FASD: Hardware Acceleration for Multi-AI-Agent Discussion

Rui Chu[*]
Department of Electrical and Computer Engineering
Tufts University
Medford, MA 02155
Rui.Chu@tufts.edu

Kaiyuan Zhang[*]
Department of Electrical and Computer Engineering
Tufts University
Medford, MA 02155
Kaiyuan.Zhang@tufts.edu

Antian Wang
Department of Electrical and Computer Engineering
Purdue University Fort Wayne
Fort Wayne, IN 46805
antian.wang@pfw.edu

Yingjie Lao
Department of Electrical and Computer Engineering
Tufts University
Medford, MA 02155
yingjie.lao@tufts.edu

Abstract

Recent development of large language models (LLMs) has empowered multi-agent systems (MAS) through external skills and inter-agent discussion to solve complex tasks. Although such a paradigm improves the capability of LLM systems, it also introduces substantial efficiency overhead. Existing approaches mainly reduce this overhead through software-level optimization. However, these methods overlook how to exploit the graph-level structure of MAS discussion and its suitability for hardware-aware acceleration, thus causing latency, especially when the number of agents, discussion rounds, or concurrent sessions increases. In this paper, we propose **FASD**, an FPGA-Algorithm Co-design framework for Acceleration towards Graph Dynamic System Agent Discussion. We argue that MAS discussion can be formed as a graph-coupled dynamical system and thus agent activation pruning can be optimized through phase-coupled dynamics method. Based on these insights, we designed the FPGA-aware optimization for the problem. FASD mainly consists of the following parts. Firstly, we profile representative MAS workflows and construct a phase profile for dynamic system modeling. Secondly, we map the agent activation pruning operator onto an FPGA-friendly datapath. Through extensive experiments on multi-agent discussion workloads, we demonstrate that FASD improves discussion efficiency while preserving task performance, with stronger benefits as the agent graph becomes larger and more complex.

1 Introduction

Recent advances in large language models (LLMs) Chu et al. [2025], Chu and Lao [2026] have enabled increasingly complex capabilities, ranging from deliberate reasoning and problem solving Yao et al. [2023a] to tool use and interactive decision making Yao et al. [2023b], Qin et al. [2023]. LLM-powered multi-agent systems (MAS) coordinate specialized agents through role assignment, tool-augmented execution, and inter-agent communication Hong et al. [2023], Chen et al. [2023], Qian et al. [2024]. Such a paradigm has shown promising capability in solving complex tasks, where

[*]Equal contribution.

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

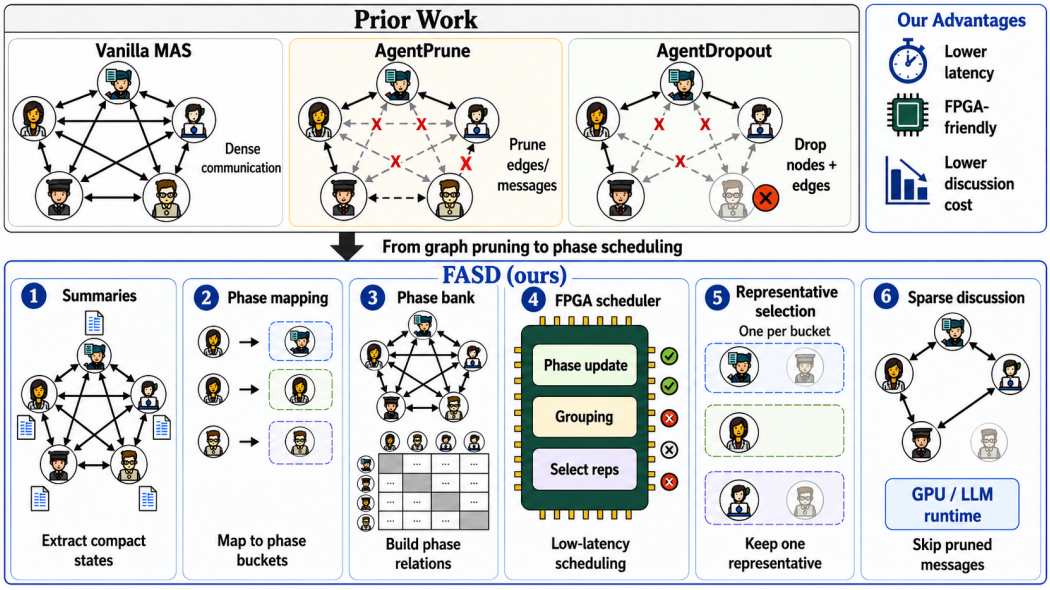

Figure 1: **Overview of FASD:** Unlike prior software-level algorithms for MAS discussion pruning, FASD maps compact agent states to phases, uses an FPGA-based adaptive dynamical-system scheduler to update phase relations and select representatives, and returns a pruning mask that skips synchronized redundant agents before downstream GPU LLM execution. Detailed workflow in Section 4.1.

individual agents can propose, critique, revise, and aggregate their responses through multi-round discussion Du et al. [2024]. Although MAS can solve more complex problems, this paradigm introduces substantial token, communication, and inference-cost overheads Li et al. [2024], Zeng et al. [2025], Zhang et al. [2025b].

In particular, discussion is a fundamental yet expensive stage in MAS because agents repeatedly exchange messages across rounds, expand each other's context windows, and increase token consumption as the number of agents and rounds grows Du et al. [2024], Zeng et al. [2025]. At the runtime level, these interactions further trigger repeated prefill computation, KV-cache movement, and cross-agent cache redundancy across LLM calls Pan et al. [2025], Ye et al. [2025]. Recent methods attempt to reduce this overhead through sparse communication, message pruning, learned topology design, and cache reuse Li et al. [2024], Zhang et al. [2025b,c], Pan et al. [2025], Ye et al. [2025]. These works suggest that MAS discussion is a structured runtime workload with graph-like communication. In such a graph, nodes represent agent states across discussion rounds, while edges represent message passing, dependency, redundancy, or learned communication relations Zhang et al. [2025b,c]. Building on this graph-based view, we interpret MAS discussion as a phase-coupled dynamical system and adopt the classical Kuramoto synchronization model Acebrón et al. [2005] as a compact graph operator to identify redundant discussion contributors. Meanwhile, hardware-friendly structure design Thorat et al. [2026], for example, on field-programmable gate array (*FPGA*), the discussion graph is still under exploration, which is especially efficient when the number of agents, discussion rounds, or concurrent sessions grows Pan et al. [2025]. The resulting pruning controller repeatedly computes phase updates, reductions, priority scores, and pruning masks before downstream LLM execution. Because these operations are regular, parallel, and fixed-point friendly Fan and Lao [2026], they are naturally suited to FPGA pipelines. However, existing FPGA-based LLM accelerators mainly target model inference rather than MAS discussion control Zeng et al. [2024], Zhao et al. [2024].

In contrast to existing optimizations, we observe that MAS discussion shows challenges in 1) the discussion pruning stage is a bottlenecked communication graph; 2) the graph grows quickly with the number of agents, which leaves room for acceleration-guided hardware design; 3) prior works overlooked modeling the whole discussion pruning as a *dynamic system* problem for better optimization, which is also easier being implemented on hardware design. We hypothesize that a hardware-algorithm aligned design can greatly accelerate the MAS discussion decision pruning.

These observations motivate the central research question we seek to address: *Designing a hardware-algorithm co-design for MAS discussion pruning.* The key challenge is to exploit the graph-level discussion structure for fast pruning while preserving the quality benefits of multi-agent discussion. We hold our hypothesis that a hardware-algorithm aligned design can greatly accelerate the MAS discussion decision pruning.

In this work, we propose FASD, an FPGA-Algorithm Co-design framework for Acceleration towards Graph Dynamic System Agent Discussion. Our key insight is that the coordination pattern in MAS discussion resembles a phase-coupled graph dynamics problem, as a *Kuramoto* model Kuramoto [2005], which describes how locally coupled oscillators can reach global phase synchronization through pairwise interactions. The proposed framework has an algorithm layer and a hardware layer: 1) At the algorithm layer, FASD formulates MAS discussion pruning as a Kuramoto-style phase-synchronization problem over the agent communication graph. Agents that synchronize in phase are treated as redundant contributors, and FASD keeps representative agents to form the active-agent mask for the next discussion round. 2) At the hardware layer, FASD maps phase update, priority scoring, and pruning-mask generation to a fixed-point FPGA datapath, reducing the latency of discussion-pruning decisions before downstream LLM execution.

