	75.84	42.74	30.80	59.68
- Pruner-Zero	11.99B	5.97	81.83	58.84	55.69	71.58	76.01	42.40	29.60	59.42
- **Moira-1.5**	**10.47B**	**4.96**	83.60	71.48	60.98	73.95	79.75	50.25	35.60	**65.09**
- RTN	**7.63B**	5.16	82.62	63.17	60.50	72.92	79.29	48.29	35.20	63.14
- GPTQ	7.79B	5.12	82.78	63.53	59.98	74.42	79.29	48.20	36.40	63.51
- AWQ	8.10B	5.06	83.05	68.23	60.61	74.11	79.50	50.00	36.00	64.50
- QLoRA F4	9.18B	5.19	82.01	63.17	60.14	73.32	78.82	48.12	33.40	62.71
- **Moira-1.1**	7.67B	**5.02**	83.63	71.48	60.61	74.26	79.54	50.68	35.80	**65.14**

Table 3: Comparison of Moira with pruning and quantization on Mistral-7B. Pruning baselines use 50% unstructured sparsity, while quantization baselines use INT4 with a group size of 128 or FP4. Models are calibrated on C4 and evaluated on WikiText-2 and zero-shot tasks. Wire count uses non-zero weights for pruning and scales and zero-point-corrected values for quantization.

Method	# Wires ($\mathcal{N}$)	Wikitext PPL	BoolQ.	RTE	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	AVG.
Qwen3.5 27B	83.52B	6.71	76.91	73.64	63.78	78.92	84.93	59.89	35.00	67.58
- **Moira-1.5**	**36.53B**	**6.71**	78.50	72.92	63.69	79.32	84.89	59.81	35.00	**67.73**
- **Moira-1.1**	**26.77B**	**6.76**	77.73	74.00	63.53	78.76	84.63	60.40	36.00	**67.87**

Table 4: Comparison of Moira and the vanilla model on Qwen3.5-27B under the same protocol.

C Experimental Settings

To maximize reproducibility and ensure a fair comparison, we depend on publicly available implementations centered on `HuggingFace`. For LLaMA models, we use the `huggyllama` checkpoints, while for LLaMA-2 models, we use the `meta-llama` checkpoints available on `HuggingFace`. For pruning baselines, Magnitude, SparseGPT, and Wanda are all evaluated using the official Wanda repository[1], while Pruner-Zero is evaluated using its official repository[2]. All pruning methods share the same C4 calibration protocol, following Pruner-Zero, and use the `HuggingFace datasets` library. Perplexity is measured following the NVIDIA MaskLLM[3] evaluation protocol. For quantization baselines, GPTQ, AWQ, and F4 are run using `HuggingFace GPTQConfig`-based, `AwqConfig`-based, and `BitsAndBytesConfig`-based tools with 4-bit quantization and group size 128, while RTN is implemented following the AWQ repository[4]. WikiText-2 is loaded through the `HuggingFace datasets` library. Zero-shot evaluation is conducted with `LM-Eval-Harness` using the `transformers AutoModelForCausalLM` class under data-parallel execution with batch size 8. For Moira, we fix the search depth to $L = 30$ and terminate early once $|\bar{N}(\lambda_{\text{mid}}) - N_{\text{target}}| \leq 0.01$, or after the full iteration budget is exhausted.

Infrastructure setup for pre-sim tests. All experiments are conducted on a single NVIDIA DGX H100 node. Specifically, Mistral 7B and LLaMA-2 7B and 13B are evaluated on a single GPU, whereas larger models are evaluated across multiple GPUs due to their parameter footprints: LLaMA 30B across 4 GPUs, and LLaMA 65B, LLaMA-2 70B, and Qwen3.5 27B across 8 GPUs. The Moira search itself is performed on a single GPU.

D Supplementary Results

Performance on other LLMs. We further evaluate Moira on Mistral-7B [44] and Qwen3.5-27B [45]. On Mistral-7B, Moira consistently outperforms pruning and quantization baselines at comparable or lower wire counts. At $N_{\text{target}} = 1.5$, Moira achieves an average zero-shot accuracy of 65.09 with fewer wires than all pruning baselines, while Moira-1.1 achieves a perplexity of 5.02 and an average zero-shot accuracy of 65.14, outperforming all quantization baselines. For Qwen3.5-27B, baseline implementations are not compatible with its newer `transformers` environment, so we compare Moira with the vanilla model across wire budgets. At $N_{\text{target}} = 1.5$, Moira preserves the vanilla perplexity of 6.71 while slightly improving average zero-shot accuracy from 67.58 to 67.73. Even at $N_{\text{target}} = 1.1$, perplexity increases by only 0.05 to 6.76. These results demonstrate that Moira maintains its near-lossless behavior across different model families and scales.