Our contributions are summarized as follows:

- To the best of our knowledge, FASD is among the first works to study hardware-assisted acceleration for the discussion stage of LLM-based multi-agent systems.
- FASD is an FPGA-aware decision pruning graph accelerator that is suitable for the multi-agent system discussion.
- FASD does not need offline-training, thus providing high adaptivity.
- Through extensive experiments, we demonstrate that FASD improves discussion efficiency while preserving task performance, especially when the agent discussion graph becomes larger and more complex.

2 Related Work

Multi-Agent Acceleration One line of work accelerates LLM-based agent workflows by exploiting runtime and cache structure. KVFlow abstracts agent execution as an Agent Step Graph and uses workflow-aware KV-cache eviction and prefetching to reduce cache-miss stalls Pan et al. [2025]. KVCOMM enables cross-context KV-cache reuse by aligning offset-variant KV states across agents Ye et al. [2025]. These systems reduce repeated prefill and cache movement after agent contexts have already been formed. FASD is complementary to this direction because it acts earlier in the workflow, generating active-agent masks before redundant agents expand future contexts.

Agent-Debate Acceleration Another line of work reduces discussion cost by changing the multi-agent communication policy. Sparse MAD studies sparse debate topologies and shows that dense all-to-all communication is often unnecessary Li et al. [2024]. S^2-MAD reduces token cost by selecting useful viewpoints and suppressing ineffective debate exchanges Zeng et al. [2025]. AgentPrune prunes the spatial-temporal message-passing graph of LLM-based MAS to remove redundant communication Zhang et al. [2025b]. G-Designer learns task-adaptive communication topologies for multi-agent systems Zhang et al. [2025c]. These methods motivate FASD by showing that MAS discussion contains exploitable redundancy. FASD differs from them by using Kuramoto-style phase synchronization as a training-free pruning signal and by implementing the pruning-mask generation path on FPGA.

Hardware and Architecture Acceleration Hardware accelerators for LLMs mainly target model-level inference kernels. FlightLLM maps LLM inference onto FPGAs through an end-to-end mapping flow Zeng et al. [2024], and ChatOPU designs an FPGA overlay for sparse LLM execution Zhao et al. [2024]. FASD instead accelerates the discussion-pruning controller, whose core operations are phase update, score accumulation, and pruning-mask generation. The method is also related to phase-dynamics models. The Kuramoto model describes synchronization among coupled oscillators Acebrón et al. [2005], adaptive Kuramoto variants model evolving pairwise couplings Ha et al. [2018], and KuramotoGNN connects oscillator dynamics with graph representation behavior Nguyen

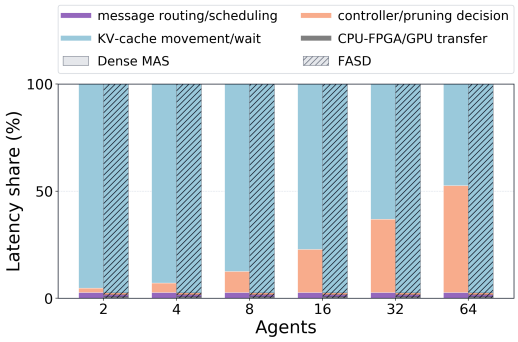

Figure 2: Latency-share breakdown of multi-agent discussion under different agent counts for the dense MAS baseline and FASD. Solid bars denote the dense MAS baseline, while hatched bars denote FASD.

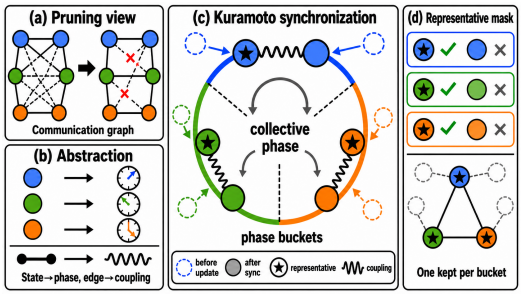

Figure 3: Balls denote agents, solid lines denote communication edges, clock pointers denote agent phases, and curved arrows denote Kuramoto phase coupling. FASD treats synchronized phases as redundant discussion states.

et al. [2024]. FASD adapts this synchronization view to MAS discussion pruning and maps the resulting fixed-point operator to an FPGA datapath.

3 Preliminaries

3.1 Bottleneck Analysis

Agent discussion, especially pruning decisions, is a major bottleneck in multi-agent systems Zhang et al. [2025b]; thus, optimization algorithms have been designed. However, these algorithms do not fully address large-scale settings with dozens of agents and many discussion rounds in a single task. As shown in Fig. 2, our preliminary results indicate that hardware-algorithm co-design can substantially reduce pruning latency, especially for larger MAS instances such as 64-agent discussions.

3.2 System Modeling

MAS Discussion as Graph Pruning. Following the spatial-temporal message-passing view in AgentPrune Zhang et al. [2025b], we treat multi-agent discussion as a communication graph over agents and rounds. At round t, the binary variable z_{ji}^t indicates whether the message from source agent a_j is routed to target agent a_i. The active routing decisions define the discussion edge set $\mathcal{E}^t$, and pruning is implemented by a source-agent mask M_j^t:

$$z_{ji}^t \in \{0,1\}, \qquad \mathcal{E}^t = \{(a_j, a_i) : z_{ji}^t = 1,\ j \neq i\}, \qquad z_{ji}^{t+1} = M_j^t. \tag{1}$$

Thus, pruning becomes a sparse graph-decision problem, as shown in *Stage A* of Fig. 3. Instead of activating all candidate edges, the controller selects which agents remain active for the next step.

Dynamic System Modeling. Optimizing the binary routing variables directly is combinatorial. As illustrated in *Stage B* of Fig. 3, FASD relaxes this graph decision into a phase coupled process, where each agent state a_i is mapped to a phase θ_i^t and each interaction is mapped to a phase difference δ_{ij}^t.

$$a_i \mapsto \theta_i^t, \qquad \delta_{ij}^t = \theta_j^t - \theta_i^t. \tag{2}$$

Following prior graph-pruning and multi-agent-dynamics studies Nguyen et al. [2024], Mitra [2025], Dubey [2026], *Stage C* in Fig. 3 uses phase synchronization as a runtime signal of redundancy. If the updated phase difference of two agents satisfies $|\theta_i' - \theta_j'| < \epsilon$, where θ_i' is the updated phase and s_i is the priority score of agent i, the controller compares (s_i, i) and (s_j, j), keeps the larger tuple as the representative, and prunes the other.

Motivated by previous works, FASD uses the adaptive *Kuramoto* formulation Acebrón et al. [2005] (detailed in Appendix B, widely used in domains like *Computer Networks*). Each agent pair has

an adaptive coupling κ_{ij}, and the coupling converges to the closed-form root, where μ and γ are adaptive-coupling constants: $\kappa^{\star}_{ij} = \frac{\mu}{\gamma}\cos(\delta_{ij})$.

3.3 Hardware Suitability of Discussion Pruning

This paper motivates the suitability from two levels. At the workload level, multi-agent discussion pruning consists of many lightweight interaction checks and mask updates across agents, which naturally exposes parallelism. At the operator level, the collapsed Kuramoto computation is regular and suitable for fixed-point operations(Appendix B, Eq. 6), making the Kuramoto abstraction both a model of redundancy and a practical hardware representation of the pruning path. Section 4 maps this operator to the FPGA datapath.

4 Method

4.1 Problem Statement

Given the scenario formulation, the goal of FASD is to design a *Kuramoto*-inspired pruning algorithm efficiently from the phase-coupled discussion state. Concretely, the controller evaluates the coupling from phase differences, uses it to reveal synchronized redundant agents, and converts the updated phases and priority scores into a pruning mask M.

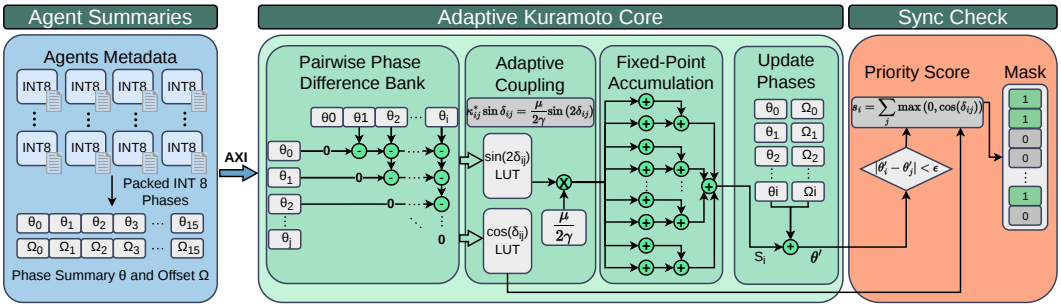

Figure 4: Overview of the FASD FPGA datapath.

Method Overview At the beginning of the workflow, as shown in Fig. 4, the CPU extracts per-agent summaries and quantizes them into INT8 phases $\Theta = \{\theta_i\}_{i=1}^{N}$ and offsets Ω_i (refer to Step 2-4 in Fig. 1), where θ_i denotes the phase of agent i, Ω_i denotes its intrinsic offset, and N is the number of active agents. On the FPGA, FASD forms pairwise phase differences $\delta_{ij} = \theta_j - \theta_i$, evaluates the coupling root $\kappa^{\star}_{ij}$ and the collapsed interaction term u_{ij}, and accumulates the interaction path into S_i. After the hardware shift, the algorithm obtains $\widetilde{S}_i$, computes the updated phase $\theta'_i = \mathrm{INT8}(\theta_i + \Omega_i + \widetilde{S}_i)$, and collects the updated phase set $\Theta' = \{\theta'_i\}_{i=1}^{N}$.

At a high level, Algorithm 1 performs one Kuramoto phase update and computes priority scores, and Algorithm 2 uses Θ', the score set $\{s_i\}_{i=1}^{N}$, the active-agent set $\mathcal{A}$, and the synchronization threshold ϵ to produce the per-agent mask $M = \{M_i\}_{i=1}^{N}$. When $|\theta'_i - \theta'_j| < \epsilon$, the algorithm keeps the higher-priority representative and suppresses redundant discussion. In the experiments, we use $\mu = 1$, $\gamma = 1$, and $\epsilon = 8$. Around this parameter setting, other parameter variations only induce approximately linear changes in the measured behavior and do not trigger nonlinear changes, so the selected parameters do not alter the main experimental conclusions.

4.2 Algorithm Implementation Details

The FPGA receives INT8 phases and offsets plus fixed constants derived from μ, γ, and N. Algorithm 1 summarizes the implemented update. For each target agent, the controller forms pairwise phase differences, evaluates the collapsed interaction $u_{ij} = \kappa^{\star}_{ij}\sin(\delta_{ij})$, accumulates S_i, applies the hardware shift r, and computes the priority score, as detailed in Appendix B and Appendix D.3.

The accumulator S_i is maintained at higher precision until the shifted term $\widetilde{S}_i = S_i \gg r$ is added to $\theta_i + \Omega_i$ and cast to INT8. Algorithm 2 then applies the same synchronization rule introduced

Algorithm 1 Phase Update and Scoring

Require: CPU-extracted INT8 phases $\Theta = \{\theta_i\}_{i=1}^{N}$, offsets Ω_i, constants μ, γ, and hardware shift r

1: **for all** target agents i **do**
2: $S_i \leftarrow 0$; $s_i \leftarrow 0$
3: **for all** neighbors $j = 1, \ldots, N$ **do**
4: $\delta_{ij} \leftarrow \theta_j - \theta_i$
5: $\kappa^{\star}_{ij} \leftarrow (\mu/\gamma)\cos(\delta_{ij})$
6: $u_{ij} \leftarrow \kappa^{\star}_{ij}\sin(\delta_{ij})$
7: $S_i \leftarrow S_i + u_{ij}$
8: $s_i \leftarrow s_i + \max\left(0, \kappa^{\star}_{ij}\right)$
9: **end for**
10: $\widetilde{S}_i \leftarrow S_i \gg r$
11: $\theta'_i \leftarrow \mathrm{INT8}(\theta_i + \Omega_i + \widetilde{S}_i)$
12: **end for**
13: **return** $\Theta' = \{\theta'_i\}_{i=1}^{N}$ and $\{s_i\}_{i=1}^{N}$

Algorithm 2 Mask Generation via Synchronization

Require: FPGA-computed updated phases $\Theta' = \{\theta'_i\}_{i=1}^{N}$, priority scores $\{s_i\}_{i=1}^{N}$, active agent set $\mathcal{A}$, and sync threshold ϵ

1: Initialize $M_i \leftarrow 1$ for $i \in \mathcal{A}$.
2: **for all** active agents i **do**
3: **for all** active agents $j \neq i$ **do**
4: **if** $|\theta'_i - \theta'_j| < \epsilon$ and $(s_i, i) < (s_j, j)$ **then**
5: $M_i \leftarrow 0$
Keep agent j as the representative.
6: **end if**
7: **end for**
8: **end for**
9: $M = \{M_i\}_{i=1}^{N}$.
10: **return** M to GPU

in Sec. 3.2 to generate the pruning mask. This rule is deterministic and simple to implement in fixed-point hardware.

Algorithm 2, line 3 compares updated phases and declares two agents redundant when $|\theta'_i - \theta'_j| < \epsilon$. When this condition holds, this algorithm follows lexicographic ordering for the tuple(s_i, i), where a larger (s_i, i) indicates a higher priority, which makes the mask keep the agent with the larger score. This rule is simple to implement in fixed-point hardware.

Algorithm Efficiency The cost in a direct adaptive Kuramoto implementation comes from the pairwise coupling state. For N active agents, the implementation must carry an $N \times N$ coupling matrix and update it over time, which leads to $O(N^2)$ complexity on storage and update overhead.

FASD reduces this scaling pressure by removing the coupling matrix. As shown in Algorithm 1 line 5, each κ_{ij} is replaced by a stable solution computed directly from the phase difference, so the controller no longer stores or iteratively updates the coupling state. As a result, storage and coupling-state update overhead drops from $O(N^2)$ to $O(N)$, while pairwise phase-difference evaluation remains a regular parallel datapath operation.

4.3 Hardware Microarchitecture

We implement the FPGA datapath as a fixed-point streaming pipeline, as shown in Fig. 4. AXI words are unpacked into phase and offset registers. A subtractor array forms the pairwise phase differences δ_{ij}, and the same wires feed a $\sin(2\delta_{ij})$ LUT bank for the phase-update path and a $\cos(\delta_{ij})$ LUT bank for the priority-score path. The $\sin(2\delta_{ij})$ LUT implements the collapsed product $\cos(\delta_{ij})\sin(\delta_{ij})$ up to a constant scale absorbed by fixed-point shifting. The datapath accumulates the two paths, applies a right shift r to obtain $\widetilde{S}_i = S_i \gg r$, computes $\theta'_i = \mathrm{INT8}(\theta_i + \Omega_i + \widetilde{S}_i)$, and compares updated phases to write the mask bits. In our RTL experiments, we set $r = 8$ for the 64-agent design; Appendix D.3 details this fixed-point choice.

In the RTL design, the trigonometric operations are implemented with small INT8 LUTs rather than Taylor units or the Coordinate Rotation Digital Computer, and scaling uses shift-based fixed-point arithmetic. Representative selection follows Algorithm 2: when two agents synchronize, the controller keeps the larger priority tuple (s_i, i) as the representative.

FPGA-Friendly Datapath This datapath is suited to FPGA fabrics because the algorithm exposes independent fixed-point LUT operations, which are supported by prior combinations of the Kuramoto algorithm with FPGA oscillator implementations Ha et al. [2018], Shchapin et al. [2023].

5 Experiments

All hardware experiments are run on an Xilinx Alveo U280 board with Vitis 2023.1, and the RTL design operates at 300 MHz. A C++ host drives RTL kernels, and the software baseline for the pruning kernel runs on an Intel i7-14900K CPU.

5.1 Experimental Setup

Following prior LLM-based MAS studies, we evaluate FASD on three categories of tasks. For general reasoning, we use MMLU [Hendrycks et al., 2021]. For mathematical reasoning, we use GSM8K [Cobbe et al., 2021], MultiArith [Roy and Roth, 2015], SVAMP [Patel et al., 2021], and AQuA [Ling et al., 2017]. For code generation [Chu et al., 2026], we use HumanEval [Chen et al., 2021]. We report accuracy for MMLU and the mathematical reasoning benchmarks, and Pass@1 for HumanEval.

Models and MAS backbones. For open-source model experiments, we build MAS using Llama3-8B-Instruct [Grattafiori et al., 2024], Qwen2.5-72B-Instruct [Qwen et al., 2025]. For direct comparison with representative GPT-based MAS settings, we additionally evaluate three-agent and five-agent GPT-based AutoGen [Wu et al., 2024] and GPTSwarm [Zhuge et al., 2024] workflows. Unless otherwise specified, the main task-level experiments use five agents. For hardware scaling experiments, we vary the number of active agents as $N \in \{2, 4, 8, 16, 32, 64\}$, and for discussion-round scaling we fix $N = 16$ and vary the number of discussion rounds.