Effects of $\tilde{\mathbf{X}}_k$ construction on Moira. Table 5 compares Moira when $\tilde{\mathbf{X}}_k$ is computed from C4 or from the dummy prompt, "*Artificial intelligence and machine learning are transforming the future of technology and society.*" At $N_{\text{target}} = 1.5$, both choices yield nearly identical wire counts and perplexity, while average

[1] https://github.com/locuslab/wanda

[2] https://github.com/pprp/Pruner-Zero

[3] https://github.com/nvlabs/maskllm

[4] https://github.com/mit-han-lab/llm-awq

Method	# Wires ($\mathcal{N}$)	Wikitext PPL	BoolQ.	RTE	HellaS.	WinoG.	ARC-E	ARC-C	OBQA	AVG.
LLaMA-2 7B	22.20B	5.12	79.35	63.53	57.10	69.45	75.42	43.00	33.20	60.15
- Moira-1.5 (D)	9.70B	5.13	78.96	63.89	56.98	69.53	75.92	42.23	32.60	60.01
- Moira-1.5	9.70B	5.13	79.54	60.64	57.09	69.37	76.17	43.25	33.60	59.95
- Moira-1.0 (D)	6.47B	5.35	78.56	63.89	55.41	68.03	74.70	40.95	32.00	59.08
- Moira-1.0	6.47B	5.33	77.92	63.17	56.82	69.21	75.63	43.43	32.80	59.85
LLaMA-2 13B	43.50B	4.57	82.14	67.14	60.18	72.53	78.95	47.35	34.60	63.27
- Moira-1.5 (D)	19.02B	4.59	81.80	66.06	59.93	71.90	78.45	46.58	34.60	62.76
- Moira-1.5	19.03B	4.59	81.80	66.06	60.18	72.21	78.99	47.61	35.60	63.21
- Moira-1.0 (D)	12.68B	4.77	81.59	66.06	59.21	72.37	78.49	45.81	34.20	62.53
- Moira-1.0	12.68B	4.76	81.40	64.25	59.82	72.21	78.66	48.03	34.60	62.71
LLaMA 30B	142.09B	4.10	83.70	68.23	63.36	75.76	80.75	51.53	36.40	65.67
- Moira-1.5 (D)	48.15B	4.12	83.66	67.87	63.14	76.55	81.01	51.87	37.00	65.89
- Moira-1.5	48.14B	4.11	83.66	67.50	63.25	76.63	81.18	51.62	37.20	65.86
- Moira-1.0 (D)	32.10B	4.31	83.51	68.23	62.50	75.21	80.26	49.91	36.20	65.12
- Moira-1.0	32.12B	4.29	83.27	67.50	63.05	75.53	81.22	51.79	36.60	65.56
LLaMA-2 70B	234.74B	3.12	85.40	69.31	65.31	80.42	82.85	53.83	36.20	67.62
- Moira-1.5 (D)	102.62B	3.14	84.86	68.95	65.25	80.58	82.65	53.83	36.40	67.50
- Moira-1.5	102.68B	3.13	85.20	67.87	65.96	80.50	83.00	54.35	36.60	67.64
- Moira-1.0 (D)	68.47B	3.27	84.70	68.23	64.73	79.08	82.74	54.77	36.80	67.29
- Moira-1.0	68.45B	3.24	85.15	67.50	64.63	79.71	83.05	54.60	36.40	67.29
LLaMA 65B	286.63B	3.53	85.30	70.39	64.94	77.19	81.05	52.21	38.40	67.07
- Moira-1.5 (D)	97.15B	3.55	86.05	71.11	64.73	77.34	80.89	52.38	39.60	67.44
- Moira-1.5	97.09B	3.54	85.00	72.39	64.46	77.11	81.00	52.38	38.60	66.99
- Moira-1.0 (D)	64.76B	3.71	85.56	72.20	64.30	76.24	81.18	52.04	38.80	67.19
- Moira-1.0	64.78B	3.69	84.55	72.56	64.48	76.79	81.90	52.64	40.40	67.62