Baselines. We compare FASD with both algorithmic and hardware baselines. For single-agent reasoning, we include Vanilla prompting, Chain-of-Thought prompting (CoT) [Wei et al., 2022], and self-consistency with CoT (SC-CoT) [Wang et al., 2023]. For multi-agent reasoning, we compare with single-round MAS ($\mathrm{MAS}_{\mathrm{round}=1}$), multi-round dense MAS ($\mathrm{MAS}_{\mathrm{round}=T}$), AutoGen, AgentVerse [Chen et al., 2023], and AgentPrune [Zhang et al., 2025b] .

Implementation Details. For general reasoning and mathematical reasoning tasks, we set the number of discussion rounds to $T = 2$; for code generation, we set $T = 4$. FASD is training-free and does not learn graph masks or optimize adjacency matrices.

Resource Utilization is reported, including LUTs, FFs, and DSPs, as a normalized utilization of baseline utilization under Alveo U280 capacity. The resource baseline is the unoptimized Kuramoto hardware implementation for the same 16-agent setting, which streams explicit K_{ij} weights and uses multiplier-based pairwise interaction terms before accumulation.

5.2 Main Results

5.2.1 Cross-Method Comparison

Accuracy Table 1 reports task performance across two base models. FASD achieves the best average performance among most multi-agent pruning baselines. The gains are consistent on reasoning and math benchmarks, while the margin becomes smaller on near-saturated tasks such as *MultiArith*. These results indicate that FASD can suppress redundant discussion states while preserving or improving task performance compared with standard MAS and communication-pruning baselines.

Latency and Cost Table 2 compares the cost of FASD with *AgentPrune* when both are attached to *AutoGen* and *GPTSwarm*. It can be seen that FASD consistently reduces the whole inference time beyond AgentPrune across both three-agent and five-agent settings.

The main reason is that AgentPrune reduces the communication graph in software, while FASD moves the pruning decision path to the FPGA controller. Meanwhile, FASD also further reduces prompt and completion tokens by suppressing synchronized redundant agents before their messages enter the downstream LLM context. These results show that FASD improves both the controller-side pruning cost and the downstream LLM inference cost, with larger gains when the host MAS contains more agents or denser communication, such as GPTSwarm.

Table 1: Performance comparison across different methods and base models. Node DR. denotes node-level pruning.

Method	Node DR.	MMLU	GSM8K	AQuA	MultiArith	SVAMP	HumanEval	Avg.
Base model: Llama3-8B-Instruct								
Vanilla	✗	53.59	70.23	41.67	91.11	75.00	53.33	64.16
CoT	✗	56.86↑3.27	70.47↑0.24	43.75↑2.08	92.25↑1.14	76.17↑1.17	54.17↑0.84	65.61↑1.45
SC (CoT)	✗	60.45↑6.86	71.59↑1.36	46.21↑4.54	92.37↑1.26	78.03↑3.03	55.46↑2.13	67.35↑3.19
$MAS_{round=1}$	✗	56.21↑2.62	69.30↓0.93	45.29↑3.62	91.67↑0.56	76.67↑1.67	48.33↓5.00	64.58↑0.42
$MAS_{round=T}$	✗	60.13↑6.54	71.48↑1.25	45.41↑3.74	90.56↓0.55	77.56↑2.56	49.17↓4.16	65.72↑1.56
AgentPrune	✗	60.78↑7.19	71.02↑0.79	47.22↑5.55	90.00↓1.11	78.34↑3.34	51.67↓1.66	66.51↑2.35
FASD (Ours)	✓	**61.93**↑8.34	**72.39**↑2.16	**47.79**↑6.12	**92.97**↑1.86	**79.37**↑4.37	**56.68**↑3.35	**68.52**↑4.36
Base model: Qwen2.5-72B-Instruct								
Vanilla	✗	82.35	91.02	83.75	98.89	92.67	85.28	88.99
CoT	✗	83.66↑1.31	92.19↑1.17	84.58↑0.83	**100**↑1.11	93.35↑0.68	86.67↑1.39	90.08↑1.09
SC (CoT)	✗	83.70↑1.35	93.67↑2.65	86.25↑2.50	**100**↑1.11	93.79↑1.12	86.83↑1.55	90.71↑1.72
Autogen	✗	82.34↓0.01	92.17↑1.15	85.73↑1.98	99.44↑0.55	93.86↑1.19	87.36↑2.08	90.15↑1.16
AgentVerse	✗	81.57↓0.78	91.59↑0.57	84.35↑0.60	**100**↑1.11	92.45↓0.22	87.49↑2.21	89.58↑0.59
$MAS_{round=1}$	✗	82.35↑0.00	93.52↑2.50	84.58↑0.83	**100**↑1.11	92.36↓0.31	84.17↓1.11	89.50↑0.51
$MAS_{round=T}$	✗	84.31↑1.96	93.28↑2.26	85.83↑2.08	**100**↑1.11	94.07↑1.40	87.08↑1.80	90.76↑1.77
AgentPrune	✗	83.66↑1.31	93.67↑2.65	87.08↑3.33	99.44↑0.55	94.33↑1.66	86.67↑1.39	90.81↑1.82
FASD (Ours)	✓	**84.43**↑2.08	**93.77**↑2.75	**87.32**↑3.57	99.86↑0.97	**95.88**↑3.21	**87.99**↑2.71	**91.54**↑2.55

Table 2: Token and normalized time cost under three- and five-agent GPT-4-based settings. Values in parentheses after prompt tokens denote the remaining prompt-token ratio relative to the corresponding unpruned AutoGen or GPTSwarm baseline. Whole Inference Time is normalized to the corresponding unpruned MAS baseline, where 100% denotes no pruning.

$N = 3$					$N = 5$				
Dataset	Method	# Prompt Tokens	# Completion Tokens	Whole Inference Time	Dataset	Method	# Prompt Tokens	# Completion Tokens	Whole Inference Time
MMLU	AutoGen	346,028	66,204	100.0%	MMLU	AutoGen	486,034	89,224	100.0%
	+AgentPrune	274,665(79.4%)	66,803	90.06%		+AgentPrune	349,583(71.9%)	86,582	85.49%
	+FASD	247,551(71.5%)	61,915	74.58%		+FASD	314,372(64.7%)	79,087	68.27%
HumanEval	AutoGen	351,985	90,762	100.0%	HumanEval	AutoGen	492,273	130,196	100.0%
	+AgentPrune	254,203(72.2%)	89,362	85.17%		+AgentPrune	315,105(64.0%)	139,714	84.06%
	+FASD	230,678(65.5%)	81,698	69.86%		+FASD	283,646(57.6%)	123,013	65.72%
GSM8K	AutoGen	2,317,937	624,055	100.0%	GSM8K	AutoGen	4,327,740	998,042	100.0%
	+AgentPrune	1,481,780(63.9%)	629,771	78.84%		+AgentPrune	3,791,251(87.6%)	1,156,884	78.48%
	+FASD	1,329,596(57.4%)	574,653	64.65%		+FASD	3,040,000(70.2%)	570,000	67.76%
MMLU	GPTSwarm	1,521,504	325,994	100.0%	MMLU	GPTSwarm	3,055,230	569,124	100.0%
	+AgentPrune	554,698(36.5%)	336,887	53.16%		+AgentPrune	990,312(32.4%)	439,551	49.95%
	+FASD	500,128(32.9%)	299,426	43.40%		+FASD	904,846(29.6%)	405,856	35.50%
HumanEval	GPTSwarm	1,478,312	612,815	100.0%	HumanEval	GPTSwarm	2,736,136	1,004,616	100.0%
	+AgentPrune	432,480(29.3%)	598,367	55.17%		+AgentPrune	745,617(27.3%)	745,926	49.37%
	+FASD	387,468(26.2%)	536,175	44.34%		+FASD	674,480(24.7%)	662,987	35.90%
GSM8K	GPTSwarm	6,274,665	186,510	100.0%	GSM8K	GPTSwarm	14,005,945	3,156,916	100.0%
	+AgentPrune	3,009,115(48.0%)	173,296	56.12%		+AgentPrune	3,526,035(25.2%)	730,552	37.56%
	+FASD	2,715,202(43.3%)	156,792	44.30%		+FASD	3,209,596(22.9%)	672,990	22.30%

5.2.2 Inner Comparison

In Fig. 5, this paper compares the four hardware settings under the same discussion pruning workload while changing the number of agents. FASD has the lowest latency for every measured agent count, except the smallest measured agent-count setting, where the workload is too small for the coupling collapse benefits to dominate the fixed interface overhead. The reason is that FASD removes the streamed K_{ij} coupling matrix and the adaptive coupling state update. Its datapath processes only phase differences, trigonometric LUTs, fixed-width adders, and shift operators, which reduces its length compared to the other three settings.

The scaling trend also differs from that in Fig. 5. The latency of FASD grows slowly as the agent count increases. The other settings grow faster because their compute paths need more per-agent work to the hardware interface. Kuramoto needs each target agent row to be processed, so the latency grows approximately linearly with N. Model Tree improves the reduction structure, but for large N it still pays the cost of moving and accumulating explicit coupling rows. This result could be scaled up to the multi-agent system setting, where workloads use tens or hundreds of agents, or if discussion

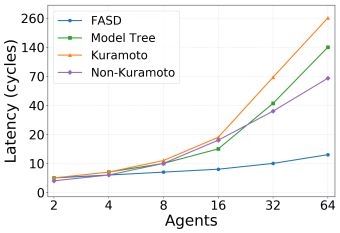

Figure 5: Pruning controller latency versus the number of active agents.

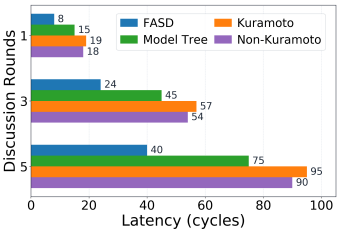

Figure 6: Pruning-controller latency versus the number of discussion rounds for the 16-agent workload.

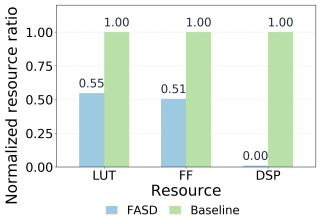

Figure 7: Normalized FPGA resource utilization of FASD and the unoptimized Kuramoto baseline.

control becomes a visible runtime bottleneck, FASD provides a practical solution for keeping the pruning decision latency small.

The discussion-round experiment shows the same growth trend across different agent counts, indicating that when rounds are executed sequentially, and there is no inter-round overlap, latency grows linearly with the number of rounds. Also, 16-agent systems have already been used as an important scale point in recent MAS studies: VAST Phan et al. [2021], AgentsNet Grötschla et al. [2025], and Global Leonardos et al. [2021]. To this end, this paper fixes the 16-agent setting in the discussion rounds experiment, the results shown in Fig. 6. FASD has the lowest latency across all round settings. This happens because the advantage is preserved and grows over cycles as more rounds are executed. All four settings scale linearly with the number of rounds. The key difference is the constant in each round. FASD lowers this constant by replacing the full coupling path with the adaptive Kuramoto coupling collapse and by avoiding runtime K_{ij} weight traffic. This saves the extra rounds in a multi-agent discussion system, which are used to improve reasoning quality through revision. When discussions become frequent, FASD provides a low-latency hardware mechanism to prevent the practice discussion pruning stage from becoming a bottleneck.

5.3 Hardware Resource Utilization

As shown in Fig. 7, FASD uses fewer hardware resources than the baseline. The baseline keeps a full adaptive Kuramoto coupling path. It stores or streams pairwise coupling values and updates coupling for every agent pair. This needs more registers for the intermediate state, more LUT logic for control, and many DSP multipliers for pairwise products. FASD removes this path through the coupling collapse. FASD only requires phase differences, small sine- and cosine-LUTs, fixed-width adders, and hardware shifts. The current RTL implements the scale factor as a fixed-point shift.

6 Conclusion

This paper presented FASD, which is the first work to study FPGA-assisted acceleration for the discussion stage of LLM-based multi-agent systems. We formulated agent pruning as a graph-dynamics problem and used the adaptive Kuramoto coupling collapse to convert pruning into an FPGA-friendly fixed-point pipeline. FASD does not require offline training, which gives it high adaptivity across different discussion workloads and agent graphs. Through experiments, FASD achieves the lowest pruning controller latency among the compared hardware settings with low hardware resource cost, while improving discussion efficiency and preserving task performance, especially as the agent graph becomes larger and more complex. Overall, these results show that graph dynamics pruning is a practical and scalable direction for hardware acceleration of MAS discussion.

Acknowledgments

This research was supported in part by the National Science Foundation (NSF) awards 2426299, 2413046, 2412357, and 2413235.
No competing interests.

References

Juan A Acebrón, Luis L Bonilla, Conrad J Pérez Vicente, Félix Ritort, and Renato Spigler. The kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of modern physics*, 77(1):137–185, 2005.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL `https://arxiv.org/abs/2107.03374`.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, et al. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*, 2023.

Hyeong Kyu Choi, Xiaojin Zhu, and Sharon Li. Debate or vote: Which yields better decisions in multi-agent large language models? In *Advances in Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=iUjGNJzrF1`.

Rui Chu and Yingjie Lao. HAS: Highlight-guided attention steering for multimodal LLM video summarization. In *Computer Vision – ECCV 2026*, volume 17077 of *Lecture Notes in Computer Science*, pages 138–157, Cham, 2026. Springer Nature Switzerland. doi: 10.1007/978-3-032-37041-9_8.

Rui Chu, Bingyin Zhao, Hanling Jiang, Shuchin Aeron, and Yingjie Lao. BAM-ICL: Causal hijacking in-context learning with budgeted adversarial manipulation. In *Advances in Neural Information Processing Systems*, volume 38, pages 14152–14183. Curran Associates, Inc., 2025. doi: 10.52202/085713-0476. URL `https://proceedings.neurips.cc/paper_files/paper/2025/hash/14f1775578f88239af1aa46c75633ede-Abstract-Conference.html`.

Rui Chu, Nicole Meng, Tanusree Das Tithy, Thomas Cao, Faysal Hossain Shezan, and Yingjie Lao. AegisCodeGen: Toward copyright-aware code generation with runnability-guided fusion. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, pages 57326–57337, Budapest, Hungary, October 2026. Association for Computational Linguistics. doi: 10.18653/2026.emnlp-main.2521. URL `https://aclanthology.org/2026.emnlp-main.2521/`.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11733–11763. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/du24e.html`.

Mohit Dubey. Phase-scheduled multi-agent systems for token-efficient coordination. *arXiv preprint arXiv:2604.17400*, 2026.

Sinuo Fan and Yingjie Lao. Budget-aware LLM quantization and low-rank correction via information-guided subspace matrices. In *Proceedings of the Thirty-Fifth International Joint Conference on Artificial Intelligence, IJCAI-26*, pages 4269–4277. International Joint Conferences on Artificial Intelligence Organization, 2026. doi: 10.24963/ijcai.2026/475.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Florian Grötschla, Luis Müller, Jan Tönshoff, Mikhail Galkin, and Bryan Perozzi. Agentsnet: Coordination and collaborative reasoning in multi-agent llms. *arXiv preprint arXiv:2507.08616*, 2025.

Seung-Yeal Ha, Jaeseung Lee, Zhuchun Li, and Jinyeong Park. Emergent dynamics of kuramoto oscillators with adaptive couplings: conservation law and fast learning. *SIAM Journal on Applied Dynamical Systems*, 17(2): 1560–1588, 2018.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Yoshiki Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In *International symposium on mathematical problems in theoretical physics: January 23–29, 1975, kyoto university, kyoto/Japan*, pages 420–422. Springer, 2005.

Stefanos Leonardos, Will Overman, Ioannis Panageas, and Georgios Piliouras. Global convergence of multi-agent policy gradient in markov potential games. *arXiv preprint arXiv:2106.01969*, 2021.

Yunxuan Li, Yibing Du, Jiageng Zhang, Le Hou, Peter Grabowski, Yeqing Li, and Eugene Ie. Improving multi-agent debate with sparse communication topology. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7281–7294, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.427. URL https://aclanthology.org/2024.findings-emnlp.427/.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 158–167, 2017.

Nicole Meng, Caleb Manicke, Ronak Sahu, Caiwen Ding, and Yingjie Lao. Advancing adversarial robustness in GNeRFs: The IL2-NeRF attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16388–16397, 2025. doi: 10.1109/CVPR52734.2025.01528.

Nicole Meng, Ronak Sahu, Miao Yin, Faysal Hossain Shezan, and Yingjie Lao. WhisperSplat: Lossless steganography in 3D Gaussian splatting. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, pages 88019–88037. PMLR, 2026. URL https://proceedings.mlr.press/v306/meng26h.html.

Chiranjit Mitra. Synchronization dynamics of heterogeneous, collaborative multi-agent ai systems. *arXiv preprint arXiv:2508.12314*, 2025.