Table 5: Sensitivity of Moira to the construction of $\tilde{\mathbf{X}}_k$. Results of the total wire counts, perplexity, and zero-shot performance are obtained when $\tilde{\mathbf{X}}_k$ is computed from C4 or from a dummy prompt, denoted as 'D', under the wire counts $N_{\text{target}} = \{1.5, 1.0\}$. Gray region indicates results using C4.

zero-shot performance shows no consistent advantage for either calibration source across models. This suggests that Moira is relatively insensitive to the calibration source in a moderately sparse regime. We then examine the extreme case $N_{\text{target}} = 1.0$. Even at this sparsity level, wire count and perplexity remain closely matched, but C4-based calibration produces stronger or equal average zero-shot performance across all models. These results suggest that Moira can tolerate minimal calibration in moderately sparse settings, but benefits from more diverse calibration data when the target wire budget becomes extremely tight.

Additional convergence plots of Moira. Figure 8 shows how closely $\bar{N}$ matches N_{target} within Moira's prescribed iteration budget $L = 30$, while varying the model size from LLaMA-2 7B to 70B and decreasing N_{target} from 3.0 to 1.0. As described in Section 5.3, the early and final blocks consistently converge more slowly than the middle blocks, regardless of model size or target wire count, suggesting outlier weights or activation values. We also observe that larger N_{target} generally requires more iterations, and that increasing model size also slightly increases the required number of iterations. Nevertheless, across all tested settings, $\bar{N}$ approaches N_{target} within the fixed budget of 30 iterations, supporting the practical adequacy of this iteration budget.

Sparsity of reconstructed weights $\tilde{\mathbf{W}}$ across N_{target}. Figure 9 shows that Moira, described in Algorithm 2, induces an implicit pruning effect on the reconstructed weights $\tilde{\mathbf{W}}$ as N_{target} decreases from 3.0 to 1.0. Across all tested LLMs, tightening the wire budget consistently increases weight-level sparsity, reaching about a 20% pruning ratio when the average wire count is reduced to $N_{\text{target}} = 1.0$. Together with the $N_{\text{target}} = 1.0$ results in Table 5 and the pruning baselines in Table 1, this suggests that selectively retaining fine-grained base components better preserves model expressiveness than aggressively pruning entire weights. This highlights a new compression regime below the weight level, where base-level selection can induce sparsity while reframing the conventional pruning trade-off.

Wall-time comparison. Figure 10 compares the wall time of Moira under varying N_{target} with the baselines in Tables 1 and 2. As N_{target} decreases from 3.0 to 1.0, the wall time of Moira decreases in a staircase-like manner, with similar runtimes within the $N_{\text{target}} \in [2, 3)$ regime and again within the $N_{\text{target}} \in [1, 2)$ regime. This is likely because the factorization depth changes in discrete levels, so nearby wire budgets induce similar workloads. Compared with pruning methods, Moira is faster than Hessian-based SparseGPT and gradient-based Pruner-Zero, while remaining comparable to the one-shot method Wanda. This suggests that the nested search in Moira is practically well-contained. Compared with quantization methods, Moira is substantially faster than GPTQ and AWQ for INT4 conversion, likely because these methods optimize iteratively over group-wise units smaller than the channel dimension. By contrast, Magnitude, RTN, and F4 are faster because they rely on simple magnitude comparison or rounding-based operations, although this simplicity does not necessarily translate into equally strong performance. Therefore, Moira achieves a favorable trade-off between runtime and performance.