Tuan Nguyen, Hirotada Honda, Takashi Sano, Vinh Nguyen, Shugo Nakamura, and Tan Minh Nguyen. From coupled oscillators to graph neural networks: Reducing over-smoothing via a kuramoto model-based approach. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, 2024. URL https://openreview.net/forum?id=1AV7zZvNGc.

Zaifeng Pan, Ajjkumar Patel, Yipeng Shen, Zhengding Hu, Yue Guan, Wan-Lu Li, Lianhui Qin, Yida Wang, and Yufei Ding. Kvflow: Efficient prefix caching for accelerating llm-based multi-agent workflows. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=5Iw1nDtYmT.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 2080–2094, 2021.

Thomy Phan, Fabian Ritz, Lenz Belzner, Philipp Altmann, Thomas Gabor, and Claudia Linnhoff-Popien. Vast: Value function factorization with variable agent sub-teams. *Advances in neural information processing systems*, 34:24018–24032, 2021.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 15174–15186, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The twelfth international conference on learning representations*, 2023.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1743–1752, 2015.

Dmitry Sergeevich Shchapin, Anastasiia Aleksandrovna Emelianova, and Vladimir Isaakovich Nekorkin. A chaotic oscillation generator based on mixed dynamics of adaptively coupled kuramoto oscillators. *Chaos, Solitons & Fractals*, 166:112989, 2023.

Kiran Thorat, Nicole Meng, Caiwen Ding, Yingjie Lao, and Zhijie Jerry Shi. CLDRoute: Conditional latent diffusion for routability map generation in physical design. *arXiv preprint arXiv:2607.16674*, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=1PL1NIMMrw`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023b.

Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, and Yiran Chen. Kvcomm: Online cross-context kv-cache communication for efficient llm-based multi-agent systems. In *Advances in Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=yGOytgjurF`.

Shulin Zeng, Jun Liu, Guohao Dai, Xinhao Yang, Tianyu Fu, Hongyi Wang, Wenheng Ma, Hanbo Sun, Shiyao Li, Zixiao Huang, Yadong Dai, Jintao Li, Zehao Wang, Ruoyu Zhang, Kairui Wen, Xuefei Ning, and Yu Wang. Flightllm: Efficient large language model inference with a complete mapping flow on fpgas. In *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 223–234, 2024. doi: 10.1145/3626202.3637562. URL `https://doi.org/10.1145/3626202.3637562`.

Yuting Zeng, Weizhe Huang, Lei Jiang, Tongxuan Liu, XiTai Jin, Chen Tianying Tiana, Jing Li, and Xiaohua Xu. S^2-MAD: Breaking the token barrier to enhance multi-agent debate efficiency. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 9393–9408, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.475. URL `https://aclanthology.org/2025.naacl-long.475/`.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025a. URL `https://openreview.net/forum?id=imcyVlzpXh`.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL `https://openreview.net/forum?id=LkzuPorQ5L`.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=LpE54NUnmO.

Tiandong Zhao, Shaoqiang Lu, Chen Wu, and Lei He. Chatopu: An fpga-based overlay processor for large language models with unstructured sparsity. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2024. doi: 10.1145/3676536.3676761. URL https://doi.org/10.1145/3676536.3676761.

Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024.

Appendix

A Summary of Appendix

We include the following supplementary materials to provide additional details on our methodology, experimental setup, and discussions.

- B **Kuramoto preliminaries and datapath details.** Background on the (adaptive) Kuramoto formulation and the derived FPGA-friendly computation flow.
- C **Additional review.** Supplementary literature review tables and discussion.
- D **More experiment settings.** Evaluation protocol, hardware setup, metrics, and other implementation details.
- E **Limitations.** Assumptions and constraints of FASD.
- F **Broader impacts.** Potential benefits, risks, and deployment considerations.
- G **LLM usage disclosure.** How LLM tools were used during manuscript preparation.

B Additional Preliminaries

Kuramoto model. The Kuramoto model is a classical model for synchronization among coupled phase oscillators Acebrón et al. [2005]. Each node i has a phase θ_i and an offset Ω_i. Its phase evolves according to

$$\dot{\theta}_i = \Omega_i + \frac{1}{N}\sum_{j=1}^{N} \kappa_{ij} \sin(\theta_j - \theta_i), \tag{3}$$

where κ_{ij} is the coupling strength from node j to node i. The sinusoidal term pulls positively coupled phases closer: if two nodes interact strongly and their phases become close, they are said to be synchronized.

In Fig. 3, we use this synchronization view as a compact abstraction for multi-agent discussion. Each agent is mapped to a phase, each communication edge induces a phase interaction, and synchronized phases indicate candidate redundant discussion states. FASD then converts this synchronization pattern into a pruning mask, while Sec. 4 instantiates Eq. 3 with an adaptive coupling collapse for hardware execution.

For N active agents, the phase dynamics with adaptive coupling can be written as

$$\dot{\theta}_i = \Omega_i + \frac{1}{N}\sum_{j=1}^{N} \kappa_{ij} \sin(\theta_j - \theta_i), \qquad \varepsilon\dot{\kappa}_{ij} = \mu\cos(\delta_{ij}) - \gamma\kappa_{ij}, \tag{4}$$

where Ω_i is the offset of agent a_i, κ_{ij} is the adaptive coupling from a_j to a_i, and $\delta_{ij} = \theta_j - \theta_i$.

For each phase difference δ_{ij}, the adaptive coupling has per-pair minimizer

$$\kappa_{ij}^{\star} = \arg\min_{\kappa_{ij}} \left(\frac{\gamma}{2}\kappa_{ij}^2 - \mu\kappa_{ij}\cos(\delta_{ij})\right) = \frac{\mu}{\gamma}\cos(\delta_{ij}), \qquad u_{ij} = \kappa_{ij}^{\star}\sin(\delta_{ij}). \tag{5}$$

Datapath

$$\delta_{ij} \to \{\sin(2\delta_{ij}), \cos(\delta_{ij})\} \to (S_i, s_i) \to \theta_i' \to M_i. \tag{6}$$

Here S_i denotes the interaction accumulator for agent i, s_i denotes its priority score, θ_i' denotes its updated phase, M_i denotes the pruning mask, and ϵ denotes the synchronization threshold. The phase difference δ_{ij} is computed by subtractors, the trigonometric terms are evaluated by small lookup tables, S_i and s_i are accumulated by fixed-width reductions, and the final mask is generated by comparing updated phases under the synchronization threshold ϵ. Therefore, the Kuramoto abstraction is not only a modeling choice for redundant discussion states, but also a hardware friendly representation of the pruning control path. Section 4 instantiates this operator as an FPGA datapath.

C Additional Review

Here we are listing some results from comparable works.

Table 3: G-Designer comparison across different agent configurations on MMLU using gpt-3.5-turbo.

Method	Metric	5 Agents	10 Agents	20 Agents
Chain	Accuracy (%)	70.59	71.24	71.98
	Time (min)	15.73	30.20	56.18
	#Prompt Tokens	351,802	702,164	1,378,328
	Cost (USD)	0.5228	1.0434	2.0482
Complete Graph	Accuracy (%)	71.90	72.16	72.51
	Time (min)	16.85	34.21	66.47
	#Prompt Tokens	545,984	1,669,451	5,648,834
	Cost (USD)	0.7161	2.1770	7.3662
GPTSwarm	Accuracy (%)	72.55	73.86	75.38
	Time (min)	62.14	186.86	412.18
	#Prompt Tokens	3,055,236	9,048,465	30,317,341
	Cost (USD)	4.2190	12.4961	41.4235
G-Designer	Accuracy (%)	73.20	74.51	77.82
	Time (min)	19.26	36.04	68.89
	#Prompt Tokens	452,329	885,332	1,852,538
	Cost (USD)	0.6036	1.2768	2.6713

Table 4: MaAS efficiency comparison on the MATH benchmark.

Method	Training				Inference				Overall
	Prompt Token	Completion Token	Total Cost ($)	Wall-clock Time (min)	Prompt Token	Completion Token	Total Cost ($)	Wall-clock Time (min)	Acc. (%)
LLM-Debate	–	–	–	–	3,275,764	10,459,097	6.76$	92	48.54
DyLAN	22,152,407	16,147,052	13.01$	508	6,081,483	3,303,522	2.89$	39	48.63
MacNet	–	–	–	–	7,522,057	2,043,600	2.35$	47	45.18
GPTSwarm	21,325,266	6,369,884	7.02$	129	3,105,571	788,273	0.93$	30	47.88
AFlow	33,831,239	29,051,840	22.50$	184	2,505,944	2,151,931	1.66$	23	51.28
MaAS	3,052,159	2,380,505	3.38$	53	1,311,669	853,116	0.42$	19	51.82

Table 5: MaAS cross-model transferability results.

Dataset	Method	gpt-4o-mini	Qwen-2.5-72b / 70b	llama-3.1-70b
HumanEval	vanilla	87.08	85.60	80.06
	+MaAS	92.85	90.14	85.26
MATH	vanilla	46.29	63.80	31.93
	+MaAS	51.82	69.35	42.97

Table 6: AgentPrune training and total token costs on GSM8K.

LLM-MAS	#Training Prompt Tokens	#Total Prompt Tokens	Ratio (%)	#Training Completion Tokens	#Total Completion Tokens	Ratio (%)
Complete Graph+AP	274,821	8,526,035	3.22	69,808	2,022,560	3.45
Random Graph+AP	269,732	7,495,738	3.59	63,054	1,796,603	3.50
AutoGen+AP	158,474	3,791,251	4.18	78,899	1,156,884	6.82
GPTSwarm+AP	91,192	3,526,035	2.80	22,982	730,552	3.10

Table 7: KVFlow runtime numbers explicitly reported in the paper text. KVFlow reports most runtime results in figures; this table only includes numeric values stated in text.

Setting	Comparison	Reported Number
Single workflow with large prompts	KVFlow vs. SGLang w/ HiCache	up to $1.83\times$ speedup
Many concurrent workflows	KVFlow vs. SGLang w/ HiCache	up to $2.19\times$ speedup
4096/32/32, branches=1	KVFlow vs. SGLang w/ HiCache	$1.24\times$
4096/32/32, branches=1	KVFlow vs. GPU-only SGLang	$1.42\times$
Fixed tokens = 8192	Average KVFlow speedup	$1.30\times$
Fixed tokens = 4096	Average KVFlow speedup	$1.22\times$
Optimization breakdown	Workflow-aware eviction only	average $1.11\times$ speedup
Optimization breakdown	Eviction + overlapped prefetching	$1.29\times$ speedup
High-concurrency workflow	KVFlow over SGLang	up to $1.25\times$ speedup
High-concurrency workflow	Naive LRU-based HiCache under 1024 fixed prompt tokens with 64 concurrent workflows	$0.57\times$ performance of SGLang
High-concurrency workflow	KVFlow over naive LRU-based HiCache	up to $2.19\times$ performance gain
PEER-style workflow	KVFlow over baselines	up to $1.12\times$ and $1.08\times$ speedups

Table 8: KVCOMM performance under different numbers of collaborating agents.

Dataset	Metric	Method	2 Agents	3 Agents	4 Agents	5 Agents
MMLU	Accuracy (%)	Original	47.1	66.7	68.0	69.9
		CacheBlend	65.4	65.4	65.4	67.3
		KVCOMM	64.7	68.6	68.0	69.9
	Reuse Rate (%)	Original	0	0	0	0
		CacheBlend	80	80	80	80
		KVCOMM	74.5	69.9	70.1	67.6
GSM8K	Accuracy (%)	Original	81.1	82.4	82.1	81.7
		CacheBlend	82.0	75.1	65.1	57.1
		KVCOMM	81.5	81.7	80.6	79.6
	Reuse Rate (%)	Original	0	0	0	0
		CacheBlend	80	80	80	80
		KVCOMM	79.6	77.0	73.4	71.0
HumanEval	Pass@1 (%)	Original	86.3	83.9	84.5	85.1
		CacheBlend	31.1	21.1	30.4	32.9
		KVCOMM	81.4	83.2	83.2	83.2
	Reuse Rate (%)	Original	0	0	0	0
		CacheBlend	80	80	80	80
		KVCOMM	87.6	84.7	81.1	77.8

Table 9: KVCOMM per-agent TTFT breakdown and speedup.

TTFT (ms)	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5
Original	125.8	192.4	258.3	330.9	428.6
KVCOMM	5.5	7.7	10.2	13.5	17.5
1st Token Decode	21.4	21.2	21.2	21.3	21.1
Others	86.6	9.1	10.7	13.5	16.2
Speedup	1.11×	5.06×	6.14×	6.85×	7.82×

Table 10: KVCOMM mean TTFT speedup using Llama-3.1 with three agents.

Out_len	In_len (Prefix Sequence)				
	64	128	256	512	1024
128	2.24×	2.31×	2.56×	3.07×	4.45×
256	2.50×	2.51×	2.83×	3.44×	4.75×
512	3.05×	3.18×	3.49×	4.09×	5.34×
1024	4.40×	4.48×	4.75×	5.35×	6.72×

Table 11: AgServe latency comparison between dynamic and static allocation strategies.

Method	Average (s)	P90 (s)	P50 (s)
AgServe	119.38	177.05	106.81
FS1	125.43	206.86	106.85
FS2	242.35	295.86	237.93

Table 12: AgServe cost and quality for Alfworld under different pricing plans.

Metric	AgServe	Llama-8B	Llama-70B	GPT-4o	RouteLLM	Cascade
Original Cost	0.0177	0.0023	0.0376	0.1073	0.0337	0.1395
Cost(-30% Rent)	0.0137	0.0016	0.0263	0.1073	0.0324	0.1353
Cost(-60% Rent)	0.0097	0.0009	0.0151	0.1073	0.0312	0.1311
Quality	90	21	72	94	25	69

Table 13: AgServe QMM overhead.

	Q-Judge	R-Judge	AW	CG	KG	M2W
Overhead (s)	0.05	0.03	0.3	0.04	0.005–2	0.03

Table 14: AgServe human-like judge evaluation.

Method	Cost (.01$)	Quality	Overhead (s/req)	Gross Cost (.01$)
Human	1.505	91.75	14.71	2.944
AgServe	1.768	89.57	–	1.768

Table 15: AgServe centralized and distributed scheduling comparison. EL and SL are in seconds.

LR (s/agent)	EL-C	Q-C	SL-C	EL-D	Q-D	SL-D
0.8	64.76	84.18	0.42	47.66	85.83	0.09
0.4	97.45	83.39	0.43	60.12	85.86	0.08
0.2	129.99	85.03	0.46	68.94	85.06	0.09

D More Experiment Settings

Evaluation protocol. We evaluate FASD from both the algorithmic and hardware perspectives. At the algorithmic level, FASD is inserted as a lightweight discussion-control module into existing LLM-based multi-agent systems (MAS). It consumes compact summaries of active agents, generates a pruning mask, and suppresses redundant agent messages or activations in the next discussion round. At the hardware level, we replay the same discussion-control workloads and measure the latency and resource cost of the FASD FPGA datapath against software and hardware baselines.

Hardware setup. All FPGA experiments are conducted on an Xilinx Alveo U280 board with Vitis 2023.1. The RTL kernel runs at 300 MHz and is driven by a C++ host program. The host packs INT8 agent phases and optional offsets into AXI stream frames, sends them to the FPGA controller, and receives the pruning mask from the FPGA. The CPU software baseline runs on an Intel i7-14900K processor. Here, controller latency refers only to the pruning-controller microbenchmark: it is measured from the arrival of the compact discussion frame at the controller interface to the emission of the pruning mask. This controller-only metric excludes downstream LLM prefill and decoding, and the manuscript does not claim a separate end-to-end breakdown of CPU preprocessing, host–FPGA transfer, FPGA kernel execution, and GPU LLM prefill/decode.

Evaluation metrics. We report task performance, prompt-token consumption, and completion-token consumption, controller latency, end-to-end latency, and FPGA resource utilization. End-to-end latency is reported as the median (p50) and 95th-percentile (p95) latency over the full MAS workflow. Speedup is computed as

$$\text{Speedup} = \frac{\text{Latency}_{\text{Dense}}}{\text{Latency}_{\text{Method}}}.$$

For hardware efficiency, we report LUT, FF, DSP, BRAM, and URAM utilization as percentages of the Alveo U280 capacity. We also report the percentage of GPU prefill/decode time saved by discussion pruning:

$$\text{GPU Saved} = 1 - \frac{\text{GPUTime}_{\text{Method}}}{\text{GPUTime}_{\text{Dense}}}.$$

For iso-quality acceleration experiments, we treat a method as preserving task quality when its performance drop relative to dense MAS is within one percentage point, or when the confidence interval overlaps with the dense baseline.