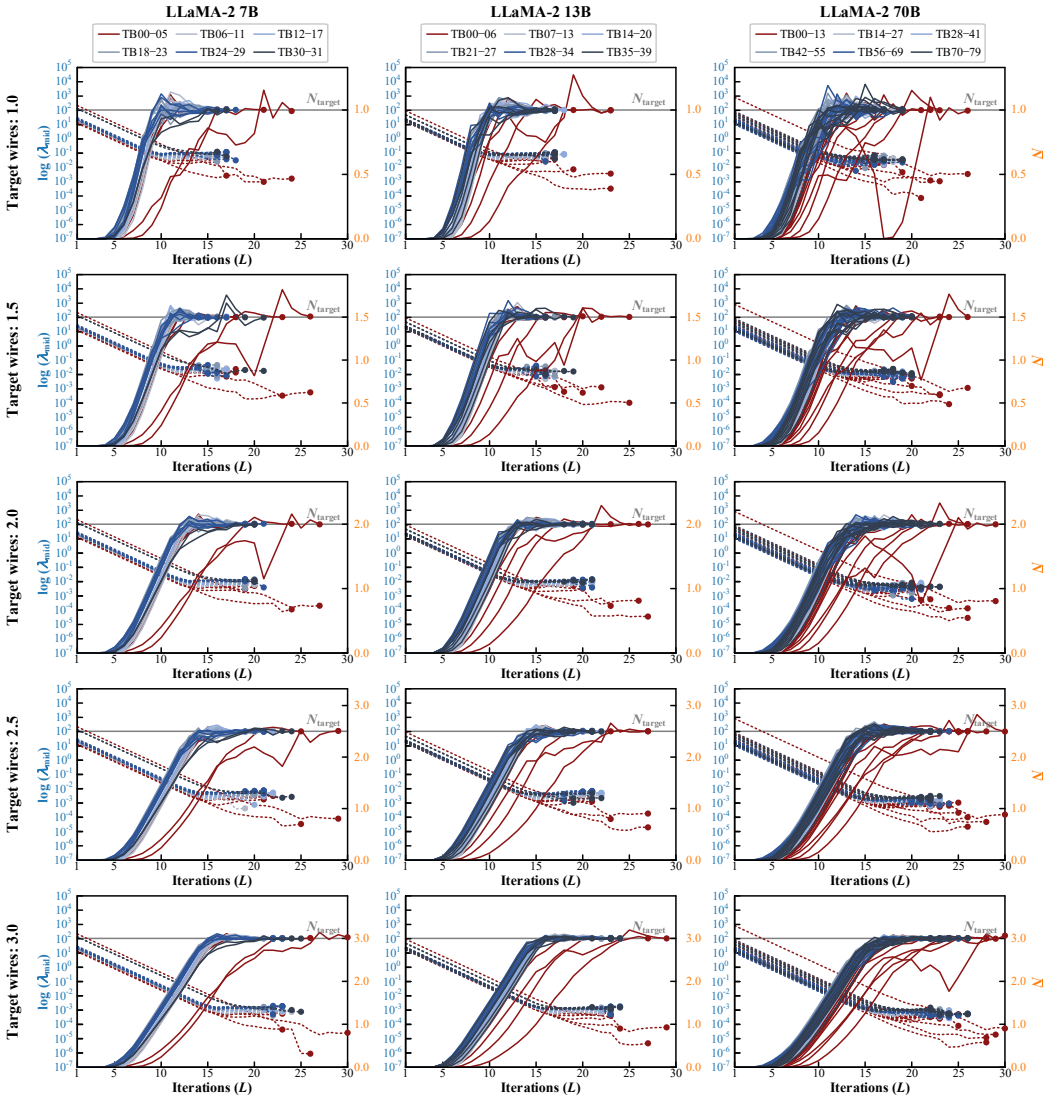


Figure 8: Convergence of Moira across LLaMA-2 models of varying size under a prescribed iteration budget $L = 30$. The left y-axis (log scale) presents λ_{mid}, and the right y-axis shows the induced average wire count $\bar{N}$. Dotted and solid lines correspond to λ_{mid} and $\bar{N}$, respectively.

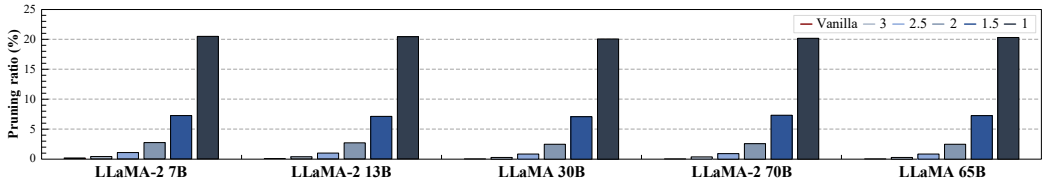
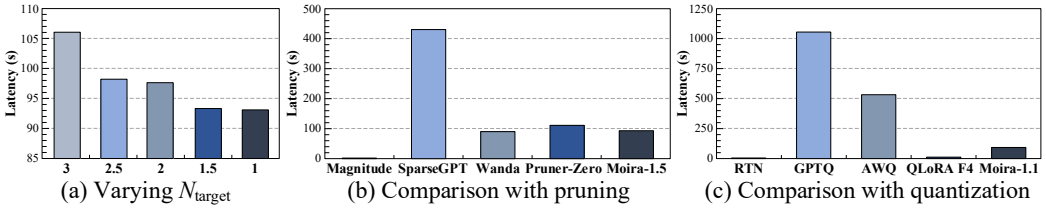


Figure 9: Implicit pruning effect on reconstructed $\tilde{\mathbf{W}}$ of Moira across LLaMA models under varying N_{target}. The pruning ratio is defined as the percentage of zero-valued weights relative to the total parameter count across all linear layers, excluding `lm_head`, consistent with the pruning baselines.