Baselines. For hardware-side comparison, we include: (1) FASD-CPU, a CPU implementation of the same FASD controller; (2) Kuramoto, a plain FPGA Kuramoto synchronization baseline with serialized row-wise computation; (3) Kuramoto-Tree, which maps the standard Kuramoto coupling summation to a balanced hardware reduction tree; and (4) Non-Kuramoto, a scalar-feature pruning baseline that does not model discussion as phase-coupled dynamics. All hardware baselines receive the same compact agent summaries and output pruning masks.

D.1 Evaluation Metrics

Latency Scaling with Agent Count and **Latency Scaling with Discussion Rounds** . This paper evaluates the latency of the same discussion pruning workload with different agent counts and discussion rounds under four settings. *FASD* is our proposed FPGA design based on the adaptive Kuramoto coupling collapse Ha et al. [2018]. *Model Tree* is the FPGA Kuramoto-tree baseline, which uses the standard Kuramoto phase synchronization update Nguyen et al. [2024], with mapping the coupling summation to a balanced hardware reduction structure. This baseline tests the benefit of hardware tree accumulation without the FASD coupling collapse. *Kuramoto* is the plain FPGA Kuramoto baseline, which uses the same standard Kuramoto synchronization rule, and the same pruning objective Nguyen et al. [2024], but processes target-agent rows in a serialized row-wise datapath. *Non-Kuramoto* is the previous non-Kuramoto FPGA baseline, which uses scalar feature pruning, which is an engineering baseline from recent multi-agent LLM pruning and sparse-discussion work Zhang et al. [2025b], but it does not model discussion as adaptive oscillator dynamics. For FASD, the input frame contains phases and optional offsets only. For Model Tree and Kuramoto, the input frame also includes the explicit Kuramoto coupling weights required by those baselines. For Non-Kur., the same phase summaries are converted to the scalar feature and agent-ID stream

Table 16: Comparison of prior agent-debate and communication-reduction works. ✓ denotes direct support, ✗ denotes no support, and —denotes partial support. The last column lists the most relevant efficiency or quality number reported by each paper.

Method	MAD / MAS	Msg. Prune	Topo. Ctrl.	Adapt. Cost	Debate Gate	Reported Evidence
Multiagent Debate Du et al. [2024]	✓	✗	✗	✗	✗	Improves factuality and reasoning through multi-agent discussion, but increases LLM calls, rounds, and context length.
Sparse MAD Li et al. [2024]	✓	✗	✓	✗	✗	Neighbor-connected topology gives $+2\%$ on MATH, keeps GSM8K accuracy unchanged, and reduces reasoning input-token cost by $> 40\%$.
AgentPrune Zhang et al. [2025b]	✓	✓	✓	—	✗	Comparable performance at \$5.6 cost versus \$43.7, with 28.1–72.8% token reduction.
S^2-MAD Zeng et al. [2025]	✓	✓	—	✗	✗	Up to 94.5% token-cost reduction compared with MAD while keeping performance degradation below 2.0%.
G-Designer Zhang et al. [2025c]	✓	✗	✓	✓	✗	Achieves 84.50% MMLU accuracy and 89.90% HumanEval pass@1; reduces HumanEval token consumption by up to 95.33%.
MaAS Zhang et al. [2025a]	✓	—	✓	✓	—	Uses only 6–45% of the inference cost of existing MAS while improving performance by 0.54–11.82%.
Debate or Vote Choi et al. [2025]	✓	✗	✗	—	✓	Across seven NLP benchmarks, majority voting alone accounts for most gains usually attributed to multi-agent debate.
Ours	✓	✓	✓	✓	✓	Lightweight communication controller for message pruning, topology sparsification, query-adaptive cost allocation, and debate gating.

accepted by the previous module. Thus, the comparison measures the hardware cost of making the same pruning decision under different discussion-control algorithms.

D.2 Motivation Experiments

Prior Works. targets a different bottleneck: redundant messages and unnecessary communication before they enter the LLM context window. Table 16 summarizes prior debate and multi-agent communication works using the same requirement-style format. The key observation is that existing debate acceleration methods reduce tokens or communication edges, but they do not provide a unified runtime controller that jointly performs message pruning, topology selection, query-adaptive cost allocation, and debate gating.

Motivating Observation. Table 16 shows that the cost of multi-agent discussion is not only a serving-system problem but also a communication-policy problem. Sparse MAD, AgentPrune, S^2-MAD, G-Designer, and MaAS all demonstrate that fully connected, fixed, all-to-all communication is unnecessary for many tasks. Debate-or-Vote further suggests that some debate rounds may be skipped entirely when majority voting already captures most of the gain. Therefore, Scheme C targets the upstream cause of discussion latency: redundant messages, redundant edges, and unnecessary rounds.

D.3 Fixed-point Scaling Shift

r is a hardware fixed-point scaling shift introduced by our INT8 RTL implementation. It is not a parameter in the continuous adaptive Kuramoto model; the model-level scaling is determined by μ, γ, and N. In our RTL experiments, we set $r = 8$ for the 64-agent design. The reason is that the $\sin(2\delta)$ LUT emits signed INT8 values whose magnitude is at most 127. Thus, the worst-case unscaled accumulation is bounded by

$$64 \times 127 = 8128.$$

Shifting by eight bits divides this sum by $2^8 = 256$, so the largest possible phase-update contribution is approximately

$$8128/2^8 \approx 32$$

INT8 phase codes before saturation. This keeps the update within the INT8 dynamic range, avoids overly aggressive phase changes, and implements the scale factor as a DSP-free arithmetic shift. Therefore, the factor $\mu/(2\gamma N)$ is absorbed into the fixed-point LUT/shift scaling of S_i.

E Limitations

System and hardware assumptions. FASD targets a CPU–GPU–FPGA heterogeneous serving stack. The reported end-to-end benefits depend on the availability of an FPGA with sufficient on-chip resources and memory bandwidth, as well as an integration that allows low-latency interaction with the GPU KV-cache path. These conditions may not hold in commodity deployments.

F Broader Impacts

Potential benefits. By accelerating and reducing redundant communication in multi-agent discussions, FASD can lower latency and token/compute cost for collaborative LLM systems. This can improve the practicality of MAS-based assistants in settings where response time and cost are critical (e.g., interactive decision support, software engineering workflows, and on-premise enterprise deployments). Reduced compute may also translate into lower energy usage per query under similar quality targets.

Potential risks and misuse. Making multi-agent systems cheaper and faster may increase their scale and frequency of deployment, which can amplify existing risks of LLMs (hallucinations, biased outputs, and harmful content) by enabling more automated decision loops. Faster generation can also increase the throughput of malicious uses (e.g., automated social engineering or content generation) Meng et al. [2025, 2026]. These risks are not unique to our approach, but efficiency improvements can make them easier to operationalize.

Deployment considerations. We recommend pairing efficiency-oriented pruning with standard safety controls (content filtering, policy enforcement, and audit logging) and evaluating safety under the same pruning/scheduling settings used for performance. Hardware acceleration may also affect accessibility and environmental footprint through device procurement and datacenter heterogeneity; these factors should be considered in deployment planning.

G LLM Usage Disclosure

Large Language Models (LLMs) were used to aid in the writing and polishing of the manuscript. Specifically, we used an LLM to assist in refining the language, improving readability, and ensuring clarity in various sections of the paper. The model helped with tasks such as sentence rephrasing, grammar checking, and enhancing the overall flow of the text.

It is important to note that the LLM was not involved in the ideation, research methodology, or experimental design. All research concepts, ideas, and analyses were developed and conducted by the authors. The contributions of the LLM were solely focused on improving the linguistic quality of the paper, with no involvement in the scientific content or data analysis.

The authors take full responsibility for the content of the manuscript, including any text generated or polished by the LLM. We have ensured that the LLM-generated text adheres to ethical guidelines and does not contribute to plagiarism or scientific misconducting.

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 contributions are claimed in Section 1.

   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: The limitations are introduced in Appendix E.

   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 explanation of the algorithms is in Section 3 and Section 4.

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: The experiment settings are introduced in Section 5, which could be used to reproduce the main experimental result of this paper.

   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: The code is in the supplemental material.

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: The experiment settings and details are introduced in Section 5.

   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: We consider error bars unnecessary for the reported results.

   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: This paper introduces all the experiment settings in Section 5.1.

   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: The code has been reviewed and conducted with the NeurIPS Code of Ethics.

   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: The broader impacts are introduced in Appendix F.

    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: [N/A]

    Justification: This paper does not have such risks.

    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: This paper uses the original RTL design introduced in the experiments.

    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: [Yes]

    Justification: All the new assets introduced in this paper are well documented in the main paper and appendix.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    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: This paper uses LLM for editing grammar, spelling, and word choice. The core method development in this research does not involve LLMs as any important, original, or non-standard components.

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.