Measurement environment. All methods are tested on the same NVIDIA DGX H100 node. To accommodate the full LLaMA-2 7B parameter footprint and method-specific scratchpad memory, we use 4 GPUs in a data-parallel setup and report the wall time measured on rank 0, which is assigned to GPU 0. The common C4 calibration stage is excluded for all methods, so the reported wall time covers only the algorithmic portion.

Additional plots for varying target wire count. Figure 11 shows the remaining results for LLaMA 30B and 65B and confirms that the phenomenon discussed in Section 5.3 also holds for larger models. In particular,



(a) Varying N_{target} (b) Comparison with pruning (c) Comparison with quantization

Figure 10: Wall-time measurements of Moira and baselines on LLaMA-2 7B: **(a)** Moira under varying N_{target}, **(b)** pruning methods versus Moira-1.5, and **(c)** quantization methods versus Moira-1.1.

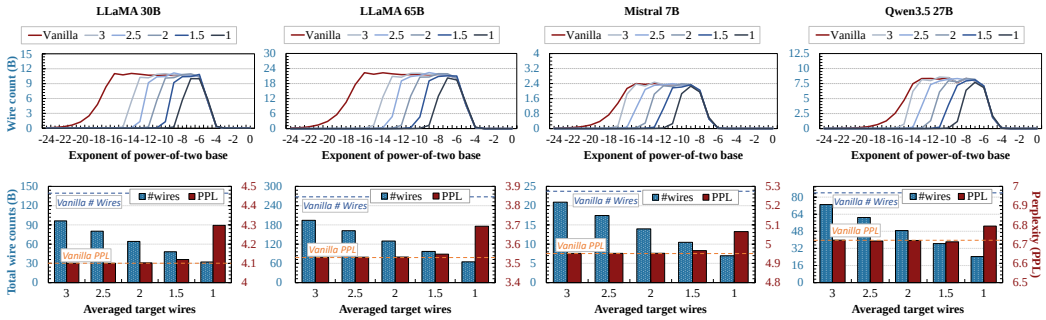


Figure 11: Effects of varying the averaged target wire count in Moira: **(top)** wire-frequency distributions over N_{target}, starting from the vanilla LLMs (FP16); **(bottom)** total wire count and performance versus N_{target}. The dotted lines denote the wire count and PPL of the vanilla model.

PPA	Component-level		Complete design		**Moira-1.1**
	Add. (①)	Acc. (②)	Fused ①+②	① and Reused ②	
Area [μm^2]	2,184.80	5,296.32	7,400.39	2,515.82	**1,173.67**
Power [mW]	2.4590	11.0806	19.6056	13.5396	**2.3551**
Latency [ns]	0.6025	1.9508	2.4751	2.5533	**1.6189**

Table 6: PPA comparison of Moira-1.1 with ShiftAddLLM, a power-of-two weight quantization method which we directly realized in RTL. ShiftAddLLM is evaluated at the component level and with fused or reused accumulation logic. RTL synthesis follows the same setup as Section 5.4.

when the average number of selected bases is around two per weight, the dominant bases again concentrate in the range 2^{-11} to 2^{-5}, while incurring negligible perplexity degradation. Consistent with the trend in Section 5.3, the degradation in this low-wire regime becomes smaller for larger models, suggesting that the larger models may admit simpler hardwired realizations with less performance loss.

Wired base distribution across LLMs and wire budgets. Figure 11 further demonstrates that the near-lossless behavior observed for the LLaMA family remains valid for both Mistral-7B and Qwen3.5-27B, despite differences in their base distributions. LLaMA concentrates more on larger-magnitude bases, whereas Mistral and Qwen increasingly use smaller bases. Nevertheless, Moira consistently favors larger-magnitude bases under constrained wire budgets while preserving near-lossless performance across all three model families. Qwen3.5-27B shows an even more favorable trend, maintaining its vanilla perplexity of 6.71 down to $N_{\text{target}} = 1.5$ and increasing by only 0.08 at $N_{\text{target}} = 1.0$. These results suggest that sparse base selection remains effective across different base distributions, while prioritizing larger-magnitude bases remains a consistent property of near-lossless configurations.

PPA comparison with power-of-two quantization. We further compare Moira with ShiftAddLLM [10], a power-of-two quantization method designed for efficient GPU execution. APoT [30], another related power-of-two quantization method, is not included since it targets CNNs. ShiftAddLLM represents weights using a $\{-1, +1\}$ basis with binary coding and LUT-based computation, leaving no zero-valued terms to eliminate in a hardwired realization. Table 6 shows that both fused and reused-accumulator implementations incur higher PPA than Moira-1.1; even the more area-efficient reused-accumulator design requires 2,515.82 μm^2, 13.5396 mW, and 2.5533 ns, compared with 1,173.67 μm^2, 2.3551 mW, and 1.6189 ns for Moira-1.1. ShiftAddLLM also uses three power-of-two terms per weight to achieve a perplexity of 5.89, whereas Moira uses only 1.1 wires on average while achieving a lower perplexity of 5.24. However, this comparison may not be entirely fair to ShiftAddLLM, as it primarily targets GPU execution rather than hardwired realization.

E Limitation

Once weights are hardwired, the resulting circuit is no longer easy to modify, update, or repurpose. This approach is therefore most meaningful when the target AI model is sufficiently mature to justify physical specialization. Transformer-based LLMs are a promising case in this regard, given their strong reproducibility and architectural stability. Moreover, when some flexibility is still needed, efficient adaptation methods such as LoRA [46] can be supported by a small portion of parameters in an on-chip, SRAM-centric manner, keeping the adaptive dataflow close to the processing unit.

System design. The full system implementation of sparsely wired digital mortal execution is beyond the scope of the present paper, as it requires concrete hardware, backend, and physical design decisions. Nevertheless, these directions illustrate the broader system-level design space that Moira opens, which we aim to explore through future system and backend co-design. Relevant choices include whether adders should be organized bit-serially; whether redundancy in base factorization should be exploited over larger tiles rather than only within channel-wise groups; how clock-domain crossings should be managed between the hardwired shift-accumulate units and other functional blocks, such as attention and RoPE, to maintain high throughput; how power should be budgeted across the system; and how hardwired connections should be mapped onto the backend interconnect stack. In particular, foundry-dependent constraints such as metal-layer assignment [9], via switching [47], routing congestion, and area overhead can directly affect whether the weight-access reduction enabled by Moira can be translated into an efficient physical implementation. These design choices, along with the required peripherals, are governed by *PPA* (performance, power, and area; performance is measured as latency). Performance is measured by inference throughput, while power and area capture the cost of logic, interconnects, and peripheral circuitry. This PPA-driven co-design is therefore a natural next step toward translating Moira's gains into economically feasible AI chips.

F Broader Impact

This work points to a new hardware direction for cutting-edge AI models that could reduce the extreme energy consumption of modern AI infrastructure. By eliminating repeated weight access and moving toward sparsely wired execution, Moira suggests a path to lower data-center power use and, in turn, reduced carbon emissions. Its impact may be especially significant for physical AI settings such as autonomous and humanoid systems, where ultra-low latency and low power are not only desirable but often safety-critical. This work may also have positive implications for security. If model parameters are embedded directly, more specifically etched, in silicon, they become harder to extract or tamper with through conventional software-level attacks. This could improve robustness against model theft and misuse. At the same time, such immutability introduces clear risks. Once a model is physically committed, failures such as harmful behavior due to faulty decisions, systematic bias, or hallucinations become much harder to correct, and compromised hardware may need to be discarded rather than patched. Therefore, the societal value of this direction will depend not only on efficiency gains but also on careful validation, secure deployment practices, and responsible decisions about when a model is sufficiently stable to justify physical implementation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect our theoretical and empirical contributions, including the proposed solutions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper clearly states all theoretical propositions and provides complete proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Results and experimental settings are clearly documented in the paper and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We include an official link to access the code.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Details of the experimental settings are included in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The reported benchmark results are deterministic post-training evaluations using publicly available `HuggingFace`-based checkpoints and implementations. Since the evaluation pipeline does not involve additional stochastic training or random initialization, separate error bars are omitted. Distributional analyses, such as the violin plot in Figure 2, are included to characterize the statistical structure of the quantities Moira analyzes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports the hardware setup and measurement environment in the Appendix, including the NVIDIA DGX H100 node and the wall-time measurement protocol.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: No harmful content or ethical risks are involved

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Societal impacts are discussed in the Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: No use of the models and data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and models are used in compliance with their licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: No new assets are introduced in this work.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Not applicable – no human subjects or crowdsourcing involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: Not applicable – no human subjects or crowdsourcing involved.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used in the core methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.